Delft University of Technology

Automated monitoring of corrosion on piling sheets: a model test to understand the possibilities for asset managers

Graduation proposal

Student: Richie Maskam 4887883
Chair committee: Dr. Alireza Amiri Simkooei
Supervisor 1: Dr.ir. Sander van Nederveen
Supervisor 2: Dr. Mohammad Fotouhi
Company supervisor: ing. Maarten Visser MSEng

September 27, 2022

TU Delft Delft University of Technology

Witteveen + Bos

Preface

This is the thesis to finalize the master's curriculum for Construction Management and Engineering. Although a somewhat unorthodox topic for this field, it is a topic that I enjoyed learning and am proud of persevering through. This thesis covers the details regarding the subject, the methodology to execute the project and current progress on the classification and object detection algorithm. Prior to starting at TU Delft, I wanted to automate various processes in construction. The topic was inspired by a lecture from the course CIE 4634 3d surveying. Witteveen+Bos was willing to explore this topic and provided their data.

Readers who are familiar with Deep Learning and computer vision and want to compare their model, can have a look at chapter 4 to see how the algorithm was made. Chapter 5 can be checked for metrics and performance of the model. The final chapter will close with the conclusion and recommendations on this development project.

I would like to express my gratitude to various people involved in this project from Witteveen+Bos, the professors from TU Delft and friends.

Kind regards,

Richie Maskam
Almere Buiten, September 2022

Summary

There are various tasks in the construction industry that are time-consuming, such as visual inspections. When assessing the condition of the HLD, Witteveen+Bos had two employees manually going through the images to assess the condition which lasted a month. The dimension of the piling sheets was unknown so if they could be estimated from images, it is a bonus. Our development statement is: *Develop a tool using Computer Vision techniques to reliably detect problematic corrosion on piling sheet within 4-5 months to understand what the state is of this topic for asset managers.* We achieve this statement through the following development steps:

- *System analysis.* Conduct a literature study.
- *System requirements.* Set the metrics values that the model should achieve.
- *System synthesis.* Write algorithm and improve.
- *System evaluation.* Verify and validate the project.

System analysis
Deep learning has presented itself as the current state of the art method in many computer vision tasks. For images there are image classification, image segmentation and object detection. Image classification concerns with predicting a label on new images. Object detection concerns itself with localizing the object within the image. Life-cycle management is another topic of interest as it is focused on maximizing the return of the investment through information on the asset its condition. Visual inspection is the oldest and well understood method. Witteveen+Bos is following the guidelines for Digigids 2019, when assessing the piling sheets in this project.

System requirement
Requirement are formed after the literature study. They are also gathered by asking what Witteveen+Bos wants. For this project they want to assess the grade according to Digids and get dimensions from the sheet. Now we know how much classes the model needs. The object detector will be used to calculate the height and the distance between bumps. With the distance between bumps the user can eventually figure out the specific piling sheet that was used.

W+B did not specify any metrics they wanted the models to achieve so we searched online some common models:

- Classification algorithm are around 90% with data-sets containing 3000 images.
- Object detection are around 90% with data-sets of 1000 images.

System synthesis
We write the algorithms for image classification and object detection in the synthesis stage. For classification we created a four and six class data-set. In our CNN we changed the structure, filter-size and use of augmentation to find the best performing model. In object detection we used YOLOv4 and annotated the data using labelImg. Using the output of the object detector, we could estimate the height of the piling sheet above water and the distance between bumps. We created a model test set of 100 images to simulate the models going over a W+B folder and store the results in a data-frame.

System evaluation
The best performing models for the four class and six class were retrained using a form three-fold cross validation. This was done by shuffling the train and validation set three times with a different seed every iteration. We evaluate using confidence matrices. In this way we can easily compare the predicted labels with the actual labels for the classification models. The object detector we used metrics gained from Darknet. This showed that the mAP is at 79% and the IoU at 67%.

	Four class	**Six class**
Material accuracy	99.0%	88%
Rust grade accuracy	96%	73%

Table 1: Results of image classification

We compared both of our models with other similar research. One research had corrosion on pipes with clear differences in classes, 102k images and an accuracy of 98.2%. Another research on weathering sheet had 4 classes, 1864 images and 90.84% accuracy. For object detection with YOLOv4 there were had 90% mAP with a data-set of 1000 images.

A superficial time-cost analysis was done. The algorithm was able to compute the results in 77s-107s for 100 images. We can extrapolate that for 40k image to 8.6-11.9 hours, which is already a huge difference compared to the 21 days done by two employees. The cost would also be lowers since the total training time is sitting around 9h for the classification and object detection models. We used Google Colab, which was free with several limitations. Another option is to utilize cloud computing platform like Microsoft Azure with price ranging between 0,67-5,70 euro per hour. Buying a physical GPU is also an option, but these are sitting around 2600 euro from amazon.

A review with employee's form w+b was also conducted. We demonstrated the algorithm and gathered their thoughts surrounding the applicability and feasibility of the algorithm. The applicability focused on how well the algorithm is working. They all agreed that the classification part is able to do what they expect, however the distance estimation should be handled with more caution. One person also added that it is not known how the algorithm can provide information to his GIS software and another mentioned that dimension estimation was not yet needed.

Conclusion
Project statement: Develop a tool using Computer Vision techniques to reliably detect problematic corrosion on piling sheet within 4-5 months to understand what the state is of this topic for asset managers.s. For classification we made 2 classifiers with 4 and 6 classes and can classify according to DIGIGIDS. The classes should carefully be considered against data availability and the difference between classes. The structure and amount of trainable parameters play a role in the accuracy For object detection we used YOLOv4 and created a data-set and model using two classes. This still took a significant amount of training time (8 hours). Estimating the size is a rough estimate, due to IoU=67%. The mAP is at 79%. We can get geometric features or the amount of an object. Available data and well-constructed training set is of great importance.

Recommendations
For classification the recommendations relate to increasing the data-set and types of classes. The dataset can be increased with images from a different seasonal period, different angles. The types can be increased with wood or concrete as these have been observed in the raw data-set.

Object detection could be improved through detecting other objects and using a recent released YOLOv7. Detection of other objects can be used to calculate subsidence or detect specific corrosion faults (pitting, waterline corrosion) to give a better view on damage. YOLOv7 has been released recently and promises better speed and accuracy.

For managers it might be interesting to detect objects to measure progress or research the interoperability between other software. Object detection could be retrained to detect vehicles, material or other construction elements. Knowing the amount of a certain object may help with the logistics. The interoperability can be explored as for now it was seen that ai can use data from other software but not the other way around.

Contents

List of Figures

List of Tables

1 Introduction

This section introduces the topic of the thesis. Background information of the contractor and its problems are given. A development statement is presented with development steps on how to improve their operations. The chapter ends with the relevance of this project and the possible benefits for various actors such as the contractor and academics.

1.1 Background information

Monitoring a project is essential for a construction project, as managers would like to assess the progress against the planned objectives (Nicholas & Steyn, 2017). Monitoring still has some problems regardless of how thorough and conscientious a manager is (Nicholas & Steyn, 2017). Nicholas and Steyn (2017) have mentioned several problems with monitoring. They state that saving costs (since cost is heavily focused on) can lead to low performance in execution. Besides having costs, they have also mentioned that people could abstain from reporting. For example, they might not understand the situation or are just hesitant. Moreover, we can also have personal bias towards certain situations and therefore automated algorithms have been introduced. The construction industry is also starting to automate various processes such as safety monitoring, quality inspection, progress monitoring, navigation assistance and automated construction (Paneru & Jeelani, 2021). If humans need eyes to monitor an asset, then a computer would need a Computer Vision (CV) algorithm. Applying an automated system, written in Python, would be able to assist those with monitoring duties.

1.1.1 Witteveen+Bos

Witteveen+Bos (W+B) is an engineering firm based in The Netherlands and has offices in Belgium, Indonesia and Kazakhstan. They are providing services related to water, infrastructure, environment and construction sectors and are among the top six largest engineering firms. Their clients consist of government, private sectors and other organizations (witteveenbos.nl, 2022). The engineering firm is known for using automated system, as they have a department that goes over Building information Modeling (BIM). BIM is one of the known trends in construction alongside CV. In the field of CV they might use classification and object detection in predicting the type of connections on a steel bridge or predicting a type, faults and profile of a piling sheets. W+B is now tasked with assessing the condition of the Hoofdvaarweg Lemmer-Delfzijl (HLD).

Rijkswaterstaat has started the preparations for the renovation of the HLD, the plans to make preparations have been awarded to the combination of Witteveen+Bos and Royal HaskoningDHV. The HLD consists of three separate channels which are Prinses Margrietkanaal (PMK), Van Starkenborghkanaal (VSK) and the Eemskanaal (EMK). This fairway is 118 km long, has three aqua-ducts, 32 bridges and five sluices. The fairway does not meet the conditions to classify as a "VA-vaarweg" and most piling sheets and banks have reached the end of the technical life (Rijkswaterstaat.nl, 2022). The "VA-vaarweg" classification will be achieved by deepening and widening the current fairway. W+B wants to assess the condition of the HLD according to NEN 2767-1+C1:2019, which is aimed at providing an unambiguous methodology to assess the condition of all assets identified in the built environment (nen.nl, 2022). Data was gathered by a boat that ferried through the fairway. The boat took pictures and created a point-cloud using Sonar and a multi-beam device. The pictures and the point-cloud were used to assess the HLD. We point out that the point-cloud has some problems as the edges show noise due to the boat creating waves.

1.1.2 Problem description

W+B is interested in novel automated detection techniques using deep learning methods and thus they show interest in the possibilities it can provide. Paneru and Jeelani (2021) have identified the availability of good data to be an issue, but W+B has a large, raw data set of piling sheets in its servers. The pictures are being analyzed by their employees and are part of a Geographic Information System (GIS) database. They are analyzed on the type of metal sheet, the water height and how severe the corrosion is. In other words, W+B is interested on how novel techniques would compare to their existing methods.The question thus becomes: "How can Computer Vision (CV) based deep learning offer benefits in assessing condition of piling sheets?". A good way to compare this is measuring the time it would take for one person to complete the task of analysing the pictures.

Figure 1: The HLD route (Source: Rijkswaterstaat.nl)

In the existing method there are two experts analysing each image on the condition of the piling sheet. This is a time consuming task, as there are 40.000 images that need to be careful assessed and described for any peculiarities. The expert has currently no way of retrieving information of the profile of the metal sheet present. It could also happen that one expert might have a different assessment on an image. This current method has taken one month to complete and has been described as doing "Monnikken werk" (Dutch way of saying to have the patience of a monk).

Going through Paneru an Jeelani review of 2021 we see that the object of interest differs for each solution and many more objects could be trained for detection. Currently, we have not found an application of piling sheet detection. A Computer Vision (CV) algorithm could be tasked with classifying the assessment per image and there are ways to detect geometrical features. Detecting these geometrical features could help estimate the profile of the piling sheet. Since this would be an automated task, it could take some work load off from the four experts.

1.2 Knowledge gap

Paneru and Jeelani (2021) have mentioned that in construction many of the objects are unique. This is also the case for piling sheet, when it comes to an image-based analysis of the condition. We then search the keywords "Piling sheet", "Piling sheet monitoring" and "Piling sheet detection" in both Google Scholar and Science direct. We have found articles that are mostly focused on the geo-technical side such as a research paper that collects various parameters on the soil to get estimates. These estimates are then used in a failure probability analysis (Chai, Árpád Rózsás, Slobbe, & Teixeira, 2022). Besides these search engines, we continue our search of the word piling sheet on Github and Kaggle to see if anyone had a database of images containing piling sheet. This gave us little to no results. It is thus safe to say that implementing an image classification and object detection task is still an uncharted territory. Thus, we can fill this gap by creating a data-set and algorithms to assess the condition of piling sheet and get certain geometrical features.

1.3 Development statement

We face this challenge by adding an automatic system to detect the piling sheet and the extent of corrosion. This system can assist with managing the asset life-cycle by giving an estimation on the extent of corrosion. The development statement becomes:

Develop a tool using Computer Vision techniques to reliably detect problematic corrosion on piling sheet within 4-5 months to understand what the state is of this topic for asset managers.

Some steps to achieve this statement are:

- *System analysis.* We need to understand what this research would improve for W+B. We will conduct a literature research and explore the available methods to automate this task. When we know the available task it is important to know what W+B is willing to change the existing system. A quick glance at the

raw data-set should be taken to determine what automation method(s) would fit.

- *System requirements.* After the analysis we can determine to what the algorithm is supposed to do exactly and to what metrics the algorithms should adhere to. In a classification task we need to know what the classes should be. A frequent used metric for classification task is the accuracy and according to Paneru and Jeelani (2021), current algorithms possess around a 90% success rate. Given our time of 4-5 months we can strive to reach a success rate of 90%. In an object detection task, we need to specify what the object of interest will be. We also need to decide on the metrics that will explain. The assessment of the piling sheet images took one month with two people working on it. They have expressed preferring to add improvements without altering the physical object, so that means we need to develop an algorithm that will analyze those pictures.

- *System synthesis.* In this phase we start writing all the algorithms and keep improving until it satisfies the system requirements. A detailed description of how the data-sets are made, the architecture used for classification and the object detector used to make calculations. All the required pieces come together to form one and the algorithm is tested on a model test data-set that would look similar to the folders in W+B server. The result should be described, discussed and improved if possible.

- *System evaluation.* We verify and validate the system. We can verify a classification task using a confusion matrix that would help us get other metrics like accuracy. The object detection task also has a similar method. Will it be something that improves their monitoring regimen on piling sheet? Or when would such a technique become acceptable for W+B?

1.4 Relevance

We believe that there are several benefits in developing such a system. First, assessing is done automatically, which can assist those with monitoring duties. AI systems are not designed to replace humans, but they can increase our efficiency at doing repetitive task. The financial impacts are one of the barriers for such a monitoring system (Alaloul, Qureshi, Musarat, & Saad, 2021). Thus, we need to look which equipment this system could present itself as an alternative. This system can be repurposed, the training material can be changed to whatever object is desired. For instance, the object of interest can be changed to concrete cracks, pipe cracks, safety helmet detection or any building materials to monitor site progress. Also, the system is a cheaper alternative to several equipment due to its flexibility. When we apply detection using laser scanner there is a clear financial difference among products. Laser equipment can vary between 700-20000 euros (surveyinghub.nl, 2022), in case we would buy vehicle counters the price ranges from 300-500 euros (diamondtraffic.com, 2022). Even if companies possess the most expensive equipment it is still useful to develop this system, as it can integrated onto other equipment or technology or as a temporary replacement when the expensive equipment is unavailable.

Image classification and object detection has been used in the construction sector, but there is more to discover when deploying an algorithm. Paneru and Jeelani (2021) have summarized various gaps in this field such as collecting good data, a detailed Building information Modeling (BIM) model, integration with BIM software and several more gaps. They have also mentioned that even though there are public data sets available like ImageNet and Microsoft Common Object in Context, the data-sets required in construction can have unique characteristics (poses, scale and environment). The availability of a well-constructed data-set of piling sheet for classification and object detection will allow others to test and enhance this specific object. The new acquired data-set directly contributes to the gap "good data".

As mentioned before, Paneru and Jeelani (2021) have identified four main uses of object detection in construction which are: safety monitoring, progress monitoring, productivity tracking and quality control. This research could fall in the quality control category, which is the most lagging of the four main uses. Using computer vision methods, we are able to obtain the geometric properties of the piling sheet. This in turn could give us insight on the usability of these geometric properties for other Building information Modeling (BIM) software that can be explored during the qualitative evaluation. During this evaluation, we discuss on how useful this algorithm could be and how it could -indirectly- contribute insights to the integration with BIM.

One of the benefits of such algorithms is its re-usability. The training images can be changed with some

minor tweaks in the algorithm. Instead of detection piling sheet, the training folder could be switched with hardhats, beams, concrete buckets or other faults. For the four major fields it could be:

- Safety monitoring
  Detecting safety equipment (hardhats and safety vests) and estimating their dimension may not be useful, but the amount per day could still be helpful in BIM or any logistics software to keep track of how many workers were present and the amount of available safety equipment.

- Progress monitoring
  Detecting the amount of columns, beams or objects of interest. Eventually the system could be improved by utilizing the data in a BIM software. Data such as amount of columns or concrete materials can be updated in such a software to show the progress.

- Productivity tracking
  Detecting how many concrete buckets have been lowered or delivery trucks that have entered a site. It can also be the amount of pipes or other building elements on the site. It could be further improved to add the data such as amount of elements to a 4d construction model (3D model + time), so it can assist on what has been achieved.

- Fault detection
  Searching for faults as concrete cracks, pipe cracks, corrosion. Eventually, it could be improved by exporting the data such as distance between reference object and object of interest, and size of the fault to a 3d model.

One last reason this could be relevant but is not the main reason for developing this system is the impact of COVID-19 on contractors. Contractors are faced with delay of their projects, the decreased availability of materials, while labour cost had to remain the same (Elnagga & Elhegazy, 2021). The construction sector had difficulties such as project delays during the COVID pandemic. For instance, in the Netherlands the capacity for available workers has decreased, which could mean delayed projects for the company due to strict government measures (www.bouwendnederland.nl, 2020). To alleviate this problem, they can resort to remote sensing techniques, but such techniques are expensive giving discrimination to companies with lesser technologies.

2 Literature

This chapter discusses the relevant topics needed for this development project. A description of monitoring assets is given with regards to Dutch norms. Some general theory is presented of piling sheets and its corrosion regime. Afterwards, a lengthy introduction of Deep Learning and computer vision is given. We will also discuss various popular Convolution Neural Network (CNN) and object detection algorithms.

2.1 Monitoring of object of interests

2.1.1 Life cycle asset management

Life cycle asset management is focused on maximizing the return of an investment in an asset by providing information throughout its life (Roberge, 2007a). Asset management tries to effectively manage the asset during the planning, acquisition and disposable stage. One major component in life-cycle asset management is the total cost of the asset over the service life-cycle. A second major component is the condition assessment. The objective in condition assessment is to provide comprehensive information on the condition of the asset.

A life cycle asset management plan is formed when we identify the operation sequence of an asset, the maintenance and repair needed, the planned life and finally the disposal of such an asset (Hastings, 2015). This is needed in order to list the resources and budget for the asset. In such a plan there is a maintenance routine that may need adjustments, if there are sudden changes in operation activity or the environment.

In the Netherlands there are Nederlandse Norm (NEN) and for assessing the condition of an asset we have NEN 2767. NEN 2767 tries to decompose the asset in smaller sections and unambiguously give a technical assessment on the condition of that part. Figure 2 shows how a part in the asset is scored. At Witteveen+Bos (W+B) the condition is scored according to the guides set by Digigids 2019 (Digigids.hetwatershapshuis.nl, 2022). This guide has four scores which are: good, acceptable, moderate and bad.

FIGUUR 9 NEN 2767-CONDITIESCORE: RELATIE ERNST, OMVANG EN INTENSITEIT

Conditiescore NEN 2767-1:2017 (en verder)

GEBREK	INTENSITEIT	OMVANG < 2% incidenteel	2 - 10% plaatselijk	10 - 30% regelmatig	30 - 70% aanzienlijk	> 70% algemeen
gering	begin	1	1	1	1	2
	gevorderd	1	1	1	2	3
	eind	1	1	2	3	4
serieus	begin	1	1	1	2	3
	gevorderd	1	1	2	3	4
	eind	1	2	3	4	5
ernstig	begin	1	1	2	3	4
	gevorderd	1	2	3	4	5
	eind	2	3	4	5	6

Figure 2: NEN 2767 (source:W+B)

2.1.2 Corrosion

Corrosion is defined as "the chemical or electro-chemical reaction between a material, usually metal, and its environment that produces a deterioration of the material and its properties" according to the American Society of for Testing and Materials (Cicek, 2014). There are several factors that influence the corrosion process and they can be categorized in the nature of the of the metal (position in galvanic series, purity of metal etc.) and the nature of the environment (temperature, humidity etc.).

Corrosion is also categorized in: uniform corrosion and localized corrosion. Uniform corrosion is the most

common type and is corrosion that is over the whole body of the object. Localized corrosion are further divided in macroscopic and microscopic level:

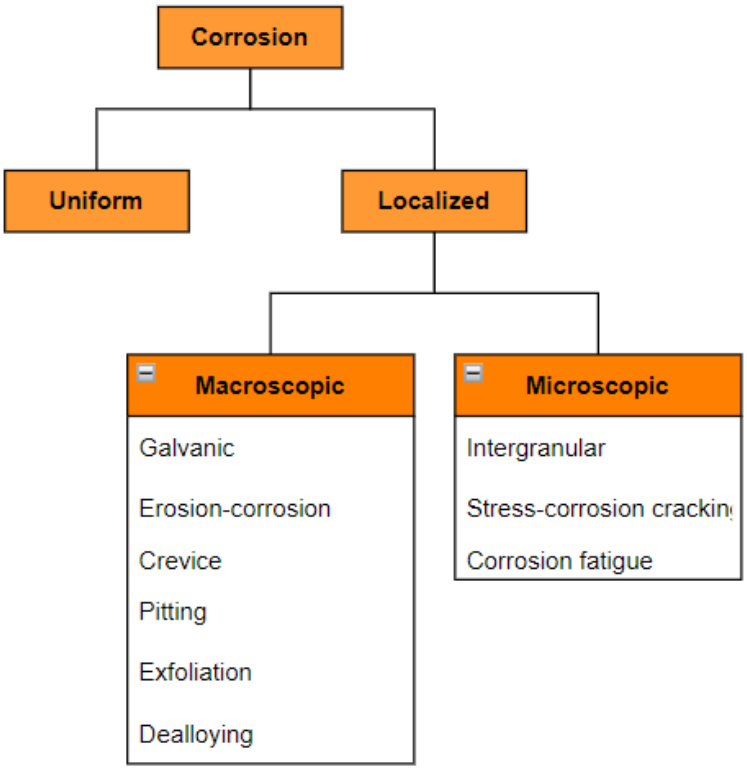


Figure 3: The types of corrosion

Almost every monitoring process that takes place virtually contains three steps: setting a standard, comparing the performance against that standard and taking corrective measures in case of under-performance (Nicholas & Steyn, 2017). Corrosion monitoring refers to the measurement undertaken to assess the corrosion.

There are several existing methods to detect corrosion and are categorized in direct methods and indirect methods (Reddy et al., 2021). Direct methods measure the parameters influencing the corrosion and are further divided in direct physical and direct electro-chemical. A few direct physical methods are measuring the mass or electrical resistance of a sample. A popular direct electro-chemical method is "Linear Polarization Resistance", which measures the corrosion rate through a potentiostatic polarization. Indirect methods give information about the parameters that affect the corrosion environment such as hydrogen monitoring, a technique to identify the amount of hydrogen permeating through the metal and give an indication of the corrosion rate. Several techniques have not been mentioned, as the focus will be on computer vision and deep learning.

A nondestructive evaluation (NDE) method has some benefits, as it means the structure operation does not have to be stopped (Roberge, 2007b). It is also very quick and easily interpretable. Visual inspection is a common NDE technique to inspect corrosion. The reliability of such a visual inspection depends on the experience of the inspector. The inspector should be able to discern critical defects and recognize when failure may occur. A problem with visual inspection is that they can be labor intensive and monotonous.

Corrosion will cause various deviation in the geometry and appearance, and it is convenient to classify corrosion after the appearance of a corroded material ("Different Forms of Corrosion Classified on the Basis of Appearence", 2004). This makes it possible to identify the corrosion form by visual inspection. This usually precedes a much in-depth method of corrosion tests. For this research a vision-based monitoring is desired, as W+B already possess the data. Also, a working monitoring system will assist them if a instrumental service is

needed to check the corrosion. Our subject concerns piling sheets which are easily accessible, unlike industrial pipes that will need operations to stop. This makes visual inspection of corrosion rather practical as the assessment according to Digigids, already requires someone to inspect the asset.

Corrosion detection using deep learning methods is still a novel idea, as researchers have just developed frameworks like CorrDetector(Forkan et al., 2022). The CorrDetector uses a combination of deep learning approaches for structural identification and corrosion detection. Another research consists CNN in combination with cycle generative adversarial network (CycleGAN) (Munawar et al., 2022). CycleGAN is a style-changing image generation network. However, as Paneru and Jeelani (2021) has stated before, the gaps are similar and estimating the size of the corrosion can be a good way to to measure the extent of corrosion or integrate with other BIM software.

2.1.3 Objects of interest

Pile sheets are a type of construction designed to retain soil by installing a vertical wall into the ground (Verruijt, 2018). They are made of either wood or steel or both. In the past wooden logs were used, but it was found out that they work better when interlocked with each other. Wooden pile sheets are a cheap alternative and are used for retaining soil up to a specific height. Steel pile sheets can endure more tensile stress and are used in more complex construction projects. In figure 4 we see a sketch of a cross section for a piling sheet with the important dimensions in a red font colour. The parameter "distance bumps" can be helpful to estimate the profile of the piling sheet, especially since we are looking at the piling sheet from the front.

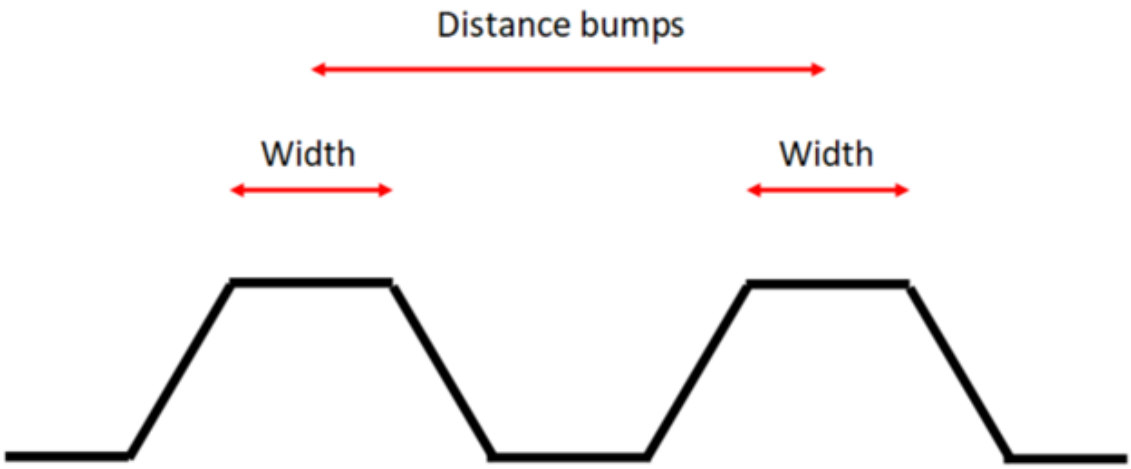


Figure 4: Sketch of cross section of a piling sheet and relevant parameters

There are various types of piling sheet manufacturers who make slight variations on their products. The profile of the piling sheet can also differ in shape such as Z-shape and U-shape. In table 2 we see a few piling sheets from ArcelorMittal (Baxter et al., 2016). In the Z-profile we see that a product would be named "AZ XX-800" in which the "XX" would show the elasticity modulus and the "800" refers to the half width of a sheet. There are more types and there are variations in the height and thickness of the sheet.

Name	Type	Width (mm)
AZ XX-800	Z-profile	1600
AZ XX-770	Z-profile	1540
AZ XX-750	Z-profile	1500
AZ XX-700	Z-profile	1400
AU	U-profile	1500
PU	U-profile	1200

Table 2: Various different piling sheet dimensions of ArcelorMittal

The inspection of piling sheet ranges from a global visual inspection to detailed measurements on thickness, strength, root of damages, tilting and displacement (Infra, 2016).

2.2 Deep learning and computer vision

Deep Learning is a subset of Machine Learning (ML), which is also a subset Artificial Intelligence (AI). AI is involved with finding ways in which machine can act like humans in terms of learning and problem solving (Ongsulee, Chotchaung, Bamrungsi, & Rodcheewit, 2018). Deep learning tries to make a model in which the data is going through successive layers in a network, thus the "deep" in deep learning refers to the idea of having multiple layers in a network. The learning can be done in the following ways:

- *Supervised*
  The data are labeled.
- *Semi-supervised*
  Combination of small labeled data and large unlabeled data.
- *Unsupervised*
  The data-set consists of unlabeled training data

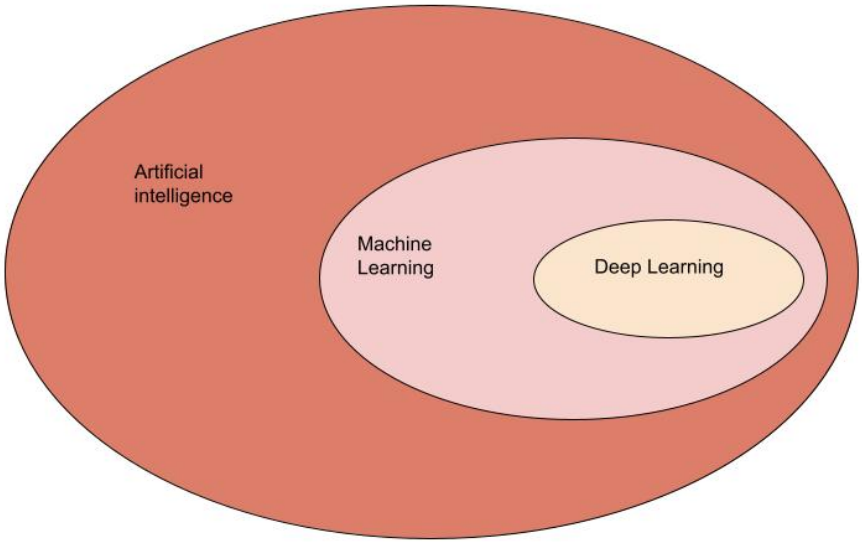


Figure 5: The field of AI

Layers Layers are part of the neural network which contains the nodes. In global terms there are the input, hidden and output layers. In a typical neural network, the hidden layers can be dense layers in which all the neurons are connected to the previous layers. In a CNN there are convolution and maxpooling layers. The convolution layer have a filter (usually 3x3 pixels) that go over the image and makes convolution operations. The maxpooling layer has a filter that goes over the image and picks the max value.

Neurons Neurons are parts of the layers. In the input layer they contain the data of the material trained. In the hidden and output layers they are activated using the activation functions.

Weights and bias Weights are values that are used to multiply the input data. Bias are the values that are added up after multiplication with the weights. Higher weights will affect the influence a small change in input will have, while higher bias will make up the difference between the input and output.

Activation function The activation function will decide whether the neuron will be activated or not based on the input [cite]. There are several commonly used activation functions used for computer vision techniques which are:

1. *Sigmoid*. The sigmoid function is a popular activation function and transform the input as:

$$f(x) = \frac{1}{1 + e^{-(x)}} \tag{1}$$

2. *Softmax.* The softmax is a type of sigmoid function and is useful in multi-classification tasks.The equation is given as:

$$f(x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \; for \; j = 1, 2.., K. \tag{2}$$

In Computer Vision (CV) we describe the world through our eyes as one or more images and reconstruct its properties such as shape, illumination, and colour distributions (Szeliski, 2011). CV covers various topics such as image processing, feature detection, 3D reconstruction, recognition and many. CV is a broad field and is already being used in photography, visual effects, medical imaging and safety applications. There are various computer vision techniques used for image-based analysis. In the past techniques like Histogram of Oriented Gradients (HOG), Support-vector Machine (SVM), decision trees, random forest and cascade classifiers. CV is the domain that led to the rise of deep learning between 2011 and 2015, when a deep learning model called Convolution Neural Network (CNN) produced good results (Chollet, 2021).

