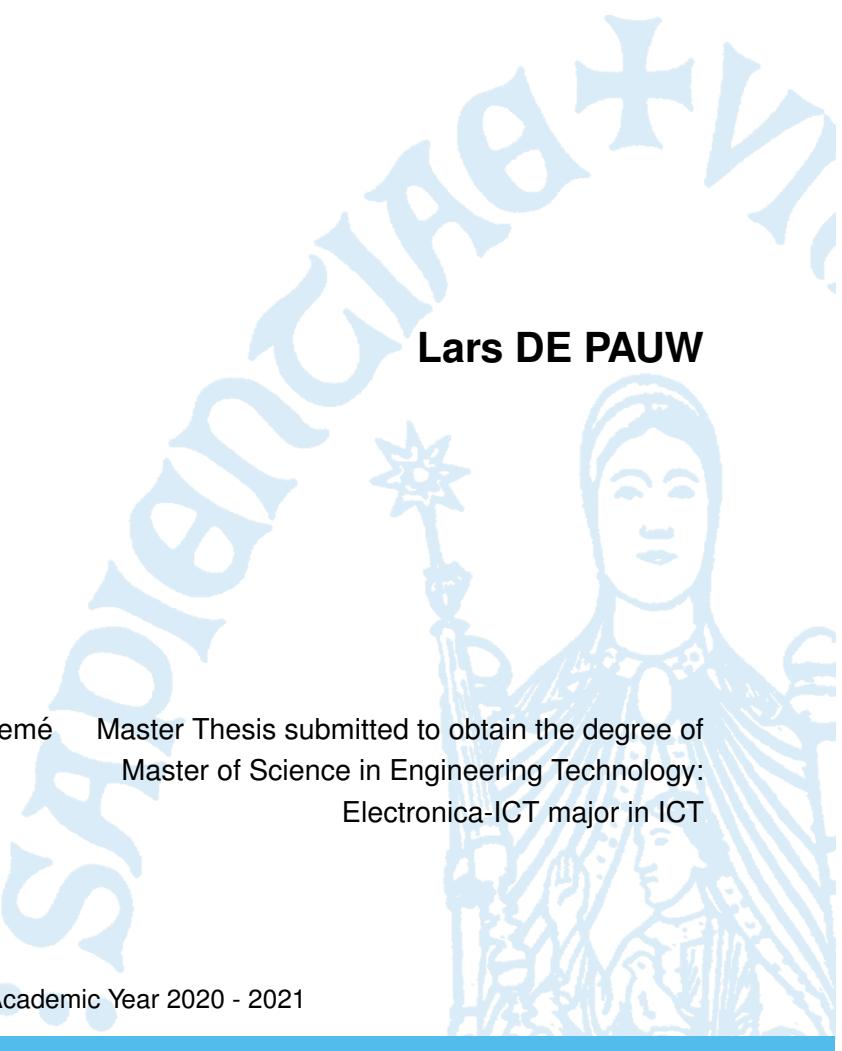


# Vision-based automatic tool-wear inspection system



Lars DE PAUW

Promotor: Prof. dr. ir. T. Goedemé Master Thesis submitted to obtain the degree of  
Master of Science in Engineering Technology:  
Electronica-ICT major in ICT

Co-promotors: Dr. ing. D. Hulens  
Dr. ir. T. Jacobs

©Copyright KU Leuven

Without written permission of the supervisor(s) and the author(s) it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilise parts of this publication should be addressed to KU Leuven, Technology Campus De Nayer, Jan De Nayerlaan 5, B-2860 Sint-Katelijne-Waver, +32 15 31 69 44 or via e-mail [fet.denayer@kuleuven.be](mailto:fet.denayer@kuleuven.be).

A written permission of the supervisor(s) is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

# Contents

<b>Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of abbreviations</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature review</b>	<b>4</b>
2.1 State of the art . . . . .	4
2.2 Information on the process . . . . .	5
2.2.1 How does the tool wear? . . . . .	5
2.2.2 Surface of the insert . . . . .	5
2.2.3 Vision algorithms . . . . .	7
2.2.4 Overview of important numbers . . . . .	10
2.2.5 Other architectures for small datasets . . . . .	10
<b>3 Implementation</b>	<b>12</b>
3.1 Camera setup . . . . .	12
3.1.1 Design of a tool holder . . . . .	12
3.1.2 Design of a camera mount . . . . .	15
3.1.3 Light configuration . . . . .	15
3.2 Datasets . . . . .	18
3.2.1 Handmade datasets . . . . .	18

3.2.2	Automated datasets . . . . .	21
3.2.3	Created datasets . . . . .	23
3.3	Vision algorithms . . . . .	27
3.3.1	Data organisation . . . . .	29
3.3.2	Model creation . . . . .	31
3.3.3	Training and testing . . . . .	32
<b>4</b>	<b>Results</b>	<b>33</b>
4.1	Setup . . . . .	33
4.2	Second handmade dataset results . . . . .	33
4.2.1	Overall results . . . . .	34
4.2.2	Comparison between different model architectures . . . . .	34
4.2.3	Influence of Transfer learning on the results . . . . .	36
4.2.4	Other parameter influences . . . . .	36
4.2.5	Test images . . . . .	38
4.3	Birthday dataset results . . . . .	38
4.4	Spaghetti dataset results . . . . .	38
4.4.1	Overall results for five best runs . . . . .	38
4.4.2	Comparison between different model architectures . . . . .	38
4.4.3	Validation Accuracy . . . . .	40
4.4.4	Influence of transfer learning on the results . . . . .	40
4.4.5	Red vs White . . . . .	41
4.4.6	Training runs . . . . .	43
4.4.7	Test images . . . . .	43
4.5	Spaghetti dataset compared with second handmade dataset . . . . .	44
4.5.1	Global comparison . . . . .	45
4.5.2	difference for different model architectures . . . . .	46
<b>5</b>	<b>Conclusion</b>	<b>48</b>
<b>6</b>	<b>Bibliography</b>	<b>49</b>
<b>A</b>	<b>Schedule</b>	<b>51</b>

# List of Figures

1.1	Images of the examined carbide tool inserts. The red square marks the area of interest for the wear prediction. . . . .	1
1.2	Milling tool with one insert in place. . . . .	2
1.3	Illustration of a good and a worn insert. . . . .	2
2.1	Construction of the Tool insert with coating. Figure from (Gu et al., 1999) . . . . .	6
2.2	Two wear types: (a) Step between substrate and coating. (b) flat wear not perpendicular to the rake face Figure from (Gu et al., 1999) . . . . .	6
2.3	Dense block with 5 layers. (Huang et al., 2017) . . . . .	8
2.4	a building block for Resnet architecture . . . . .	9
2.5	Summary of Resnet 18 architecture . . . . .	10
3.1	First design of a tool holder oriented at an angle of 40° to the camera. . . . .	13
3.2	Desk lamp setup without shadowing the background. . . . .	13
3.3	First design of wheel holder with inserts mounted in place. . . . .	14
3.4	Overview of second wheel holder . . . . .	14
3.5	Overview of the camera mount system. . . . .	15
3.6	Desk lamp setup for initial testing. . . . .	16
3.7	Desk lamp setup with shadowing the background. . . . .	17
3.8	White LED strips test circuit. . . . .	17
3.9	Visualisation of the reflections from the LED strips into the camera. . . . .	18
3.10	Example of initial dataset . . . . .	19
3.11	Marked side of an insert. . . . .	19
3.12	Bulled mark marked on the top view of an insert. . . . .	19
3.13	Comparison of the same insert 19 from batch 5 the side with marking. . . . .	20
3.14	Camera setup for testing the camera angle. . . . .	21
3.15	Side camera view for LED 8 through 10 listed from left to right. . . . .	22
3.16	Top camera view for LEDs 5 through 7 listed from left to right. . . . .	22

3.17 Fully lit example for insert 5 from batch 3 . . . . .	24
3.18 Red, green, blue and white lighting on insert 5 from batch 3 LED strip one. Left to right respectively . . . . .	24
3.19 Red, green, blue and white lighting on insert 5 from batch 3 LED strip two. Left to right respectively. . . . .	24
3.20 Errors in the Birthday dataset. . . . .	25
3.21 Overview of different insert types. . . . .	26
3.22 Overview of setup for the Spaghetti dataset. . . . .	27
3.23 Example images for red and white light for spaghetti dataset. . . . .	27
3.24 Program sequention for testing different setups. . . . .	28
3.25 Class distributions for the inserts used for the algorithm training and testing . . . . .	30
4.1 Example of second handmade dataset batch 3 insert 4. . . . .	33
4.2 Test, train and validation accuracy for five best runs with second handmade dataset. .	34
4.3 Test accuracy for different model architectures with maximum and minimum values. .	35
4.4 Best validation accuracy for different model architectures. . . . .	36
4.5 Validation and test accuracy box plot for different model architectures with or without transfer learning (TL). . . . .	37
4.6 Parameter influence on validation accuracy. . . . .	37
4.7 Test, train and validation accuracy for the five best runs based on their test accuracy scores. . . . .	39
4.8 Test accuracy . . . . .	40
4.9 Best validation accuracy for different model architectures plotted in relation to the amount of epochs. . . . .	41
4.10 Test accuracy for models trained with and without transfer learning. In picture a) a general overview of all performed tests. in picture b) a comparison between different model architectures. Picture c and d provide the differences during training for validation accuracy and training accuracy respectively. . . . .	42
4.11 Test and validation accuracy for a different led color. . . . .	43
4.12 Parameter influence on validation accuracy. . . . .	43
4.13 Images with predictions from the spaghetti dataset with white lighting. . . . .	44
4.14 Images with predictions from the spaghetti dataset with white lighting. . . . .	44
4.16 Comparison for test, train and validation accuracy for different datasets. . . . .	45
4.17 Difference in test accuracy for different datasets trained with different model architectures. . . . .	46
A.1 Schedule for semester 2. . . . .	53

# List of Tables

2.1	Parameters and depth in layers for every model accuracy . . . . .	10
3.1	Explanation for the image names for the second handmade dataset. . . . .	20
3.2	Overview of folder structure. . . . .	30
4.1	Confusion matrix for tests of the second handmade dataset. . . . .	34
4.2	Best validation accuracy for different deep learning architectures. Sorted on best validation accuracy in column 1 and sorted on test accuracy in column 2. . . . .	36
4.3	Validation accuracy plotted for best model per architecture. In first column sorted on best validation accuracy, in second column sorted on best test accuracy. . . . .	41
4.4	Validation accuracy for the white part of the Spaghetti dataset. First column is sorted on best validation accuracy, second column is sorted on best Test accuracy. . . . .	46

# List of Acronyms

<i>GPU</i>	Graphics Processing Unit
<i>CNN</i>	Convolutional Neural Network
<i>SSD – CNN</i>	Small Data Driven Convolutional Neural Network
<i>TL</i>	Transfer Learning
<i>WC</i>	Wolfram Carbide
<i>SGD</i>	Stochastic Gradient descend
<i>wandb</i>	Weights and Biases

# Chapter 1

## Introduction

The steel industry is a big industry in the world and provides 2.5 million jobs across Europe according to the European commission. Due to high labour costs in western Europe and more specific in Belgium where they are 40% above the average of Europe, production and manufacturing costs should decrease to be able to compete with other companies around the world.

For this thesis we focus on metal carving which is done by spinning a spindle over the metal work piece that carves away material. This spindle holds inserts as seen in Figure 1.1 where a top and side view of the insert are provided. A full assemble of the tool and the insert can be found in Figure 1.2. The inserts are kept in place with a screw for easy removal and replacement and are made of different compositions of Wolfram carbide. After creation they get a coating on the outer layer to provide extra strength and more capabilities for cutting different materials.



**Figure 1.1:** Images of the examined carbide tool inserts. The red square marks the area of interest for the wear prediction.

The cutting part on the corner of the insert - as marked on Figures 1.1a and 1.1b with a red square - will wear when the work piece is carved and will result in a bad finish on the actual product. This can be referred to as surface roughness. To prevent the formation of surface roughness the tool inserts must be replaced before they are worn to much. For this replacement there are two policies used in the industry today:

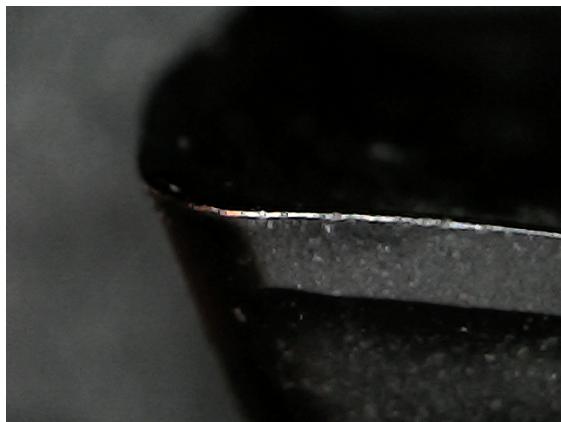


**Figure 1.2:** Milling tool with one insert in place.

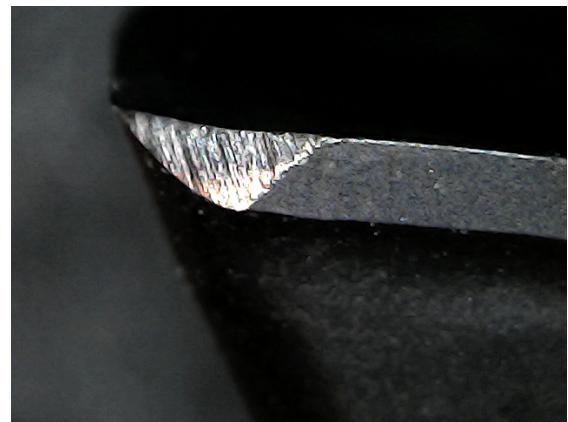
1. Check the inserts when a work piece is finished.
2. Replacing all inserts on set time intervals (e.g. every 30 minutes).

Although option number one will optimize the lifespan of the inserts, it is very time consuming and labour intensive. Option number two is less labour intensive and thus less time consuming. On the other hand will it produce a lot of waste in those inserts when incorporating safety levels for the wear.

What does a worn insert look like? This is shown in Figure 1.3 which is a close up from the area marked with a red square from Figure 1.1b. Figure 1.3 gives examples of a barely used insert and one of an unusable insert due to the high wear.



(a) Barely used carbide insert.



(b) Worn carbide insert.

**Figure 1.3:** Illustration of a good and a worn insert.

The goal for this master thesis is to create a direct measurement method to quantize tool wear on these inserts. From this goal the research questions can be extracted:

- "How can tool wear on carbide inserts be quantized by using a direct (vision based) method?"
- "How can a vision set up be implemented in a production line?"
- "What should the vision set up look like?"
- "What computer vision algorithms perform best for this task?"

