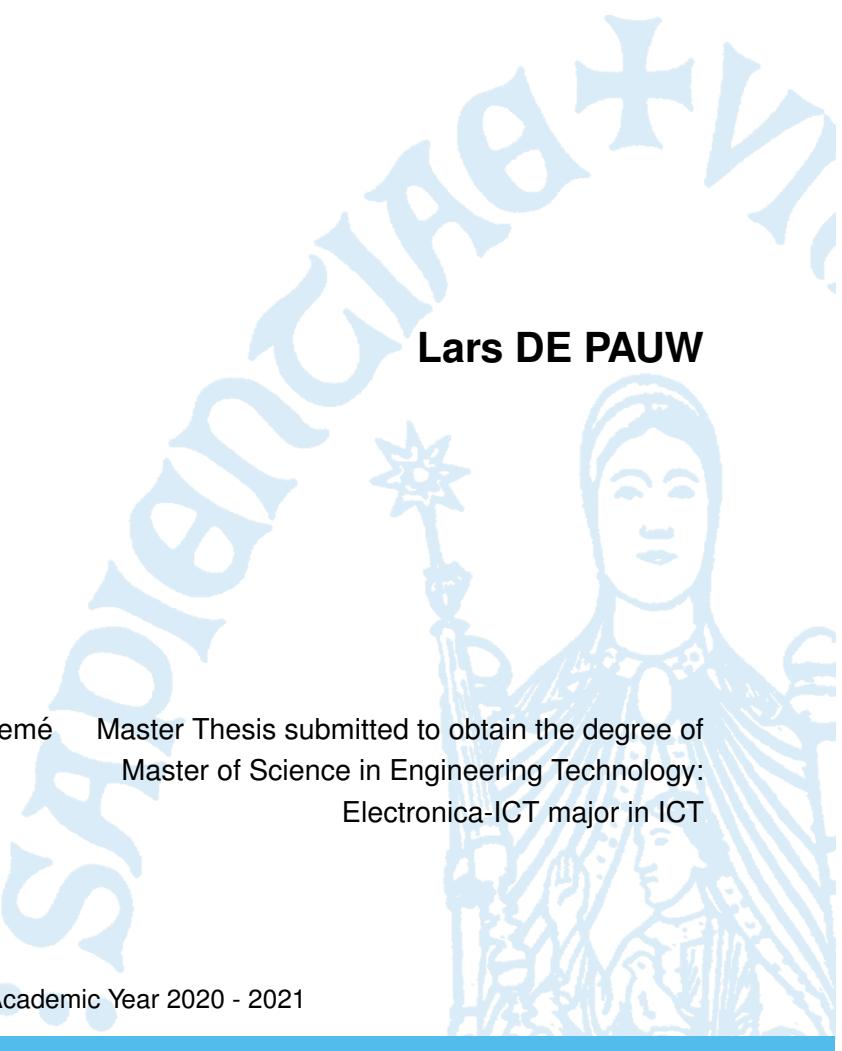


Vision-based automatic tool-wear inspection system



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.

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

Contents	v
List of Figures	vi
List of Figures	vii
List of Tables	viii
List of Tables	viii
List of symbols	ix
List of abbreviations	x
1 Introduction	1
1.1 Problem description	1
2 Literature review	3
2.1 State of the art	3
2.2 Light reflections on the tools	4
2.2.1 How does the tool wear?	4
2.2.2 Surface of the insert	4
2.3 Vision algorithms	6
2.3.1 Frameworks	6
2.3.2 Architectures for set-up verification	6
2.3.3 Overview of important numbers	9
2.3.4 Other architectures for small datasets	9
3 Implementation	11
3.1 Camera setup	11
3.1.1 Design of a tool holder	11

3.1.2 Design of a camera mount	13
3.1.3 Light configuration	14
3.2 Datasets	17
3.2.1 Handmade datasets	17
3.2.2 Automated datasets	19
3.2.3 Created datasets	21
3.2.4 Dataset explanation	25
3.2.5 Setup	25
3.2.6 Results	26
3.3 Google Colab	28
3.4 Test Camera Setup	28
3.4.1 Resnet18	28
3.4.2 Inception v3	28
3.4.3 All Networks	30
3.5 All Networks	30
3.6 All Networks 1	30
3.6.1 Network architectures	30
3.6.2 Dataset	30
3.6.3 Results	31
3.7 All Networks 2	31
3.7.1 Network architectures	31
3.7.2 Dataset	31
3.7.3 Results	32
3.8 All Networks 3 Birthday	32
3.9 All Networks 4 Spaghetti	32
3.10 All networks 4 spaghetti first 5 batches	32
3.11 Resnet18	33
3.11.1 Creating first file	33
3.11.2 TSU_Resnet18_1	33
3.11.3 TSU_Resnet18_2	33
4 Results	36
4.1 Setup	36
4.2 Second handmade dataset results	36
4.2.1 overall results for the five best runs.	37

4.2.2	Comparison between the different model architectures	37
4.2.3	Influence of Transfer learning on the results	39
4.2.4	Other influences of parameters	39
4.2.5	test images	41
4.3	Birthday dataset results	41
4.4	Spaghetti dataset results	41
4.4.1	Overall results for five best runs	41
4.4.2	comparison between different model architectures	42
4.4.3	Validation Accuracy	43
4.4.4	Influence of transfer learning on the results	43
4.4.5	Red vs White	44
4.4.6	training runs	44
4.4.7	Test images	46
4.5	Spaghetti dataset compared with second handmade dataset	46
4.5.1	Global comparison	47
4.5.2	difference for different model architectures	49
5	Conclusion	50
5.1	Inleiding	50
5.2	Referentiestijl	50
6	Bibliography	52
A	Uitleg over de appendices	54

List of Figures

1.1	Picture of an insert on 1cm by 1cm grid	1
1.2	Rotary holder with one insert in place.	2
2.1	Construction of the Tool insert with coating. Figure from (Gu et al., 1999)	5
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)	5
2.3	a building block for Resnet architecture	7
2.4	Summary of Resnet 18 architecture	8
3.1	First design of a tool holder oriented at an angle of 40° to the camera.	12
3.2	First design of wheel holder with inserts mounted in place.	12
3.3	Second wheel holder with clip and insert.	13
3.4	Desk lamp setup for initial testing.	14
3.5	Desk lamp setup without shadowing the background.	15
3.6	Desk lamp setup with shadowing the background.	15
3.7	White led strips test circuit.	16
3.8	Example of initial dataset	17
3.9	Camera setup for testing the camera angle.	20
3.10	Side camera view for led 8 through 10 listed from left to right.	21
3.11	Top camera view for leds 5 through 7 listed from left to right.	21
3.12	some caption	22
3.13	Fully lit example for insert 5 from batch 3.	23
3.14	Red, green, blue and white lighting on insert 5 from batch 3 led strip one. Left to right respectively	23
3.15	Red, green, blue and white lighting on insert 5 from batch 3 led strip two. Left to right respectively.	24
3.16	left: batch 3 insert 10 the side without bullet is off to the right side. right: batch 3 insert 10 The side with bullet. Is far off to the left side.	24

3.17 a. Plastic occludes the worn area. batch 3 plate 7 no bullet: had restants of 3D print plastic on worn area	24
3.18 Test setup camera angle more to top.	27
3.19 Example images for red and white light for spaghetti dataset.	27
3.20 Unsharp image	28
 4.1 Example of second handmade dataset batch 3 insert 4.	36
4.2 Test, train and validation accuracy for five best runs with second handmade dataset. .	37
4.3 Test accuracy for different model architectures with maximum and minimum values. .	38
4.4 Best validation accuracy for different model architectures.	39
4.5 Validation and test accuracy box plot for different model architectures with or without transfer learning (TL).	40
4.6 Parameter influence on validation accuracy.	40
4.7 Result images with their results.	41
4.8 Test, train and validation accuracy for the five best runs based on their test accuracy scores.	42
4.9 Test accuracy	43
4.10 Best validation accuracy for different model architectures plotted in relation to the amount of epochs.	44
4.11 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.	45
4.12 Test and validation accuracy for a different led color.	45
4.13 Parameter influence on validation accuracy.	46
4.14 Images with predictions from the spaghetti dataset with white lighting.	46
4.15 Images with predictions from the spaghetti dataset with white lighting.	47
4.17 Comparison for test, train and validation accuracy for different datasets.	48
4.18 Difference in test accuracy for different datasets trained with different model architectures.	49

List of Tables

2.1	Parameters and depth in layers for every model accuracy	9
4.1	Validation accuracy plotted for best model per architecture. In first column sorted on best validation accuracy, in second column sorted on best test accuracy.	43

List of symbols

Maak een lijst van de gebruikte symbolen. Geef het symbool, naam en eenheid. Gebruik steeds SI-eenheden en gebruik de symbolen en namen zoals deze voorkomen in de hedendaagse literatuur en normen. De symbolen worden alfabetisch gerangschikt in opeenvolgende lijsten: kleine letters, hoofdletters, Griekse kleine letters, Griekse hoofdletters. Onderstaande tabel geeft het format dat kan ingevuld en uitgebreid worden. Wanneer het symbool een eerste maal in de tekst of in een formule wordt gebruikt, moet het symbool verklaard worden. Verwijder deze tekst wanneer je je thesis maakt.

<i>b</i>	Breedte	[mm]
<i>A</i>	Oppervlakte van de dwarsdoorsnede	[mm ²]
<i>c</i>	Lichtsnelheid	[m/s]

List of Acronyms

WC Wolfram Carbide

Convolutional Neural Network (CNN)

Transfer learning TL

Chapter 1

Introduction

1.1 Problem description

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 labor costs in western Europe and more specific in Belgium here labour costs are about 40% above the average of labour costs in Europe. For this reason production and manufacturing costs should lower to be able to compete with other companies around the world.

Metal carving is done by using a rotary holder which spins over the metal work piece and carves away material. This rotary holder keeps inserts as seen in Figure 1.2. They are kept in place with a screw for easy removal and replacement. The inserts are made of different carbides mostly Wolfram carbide and get a coating on the outer layer to provide extra strength and more capabilities to cut different materials. The cutting part on the corner of the insert will wear when the work piece is carved and will result in a bad finish on the actual product if it is not replaced in time.



Figure 1.1: Picture of an insert on 1cm by 1cm grid

For the replacement of the inserts there are two policies used today:

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 labor intensive. Option number two is less labor intensive and thus less time consuming but will produce a lot of waste in those inserts. Having safety levels there will be a lot of inserts thrown away even when they are still usable.



Figure 1.2: Rotary holder with one insert in place.

Insert more context on the equipment

insert overview of the whole paper

Chapter 2

Literature review

2.1 State of the art

In this field there is not much research done over all and mostly the resulting workpiece is inspected. 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. Creates a nice overview of tests on different materials and different coatings, this gives the reason why it is important to detect toolwear in an early stage to produce as many good materials as possible.

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 errors 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 labor 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. But we will need more labeled data to be able to perform such a task which may be expensive to create. A accuracy of 7.5 micrometer is achieved.

2.2 Light reflections on the tools

To get a better understanding of the reflections of the tool inserts we take a look at how the inserts wear over time and what characteristics they have.

2.2.1 How does the tool wear?

This base material will mostly be Wolfram carbide (WC) due to 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 to 2 μm 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 layeres tools. article: New color from multilayer coating applied machining tools based on tungsten carbide insert J. C. Caicedo1 Caicedo (2019) Researched the reflectance of coating layers on tool inserts. Here is described that the best reflection occurs at the highest wavelength. This translates to the visible color red and will mean that the reflection should be the highest when the lighting is on the top of the spectrum of the camera lens.