After 2017, it became common that an Artificial Neural Network (ANN) is used for image-based analysis (Mostafa & Hegazy, 2021). The proposed CV technique will utilize a Deep Learning (DL) algorithm to detect. An image is a 3D matrix and by feeding several of these matrices to an algorithm, it can eventually recognize the matrix. This system uses a camera and needs training material on the object of interest. By using a reference object it should be possible to estimate the dimensions of the object of interest. This system should be able to keep count of the object of interest. Currently object detection is used in the construction for safety monitoring, progress monitoring, productivity tracking and quality control (Paneru & Jeelani, 2021).

Deep learning in CV can be divided into three branches which are: image classification, image segmentation and object detection. Regarding image classification, the model assigns one or more labels to an image. Whereas, in image segmentation the model partitions the image into different areas in which each area represents a category. In object detection we want to draw bounding boxes around an object of interest. Deciding on which to use is dependent on our task, it could also mean that we could utilize a combination of the branches.

Deep learning based CV techniques are predominantly used on 2D image and very few on 3D point-clouds (Reja, Varghese, & Ha, 2022). Point-clouds possess other challenges such as the data being unstructured and large, making detection computing intensive. The data is generally also noisy and susceptible to changes in the environment. The learning models have problems performing well when the differences between training and test data are apparent. Lastly, supervised learning requires a large amount of training and testing data sets which is limited. A 2D image however is given as a 3D matrix array with the x and y axis being the dimensions of the picture and the colour channels being the z- axis. In python we can use the CV2 library to edit the 3D matrix array of an image.

2.2.1 Image classification

Image classification is the task of classifying an input image with a class that it belongs to. Figure 6 shows an example where the model is fed images of various heavy equipment vehicles and what the classifier would give as output. There are various different deep learning architectures, but the most used architecture in computer vision is the CNN (Khallaf & Khallaf, 2021). A CNN is widely used for processing application on images (Schmidhuberaf, 2015). Every deep learning architecture consists of layers, neurons, activation functions and weights.

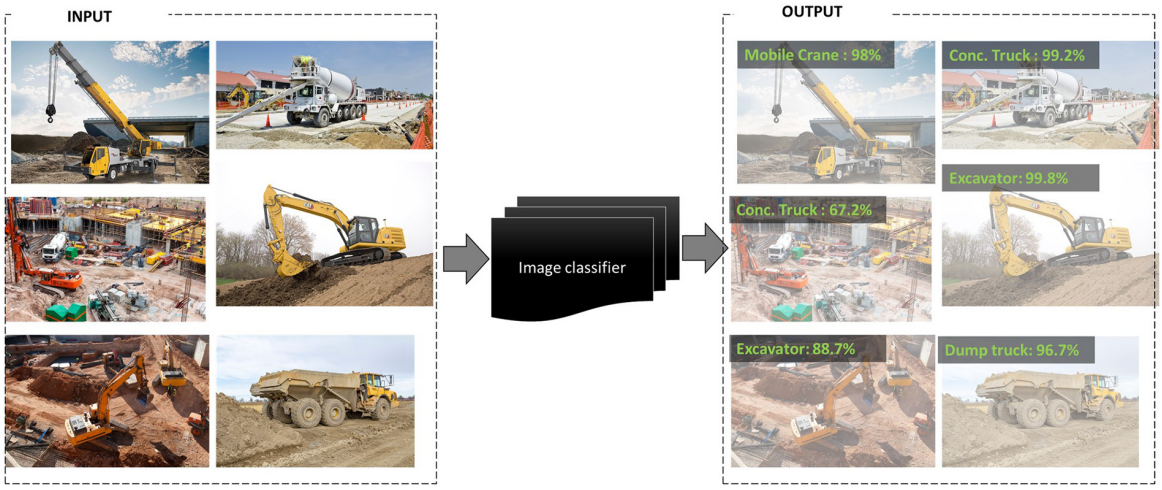


Figure 6: Example of how a classifier would label images (Source: Paneru and Jeelani, 2021)

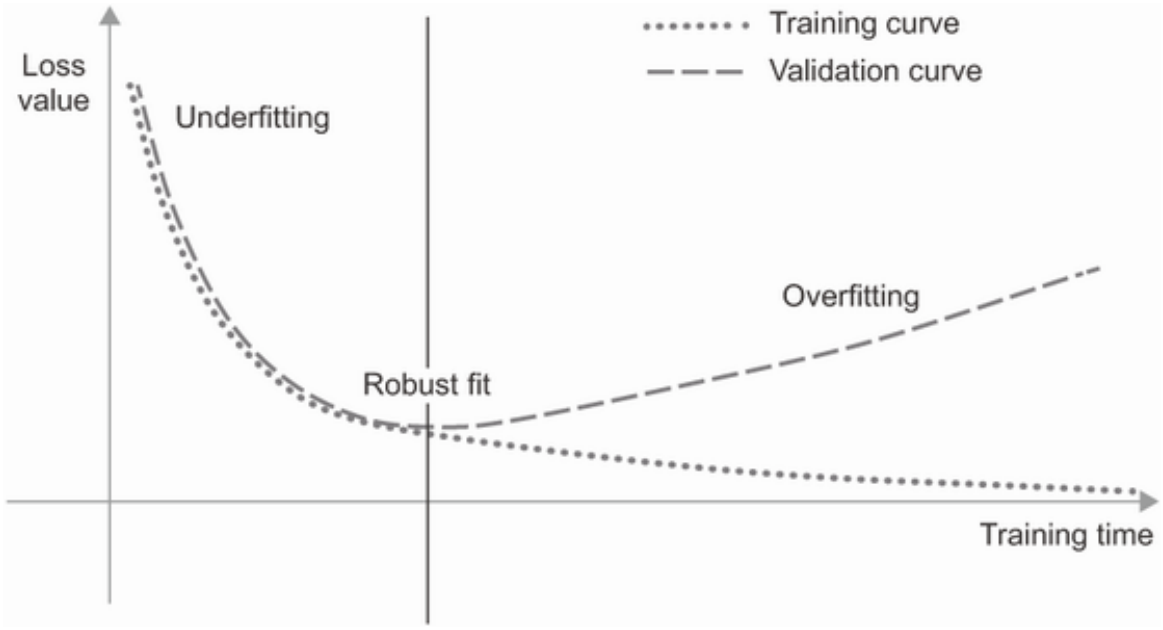


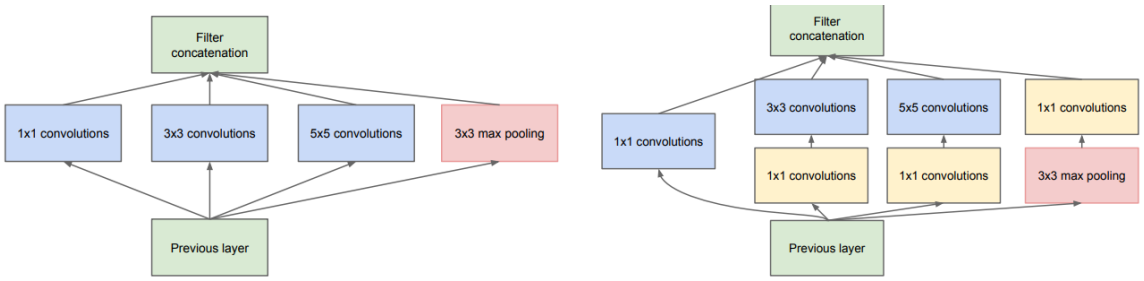
Figure 7: Characteristic of overfitting and underfitting (source: F. Chollet, 2021)

To evaluate a model we need to consider the loss value over the amount of epochs during training. Figure 7 shows some characteristics when over-fitting or under-fitting of a model occurs. In case of under-fitting, then we should increase the training time so the model can improve. Over-fitting is a common problem in a classification task, but there are several techniques to remedy this. A common practice is data augmentation in which we make small changes to an image. These changes can be a slight rotation, zoom, flipping and colour changes. Another method is to use residual blocks, batch normalization and dropout. We can also make the architecture have some features of an inception module. Finally, we can fine-tune the hyper-parameters of the architecture. For example the filters and batch size can be changed.

VGG16 The VGG network was used in Alexnet and was the start of the boom of CNN in image classification. The vgg16 had the structure of using convolution layers and maxpool layers. The first parts of vgg contained one convolution layer and one maxpool layer. After a while they would be followed by a block containing 2 convolution layers and one maxpool layer. There were also versions that contained three convolution layers and one maxpool.

GoogLenet GoogLenet, also called Inception, is a CNN architecture with the hallmark of improved utilization of computing resources (Szegedy et al., 2014). It was used in the "ImageNet Large-Scale Visual Recognition

Challenge 2014" and had the second-best result. GoogLenet is already being used by other researchers in object detection research on concrete cracks (Miao & Srimahachota, 2021). The inception model has some characteristics in its architecture in that there are convolution layers walking parallel to others (see figure 8)



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 8: Inception module (source: C. Szegedy et al, 2014)

Resnet Resnet is a CNN created by Microsoft researchers which eases the training of networks (He, Zhang, Ren, & Sun, 2015). It was used "ImageNet Large-Scale Visual Recognition Challenge 2015" in which they ended on the first place. Resnet was also used by Miao and Srimahachota (2021) for crack detection and compared with other popular algorithms. The architecture is known for using residual blocks, in which the input value is added to the output of several convolution operations (see figure 9).

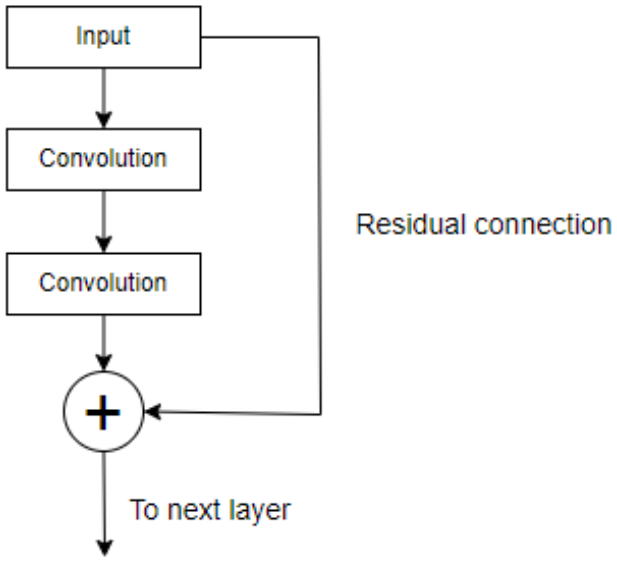


Figure 9: A sketch of residual connections

2.2.2 Object detection

YOLO Redmon et al. (2016) created a detector called You Only Look Once (YOLO) and was aimed to create a bounding box around the object and assign a class to them. YOLO needs to be trained and its framework is called DarkNet, which is an open-source neural network written in C and CUDA (Redmon, 2013–2016). Joseph Redmon continued improving this model, but stopped developing after YOLOV3 due to ethical concerns (twitter.com, 2020).

YOLOv4 is state-of-art detector which is fast and accurate compared to alternative detectors and is made by Alexey Bochkovski,Chien-Yao Wang and Hong-Yuan Mark Liao (Bochkovskiy, Wang, & Liao, 2020). The algorithm gives as output the following: a vector containing the x-coordinate of the center, the y-coordinate of the center, length, width, probable class and score of all the classes. The coordinates are presented in pixels.

However, if we know the actual dimensions of the reference object, we should be able to give an estimate on the actual size. YOLOv4 is already being used to detect bridge cracks (Yu, Shen, & Shen, 2021) and detecting apple flowers (Wu, Lv, Jiang, & Song, 2020).

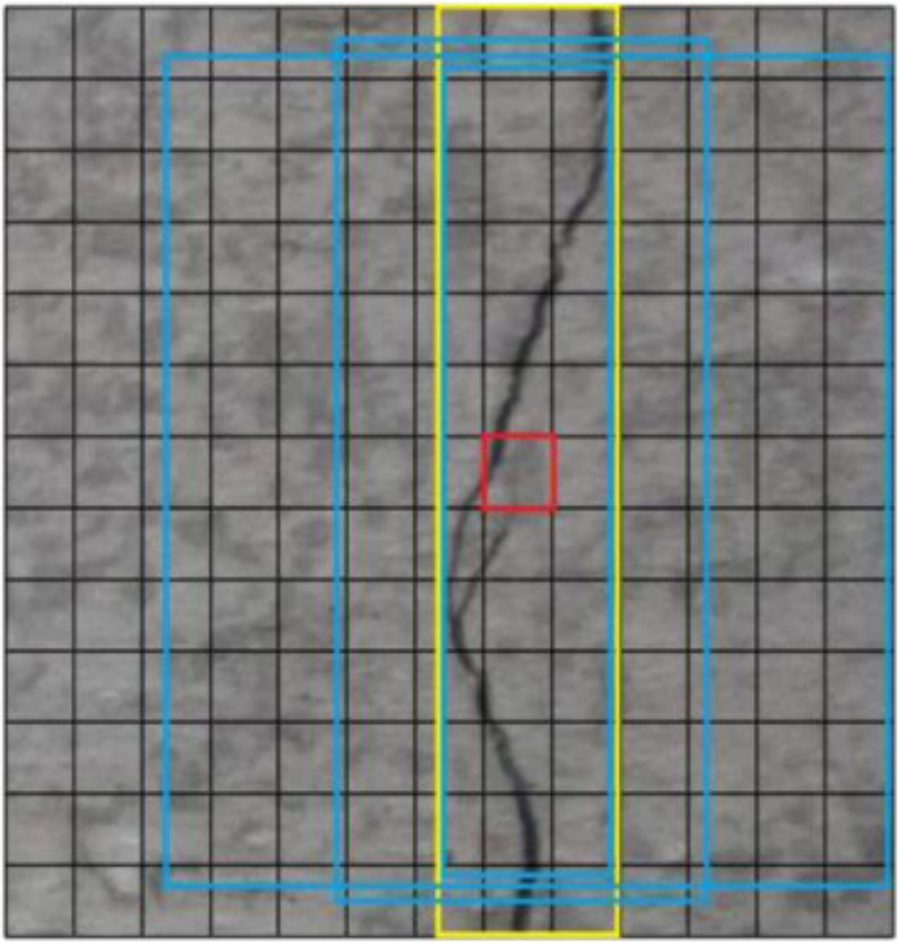


Figure 10: Crack detection (source: Yu et al,2021).

Figure 11: Apple flower detection (source: Wu et al,2020).

2.3 Summary literature

This chapter briefly went over monitoring, Computer Vision (CV), deep learning, detectors and how all these topics are being implemented in the construction sector. For this project we should create an algorithm that will resemble an assessment similar to NEN 2776, thus the algorithm should be able to classify an image on the severity of the corrosion and detect dimensional features to get the height above water and profile of the piling

sheet. First, some knowledge of CV is needed to do basic operations on images or videos. Secondly, the raw data should be labelled and annotated accordingly. Lastly, we need to be able to use classifier and a detector and store the data in a data-frame. The classifier will be used to label images in categories such as rock, grass and the degree of corrosion. The degree of corrosion could be "metal good " and "metal bad", this should be verified with whomever would score the images based on Digigids 2019. In object detection we should try to detect features that would help us in determining the type of the piling sheet profile. A reference object should also be considered to couple the pixel dimensions to actual dimensions. We should annotate images using an annotation software in which we put bounding boxes over dimensional features and the object of reference. The information on the bounding boxes from YOLOv4 should be accessed, stored and used to estimate the actual dimension. In figure 13 the Unified Modelling Language (UML) diagram of the object detection system is made of is shown. The UML diagram for the classifier would look similar, but would need separate training.

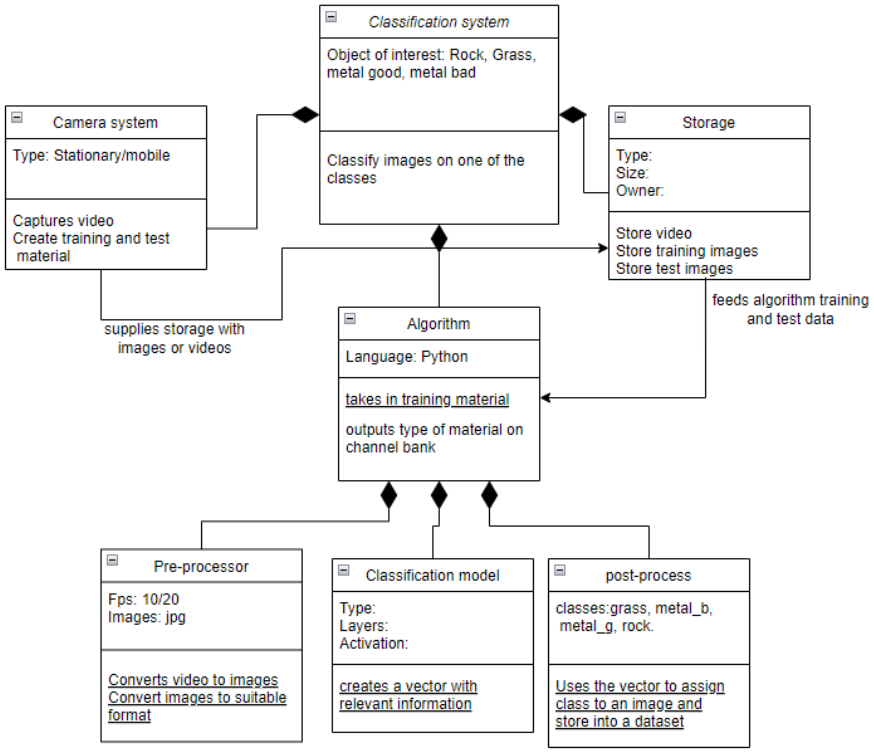


Figure 12: UML diagram of the classification system

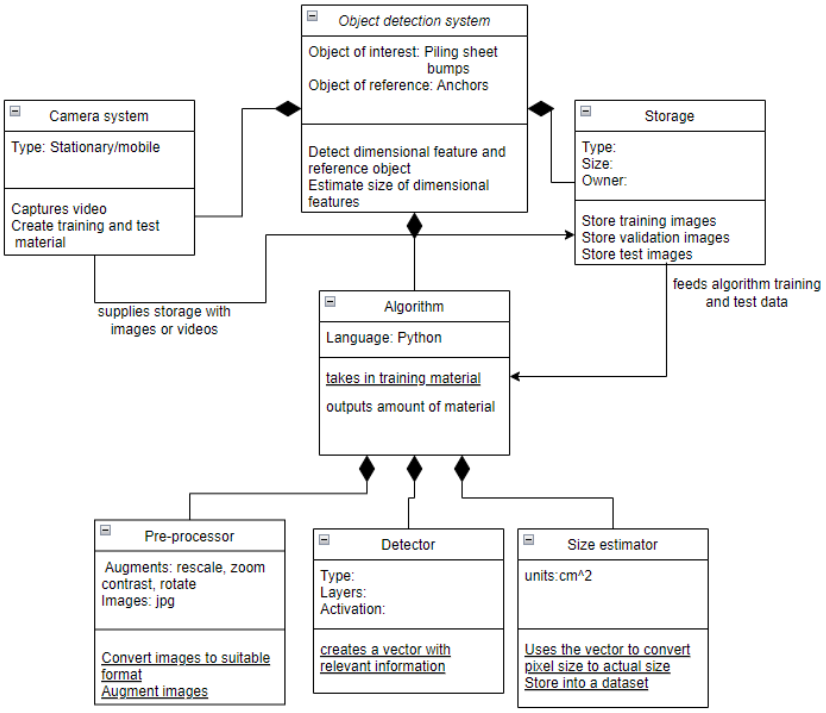


Figure 13: UML diagram of the object detection system

3 Methodology

This chapter will present the method used to conduct this development project. A description is given on how the algorithm will be conceived. A method on how we will evaluate the algorithm is also given. Given the development steps, this section covers the **system analysis** and **system requirement** step discussed in chapter 1. The codes for all the algorithms can be found on the authors Github page.

3.1 Project strategy

Our project is about developing alternative method to monitor piling sheet by assessing the corrosion class and estimating the profile of the piling sheet for Witteveen+Bos (W+B). Thus, our method should consist of model testing and validation (Binnekamp, 2020). We will develop a system that detects the desired object from other existing algorithms and produces a prototype. We apply this to our case, which will provide us with experience and its impact within the company. Then we evaluate the system to see how good it works and what it can mean for W+B. Figure 14 shows the steps we will take on how to reach the desired results for this development project.

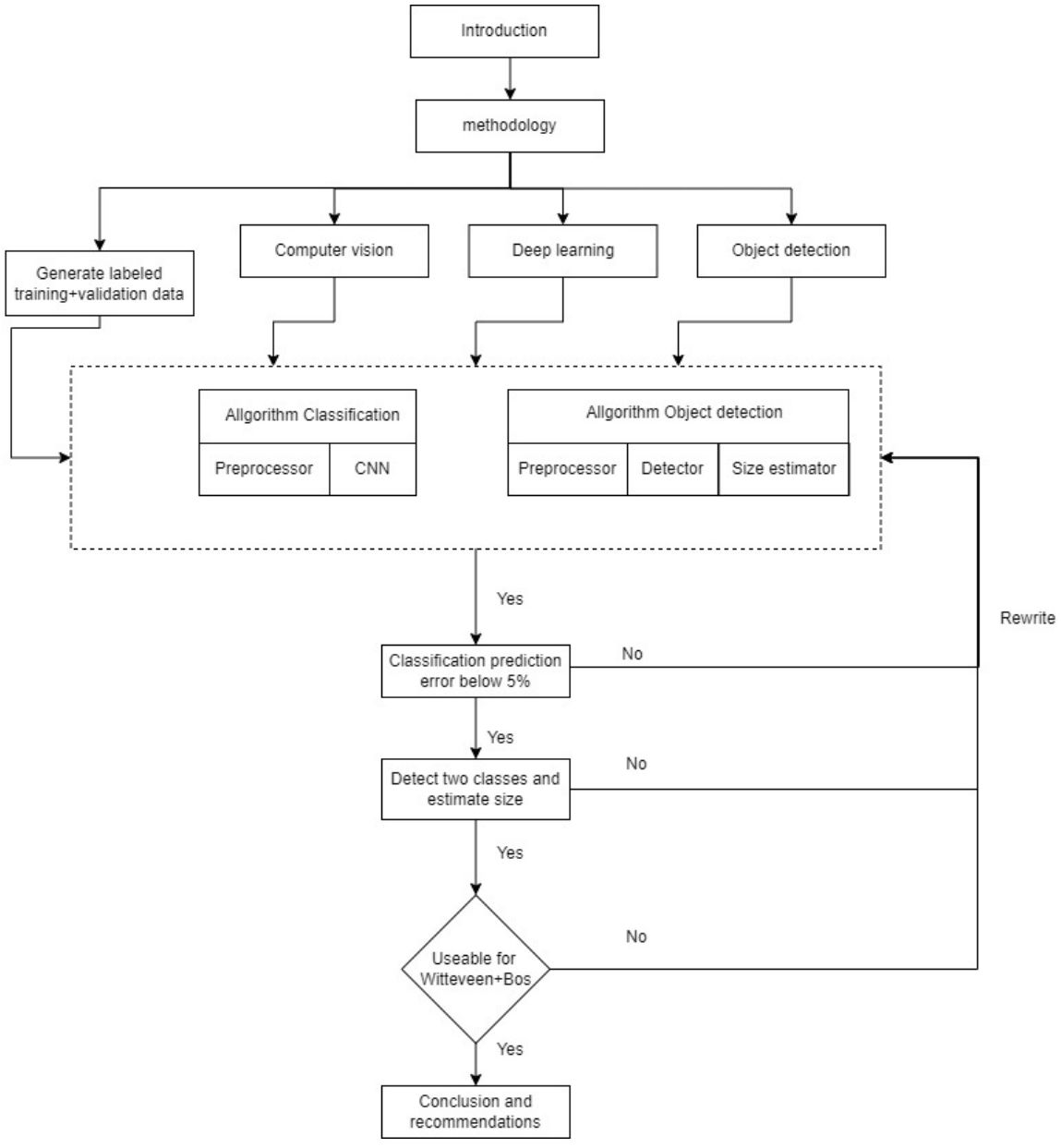


Figure 14: Framework

In figure 14 we see that after the introduction to the topic is followed by the relevant literature needed for this project. This is meant to gather information on deep learning, object detection and how to pick an algorithm. Then the methodology is presented to lay out how the project will be executed. We will be making an algorithm for a classification task and one for the object detection task. An object detection algorithm will be used based on the ease of reproducing the results. We will also consider whether the algorithm will be operated in the cloud (Google Colab) or on a device. After this phase, we can design the algorithm which consist of the pre-processor (convert image or video to a suitable format), detector and size estimator. While it is being designed, we can also start to generate training data.

W+B is providing 40.000 High Definition (HD) images containing piling sheets and natural banks, which need to be labeled for classification and object detection. In the classification task we prepare the data by storing images in a folder for each class. The classes for classification should be similar to the assessment scores from Digigids or similar. They also need to be discussed with W+B and the supervisors. For the object detection algorithm, we will only use images of piling sheets and annotate them. There are Python packages such as LabelImg (Tzutalin, 2022) in which the user must specify the region of interest of the object. When exploring the raw data-set, we should consider what the object of interest and the reference object will be. The reference object should be clearly distinguishable, for instance a stone is either red or blue. The dimension of the reference object should be known so we can estimate the actual size from the images. Ultimately, we will have annotated data of the classes:

- Dimensional feature
  A feature on a piling sheet so we can determine the profile.
- Reference object
  This will be needed to link the size of the corrosion to metric units. The anchors on piling sheet might be considered for this role.

After the analysis stage we should determine the **requirements** of the algorithm. This includes the input from W+B and relevant literature. W+B has currently finished assessing all the images with four people within two months. This will provide us with ground truths for a classification task. A review of Computer Vision (CV) on livestock (Borges Oliveira, Ribeiro Pereira, Bresolin, Pontes Ferreira, & Reboucas Dorea, 2021) has shown that classification algorithms have an accuracy of close to 90%. This, however, should be taken with a grain of salt, as these have been acquired with large data-sets. The object detector also has an accuracy of around 90% and with bounding boxes it should be present at the location that we want. Since W+B has finished the assessment task within one month with two people, it should also mean that the algorithm should be close the to what the specialist have performed.

When we know what to create and what the desired accuracy metrics are, we can start the **system synthesis** of this research. This consists of an iterative process to reach the desired algorithm. We start by creating a classifier and train it until it reaches an error rate of 5%. We can start by creating a data-set with a few classes with 100 images per class. A criteria should be made what images to put in what class. A general criteria is:

1. A human should be able to classify the image.
2. The images should be perpendicular to the natural bank or piling sheet.
3. There should be a body of water visible in the image.

The amount of data per class should be the same for all to avoid class imbalance. In figure 15 and figure 16 we can see what the classifications can look like, and we take into consideration that the amount per class is roughly of the same size. The classifier should be able to access a data-frame and store its findings as seen in table 3. The architecture of the algorithm will take inspiration from other existing neural networks such as inception and residual networks. We play around with the amount of layers and compare what performs better.

Next, we continue with object detection try to detect dimensional features of the piling sheet and estimate its size in pixels. The training data needs to be annotated and we can use LabelIMG for this task. The first algorithm up for exploration is the YOLOv4 algorithm that uses a Convolution Neural Network (CNN). At later stages a different algorithm (Retinanet) can be used for model comparison. The algorithm should first try to identify the objects correctly. We expect that there will be some wrong predictions that we will discuss and analyze. During this process, we are still in search for other methods through similar research and take

into consideration whether they are reproducible. When the algorithm works sufficiently enough (adding a bounding box and correct prediction), we can increase the amount objects to detect. If this is not the case, measures should be taken such as increasing the amount of training material or resorting to a different algorithm (Retinanet).

(a) Grass

(b) Rock

Figure 15: Classification category that are not piling sheet.

(a) Mild

(b) Heavy

Figure 16: Different kind of piling sheet corrosion.

The next stage is to adjust our object detection algorithm so that it can detect the reference object and dimensional features of the piling sheet. We will have to know the actual dimensions of the reference object and store it as a global parameter in the algorithm. In case an image contains the reference object, we should be able to create a pixel to actual size conversion rate. This conversion rate can be used for other images without a reference object, as we believe the boat is not deviating too much off course. In case the model finds another image containing the reference object, then the pixel to actual size can be adjusted and used for the following images. Important data that we need from the object detector is the height of the dimensional feature and the distance between the two closest dimensional features. Figure 17 shows an example of what can be seen. The data of the bounding box should also be able to be exported to a data-frame which will store the height of the dimensional feature and calculations.

Figure 17: Example of what the detector should get

The last step is to merge the algorithms so that all relevant data from the classifier and the object detector can be mixed. The images will first run through the classifier and if the classifier predicts "metal mild" or "metal heavy", it can be run through the object detection model. The images containing grass or rock do not have to go through the object detector, since W+B wants to know the profile of the piling sheet and something about the corrosion. All the data from the images are then stored into a data-frame shown in table 3.

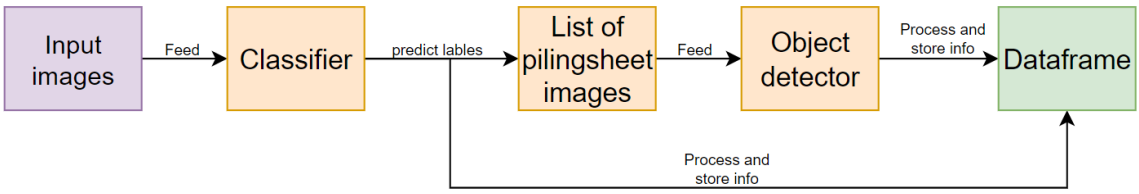


Figure 18: Flow of the algorithm

Filename	cameraLon	cameraLat	cameraAlt	Material	Distance bumps	Rust grade	Reference object	Height
HigRes_1	x.xx	y.yy	z.zz	Metal	100	Bad	No	70
HigRes_2	x.xx	y.yy	z.zz	Grass	Nan	Nan	Nan	Nan
.....	.....	.....	.....	.....	.....	.....	.....	.....
HigRes_N	x.xx	y.yy	z.zz	Rock	Nan	Nan	Nan	Nan

Table 3: Example output of data-frame.

3.2 Requirements

The requirements can now be set to cater towards W+B:

- The classification algorithm should be based on the Digigids.
- An algorithm should also have all the 4 classes of Digigids.
- Dimensional features needed are the height above the water and the distance between the bumps.

- Most classification projects reach around 90% accuracy. No accuracy has been set by W+B, but we have decided that ours should reach similar values.
- The metrics for object detection also reach around 90% for both the Intersection over Union (IoU) and the mean Average Precision (mAP) with large data-sets. We have also decided that ours should be around a mAP of 90% and IoU of around 70%.

3.3 Evaluation metrics

A satisfactory result is received by evaluating the algorithm, which consist of verification and validation process (**System evaluation**). Since this is a model test, verification is done by manually comparing what the algorithm has produced and how much is predicted correctly. This can be approved by W+B and TU Delft supervisors. We can validate the algorithm by considering the commentary of W+B whether the algorithm has the potential to increase their efficiency for monitoring piling sheets or for future projects. The results will be discussed and recommendations will be compiled.