For this report however only the foundations are laid where a highly adjustable setup is created as well as some datasets and an algorithm to measure the performance of a specific setup.

The problem will be split up into two main parts:

- Creating a camera setup to capture images of the inserts.
- Create a vision algorithm to process the images and output the tool wear.

We start with a study on the "state of the art" in solving similar problems in chapter 2. The construction and implementation of our own method will be discussed in chapter 3. Results and datasets will be handled in chapter 4. And in chapter 5 we draw conclusions and the research questions are revisited.

## **Chapter 2**

# **Literature review**

### **2.1 State of the art**

In this research area there are a lot of enhancements taking place. The industry 4.0 makes it easier to collect data and process this into usable numbers. With this indirect measured sensor data a lot of research is done on the prediction of tool wear. For example Ma et al. (2020) uses cutting force as input for a Convolutional Neural Network (CNN) which predicts the tool wear. Li et al. (2013) proposes a setup which will detect the tool wear in-line and creates an overview of the tool life with three different tool wearing categories namely nose wear, flank wear and crater wear. The wear is displayed for different machining times. Li et al. (2013) also creates an overview of tests on different materials and different coatings, this gives the reason why it is important to detect tool wear in an early stage to produce high grade products.

Cerce et al. (2015) provides a way to measure tool-wear with a 3D laser profile sensor. This would be more accurate since more data is available to the algorithms. Here tool-wear is divided in two categories: premature tool failure and progressive tool-wear. The premature tool failure "mostly occurs as sudden and unpredictable breakage of the cutting edge" these types of error's won't be detected. The progressive tool-wear on the other hand is easier to predict and measure. Here the inability of measuring wear profiles in depth is the main disadvantage of direct measuring methods. The results of this paper are really good, they detect the numbers on the crater wear and nose wear of the tool. This with an accuracy of 1 micrometer.

A remarkable study is made by Pagani et al. (2020) who uses the chips cutoff by the tool to predict the tool wear.

Although many researchers choose for indirect measurement methods there are some that use direct methods like Ambadekar and Choudhari (2020). They use a microscope to perform off-line tool wear classification in three categories. What they provide is a very similar research as the one done for the setup verification model. We will go even further than classification by also predicting the exact flank wear in micrometer. The architecture used by Ambadekar and Choudhari is Resnet 50 which will also be tested to compare with this study findings. The accuracy achieved is 87% for the three classes.

Schmitt et al. (2012) describes a way of using machine vision to inspect flank wear on cutting tools. The process they use is very labour intensive and should be redone when inspecting a new tool. Their steps are "image acquisition, tool edge detection, highlighting wear region, feature extraction, wear type classification and finally wear measurement." In our paper we will try to make this process a lot simpler by using deep neural networks which will be trained on different tool types. We will need more labelled data to be able to perform such a task which may be expensive to create. An accuracy of 7.5 micrometer is achieved.

## 2.2 Information on the process

Our proposal to address this problem of quantizing tool wear consists of using light to reflect on the worn part of the insert into the camera. The image that will be outputted from the camera will then be used to train a neural network that will actually quantize this wear based on the reflection. For this reason we will first look into the characteristics of the insert including its reflection. After this the used neural networks will be explained for a better understanding of how they work.

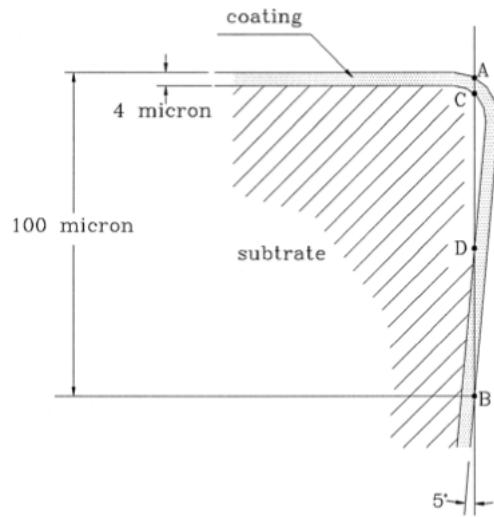
### 2.2.1 How does the tool wear?

The base material of the insert will mostly be Wolfram carbide (WC) for its strength and heat resistance. The same material can also be referred to as tungsten carbide. This base material is covered in a coating to provide extra strength and durability. Gu et al. (1999) Creates an overview of the different wear types with or without coating and the durability under different testing conditions. Figure 2.1 shows how the material is coated and not yet used. The wear of the tool begins with wear on the coating and goes right through the the coating in the base material of the tool. Figure 2.2 Shows the two types of wear seen on the surface of the inserts. Figure 2.2a shows the result of a slow spinning mill where the tool chips off. Figure 2.2b shows the wear on an insert which was used under high speeds and where the worn area is flat but not perpendicular to the rake face. This will have to be considered for the light set-up to be able to handle both wear types.

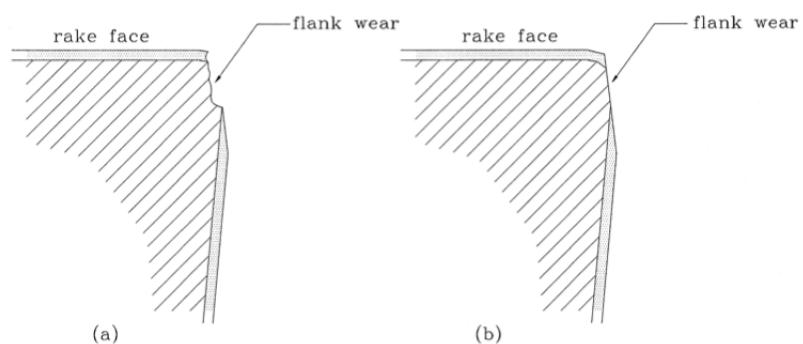
### 2.2.2 Surface of the insert

WC typically has a grain size of  $0.5 \mu\text{m}$  to  $2 \mu\text{m}$  which is bonded with cobalt that acts as cement between the WC grains. This declares the name of cemented carbides.

The next paper is good to get an overview of the light reflection seen in different types of materials and even multi layered tools where Caicedo (2019) researched the reflectance of coating layers on tool inserts. The study suggests that that the best reflection occurs at the highest wavelength. This translates to the visible color red and means that the reflection should be the optimal when the lighting is on the top of the spectrum of the camera.



**Figure 2.1:** Construction of the Tool insert with coating. Figure from (Gu et al., 1999)



**Figure 2.2:** Two wear types: (a) Step between substrate and coating. (b) flat wear not perpendicular to the rake face Figure from (Gu et al., 1999)

### 2.2.3 Vision algorithms

The problem of tool wear quantisation on carbide inserts is not much researched yet. For this reason there are very little relevant datasets available so the full dataset must be self-created, therefore we will only have a small dataset that consists of a little less than 300 images to work with. Starting from a very small dataset for the creation of a vision set-up, it is necessary to choose a good deep learning network architecture that performs well with little data. This constraints the choice of architectures to the ones with very little training parameters. Khan et al. (2020) extensively describes most available CNN architectures a few of them are used and repeated underneath for a better understanding of the architectures.

The network will be used to confirm whether a setup is good or not therefore a classification of the images is performed where an image can belong to three classes.

In this section a brief background will be given for the used software and computer vision algorithms. Starting with a listing of different frameworks and continuing with an explanation for the used neural network architectures.

#### 2.2.3.1 Frameworks

There are many frameworks to perform computer vision tasks and make it easy for the user to get a lot of results in a short amount of time. In what follows a list of the frameworks used in this thesis is given.

- Fast.ai (J, 2020)
- Keras (I, 2020)
- PyTorch (Adam et al., 2020)

Basnet et al. (2019) provides an overview of different deep learning frameworks, a short summary is given below. Fast.ai provides a very high level programming experience designed to make deep learning very easy. We found this to be too high level which would affect the customizability of the algorithms. Keras is also a high level framework which works with very little programming and is designed for experimenting. PyTorch leaves a little more room for customization. This is the reason PyTorch was used for the most of this thesis.

We will continue with a documentation on the used neural network architectures starting with VGG.

#### 2.2.3.2 VGG11.bn

VGG is a network architecture found by Simonyan and Zisserman (2015). It was designed to get better results at the famous Imagenet classification task and does this by using a deeper structure with smaller convolution filters. The original architecture was designed for large scale image classification tasks but will be used for a small dataset in this thesis where a variant of VGG is used with 11 layers instead of the original 19 layered architecture.

The bn stands for batch normalisation which will normalize the output of previous activation layers before passing it on to the next layers. The normalisation will be calculated on every batch instead of the whole dataset.

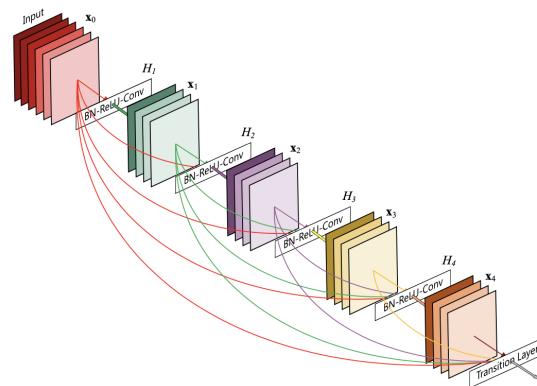
### 2.2.3.3 Alexnet

Alexnet by Krizhevsky et al. (2017) was one of the first to implement deeper structures in the before known CNN architectures. The implementation of deeper network led to over fitting which was countered with including dropout in the fully connected layers. This makes the model more stable even with more layers. In the time Graphics processing units (GPU's) where not as developed as today so the initial architecture consisted of two parallel paths which could run on two separate GPU's.

### 2.2.3.4 Densenet 121

DenseNet (Huang et al., 2017) tries to solve the problem of vanishing gradients where information is lost in the back propagation due to too much layers. This is also the problem which ResNet tries to address. ResNet used skip connections to create a bigger information flow to the first layers. DenseNet has a similar aproach where information flow is optimized by connecting every layer to all subsequent layers. This will create better feature propagation through the layers and increase feature reuse.

In Figure 2.3 a schematic is given of a dense block where the connections are clearly visible. Although we implement DenseNet with 121 layers, it only has 7.9 million parameters to be trained which is an enormous reduction compared with ResNet which has 11.7 million parameters for 18 layers.



**Figure 2.3:** Dense block with 5 layers. (Huang et al., 2017)

### 2.2.3.5 SqueezeNet

If the amount of parameters must be reduced, we can look at implementing SqueezeNet which is created by Iandola et al. (2017). SqueezeNet drives the parameter reduction to an extreme where it achieves similar results to AlexNet using 50 times less parameters (1.2 million). The network does this by:

1. Reducing the size of convolution filters from  $3 \times 3$  to  $1 \times 1$ .
  2. Decreasing the number of input channels by using so-called squeeze layers.
  3. Using delayed downsampling to increase the accuracy without adding much parameters.

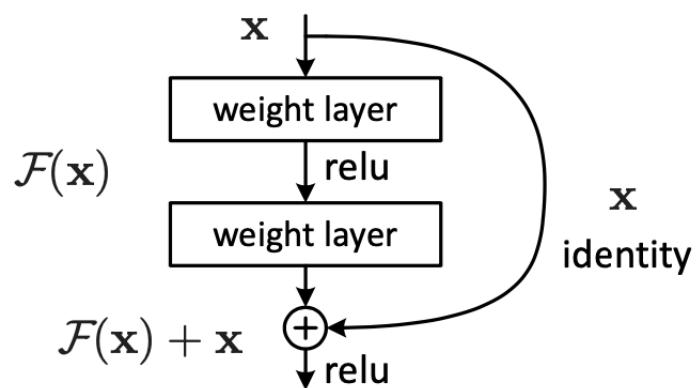
These strategies are incorporated in Fire modules that will be used to form the neural network.

### 2.2.3.6 Resnet18

First we define what Resnet (He et al., 2016) means, after that we dig deeper into the Resnet18 architecture.

Resnet is a deep residual neural network that can reach deeper networks than its predecessors without affecting the complexity of the network. What means the amount of parameters or the complexity of the training will not increase much when extra layers are added to the model. The depth of the network plays an important role in the performance.

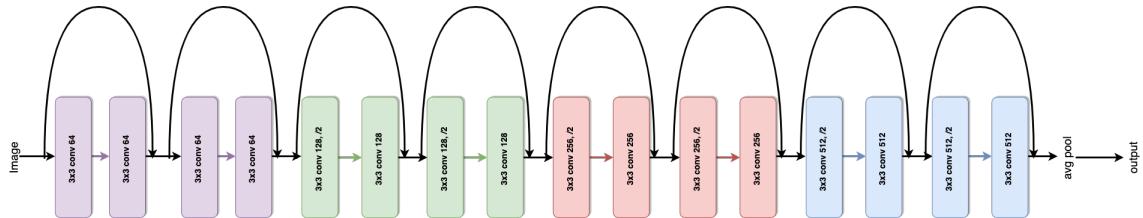
Figure 2.4 gives an overview of the important building block that lead to the success of Resnet. Rather than just stacking layers on top of each other, they provide shortcut connections that skip one or more layers. These shortcut connections perform identity mapping where the output of the previous blocks are added to the output of the stacked layers.



**Figure 2.4:** a building block for Resnet architecture

Figure 2.5 displays a summary of all blocks in the network architecture of Resnet 18. The arrows represent shortcut connections over a few layers. The blocks represent a  $3 \times 3$  convolution layer.

where a 3 by 3 grid goes over the previous output and calculates the product with a weight matrix which is updated during training. These calculations provide a new input for the next layer. The total parameter count for this architecture is 11 million. This seems quite a lot but is actually not much. This will be compared with other network architectures in the next section 2.2.4.



**Figure 2.5:** Summary of Resnet 18 architecture

## 2.2.4 Overview of important numbers

Parameters and depth for all architectures are given in table 2.1. The amount of parameters and the depth define the complexity of the architecture. Where more layers and more parameters increase the complexity.

**Table 2.1** Parameters and depth in layers for every model accuracy

Model name	parameters (million)	depth (layers)
Resnet 18	11.7	18
VGG11.bn	123.6	11
Alexnet	61.1	8
Densenet 121	7.9	121
Squeezezenet	1.2	10

## 2.2.5 Other architectures for small datasets

Very little datasets are available for this problem so all data used in this thesis will be self created. For this reason there are very few images to train and test algorithms on. In what follows are some interesting network architectures given for small dataset image classification. These can later be adjusted to perform image regression tasks.