The material is best cut with a laser at wavelengths 1030nm and 515nm. This is proved in: article: Fundamental investigations of ultrashort pulsed laser ablation on stainless steel and cemented

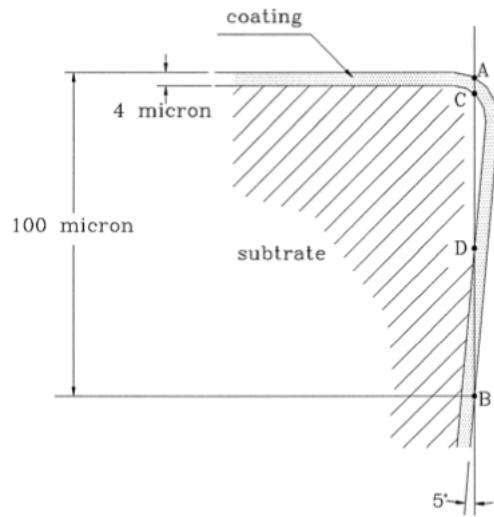


Figure 2.1: Construction of the Tool insert with coating. Figure from (Gu et al., 1999)

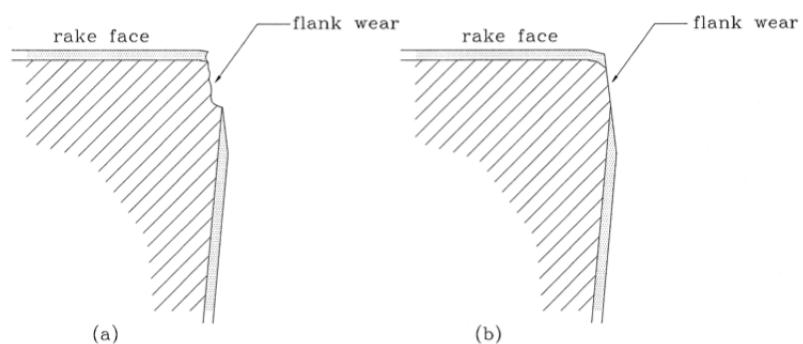


Figure 2.2: Two wear types: (a) Step between substrate and coating. (b) flat wear not perpendicular to the rake face Figure from (Gu et al., 1999)

tungsten carbide is the good removal also a good reflector?

2.3 Vision algorithms

In this section different vision algorithms will be discussed.

2.3.1 Frameworks

- this section may be deleted

There are many frameworks to perform computer vision tasks and make it as easy as possible for the user to get a lot done in a short amount of time. In what follows the different frameworks are listed and shortly documented. Basnet et al. (2019) provides an overview of different deep learning frameworks. The frameworks used in this thesis are:

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

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 which is designed for experimenting. PyTorch leaves a little more room for customization. This is the reason PyTorch was used for this thesis.

2.3.2 Architectures for set-up verification

Starting from a very small dataset for the creation of a vision set-up, it is necessary to choose some relevant deep learning network architectures to obtain results which can be compared for different setups. The dataset consists of a little less than 300 images. This constraints the choice of architectures to the ones with very little training parameters since there are not much input images to which the parameters can be trained. In the next sections are some well-known architectures that are used for this determination.

First there must be found an algorithm that can quickly confirm whether a setup is good or not. This will be done by taking pictures from different camera positions with the same lighting. After this the images go through a simple model and the output is verified with a test set.

The first algorithm will be Resnet 18 since this is known to perform well with less parameters and deeper architecture. 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.

2.3.2.1 Resnet18

First define what Resnet He et al. (2016) means than dig deeper into the Resnet18 architecture.

Resnet is a deep residual neural network that can reach deeper networks 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 2.3 gives an overview of the important building block which 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 outputs are added to the outputs of the stacked layers. This doesn't add extra complexity to the model but allows for more layers to be stacked on top of each other.

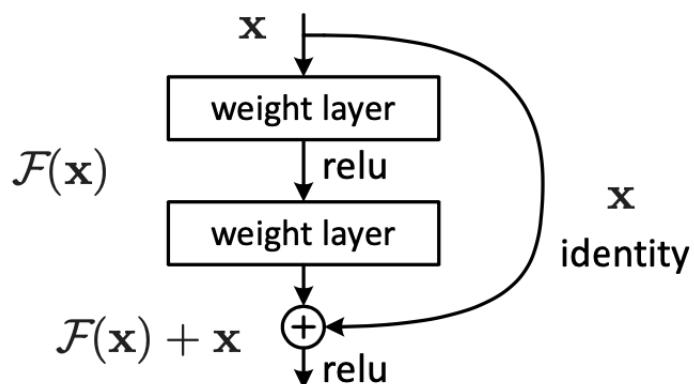


Figure 2.3: a building block for Resnet architecture

Figure 2.4 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 3x3 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 later be compared with other network architectures.

2.3.2.2 VGG11.bn

VGG is a network architecture found by Simonyan and Zisserman (2015) it was designed to get better results at the 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 VGG is used with 11 layers instead of the original 19 layered architecture.

Batch normalization

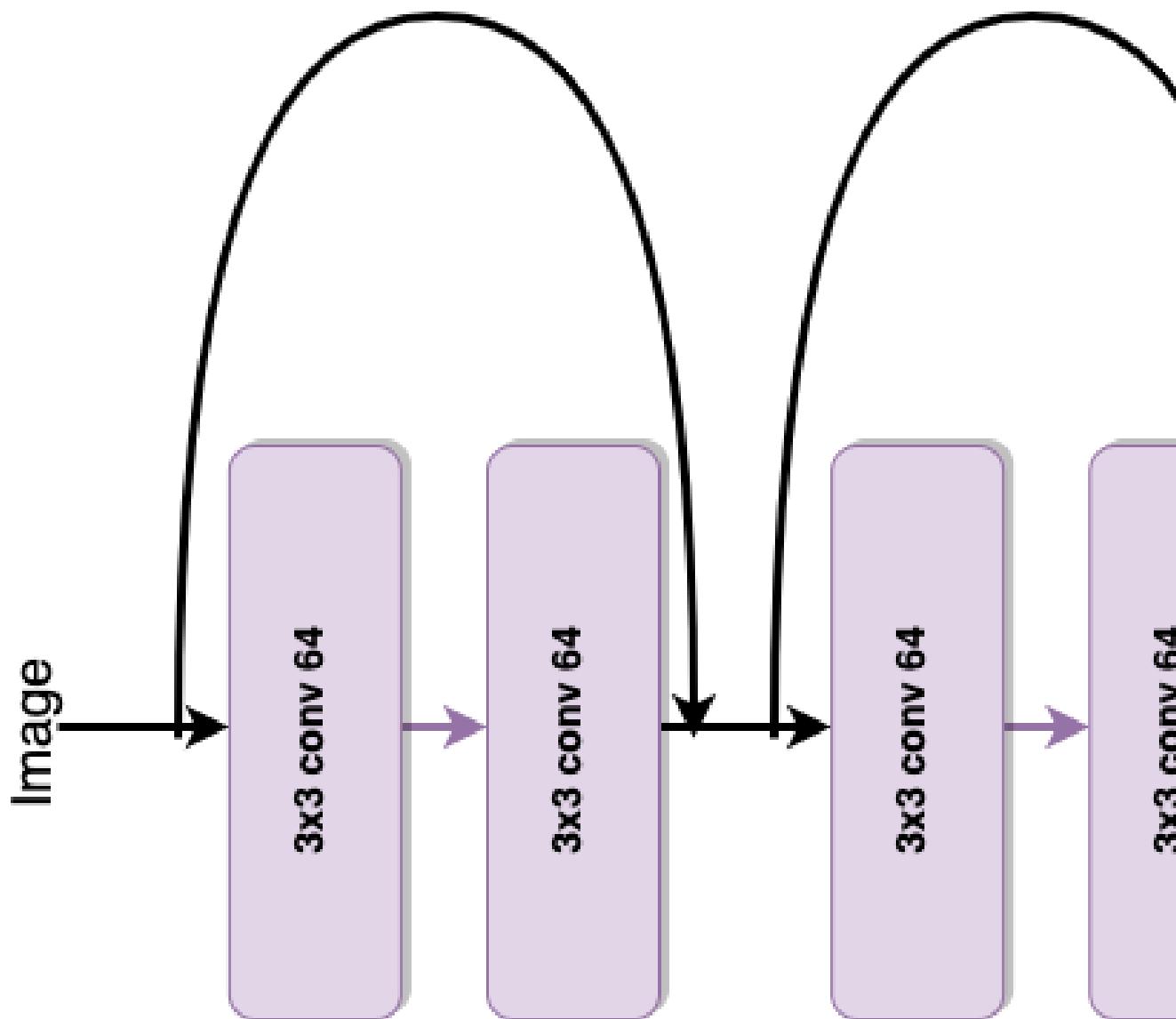


Figure 2.4: Summary of Resnet 18 architecture

2.3.2.3 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.

With 60 million parameters this is more complex as the Resnet 18 architecture

2.3.2.4 Densenet 121

2.3.2.5 SqueezeNet

ToDo

2.3.3 Overview of important numbers

Parameters and depth for all architectures

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

Table 2.1 Parameters and depth in layers for every model accuracy

2.3.4 Other architectures for small datasets

In this section other architectures are explored which received good results on very small image datasets. The next data input structure is made:

- 20% train images
- 10% validation images
- 10% test images

These images will go through different algorithms multiple times and the outputs are verified for every different algorithm.

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 and with 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 imbalance in amount of positive and negative examples. This label dilation method generates random classes where the amount of items per class equals the amount of items in the biggest class. 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. SqueezeNet 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

3.1 Camera setup

Carbide tool inserts are not yet items that are much researched so there are no big datasets that can be used to train and test algorithms on. Therefore a new dataset must be created with worn tool inserts. For the creation of the dataset a setup is created. This setup consists of three different parts:

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

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. The requirements for this holder are that the tool is easily removable to replace inserts. It may not be visible to the camera or provide very small mounts of distraction or occlusion.

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, a holder which kept the insert at an angle of 40°. This showed potential in the current setup as seen in Figure ???. The tool wear is nicely indicated with the reflections of the light.

3.1.1.2 Wheel holder

The previous first tool holder was good for taking a sample picture of an image 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 consistent

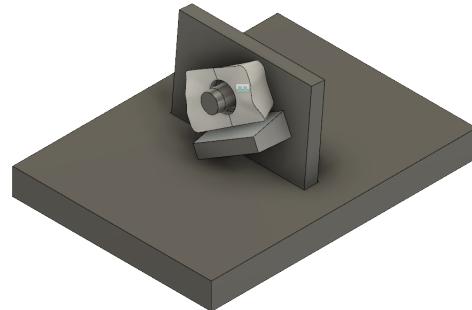


Figure 3.1: First design of a tool holder oriented at an angle of 40° to the camera.

way, a wheel is designed to mount 20 tools at once using a stepper motor and a fixed camera and lighting setup. The process of taking images would be 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 tools for an efficient dataset creation.

The first test of this wheel holder can be seen in Figure 3.2. Here everything is printed in one piece. The inserts are kept in place with brackets that go over the cutting edge of the insert. This wasn't ideal since the sharp edges of the insert would clamp in the bracket what made it very hard to remove the tools and didn't comply with the easy removal constraint.

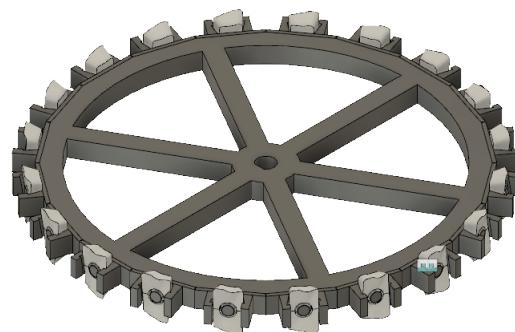


Figure 3.2: First design of wheel holder with inserts mounted in place.

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 insert separately. Figure 3.3 shows the full assembly of this second wheel holder.



Figure 3.3: Second wheel holder with clip and insert.

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 slabs 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.

The printing of these wheels was very difficult since it was with another material as we were used to and the print wouldn't stick to the print bed. This made a few bad runs and hours of wasted printing time. At the end 8 wheels of this are printed correctly and were used to create the datasets.

3.1.2 Design of a camera mount

In this page the camera mount will be discussed along the design process of the setup. Since the camera mount provided by the factory was not up to the task of taking consistent pictures and controlling the camera position precisely, 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.

insert assembly of full camera mount

3.1.3 Light configuration

In order to test different lighting options there must be a simple to configure lighting solution. Which is 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 light.

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 3.4.