We evaluate classification with precision, recall, f1-score and model accuracy. To get these we let the model predict several images and compare the predicted value with the actual value. These will result in an amount of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The precision can be calculated with the equation given in 3 and concerns itself with the amount of FP's the model has made. Another metric we can look at is the recall value, which focuses on the amount of FN's the model has made. The $F_1 - score$ can tell us about the balance of FP's and FN's. The accuracy is our main metric and can be calculated using equation 6.

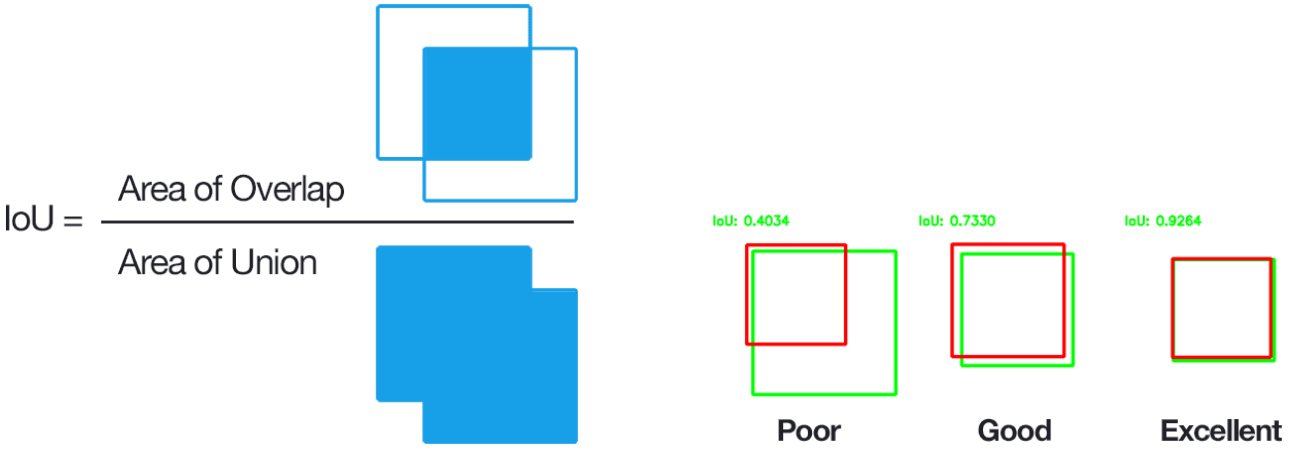
$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 score = \frac{2}{(\frac{1}{Precision}) + (\frac{1}{Recall})} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

$$Accuracy = \frac{TP + FN}{TP + TN + FP + FN} \tag{6}$$

We evaluate the object detection algorithm with the addition of Intersection over Union (IoU). This is computed using the validation data-set in which the YOLOv4 algorithm itself will compare the bounding boxes of the "Ground truth" with the predicted box. The IoU is defined as the ratio between the overlapping area with the total combined area (see figure 19a). In figure 19b we can see different values for IoU and how they would look. Since we are concerned with calculating the height and the distance between bumps, we feel that the IoU of our algorithm should be around 70%. When our algorithms do not meet the requirements or is not satisfactory of W+B, we shall continue improving the data-set or the architecture of the classifier.



(a) Intersection over Union.

(b) Different IoU values.

Figure 19: Used metrics for object detection (source:Pyimagesearch.com, 2016).

4 Design of the system

This chapter describes the various decisions made, when creating the parts of system. It will go through the platform and storage used. The creation of the data-set and any tools used are explained. The architecture of the classification Convolution Neural Network (CNN) and You Only Look Once (YOLO) are also explained. Lastly, we will go over how the outputs of both algorithms will be used and stored in a data-frame.

4.1 General

4.1.1 The notebook environment

There are three main ways to develop deep learning algorithms (Chollet, 2021) and we will choose which platform fits us. These are: a physical workstation with an NVIDIA Graphical Processing Unit (GPU), Cloud computing services like Google or Amazon and a hosted notebook service like Google Colab. The decision on which to choose will rely on cost and the extent of the usage of deep learning algorithms.

A physical GPU installed in a workstation is the most preferred when the use of deep learning algorithms is extensive. These however can be expensive if the costs fall on one individual. Prices for consumer GPU's used for deep learning applications usually range between $1000-$3000.

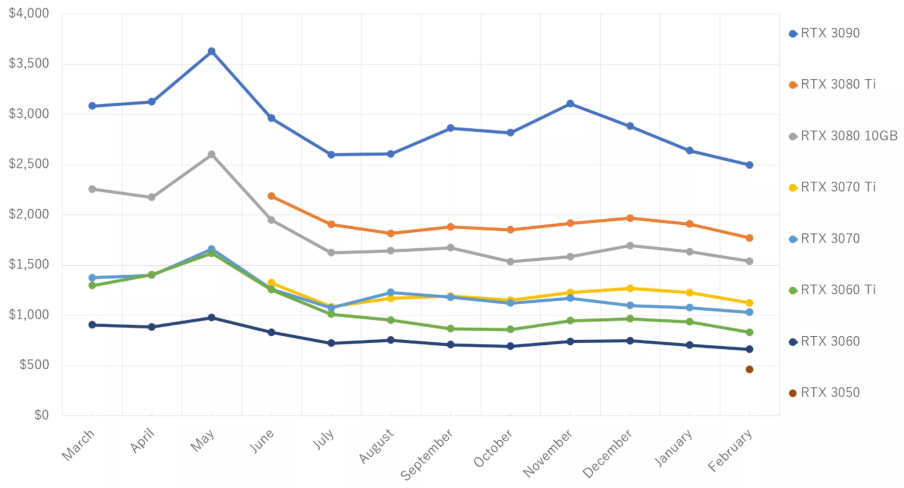


Figure 20: Prices of NVIDA GPU's (source:Techspot.com.)

The alternatives to having a physical Graphical Processing Unit (GPU) installed are a paid cloud computing service and a free cloud computing service. A cloud computing service from Google or Amazon is an option, if deep learning projects are needed for a short amount of time. When more workloads are needed over a longer time and privacy is of concern, a cost analysis should be made whether this option is worthwhile. The cheapest way is to make use of the service Google Colab, which is free and can be used to test small scale deep learning projects. Getting access to faster GPU's or a subscription to a cloud computing service for a fee in Google Colab.

The preferred way in this project is to use Google Colab and slowly move them to a Microsoft ecosystem. Visual Studio Code (VS Code) is a similar service from Microsoft that can read documents created in Google Colab. Using VS Code is beneficial, as W+B is using Microsoft Azure to some extent. Microsoft Azure is a cloud computing platform with various services, and a way to deploy applications from VS Code. Training of the Artificial Neural Network (ANN)'s will be done on cloud, while inference can be performed on a personal laptop. This is done so the algorithm can be used in areas with no internet connection.

Google Colab has some some restrictions on the use of their GPU with the free account. First, our total training time is limited to six hours after which the notebook would forcibly stop computations. Second, during those six hours, we had to be present within 30 minutes to keep the tab active. Else, it will be stopped and the results are gone. We searched on online forums in finding ways to alleviate this . They have mentioned the use of a browser code to automatically keep the tab active, but these did not work for the author or it has been addressed by Google. Finally, the memory resources are also limited. After compiling and training around four

models, Google Colab would cut us off from using their GPU. The time in which we could use the GPU again after a suspension would vary from several hours to a day.

4.1.2 The storage

Google Drive will be used as storage for training material, algorithm, saved models and other output material. The student, however, does not possess a lot of space on Google drive and therefore only a subset of the images will be used. The subset must contain 50-100 images of each class. More images will be added as the algorithm takes shape and after data-set is reviewed by an expert. A possible way to alleviate this is to resize the images and then upload them to the Google drive. The other being buying more space from this drive.

Resizing the images is done to create a large data-set as possible in the cloud drive. The ratio is kept the same, by first copy pasting the original image to Paint and then checking the dimensions of the image. Knowing these dimensions, we can download a batch of images and then automate the resizing process in Python.

4.1.3 The Python libraries

There are several Deep learning libraries that can be utilized in Python and we need to find which one to use. A good way to know what libraries to use is to look at ML competitions organized by Kaggle.com. Figure 21 shows the most used python libraries from competitors. We see that Keras, Pytorch and TensorFlow are some of the popular libraries when it comes to deep learning. When it comes to Machine Learning (ML) we see that LightGBM, XGBoost and Scikit-learn are popular.

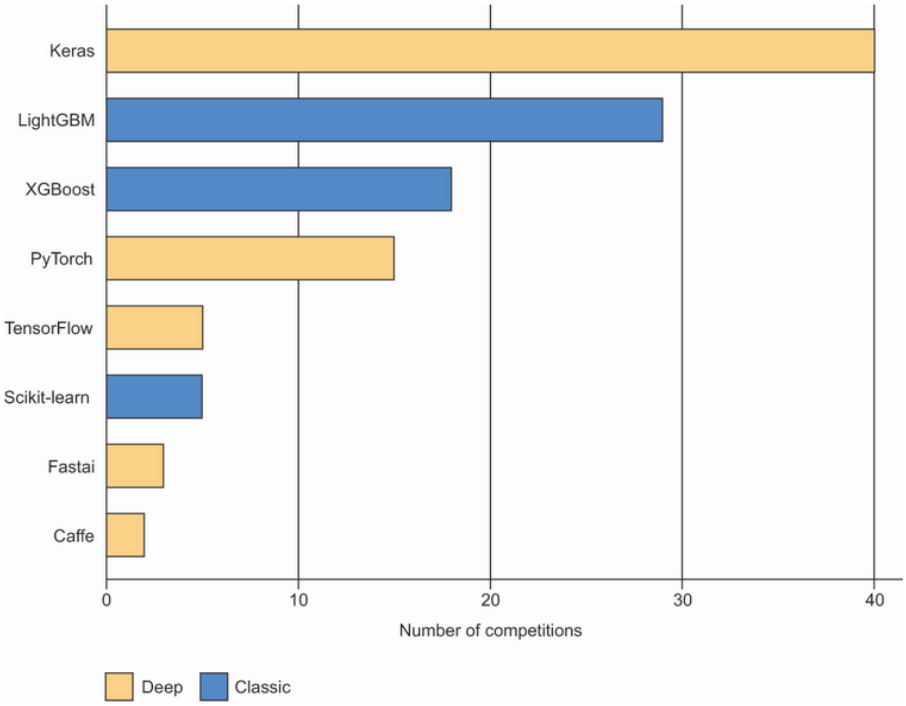


Figure 21: Popular libraries (source:Chollet, 2021)

We decided to use Keras and TensorFlow, as the community support is relatively active, and the TU Delft library has resources on Keras. Scikit-learn will be used for verification purposes. The SK-learn Application Programming Interface (API) already has some known verification methods built like confusion matrices and how to report those results. To create data-frames we will resort to pandas and for any image operations we use "OpenCV".

4.2 Data-set

In the servers of W+B there are 40.000 images of the Hoofdvaarweg Lemmer-Delfzijl (HLD). The images all are assigned to a geo-coordinate and are 10328x7760 pixels. We have downloaded several images and wrote separate helper algorithms to loop through all the original images and then resized them to 1282x963 pixels.

This aspect ratio was chosen, as it was the recommended aspect ratio when copy pasting the original image to an empty canvas in the software paint. After the images were resized, it was time to add them in separate folders.

Labelling for a classification task is done by storing the images to a different folder. In our data-set we have a training set and a validation set. We have separated such that it closely resembles other similar data-sets like SDNET2018. Within a training and validation folder, we need one folder for every class and in our case we tried making four classes which are: grass, rock, metal_bad and metal_good. The norm of DIGIGIDS2019 had scores such as "goed", "redelijk","matig" and "slecht" (Good, Average, Moderate" and Bad), however we were concerned on the availability of images for all the classes. We then considered two data-sets for classification: one with four classes and one with 6 classes. We combined the classes metal_good and metal_acceptable into one and metal_moderate and metal_bad into the other. However, W+B suggested to try a classification with six classes which would divide the metal section in "good", "acceptable", "moderate" and "bad". This is more in line with the DIGIGIDS2019 that W+B is using to assess the piling sheets.

We have considered some general rules for the data-sets of both four classes and six classes:

- There needs to be a body of water at the bottom of the image.
- There needs to be either grass, rocks or metal piling sheet. All others like wood or concrete are excluded.
- The image should be perpendicular on the banks.

4.2.1 Data-set for four class classification

We made a data-set with four classes, since there were several reasons we were unsure of. First, we were not sure if we had sufficient images for six classes. By combining classifications for corrosion, we could avoid having an imbalanced data-set. Second, the differences between the classes according to the DIGIGIDS2019 can be hard to discern. There are instances where more context, like subsidence, or an experienced specialist is needed to asses the piling sheet. The piling sheet were assessed on a specific length and not per image. In one section an image might show a piling sheet with deformations and corrosion, while the next image might show a piling sheet with just corrosion. This can be a problem for the algorithm as the corrosion looks identical. This might make classifying according Digigids difficult and a four-class classifier with just "Grass", "Rock", "metal good" and "metal bad" was made.

The classes "Grass" and "Rock" are in both data-sets, the only difference is that the amount present in the four class data-set is larger. This is done to match the combined total of the metal classes. These two classes are the non-piling sheet materials and are at time hard to discern from each other. In most cases there is rock present, but the vegetation has grown to such a state in which the rocks are not visible. Figure 15, in chapter 3, shows what we have trained.

The class "metal bad" will consist of images labelled as "Bad" and "Moderate" by W+B. These images will usually exhibit deformations or various type of corrosion. The corrosion will usually be localized as pitting, crevice, waterline and galvanic corrosion. The texture of the corrosion in this case would be rougher. The images in figure 23 are some pictures in this class.

The class "metal good" will consist of images labelled as "Acceptable" and "Good". These images would usually contain uniform corrosion and few to no other types of of localized corrosion. The texture of the piling sheet would be much smoother than the "Metal Bad" class.The images in figure 22 are some pictures that would be in this class.

4.2.2 Data-set for six classes classification

We then created a data-set that has the same amount of assessment scores as Digigids, that is supposed to give more insight on the condition of the piling sheet. The first only had metal_mild and metal_heavy and focused on the scale of the corrosion. W+B also desired this to fully comprehend what CNN's are capable of. This data-set had the classes rock, grass, metal good, metal acceptable, metal moderate and metal bad. The data-set was made with an assessment report of Heijmans. This report was made in 2021 and the assessment should still be valid and contain images that can help us decide for our self what the conditions are. Within Heijman's report we searched the words "goed", "acceptabel, "matig" and "slecht" and noted the coordinates. Each time we would check if it concerned a piling sheet and then write down the coordinates. The coordinates

were then searched in a file from W+B, that would compile a list of images in that region. This was all done in python, to make automating easier. The final data-set was made with assistance from a specialist of W+B. A brief description of the classes is given below.

The class "Metal good" will only contain images of piling sheet that are in good condition according to the expert from W+B. The metal sheets can have uniform corrosion with an overall smooth surface. Localized corrosion could be waterline corrosion and galvanic corrosion on few bolts. The surface of the corrosion will generally be somewhat smooth. The piling sheets in these conditions can usually function for another 30 years. In figure 22 we see some pictures of what was is "good" according to Digigids.

The piling sheets that are classified as acceptable show at least all the characteristics of the class "metal good", but with a more intense localized corrosion. The waterline corrosion will have a clear discoloured and rougher surface. There are more locations of bolts or plates showing galvanic corrosion. In figure 22 we see an image of what is "acceptable".

(a) Good

(b) Acceptable

Figure 22: Classification six classes

The "metal moderate" class has images that that are sometimes hard to discern with those classified as acceptable. This is the case since specialist have a more semantic understanding on what is there and also the data they have from the point cloud. In most cases the faults are like that of acceptable, but with a more recurring galvanic corrosion or even a missing bolt. The waterline corrosion has a more discolored and rougher surface. Some deformation can also be present, but these could be hard to discern from the images.

The images labelled with "metal bad" had many defects besides just corrosion. The piling sheet would usually have a darker tint of uniform corrosion. The localized corrosion would also be intenser. Waterline corrosion had a strong discolouring in the area in which water would be present during different tides. The area with waterline corrosion would be rougher in texture. Often the corrosion would create crevices and pitting. There would also be more galvanic corrosion present and even missing bolts. It can also be that the piling sheet is deformed. This could be caused by clashes with a boat or just the metal failing.

(a) Moderate

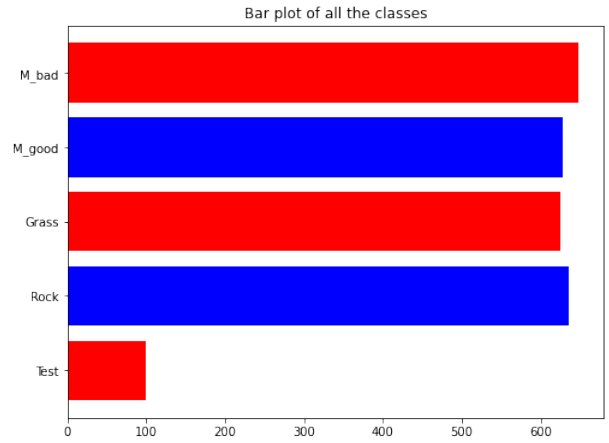
(b) Bad

Figure 23: Classification six classes

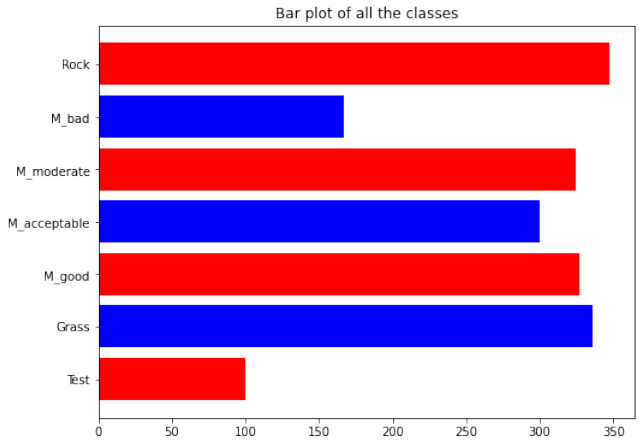
A script was built to assist in getting the data from the W+B server to the user's local device. Since Google drive would be used, we needed to resize the images as the original images are around 20 megabytes and a data-frame was built for the contents in the total data-sets for some basic data management. This data-frame gives us insight on the quantity of images in a class and the relative size compared to the total data-sets. The data-frame is also helping us avoid having duplicate images in other classes.

4.2.3 Creating image classification data-set

Labelling data for classification is sorting the images in a separate folder in which each folder would represent a class. In figure 24 we can see the amounts images present per class. There are two data-sets created, one with four classes and one with six classes. The reasons for this is due to possible imbalance of one class. A set was also created for a model test, this was needed to simulate the model going over images in a W+B folder.

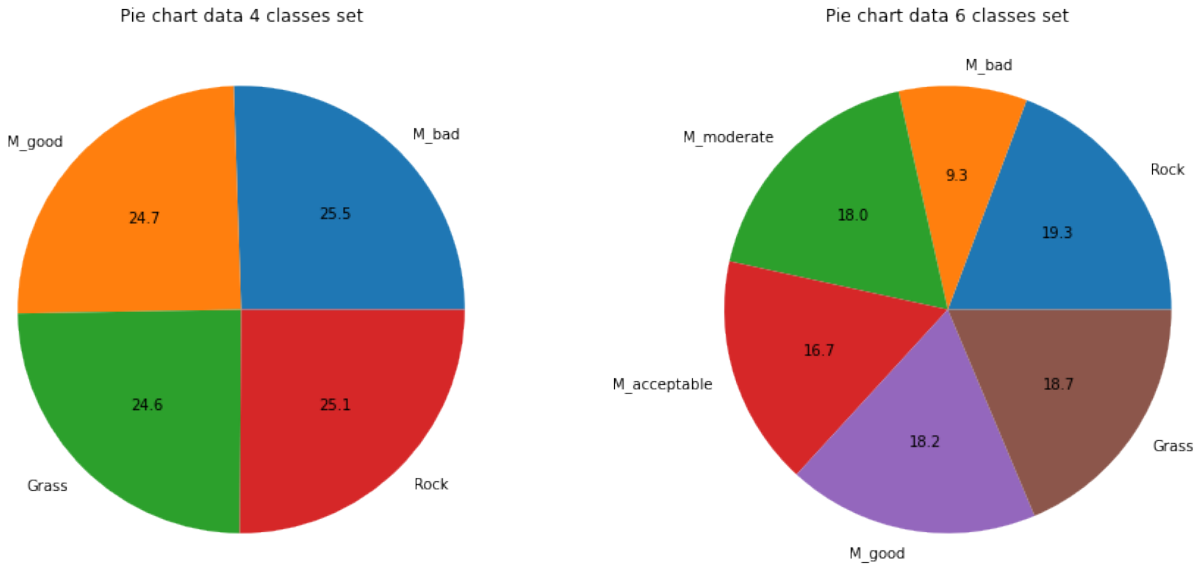


(a) Four classes



(b) Six classes

Figure 24: Overview of the classification data



(a) Four classes

(b) Six classes

Figure 25: Pie chart to check balance of classification data-sets

In figure 25 we can see the portions of each class for the four and six class data-set. The four class data-set has roughly the same proportions for each class, while in the six class set we see that "M_bad" has just 9.3% of the total set. The raw data-set was carefully searched, but no more images could be found that could be classified as "M_bad". In case we would increase the other classes, then we would have to account for class imbalance in our script. A multi-class data-set is already expected to perform less than binary cases, but the issue becomes more complex when class imbalance is present (Haixiang et al., 2017).

Classes	Total images	Selection	Number of images	Percentage (%)
4	2534	Training	1774	70
4	2534	Validation	760	30
6	1801	Training	1301	70
6	1801	Validation	540	30

Table 4: Criteria for selection of training and validation set for four and six classes.

In table 4 we see the amount of images for the four class and six class data-set. The four classes has more as we wanted to consider the imbalance of a class in the six classes data-set. The amount of images after the split is also given. The split between training and validation happens with the shuffle on, as turning off the shuffle will result in the last class not being in the training set or the first classes not present in the validation set.

4.2.4 Data-set for object detection

Labelling data for an object detection task is vastly different than a classification task. The general task is to annotate images by putting bounding boxes on images and exporting the class of the bounding box and the coordinates to a file. This can be done using various annotation software. The software of our choice is LabelImg as it is open source and thus free. LabelImg does require a python version lower than 3.8, so we created a separate python environment for labelling.

Different detectors can require a different kind of training setup. The YOLO algorithm requires a text file with the image in both training and validation folders, while others like Retinanet requires an eXtensible Markup Language file (XML) file. Since we were still in the process of finding out the differences between various detectors, we had decided on outputting to the yolo format and the XML format. Retinanet has the added complexity that the network needs the training and validation material in a TFRecord format, while YOLO just needs the images and txt files uploaded to the darknet environment. During the creation of training

data-set for Retinanet we came across compiling errors related to the format of the training file. It was then decided to drop Retinanet for YOLO.

We then drew bounding boxes on all the object of interests that are necessary to estimate the profile of the piling sheets. We only used images containing the piling sheet, as we did not need anything from images that were labelled "Grass" or "Rock" by the classifier. In this case the object of interests are the dimensional features of the piling sheet and a reference object. Going through the images we decided to use the metal anchors on the piling sheet as a reference object. In figure 26 the exact method to label our images can be seen. The dimensional features are the bumps in the piling sheet and make sure that the bounding box encompasses the object with a small part of the water below and small part of grass above. The is true for our reference object, we make sure that our object sits in our bounding box.

(a) Good labelling

(b) Bad labelling

Figure 26: Labelling our objects in LabelImg.

We have compiled some rules on what images we label for the training data:

- The images need to show the metal piling sheet from a perpendicular stance.
- The image needs to show a body of water and a sky.
- Images where we would think the network would have trouble recognizing the features would be excluded

Labelling data for object detection concerns using annotation software to go over images in a train and validation folder. In each folder an image is paired with its annotation file, which could be a text file or XML file. The annotation file contains the class of an object and the bounding box coordinates provided by the user. In figure 27 we can see the amounts images present per training and validation folder. We tried to keep the ratio between train and validate to 70:30. The model test folder in this case is the same one from the classification task. The model will only receive images that has been predicted as "metal" by the classifier.

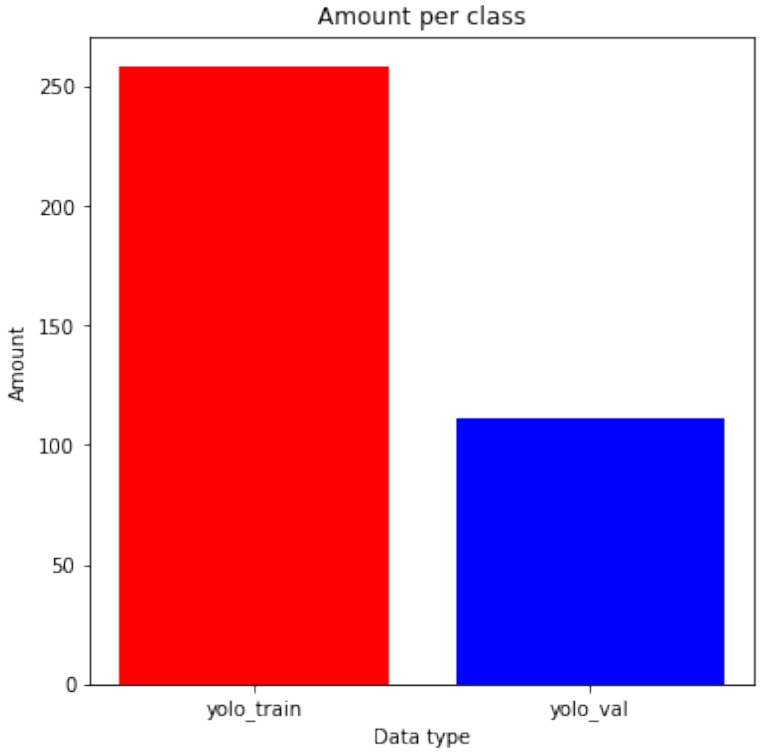


Figure 27: Overview of the object detection data.

4.2.5 Model test set.

A set of images will be created to simulate the images in a folder on the server of W+B, this will be used for the Model Test (MT). The way the images are loaded into the model we do not have access to other information of the images. This is unfortunate as the name of an image can tell us about the geographic location of that image. Thus, this data-set has been made with 100 images. The images have been selected to represent all the classes from the 6 class data-set. A notebook was made to make sure the contents of this data-set is not available in the main data-sets.

The model test set will be used to pass inference on the models. A data-frame, containing geo-coordinates of all the images, was provided and a sub data-frame is created by searching up the names of the images in the model test set. The images will be passed to the classification model and to the object detection model if needed. Both models will in turn provide all the predictions to the same data-frame.

4.3 Classification algorithm

4.3.1 Architecture of the CNN

A basic Convolution Neural Network (CNN) contains the input layers, convolution layers, maxpooling layers and the output layers. Our starting model will look like a smaller VGG16 model. We then made changes to the structure like adding residual connections and making it look inception like. Afterwards we explore what adding layers does, changing the filter size of convolution layers and the use of augmented data. We will try to show all the steps at improving the model that is possible within the restrictions of using the GPU of Google Colab. The amount of convolution layers is up for debate and figuring the exact amount is done by experimenting and documenting the outcome.

The model has a general structure in that will be the same no matter what block we will use. The input layer is needed for any neural network. In this layer we set the size that will be used throughout the network. Following the input follows either the augment layer or the rescaling layer. The augment layer is made part of the network instead of the straight to the data-set instead. The re-scale layer is needed to keep the values, which range from 0 to 255, between 0 and 1. Then follows a block of code, which are explained a later paragraph. After the blocks we have a dense layer which functions as the output layer. The amount of units in

the output layer is dependent on the amount of classes in our data-set. Since we have more than two classes, we need a "softmax" activation function in the output layer. Using "softmax", we can get the probability of each class when inference is passed. The model is then compiled with Adam optimizer and a sparse categorical cross entropy loss function. Adam was chosen because it is supposed to be computational efficient, low memory required (Kingma & Ba, 2014). A sparse categorical loss entropy loss function was used since we have more than two labels in our data-set. This will give as output a tensor with the probabilities of all the labels for that image. As metrics we want to monitor the accuracy of our model.

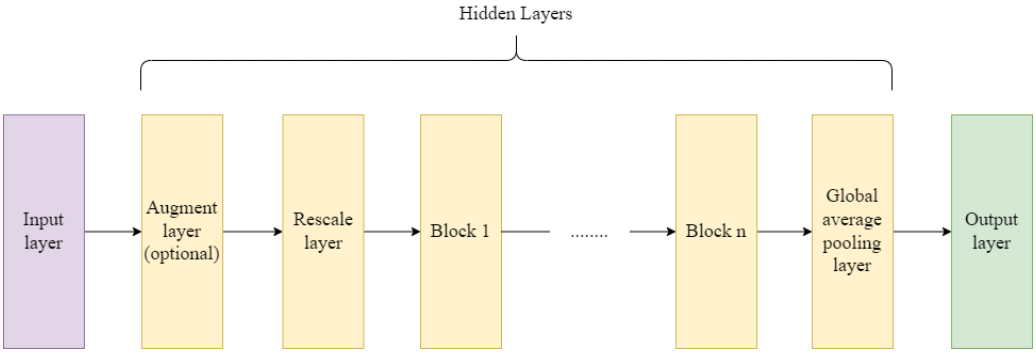
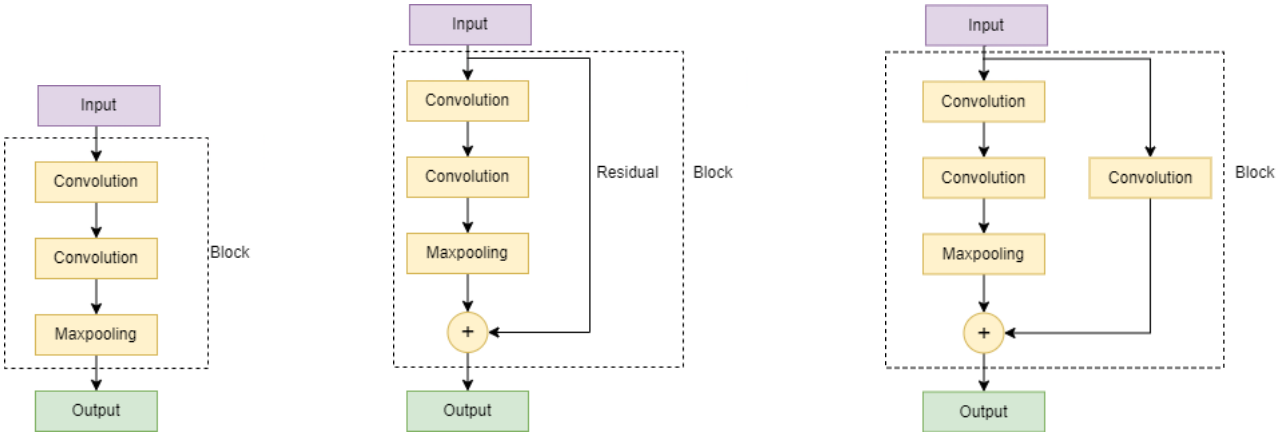


Figure 28: The general structure of our CNN.