A first architecture is proposed by Chandrarathne et al. (2019). They compare training a five layered neural network from scratch using transfer learning on the Imagenet dataset. The findings were that transfer learning could increase the testing accuracy by 10% on small datasets.

Xu et al. (2019) proposes a so called SDD-CNN what stands for small data driven convolution neural network. This was used for the inspection of roller bearings. A preprocessing method called label dilation is used for dataset imbalance because there is a big difference in amount of positive and negative examples. This label dilation method generates random classes where the amount of

items per class after dilation equals the amount of items in the biggest class before dilation. For the roller bearings there are 300 samples per class. After preprocessing, the images are augmented to extend the dataset in a controlled way using semi-supervised data augmentation. This process is done by cropping the worn area of the bearing rather than center cropping. Then four networks are trained with the received data. These networks are:

1. Squeezener v1.1
2. Inception v3
3. VGG-16
4. ResNet-18

The findings of the research are that inception v3 performed best for this small dataset when used with transfer learning.

# **Chapter 3**

# **Implementation**

The implementation consists of three main parts. First the camera setup will be discussed which will be used to create the datasets in section 3.2. The images from the dataset will later be processed using convolutional neural networks which will be covered in section 3.3.

## **3.1 Camera setup**

As stated in the previous chapter, a new dataset must be created with images of worn tool inserts. For the creation of the dataset a setup is created that consists of three different parts:

1. A tool holder which will provide the same position of the tool insert for every picture.
2. A mount for the microscopic camera which is stable and easy to finetune.
3. Some lighting solution that provides the right angle of light to the tool insert and reflects into the camera.

These parts will be handled in the following sections starting with the design of a tool holder

### **3.1.1 Design of a tool holder**

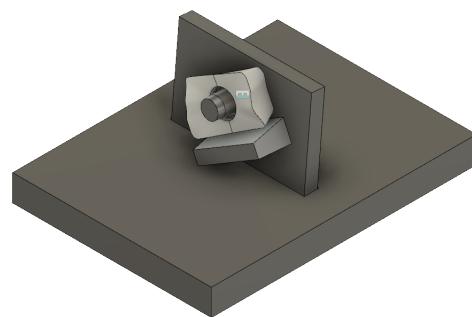
The tool holder will be the most important part of the setup since this part will be holding the tool insert where every imperfection will be seen on the output images. The requirements for this holder are:

- The inserts should be easy to remove and to replace.
- The holder may not block the view from the camera to the insert.

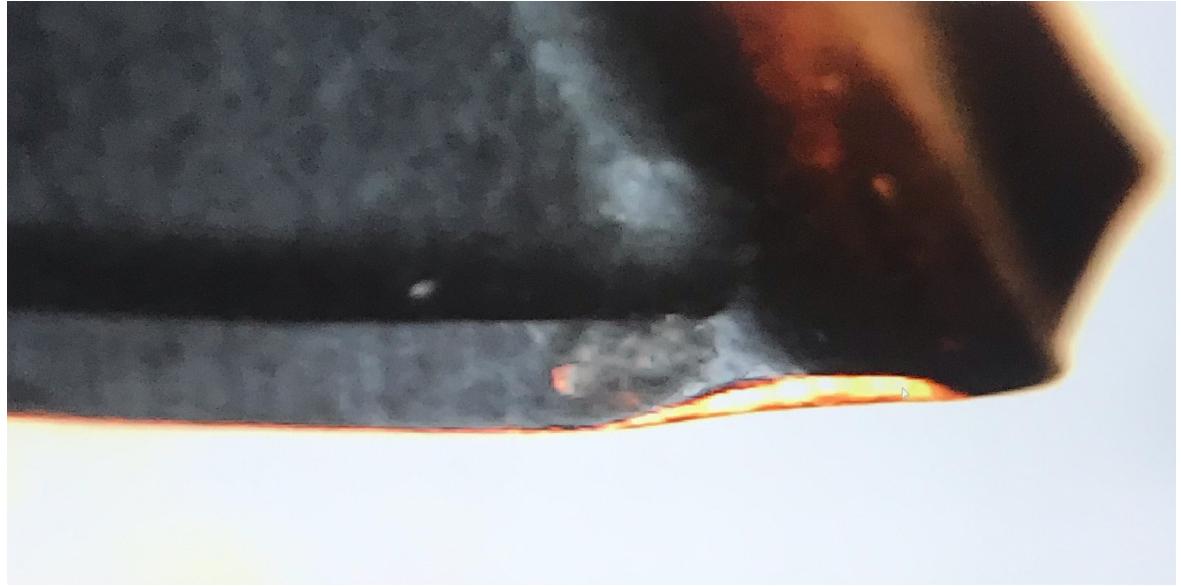
Since this is a design process there are a few iterations made on these tool holders. The most important are observed in the succeeding sections.

### 3.1.1.1 First tool holder

All tool holder prototypes were 3D printed. A first holder was made to check how the lighting would be transferred to the camera. This was put in a simple setup with a desk lamp and a socket which kept the insert at an angle of 40°. The socket can be found on Figure 3.1. This showed potential in the current setup as seen in Figure 3.2. The tool wear is nicely indicated with the reflections of the light.



**Figure 3.1:** First design of a tool holder oriented at an angle of 40° to the camera.



**Figure 3.2:** Desk lamp setup without shadowing the background.

### 3.1.1.2 Wheel holder

?? The previous first tool holder was good for taking a sample picture of an insert but isn't scalable to take pictures of a few hundreds of inserts. To be able to quickly create a lot of photos in a

consistant manner, a wheel is designed to mount 20 inserts at once. With the use of a stepper motor, a fixed camera and lighting setup, the process of taking images is automated for every 20 tools. The holder is 3D printed so a few wheels can be made to be able to swap the wheels with new inserts for an efficient dataset creation.

The first design of this wheel holder can be seen in Figure 3.3. The whole wheel is printed in one piece. The inserts are kept in place with brackets that go over the cutting edge of the insert. This didn't seem ideal since the sharp edges of the insert would cut in the bracket what made it very hard to remove the tools which didn't comply with the easy removal constraint.



**Figure 3.3:** First design of wheel holder with inserts mounted in place.

For this reason a second wheel was designed which was a little more flexible and made the process of inserting and replacing tools a lot easier by providing a clip which was printed and added to the wheel separately. Figure 3.4a shows the full assembly of this second wheel holder.



(a) Illustration of second wheel holder.

(b) Zoomed illustration with clip and insert.

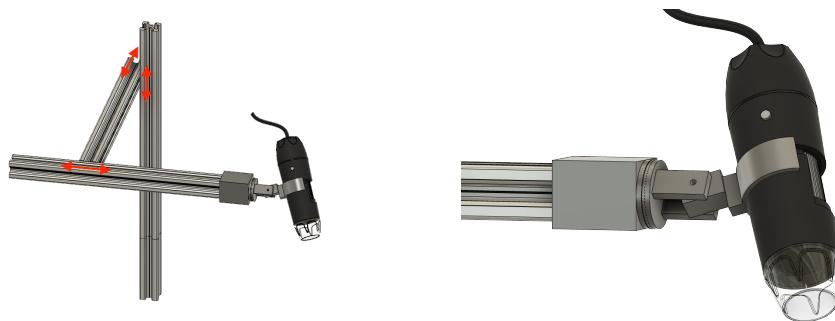
**Figure 3.4:** Overview of second wheel holder

In the wheel there are holes in which the clips fit. The center hole is as big as the hole in the inserts. This keeps the insert from moving when the clip is on. The slads between the holes are made to fit the insert perfectly so this doesn't rotate around the axis of the clip. On the clip there is an extra long middle tube that makes it easier to push the clip with the insert out of the wheel. This can all be seen in a close up on Figure 3.4b.

### 3.1.2 Design of a camera mount

In this section the camera mount will be discussed along the design process of the setup. The camera mount provided by the factory was not up to the task of taking consistent pictures and controlling the camera position precisely so a new camera mount was designed. For this task some 20mm by 20mm aluminium profiles where used as a base for the stand. On that stand an assembly of 3D printed parts where mounted to be able to rotate the camera around two different axis as seen on the close up on Figure 3.5b.

A render of the full assembly of this new camera mount can be found in Figure 3.5a.



(a) A render of the camera mount with the given movement options.

(b) Close up to the rotation and hinge for the camera.

**Figure 3.5:** Overview of the camera mount system.

### 3.1.3 Light configuration

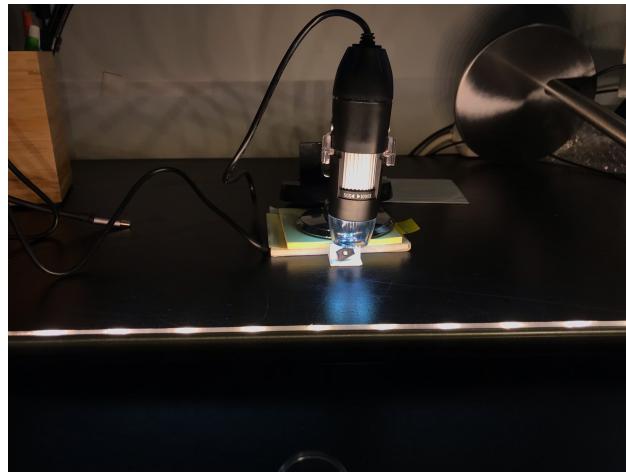
In order to test different lighting options there must be a simple to configure lighting solution. This should be configurable in both color and light direction to create a maximum reflection of the wear area into the camera lens. This is achieved with single LED adressable LED strips.

Multiple different light sources where tested as listed underneath. Some worth noticing are declared after.

1. Warm white LED desk Lamp
2. Long hard white LED strips
3. single LED adressable RGB LED strip
4. Multiple single LED adressable RGB strips
5. USB microscope camera light

### 3.1.3.1 Desk Lamp Test

As an initial setup to test the camera and be able to see the effects of lighting on the image this setup was created and a desk lamp was used for the lighting as seen in Figure 3.6. On this Figure we can also see the initial tool holder from section 3.1.1.1.



**Figure 3.6:** Desk lamp setup for initial testing.

This setup used the top light of the camera which was good to lighten a whole area to adjust the camera. The worn places where more visible without that top lighting. The desk lamp added to much brightness and didn't leave enough room for adjusting different settings. For this reason other lighting options where explored.

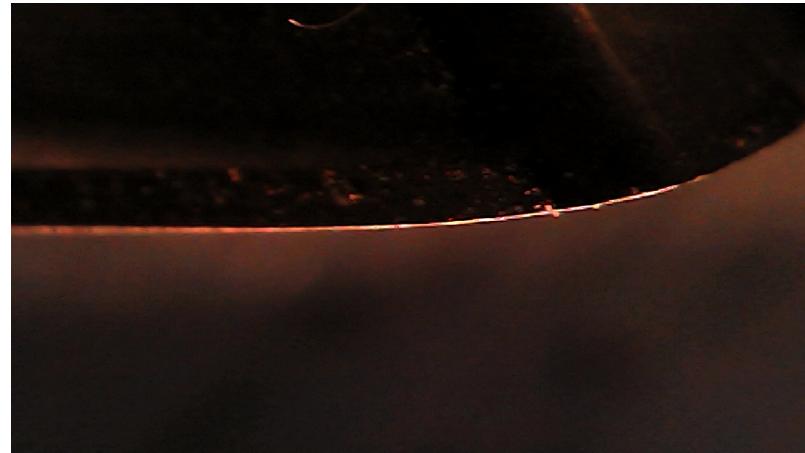
From this setup it can also be seen that the lighting on the background is important. Figure 3.2 shows a picture of the tool with desk lamp lighting without blocking the light from the background. Here we can see that there is a bright white background behind the tool.

The result of this is shown in Figure 3.7 where the light is blocked off of the rest of the tool and only the erroneous part is lightened. This would be a good start for creating a dataset.

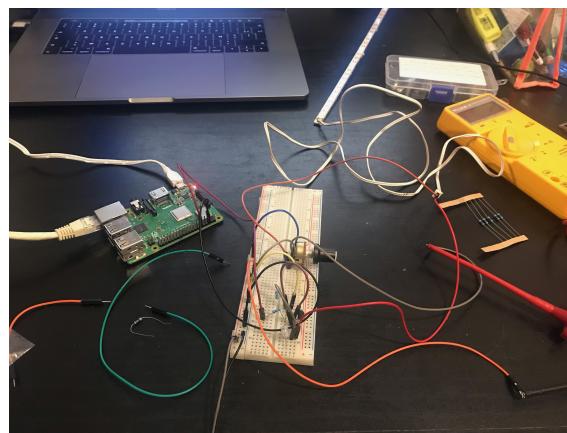
The color of the desk light set a good gradient of bad vs good sets. White areas are worn very hard while orange is not worn that hard. By tilting the lamp in a horizontal way, all the areas where lighted.

### 3.1.3.2 White Led Strips

A second lighting condition created for this setup uses 3 the similar LED strips controllable with a micro controller. For the purpose of this test a small electric circuit is created to control these strips since they use more power than the raspberry pi can deliver (rpi, 2020). An overview of this test circuit can be found in Figure 3.8. A potentiometer was used to control the voltage over the LED strip in order to control the brightness.



**Figure 3.7:** Desk lamp setup with shadowing the background.



**Figure 3.8:** White LED strips test circuit.

### 3.1.3.3 Addressable Colour Changeable LED Strip

A third option of lighting is playing with the colours of the light. To archive this a setup will be created with a single addressable light strip where the colour and LED can be freely chosen. To assign a colour which works best; a study is made to find the wavelengths where the light reflects most on the used materials of the tool. This can be found in section 2.2.

From all these settings the single addressable LED strip was chosen due to the amount of options this creates. This choice is documented in future tests which will be discussed in section 4.4.5. The LED strips were mounted on metal strips which provided high adaptability. This made it possible to get lighting on different parts of the tool insert what makes later research easier. In Figure 3.9 the light configuration is displayed. A lot of options can be fulfilled with this set-up. Each LED strip can rotate and fifteen LEDs can be individually set to a colour. The arc keeps a consistent distance between the LED and the insert.



**Figure 3.9:** Visualisation of the reflections from the LED strips into the camera.

## 3.2 Datasets

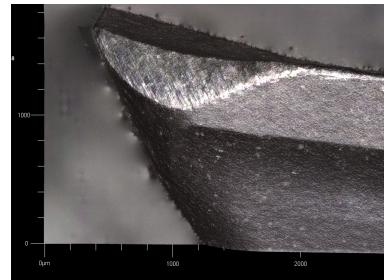
A separation is made between hand made datasets and automated datasets because they take a very different approach and produce very differing results.