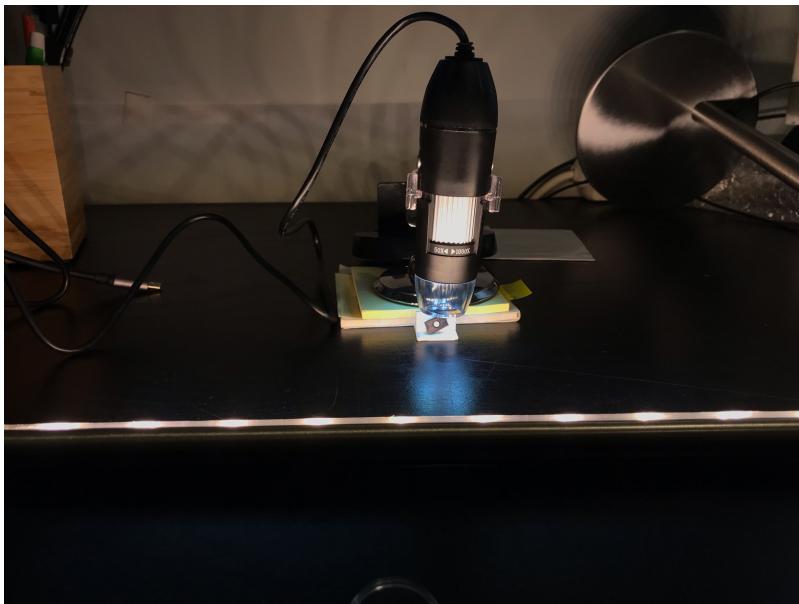


Figure 3.4: Desk lamp setup for initial testing.

This setup was using the top light of the camera. This light was good to be able to adjust the camera. Without the light, the erroneous places where more visible. But the desk lamp had 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 3.5 shows a picture of the tool with desk lamp lighting without blocking the light from the areas where it shouldn't be. Here we can see that there is a bright white background behind the tool.

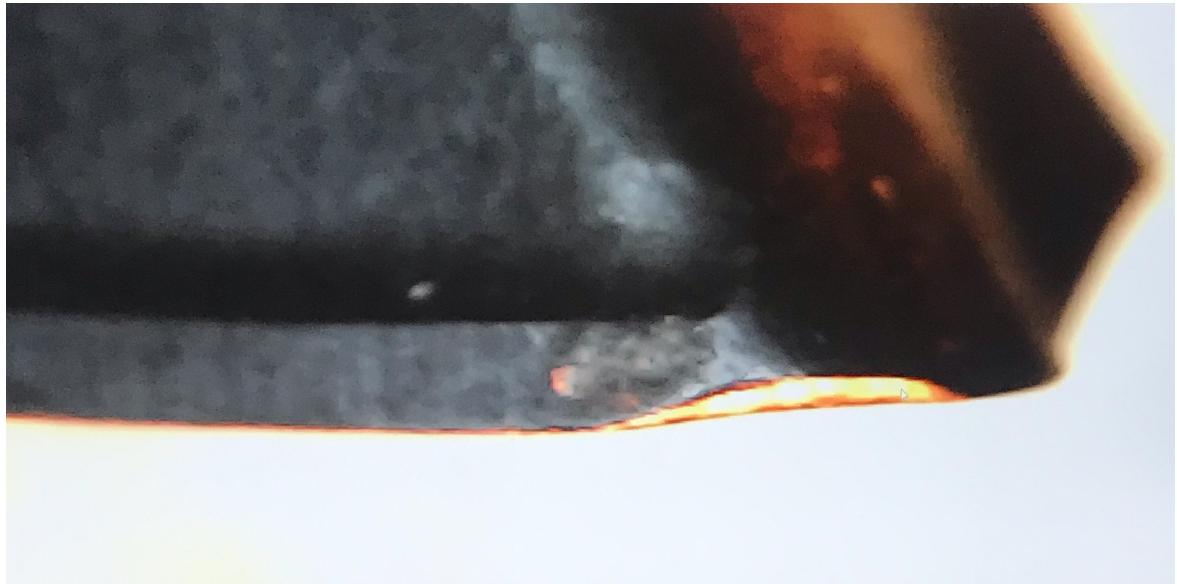


Figure 3.5: Desk lamp setup without shadowing the background.

The result of this is shown in Figure 3.6 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 to start creating a dataset.

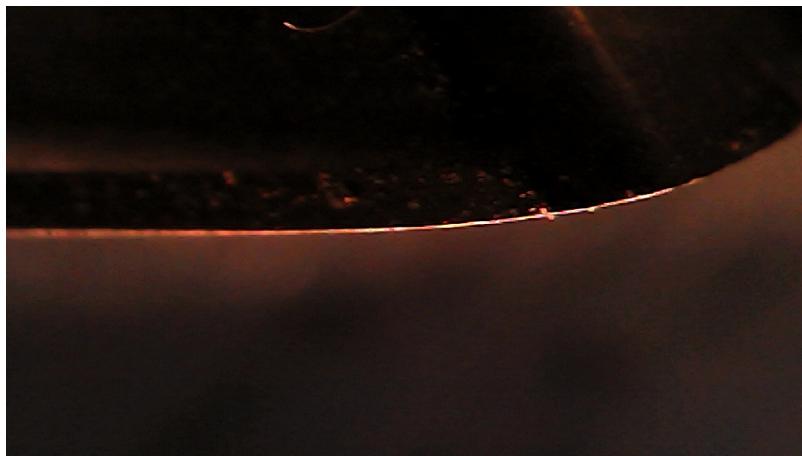


Figure 3.6: Desk lamp setup with shadowing the background.

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 up and down in a horizontal way, all the areas where light tend.

3.1.3.2 White Led Strips

A second lighting condition created is the lighting with 3 the same led strips controllable with a raspberry pi. 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 the test circuit can be found in Figure 3.7. A potentiometer was used to control the voltage over the led strip in order to control the brightness.

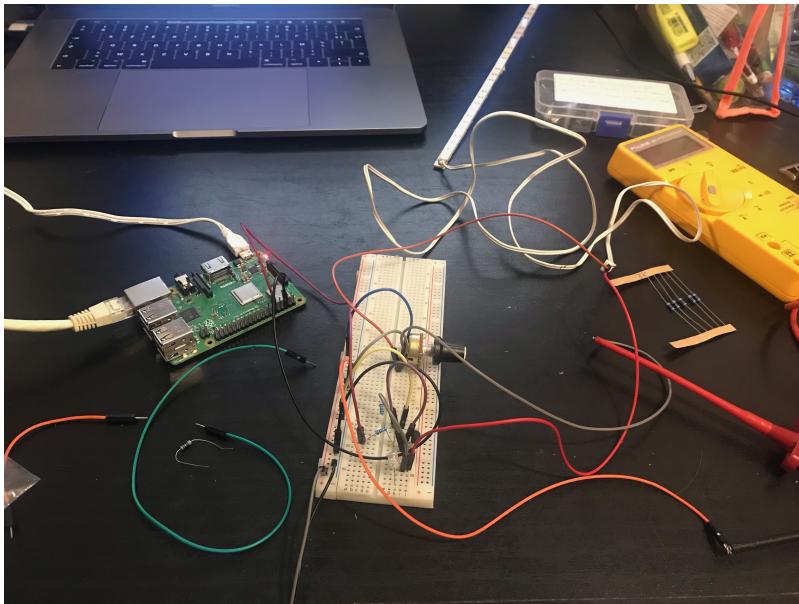


Figure 3.7: White led strips test circuit.

3.1.3.3 Adressable Color Changeable Led Strip

A third option of lighting is playing with the colors of the light. to archive this a setup will be created with a single adressable light strip where the color and led can be freely chosen. To assign a color 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 Light Reflection

From all these settings the single adressable 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 The led strip were mounted on metal strips which provided high adjustability. This made it possible to get lighting on different parts of the tool insert what makes later research easier. In Figure the light configuration is displayed for a partly worn tool.

insert tool with reflection

As seen on this render there are a lot of options with this setup. Each led strip can rotate and fifteen leds can be individually set to a color. The arc keeps a consistent distance between the led and the insert.

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 were 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 initial dataset where the images made by a microscopic camera at Sirris. These pictures were taken for the measurement of the tool wear. This dataset provided the labels for the first 5 batches labeled with 00x as batch number. This data was directly saved from the output of the microscopic camera. For this reason there is a lot of extra information on the picture as seen in Figure 3.8. The labels for this dataset were sorted with or without tag on the insert but the transcription code was lost. For example the insert had a mark on one side and no mark on the other side and the labels had a value for some insert a and some insert b. For this combination it wasn't known whether label a was the side with the mark or the other way round.

Since the labels weren't clear and the images were not optimal these inserts were all done again in a new hand made dataset called the second handmade dataset.

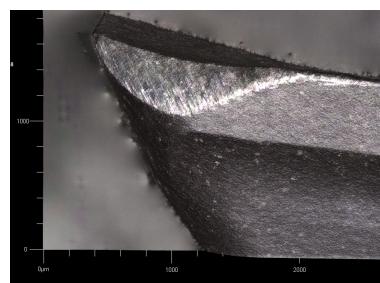


Figure 3.8: Example of initial dataset

Form The dataset given was 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.

Every insert that was measured had a little mark on one side to mark the a and b. 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.

While again checking the inserts out, a new dataset is created since this didn't ask much more time. During this proces there is also a new way of separating the sides of the inserts. There is a bullet at one side on every insert. This is an easy way to recognise a side and wont dissapear like the marker line.

3.2.1.2 Second handmade dataset

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 marker meant. This dataset also handles the first 5 batches labeled with 00x. For this dataset a setup much like the setup with the desk light was used but instead of lighting with the desk light only the light from the camera was used.

A second handmade dataset is made to compare with the first dataset and get to know what the stripes mean on the first dataset which was used to measure the wear.

The next setup is used:

The light came primarily from the camera itself which was set in the first brightness setting.

The camera was located at a distance of 2cm between the housing and the measured point on the insert. The housing starts at the black part; not the plexi glass protector.

During the making of this dataset the inserts where labeled with bullet or no bullet. This was not setup this way in the first place; instead there was a marker line on the insert. This marker line is also noted in the labels of the dataset.

An "s" means it is with a marker line. A

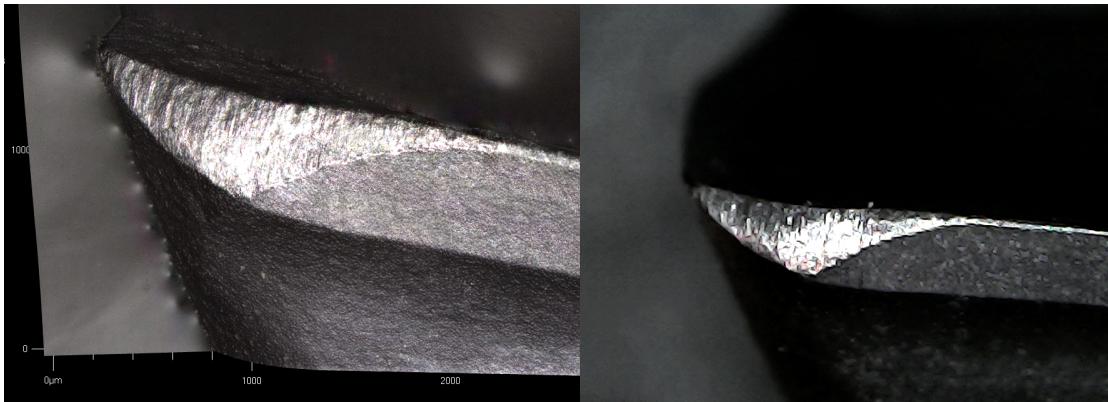
symbol	explanation
s	side with marker line
n	side without marker line
batch number	number specified on the box in which the inserts are kept
plate number	number specified inside the box; this goes from 1 to 10 per batch

The naming of the dataset is the following

b_<batch number>_p_<plate number>_<identifier of side>

After looking at the pictures of the datasets it can be confirmed that the 'b' side of the inserts is the side with the mark on.

This can be seen on the folowing two pictures whis correspond. The first one is from the first dataset, the second from the second dataset.



3.2.1.3 second initial dataset

The second initial dataset was made with the inserts from batches 11 to 19 labeled with 01x. Here 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.

With the second number of inserts (batch 11 to 19) another dataset was created for measurement. Since the photo's were labeled with two inserts at a time; the images taken were not relevant for the data processing and were not saved.

The distance for these inserts is measured between the line connecting the two highest points visible on the insert and the outer most point of the wear area.