In our model we opted for an augmentation layer to utilize the GPU for augmentation techniques. Augmentation is useful technique to diversify the training data in a realistic manner. this helps expose the model to different data and slows down over-fitting. Over-fitting is a term use to denote when a model fits exactly to the training data which makes it hard to predict unseen data. Our augmentation layer consists of a rotation layer, zoom layer, a horizontal flip and a contrast layer. in figure 31 we see the difference we are using the basic vgg16 with and without augmentation.

We have created a main structure with three different blocks that could make up our custom build CNN. The first block is the basic network that is similar to a vgg16 neural network in where the maxpooling layer usually follows a convolution layer. we have decided to have two convolution layers and one maxpool layer (see fig 29a). The second block is the same as block 1, but with a residual connection that will be added to the output (see figure 26b). This type is inspired by the Resnet neural network. The last block, in figure 26c, goes further by adding a convolution layer in the residual connection. This is has taken the inspiration of the inception network. The filters in each block are either set to all 32 or to increase them by two for each sequential block. This is done so the output shape of each layer is getting smaller the "deeper" in the network we go.

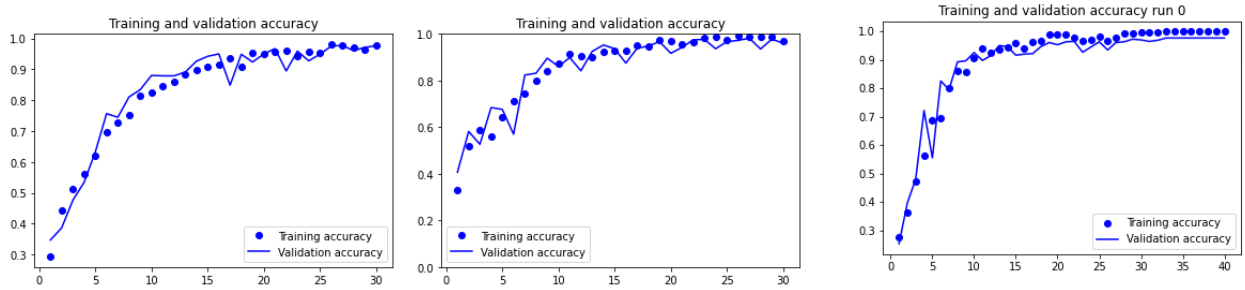


(a) Convolution (b) Residual connection (c) Inception like

Figure 29: Process of changing the structure.

Using residual connections is a method to increase our model performance. Residual connections refer to copying the input as a residual and adding it later to the output after a few hidden layers as can be depicted in figure 29b. The amount of hidden layers after adding the residual is dependant by trial and error, but we decided on

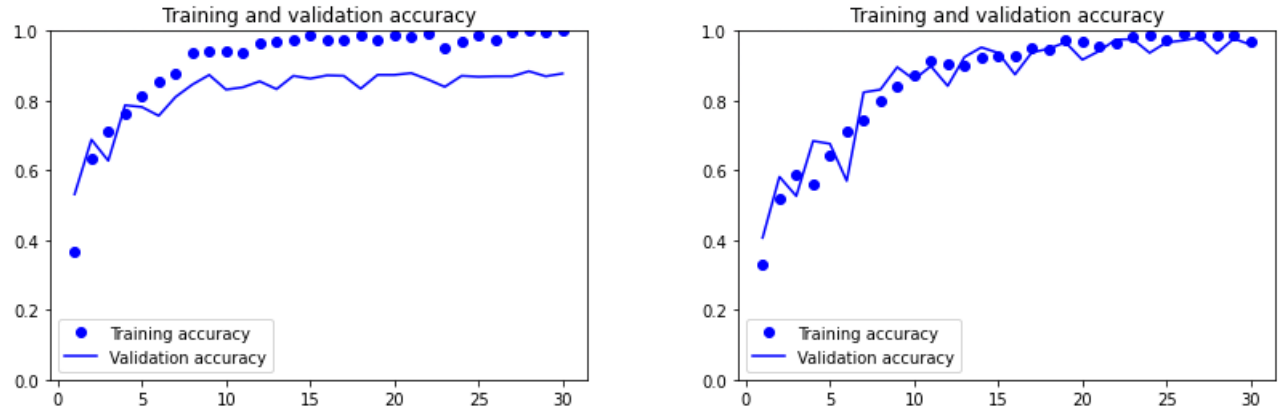
after the 2 convolution and the maxpool layer. The use of special layers such as batch normalization, separable convolution and a dropout layer are considered, but not the focus right now. After these techniques have been implemented, we can tune parameters like adding more blocks, changing filter size or turning of the augment layer.



(a) four blocks (b) five blocks (c) six blocks

Figure 30: Changing the amount of layer blocks for residual connections and no augments used.

We then tried changing one aspect of the model to see where the performance comes from. We tried different amount of layers, normal and residual connections, with and without augmentation and keeping the filter size the same or increasing per layer. For the residual block we can see in figure 30 that the model reaches an accuracy of above 90% faster when adding more blocks. The model with four blocks does that around epoch 17, while the model with 6 blocks reaches that around 10 epochs. In figure 30 we compare the same model from figure 30b with another model that has no residual connections, but same amount of block layers. We can see that model 31a starts over-fitting around epoch 8, while the model with residual connections is still following the curve so far.



(a) Without residual connection (b) With residual connection

Figure 31: Using a model without residual and with residual connection.

4.3.2 Post processing the results for classification

The classification outputs a vector containing the probabilities per class that needs to be worked into a data-frame. We use a python command to determine the index containing the maximum number (np.argmax) and using a switch statement we insert a the desired text into a the data-frame. In the case of the data-set with four classes we want the algorithm to write in two columns: "Material" and "Rust grade". We use the material column, so we can easily measure how well the algorithm detects piling sheet and is also convenient to feed the object detector a specific type of image. The rust grade is our main point of interest as we want to know how these algorithms could assist the specialist/managers in W+B. Material will have the options of Grass, Rock and Metal while rust grade will contain good or bad. in the case of the data-set with six classification the rust grade would contain good, acceptable, moderate and bad.

4.4 Object detection algorithm

4.4.1 Detector

We have tried using two detectors that were reproducible within the time frame. The first was YOLOv4 in which there is a growing community providing tutorials on this subject. The algorithm we used is taken from Alexey Bochovski Github page, who has forked his version from Joseph Redmond's Darknet. Custom training is done on the Darknet network in which the training and validation set should be sent to. Besides the annotated data-set, we had to upload a configuration file that holds the information on the neural network. On the authors GIT page there are rules given on what to change depending on the amount of classes.

A method to get the algorithm to work outside of Darknet needed to be written so we could also get information on the detections. Two methods, Open Neural Network Exchange (ONNX) and OpenCV, are available and both require the weights and the configuration file to create a Tensorflow format or an OpenCV Deep Neural Network (DNN) module. We choose the latter as that was easily reproducible and yielded similar inference results as in the DarkNet module. A comparison of the results can be seen in figure 32. OpenCV only allows us to use inference with a YOLO algorithm, thus training the algorithm should still be done using Darknet. With OpenCV however, we can get the coordinates of the bounding boxes, the width, the length and class of each box.

We have two classes that we want to detect and have around 350 images. This means we have to change the configurations file in which parameters are set in the algorithm. In that configurations file the max batches amount is set according to the rule $Classes * 2000$, the steps to $0.8 * maxbatches$ and the filters in (each of the YOLO layers) to $(classes + 5) * 3$. A text file called *obj.names* is made, which contains the names of the classes. Another text file is made containing the paths to the data and where the weights should be saved. As reference object we have chosen the anchor width, as this object has been observed in multiple images and has a fixed dimension.

The next detector we tried was a Keras implementation of object detection in which we use Retinanet. This algorithm needs the data-set in a specific format. The annotation file is given in XML files and the algorithm only takes the input files in a special "tfrecord" format. We then had to create a "tfrecord" data-set which can be an arduous task if unfamiliar with the subject. During this portion we had dropped on using Retinanet as we could not solve some programming issues. In most researches we have also seen that YOLOv4 is performing better compared to Retinanet. Yolov4 is also recommended as a starting point as it has a great generalization and is helpful when trying to get a proof of concept as quick as possible (Tulbure, Tulbure, & Dulf, 2022).

4.4.2 Post processing for object detection

The object detector model returns a vector contains a series of bounding box. Each bounding box contains the upper left x-coordinate, the above y-coordinate, the width and the height. Unlike the classification algorithm the class is already given to us with the confidence score and we write algorithms for both cases. In case a reference object is detected we can calculate the actual size to pixel ration. In case multiple reference objects are detected we just use the average between them. In the event no reference object was detected we use a starting estimation, that would be adjusted in case the next image does contain a reference object.

When the object detector has detected our object of interest, we then start a process to calculate the average height and the distance between the bumps. Each vector already contains the height of the box in pixels, we only multiply this with the actual size to pixel ratio and take the average in case there are multiple detection. To calculate the distance between bumps we first order the bounding boxes from left to right. We then take the left x-coordinate of the first two boxes and calculate the pixel distance between each box. We only store the distances that are larger than half of the average bounding box width and smaller than 3.5 times the average bounding box width. This is done to weed out any distance that are too small or too big. Finally, we convert the pixel distance to an actual distance in centimeters.

Algorithm 1 Inference algorithm

input: Path to the images
output: Data-frame with relevant info
Create data-frame
Load classifier
for Images **in** path **do**
    model_classifier predict image
    Store information in data-frame
end for
Load object detector
for Images **in** Metallist **do**
    model_objectdetection predict image
    heigth = average heigth bounding boxes
    width = average width bounding boxes
    if distance between bumps $>$ 0.5 width **And** distance between bumps $<$ 3 times width **then**
        append distance between bumps to *Distance list*
    end if
    Distance= average *Distance list*
end for

4.5 Summary of design

In this chapter we went over the various aspects of the design system. we first started with general specifics such as the programming language, python packages and storage. The we went into detail how the data-sets should look. We decided on using a data-set with four and another with six classes for our classification algorithm. We also decided on how the data-set for the object detection will look. Afterwards we created the algorithms for both classification and object detection algorithm. In the classification we went over what is in the network (the architecture) and what we would do after inference. In the object detector we also went into what the configurations are for the YOLO network and how we would get height and distance between bumps.

5 Results and analysis

This chapter will discuss the results of training and testing the algorithms. The results of passing inference of both algorithms are presented and compared to other similar research. The research is then evaluated through verification and validation methods. The verification is a quantitative method to justify the algorithms, while validation is a qualitative judgment if the algorithm has a right of existence.

5.1 Results from model test set

5.1.1 Classification results

We made variations of models for the four classes and six classes data-set. The variations are based on the different blocks mentioned in figure 30, adding blocks, turning off the augment layer and varying the filters per block. When compiling these models we also documented the trainable parameters, the train and validation accuracy and the loss graphs. This process was limited, because Google Colab allowed us to compile around three models per day, after which access to their GPU is denied. After compiling the models, we passed inference on the model test data-set in which we observed the accuracy. We made a "material accuracy" which is designed to see if the model can at least see piling sheets, while the "Model Test accuracy" concerns the same metrics that is shown in the train and validation accuracy.

Type	Blocks	Trainable parameters	Augment layer	Filters	Train accuracy (%)	Validation accuracy (%)	MT accuracy material (%)	MT accuracy (%)
VGG16	5	2353444	No	32-64-128-256-256	25	25	22	22
VGG16	5	84260	No	32-32-32-32-32	93	92	100	84
VGG16	5	164132	Yes	32-32-32-32-32	25	25	54	13
VGG16	4	1173284	Yes	32-64-128-256	25	25	54	13
ResCon	4	1216868	No	32-64-128-256-256	98	98	91	87
ResCon	4	1216868	Yes	32-64-128-256-256	97	97	92	83
ResCon	5	87556	No	32-32-32-32-32	98.3	94.4	99	96
ResCon	5	2397028	No	32-64-128-256-256	98.6	97.7	90	85
ResCon	5	2397028	Yes	32-64-128-256-256	96.8	95.5	90	67
ResCon	5	87556	Yes	32-32-32-32-32	94.4	92.7	94	9
Inception	5	126372	No	32-32-32-32-32	96.8	95.7	98	91
Inception	4	1583428	Yes	32-64-128-256	92.9	87	92	80
Inception	5	3149460	No	32-64-128-256	27.17	25.26	22	22
Inception	5	3149460	Yes	32-64-128-256	88.7	82.5	89	80

Table 5: Models with different structures, blocks, augmentation and filters for four classes.

In table 5 we see different variations with the amount of parameters and all the accuracy's obtained. We see that the vgg16 type is capable of classification as long as the trainable parameters are set low. This could be attributed to the vanishing gradients on such a type of CNN. This problem becomes significant when a network becomes too deep (Basodi, Ji, Zhang, & Pan, 2020). A way to relieve this is to change the structure

to residual block. we see that residual blocks all perform well enough. The next structure is the inception like. These perform mostly favorable too there does seem to a correlation to the trainable parameters and the use of augmentation. In general, there does not seem a vast difference between the model test accuracy, training accuracy and validation accuracy. We also see that the residual connection blocks perform generally well on this data-set.

Type	**Blocks**	**Trainable parameters**	**Augment layer**	**Filters**	**Train accuracy (%)**	**Validation accuracy (%)**	**MT accuracy material (%)**	**MT accuracy (%)**
VGG16	5	84326	No	32-32-32-32-32	90	90	84	57
VGG16	6	102822	No	32-32-32-32-32-32	98.8	87.5	82	52
Rescon	6	3643949	yes	32-64-128-3x256	97.5	95.3	93	68
Rescon	6	3643494	No	32-64-128-3x256	98	95	78	65
Rescon	6	107174	No	32-32-32-32-32-32	99.99	95	83	65
Inception	5	126438	No	32-32-32-32-32	95	93	91	65
Inception	6	155238	No	32-32-32-32-32-32	99	94	89	66
Inception	6	155238	Yes	32-32-32-32-32-32	92.4	91.8	84	66

Table 6: Models with different structures, blocks, augmentation and filters for six classes.

In table 6 the different models for the six classes data-set is shown. The model types that have performed well on the four class data-set where used. We can see that by adding six blocks instead of five usually results in better accuracy, this can be observed with VGG16 and the Inception like blocks. The total training epoch was also adjusted from 30 to 50, as we observed the model was still learning at epoch 30. We also observe that the Model Test accuracy is overall much lower that the train and validation accuracy. This means the model has some problems differentiating between the corrosion classes.

Seed	**Train accuracy (%)**	**Validation accuracy (%)**	**MT accuracy material (%)**	**MT accuracy (%)**
0	98.3	94.4	99	96
1	97.3	95	96	92
2	96.3	95.6	90	85

Table 7: Best performing 4 classes model with different shuffled training and validation data.

The cross validated results for the four- and six-class classifier can be seen in tables above and below. The model used for the four classes is the residual connection with five blocks, where all filters are 32 and no augmentation. The model used for the six classes is the residual connection with six blocks, augment and increasing filter sizes. There are three different seeds used when splitting the total data into a training portion and validation portion, the seeds represent a random shuffle during the split. Training the object detection model was also inconvenient using Google Colab, as we had to take measure against being disconnected. In table 7 we see the results of the model trained on different portions of the total data-set. We see that the model test accuracy ranges from 85%-96% and thus the average is $(85 + 92 + 96)/3 = 91\%$. The accuracy of the model test set compared to the training and validation accuracy seem relatively close to each other. Thus, we can say the model has an easy time differentiating between a binary classification such as good and bad.

Seed	Train accuracy (%)	Validation accuracy (%)	MT accuracy material (%)	MT accuracy (%)
0	97.5	95.3	93	68
1	97.5	95.3	84	50
2	97.5	95.3	88	72

Table 8: Best performing six classes model with different shuffled training and validation data.

The best performing model for the six classes data-set was also retrained on different portions of the total data-set. In table 8 we see the results for the model with a residual connection type, with augment and increasing filters. We can see that all the models have a relatively good material accuracy, with "seed 0" having the highest of 93%. On the normal test accuracy however we see that "seed 1" has a difference of around 20% with the others. This could imply that the total training data can look different on different portions of the total data. The average accuracy of all three models is around $(68+72+50)/3 = 63\%$ which is satisfying enough. However, we can conclude that the model has some difficulties differentiating between the various corrosion levels.

5.1.2 Object detection results

The goal of the object detector is to detect dimensional features, so extract dimensions that could tell us about the profile of the piling sheet. The object detector needs to detect the bumps, so we can estimate the height of the piling sheet and estimate the distance between those bumps. The object detector will also try to detect a reference object, so we can convert the pixel distance to an actual metric. A visualization of the detections per images can be seen in appendix E.

There are four scenarios that could happen when a detector does a forward pass of an image:

1. Detect nothing.
2. Detect only reference object.
3. Detect only bumps.
4. Detect both bumps and reference object.

Case 1 means that the model failed to detect a thing and thus no dimension is calculated. In case 2 the reference object can adjust the pixel to actual size ratio, but still no dimensions will be calculated. Case 3 is a bit more fortunate, since we can calculate some dimensions. The pixel to actual size ration will however be a standard value. No reference object could also mean that the model has failed to detect such an object (False Negative). Case 4 is our most desired scenario as we have the dimensions and they could be converted using the pixel to actual size retrieved with the help of the reference object. We should still make sure there is a reference object in the image and not a False Positive.

The test subject has revealed some notable points. First, using inference on openCV seems to have a lesser result than using inference on DarkNet. Figure 32 shows the same image inference on openCV and DarkNet and we can see that DarkNet has detected two bumps more than openCV. We must use the validation set when evaluating the metric, since the test set contains raw data. To compare the bounding boxes of the ground truth with predicted bounding boxes, the validation set is used. The training of the model also took a significant amount of time. This model was trained with 253 images and 109 validation images, which in total took eight hours. Google Colab will shut down in 30 minutes inactivity and forcefully shutdown after six hours. A way to work around this is to set alarms every 29 minutes, but is still somewhat inconvenient as we need to be near the computer for six hours.

Figure 32: Inference using the last saved weights in DarkNet and OpenCV.

5.2 Analysis of the results

5.2.1 Verification of the algorithm

We compiled three models, each with a different, shuffled training and validation set from the total data and do this for a classifier with both four and six classes. We then pass inference of new images through the model. The test images or taken from the W+B server and should represent the raw data-set as much as possible. We used 100 images and can be seen in appendix D. The amount if images could be increased at a later point.

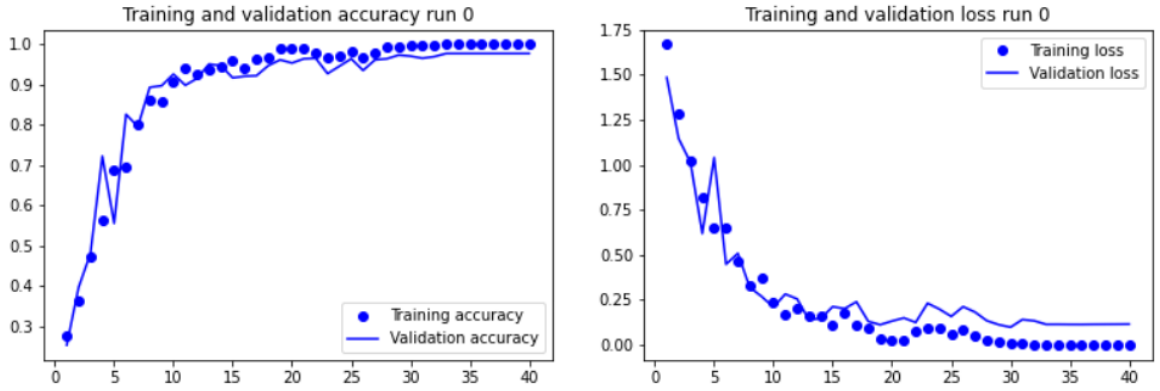
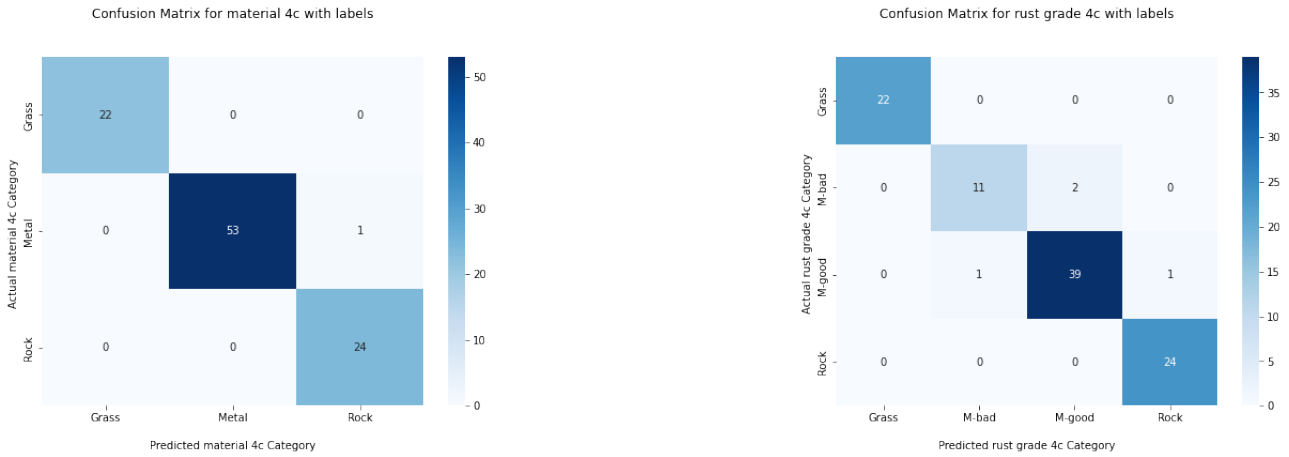


Figure 33: Accuracy and loss metrics with 4 classes.

In figure 33 we see the accuracy and the loss plot of the best performing four class model. We see that the training and validation accuracy are following each other very closely. When looking at the loss plot, we see that the difference between the training loss and validation loss is not changing around epoch 30. This indicates that the model does not need to train further and is over-fitting.

We then evaluate the model on the model test set and check the accuracy for the material and the rust grade. As can be seen in the previous sub-chapter the accuracy for material is 99%, while the rust grade is sitting at 96%. This could be attributed as the differences between the two classes are a bit unclear for the author. There were two metal classes, metal_bad and metal_good, to simulate the scores given by the NEN norm. For those two classes, the algorithm would fill in "Metal" in the "Material" column and then "good" or "bad" in the "grade" column. Whether the score for the condition is correct still needs to be verified by Witteveen+Bos. When the classifier has finished all its task, a list is extracted containing the names of images in which the material is labelled as "Metal". This list is needed to feed the object detection algorithm the specific images required.



(a) material

(b) rust grade

Figure 34: confusion matrix for four classes.

Figure 34 shows the confusion matrices for the four class models on the MT set. We can then create a confusion matrix using the API from the SKLearn python package (Pedregosa et al., 2011). We can see that the model is good at detecting just the materials (grass, rock or metal). The models do make some mistakes when differentiating between the metal good and metal bad, but the MT accuracy is still at a justifiable level. Thus, we can conclude that classifying the corrosion level in a binary fashion like "good" or "bad" works well enough. Table 9 shows that the material column has an accuracy of 99%, while the Rust grade column has an accuracy of 96%.

	Class	**Precision**	**Recall**	**F1**	
Material	Grass	1	1	1	
	Metal	1	0.98	0.99	
	Rock	0.96	1	0.98	
	Accuracy				0.99
Rust grade	Grass	1	1	1	
	M-bad	0.92	0.85	0.88	
	M-good	0.95	0.95	0.95	
	Rock	0.96	1	0.98	
	Accuracy				0.96

Table 9: Metrics for material and Rust grade of four classes.

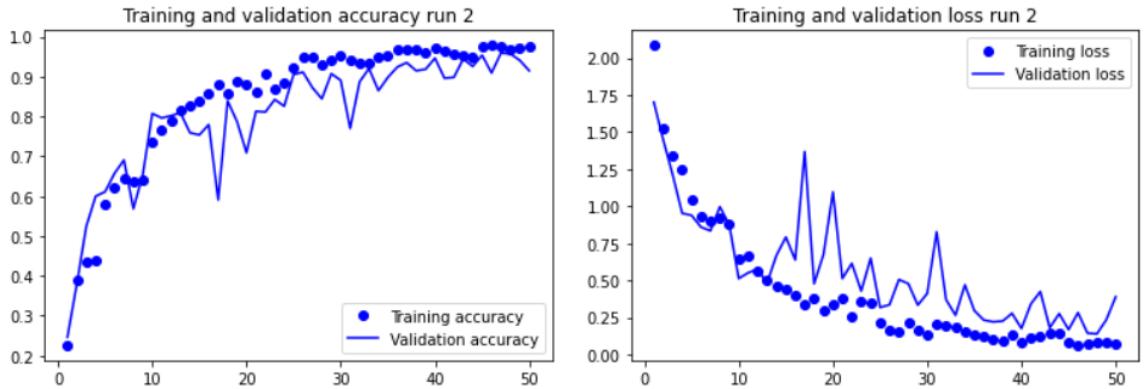
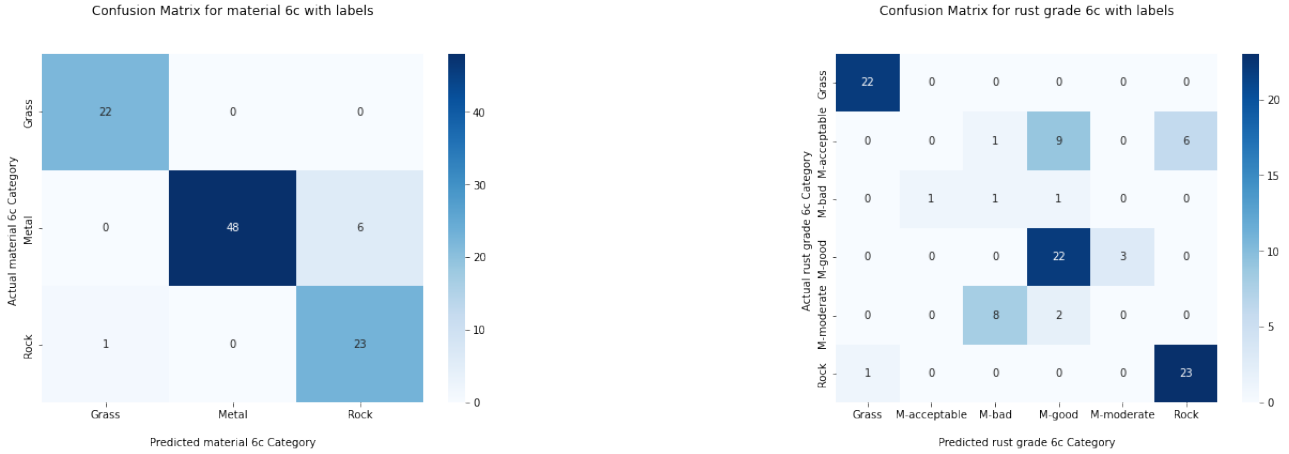


Figure 35: Accuracy and loss metrics for 6 class classifier.

In figure 35 we can see the accuracy and loss plot for the six-class classification. We can see that the accuracy is following the validation accuracy. From the loss function we notice some spikes and around epoch 30 we can see that the validation loss is barely decreasing.



(a) material

(b) rust grade

Figure 36: confusion matrix for six classes.

In figure 36 we see the confusion matrices for material and rust grade for the six-class model, the matrices convey the TP, FP, TN and FN in one matrix. The confusion matrix for material combines the "metal_b" and "metal_g" into one class, which is beneficial for us since the material column will decide what goes to the object detector algorithm. We can see that the material confusion matrix is performing a bit worse than the four-class model. When it comes to the rust grade, we can see that the model is performing poorly compared to the rust grade of the four-class model. This can be caused by the different rust grades classification that are vague and need help from an expert. It can also be due to the smaller data-set used, since we were at risk of creating an imbalanced data-set.

	Class	**Precision**	**Recall**	**F1**	
Material	Grass	1	0.96	0.98	
	Metal	1	0.8	0.89	
	Rock	0.67	1	0.8	
	Accuracy				0.88
Rust grade	Grass	1	0.95	0.98	
	M-acceptable	0	0	0	
	M-bad	0.25	1	0.4	
	M-good	0.77	0.96	0.86	
	M-moderate	0	0	0	
	Rock	0.67	1	0.8	
	Accuracy				0.72

Table 10: Metrics for material and Rust grade of six classes.

In table 10 we can see some important metrics. First the accuracy for material and rust grade are much lower than the four-class model. There are two classes, M-acceptable and M-moderate, in which the model failed to even predict. We also notice the low score for the precision and the rust-grade. This could be result for the sparse amount of images in the M-bad class. In short, we can conclude that adding more classes and not accounting for the data-set is a problem.

Figure 37: Transition into metal sheet section in HighRes_10556.

The models still classified some images with the wrong label. Both models have classified image "Highres_10556" (figure 37) as rock, while part of it is piling sheet. This may have happened due to the training set not containing images with multiple materials. The six-class model has made some errors when differentiating between metal and rock. In figure 38 we see the images that have been labelled rock, while the ground truth is metal. There are more of these errors when comparing the appendices F and D. This would mean that less images are sent to the object detection algorithm.

(a) Predicted: Rock; True:metal

(b) Predicted: Rock; True:metal

Figure 38: Wrong classified examples by the six-class model.

The results of the calculations done by the object detector can be seen in appendix F. The image "HighRes_10556" was not classified as metal and thus did not went through the object detector. The actual detected bounding boxes can be seen in appendix E. In figure 39 we see the result of image "HighRes_25584" in which all the objects have been detected and the height was estimated as 56.1 cm and the distance between bumps as 85.5 cm. Currently, it is not possible to measure the actual dimensions, but the values seem believable according to the experts of W+B. This does mean that object detection with a reference can provide geometrical data of an asset. Computer vision based sensing might be useful as input for digital twins (Reja et al., 2022).

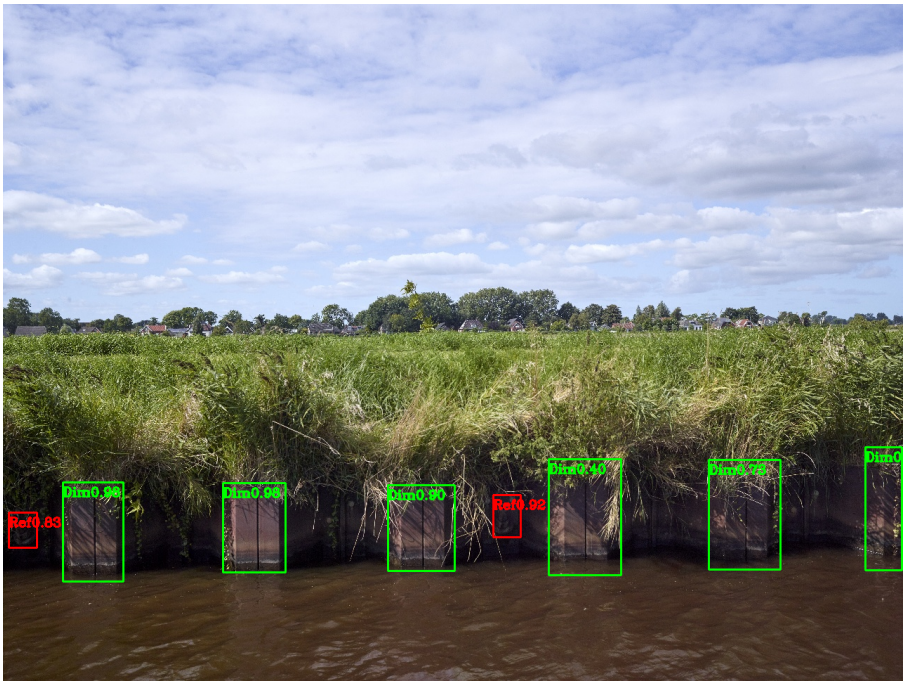


Figure 39: Result of image HighRes_25584 from object detection.