### 3.2.1 Handmade datasets

The following datasets where produced using a microscopic camera to take pictures of single inserts all placed under the camera by hand. The initial dataset is produced by Sirris.

#### 3.2.1.1 initial dataset

The dataset given where taken with a microscopic camera at Sirris. These images were in good lighting conditions for the measurement, but had a lot of extra "unwanted" features on it. The background was very like the wear and the rest of the tool insert.



**Figure 3.10:** Example of initial dataset

Every insert that was measured had a little mark on one side to mark the a and b side of the insert. An example of this mark is given in Figure 3.11. Sadly the information about what the marker meant is lost. To find the corresponding values, the inserts were once again put through a microscopic camera and the new pictures were compared against the old ones. This will be discussed in the following section.



**Figure 3.11:** Marked side of an insert.

### 3.2.1.2 Second handmade dataset



**Figure 3.12:** Bulled mark marked on the top view of an insert.

A second dataset was made to compare the pictures with the previous dataset. This is done to verify the images and the results and to determine what the markings on the inserts meant. This

**Table 3.1** Explanation for the image names for the second handmade dataset.

symbol	explanation
s	side with marker line
n	side without marker line
batch number	number specified on the box in which the inserts are kept
insert number	number specified inside the box; this goes from 1 to 10 per batch

dataset handles the first 5 batches labeled with 00x. For this dataset a setup was made where the insert is placed on its side on a black background with the microscope camera vertical above the insert. Only the light from the camera was used.

During the creation of this dataset the inserts were labeled with bullet or no bullet. What we mean by bullet is marked on Figure 3.12. Now all labels have the following information:

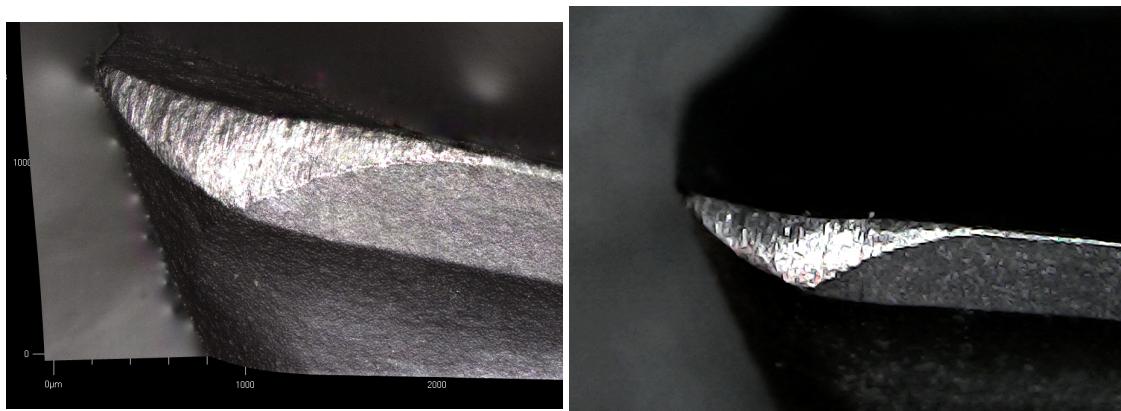
- A value of tool wear.
- Whether the side of the insert is marked.
- If a bullet mark can be found on that side of the insert.

Later on this bullet can be used as an universal identifier for the sides of the inserts. The most information is also stored in the name of the image. This is indicated in table 3.1.

The naming of the dataset is as follows:

b\_<batch number>.p\_<plate number>\_<identifier of side>

When all data images were created, we took a look at the pictures from the dataset as specified in Figure 3.13. It can now be confirmed that the two images correspond and that the 'b' side of the inserts is the side with the marker line.



(a) Image from the first initial dataset.

(b) Image from the second handmade dataset.

**Figure 3.13:** Comparison of the same insert 19 from batch 5 the side with marking.

### 3.2.1.3 second initial dataset

The second initial dataset was made with the inserts from batches 11 to 19 labelled with 01x. The images were taken with the same microscope as the first initial dataset but instead of photographing only the one insert at a time, two inserts are photographed per shot. Since the photo's where labelled with two inserts at a time, the images taken where not relevant for the data processing and were not saved.

## 3.2.2 Automated datasets

The handmade datasets are not scalable for a hundred or even a thousand inserts so a way of automatically creating datasets was researched in 3.1. When all building blocks for an automated set-up where created, a program was written to control the set-up and save the taken images in a logical order. This process took a lot of test tries to find out a good camera position which will be handled first. After that the two created datasets will be discussed.

### 3.2.2.1 Camera position

In this section the best camera position will be defined for the creation of the dataset. Top view as well as side view will be tested.

**Side camera position** The first four plates of batch 004 are used to test the side view camera position. The lighting is red light coming from the addressable LED strips that shine from different directions as specified in section 3.1.3.3. The practical implementation of the setup can be seen on Figure 3.14.



**Figure 3.14:** Camera setup for testing the camera angle.



**Figure 3.15:** Side camera view for LED 8 through 10 listed from left to right.



**Figure 3.16:** Top camera view for LEDs 5 through 7 listed from left to right.

During the creation of the dataset, there were some problems with the microprocessor that controlled the lights. It would take one full second to process a command sent by the computer connected to it. This resulted in an unsuccessful test because the LEDs didn't turn on when the photo's were taken. The produced images were all black with a little shadow of the insert caused by the polluting light from the room.

The error was temporarily resolved by putting a delay before sending a command to the micro processor to provide the wanted lighting conditions. Although this was a very long process the results are way better than the ones from the first take on this camera set-up.

These can be found in Figure 3.15. The insert is lighted with two corresponding LEDs coming from the two LED strips described in 3.1.3.3. On the left is the result for LED number 8; in the middle LED number 9 and on the right LED number 10. This lightens the worn area very good. Although the background is lighted as well and makes it harder to only see the worn area. Nevertheless we can build the first dataset with this. First the top camera view is tested.

**Top camera position** A second test was conducted with the camera mounted a little more to the top of the inserts. This made the reflection from the worn area to the camera better.

For this test the issue with the micro processor communication was resolved by changing the input delay from the serial input reader. The resulting pictures can be found in Figure 3.16. Like on the previous test one picture was taken for every two LEDs of the strip with red light. This was done for batch number 4 insert 3 this time with LEDs 5,6 and 7 turned on since the position of the camera relative to the LEDs changed.

On this data we can see the LEDs going up on the insert wear area. Which is what we tried to

obtain. Now the LEDs are mapped to specific positions on the inserts and the amount of LEDs that need to be turned on for taking pictures can be reduced so no extra time is wasted.

### 3.2.3 Created datasets

The full datasets will be discussed in this section where we start with a brief explanation of the dataset and proceed with a view of the set-up after which the results are illustrated.

#### 3.2.3.1 Birthday dataset

The first full dataset is the Birthday dataset which was meant to take a picture of every insert with all possible lighting settings. After the first 6 batches the output was inspected and the process was halted. The process will be discussed in the next paragraphs.

**Dataset explanation** The birthday dataset was created on 27/11/2020 with a part of the given inserts for every possible colour and led setting where pictures are taken from inserts using the two separate LED strips where every LED is turned on separately. This is done for white, red, green and blue colours as listed below. This results in a total of 91 images per insert side.

- 1 picture with all LEDs on white
- 30 pictures with red lighting; 15 of LED strip A and 15 of LED strip B
- 30 pictures with blue lighting; 15 of LED strip A and 15 of LED strip B
- 30 pictures with green lighting; 15 of LED strip A and 15 of LED strip B

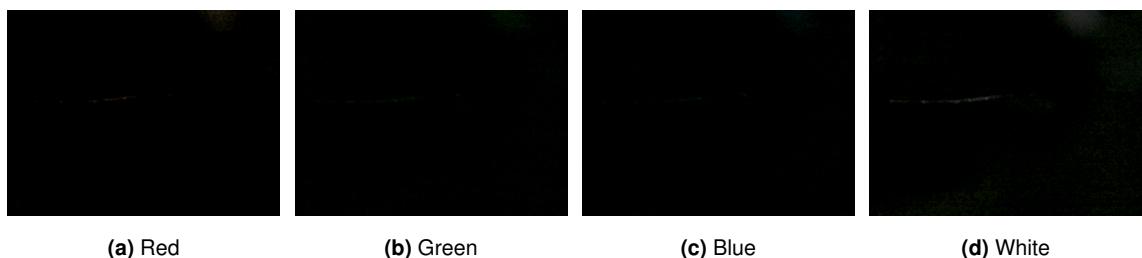
**Set-up** Two ledstrips where mounted and pointed at the photographed insert. The brightness is set to 80% for all LEDs to make sure to not clip against the saturation values for the pixels. The inserts where attached to the wheel from ?? with black clips 3D printed with PETG. This was chosen above white clips to lower the light reflections into the camera. This was a problem in previous set-ups. Also the sturdy camera mount from section 3.1.2 was used to get better notice of the placement of the camera opposed to the insert.

**Results** The image taking process took about 1 minute per insert because of the amount of pictures taken. Since time was limited and the set-up wasn't yet perfect we decided to only run 60 inserts. These where from batches 1 to 11.

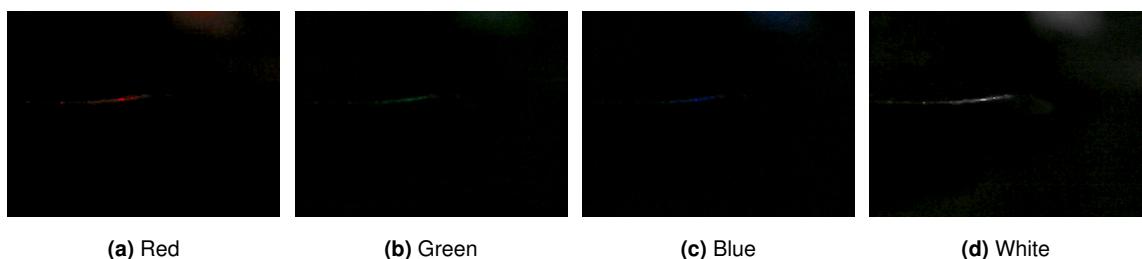
Sample pictures of this dataset can be found in Figures 3.18 and 3.19. Where 3.18 are images from LED strip 1 and 3.19 are images from LED strip 2. We can see that LED strip 1 wasn't configured well where it resulted in all black pictures. For LED strip one we can asses the use of colours. Only white and red are really visible, blue and green are not visible on the images thus these colours won't be used for later research.



**Figure 3.17:** Fully lit example for insert 5 from batch 3.

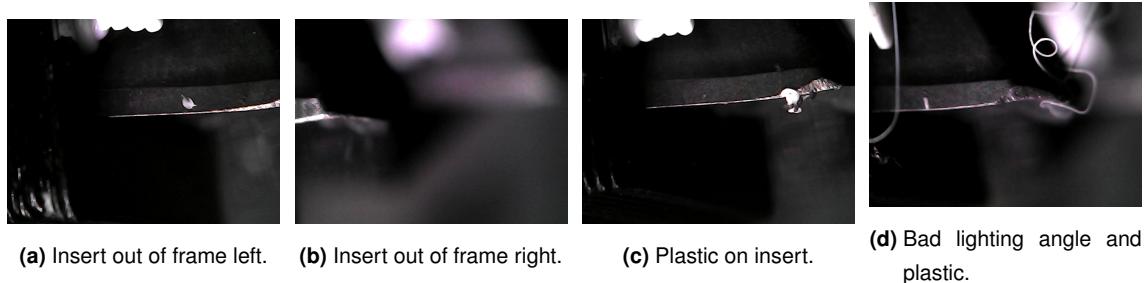


**Figure 3.18:** Red, green, blue and white lighting on insert 5 from batch 3 LED strip one. Left to right respectively



**Figure 3.19:** Red, green, blue and white lighting on insert 5 from batch 3 LED strip two. Left to right respectively.

In this dataset, there are some pictures unusable, an overview of the errors is given in Figure 3.20. As some worn areas are not even in the frame. Some to the left side (Figure 3.20a) and some to the right side (Figure 3.20a). On 3.20c we see a little plastic that was still on the insert which blocks the view of the wear area. This must be cleaned before conducting a new dataset. On 3.20d the lighting is not correctly reflected into the camera.



**Figure 3.20:** Errors in the Birthday dataset.

### 3.2.3.2 spaghetti dataset

After the Birthday dataset a new dataset was created implementing the things learned from it. The amount of LED's driven is reduced to 5 LED's. Where LED 6 to 11 is used to lighten the inserts.

Only colours white and red are used which was also discussed in section 2.2.2 that the red colour could have an influence on the reflection of the carbide of which the inserts are made.

**Dataset explanation** For every insert, two pictures are taken. One with LEDs 6 to 11 on red and one with these LEDs white. This was experimentally found to be the best setting for reflecting the light off the worn area and into the camera.

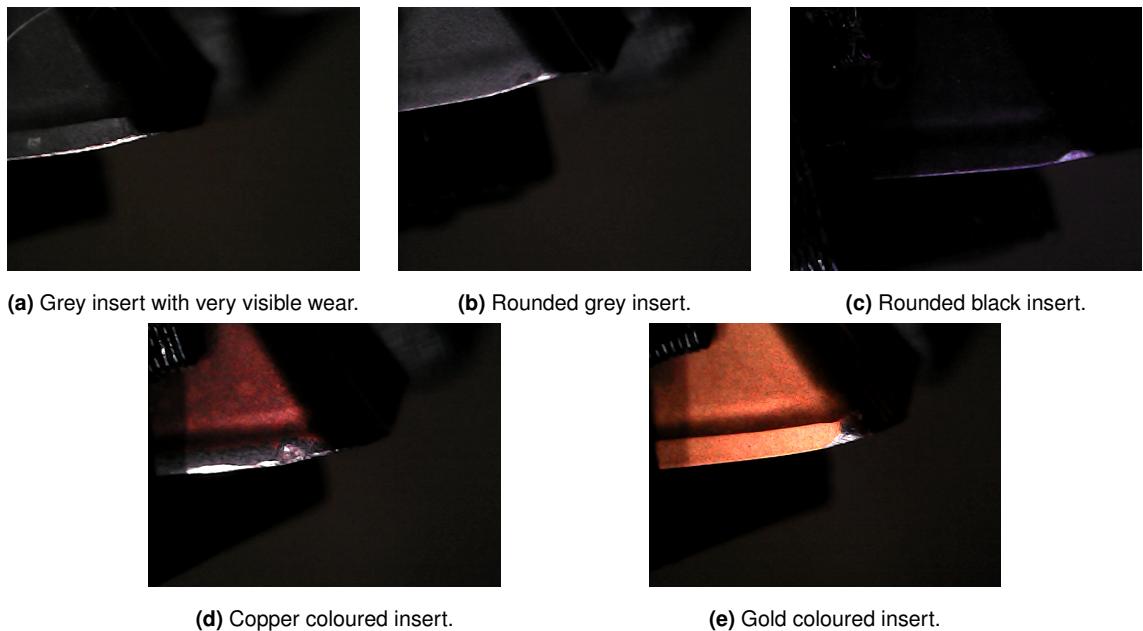
The choice of turned on LEDs is important because turning on a LED to much on the upper boundary will lighten the side of the insert which isn't of much use in this research. Turning on a LED to much on the bottom boundary makes the background very bright which would provide unnecessary information. The images are separated in a folder for every insert named with batch number and insert number: batch\_aaa\_plate\_bbb where aaa is the batch number and bbb is the insert number.