3.2.2 Automated datasets

The hand made datasets are not scalable for a hundred or even a thousand inserts so a way of automatically creating datasets was researched. For this creation the past sections were used to create a setup and lighting conditions. With those things figured out a program was written to control the setup 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 main created datasets will be discussed.

1 birthday dataset

a conducted tests

2 Spaghetti dataset

a conducted tests

3.2.2.1 Camera position

With the help of 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 For these tests the first four plates of batch 004 are used. The lighting is red light coming from the addressable led strips that shine from different directions. This setup can be seen on Figure 3.9.



Figure 3.9: Camera setup for testing the camera angle.

Due to the problems with the arduino communications described here the first dataset wasn't successfull because the leds didn't turn on when the photo's where taken. So the results are all black pictures with al little shadow of the insert caused by the polluting light.

By putting a delay before sending a command to the Arduino 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 setup.

The results of this test setup can be found in Figure 3.10. The insert is lighted with two corresponding leds coming from the two led strips. 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. On this we can build the first dataset but first some other camera positions are tested.

Top camera position The same inserts where used 04 -> 1until 5.

Every pair of led is lighted separately to generate the photo's extra documents of the results in the folders images/dataset/.....



Figure 3.10: Side camera view for led 8 through 10 listed from left to right.



Figure 3.11: Top camera view for leds 5 through 7 listed from left to right.

The 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 arduino communication was resolved by changing the input delay from the serial input reader. The following pictures were the result. Like on the previous test one picture was taken for every two leds of the strip with red light. 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 is 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 page where firstly the dataset is documented and after that the tests that lead to this dataset are discussed.

3.2.3.1 Birthday dataset

Created a dataset on 27/11/2020 with a part of the given inserts for every possible color and led setting where pictures are taken from two separate led strips and every led one after another. This is done for white, red, green and blue colors.

Done for batch 1 to 11

On November 27th a new dataset is created where for every plate 91 pictures are taken.

The following pictures are available for the dataset:

- 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

The brightness is set to 80% for all lights to make sure to not clip against the top values.

The dataset consists of 120*91 photos of 60 inserts with 120 worn sides. After 120 the quality was evaluated and the red, green and blue colors didn't seem to add more information to the pictures.

In this dataset, there are some pictures unusable. As some worn areas are not even in the frame. Some to the left side and some to the right side. The placement of the wheel which holds the plates wasn't checked thoroughly.

The setup for creating this dataset was as follows:



Figure 3.12: some caption

Here two ledstrips were mounted and pointed at the photographed insert. The inserts were attached to the wheel with black clips 3D printed with PETG. This was chosen above white clips to lower the light reflections into the camera lens. This was a problem in previous setups.

Also a sturdy camera mount was fabricated out of metal profiles and 3D printed parts to get better notice of the placement of the camera opposed to the insert.

The image taking process took about 1 minute per batch because of the amount of pictures taken. Since time was limited and the setup wasn't yet perfect we decided to only run 60 inserts through it. These were from batches 1 to 11.

Sample pictures of this dataset can be found underneath.

For every led one picture is taken. These images can be put together to create images on a full spectrum of leds and take out the best conditions.

This can be done by inserting the images as different channels in the model.

The next pictures are from batch 3 insert 5 with led 6 turned on for both led strips and four colors.



Figure 3.13: Fully lit example for insert 5 from batch 3.

On these pictures we can see only red and white are visible and the B led strip wasn't visible. For this purpose the color of the wheel is changed to Black for less reflections. In the next dataset only white and red will be used for lighting.

Errors batch 5 plate 6 no bullet lighting not good

batch 5 plate 6 bullet has string from 3D printing on wear area batch 11 plate 5 bullet had hair on wear area

3.2.3.2 spaghetti dataset

After the Birthday dataset a new dataset was created using the things learned from that. Now the amount of leds driven is reduced to 5 leds. Where led 6 to 11 is used to lighten the inserts.

only the colors white and red are used for this dataset. Previously was discussed that the red color

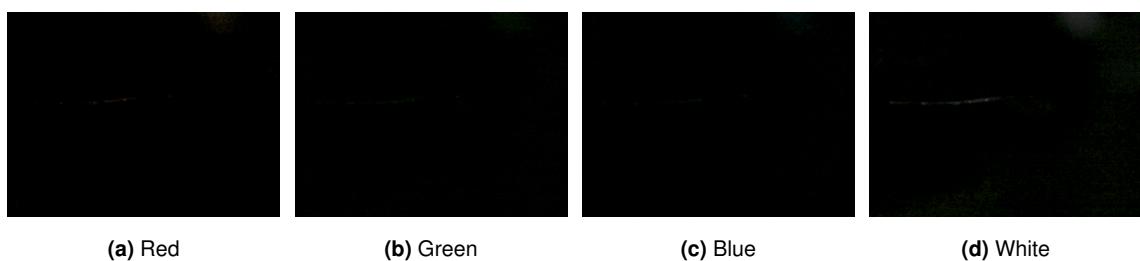


Figure 3.14: Red, green, blue and white lighting on insert 5 from batch 3 led strip one. Left to right respectively

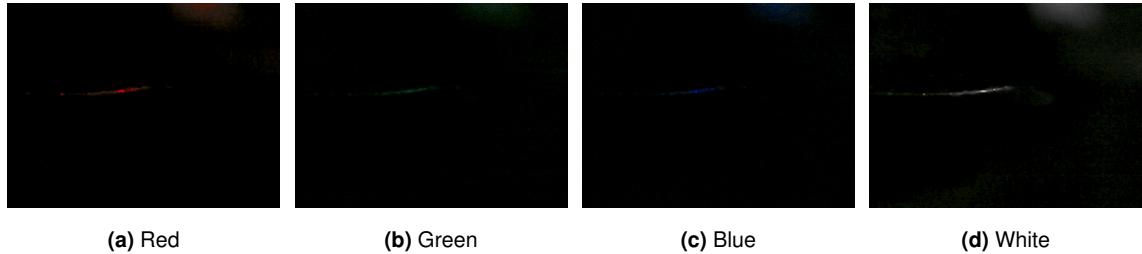


Figure 3.15: Red, green, blue and white lighting on insert 5 from batch 3 led strip two. Left to right respectively.

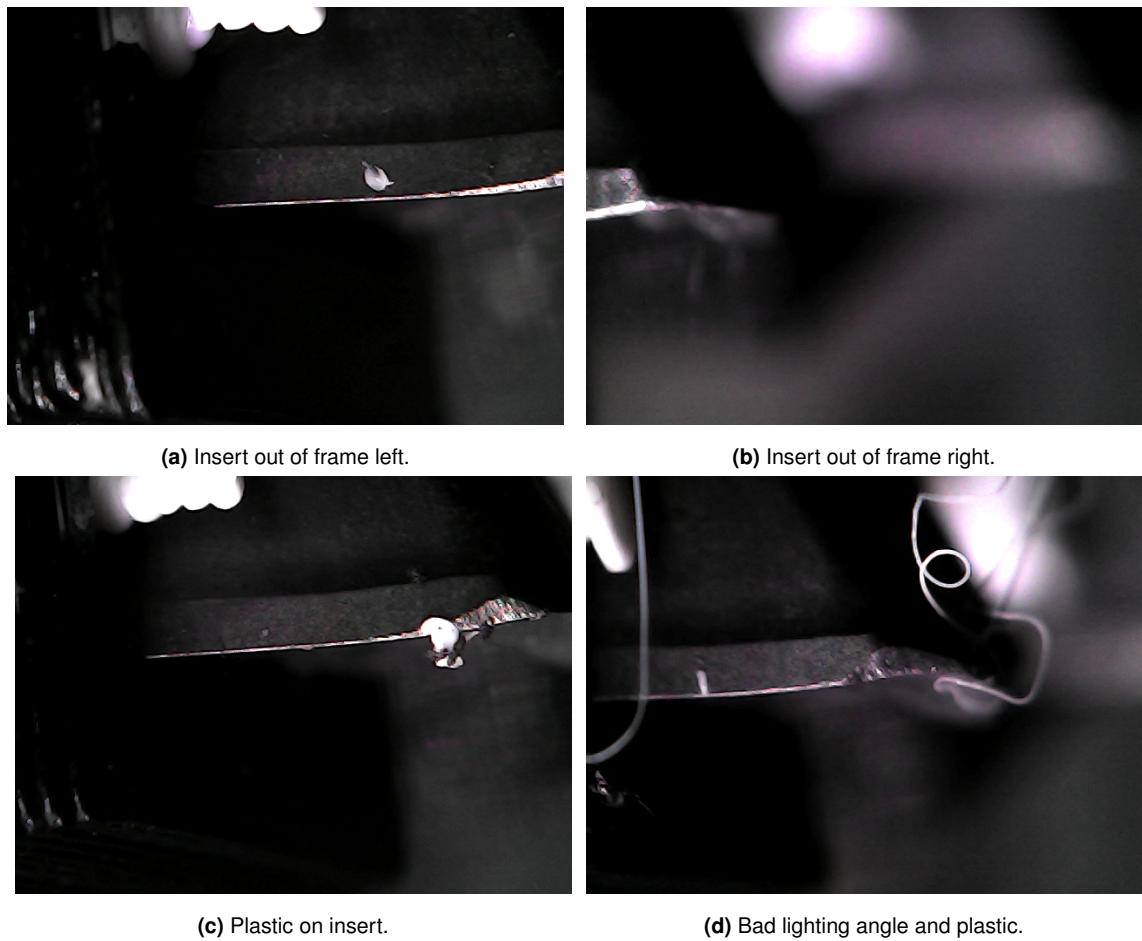


Figure 3.16: left: batch 3 insert 10 the side without bullet is off to the right side. right: batch 3 insert 10 The side with bullet. Is far off to the left side.

Figure 3.17: a. Plastic occludes the worn area. batch 3 plate 7 no bullet: had restants of 3D print plastic on worn area

could have an influence on the reflection of the carbide of which the inserts are made see light reflection.

3.2.4 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.

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 paper. Turning on a led to much on the bottom boundary makes the background very bright which supports 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

where

- aaa is the batch number;
- bbb is the plate number,
- 006-011 are the leds that turned on at the same time;
- color is the color: 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.

The dataset inserts consisted of a few different types and coatings.

3.2.5 Setup

The setup used is exactly the same as on the Birthday dataset where the camera is positioned as much to the top as possible. This can be seen on the picture:

Here 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 camera angle is kept the same a little to the top and very close to the inserts as seen on Figure 3.18

Through the whole dataset the direct light coming from other sources eg. the light of the room was blocked to have full control of the lighting conditions.

3.2.6 Results

Underneath are some pictures of good examples in the dataset.

Underneath are two pictures of batch 3 insert 6. These were lid with red light on the left and white light on the right. 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. The discussion of these difficulties will be bespoken in vision algorithm.

Some pictures aren't sharp like the one shown next.

like batch 5 insert 5 without bullet.

Different insert types First type are the grey inserts with very visible wear. These are seen in batches 1 to 5 consistently. This type will be called grey inserts.

Then there are other inserts in batch 11 which are also grey but have a different shape on the cutting part. These will be called rounded grey inserts.

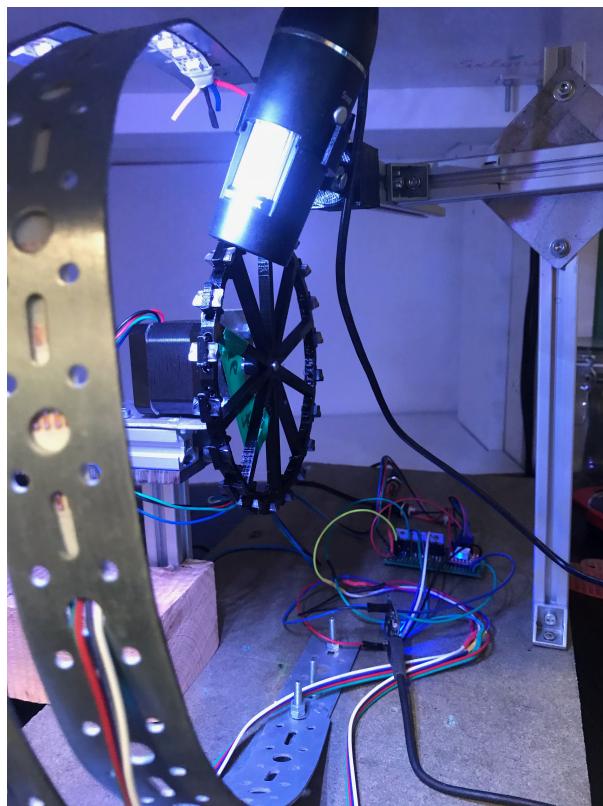


Figure 3.18: Test setup camera angle more to top.