Model confidence threshold:25	
Precision	0.84
Recall	0.91
F1-score	0.88
TP	768
FP	144
FN	72
Avg IoU	66.40%
Mean AP@0.50	79.23%

Table 11: Model with confidence threshold 25%.

Within Darknet we can check the metrics of our YOLOv4 model. Table 11 shows the metrics if our model. We see the average IoU sits around 66.40%, this means the predicted bounding box is close to the actual object. The average precision of our model sits around 79.23%. In table 12 we can see some more class specific metrics. We see that the class 'Dim' has a higher precision than class 'Ref'. This was expected as every image used as training material had the multiple bumps present, while the presence of the reference object was scarce and few per image.

Class ID	Name	Average precision (%)	TP	FP
0	Dim	96.98	684	131
1	Ref	62.88	84	16

Table 12: Findings per class.

In appendix F we can see the results of both the classifier and the object detection. We can see that the object detection will provide numerical values for the height and average distance between bumps.

5.2.2 Time, cost and risk analysis

We can compare the time it takes to assess the images by the algorithm with the time employees at W+B would take. The assessment of HLD was completed in four weeks with four employees from W+B, so roughly $20 days * 8 hours/day = 160 hours$. The test subject of 100 images has been completed by the model within 77 seconds (21 seconds for classification and 56 for object detection). If we extrapolate this result, we can estimate

that the total time would be $40000/100 * 77 = 30800s \approx 8.6hours$. The inference times are different for the algorithm compiled on six classes. The inference times can be seen in table 13. These results are achieved on a laptop with the following specs: CPU: Core i5 8265U 1.6 GHz, GPU: UHD Graphics 620, Memory: 8GB RAM.

	4 class algorithm	**6 class algorithm**
Classification	21s	32.3s
Object detection	56s	75s
Total time on 100 images	77s	107.3s
Expected on 40k images	8.6h	11.9h

Table 13: Expected inference times.

It should be noted that they also had other tasks to complete, so it could very well be that one person is able to label one image in 10-20 seconds. Not all the images were assessed, but sections of the fairway. A person would normally not do such a task for longer periods without stopping. This already gives automation an advantage since the algorithm can run without needing a break; however, we should also consider the time that was spent writing and making the data-set. The author has finished studying and writing the classification algorithm in a month, but the creation of the data-set could take longer. An experienced programmer could write classifier/detector in less time or even re-use existing algorithms, however the labelling of data would still take time and is also dependant on how well he or she would want the classifier or detector to work.

When looking at the cost we could roughly estimate the combined salaries over a month versus the cost of compiling such an algorithm. Depending on the future policy of AI within W+B, they could either decide to invest on a workstation or cloud computing service. We worked with the free version of Google Colab, which uses an NVIDIA Tesla T4. The limitations of the free version are that we cannot use the GPU for longer periods and memory resources are limited. The price for physical NVIDIA Tesla T4 is around USD 1670,00 on Amazon.com; note there are hidden costs such as desktop casing, suitable motherboards and maintenance over time. The other option is using Microsoft Azure which has the following rates of USD 0,651-5,70 per hour for a virtual machine with a NVIDIA Tesla T4. Microsoft Azure also has other GPU's with higher prices. Hidden costs for Microsoft Azure would be getting familiar with Azure self and cloud storage for the training data. The total hours spent training the classifier and object detector is now sitting around 11 hours; considering this with the salary of one or two person. The creation of a data-set is the most excruciating part in this process, as we do need a person to label the files. If we assume that the data is already available then, we will still need a lot of time to create a good data-set. If it concerns a new object then of course cost will be made to generate data, which could drive the cost up again. there are however various ways to generate crowd sourced materials.

Since the model reached has an error rate of 17%, it means there is a 17% chance we would wrongfully consider the piling sheet to function longer or shorter. A qualitative risk analysis can identify the possible outcomes. In the case of predicting a longer live while it is not, the risks would involve failure of the construction with secondary damages to the land around it. This would also affect the reputation of W+B and result to other fines. In other words, it might not be beneficial to completely rely on this model if the costs of the risks would be of titanic proportions. Figure 40 shows these events, unfortunately it is now not possible to make cost analysis as the risks are not completely understood in terms of financial repercussions.

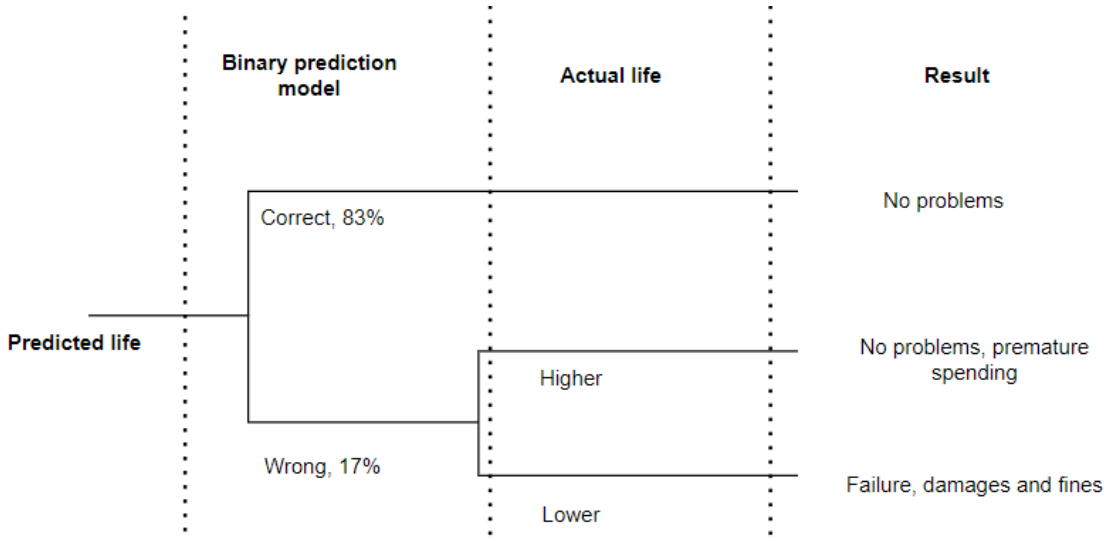


Figure 40: An event diagram for classifier of six classes.

5.2.3 Comparable projects

There are several other classification and object detection tasks we can compare our project with. A similar classification task to ours, is corrosion detection on oil and gas pipelines (Bastian, N, Ranjith, & Jiji, 2019). In that project they trained their classifier with four different classes of corrosion levels. We can see in figure 41 the different classes and that it is quite different than the one we used. Ours is on piling sheets which already have some corrosion present, and is based on DIGIGIDS2019. Their classifier has an accuracy of 98.2%, but they also have a data-set of 102800 images. Given that our data-set has a total of around 2000 images, an accuracy of 95% is not bad.

(a) *No corrosion* (b) *Low-level corrosion* (c) *Medium-level corrosion* (d) *High-level corrosion*

Figure 41: Classes from Bastian et al (2019).

Another similar research was done on weathering steel with a classification algorithm of four classes (Wang, Shen, Wu, & Huang, 2022). A data-set was created by spraying plates with salt and divided into the four classes (see fig 42). They have created a CNN that is similar to VGG16 in structure. They have changed the learning rate and batch size to get to an optimized model. Their model achieved an accuracy of 90.96% on their test set. The biggest difference to ours is that our data-set consist of images that are in a natural environment. We also have classes that are more vaguer to classify and our focus is on different structures and depth of a model.

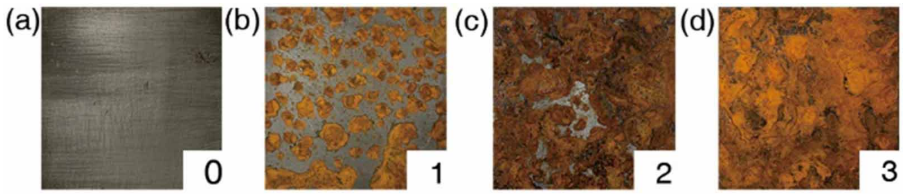


Figure 42: Classes from Wang et al (2022).

We can compare our object detector with a project that tried to detect railroad components (Guo, Qian, & Shi,

2021). Their object detection model was trained to detect three classes: spike, clips and rail. They have used several versions of YOLO and compared it to other detectors. Their mAP sits around 94.4% and the IoU sits around 87.4%. Their model was trained on a total of 1000 images. They concluded that an improved version of YOLOv4

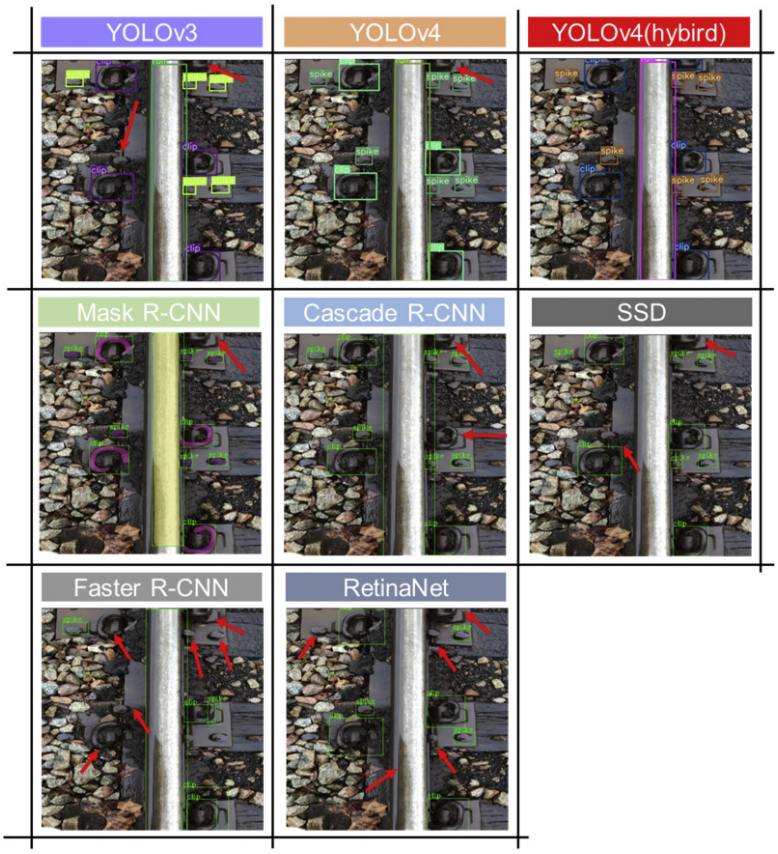


Figure 43: Detection using different detectors from Guo et al (2021).

Another use case of YOLOv4 is detecting faults in the airplane section (Zhang, Wei, Qi, & Wang, 2022). In their YOLO model a few changes were made in the architecture itself. The main goal was to detect scratches, gaps and cracks. Their model has reached a mAP of 96.41%. They did have a data-set of 3000 labelled images and a DL workstation. They also used the base YOLOv4 model which resulted in a lower mAP of 90.70%. The original YOLOv4 still has a faster inference time.

We have found another article using YOLOv4 to detect vehicles and estimate the distance from the camera (Qiao & Zulkernine, 2020). They have adjusted the algorithm too and used the Microsoft Common Objects in Context (MSCOCO) data-set. The MSCOCO data-set has around 300.000 images of which 200.000 are labelled (Lin et al., 2014). The detector of Qiao and Zulkernin can detect the cars with mAP of 99.16%, they however have not mentioned their IoU value. The distance estimation they will use is also different as they will first try to find the vanishing point between lanes and using the standard distance between the lanes, a conversion is made to metric units. Their mAP is not so far from us considering our data-set for object detection is just around 300 images.

Paper	Topic	Type	Accuracy (%)	Data-set size	Classes	mAP (%)	IoU (%)
Bastian et al., 2019	corrosion on steel pipes	Image classification	98.2	102800	4	-	-
Wang et al., 2022	corrosion on weatherin sheet	Image classification	90.98	1864	4	-	-
Guo et al., 2021	rail road components	Object detection	-	1000	3	94.4	87.4
Zhang et al., 2022	faults on aircraft tube	Object detection	-	3000	3	90.7	-
Qiao and Zulkernin, 2020	vehicle distance	Object detection	-	MSCOCO	1	99.16	-

Table 14: Comparison of similar research projects.

5.2.4 Validation of the project

We validate this research by asking qualitative questions from several experts at W+B. The experts are all involved in the HLD project and have been shown how the model is working. There are three experts who were willing to see the algorithm and after the presentation a discussion ensued based on two questions. The experts are familiar with the company policy, future projects and other disciplines. They all have no prior experience in working with computer vision based Deep Learning. The questions will concern the applicability and the feasibility of the algorithm.

We first explored the applicability by asking the question: *How well is the model working?* The experts are shown how the algorithm will present the predictions to them. This question will consider the predictions made by the model and how close it is to their own estimation. All the experts find the end result easily understandable. It is unfortunate that the six-class classifier is not working perfectly, but the four-classes model can still provide assistance. They all agree that just assessing the corrosion using only an image is not enough. In most cases further research would be done. It does however assist by automating an arduous process of going through 40000 images. The object detection is a great way to quickly estimate the dimensions, but the dimensions should not be used for any stress calculations. The estimation of the profile sheets does not seem to be highly needed at the moment, but does provide additional information. OD is also good for counting the amount of times the object is present.

To tackle the feasibility aspect, the question asks is: *When will this algorithm be part of a standard routine within W+B?* Currently, this novel visual inspection AI is not the standard practice. Knowing what needs to be done in order to be of a proper use for W+B. The experts all think that AI should definitely make progress within construction and testing these novel techniques provides a way to gain more insight. For now it should be enough if assisting someone is enough. In case it should do the more complex task, like the four classifications of corrosion, then a policy should be build around getting good data. Even then it would merely act as assistance to the expert who is tasked with assessing the piling sheet. There are several other projects in which these techniques could be used within W+B which is a motive to improve these techniques. A classification algorithm can be used in other projects which is supposed to detect humans or rust on bridges. The object detection can also be used in a way to measure how extensive a wall is sloping.

5.3 Summary of results and analysis

In this chapter we showed the size of the data-sets, the results from both the classification and object detection algorithms and the evaluation techniques used. We presented the data-set created for both algorithms to show they are balanced. A few images from the classification and object detection algorithms are shown. The full result can be seen in the appendixes. The research was then evaluated by verifying the metrics of our algorithms by comparing them to the requirements. We also made a brief time-cost analysis on the algorithm, which could serve as incentives to develop such an algorithm. The metrics of our algorithm have also been compared to other similar research projects and seen that our algorithm is showing a similar performance. Lastly, we validated the research with the help from insights from experts at W+B.

6 Conclusions and recommendations

This chapter will present a summary of the research and how the development statement fares. Recommendations are then made for possible future research. The author then ends with a reflection on his experiences with this topic.

6.1 Conclusions

This researched was aimed at understanding how DL based algorithms could help in assessing the corrosion damage on piling sheet and also get dimensions from only images. We conducted a literature search and concluded the application of existing algorithms on piling sheet was new. From there we also decided that image classification and object detection were the best types of algorithms to deal with our task. The classification would assess the piling sheets and detect whether there is a piling sheet and if there is to what grade it belongs to according to the *Digigids* site. The object detector would be trained to detect features on the piling sheet to estimate the actual dimensions. We also searched what metrics our algorithm had to adhere to given our data-set. We then started to create data-sets for classification with four classes, six classes and an object detection set. Alongside that process we started to write the classification and object detection algorithm. We also set the evaluation metrics to see how the model can perform. Both the data-set and algorithms would continue to improve until a satisfactory performance was achieved. We then validated the data through expert review to assess the prospects of this algorithm in w+B.

The development statement was: *Develop a tool using computer vision techniques to reliably detect problematic corrosion on piling sheet within 4-5 months to understand what the state is of this topic for asset managers.* First, the classification algorithm has proved to be helpful in assessing the condition of the piling sheets. Both classification data-sets yielded networks that were capable to detect a piling sheet, the challenge however lies in predicting the right assessment score according to Digigids2019. The algorithm with six classes had difficulty in predicting between the corrosion classes. This could be attributed as the assessments from W+B are also supplemented by the point-cloud and as humans we can understand an image with some more context. The algorithm with four classes, where two scores were combined to create "metal bad" and "metal good", proved to more accurate. This is enough for W+B as humans are still needed in the process, but it would assist them to process this task faster. The algorithm is also set in a way that it could easily be used to train other types of objects like columns or people, which provides the opportunity to implement the classification algorithm in a progress or safety monitoring task. We can conclude that the amount of classes should carefully be considered against the availability of the data and how different the classes are. The different structure compilations tell us that the structure is an important aspect to consider when building neural networks.

The object detection algorithm is a good tool to estimate the size of any features, given a reference object is present. Our algorithm can detect the "bumps" in the piling sheet with a satisfactory IoU value. The reference object was occasionally missed, but this should not be a problem in case this algorithm would go through images that were made in a sequence after each other. The exact dimension however was a rough estimation as was expected. The result could not be as accurate and precise compared to using tape measure. Training the object detection model was also inconvenient using Google Colab, as we had to take measure against being disconnected. Retraining for other kinds of objects would be time consuming, but is still doable and helpful in case W+B would want to estimate other dimensions in the field. A possible use case would be to detect people and if they are not in range of heavy equipment.

6.2 Recommendations

Based on this research we recommend the following points to improve upon this algorithm:

- Keep expanding the data-set for classification task. During the creation of the data-set we were limited to time constraint and to the HLD project. As a result, it was harder to get satisfying results for the algorithm for 6 classes. Many other research contain around 1000 images per class without augmentation, this is stark difference with our data-set containing 300 and 600 images for the six and four class respectively.

- Expand the amount of classes for the classification. Expand the amount of classes for the classification. The raw of w+b contained images of other types of materials like wood and concrete. The current network can be further trained using “transfer learning”. This would increase the usability of the network. Transfer learning concern in using a pre-trained model and train it again on new objects. CITE

- Use a dedicated DL workstation when training object detection models. The computing time was of great concern to us. Using Google Colab meant we had a daily limit of 6h with constant being present for 30 min. The object detection model contained the most training time.
- Detect other parameters that could help in assessing the health of the piling sheet. In this one reference object can be connected to the NAP and the other can be a point on the piling sheet. The vertical subsidence is one of the parameters used in other structural life monitoring system CITE. We can also use the object detector to detect specific corrosion features, such as galvanic corrosion, waterline corrosion, crevice corrosion etc. This way we could quantify the different types of corrosion present.
- A robust research on the feasibility. Cost of operating, maintenance, risk management. This could help with the decision on whether to use this novel technique. We have created a rough estimate of the direct cost and some risks, but it could be a separate research on identifying more risks and indirect cost. We did not have access to the contract and other possible clauses made. A risk register should contain the risk perceived from different specialist.
- Explore data generation through crowd sourcing. A research paper on corrosion detection used images that were acquired by using public data. A person could take any amount of images of a corroded object and upload it to their repository. Another way is also scraping the web for publicly available images, however some care should be taken if they are copyrighted. In W+B the first can be applied by several employees, around several office locations. It could also be joint venture between multiple firms for which this would be relevant. Scraping the web for any copyright free images might also help, however it should always be checked if it is within the same category.
- YOLOv7 has just released. The authors of YOLOv4 have released an updated version that showed higher accuracy and faster results.
- Explore applying object detection in progress monitoring. Since we can detect anchors, it can easily be used to detect the amount of anchors in a new project. The algorithm can be repurposed to detect and count other construction objects in which the amount can tell us something about the progress.
- The interoperability between a computer vision-based DL and other software can be further explored. Preferably software that the manager would use.

6.3 Reflection

This thesis has provided a lot of insight on and around AI for the author. The curriculum for Construction Management and Engineering has three main branches such as "Projects and people", "Design and integration" and "Engineering and systems". Of those three it is "Engineering and systems" provide the most topics regarding automation within construction. At first this project seemed like going through a tunnel and seeing the light at the end, but the end kept getting further as there were still a lot of self-study necessary. This was a fun process as the author gained knowledge on what neural networks are, how they are made, what different types there are and how to create good networks. The author of "deep learning with python", Francois Chollet, has mentioned that creating neural networks is an art. An example is baking a cake; you can get the ingredients and some instruction, but you will have to make a lot of cakes and experiment with what works to eventually make great cakes. The same thing can be written about neural network as you have all the tools and layers, but only by experimenting will you get a feel on creating good networks. Other insights gained around AI were Societal and ethical issues.

During the process of learning AI the author had joined various communities with both positive and negative sentiments towards AI. The communities with positive sentiments were usually in a computer science field in which various peoples with different backgrounds would be present. There were people who talked about the structures of an ANN and various project they were working on or problems they encountered. The community with negative sentiments were of a civil engineering. Most of the negative sentiments were "AI will take our jobs", "AI can never achieve the same quality as humans", "AI is a black box so don't use it" or "AI has ethical concerns". In the authors eyes all these sentiments have some truth, but that does not mean it is a bad thing. One such example is the "Roomba". an automatic vacuuming device, which needs someone to maintain that device. The best solution is a coexisting system with each other by utilizing what AI is good at (automation) and complementing it with a human (emotion).

The different views on AI might need a process approach when searching for a solution. The author was

reminded of a lecture in Process management in which the professor asked students what they would do to prove a point. After they answered, he would say that the opposing party would try their equal best to prove their opposing point. This would also happen when discussing AI topics and at one point the author did not feel like discussing it any further. The author believes this problem is a "wicked problem" and a process approach would work better in which we identify the actors and explore their core values. At the end of such a process it should be made sure that every actor is satisfied with the results. This is a management way to approach any AI venture within an organisation as AI will most likely keep developing as time passes.

Ethical issues will always be a concern and thus responsible innovation is needed. On various internet media and during the process of learning AI, it was clear that the possibility of misusing AI is present. One person for example, asked in the community for help to detect *captcha* images and how to work around it. Luckily most of the members condemned such a project as it was clearly intended to bypass website security. The creator of the YOLO algorithm, Joseph Redmond, had stated that he is dropping his research due to the inevitable use of this subject by the military. So even if someone has positive sentiments towards AI, their intentions regarding it might be of malicious intent.

Learning DL was a challenge in itself. Due to the popularity of AI many media are now available, with varying degree of quality. Going through the abundance of online videos, online courses and books can be daunting to someone dipping their toes in this field. The field is overlapping with other fields such as data science, computer vision, time series, natural language processing etc. All of them require some prior knowledge of that field itself and mathematics such as calculus, algebra and statistics. The author has tried tackling this by adding them as extra courses in his curriculum, but only some parts of a course were useful for this thesis.

In case this process would be done over again or for anyone who would like to enter this exciting field, the following tips would be given:

- Research the most popular libraries and who contributed to them.
- People who contributed to a library usually made a book or have online courses. For the author a book from Francoise Chollet (made Keras) and the "Datascience handbook" from Jake van der Plas (contributed in Scikit-learn) were helpful.
- Consider the Machine learning workflow. This means knowing who the actors are, what the data is or if it needs to be created.
- Join various online forums and communities to meet people in similar fields.

References

Alaloul, W. S., Qureshi, A. H., Musarat, M. A., & Saad, S. (2021). Evolution of close-range detection and data acquisition technologies towards automation in construction progress monitoring. *Journal of Building Engineering*, *43*, 102877. Retrieved from https://www.sciencedirect.com/science/article/pii/S235271022100735X doi: https://doi.org/10.1016/j.jobe.2021.102877

Basodi, S., Ji, C., Zhang, H., & Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, *3*(3), 196-207. doi: 10.26599/BDMA.2020.9020004

Bastian, B. T., N, J., Ranjith, S. K., & Jiji, C. (2019). Visual inspection and characterization of external corrosion in pipelines using deep neural network. *NDT & E International*, *107*, 102134. Retrieved from https://www.sciencedirect.com/science/article/pii/S096386951930060X doi: https://doi.org/10.1016/j.ndteint.2019.102134

Baxter, D., Hechler, O., Martins, J., Weber, E., Werner, P.-N., White, G., ... Zuck, F. (2016). 1 product information. In *Piling handbook, ninth edition.* ArcelorMittal Commercial RPS.

Binnekamp, D. I. R. (2020). *Cie 4030 methodology for scientific research cme addendum.*

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *Yolov4: Optimal speed and accuracy of object detection.* arXiv. Retrieved from https://arxiv.org/abs/2004.10934 doi: 10.48550/ARXIV.2004.10934

Borges Oliveira, D. A., Ribeiro Pereira, L. G., Bresolin, T., Pontes Ferreira, R. E., & Reboucas Dorea, J. R. (2021). A review of deep learning algorithms for computer vision systems in livestock. *Livestock Science*, *253*, 104700. Retrieved from https://www.sciencedirect.com/science/article/pii/S1871141321003085 doi: https://doi.org/10.1016/j.livsci.2021.104700

Chai, X., Árpád Rózsás, Slobbe, A., & Teixeira, A. (2022). Probabilistic parameter estimation and reliability assessment of a simulated sheet pile wall system. *Computers and Geotechnics*, *142*, 104567. Retrieved from https://www.sciencedirect.com/science/article/pii/S0266352X21005450 doi: https://doi.org/10.1016/j.compgeo.2021.104567

Chollet, F. (2021). *Deep learning with python, second edition.* Manning. Retrieved from https://books.google.nl/books?id=XHpKEAAAQBAJ

Cicek, V. (2014).
In *Corrosion engineering.* John Wiley & Sons. Retrieved from https://app.knovel.com/hotlink/toc/id:kpCE00004B/corrosion-engineering/corrosion-engineering

diamondtraffic.com. (2022). *Diamond traffic products.* Retrieved 08-01-2021, from https://diamondtraffic.com/

Different forms of corrosion classified on the basis of appearence. (2004). In E. Bardal (Ed.), *Corrosion and protection* (pp. 89–191). London: Springer London. Retrieved from https://doi.org/10.1007/978-1-85233-845-9_7 doi: 10.1007/978-1-85233-845-9_7

Digigids.hetwatershapshuis.nl. (2022). *Digigids 2019.* Retrieved 10-06-2022, from https://digigids.hetwaterschapshuis.nl/index.php?album=Bijzondere-constructies-%282019%29/damwand%20of%20beschoeiing/conditie

Elnagga, S. M., & Elhegazy, H. (2021). Study the impact of the covid-19 pandemic on the construction industry in egypt. *Structures*. doi: https://doi.org/10.1016/j.istruc.2021.09.028

Forkan, A. R. M., Kang, Y.-B., Jayaraman, P. P., Liao, K., Kaul, R., Morgan, G., ... Sinha, S. (2022). Corrdetector: A framework for structural corrosion detection from drone images using ensemble deep learning. *Expert Systems with Applications*, *193*, 116461. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417421017437 doi: https://doi.org/10.1016/j.eswa.2021.116461

Guo, F., Qian, Y., & Shi, Y. (2021). Real-time railroad track components inspection based on the improved yolov4 framework. *Automation in Construction*, *125*, 103596. Retrieved from https://www.sciencedirect.com/science/article/pii/S0926580521000479 doi: https://doi.org/10.1016/j.autcon.2021.103596

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220-239. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417416307175 doi: https://doi.org/10.1016/j.eswa.2016.12.035

Hastings, N. A. J. (2015). Life cycle costing. In *Physical asset management: With an introduction to iso55000* (pp. 149–158). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-14777-2_8 doi: 10.1007/978-3-319-14777-2_8

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition.* arXiv. Retrieved from https://arxiv.org/abs/1512.03385 doi: 10.48550/ARXIV.1512.03385

Infra, C. B. . (2016). 6.15 inspectiemethoden. In *Cur 166 damwandconstructies – deel 2. 6° herziene druk.* CUR-publicatie.