Images are named with their settings, batch number and insert number: b\_aaa\_p\_bbb\_l\_006-011\_color\_bullet.png  
In this name, the following fields are used:

- aaa is the batch number;
- bbb is the plate number,
- 006-011 are the LEDs that turned on at the same time;
- colour is the colour: red or white
- bullet is the appearance of a bullet on the side of the inset and has values of b for bullet or nb for no bullet.

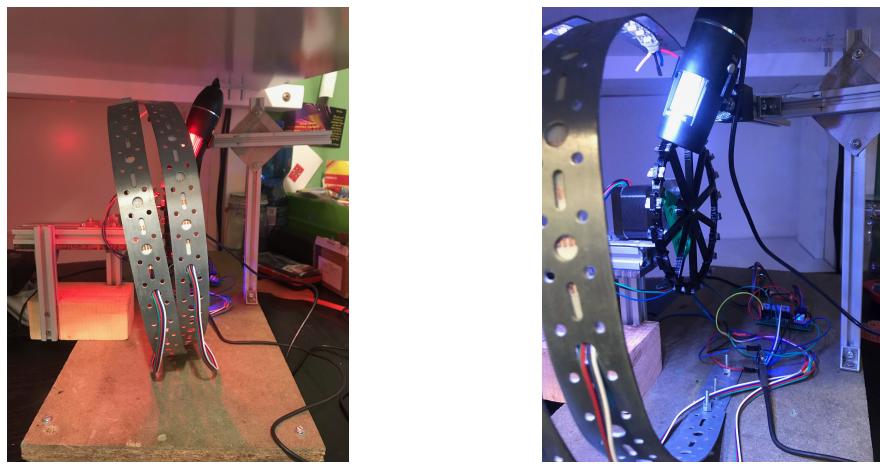
**Different insert types** The dataset inserts consisted of a few different types and coatings. These will be displayed in Figure 3.21. The types are listed underneath, we can distinguish two form factors and four different colours.

- a Grey inserts with very visible wear. These are seen in batches 1 to 5 consistently. This type will be called grey inserts. (3.21a)
- b Rounded grey inserts that have a different shape on the cutting part. (3.21b)
- c Rounded black inserts with a coating that results in way darker pictures. (3.21c)
- d Rounded copper inserts that have a very differing wear to the other inserts. This seems much more brittle than the other inserts.
- e Gold coated inserts with a high reflectance.



**Figure 3.21:** Overview of different insert types.

**Set-up** The set-up used is exactly the same as on the Birthday dataset where the camera is positioned as much to the top as possible as can be seen on Figure 3.22b.. On Figure 3.22a we can see the LED strips are a little bit twisted and are positioned very close to each other. This made the reflection better and should result in better outcome of the algorithm. The room in which the dataset was created was fully dark to have no other light sources influencing the reflections on the inserts.

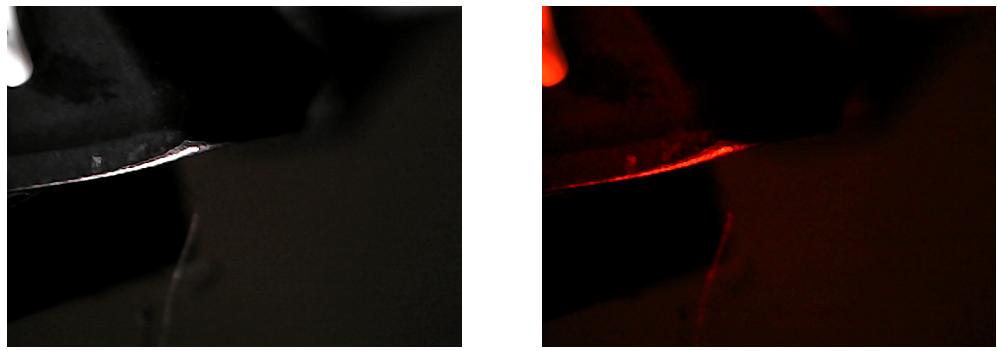


(a) Setup with twisted LED strips.

(b) Test setup camera angle more to top.

**Figure 3.22:** Overview of setup for the Spaghetti dataset.

**Results** Figure 3.23 shows two pictures of batch 3 insert 6. These were lit with red light in 3.23b and white light in 3.23a. Here is a nice wear shown and lighted. However if we zoom in to the picture the top part of the wear is not lighted that well. We can also see a white piece of the insert holder on the image which is providing some extra difficulties. There were also a few images not fully in focus in this dataset. The influence of these difficulties on the output of the neural network will be reviewed in section 4.5.



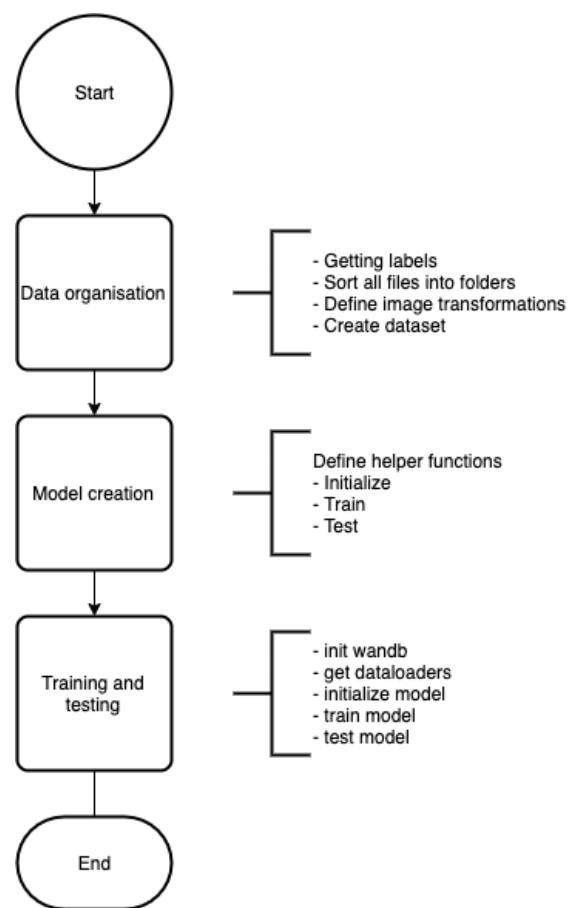
(a) White light

(b) Red light

**Figure 3.23:** Example images for red and white light for spaghetti dataset.

### 3.3 Vision algorithms

To test the camera setup, a classification model is made where inserts will be classified into three classes. The program overview is given in Figure 3.24. This section will be following the path on the figure starting with data organisation, going to model creation and ending with training and testing. The results obtained with these algorithms will be analysed in chapter 4.



**Figure 3.24:** Program sequention for testing different setups.

### 3.3.1 Data organisation

Before anything can be learned the data must be adjusted to provide a proper input to the model architectures. All these models require an input image of 224 by 224 pixels. First the labels are obtained from the labels file. Then the files are sorted into folders based on the class and function. After that the transformations on the images are defined to finally create a dataset.

#### 3.3.1.1 Getting labels

All labels were saved in an Excel file which was exported to the csv file type. The file was loaded using the pandas library for data analysis (McKinney and Team, 2015). This file includes following columns for every insert:

1. Batch number (bn): provides the number of the batch, each batch consists of 10 inserts.
2. Insert number (pn): The number of the insert in the batch in range 1 to 10.
3. Bullet (b): Whether or not the side of insert has a "bullet" mark on it. An example was given in Figure 3.12
4. Folder name: The name of the folder where the images of this insert are saved. This will contain the images for the two sides of the insert.
5. Image name: Name of the image taken for the second handmade dataset. This field doesn't serve for the other datasets.
6. Value: The measured wear value in  $\mu\text{m}$ .

#### 3.3.1.2 Sorting images

The labels from 3.3.1.1 can now be used to divide images into the needed classes. For this task we will iterate over all possible images and decide to what set every image belongs and what class its value represents.

For every image there will be decided whether it goes in the training set, the test set or the validation set. The training set will hold 70% of the images, the validation set will contain 20% of the images. The remaining images will represent the test set which will be 10% of all data.

A random number will be generated between 1 and 3 which will represent one of these sets. According to this number the image belongs to a specific set. When a set is filled so it represents the amount specified by the percentage, the random number generator will only generate numbers 1 or 2 representing the remaining sets and so on.

When the set is declared, the image must be divided into the three classes. This will be fulfilled by comparing the wear value of the image to two set limits. A lower limit of 130  $\mu\text{m}$  and an upper limit of 230  $\mu\text{m}$ . The division is given below:

**Table 3.2** Overview of folder structure.

Train		Test	Validation
Low	Low	Low	Low
Medium	Medium	Medium	Medium
High	High	High	High

- Class 1: Low wear between  $0 \mu\text{m}$  and  $130 \mu\text{m}$
- Class 2: Medium wear between  $130 \mu\text{m}$  and  $230 \mu\text{m}$
- Class 3: High wear between  $230 \mu\text{m}$  and  $1000 \mu\text{m}$

Now the set and class are known so the given image can be saved in a folder. The resulting folder structure is given in table 3.2.

Figure 3.25 shows the distribution of the classes for the first five batches in 3.25a and the distribution for all data in 3.25b. The class distribution should be 1/3 in the best case. The first graph shows a distribution which is close enough to this optimal distribution yet in the second graph we see the distribution is far from even which must be taken into account when analysing the results.

**Figure 3.25:** Class distributions for the inserts used for the algorithm training and testing

### 3.3.1.3 Image transformations

To provide more data from the images we already have, data augmentation is used where the images will be duplicated and transformed before being used as training and validation images. These transformations consist of a rotation between 0 and 4 degrees, a translation in both x and y direction of 0.2 times the width or height respectively and a shear of 0.1 also in both directions. After these transformation the images are rescaled to a 224 by 224 resolution since this is what the model architectures expects. The images are also normalized using the mean and standard deviation values from the Imagenet dataset (Deng et al., 2009). This can be further optimized by calculating the mean and standard deviation of the full dataset ourselves but wasn't done for these tests.

### 3.3.1.4 Dataset creation

With all this done a dataset can finally be created by using PyTorch's ImageFolder function from the torchvision library. (Adam et al., 2020)

## 3.3.2 Model creation

A few helper functions are defined for the model creation. One for initialisation of different model architectures. A second one to train the model and a third to test the model after training.

### 3.3.2.1 Model initialisation

Every CNN architecture has an initialisation where the input and output sizes must be set. This input size will be 224 pixels by 224 pixels for all architectures used for the setup validation. The output size is a vector of size 3 which will store the probability for every class. Since this initialisation step is clustered in one function, more model architectures can be added easily. The following architectures are already implemented:

- Resnet18
- Alexnet
- VGG11\_bn
- SqueezeNet
- Densenet

An initialisation for Inception v3 was also implemented as this was said to be well performing by Xu et al. (2019) in section 2.2.5 but this did not seem to work. Due to time limitations the issue wasn't solved but will be solved in further researches on the exact value prediction of the tool wear.

### 3.3.2.2 Define training

The training of a CNN happens in two phases: a training phase and a validation phase. First the model will be trained with images gathered from one batch of images. This training phase will let some pictures of the batch go through the network and will adjust the weights of that network using back propagation. The back propagation can be done using different optimizers which will define how the weights are adjusted relative to the loss between the output of the model and the ground truth. As optimizer Stochastic Gradient Descent (SGD) is used. Cross entropy loss is used to calculate the loss between the output and the ground truth.

When the training phase is finished, the training takes some images from the validation set to verify or validate the weights set by the training phase. This validation will check how the model reacts to pictures it wasn't trained on and therefore shows how the model works for general cases.

The two described phases will be repeated for every epoch which will represent the duration of the training.

During the whole process of training some values are logged to be able to check the training after it is done. For this purpose the training accuracy and loss are calculated and saved. The same values are stored for the validation phase. These values are logged using Weights and Biases (wandb) (Biewald, 2020) which will transform these logs into graphs and other useful data which can be seen in chapter 4.

### 3.3.2.3 Define testing

In the data organisation section (3.3.1) a separate test set of images is created, this set will be used to evaluate the models performance. For this evaluation every picture from the test set will go through the model and the output will be compared to the ground truth. To log these tests, Weights and Biases is used as well where it saves some test pictures with their outputted class and the ground truth.

### 3.3.3 Training and testing

Finally the above functions will be used to actually train and test the model's performance. For the training all data will be pushed to a GPU from Google Colaboratory. To perform the training a sweep is created on wandb which will generate a set of hyper parameters that will be passed on to the training function of the model. With these parameters a dataloader is created that will generate image batches using the created dataset from section 3.3.1.4. These image batches will contain a specified amount of images defined by the batch size parameter. After this the model gets initialized, an optimizer is created as well as a loss criterion that will define the back propagation steps. Then the model is trained for a set amount of epochs hereafter the model is tested and the results are written to wandb.

# **Chapter 4**

# **Results**

## **4.1 Setup**

A fully adjustable setup is created and used to create two datasets with different lighting conditions. The two datasets except for the Birthday dataset are tested with the described program from 3.3. The results and graphs were obtained with Google Colab on a regular GPU, wandb was used for the processing of the logs.

The discussion is listed for every dataset starting with the second handmade dataset then the results of the Birthday dataset are listed. The Spaghetti dataset is listed third and finally the second handmade dataset is compared with the spaghetti dataset.

## **4.2 Second handmade dataset results**

The second handmade dataset was created in section 3.2.1.2. An example of this dataset is given in Figure 4.1. For this dataset 905 test runs were conducted with different parameters to obtain the best possible results. First an overview is given on the overall performance by listing the five best runs. Next these results are compared for the different initialized model architectures as described in 3.3.2.1. The influences of parameters are discussed to conclude with a few example results on the test set.