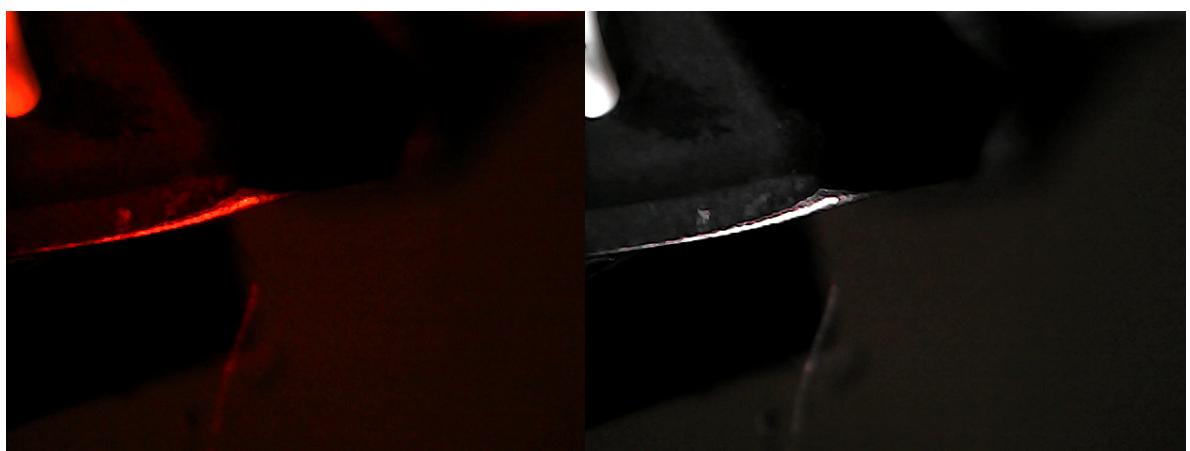


Figure 3.19: Example images for red and white light for spaghetti dataset.

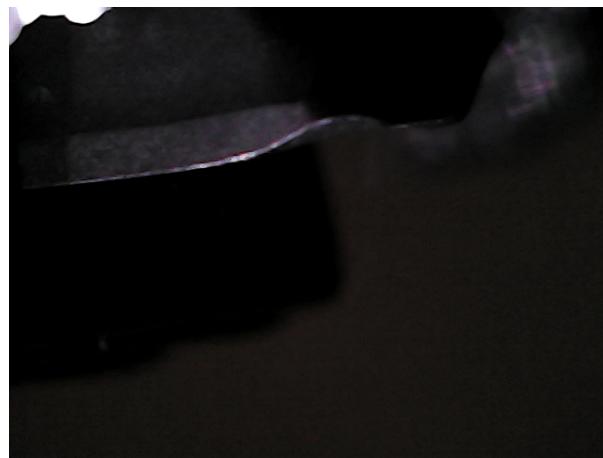


Figure 3.20: Unsharp image

In batch 12 and 13 there are the same shape of inserts but with a black coating which results in way darker pictures as seen here on batch 13 insert 2 no bullet. These inserts are the rounded black inserts.

The next type are copper colored inserts with the rounded shape. Seen in batch 14 insert 5 no bullet.

Than we have inserts with a gold coating and hooked shape. For batch 15 insert 6 no bullet that gives the next picture:

3.3 Google Colab

3.4 Test Camera Setup

Created woensdag 04 november 2020

To test the camera setup, a binary classification model is made. This model will tell with a threshold of 150 (200 the real threshold but 150 to warn before tool is worn out) whether a tool is good to work with or must be removed from the machine.

This model should be trainable with as less images as possible, preferably 20 because that is the amount of pictures taken in one batch.

3.4.1 Resnet18

First we will try to implement SDD-Resnet18 to classify the few images in good or bad.

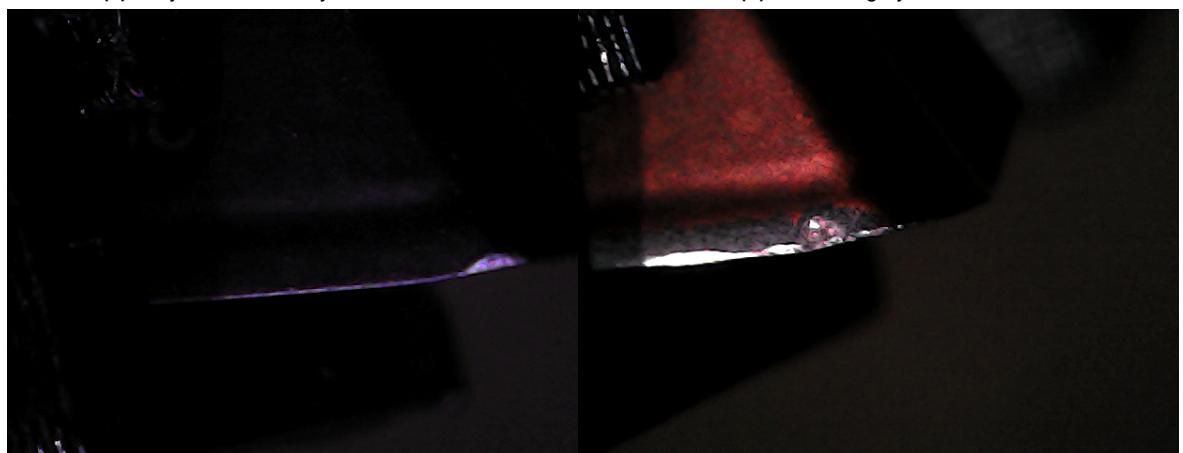
3.4.2 Inception v3

Than we will implement SDD-inception v3



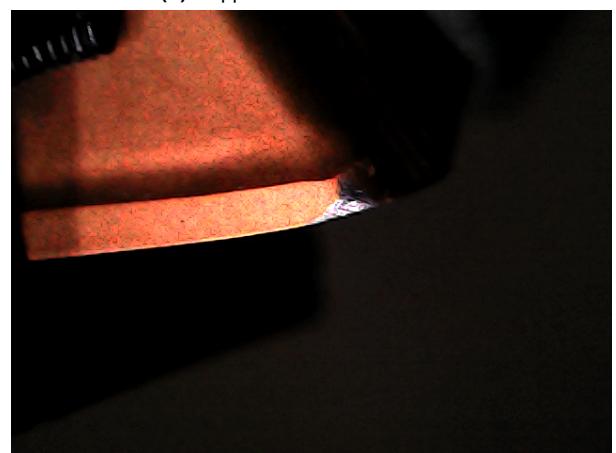
(a) Grey insert with very visible wear.

(b) Rounded grey insert.



(c) Rounded black insert.

(d) Copper coloured insert.



(e) Gold coloured insert.

These described models should perform rather good without any transfer learning.

After the first tests these results can be compared with a transfer learning model.

3.4.3 All Networks

For this all networks test, some networks where tested and an algorithm is created to make it possible to add more networks along the way

3.5 All Networks

Created woensdag 02 december 2020

3.6 All Networks 1

Created woensdag 02 december 2020

All networks 1 can be found [here](#).

3.6.1 Network architectures

This model is used to test different model architectures namely:

- Resnet18
- Alexnet
- VGG11.bn
- Squeezezenet
- Densenet

inception v3 didn't seem to work

These models are all relatively small and should provide quite good results for a small dataset.

More on why the models should work can be found [here](#)

3.6.2 Dataset

This algorithm had the input of images from the Second handmade dataset which was devided into 3 classes based on their measured wear value.

class	min value (micron)	max value (micron)
good	0	130
medium	130	230
bad	230	infinity

3.6.3 Results

Results of this notebook are available on wandb as pytorch-TWI_second_handmade

Interesting results will be bespoken here;

Tests for different models:

model name	test accuracy %	validation accuracy%	transfer learning
Alexnet	100	90	yes
VGG11.bn	89	85	yes
Densenet	89	85	yes
Squeezezenet	89	85	yes
Resnet18	89	90	no

An overview of the best runs for every model architecture. Since there are only nine test images; the test scores are set to a very high granularity. Further results of this test are to be found on wandb as Testing on first handmade dataset

3.7 All Networks 2

Created vrijdag 04 december 2020

3.7.1 Network architectures

This model is used to test different model architectures namely:

- Resnet18
- Alexnet
- VGG11.bn
- Squeezezenet
- Densenet

inception v3 didn't seem to work

These models are all relatively small and should provide quite good results for a small dataset.

More on why the models should work can be found [here](#)

3.7.2 Dataset

This algorithm had the input of images from the Second handmade dataset which was devided into 3 classes based on their measured wear value.

class	min value (micron)	max value (micron)
good	0	130
medium	130	230
bad	230	infinity

3.7.3 Results

Results of this notebook are available on wandb as pytorch-TWI_second_handmade

Interesting results will be bespoken here;

Tests for different models:

model name	test accuracy %	validation accuracy%	transfer learning
Alexnet	100	100	yes
Resnet18	100	95	no
Densenet	100	85	yes
Squeezezenet	100	85	yes
VGG11.bn	100	90	no

An overview of the best runs for every model architecture. Since there are only nine test images; the test scores are set to a very high granularity. Further results of this test are to be found on wandb as Testing on first handmade dataset

Trying to sweep over different parameter settings didn't work on my local computer; all runs failed or crashed

3.8 All Networks 3 Birthday

Created vrijdag 04 december 2020

The code for this project is to be found here: TSU_AllNetworks_3_Birthday

3.9 All Networks 4 Spaghetti

Created vrijdag 04 december 2020

The code for this can be found in here: TSU_AllNetworks_4_spaghetti

The report is noted in Spaghetti sweep with TSU_AllNetworks_4_spaghetti This can be transformed into latex without further problems i hope.

3.10 All networks 4 spaghetti first 5 batches

Created vrijdag 04 december 2020

Only the first five batches are analysed in this report to check if the dataset is as good as the dataset created by hand of these batches.

Also the difference is checked between the red and white leds.

3.11 Resnet18

Created woensdag 18 november 2020

In this page we will describe the results and actions taken to get results out of Resnet18
this paper suggests that this is a good architecture for a quite like problem. where a low amount of data is used.

SDD-CNN: Small data-driven convolution neural networks for subtle roller defect inspection

3.11.1 Creating first file

3.11.2 TSU_Resnet18_1

failed to load data, did copy files into correct directories and created dataset class

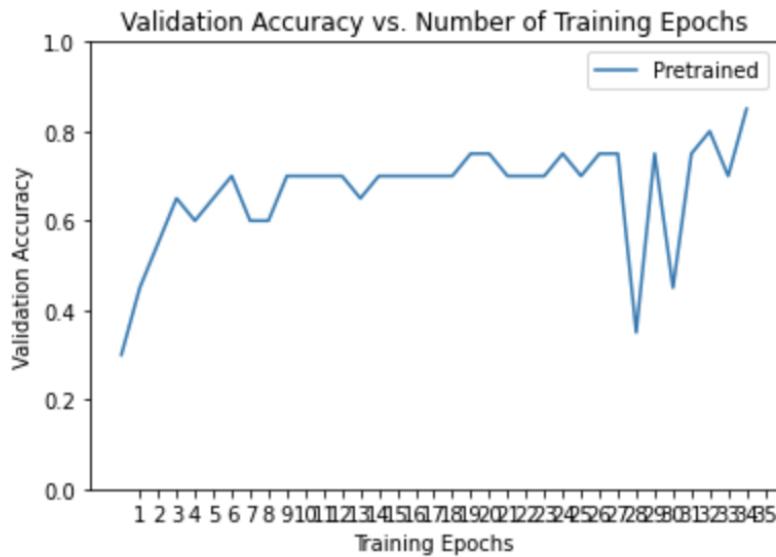
3.11.3 TSU_Resnet18_2

Simpeler method to read in data and not be able to change a lot of things;
next time build dataset class self with the same model.