Khallaf, R., & Khallaf, M. (2021). Classification and analysis of deep learning applications in construction: A systematic literature review. *Automation in Construction*, *129*, 103760. doi: https://doi.org/10.1016/j.autcon.2021.103760

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv. Retrieved from https://arxiv.org/abs/1412.6980 doi: 10.48550/ARXIV.1412.6980

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... Dollár, P. (2014). *Microsoft coco: Common objects in context.* arXiv. Retrieved from https://arxiv.org/abs/1405.0312 doi: 10.48550/ARXIV.1405.0312

Miao, P., & Srimahachota, T. (2021). Cost-effective system for detection and quantification of concrete surface cracks by combination of convolutional neural network and image processing techniques. *Construction and Building Materials*, *293*, 123549. Retrieved from https://www.sciencedirect.com/science/article/pii/S095006182101309X doi: https://doi.org/10.1016/j.conbuildmat.2021.123549

Mostafa, K., & Hegazy, T. (2021). Review of image-based analysis and applications in construction. *Automation in Construction*, *122*, 103516. Retrieved from https://www.sciencedirect.com/science/article/pii/S0926580520310967 doi: https://doi.org/10.1016/j.autcon.2020.103516

Munawar, H. S., Ullah, F., Shahzad, D., Heravi, A., Qayyum, S., & Akram, J. (2022). Civil infrastructure damage and corrosion detection: An application of machine learning. *Buildings*, *12*(2). Retrieved from https://www.mdpi.com/2075-5309/12/2/156 doi: 10.3390/buildings12020156

nen.nl. (2022). *Renovatie kanaal fase 1.* Retrieved 14-04-2022, from https://www.nen.nl/en/nen-2767-1-c1-2019-nl-256366

Nicholas, J., & Steyn, H. (2017). *Project management for engineering, business and technology,*. doi: 10.4324/9781315676319

Ongsulee, P., Chotchaung, V., Bamrungsi, E., & Rodcheewit, T. (2018). Big data, predictive analytics and machine learning. In *2018 16th international conference on ict and knowledge engineering (ict ke)* (p. 1-6). doi: 10.1109/ICTKE.2018.8612393

Paneru, S., & Jeelani, I. (2021). Computer vision applications in construction: Current state, opportunities & challenges. *Automation in Construction*, *132*, 103940. Retrieved from https://www.sciencedirect.com/science/article/pii/S0926580521003915 doi: https://doi.org/10.1016/j.autcon.2021.103940

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Qiao, D., & Zulkernine, F. (2020). Vision-based vehicle detection and distance estimation. In *2020 ieee symposium series on computational intelligence (ssci)* (p. 2836-2842). doi: 10.1109/SSCI47803.2020.9308364

Reddy, M. S. B., Ponnamma, D., Sadasivuni, K. K., Aich, S., Kailasa, S., Parangusan, H., ... Zarandah, R. (2021). Sensors in advancing the capabilities of corrosion detection: A review. *Sensors and Actuators A: Physical*, *332*, 113086. Retrieved from https://www.sciencedirect.com/science/article/pii/S0924424721005513 doi: https://doi.org/10.1016/j.sna.2021.113086

Redmon, J. (2013–2016). *Darknet: Open source neural networks in c.* http://pjreddie.com/darknet/.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You only look once: Unified, real-time object detection.* arXiv. Retrieved from https://arxiv.org/abs/1506.02640 doi: 10.48550/ARXIV.1506.02640

Reja, V. K., Varghese, K., & Ha, Q. P. (2022). Computer vision-based construction progress monitoring. *Automation in Construction*, *138*, 104245. Retrieved from https://www.sciencedirect.com/science/article/pii/S0926580522001182 doi: https://doi.org/10.1016/j.autcon.2022.104245

Rijkswaterstaat.nl. (2022). *Renovatie kanaal fase 1.* Retrieved 12-04-2022, from https://hld.rws.nl/renovatie+kanaal+fase+1_/waarom+-+vervanging+en+renovatie/default.aspx

Roberge, P. R. (2007a). 3.4 life cycle asset management. In *Corrosion inspection and monitoring.* John Wiley & Sons. Retrieved from https://app.knovel.com/hotlink/khtml/id:kt007WP8B1/corrosion-inspection/life-cycle-asset-management

Roberge, P. R. (2007b). 5.3 visual and enhanced visual inspection. In *Corrosion inspection and monitoring.* John Wiley & Sons. Retrieved from https://app.knovel.com/hotlink/khtml/id:kt007WPEA2/corrosion-inspection/visual-enhanced-visual

Schmidhuberaf, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85-117. doi: https://doi.org/10.1016/j.neunet.2014.09.003

surveyinghub.nl. (2022). *Lidar & 3d scanners categories.* Retrieved 08-01-2021, from https://surveyinghub.nl/lidar-3d-scanners/

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2014). *Going*

deeper with convolutions. arXiv. Retrieved from https://arxiv.org/abs/1409.4842 doi: 10.48550/ARXIV.1409.4842

Szeliski, R. (2011). Introduction. In *Computer vision: Algorithms and applications* (pp. 1–25). London: Springer London. Retrieved from https://doi.org/10.1007/978-1-84882-935-0_1 doi: 10.1007/978-1-84882-935-0_1

Tulbure, A.-A., Tulbure, A.-A., & Dulf, E.-H. (2022). A review on modern defect detection models using dcnns – deep convolutional neural networks. *Journal of Advanced Research*, *35*, 33-48. Retrieved from https://www.sciencedirect.com/science/article/pii/S2090123221000643 doi: https://doi.org/10.1016/j.jare.2021.03.015

twitter.com. (2020). *Joseph redmon.* Retrieved 10-01-2021, from https://twitter.com/pjreddie/status/1230524770350817280

Tzutalin. (2022). *Labelimg. git code.* Retrieved 22-04-2022, from https://github.com/tzutalin/labelImg

Verruijt, A. (2018). Sheet pile walls. In *An introduction to soil mechanics* (pp. 277–286). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-61185-3_35 doi: 10.1007/978-3-319-61185-3_35

Wang, Y., Shen, X., Wu, K., & Huang, M. (2022, jun). Corrosion grade recognition for weathering steel plate based on a convolutional neural network. *Measurement Science and Technology*, *33*(9), 095014. Retrieved from https://doi.org/10.1088/1361-6501/ac7034 doi: 10.1088/1361-6501/ac7034

witteveenbos.nl. (2022). *Over ons.* Retrieved 31-01-2022, from https://www.witteveenbos.com/nl/over-ons/

Wu, D., Lv, S., Jiang, M., & Song, H. (2020). Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments. *Computers and Electronics in Agriculture*, *178*, 105742. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168169920318986 doi: https://doi.org/10.1016/j.compag.2020.105742

www.bouwendnederland.nl. (2020). *Coronacrisis | levert het coronavirus overmacht op voor aannemers en andere betrokken partijen?* Retrieved 15-01-2021, from https://www.bouwendnederland.nl/actueel/nieuws/11272/coronacrisis-levert-het-coronavirus-overmacht-op-voor-aannemers-en-andere-betrokken-partijen

Yu, Z., Shen, Y., & Shen, C. (2021). A real-time detection approach for bridge cracks based on yolov4-fpm. *Automation in Construction*, *122*, 103514. Retrieved from https://www.sciencedirect.com/science/article/pii/S0926580520310943 doi: https://doi.org/10.1016/j.autcon.2020.103514

Zhang, J., Wei, S., Qi, M., & Wang, P. (2022, apr). Improved aircraft flared tube defect detection algorithm of YOLOv4 network structure. *Journal of Physics: Conference Series*, *2252*(1), 012050. Retrieved from https://doi.org/10.1088/1742-6596/2252/1/012050 doi: 10.1088/1742-6596/2252/1/012050

A List of abbreviations

Acronyms

AI Artificial Intelligence. 15, 50, 52, 53

ANN Artificial Neural Network. 16, 27, 52

API Application Programming Interface. 28, 43

BIM Building information Modeling. 8, 10, 11, 14

CNN Convolution Neural Network. 6, 12, 14–18, 23, 27, 29, 34, 35, 39, 59

CV Computer Vision. 8, 9, 16, 19, 20, 23

DL Deep Learning. 12, 16, 49–53

DNN Deep Neural Network. 37

EMK Eemskanaal. 8

FN False Negative. 26, 41, 44

FP False Positive. 26, 41, 44

GIS Geographic Information System. 8

GPU Graphical Processing Unit. 3, 27, 28, 34, 35, 39, 47

HLD Hoofdvaarweg Lemmer-Delfzijl. 8, 28, 46, 50, 51

HOG Histogram of Oriented Gradients. 16

IoU Intersection over Union. 26, 46, 49, 51

mAP mean Average Precision. 26, 49

ML Machine Learning. 15, 28

MSCOCO Microsoft Common Objects in Context. 49

MT Model Test. 34, 43

NDE nondestructive evaluation. 13

NEN Nederlandse Norm. 12, 42

ONNX Open Neural Network Exchange. 37

PMK Prinses Margrietkanaal. 8

SVM Support-vector Machine. 16

TN True Negative. 26, 44

TP True Positive. 26, 44

UML Unified Modelling Language. 20

VS Code Visual Studio Code. 27

VSK Van Starkenborghkanaal. 8

W+B Witteveen+Bos. 2, 8–10, 12, 13, 22, 23, 25–31, 34, 36, 42, 45–47, 50–52, 72

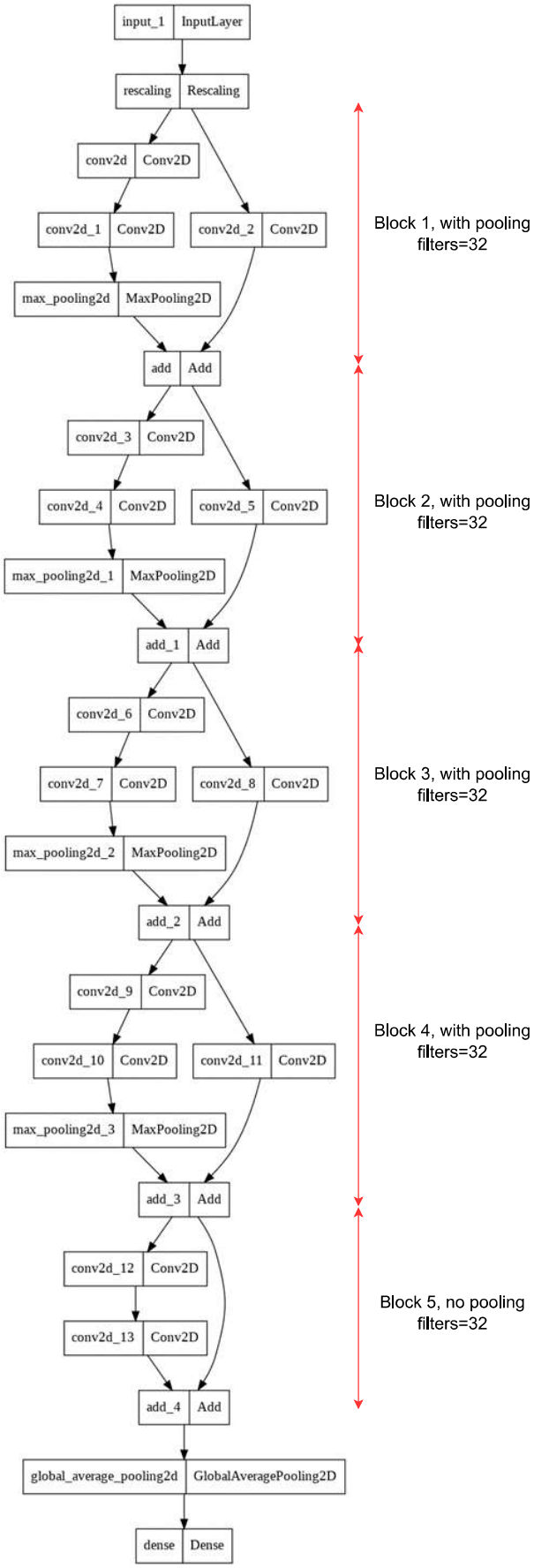
XML eXtensible Markup Language file. 32, 33, 37

YOLO You Only Look Once. 18, 26, 27, 32, 33, 37, 38, 46, 49, 53

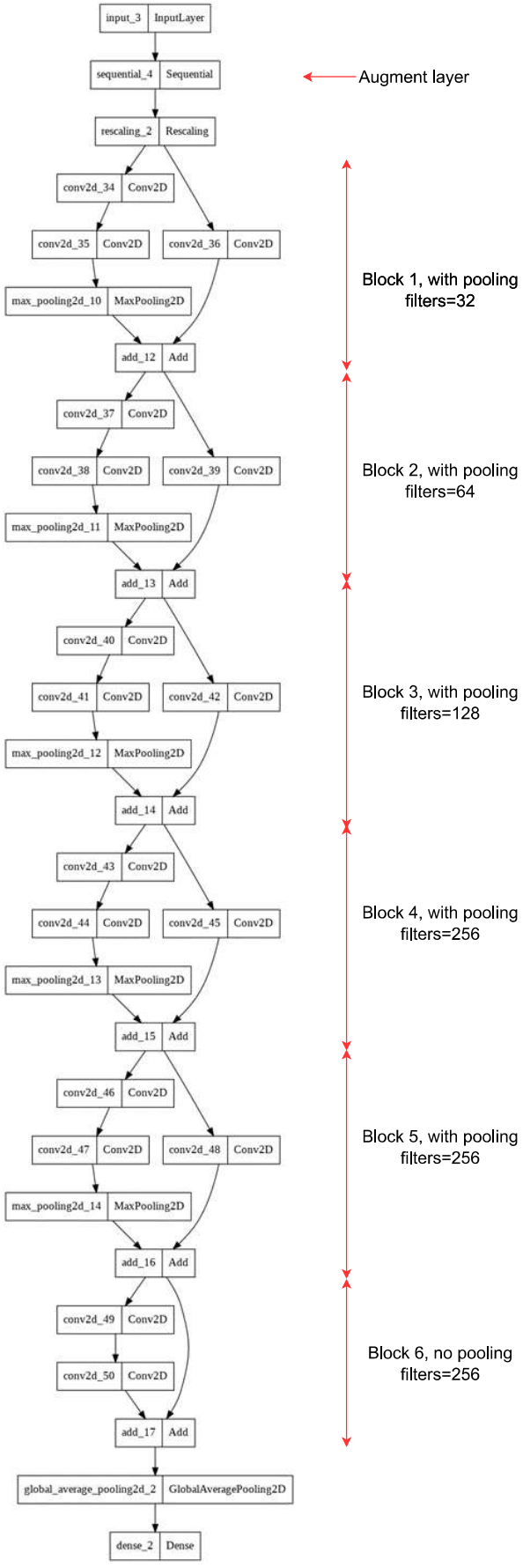
B Model plot

The plot below shows the structure of the CNN for the four classes and the six classes. Within the structure you can see the how a block is added several times. The models all have an input layer, rescaling layer, global average pooling layer and a dense layer. The dense layer contains the same amount of units as the different classes per model.

Four class model



Six class model



C Accuracy and Loss cross-validate

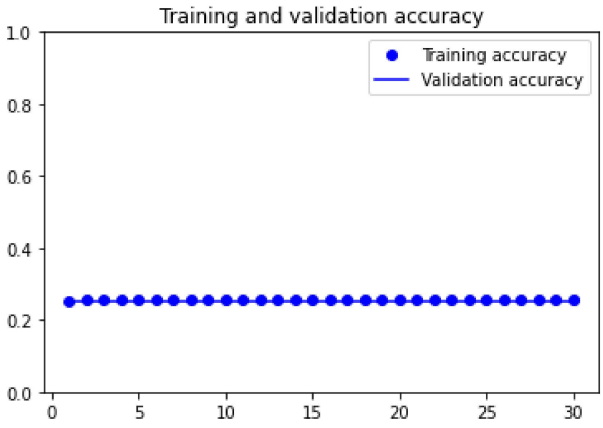
This appendix shows all the accuracy and loss diagrams. The four classes models are in the order presented in table 5. The same is true for the six classes models and follow table 6. The appendix ends with the cross validated accuracy and loss diagrams for the four classes and 6 classes models.

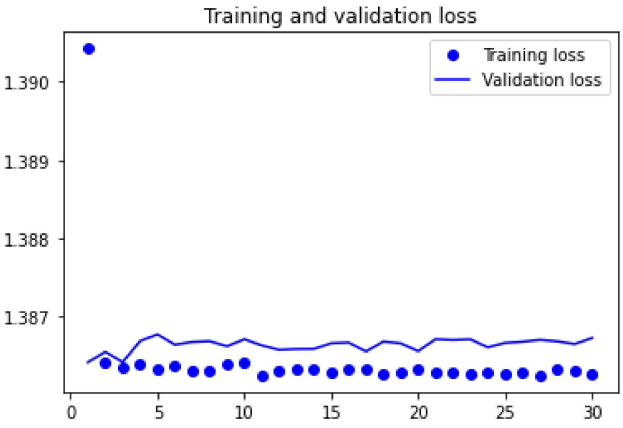
In each model we would write:

- Type. This could either be vgg16 like, resblock (residual connections) or inception like.
- Filters. This could be all are 32 or increasing.
- Augmentation. If augmentation is used or not.
- Blocks. The amount of blocks present in that model. It would sometimes be denoted as "4bl" (4 blocks) or "5bl" (5 blocks).

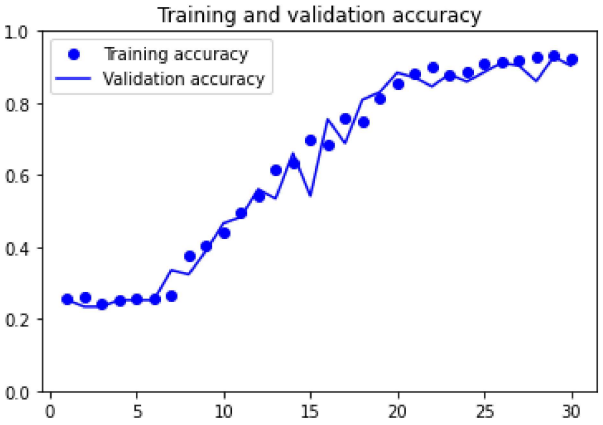
Four classes models

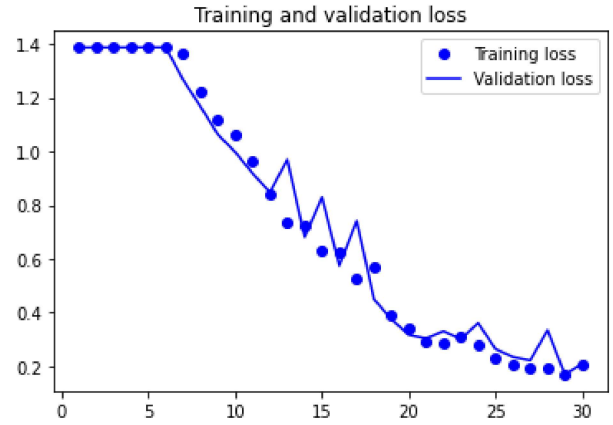
VGG16 5blocks all filter size=32-256, no augm, 2nd try



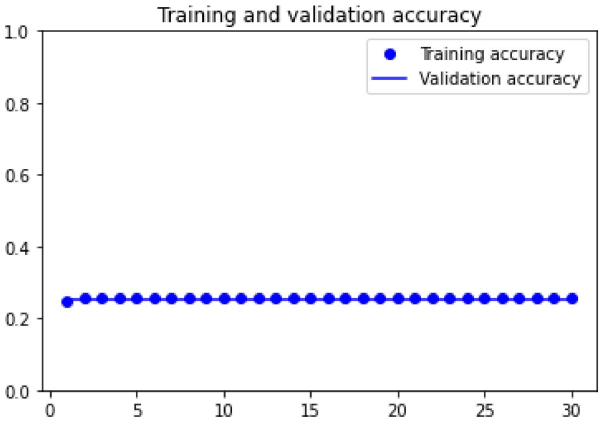


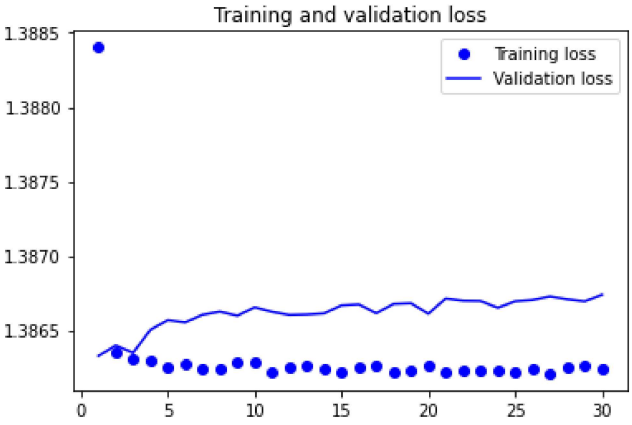
VGG16 5blocks all filter size=32, no augm, 2nd try



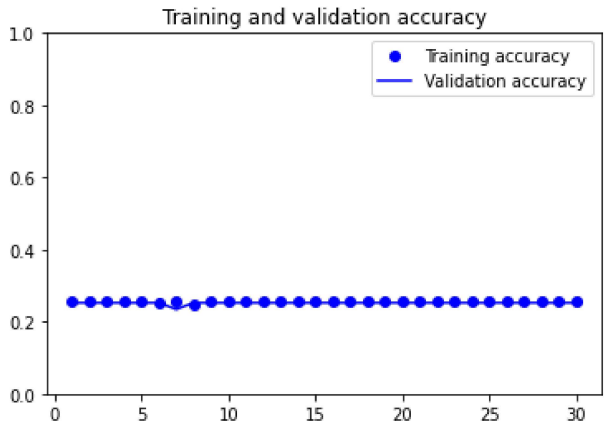


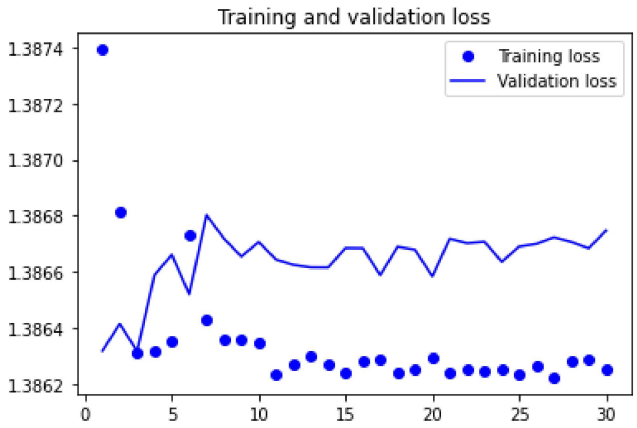
vgg16 5 blocks+augment+ep 30



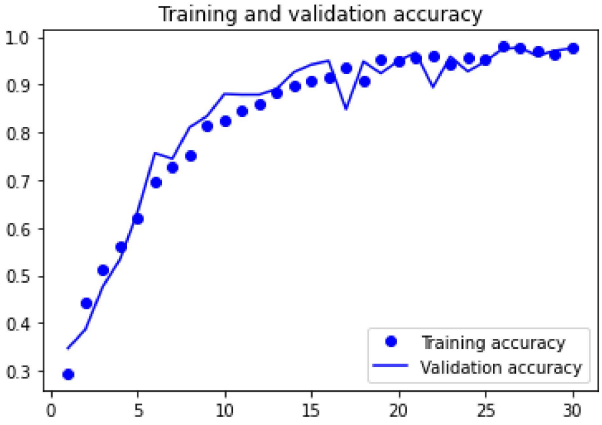


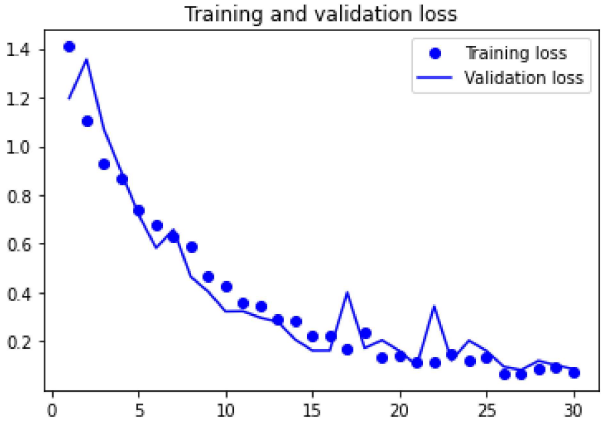
vgg16 4blocks filter size=32-256, yes augm



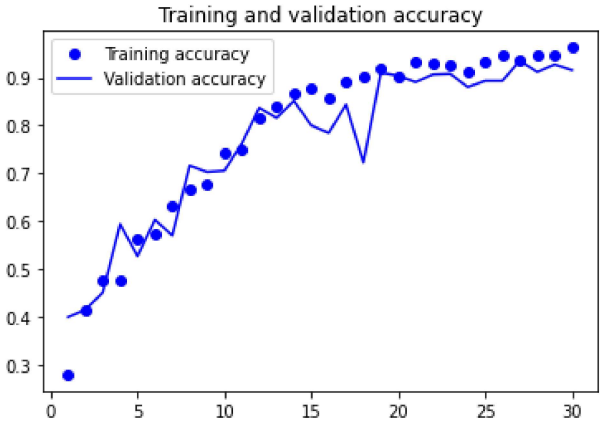


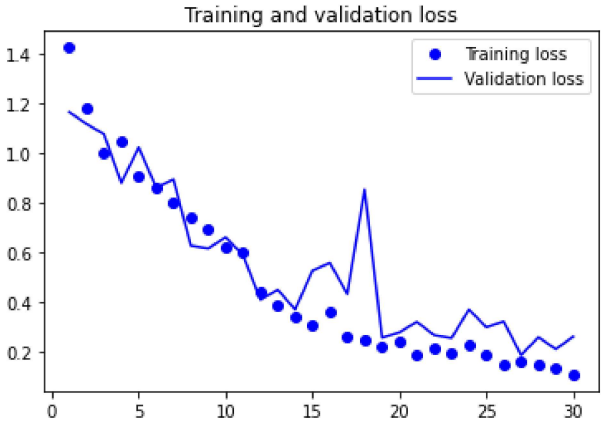
resblock 4 blocks, no augment+filter increas



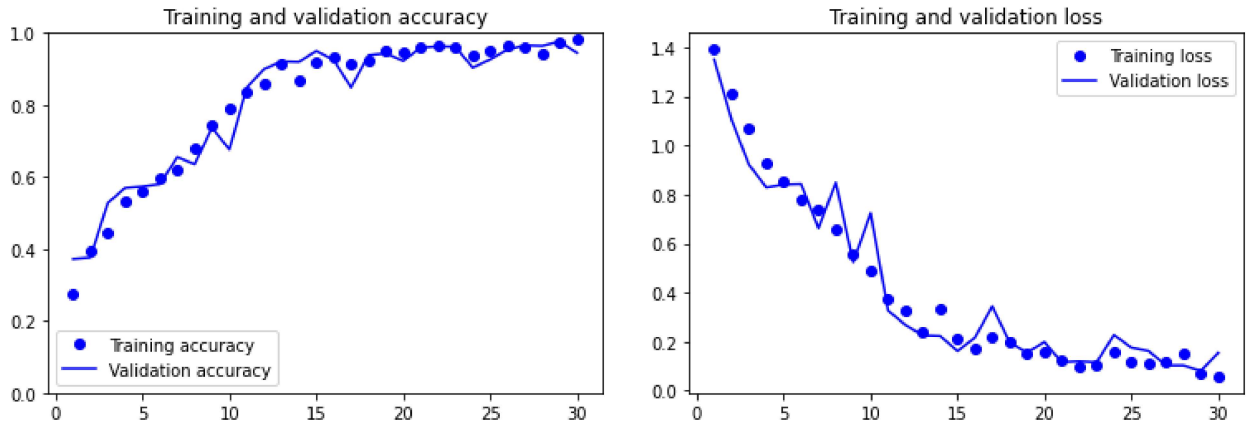


resblock 4 blocks+augment filt increase

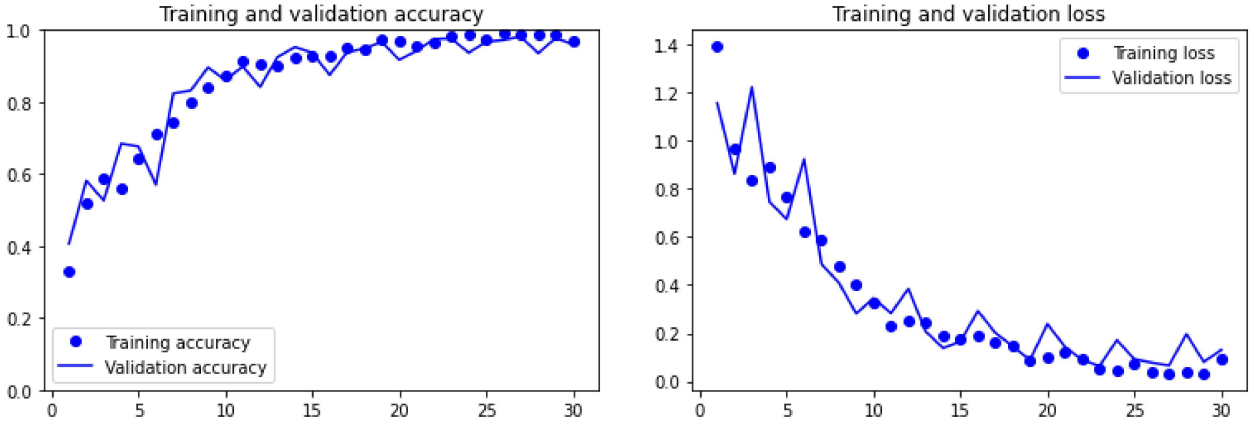




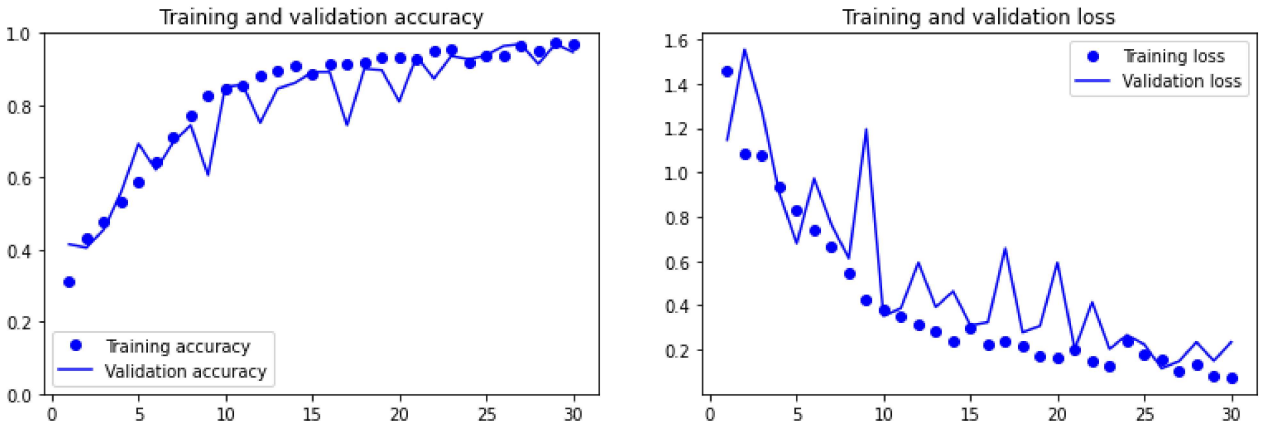
resblock5 blocks+ no augment+ep 30, filter=32



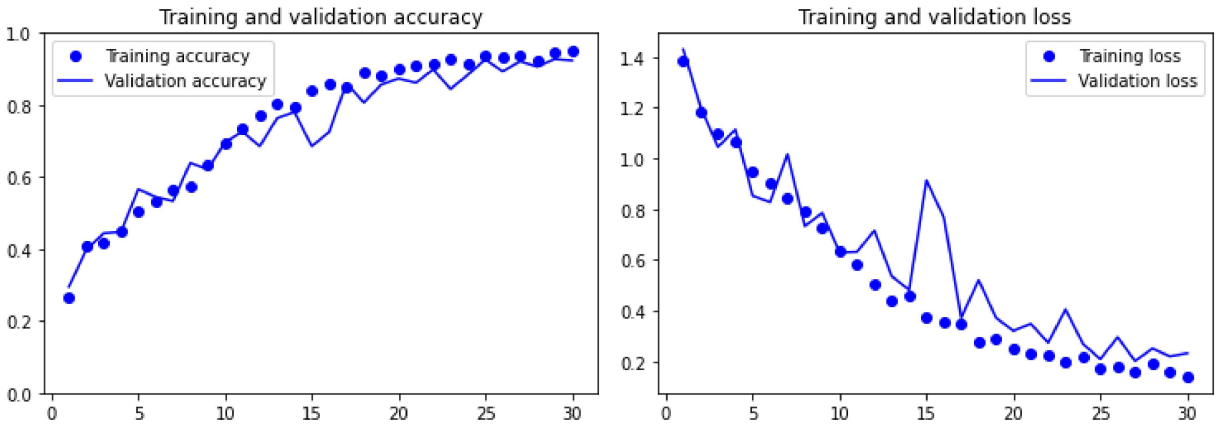
resblock5 blocks+ no augment+ep 30, filter increases



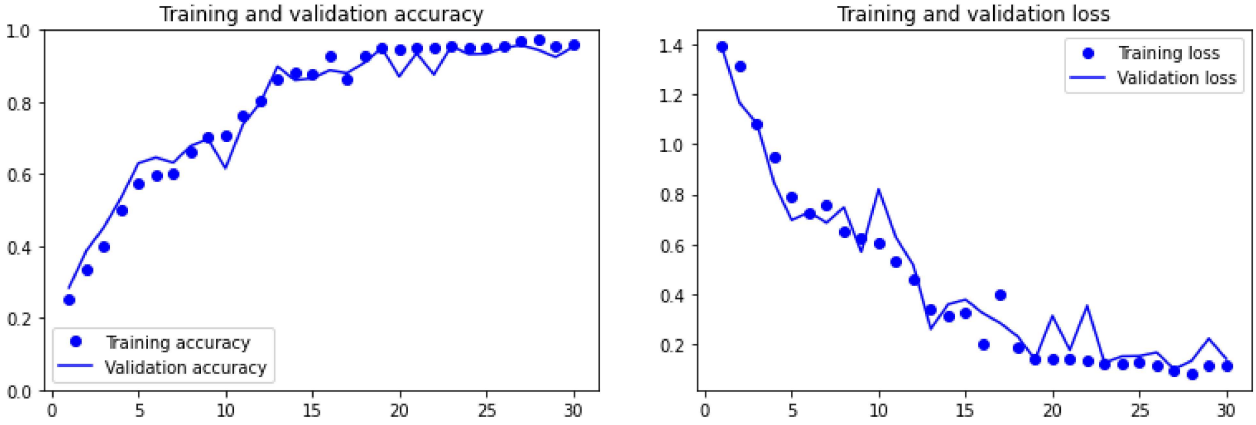
resblock5 blocks+ augment+ep 30, filter increases