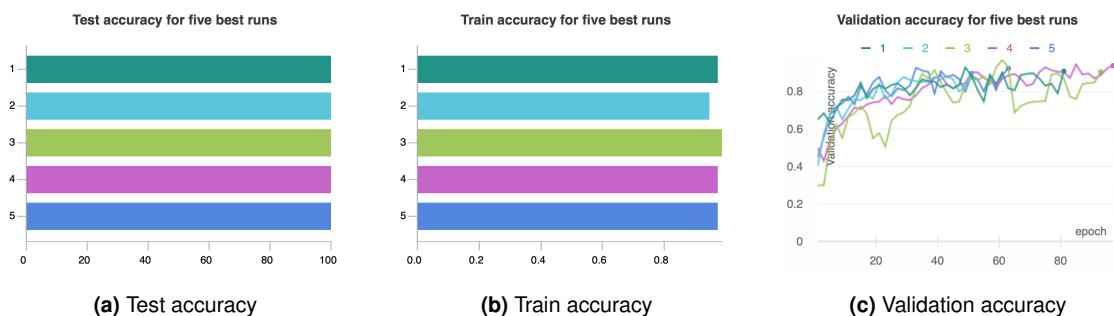
**Figure 4.1:** Example of second handmade dataset batch 3 insert 4.

**Table 4.1** Confusion matrix for tests of the second handmade dataset.

	Low	Medium	High
Low	x	2	0
Medium	2	x	5
High	0	2	x

### 4.2.1 Overall results

For the five best runs we get up to a validation accuracy of 100%, a test accuracy of 100% and a train accuracy of 97% as we can see on Figure 4.2. This is almost a perfect result for the classification model. Bear in mind that this is just performed on 100 pictures separated in three categories: train, validation and test. These categories have 71, 20 and 9 pictures respectively. Since there are so little test pictures the test score is not fully valid but we can conclude this is a very good test run.

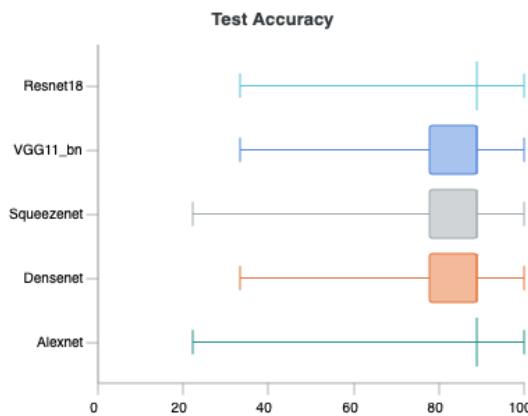
**Figure 4.2:** Test, train and validation accuracy for five best runs with second handmade dataset.

### 4.2.2 Comparison between different model architectures

To get a better overview on how different architectures perform on the dataset, we compare the main results for all these tested architectures. Starting with Test accuracy on these models, later the validation accuracy is discussed.

#### 4.2.2.1 Test accuracy

The test accuracy on 9 pictures from different classes that were randomly picked is really high. For 4 out of 5 model architectures the test accuracy is 100% so all inserts where predicted correctly. Densenet is performing a little worse with an accuracy of 89% what means that only one image wasn't predicted correctly. This is the result of testing 905 different parameters during training of the model. VGG11.bn is showing a smaller difference between the minimum test accuracy of all tests and the maximum accuracy of all tests. For now the VGG11.bn is the best model architecture based on the test accuracy.



**Figure 4.3:** Test accuracy for different model architectures with maximum and minimum values.

A simple confusion matrix is created in table 4.1 for 9 runs with 9 images each. Where the truth values are listed from top to bottom and the predicted values left to right. To interpret this table we look at the border between low and medium wear which is at 130 micrometer and the border between Medium and High wear at 230 micrometer. Since these limits are fixed numbers there are a lot of values just before and just after the limit. This makes it hard for the algorithm since it isn't even possible to see a difference between these images for the human eye. The confusion shows that the predictions between Medium and High are the most difficult. What we can also find is that there are no predictions totally wrong where High wear is predicted as low or vice versa.

#### 4.2.2.2 Validation accuracy

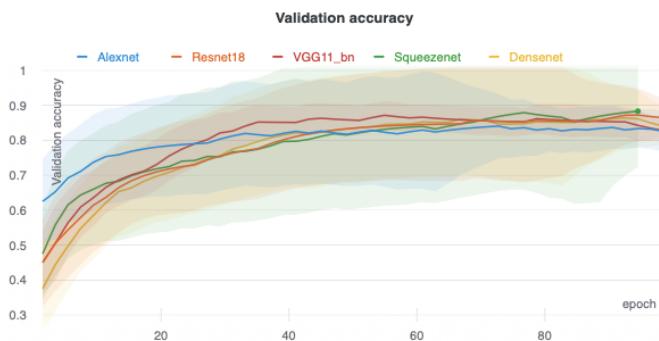
The validation accuracy is plotted relative to the epochs in Figure 4.4 which gives an overview of how the training went. The beginning represents a learning curve which stabilises at the end of the graph. This means the training got a good amount of epochs to reach the optimal point of maximum validation accuracy.

The graph is smoothed to create a better overview. The shadows behind are the standard deviation for each model. Some models tend to train faster to a higher validation accuracy like Alexnet and SqueezeNet. However this doesn't result in a better outcome for Alexnet where the median scores worse.

Validation accuracy for best runs is given in table 4.2. There are two measurement values we can use namely the highest test accuracy and the highest validation accuracy. In the first column the highest reached validation accuracy is given for each model. We can see here that more models reach 100% validation accuracy which should translate into a good generalisation. The results of the generalisation are given in the second column. Here the numbers represent the highest values for validation accuracy from the runs with the highest values for test accuracy. Or the table of all runs is sorted on test accuracy and the validation accuracy is taken from the first five runs. Here we can see the order isn't changed, there is just 5% lost in accuracy. It declares the link between validation accuracy and test accuracy.

**Table 4.2** Best validation accuracy for different deep learning architectures. Sorted on best validation accuracy in column 1 and sorted on test accuracy in column 2.

Model name	Validation Accuracy	Validation accuracy sorted on test accuracy
Alexnet	100%	100%
Resnet 18	100%	95%
Squeezezenet	100%	95%
VGG11.bn	95%	90%
Densenet	95%	90%



**Figure 4.4:** Best validation accuracy for different model architectures.

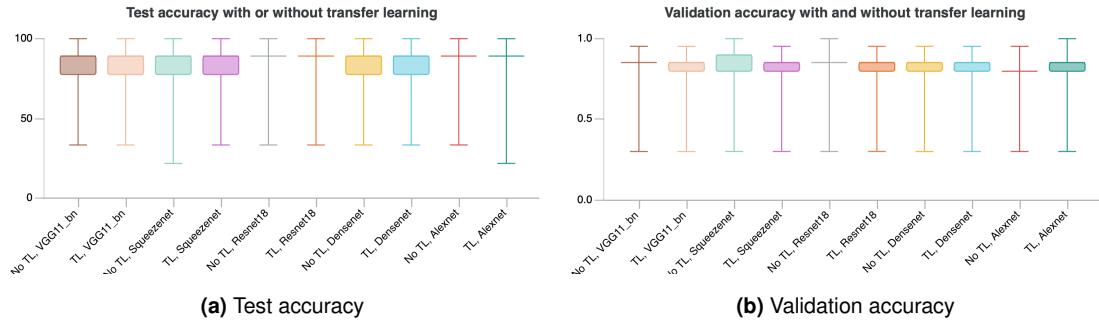
### 4.2.3 Influence of Transfer learning on the results

The influence of transfer learning from the imangenet dataset is visualised in the graphs on Figure 4.5. The test accuracy is plotted first for every model architecture and with transfer learning (TL) or without transfer learning (No TL) in Figure 4.5a. It can be seen that the results are almost identical for the two different settings. The maximum test accuracy measured is 100% for every model. The distributions changed a little bit which may be caused by the amount of test runs conducted for that specific model architecture.

The second graph on Figure 4.5b plots the validation accuracy for the different networks with and without transfer learning. From these boxplots we can see that Alexnet benefits from transfer learning. What would be due to the architectures small amount of parameters. Squeezezenet and Resnet 18 actually get worse results with transfer learning opposed to training without transfer learning.

### 4.2.4 Other parameter influences

The settings for the 40 best runs are visualized in Figure 4.6. For every model architecture some different hyperparameters are listed with their values for different training runs. From left to right we see the model name, the amount of epochs for the training, the batch size, whether or not transfer learning is used and finally the learning rate. Since this graph shows the top 40 runs the amount of runs starting from a single model name defines its presence in this top 40. Squeezezenet and Resnet18 are represented by a lot of runs where the most green lines start from Squeezezenet. This

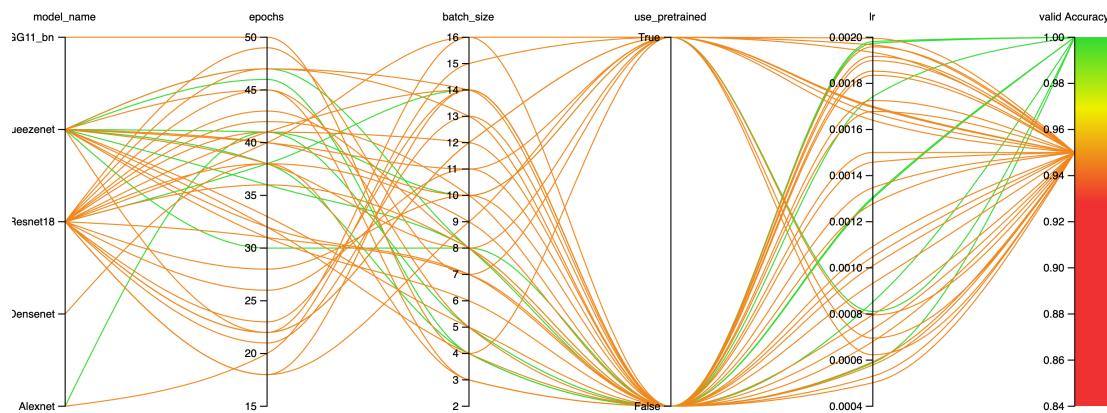


**Figure 4.5:** Validation and test accuracy box plot for different model architectures with or without transfer learning (TL).

green lines indicate a validation accuracy of more than 80% for that run.

Other things we can learn from this graph are:

- The epochs have a significant contribution to the validation accuracy where the amount of epochs can be finetuned between 30 epochs and 47 epochs for best results.
- A batch size between 10 and 4 seems good but doesn't contribute that much to the final results since the spread of runs across the different batch sizes is pretty even.
- There are more runs in the top 40 that doesn't use transfer learning. This means the transfer learning doesn't affect the results in a positive way for the most part.
- The learning rate used for adjusting the weights can differ highly from one run to the other. No points on the learning rate graph actually are significantly more populated than others. The learning rate can be set to higher and lower values for further testing.



**Figure 4.6:** Parameter influence on validation accuracy.

### 4.2.5 Test images

Since the test accuracy was so high only a few images were falsely predicted thus it is not relevant to show images of these results.

## 4.3 Birthday dataset results

For the birthday dataset there are no results available because the dataset was not considered good which was discussed in 3.2.3.1.

## 4.4 Spaghetti dataset results

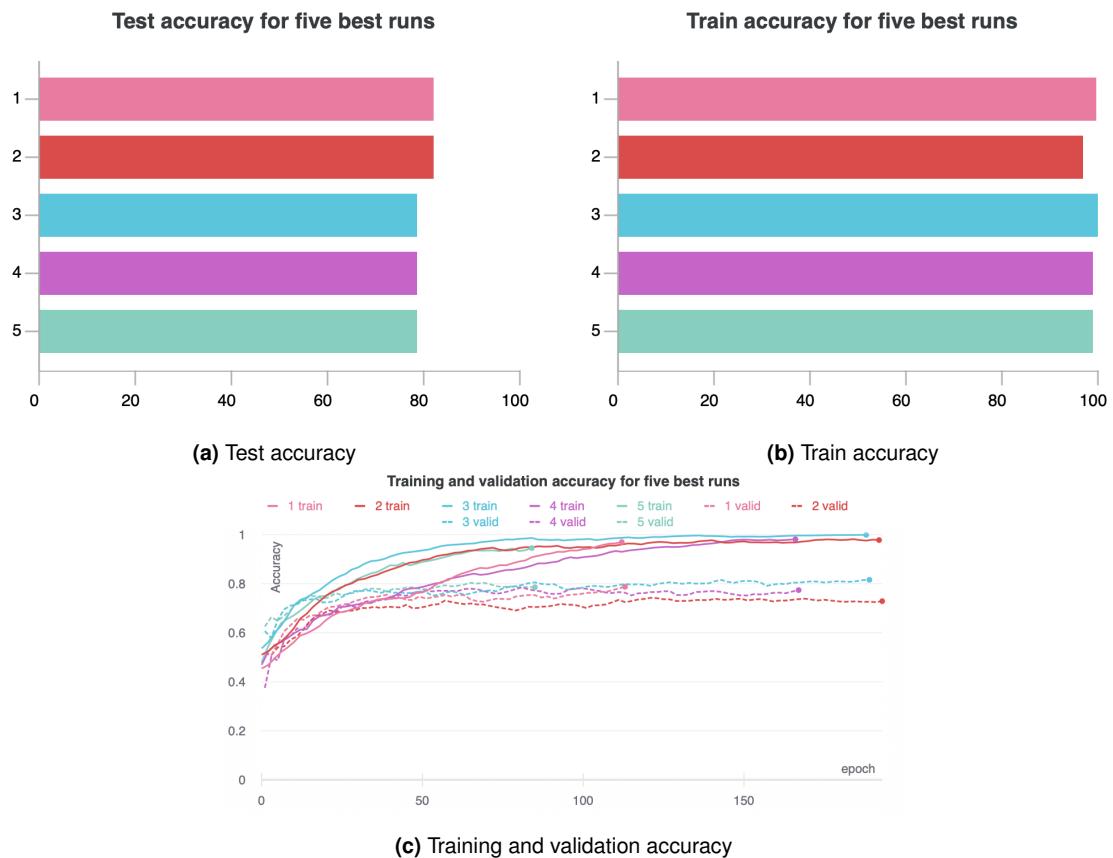
The spaghetti dataset was the first dataset to capture all of the currently available tool inserts. The different insert types are all in this one dataset where white as well as red lighting was used. 158 runs were completed with different parameters to generate the following results. For this third dataset the overall results are discussed in the first section where we take a look at the five best runs. After the general results we dive deeper into the different model architectures. After which some differences will be discussed.