After that the regression model would be easy

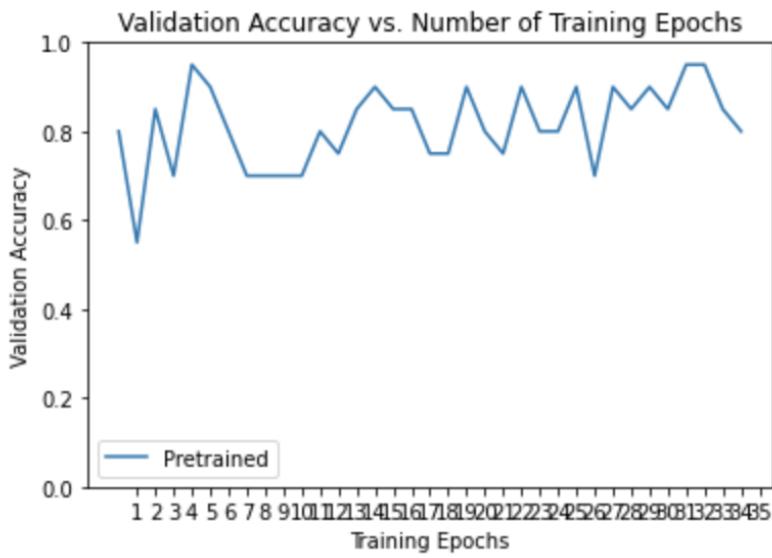
First result of the training.

```
num_epochs: 35
num_classes: 3
batch_size: 12
learning_rate: 0.001
momentum: 0.9
feature_extract: False
```



Tweede test met een aanpassing van de learning rate en een foto van de training set naar test set gebracht

```
num_epochs: 35
num_classes: 3
batch_size: 12
learning_rate: 0.005
momentum: 0.9
feature_extract: False
```



Next time create results file to get nice overview of all results

Chapter 4

Results

4.1 Setup

A fully adjustable setup is created and used to create datasets.

To obtain results all datasets are tested with the described program from ???. The output is listed for every dataset starting with the second handmade dataset than 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.

The results and graphs where obtained with Google Colab on a regular GPU and for the processing of the logs Weights and Biases was used.

4.2 Second handmade dataset results

The second handmade dataset was created in 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 different model architectures as described in ???. The influences of parameters are then discussed to conclude with a few example results on the test set.



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

4.2.1 overall results for the five best runs.

For the five best results we get up to a validation accuracy of 100%, a test accuracy of 100% and a train accuracy of 97%. 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.

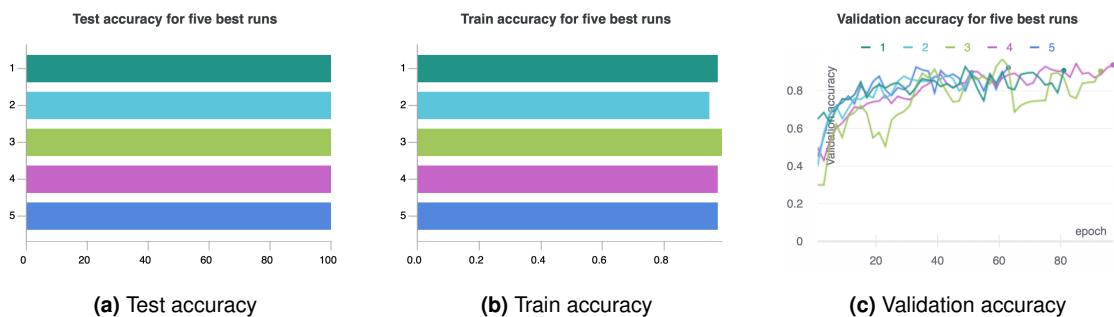


Figure 4.2: Test, train and validation accuracy for five best runs with second handmade dataset.

4.2.2 Comparison between the different model architectures

To get a better overview on how different architectures perform for the dataset, we compare the main results for all these tested architectures. Starting with Test accuracy on these models, than 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 class wasn't predicted correctly. This is result of testing 900 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.

A simple confution matrix is created for 9 runs with 9 images each. top to bottom truth left to right predicted

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

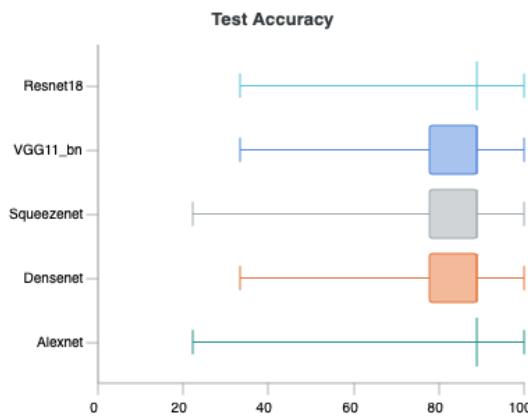


Figure 4.3: Test accuracy for different model architectures with maximum and minimum values.

Compare this with the distribution and values. The border between low and medium is at 130 micrometer and the border between Medium and High wear is at 230 micrometer. Since these limits are just a number there are a lot of values just before and just after the limit. This makes it hard for the algorithm and isn't even possible to see a difference between these 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 the other way.

4.2.2.2 Validation accuracy

The validation accuracy is plotted in function of the epochs which gives an overview of how the training went. In the beginning there is a learning curve and at the end the graph stabilises. This means the training got a good amount of epochs to reach the optimal point.

This 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 ... There are two measurement methods 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.

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%

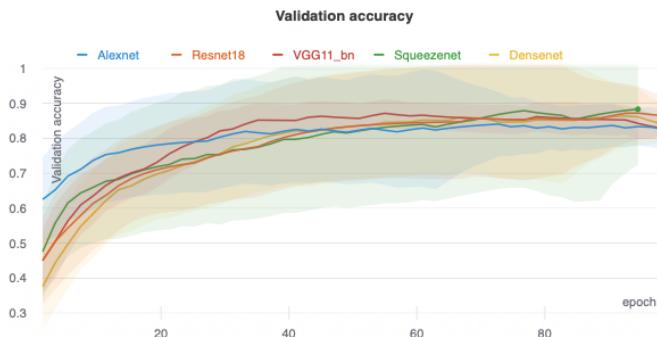


Figure 4.4: Best validation accuracy for different model architectures.

4.2.3 Influence of Transfer learning on the results

The influence of transfer learning from the imangenet dataset is visualised in the following graphs. First the test accuracy is plotted for every model architecture and with transfer learning (TL) or without transfer learning (No TL) in Figure 4.5a. Here 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 model architecture.

The second graph on Figure 4.5b plots the validation accuracy for the different networks with or 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 influences of parameters

The settings for the best 40 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 green lines indicate a validation accuracy of more than 80% for that run.

Other things we can learn from this graph are:

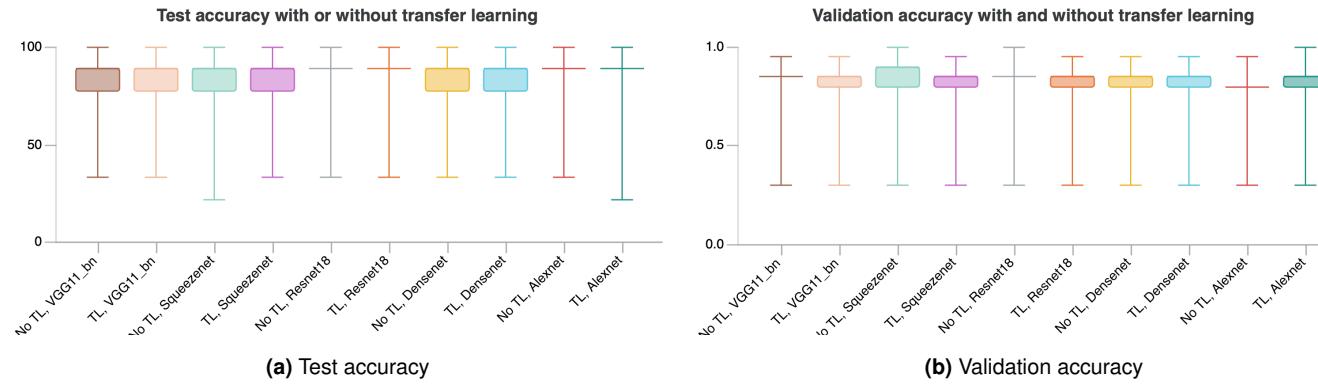


Figure 4.5: Validation and test accuracy box plot for different model architectures with or without transfer learning (TL).

- 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.

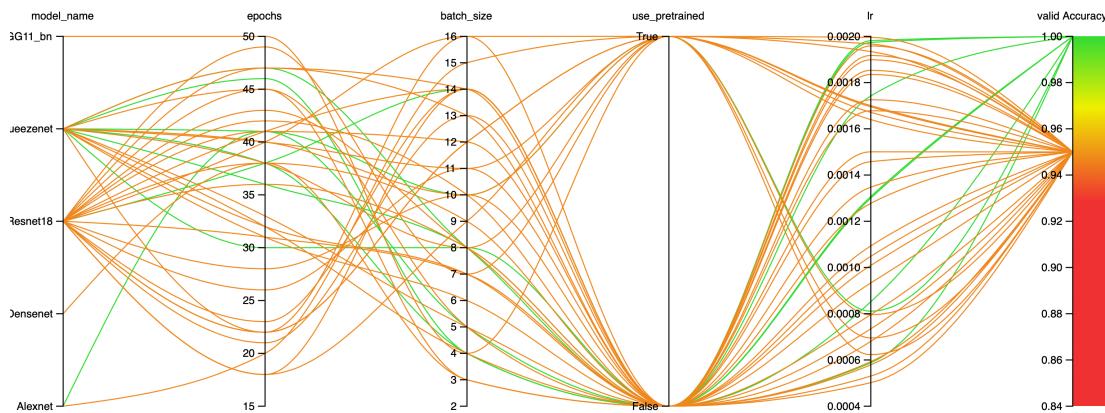


Figure 4.6: Parameter influence on validation accuracy.

4.2.5 test images

Use these images to show what high wear, medium wear and low wear looks like? and leave this section out Since the test accuracy was so high only a few images were mispredicted



(a) Truth: High
Predicted: High

(b) Truth: Medium
Predicted: Medium

(c) Truth: Low
Predicted: Low

Figure 4.7: Result images with their results.

4.3 Birthday dataset results

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

4.4 Spaghetti dataset results

The spaghetti dataset was the first dataset to capture all of the currently available tool inserts. Here the different insert types are all in 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 4.8a. The maximum test accuracy is 82.1%, this value is not as expected. To dive deeper in the training we take a look at the training accuracy in Figure 4.8b 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 in 20% of the cases. 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 over fitting. Figure 4.8c shows the training and validation accuracy during training. There is no overfitting seen on this graph so the model just isn't powerful for the task or the size of the dataset is not large enough.