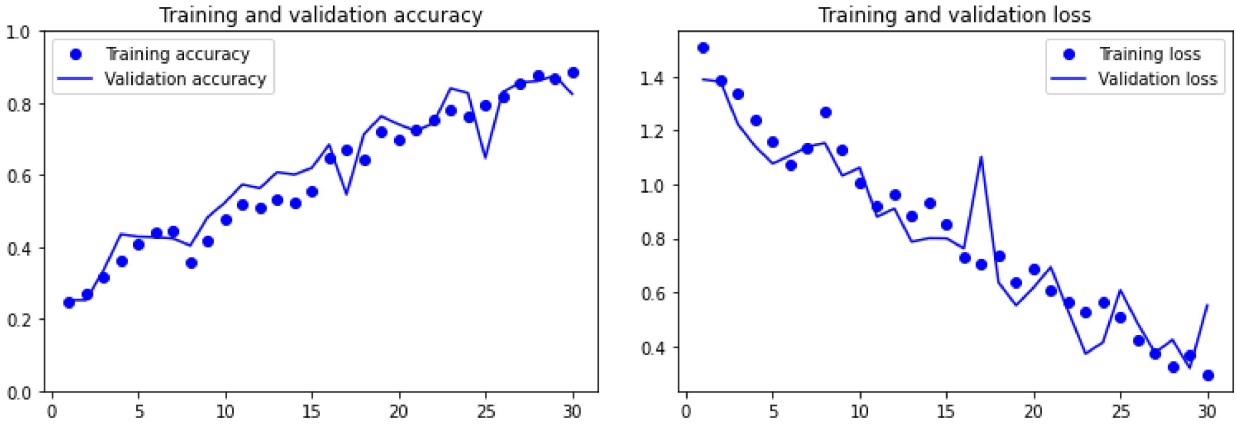
resblock 5 blocks filter all size=32, augm, ep30



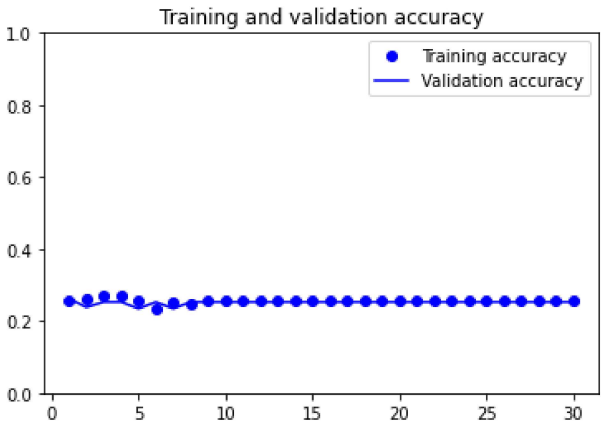
inception 5 blocks filter all size=32, no augm, ep30

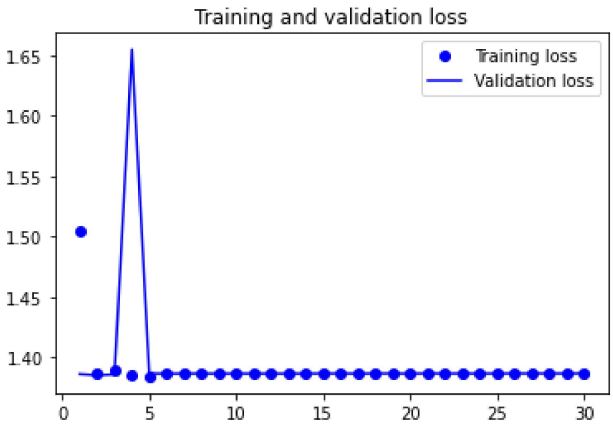


inception 5 blocks filter all size=32-256, augm, ep30

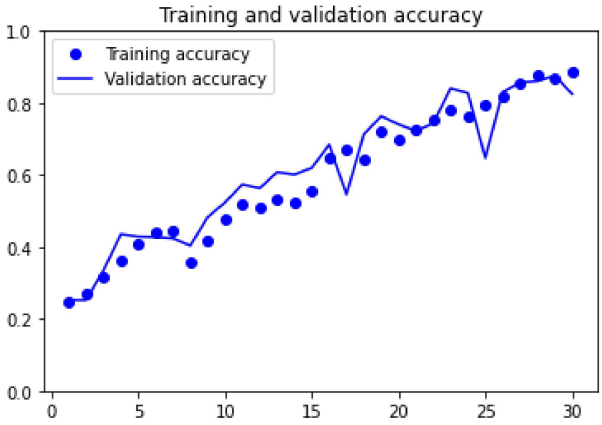


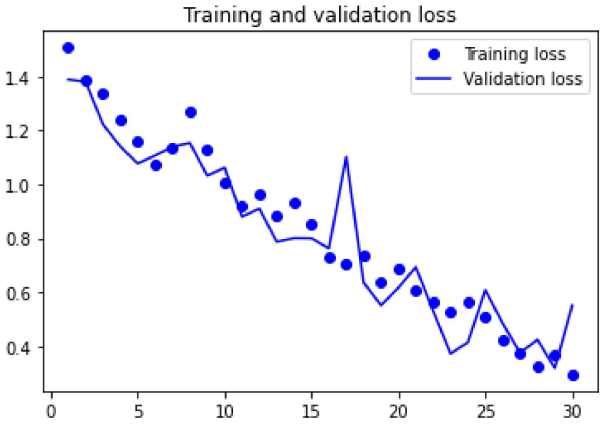
inception 5 blocks filter all size=32-256, no augm, ep30





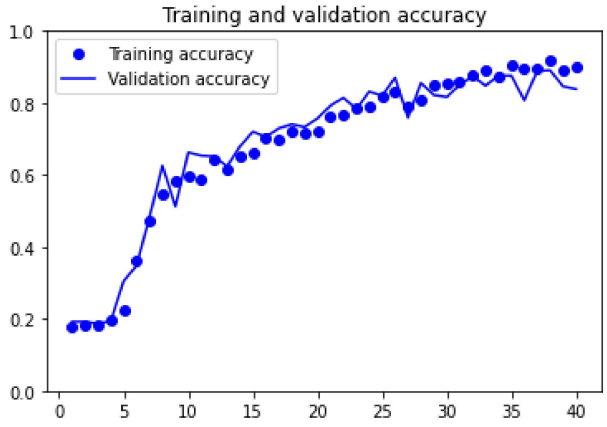
inception 5 blocks filter all size=32-256, augm, ep30

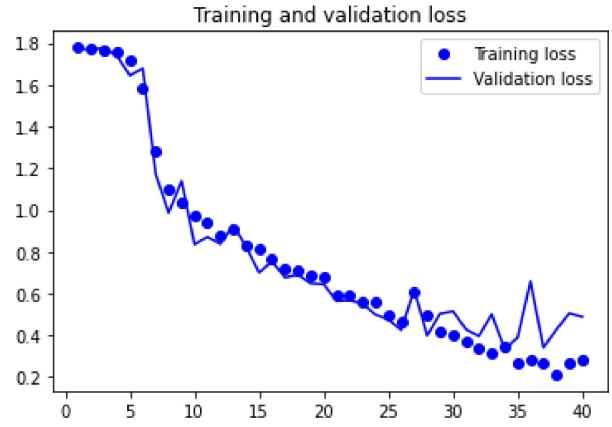




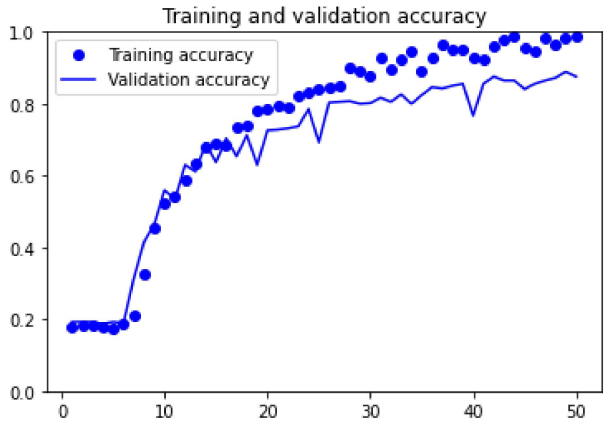
Six classes models

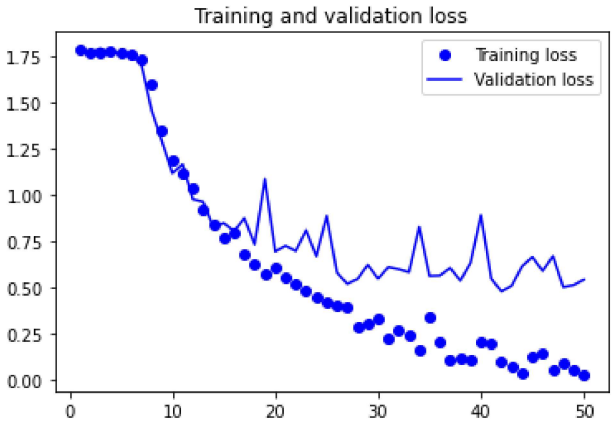
6c vgg16 5bl no augm filt=32, ep40



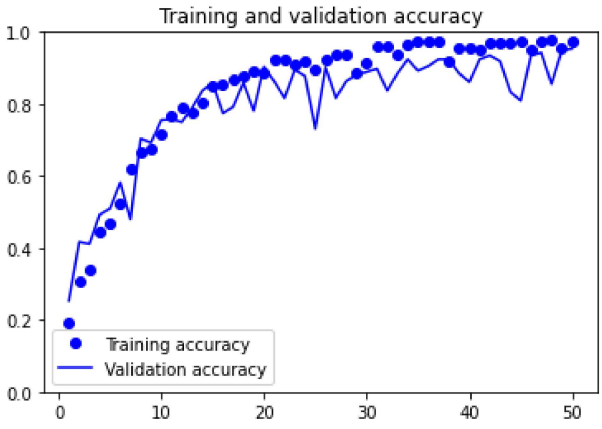


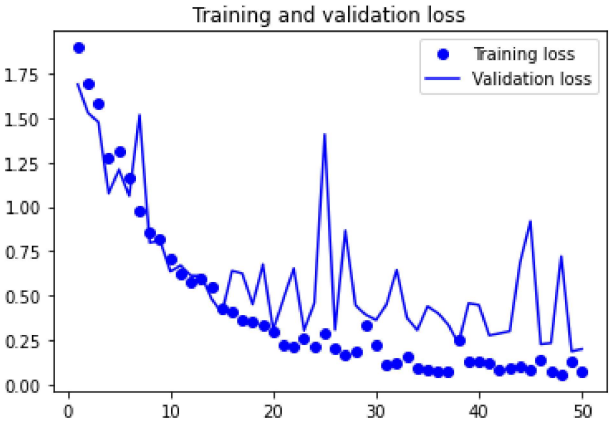
6c vgg16 6bl noaugm filter=32



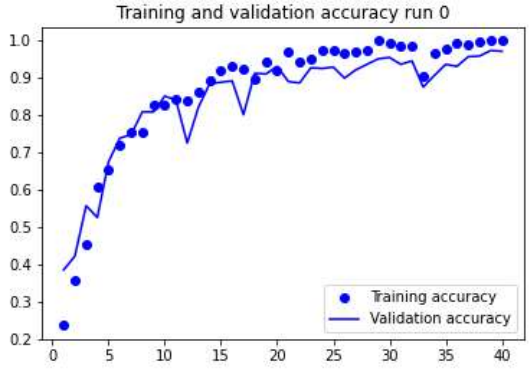


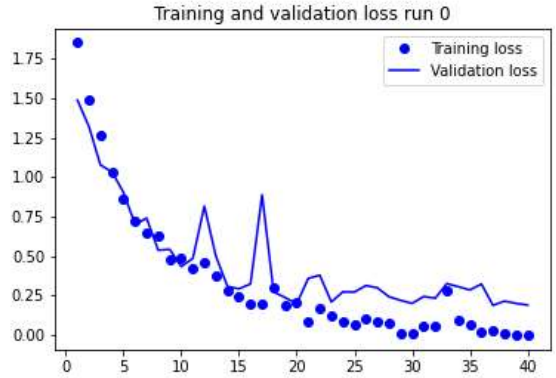
6c resblok 6bl augm filter=32-256



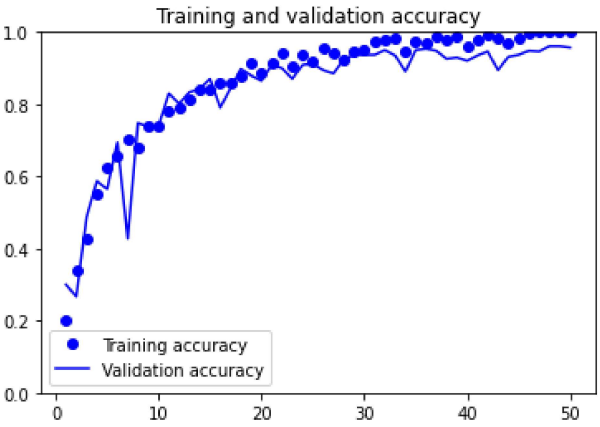


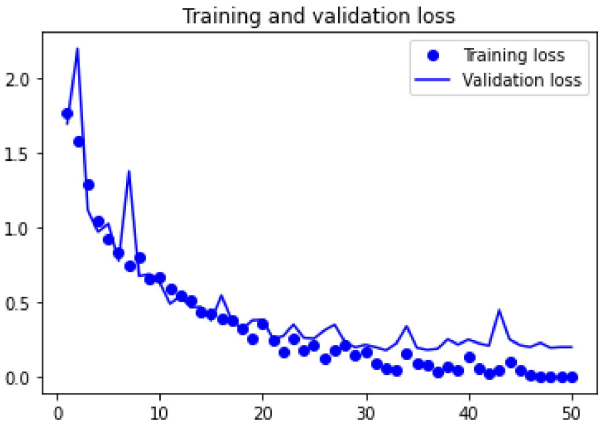
6c resblok 6bl noaugm filter=32-256



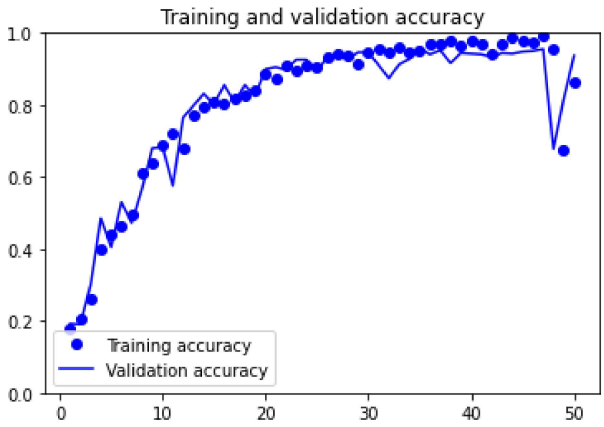


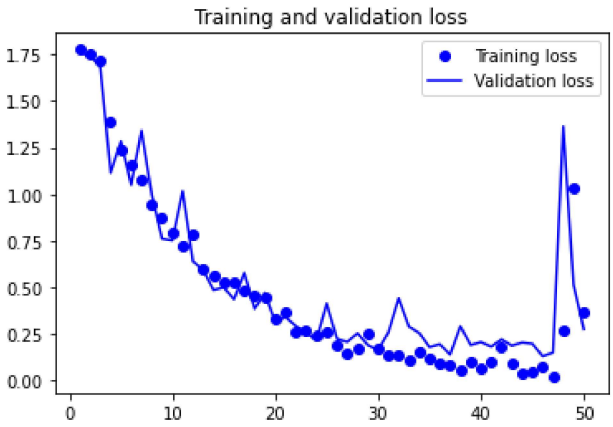
6c resblok 6bl noaugm filter=32



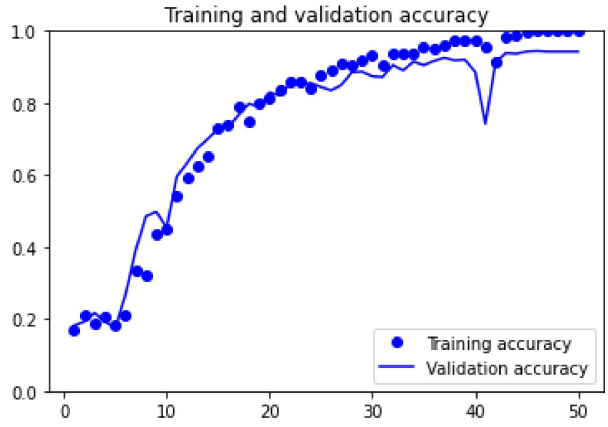


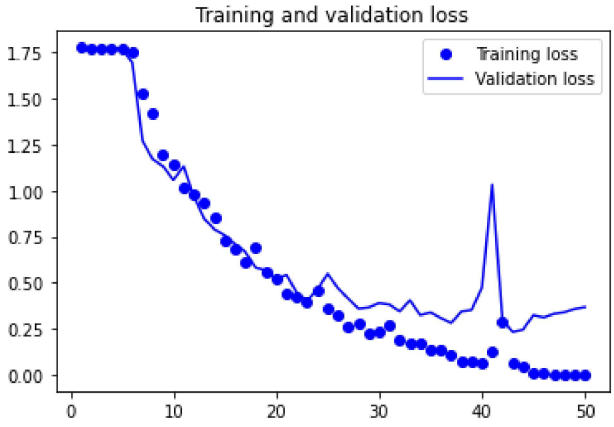
6c inception 5bl no augm filt=32



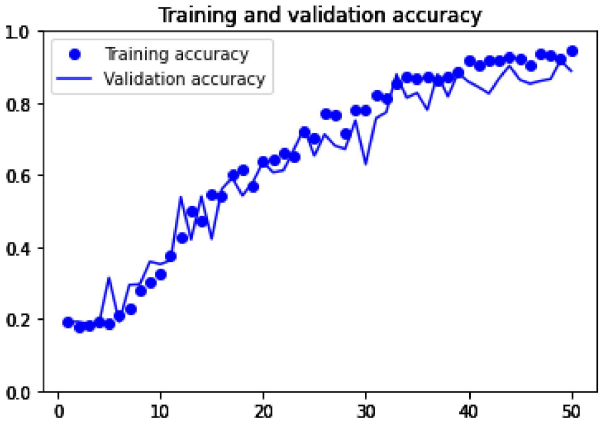


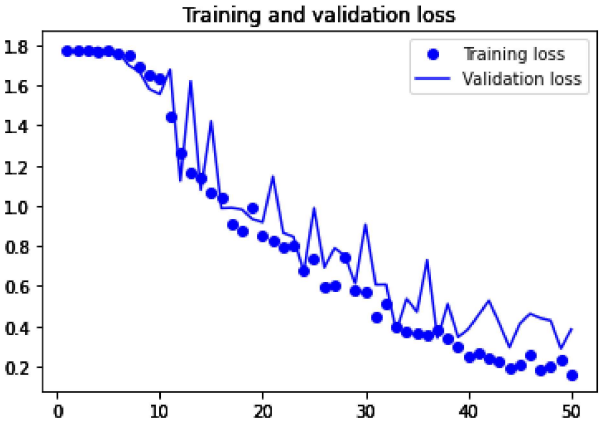
6c inception 6bl no augm filt=32





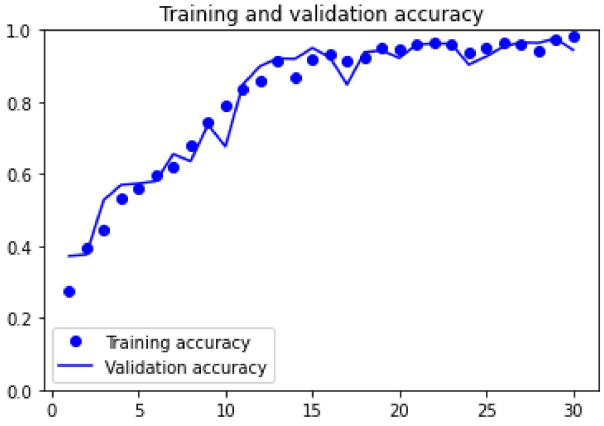
6c inception 6bl augm filt=32

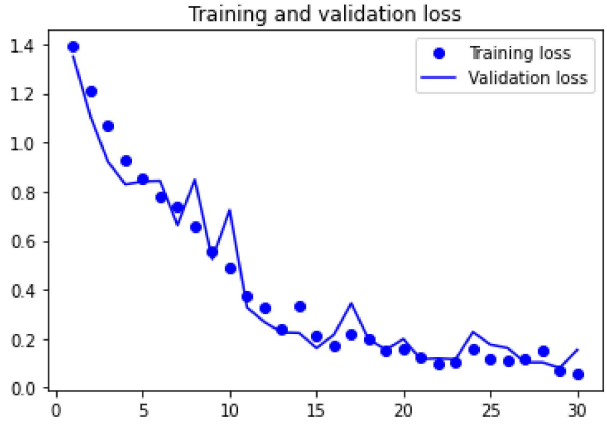




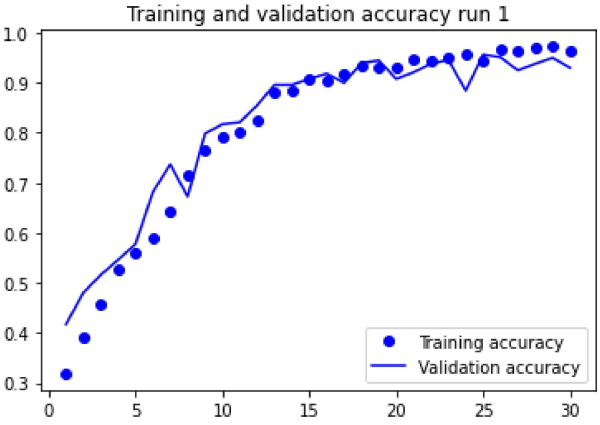
Cross validation four classes

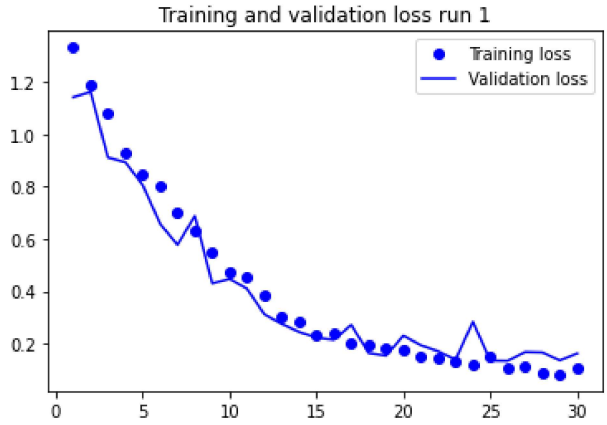
resblock5 blocks+ no augment+ep 30, filter=32 seed=0



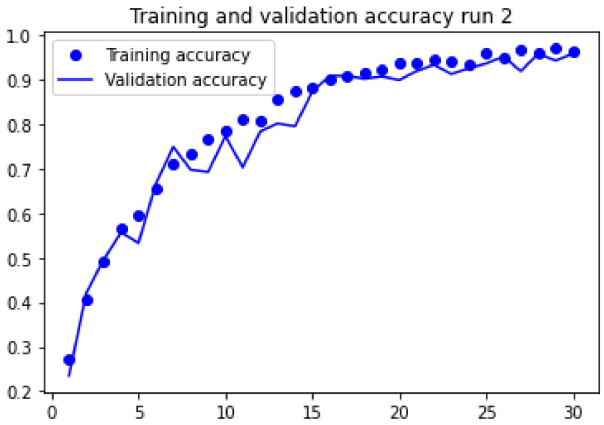


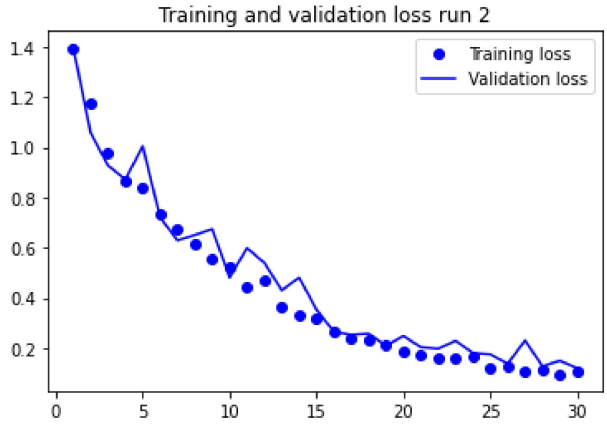
resblock5 blocks+ no augment+ep 30, filter=32 seed=1





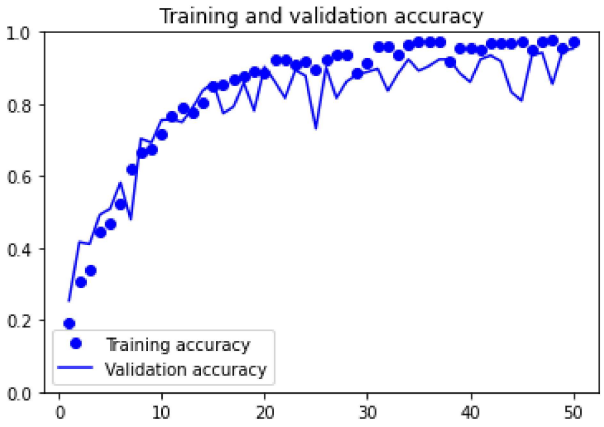
resblock5 blocks+ no augment+ep 30, filter=32 seed=2

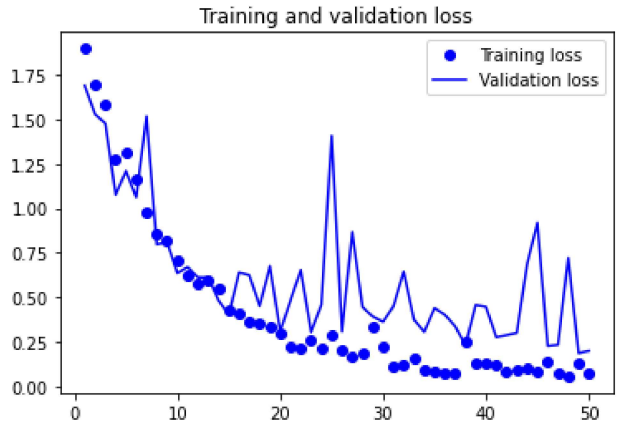




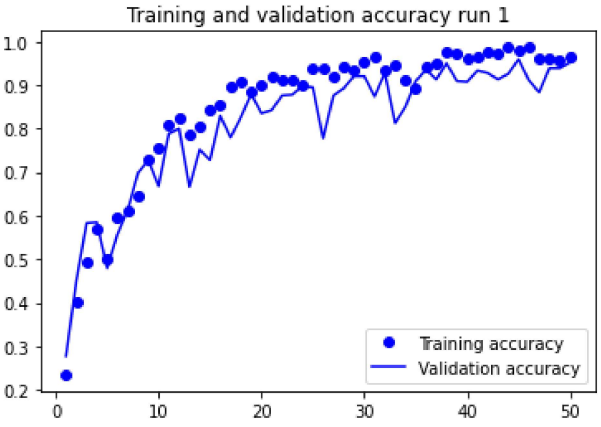
Cross validation six classes

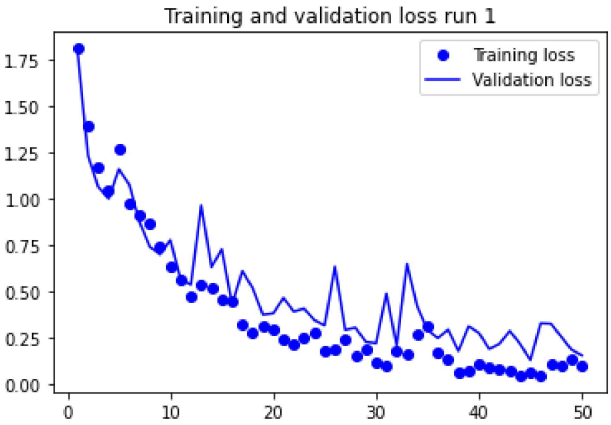
6c resblok 6bl augm filter=32-256, seed=0



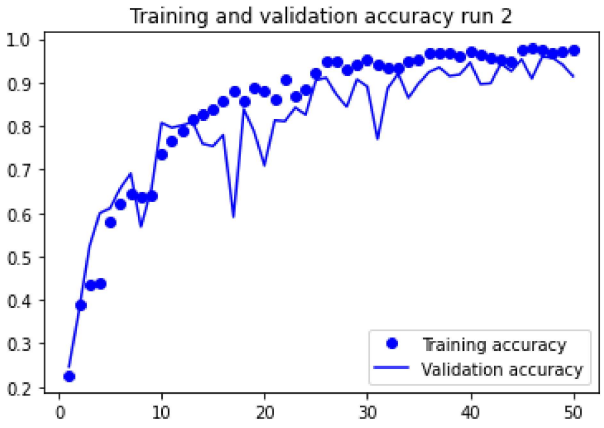


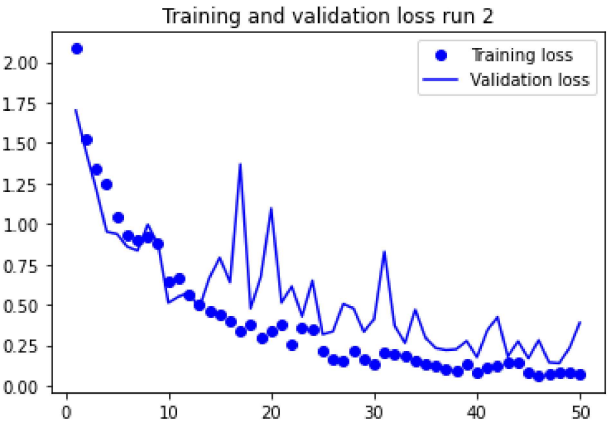
6c resblok 6bl augm filter=32-256, seed=1





6c resblok 6bl augm filter=32-256, seed=2





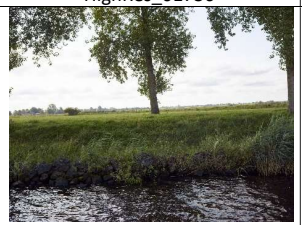
D Test images

The images below are used as test images for the classifier and the object detector. These are resized versions of the original as the original took a lot of space and we needed an size that could easily fit in the author personal cloud drive. The table below also shows the ground truth for the six classes model. This was made with the help of the expert from Witteveen+Bos. The ground truths of the four classes models can be derived from this table by:

- Metal good of four class is the combined of metal good and metal acceptable from the six class.
- Metal bad of four class is the combined of metal moderate and metal bad from the six class.