### 4.4.1 Overall results for five best runs

The five best results are discussed to get a quick overview of the performance of this dataset on the chosen model architectures. The test accuracy for the five best runs is shown in Figure 4.7a. The maximum test accuracy is 82.1%. To dive deeper in the training we take a look at the training accuracy in Figure 4.7b the training accuracy is very high near 100%. The high training accuracy means the training pictures get predicted correctly but the other general pictures get predicted wrong for almost 20% of the pictures. To take a deeper look into this difference between testing and training accuracy we take a look at the validation accuracy during training. If the training accuracy would increment and the validation accuracy would decrement after a time the model might be overfitting. Figure 4.7c shows the training and validation accuracy during training. There is no overfitting seen on this graph so the model just isn't powerful enough for the task or the dataset is too small.

### 4.4.2 Comparison between different model architectures

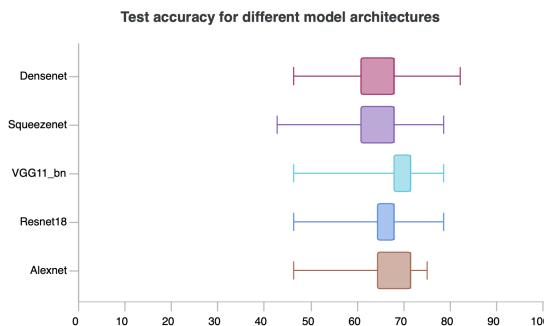
Just like we did for the Second handmade dataset, the results for different model architectures are compared. The test accuracy, validation accuracy and influence of transfer learning are analysed.



**Figure 4.7:** Test, train and validation accuracy for the five best runs based on their test accuracy scores.

#### 4.4.2.1 Test accuracy

On the first graph the test best test accuracy for every model is plotted. Densenet achieved the best accuracy for predicting the wear class of images on the test set. 78% of images were correctly predicted. The other architectures aren't far behind on the accuracy with 75%. Resnet 18 on the other hand is performing worse with a test accuracy of only 67%. On this first opinion Densenet seems to be the best model for this classification task.



**Figure 4.8:** Test accuracy

#### 4.4.3 Validation Accuracy

To further check the performance of the models a graph is shown in figure below. Here the validation accuracy is plotted for every model with its minima and maxima in shadow. There is not much of an inclining curve to this graph where normally the graph would go up in the first half and stabilise at the end.

Squeezezenet and VGG11.bn seem to have trained to the highest validation accuracy of 82%. This is an indication on how well the model will react on new images like the images from the test set. Densenet has an accuracy of 78%, Alexnet an accuracy of 75% and finally Resnet 18 with an accuracy of 73%. There must be noticed that Resnet 18 only got tested in 3 different parameter configurations whereas the others where optimized a bit more.

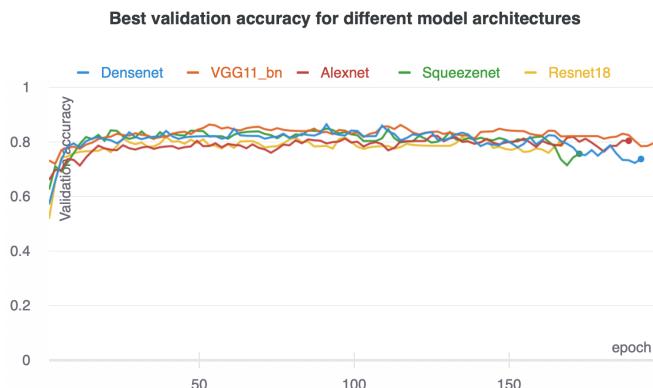
Table 4.3 shows the results for the different model architectures where the test and validation accuracy predict almost the same order for best model architecture. VGG11.bn and Densenet take the lead here. These networks are complexer with more layers and more parameters. The validation accuracy during training is also plotted for the best runs of each architecture. This can be seen in Figure 4.9

#### 4.4.4 Influence of transfer learning on the results

The transfer learning influence is plotted in Figure 4.10a for the test and validation accuracy. Transfer learning generally seems to have an effect on this dataset where the runs with transfer learning score on average 2% better and the maximum result is 5% better which is a significant improvement

Model name	Validation Accuracy	Validation accuracy sorted on test accuracy
VGG11.bn	84%	82%
Densenet	84%	82%
Alexnet	82%	82%
Squeezezenet	82%	78%
Resnet 18	80%	80%

**Table 4.3** Validation accuracy plotted for best model per architecture. In first column sorted on best validation accuracy, in second column sorted on best test accuracy.



**Figure 4.9:** Best validation accuracy for different model architectures plotted in relation to the amount of epochs.

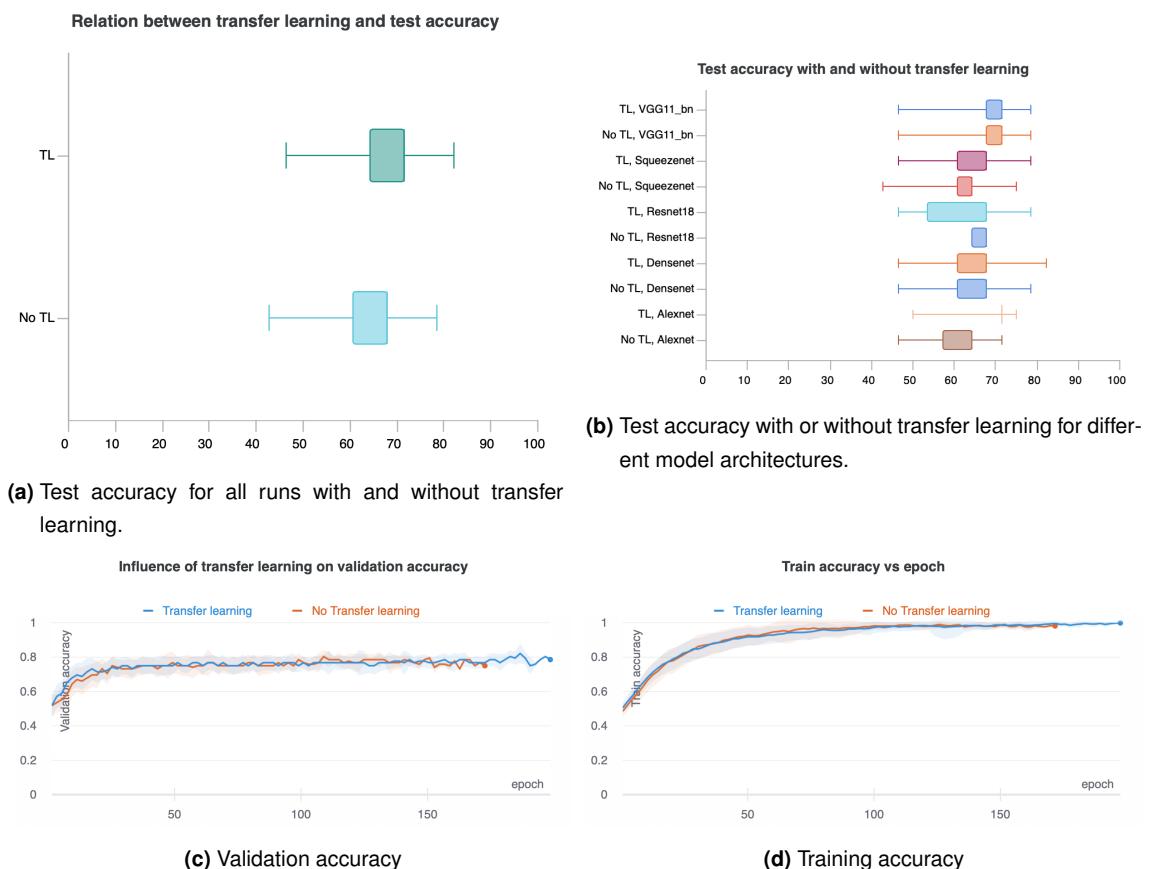
over the runs without transfer learning.

The same difference is also compared for all different model architectures in Figure 4.10b where the test accuracy is laid out. Squeezezenet, Resnet 18, Densenet and Alexnet all benefit from transfer learning. VGG11.bn on the other hand doesn't benefit from transfer learning and produces the same results for both settings.

As there are significant differences in the results for training with or without transfer learning the training of the models is examined. First the median and standard deviation of the validation accuracy during training is plotted in Figure 4.10c, secondly the train accuracy is plotted in Figure 4.10d. When looking at the validation accuracy, the learning curve with transfer learning inclines a bit before the curve without transfer learning. Using a pre-trained network speeds up the initialisation of the network but also uses more epochs to finish the training. There are no differences worth noting in the course of the training accuracy.

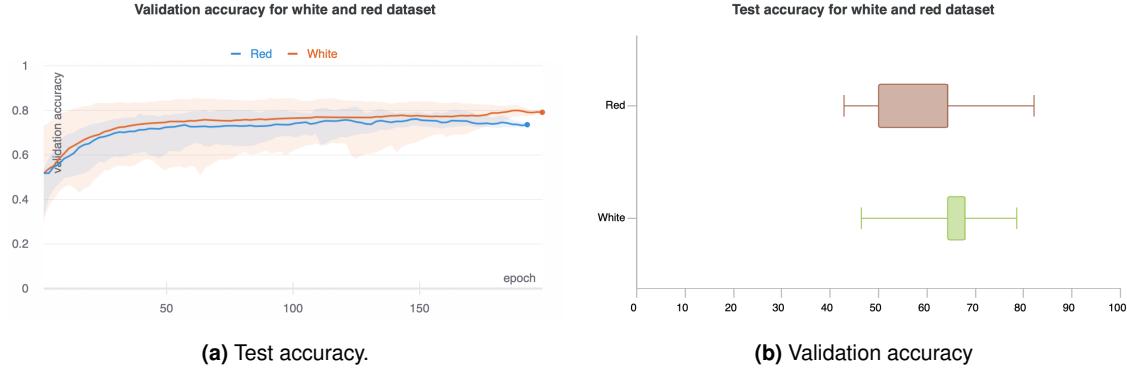
#### 4.4.5 Red vs White

For further investigation of the lighting color effects on the results, the median and standard deviation of the test and validation accuracy are plotted in Figure 4.11a and 4.11b. For the validation accuracy there is a small difference of 4% between the two curves where the white lighting seems



**Figure 4.10:** Test accuracy for models trained with and without transfer learning. In picture a) a general overview of all performed tests. in picture b) a comparison between different model architectures. Picture c and d provide the differences during training for validation accuracy and training accuracy respectively.

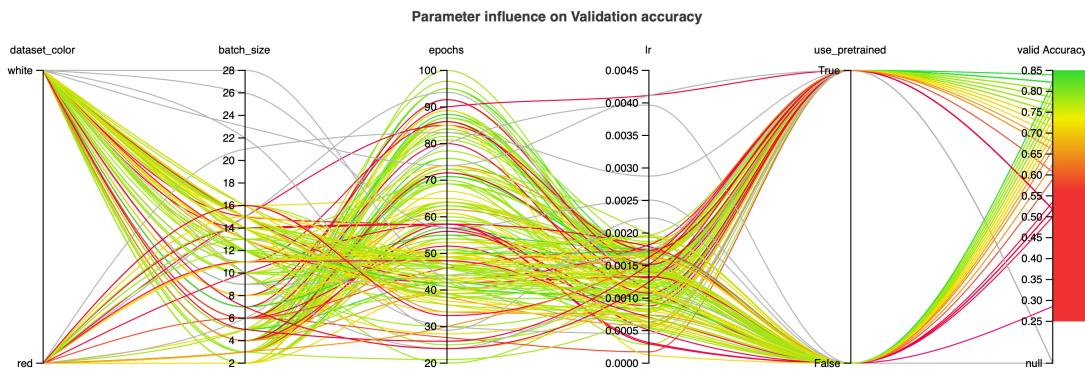
to perform better. A similar result can be seen for the median of the test accuracy however the best results got achieved by the dataset with red lighting.



**Figure 4.11:** Test and validation accuracy for a different led color.

#### 4.4.6 Training runs

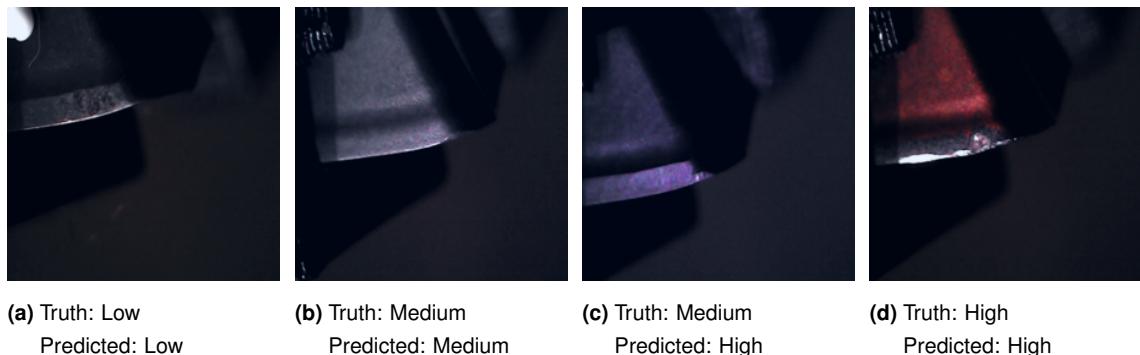
On figure 4.12 different runs with different parameters are displayed. Here can be seen that more green lines start from the white dataset. This corresponds to earlier findings where was stated that the white dataset performed better on average. The batch size can be finetuned between 2 and 28. The amount of epochs can be increased in an attempt to improve the results, however on Figure 4.7c the validation accuracy seems to be stable at the end where adding more epochs wouldn't change much on the results. The learning rate can be kept between 0 and 0.002. Using transfer learning isn't a deciding factor for good results.