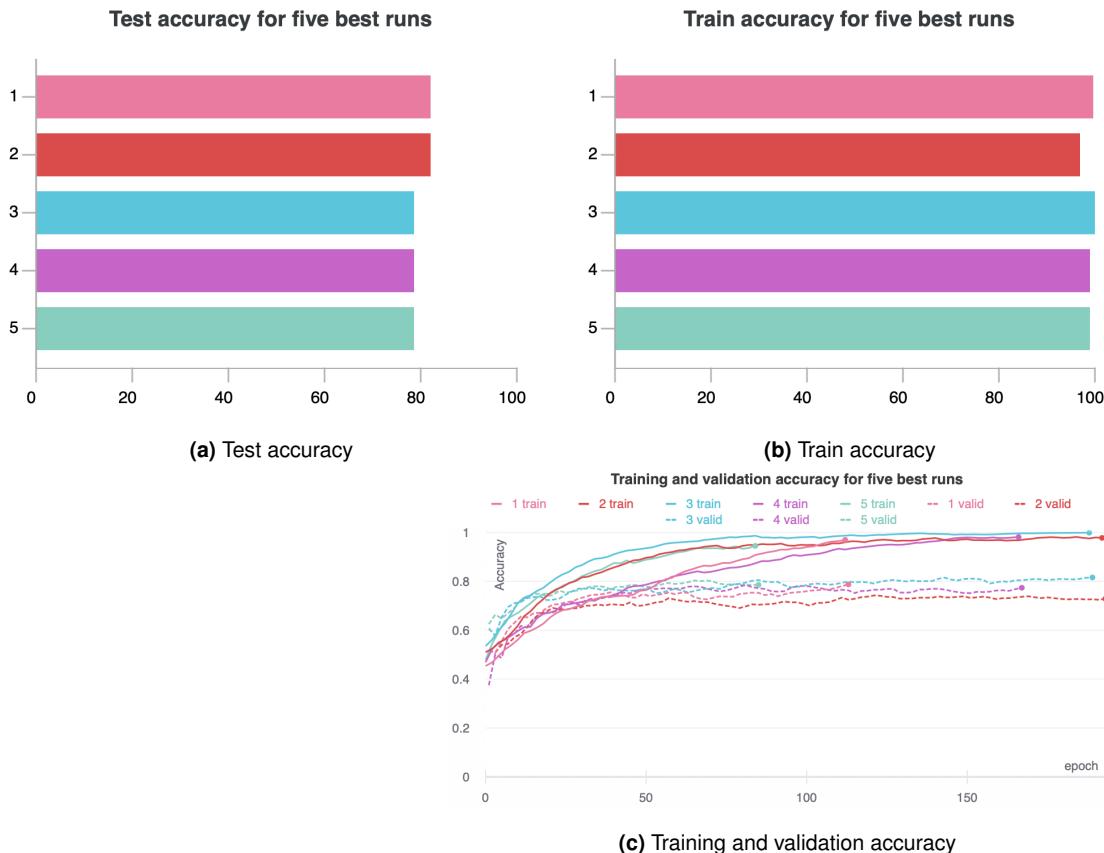


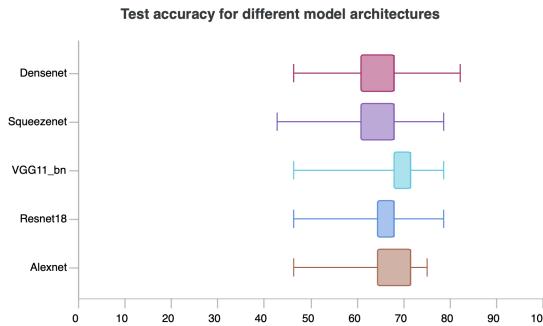
Figure 4.8: Test, train and validation accuracy for the five best runs based on their test accuracy scores.

4.4.2 comparison between different model architectures

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

4.4.2.1 Test accuracy

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

**Figure 4.9:** 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 of the test set. After those two Densenet with an accuracy of 78%, Alexnet with an accuracy of 75% and finally Resnet 18 with an accuracy of 73%. Here must be noticed that Resnet 18 only got tested in 3 different parameter configurations whereas the others where optimized a bit more.

Table ... shows the results for the different datasets 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.

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

4.4.4 Influence of transfer learning on the results

For comparison with the second handmade dataset the transfer learning influence is plotted in Figure 4.11a for the test and validation accuracy. Overall Transfer learning 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 over the runs without transfer learning.

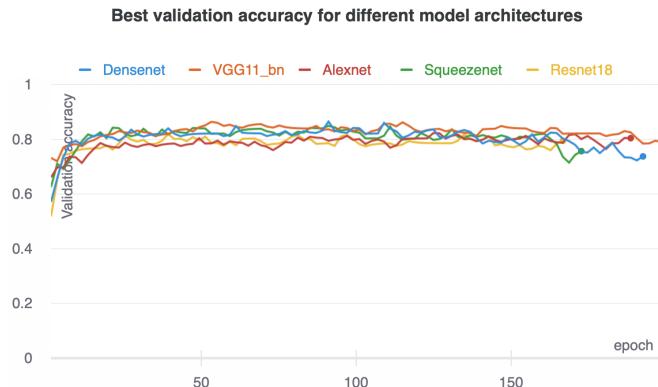


Figure 4.10: Best validation accuracy for different model architectures plotted in relation to the amount of epochs.

The same difference is also compared for all different model architectures in Figure 4.11b. SqueezeNet, 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 4.11c, secondly the train accuracy is plotted in Figure 4.11d. For the validation accuracy the learning curve with transfer learning inclines a bit before the curve for the training without transfer learning. using a pre-trained network speeds up the initialisation of the network but also uses more epochs to finish the training. For the training accuracy this curve there is no significant difference.

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 4.12a and 4.12b. For the validation accuracy there is a small difference of 4% between the two curves where the white lighting seems to perform better. On the test accuracy there is a similar outcome for the median results but the best results got achieved by the dataset with red lighting.

4.4.6 training runs

On figure 4.13 different runs with different parameters are displayed. Here can be seen that more green lines start from the white dataset. This is in line with previous findings where was stated that the white dataset performed better on average. The batch size can be finetuned between 16 and 2. The amount of epochs can be increased in an attempt to better the results, however on Figure 4.8c 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.

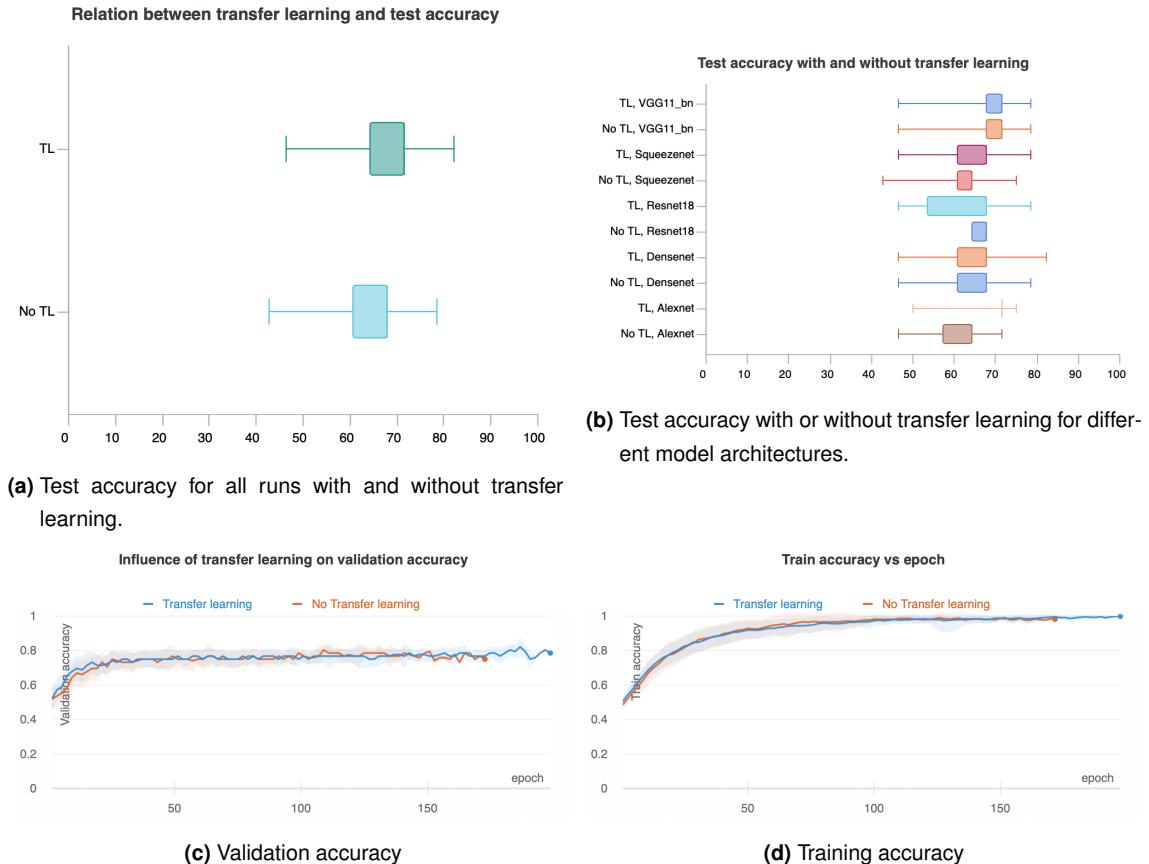


Figure 4.11: Test accuracy for models trained with and without transfer learning. In picture a) a general overview of all performed tests. in picture b) a comparison between different model architectures. Picture c and d provide the differences during training for validation accuracy and training accuracy respectively.

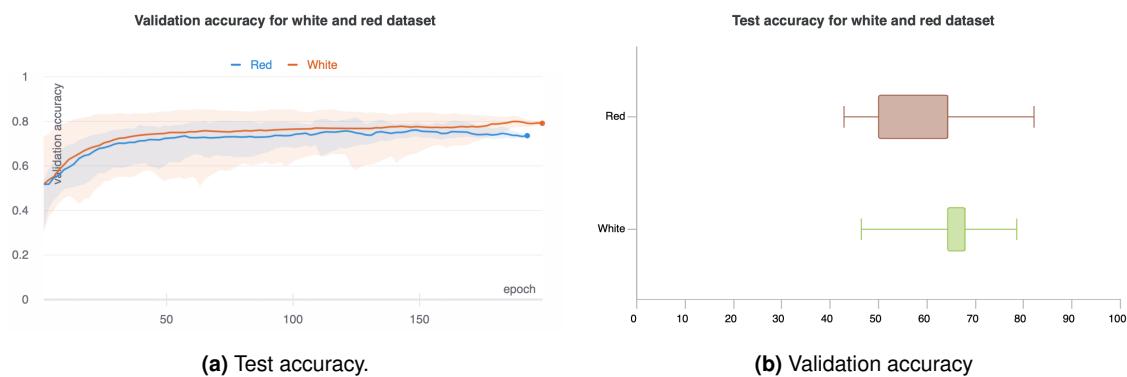


Figure 4.12: Test and validation accuracy for a different led color.

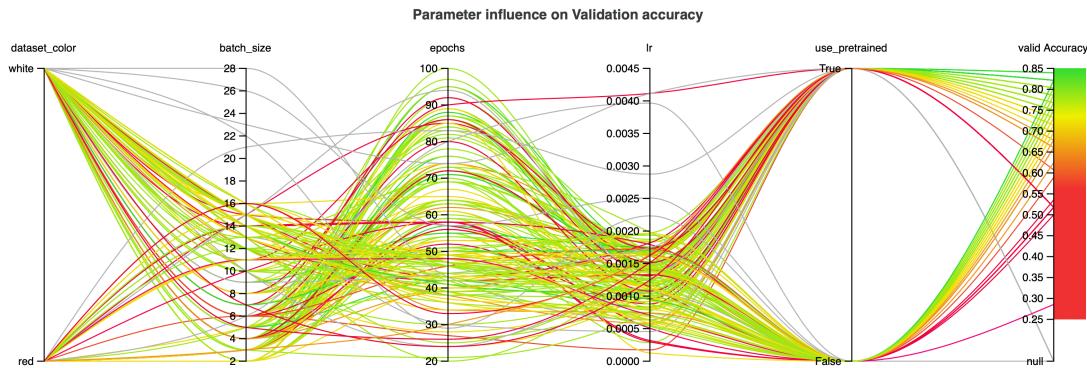


Figure 4.13: Parameter influence on validation accuracy.

4.4.7 Test images

To illustrate the results, some test images from the white lighted part of the dataset are listed in Figure 4.14. One error is given in Figure 4.14c Where high wear was predicted for a medium wear insert. No hard mistakes where resulting in this test set where low wear was predicted as high wear or the other way.

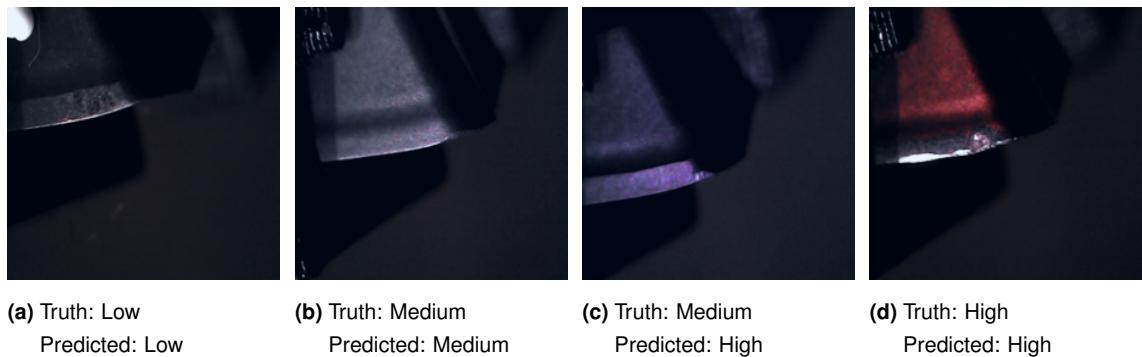


Figure 4.14: Images with predictions from the spaghetti dataset with white lighting.