File	Truth	File	Truth	File	Truth	File	Truth
HighRes_01386	Grass	HighRes_10770	M-good	HighRes_25664	M-good	HighRes_38901	Rock
HighRes_01387	Grass	HighRes_10771	M-good	HighRes_25665	M-good	HighRes_38902	Rock
HighRes_01388	Grass	HighRes_10772	M-good	HighRes_25666	M-good	HighRes_38903	Rock
HighRes_01389	Grass	HighRes_10773	M-good	HighRes_25667	M-good	HighRes_38904	Grass
HighRes_01390	Grass	HighRes_10774	M-good	HighRes_25668	M-good	HighRes_38907	Grass
HighRes_01605	Grass	HighRes_10775	M-good	HighRes_25669	M-good	HighRes_38908	Grass
HighRes_01606	Grass	HighRes_10776	M-good	HighRes_25670	M-good	HighRes_38909	Grass
HighRes_01735	Rock	HighRes_10777	M-good	HighRes_25671	M-good	HighRes_38910	Grass
HighRes_01736	Rock	HighRes_10778	M-good	HighRes_25672	M-good	HighRes_38911	Grass
HighRes_10556	M-acceptable	HighRes_21868	M-bad	HighRes_25673	M-good	HighRes_38912	Grass
HighRes_10557	Rock	HighRes_21869	M-bad	HighRes_25674	M-good	HighRes_38915	Grass
HighRes_10563	Rock	HighRes_22695	M-bad	HighRes_25675	M-good	HighRes_38916	Grass
HighRes_10564	Rock	HighRes_22696	M-moderate	HighRes_25676	M-good	HighRes_38918	Grass
HighRes_10571	Rock	HighRes_22697	M-moderate	HighRes_25996	M-good	HighRes_38920	Grass
HighRes_10572	Rock	HighRes_22698	M-moderate	HighRes_25997	M-good	HighRes_38922	Grass
HighRes_10575	Rock	HighRes_22699	M-moderate	HighRes_25998	M-good	HighRes_38923	Grass
HighRes_10576	Rock	HighRes_22732	M-moderate	HighRes_25999	M-good	HighRes_38924	Grass
HighRes_10577	Rock	HighRes_23655	M-moderate	HighRes_26000	M-good	HighRes_38926	Rock
HighRes_10601	Rock	HighRes_25584	M-good	HighRes_30101	M-good	HighRes_38927	Rock
HighRes_10602	Rock	HighRes_25585	M-good	HighRes_30102	M-acceptable	HighRes_38928	Rock
HighRes_10765	Rock	HighRes_25586	M-good	HighRes_32120	M-acceptable	HighRes_38929	Rock
HighRes_10766	M-good	HighRes_25587	M-good	HighRes_32121	M-acceptable	HighRes_38930	Rock
HighRes_10767	M-good	HighRes_25588	M-good	HighRes_32125	M-acceptable	HighRes_38931	Rock
HighRes_10768	M-good	HighRes_25589	M-good	HighRes_38898	Grass	HighRes_38932	Rock
HighRes_10769	M-good	HighRes_25663	M-good	HighRes_38900	Rock	HighRes_38933	Rock

Table 15: Ground truths of all images in model test.

						
HighRes_01386	HighRes_01387	HighRes_01388	HighRes_01389	HighRes_01390	HighRes_01605	HighRes_01606
						
HighRes_01735	HighRes_01736	HighRes_10556	HighRes_10557	HighRes_10563	HighRes_10564	HighRes_10571
						
HighRes_10572	HighRes_10575	HighRes_10576	HighRes_10577	HighRes_10601	HighRes_10602	HighRes_10765
						
HighRes_10766	HighRes_10767	HighRes_10768	HighRes_10769	HighRes_10770	HighRes_10771	HighRes_10772
						
HighRes_10773	HighRes_10774	HighRes_10775	HighRes_10776	HighRes_10777	HighRes_10778	HighRes_21868

						
HighRes_21869	HighRes_22695	HighRes_22696	HighRes_22697	HighRes_22698	HighRes_22699	HighRes_22732
						
HighRes_23655	HighRes_25584	HighRes_25585	HighRes_25586	HighRes_25587	HighRes_25588	HighRes_25589
						
HighRes_25663	HighRes_25664	HighRes_25665	HighRes_25666	HighRes_25667	HighRes_25668	HighRes_25669
						
HighRes_25670	HighRes_25671	HighRes_25672	HighRes_25673	HighRes_25674	HighRes_25675	HighRes_25676
						
HighRes_25996	HighRes_25997	HighRes_25998	HighRes_25999	HighRes_26000	HighRes_30101	HighRes_30102

 HighRes_32120	 HighRes_32121	 HighRes_32125	 HighRes_38898	 HighRes_38900	 HighRes_38901	 HighRes_38902
 HighRes_38903	 HighRes_38904	 HighRes_38907	 HighRes_38908	 HighRes_38909	 HighRes_38910	 HighRes_38911
 HighRes_38912	 HighRes_38915	 HighRes_38916	 HighRes_38918	 HighRes_38920	 HighRes_38922	 HighRes_38923
 HighRes_38924	 HighRes_38926	 HighRes_38927	 HighRes_38928	 HighRes_38929	 HighRes_38930	 HighRes_38931
 HighRes_38932	 HighRes_38933					

E YOLOv4 detections

This appendix contain the images in which the object detector predicted dimensional features in order to estimate the specific dimensions of the piling sheet. There are two types of classes the object detector will predict:

1. *Dim.* The bumps on the piling sheet.
2. *Ref.* The reference object with known dimensions. In this case an anchor with horizontal width of 15 cm

The detector was trained using the instructions on Alexey Bochkovskiy's Github page. After downloading the weights file it is possible to run inference outside of Darknet using OpenCV. Notice that not all images, that contained part of a piling sheet are present. This is due to fact that the classifier failed to classify the image as metal.


HighRes10765


HighRes10766


HighRes10767


HighRes10768


HighRes10769


HighRes10770


HighRes10771


HighRes10772


HighRes10773


HighRes10774


HighRes10775


HighRes10776

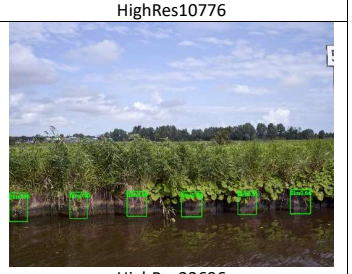

HighRes10777

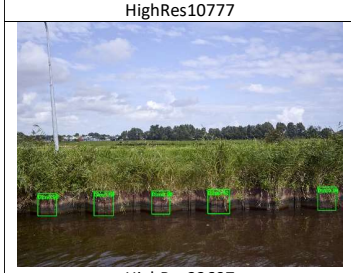

HighRes10778


HighRes21868


HighRes21869

HighRes22695


HighRes22696

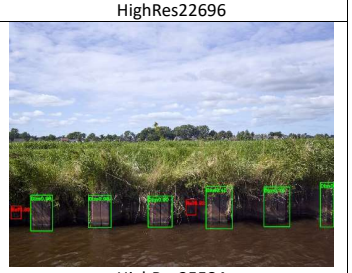

HighRes22697


HighRes22698

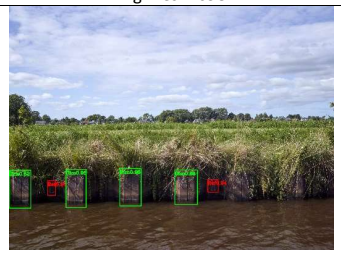

HighRes22699


HighRes22732


HighRes23655

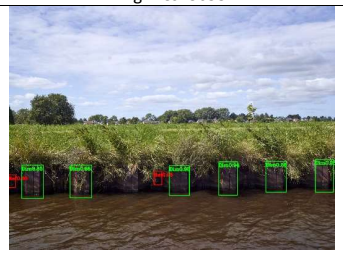

HighRes25584

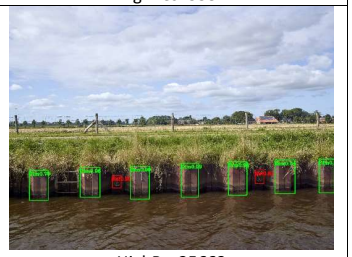

HighRes25585

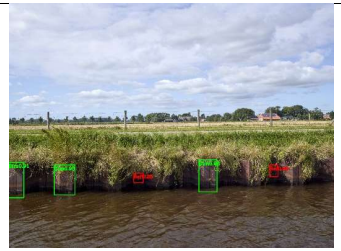
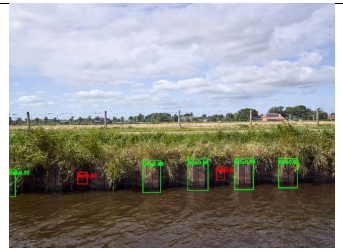
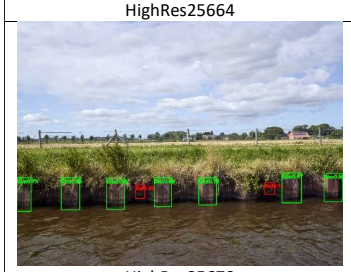
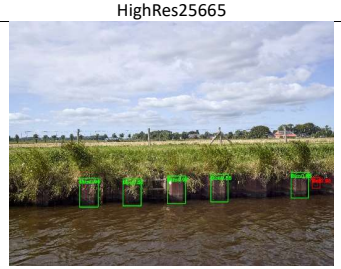
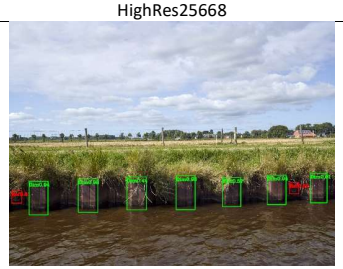
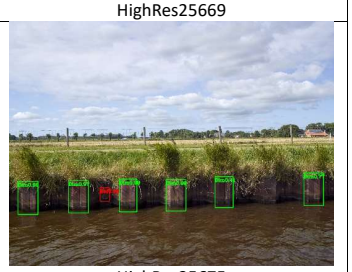
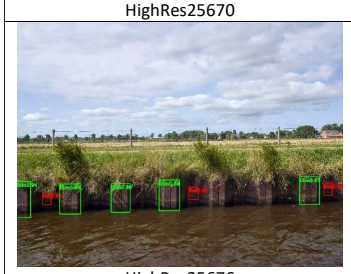

HighRes25586


HighRes25587


HighRes25588


HighRes25589


HighRes25663

 HighRes25664	 HighRes25665	 HighRes25666	 HighRes25667	 HighRes25668	 HighRes25669
 HighRes25670	 HighRes25671	 HighRes25672	 HighRes25673	 HighRes25674	 HighRes25675
 HighRes25676	 HighRes25996	 HighRes25997	 HighRes25998	 HighRes25999	 HighRes26000
 HighRes30101	 HighRes30102	 HighRes32120	 HighRes32121	 HighRes32125	

F Final Data-frames

This appendix contains the data-frame in which the results of the classification and the object detection algorithm got stored. There is a data-frame in which the classifier has four classes and another data-frame with the classifier of six classes. Brief explanation of the columns:

- **Filename**. The name of the file, this is needed to add additional information such as coördinates.
- **cameraLon, cameraLat, cameraAlt**. The geographic coordinates to which that images belongs to.
- **Distance bumps**. The distance, in cm, between the bumps as shown in chapter 2.
- **Reference object**. A binary case whether a reference object is present in the image.
- **Height**. The height, in cm, of the piling sheet that is above water.
- **Material**. The main material present in the image in which all metal grades would be labeled metal. This is done so the object detection algorithm can go over all the metal labeled images. The subjects in this column can be *Grass*, *Metal* or *Rock*.
- **Rust grade**. This is the actual prediction the classification would make. The classes *Grass* and *Rock* were kept for the ease of making confusion matrices.

Dataframe four class classifier

Filename	cameraLon	cameraLat	cameraAlt	Distance bumps	Reference object	Height	Material	Rust grade
HighRes_01386	5.692051	52.968265	-1.51	NaN	NaN	NaN	Grass	Grass
HighRes_01387	5.692022	52.96832	-1.51	NaN	NaN	NaN	Grass	Grass
HighRes_01388	5.691991	52.968376	-1.51	NaN	NaN	NaN	Grass	Grass
HighRes_01389	5.691958	52.96843	-1.52	NaN	NaN	NaN	Grass	Grass
HighRes_01390	5.691923	52.968484	-1.54	NaN	NaN	NaN	Grass	Grass
HighRes_01605	5.690281	52.975704	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_01606	5.690255	52.975772	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_01735	5.688134	52.983188	-0.38	NaN	NaN	NaN	Rock	Rock
HighRes_01736	5.688145	52.983254	-0.34	NaN	NaN	NaN	Rock	Rock
HighRes_10556	6.220051	53.237499	-4.16	NaN	NaN	NaN	Rock	Rock
HighRes_10557	6.220186	53.237506	-4.18	NaN	NaN	NaN	Rock	Rock
HighRes_10563	6.22093	53.237596	-4.23	NaN	NaN	NaN	Rock	Rock
HighRes_10564	6.221048	53.237616	-4.21	NaN	NaN	NaN	Rock	Rock
HighRes_10571	6.221783	53.237778	-4.07	NaN	NaN	NaN	Rock	Rock
HighRes_10572	6.22189	53.237805	-4.03	NaN	NaN	NaN	Rock	Rock
HighRes_10575	6.222209	53.237891	-3.91	NaN	NaN	NaN	Rock	Rock
HighRes_10576	6.222318	53.237918	-3.88	NaN	NaN	NaN	Rock	Rock
HighRes_10577	6.222429	53.237946	-3.84	NaN	NaN	NaN	Rock	Rock
HighRes_10601	6.224834	53.238695	-2.88	NaN	NaN	NaN	Rock	Rock
HighRes_10602	6.224937	53.238724	-2.83	NaN	NaN	NaN	Rock	Rock
HighRes_10765	6.241695	53.241041	-1.18	NaN	No	59.5	Metal	M-good
HighRes_10766	6.241794	53.241052	-1.19	95.4	No	57.3	Metal	M-good
HighRes_10767	6.241892	53.241063	-1.23	NaN	No	51.7	Metal	M-good
HighRes_10768	6.24199	53.241073	-1.23	183.3	No	52.7	Metal	M-good
HighRes_10769	6.242091	53.241081	-1.26	NaN	No	45.5	Metal	M-good
HighRes_10770	6.242189	53.241096	-1.24	NaN	No	40	Metal	M-good
HighRes_10771	6.242286	53.241107	-1.29	66	Yes	31.5	Metal	M-good
HighRes_10772	6.242384	53.241116	-1.33	100.8	No	51.2	Metal	M-good
HighRes_10773	6.242481	53.241127	-1.26	99.3	No	48	Metal	M-good
HighRes_10774	6.242576	53.241138	-1.27	NaN	No	51	Metal	M-good
HighRes_10775	6.242673	53.241146	-1.24	NaN	No	49	Metal	M-good
HighRes_10776	6.242771	53.241154	-1.19	105.3	No	55.2	Metal	M-good

HighRes_10777	6.242871	53.241164	-1.17	NaN	No	62.1	Metal	M-good
HighRes_10778	6.242969	53.241176	-1.17	NaN	No	60.6	Metal	M-good
HighRes_21868	6.937547	53.317986	1.38	77.6	No	38.1	Metal	M-bad
HighRes_21869	6.937496	53.317935	1.38	74.7	No	38.3	Metal	M-bad
HighRes_22695	6.88108	53.311005	-0.2	120.7	No	54.8	Metal	M-bad
HighRes_22696	6.880986	53.310988	-0.18	133.6	No	59.9	Metal	M-bad
HighRes_22697	6.880893	53.310967	-0.18	134.9	No	55.7	Metal	M-bad
HighRes_22698	6.880802	53.31095	-0.21	117.7	No	53.8	Metal	M-bad
HighRes_22699	6.880711	53.310941	-0.25	112.3	No	52.3	Metal	M-bad
HighRes_22732	6.877594	53.310419	-0.33	134.7	No	58.9	Metal	M-bad
HighRes_23655	6.80939	53.295821	-3	83.4	No	77.7	Metal	M-bad
HighRes_25584	6.662003	53.24255	4.04	85.5	Yes	56.1	Metal	M-good
HighRes_25585	6.66193	53.242511	4.03	82	Yes	51.8	Metal	M-good
HighRes_25586	6.661856	53.242472	4.04	91	Yes	60.8	Metal	M-good
HighRes_25587	6.661782	53.242432	4.06	93.4	Yes	61.2	Metal	M-good
HighRes_25588	6.661705	53.242393	4.05	76.5	Yes	49.8	Metal	M-good
HighRes_25589	6.661626	53.242354	4.07	98.3	Yes	65.3	Metal	M-good
HighRes_25663	6.655831	53.239709	4.27	81.1	Yes	54.9	Metal	M-good
HighRes_25664	6.655751	53.239671	4.26	79.6	Yes	51.4	Metal	M-good
HighRes_25665	6.655671	53.239633	4.27	81.9	Yes	54.7	Metal	M-good
HighRes_25666	6.655589	53.239596	4.26	78.8	Yes	52.6	Metal	M-good
HighRes_25667	6.655507	53.23956	4.25	81.7	Yes	54.2	Metal	M-good
HighRes_25668	6.655428	53.239521	4.25	79	Yes	50.1	Metal	M-good
HighRes_25669	6.65535	53.239483	4.28	78.5	Yes	50.1	Metal	M-good
HighRes_25670	6.65527	53.239446	4.31	70.8	Yes	47.4	Metal	M-good
HighRes_25671	6.65519	53.23941	4.3	80.5	Yes	50.3	Metal	M-good
HighRes_25672	6.655109	53.239376	4.3	79.8	Yes	54.5	Metal	M-good
HighRes_25673	6.655026	53.239343	4.28	86.1	Yes	53.1	Metal	M-good
HighRes_25674	6.654944	53.239308	4.27	72.5	Yes	49.2	Metal	M-good
HighRes_25675	6.654865	53.239271	4.28	80.6	Yes	52.7	Metal	M-good
HighRes_25676	6.654788	53.239235	4.25	83.4	Yes	52.8	Metal	M-good
HighRes_25996	6.633336	53.232228	2.8	91.8	Yes	37.2	Metal	M-good
HighRes_25997	6.633236	53.2322	2.9	88	Yes	39.8	Metal	M-good
HighRes_25998	6.633141	53.23217	2.88	116.4	Yes	47.6	Metal	M-good
HighRes_25999	6.63304	53.232138	2.9	96.2	Yes	42.4	Metal	M-good

HighRes_26000	6.632938	53.232111	2.9	NaN	No	55.8	Metal	M-good
HighRes_30101	6.251881	53.242587	2.11	NaN	No	82.2	Metal	M-good
HighRes_30102	6.251773	53.242577	2.09	149.1	No	88	Metal	M-good
HighRes_32120	6.090108	53.204385	-2.38	121.3	No	202.1	Metal	M-bad
HighRes_32121	6.090003	53.204354	-2.37	119.7	No	187.1	Metal	M-bad
HighRes_32125	6.089582	53.204238	-2.38	113	No	181.8	Metal	M-bad
HighRes_38898	5.694125	52.923399	-1.05	NaN	NaN	NaN	Grass	Grass
HighRes_38900	5.694204	52.923283	-1.02	NaN	NaN	NaN	Rock	Rock
HighRes_38901	5.694241	52.923225	-1	NaN	NaN	NaN	Rock	Rock
HighRes_38902	5.694278	52.923167	-0.99	NaN	NaN	NaN	Rock	Rock
HighRes_38903	5.694313	52.923108	-0.97	NaN	NaN	NaN	Rock	Rock
HighRes_38904	5.694344	52.923048	-0.99	NaN	NaN	NaN	Grass	Grass
HighRes_38907	5.694445	52.922868	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_38908	5.694475	52.922807	-0.95	NaN	NaN	NaN	Grass	Grass
HighRes_38909	5.6945	52.922745	-0.94	NaN	NaN	NaN	Grass	Grass
HighRes_38910	5.694532	52.922685	-0.95	NaN	NaN	NaN	Grass	Grass
HighRes_38911	5.694559	52.922625	-0.96	NaN	NaN	NaN	Grass	Grass
HighRes_38912	5.694589	52.922565	-0.94	NaN	NaN	NaN	Grass	Grass
HighRes_38915	5.694686	52.922386	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_38916	5.694719	52.922326	-0.93	NaN	NaN	NaN	Grass	Grass
HighRes_38918	5.694797	52.922209	-0.87	NaN	NaN	NaN	Grass	Grass
HighRes_38920	5.694849	52.92209	-0.89	NaN	NaN	NaN	Grass	Grass
HighRes_38922	5.694915	52.921969	-0.9	NaN	NaN	NaN	Grass	Grass
HighRes_38923	5.694949	52.921908	-0.88	NaN	NaN	NaN	Grass	Grass
HighRes_38924	5.694982	52.921846	-0.88	NaN	NaN	NaN	Grass	Grass
HighRes_38926	5.695047	52.921724	-0.84	NaN	NaN	NaN	Rock	Rock
HighRes_38927	5.695083	52.921663	-0.83	NaN	NaN	NaN	Rock	Rock
HighRes_38928	5.695116	52.921602	-0.81	NaN	NaN	NaN	Rock	Rock
HighRes_38929	5.695148	52.921541	-0.8	NaN	NaN	NaN	Rock	Rock
HighRes_38930	5.695183	52.921481	-0.81	NaN	NaN	NaN	Rock	Rock
HighRes_38931	5.695213	52.92142	-0.79	NaN	NaN	NaN	Rock	Rock
HighRes_38932	5.695238	52.921359	-0.78	NaN	NaN	NaN	Rock	Rock
HighRes_38933	5.695265	52.921298	-0.77	NaN	NaN	NaN	Rock	Rock

Dataframe six class classifier

Filename	cameraLon	cameraLat	cameraAlt	Distance bumps	Reference object	Height	Material	Rust grade
HighRes_01386	5.692051	52.968265	-1.51	NaN	NaN	NaN	Grass	Grass
HighRes_01387	5.692022	52.96832	-1.51	NaN	NaN	NaN	Grass	Grass
HighRes_01388	5.691991	52.968376	-1.51	NaN	NaN	NaN	Grass	Grass
HighRes_01389	5.691958	52.96843	-1.52	NaN	NaN	NaN	Grass	Grass
HighRes_01390	5.691923	52.968484	-1.54	NaN	NaN	NaN	Grass	Grass
HighRes_01605	5.690281	52.975704	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_01606	5.690255	52.975772	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_01735	5.688134	52.983188	-0.38	NaN	NaN	NaN	Rock	Rock
HighRes_01736	5.688145	52.983254	-0.34	NaN	NaN	NaN	Rock	Rock
HighRes_10556	6.220051	53.237499	-4.16	NaN	NaN	NaN	Rock	Rock
HighRes_10557	6.220186	53.237506	-4.18	NaN	NaN	NaN	Rock	Rock
HighRes_10563	6.22093	53.237596	-4.23	NaN	NaN	NaN	Rock	Rock
HighRes_10564	6.221048	53.237616	-4.21	NaN	NaN	NaN	Rock	Rock
HighRes_10571	6.221783	53.237778	-4.07	NaN	NaN	NaN	Rock	Rock
HighRes_10572	6.22189	53.237805	-4.03	NaN	NaN	NaN	Rock	Rock
HighRes_10575	6.222209	53.237891	-3.91	NaN	NaN	NaN	Rock	Rock
HighRes_10576	6.222318	53.237918	-3.88	NaN	NaN	NaN	Rock	Rock
HighRes_10577	6.222429	53.237946	-3.84	NaN	NaN	NaN	Rock	Rock
HighRes_10601	6.224834	53.238695	-2.88	NaN	NaN	NaN	Rock	Rock
HighRes_10602	6.224937	53.238724	-2.83	NaN	NaN	NaN	Rock	Rock
HighRes_10765	6.241695	53.241041	-1.18	NaN	No	59.5	Metal	M-good
HighRes_10766	6.241794	53.241052	-1.19	95.4	No	57.3	Metal	M-good
HighRes_10767	6.241892	53.241063	-1.23	NaN	NaN	NaN	Rock	Rock
HighRes_10768	6.24199	53.241073	-1.23	NaN	NaN	NaN	Rock	Rock
HighRes_10769	6.242091	53.241081	-1.26	NaN	NaN	NaN	Rock	Rock
HighRes_10770	6.242189	53.241096	-1.24	NaN	NaN	NaN	Rock	Rock
HighRes_10771	6.242286	53.241107	-1.29	NaN	NaN	NaN	Rock	Rock
HighRes_10772	6.242384	53.241116	-1.33	NaN	NaN	NaN	Rock	Rock
HighRes_10773	6.242481	53.241127	-1.26	NaN	NaN	NaN	Rock	Rock
HighRes_10774	6.242576	53.241138	-1.27	NaN	NaN	NaN	Rock	Rock
HighRes_10775	6.242673	53.241146	-1.24	NaN	No	49	Metal	M-good
HighRes_10776	6.242771	53.241154	-1.19	105.3	No	55.2	Metal	M-good

HighRes_10777	6.242871	53.241164	-1.17	NaN	NaN	NaN	Rock	Rock
HighRes_10778	6.242969	53.241176	-1.17	NaN	NaN	NaN	Rock	Rock
HighRes_21868	6.937547	53.317986	1.38	77.6	No	38.1	Metal	M-bad
HighRes_21869	6.937496	53.317935	1.38	74.7	No	38.3	Metal	M-bad
HighRes_22695	6.88108	53.311005	-0.2	120.7	No	54.8	Metal	M-bad
HighRes_22696	6.880986	53.310988	-0.18	133.6	No	59.9	Metal	M-bad
HighRes_22697	6.880893	53.310967	-0.18	134.9	No	55.7	Metal	M-bad
HighRes_22698	6.880802	53.31095	-0.21	117.7	No	53.8	Metal	M-bad
HighRes_22699	6.880711	53.310941	-0.25	112.3	No	52.3	Metal	M-bad
HighRes_22732	6.877594	53.310419	-0.33	134.7	No	58.9	Metal	M-bad
HighRes_23655	6.80939	53.295821	-3	83.4	No	77.7	Metal	M-bad
HighRes_25584	6.662003	53.24255	4.04	85.5	Yes	56.1	Metal	M-good
HighRes_25585	6.66193	53.242511	4.03	82	Yes	51.8	Metal	M-good
HighRes_25586	6.661856	53.242472	4.04	91	Yes	60.8	Metal	M-good
HighRes_25587	6.661782	53.242432	4.06	93.4	Yes	61.2	Metal	M-good
HighRes_25588	6.661705	53.242393	4.05	76.5	Yes	49.8	Metal	M-good
HighRes_25589	6.661626	53.242354	4.07	98.3	Yes	65.3	Metal	M-good
HighRes_25663	6.655831	53.239709	4.27	81.1	Yes	54.9	Metal	M-good
HighRes_25664	6.655751	53.239671	4.26	79.6	Yes	51.4	Metal	M-bad
HighRes_25665	6.655671	53.239633	4.27	81.9	Yes	54.7	Metal	M-good
HighRes_25666	6.655589	53.239596	4.26	78.8	Yes	52.6	Metal	M-good
HighRes_25667	6.655507	53.23956	4.25	81.7	Yes	54.2	Metal	M-good
HighRes_25668	6.655428	53.239521	4.25	79	Yes	50.1	Metal	M-good
HighRes_25669	6.65535	53.239483	4.28	78.5	Yes	50.1	Metal	M-good
HighRes_25670	6.65527	53.239446	4.31	70.8	Yes	47.4	Metal	M-good
HighRes_25671	6.65519	53.23941	4.3	80.5	Yes	50.3	Metal	M-good
HighRes_25672	6.655109	53.239376	4.3	79.8	Yes	54.5	Metal	M-good
HighRes_25673	6.655026	53.239343	4.28	86.1	Yes	53.1	Metal	M-good
HighRes_25674	6.654944	53.239308	4.27	72.5	Yes	49.2	Metal	M-good
HighRes_25675	6.654865	53.239271	4.28	80.6	Yes	52.7	Metal	M-good
HighRes_25676	6.654788	53.239235	4.25	83.4	Yes	52.8	Metal	M-good
HighRes_25996	6.633336	53.232228	2.8	91.8	Yes	37.2	Metal	M-good
HighRes_25997	6.633236	53.2322	2.9	88	Yes	39.8	Metal	M-good
HighRes_25998	6.633141	53.23217	2.88	116.4	Yes	47.6	Metal	M-good
HighRes_25999	6.63304	53.232138	2.9	96.2	Yes	42.4	Metal	M-good

HighRes_26000	6.632938	53.232111	2.9	NaN	No	55.8	Metal	M-good
HighRes_30101	6.251881	53.242587	2.11	NaN	No	82.2	Metal	M-good
HighRes_30102	6.251773	53.242577	2.09	149.1	No	88	Metal	M-good
HighRes_32120	6.090108	53.204385	-2.38	121.3	No	202.1	Metal	M-bad
HighRes_32121	6.090003	53.204354	-2.37	119.7	No	187.1	Metal	M-good
HighRes_32125	6.089582	53.204238	-2.38	113	No	181.8	Metal	M-bad
HighRes_38898	5.694125	52.923399	-1.05	NaN	NaN	NaN	Grass	Grass
HighRes_38900	5.694204	52.923283	-1.02	NaN	NaN	NaN	Rock	Rock
HighRes_38901	5.694241	52.923225	-1	NaN	NaN	NaN	Rock	Rock
HighRes_38902	5.694278	52.923167	-0.99	NaN	NaN	NaN	Rock	Rock
HighRes_38903	5.694313	52.923108	-0.97	NaN	NaN	NaN	Rock	Rock
HighRes_38904	5.694344	52.923048	-0.99	NaN	NaN	NaN	Grass	Grass
HighRes_38907	5.694445	52.922868	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_38908	5.694475	52.922807	-0.95	NaN	NaN	NaN	Grass	Grass
HighRes_38909	5.6945	52.922745	-0.94	NaN	NaN	NaN	Grass	Grass
HighRes_38910	5.694532	52.922685	-0.95	NaN	NaN	NaN	Grass	Grass
HighRes_38911	5.694559	52.922625	-0.96	NaN	NaN	NaN	Grass	Grass
HighRes_38912	5.694589	52.922565	-0.94	NaN	NaN	NaN	Grass	Grass
HighRes_38915	5.694686	52.922386	-0.92	NaN	NaN	NaN	Grass	Grass
HighRes_38916	5.694719	52.922326	-0.93	NaN	NaN	NaN	Grass	Grass
HighRes_38918	5.694797	52.922209	-0.87	NaN	NaN	NaN	Grass	Grass
HighRes_38920	5.694849	52.92209	-0.89	NaN	NaN	NaN	Grass	Grass
HighRes_38922	5.694915	52.921969	-0.9	NaN	NaN	NaN	Grass	Grass
HighRes_38923	5.694949	52.921908	-0.88	NaN	NaN	NaN	Grass	Grass
HighRes_38924	5.694982	52.921846	-0.88	NaN	NaN	NaN	Rock	Rock
HighRes_38926	5.695047	52.921724	-0.84	NaN	NaN	NaN	Rock	Rock
HighRes_38927	5.695083	52.921663	-0.83	NaN	NaN	NaN	Rock	Rock
HighRes_38928	5.695116	52.921602	-0.81	NaN	NaN	NaN	Rock	Rock
HighRes_38929	5.695148	52.921541	-0.8	NaN	NaN	NaN	Rock	Rock
HighRes_38930	5.695183	52.921481	-0.81	NaN	NaN	NaN	Rock	Rock
HighRes_38931	5.695213	52.92142	-0.79	NaN	NaN	NaN	Rock	Rock
HighRes_38932	5.695238	52.921359	-0.78	NaN	NaN	NaN	Rock	Rock
HighRes_38933	5.695265	52.921298	-0.77	NaN	NaN	NaN	Rock	Rock