**Figure 4.12:** Parameter influence on validation accuracy.

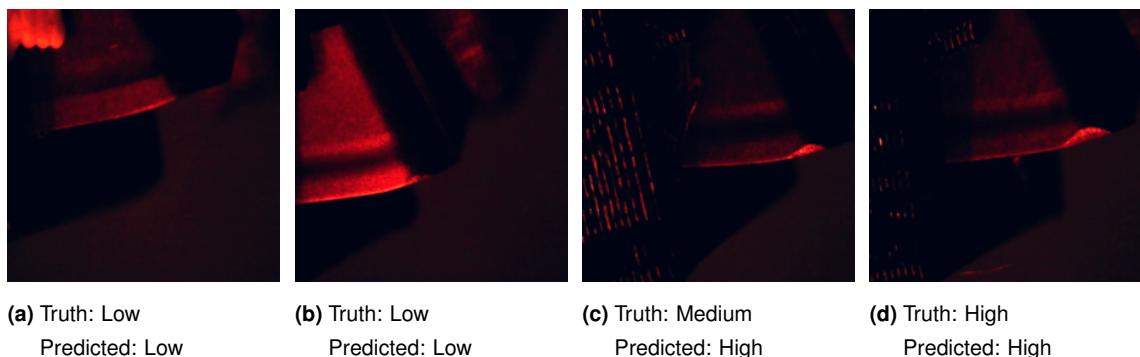
#### 4.4.7 Test images

To illustrate the results, some test images from the white lighted part of the dataset are listed in Figure 4.13. One error is given in Figure 4.13c where high wear was predicted for a medium wear insert. No hard mistakes where found in the outcome of this test.



**Figure 4.13:** Images with predictions from the spaghetti dataset with white lighting.

In Figure 4.14 some outputs for the red dataset are listed. Figure 4.14c displays an error where a medium wear inset was predicted as high wear. On this same picture is a big plastic piece visible in the picture. This possibly affects the result.



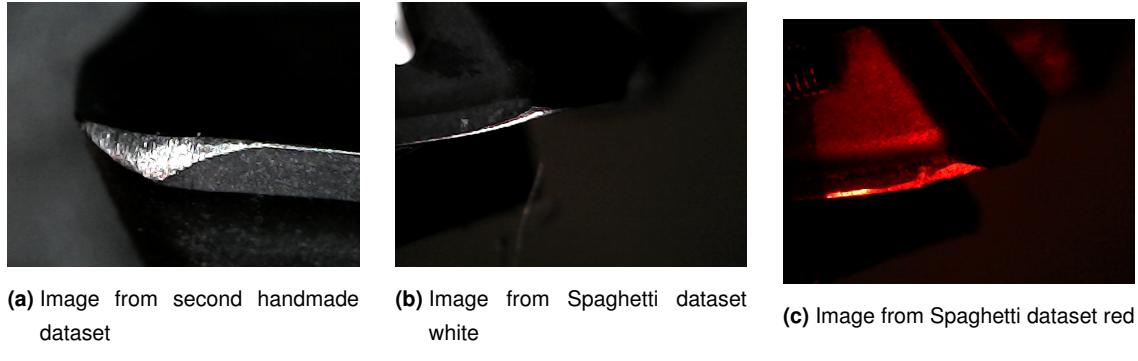
**Figure 4.14:** Images with predictions from the spaghetti dataset with white lighting.

## 4.5 Spaghetti dataset compared with second handmade dataset

If we compare the results from the spaghetti dataset with the results that were obtained from the second handmade dataset, these results aren't as expected since the amount of data in the Spaghetti dataset is more than doubled but the results for the test are worse. To give an overview of the dataset three inserts are listed in Figure ??:

- One from the second handmade dataset in Figure 4.15a
- One from the white part of the Spaghetti dataset in Figure 4.15b
- One from the red part of the Spaghetti dataset in Figure 4.15c

For further investigation the datasets are globally compared first and after that the influence of the different model architectures on the datasets will be covered.



#### 4.5.1 Global comparison

For this global comparison will the Spaghetti dataset be split up into a white and a red part so the results from the white part should match the results from the second handmade dataset.

First we compare the train accuracy and the validation accuracy to spot differences in the learning curve of these two datasets. For the training accuracy of Figure 4.16a we see that the second handmade dataset requires fewer training epochs and increases in train accuracy much quicker. For the validation accuracy in Figure 4.16b there is a similar trend. On both figures can be noticed that the standard deviation given in shadowed color is much larger for the Spaghetti dataset which means the training runs where much more fluctuating. We can conclude from these results that the spaghetti dataset is much harder to learn. For the confirmation we look at Figure 4.16c where the test accuracy is plotted. The same conclusion can be extracted from this figure. The second handmade dataset has less fluctuation and produced much more consistent good results and the spaghetti dataset also reached an accuracy of 100% but has much more runs with a lower accuracy.



**Figure 4.16:** Comparison for test, train and validation accuracy for different datasets.

Table 4.4 shows the validation accuracy for the white part of the Spaghetti dataset. These results can be compared with the results from the second handmade dataset in table 4.2. The same order

is kept and even the values don't differ much. Alexnet, Resnet 18 and Squeezezenet can be said to be the best architectures for this problem with a smaller, simpler dataset. This table can also be compared against table 4.3 from the full spaghetti dataset where the order is basically reversed.

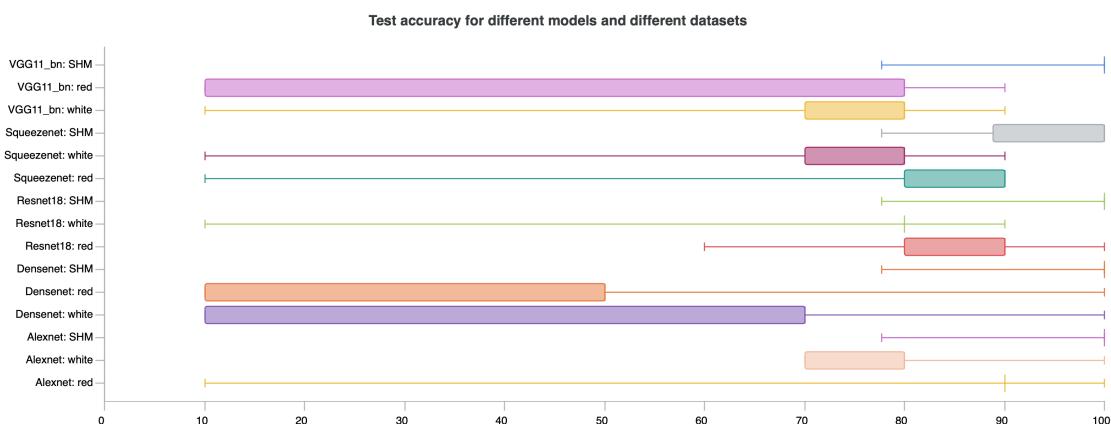
This observation can be declared based on the Table 2.1 Alexnet, Resnet 18 and Squeezezenet all have similar values for the number of parameters and depth (61.1 M, 11.7 M, 1.2 M) and (8, 18, 10) respectively. VGG11 has 123.1 M parameters and 11 layers and Densenet 121 has 7.9 M parameters and 121 layers. From this information we can conclude that a wider network with less parameters and layers produces better results for this easy problem with a small dataset. But for the more difficult problem with the Spaghetti dataset where more data is provided, the architectures with more parameters and layers like VGG11 and Densenet 121 are better.

**Table 4.4** Validation accuracy for the white part of the Spaghetti dataset. First column is sorted on best validation accuracy, second column is sorted on best Test accuracy.

Model name	Validation Accuracy	Validation accuracy sorted on test accuracy
Alexnet	100%	100%
Resnet 18	100%	100%
Squeezezenet	100%	100%
VGG11.bn	90%	90%
Densenet	90%	90%

#### 4.5.2 difference for different model architectures

To check the influence of the model architecture on the dataset we plot the test accuracy for every dataset and all different architectures. This is shown in Figure 4.17 where SHM stands for second handmade dataset; red stands for the red part of the spaghetti dataset and white for the white part. The same conclusions can be made for these models as for the global results for these datasets. The model architecture doesn't incorporate a sufficient difference in results.



**Figure 4.17:** Difference in test accuracy for different datasets trained with different model architectures.

As already noted in the section about the spaghetti dataset, there are disturbances in the images where the white or black plastic from the tool holder comes into the frame. This makes it harder to separate some pictures. Secondly the position of the worn area isn't always the same like in the second handmade dataset.

To conclude: the images must be taken a bit better so less unwanted things are in the picture and the wear should be lightened in a way that makes it possible to detect the different wear areas on different inserts.

## Chapter 5

# Conclusion

This paper proposed a solution for the classification of tool wear on carbide inserts used in the milling industry and will be used for the validation of camera setups in a bigger research where the research question: How can tool wear on carbide inserts be quantized using a direct measurement method? will be answered. This will also lead to the answer on another research question where this will validate a vision set-up.

We obtained a full dataset of white and red pictures of 140 Carbide inserts.

The needed depth of the algorithm was discussed where the more complex problem with different insert types needs a deeper neural network architecture

With a test accuracy of 82.1% on the mixed dataset, these results compare well to the state of the art where similar experiments were conducted with a test accuracy of 87%. This is even outperformed with a test accuracy of 100% on the dataset with only one type of insert.

**Future work** This paper will be implemented to find the best method for tool wear quantisation on carbide inserts. The best set-up will be defined upon the results gathered with this research. This will be used to create a new and bigger dataset which will be used for image regression which will predict exact values for the tool wear.

**Planning** A renewed planning is given in Appendix A.

# Chapter 6

## Bibliography

- (2020). Raspberry pi user manual. <https://www.raspberrypi.org/documentation/>. Accessed: 2020-12-10.
- Adam, P., Sam, G., Soumith, C., and Gregory, C. (2020). Pytorch. <https://pytorch.org>. Accessed: 2020-12-09.
- Ambadekar, P. K. and Choudhari, C. M. (2020). CNN based tool monitoring system to predict life of cutting tool. *SN Applied Sciences*.
- Basnet, R. B., Shash, R., Johnson, C., Walgren, L., and Doleck, T. (2019). Towards detecting and classifying network intrusion traffic using deep learning frameworks. *J. Internet Serv. Inf. Secur.*, 9(4):1–17.
- Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.
- Caicedo, J. C. (2019). New color from multilayer coating applied machining tools based on tungsten carbide insert. *International Journal of Advanced Manufacturing Technology*.
- Cerce, L., Pusavec, F., and Kopac, J. (2015). Novel spatial cutting tool-wear measurement system development and its evaluation. In *Procedia CIRP*.
- Chandrarathne, G., Thanikasalam, K., and Pinidiyaarachchi, A. (2019). A Comprehensive Study on Deep Image Classification with Small Datasets. In *Lecture Notes in Electrical Engineering*.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.
- Gu, J., Barber, G., Tung, S., and Gu, R. J. (1999). Tool life and wear mechanism of uncoated and coated milling inserts. *Wear*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

- Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*.
- I, i. (2020). Keras. <https://keras.io>. Accessed: 2020-12-09.
- Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2017). 50 X Fewer Parameters and < 0 . 5Mb Model Size. *Iclr*.
- J, H. (2020). Fast.ai. <https://docs.fast.ai>. Accessed: 2020-12-09.
- Khan, A., Sohail, A., Zahoor, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*.
- Li, W., Singh, H. M., and Guo, Y. B. (2013). An online optical system for inspecting tool condition in milling of H13 tool steel and in 718 alloy. *International Journal of Advanced Manufacturing Technology*.
- Ma, J., Luo, D., Liao, X., Zhang, Z., Huang, Y., and Lu, J. (2020). Tool wear mechanism and prediction in milling TC18 titanium alloy using deep learning. *Measurement: Journal of the International Measurement Confederation*.
- McKinney, W. and Team, P. D. (2015). *Pandas - Powerful Python Data Analysis Toolkit*.
- Pagani, L., Parenti, P., Cataldo, S., Scott, P. J., and Annoni, M. (2020). Indirect cutting tool wear classification using deep learning and chip colour analysis. *International Journal of Advanced Manufacturing Technology*.
- Schmitt, R., Cai, Y., and Pavim, A. (2012). Machine Vision System for Inspecting Flank Wear on Cutting Tools. *ACEEE International Journal on Control System and Instrumentation*.
- Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.
- Xu, X., Zheng, H., Guo, Z., Wu, X., and Zheng, Z. (2019). SDD-CNN: Small data-driven convolution neural networks for subtle roller defect inspection. *Applied Sciences (Switzerland)*, 9(7).

## **Appendix A**

## **Schedule**

FACULTY OF ENGINEERING TECHNOLOGY  
DE NAYER (SINT-KATELIJNE-WAVER) CAMPUS  
Jan De Nayerlaan 5  
2860 SINT-KATELIJNE-WAVER, België  
tel. + 32 16 30 10 30  
[fet.denayer@kuleuven.be](mailto:fet.denayer@kuleuven.be)  
[www.fet.kuleuven.be](http://www.fet.kuleuven.be)



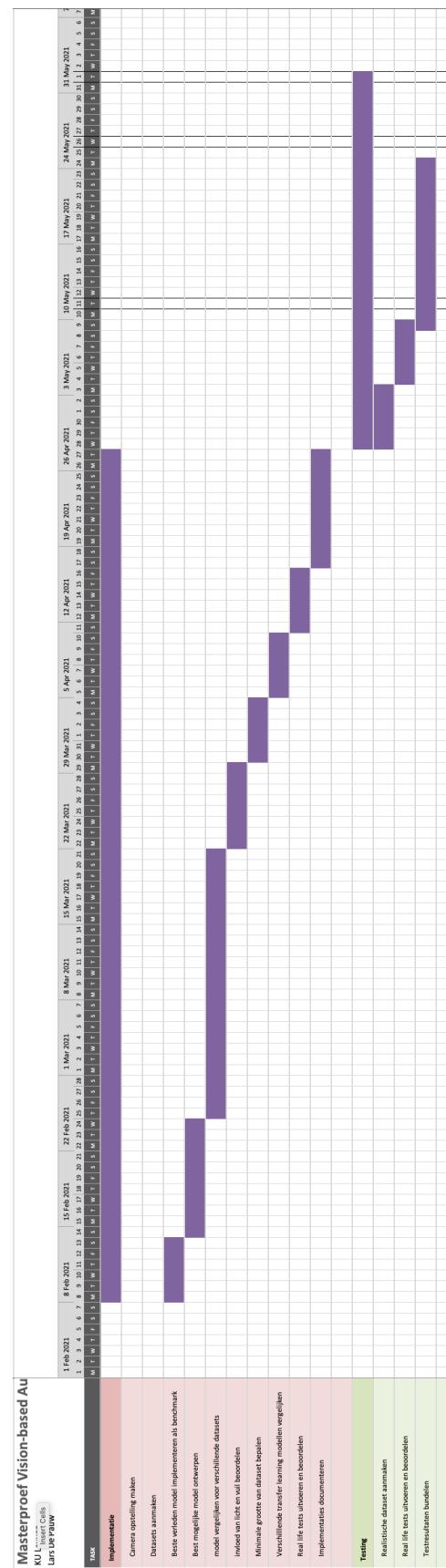


Figure A.1: Schedule for semester 2.