For the test set with red lighting are also images listed in Figure 4.15. Figure 4.15c displays an error where a medium wear inset was predicted as high wear. This also is not a big error.

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 dataset is more than doubled but the results for the test are worse. To give an overview of the dataset three inserts are listed from the datasets:

- one from the second handmade dataset in Figure 4.16a

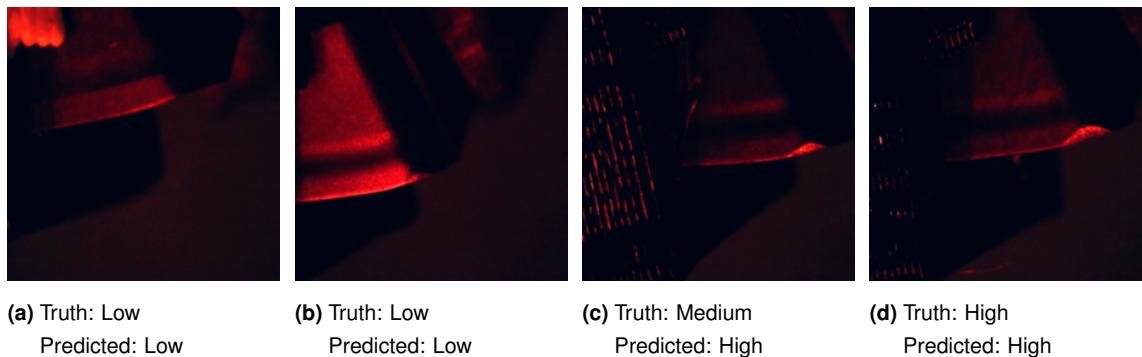
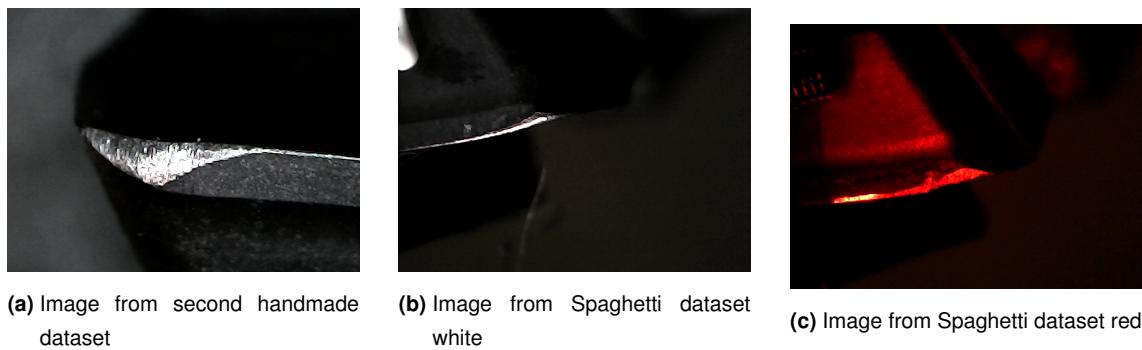


Figure 4.15: Images with predictions from the spaghetti dataset with white lighting.

- one from the white part of the Spaghetti dataset in Figure 4.16b
- one from the red part of the Spaghetti dataset in Figure 4.16c



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

4.5.1 Global comparison

For the comparison the Spaghetti dataset is split up into a white and a red part so the results of 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 4.17a we see that the second handmade dataset requires much less training epochs and increases in train accuracy much quicker. For the validation accuracy in Figure 4.17b 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 what means the training runs where much more fluctuating. From these results we can conclude that the spaghetti dataset is much harder to learn. For the confirmation we look at Figure 4.17c where the test accuracy is plotted. The same can be extracted from this figure as from the orhter two. Where 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.

max accuracy for second handmade=

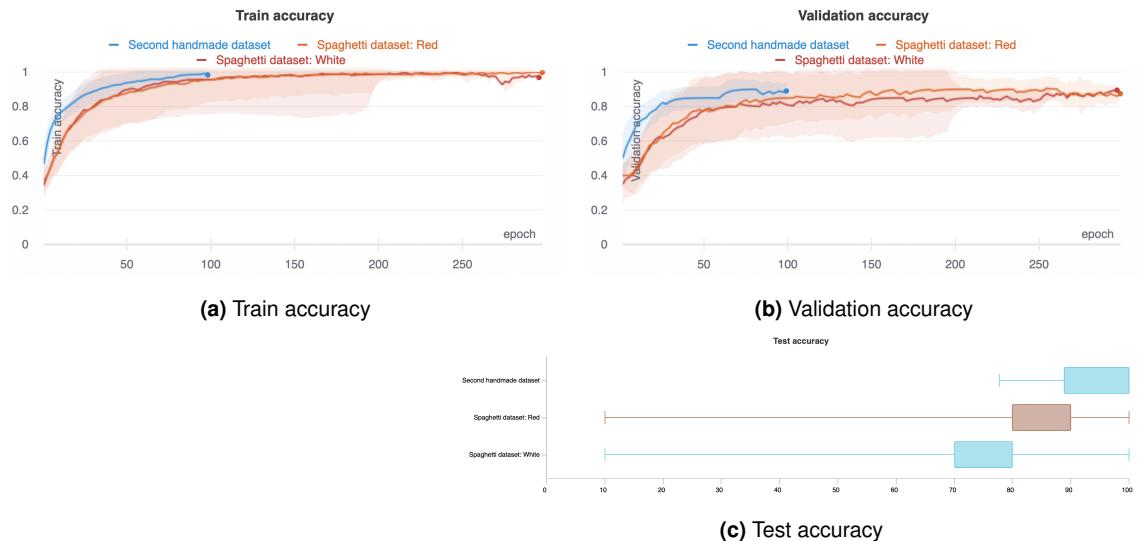


Figure 4.17: Comparison for test, train and validation accuracy for different datasets.

If we compare the best results obtained for the second handmade dataset and the spaghetti dataset based on the validation accuracy (first column) and on the test accuracy (second column) the results seem very familiar. The same order is kept and even the values don't differ much. Alexnet, Resnet 18 and SqueezeNet can be said to be the best architectures for this problem with the smaller, simpler dataset. This table can also be compared against the table from the full spaghetti dataset where the order is basically reversed.

These results can be declared based on the Table 2.1 Alexnet, Resnet 18 and SqueezeNet 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.

Model name	Validation Accuracy	Validation accuracy sorted on test accuracy
Alexnet	100%	100%
Resnet 18	100%	100%
SqueezeNet	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 4.18 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 taken for these models as for the global results for these datasets. The model architecture doesn't incorporate a sufficient difference in results.

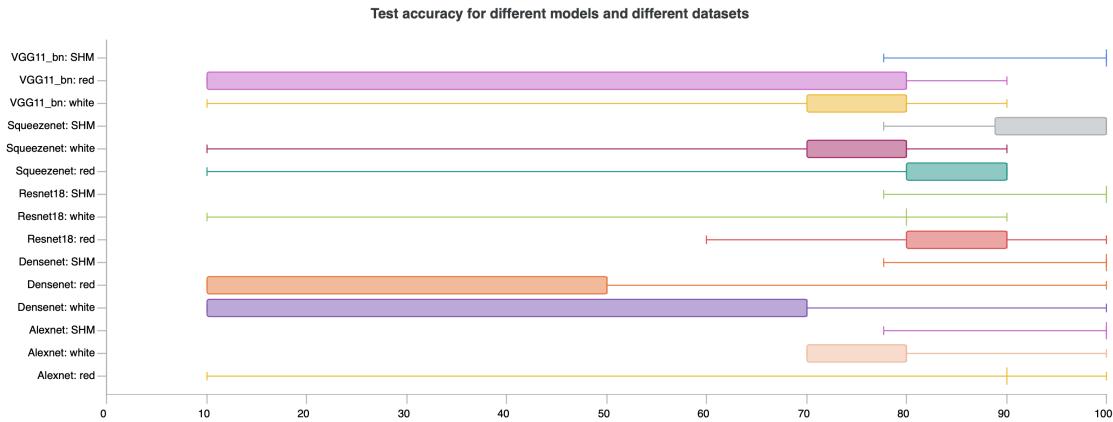


Figure 4.18: Difference in test accuracy for different datasets trained with different model architectures.

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

5.1 Inleiding

De referentielijst bevat de volledige lijst van literatuur en bronnen waarnaar in de tekst wordt verwezen. Door systematisch de referentielijst aan te vullen bij het schrijven van het literatuuroverzicht gaat er achteraf geen tijd verloren aan het opnieuw opzoeken van referenties.

5.2 Referentiestijl

Voor het verwijzen naar informatiebronnen wordt gebruik gemaakt van het numerisch systeem of van het auteur-jaar systeem. Dit kies je door volgend commando in het latex bronbestand aan te passen:

- numerisch (IEEE) : \bibliographystyle{ieee}
- alfabetisch (APA) : \bibliographystyle{apalike}

Plaats je bronnen in een *bibtex* bestand (evt. via software zoals bv. Jabref Endnote of Mendeley), waarnaar je verwijst vanuit je thesis text a.d.h.v. het commando \cite. Enkele links naar nuttige software in deze context:

- JabRef (Open Source)
- Mendeley (Freeware)
- EndNote (Paid license)

Indien je zelf een .bibtex bestand wil aanleggen dien je volgende syntax te volgen voor een tijdschriftartikel:

```
@article{hughes2005,  
title={Isogeometric analysis: CAD, finite elements, NURBS, exact geometry  
and mesh refinement},  
author={Hughes, Thomas JR and Cottrell, John A and Bazilevs, Yuri},  
journal={Computer methods in applied mechanics and engineering},  
volume={194},  
number={39},  
pages={4135--4195},  
year={2005},  
publisher={Elsevier}  
}
```

Enkele voorbeelden van het gebruik van bronnen in een tekst (in APA stijl):

Recent werd het Higgs boson experimenteel vastgesteld door Aad et al. Aad et al. (2012) (syntax: \cite{aad2012}).

Als alternatief voor het discretiseren van een CAD model vooraleer een eindige elementenanalyse te kunnen toepassen, stellen Hughes et al. voor om de nodige elementenformulering rechtstreeks uit de NURBS beschrijving van de CAD geometrie te halen Hughes et al. (2005) (syntax: \cite{hughes2005}). Daarnaast introduceren ze tevens een k-iteratieve procedure als een verificatie van de geldende p- en h-iteratieve procedures in eindige elementen methoden Cottrell et al. (2009) (syntax: \cite{cottrell2009}).

Chapter 6

Bibliography

- (2020). Raspberry pi user manual. <https://www.raspberrypi.org/documentation/>. Accessed: 2020-12-10.
- Aad, G., Abajyan, T., Abbott, B., Abdallah, J., Khalek, S. A., Abdelalim, A., Abdinov, O., Aben, R., Abi, B., Abolins, M., et al. (2012). Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29.
- 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.
- 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*.
- Cottrell, J. A., Hughes, T. J., and Bazilevs, Y. (2009). *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons.
- 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*.
- Hughes, T. J., Cottrell, J. A., and Bazilevs, Y. (2005). Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39):4135–4195.
- I, i. (2020). Keras. <https://keras.io>. Accessed: 2020-12-09.
- 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*.
- 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

Uitleg over de appendices

Bijlagen worden bij voorkeur enkel elektronisch ter beschikking gesteld. Indien essentieel kunnen in overleg met de promotor bijlagen in de scriptie opgenomen worden of als apart boekdeel voorzien worden.

Er wordt wel steeds een lijst met vermelding van alle bijlagen opgenomen in de scriptie. Bijlagen worden genummerd het een drukletter A, B, C,...

Voorbeelden van bijlagen:

Bijlage A: Detailtekeningen van de proefopstelling

Bijlage B: Meetgegevens (op USB)

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
www.fet.kuleuven.be

