

# Can depth data improve the accuracy when classifying mops?

Stian Blomdal



Thesis submitted for the degree of  
Master in Applied Computer and Information Technology (ACIT) -  
Robotics and Control  
30 credits

Department of Computer Science  
Faculty of Technology, Art and Design

OSLO METROPOLITAN UNIVERSITY

Spring 2022



# **Can depth data improve the accuracy when classifying mops?**

Stian Blomdal

© 2022 Stian Blomdal

Can depth data improve the accuracy when classifying mops?

<http://www.oslomet.no/>

Printed: Oslo Metropolitan University

# Preface

This thesis is based on a collaboration project, and all the people who played a part deserves my gratitude. First of I would like to express my gratitude to my main supervisor Henrik Lieng. Thank you for the support and insight. I would also like to thank Dudor Morar from Inwatec for your patience and willingness to always help. Thank you Børge Aguirre for your support and trust the last couple of years.



# Abstract

Since the emergence of low cost RGB-D cameras, a new world of possibilities have opened up in the field of Computer Vision. This projects focuses on both the practical and theoretical part of how depth data can improve the accuracy when classifying objects in an industrial environment. We have tested both classical machine learning methods and Google's Residual Neural Network: MobilNetV2. The goal was to achieve an accuracy that can match HF RFID-tag(95-97%). The purpose of the research question was to find out more about RGB-D images and what methods that can be used best for classification.





# Contents

<b>Preface</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Statement . . . . .	1
1.3 Constraints . . . . .	2
<b>2 Background and Related Work</b>	<b>5</b>
2.1 Background . . . . .	5
2.1.1 Interpretability and Explainability . . . . .	5
2.1.2 Machine learning . . . . .	6
2.1.3 SVM . . . . .	6
2.1.4 KNN . . . . .	7
2.1.5 Neural Networks . . . . .	7
2.1.6 Deep Learning . . . . .	7
2.1.7 Deep Neural Networks . . . . .	8
2.1.8 Convolution Neural Network . . . . .	9
2.1.9 Padding and Stride . . . . .	11
2.1.10 Residual Networks . . . . .	12
2.1.11 Transfer Learning . . . . .	13
2.1.12 Stereo Camera Sensing . . . . .	13
2.2 Related Work . . . . .	14
2.2.1 RGB-D . . . . .	14
2.2.2 Semantic Segmentation . . . . .	15
2.3 Tools . . . . .	15

2.3.1	Realsense software development kit . . . . .	15
2.3.2	Realsense D435 . . . . .	16
2.3.3	C++ and Python . . . . .	17
<b>3</b>	<b>Methodology</b>	<b>19</b>
3.1	Transfer learning . . . . .	19
3.2	Preprocessing . . . . .	20
3.3	SVM and kNN . . . . .	21
3.4	Experiments . . . . .	21
3.5	Data collection . . . . .	22
3.6	Data analysis . . . . .	22
<b>4</b>	<b>Implementation</b>	<b>23</b>
4.1	Hardware . . . . .	23
4.2	Image capturing . . . . .	24
4.2.1	Auto-labeling . . . . .	24
4.2.2	Manual labeling . . . . .	24
4.3	Model development . . . . .	26
<b>5</b>	<b>Results</b>	<b>27</b>
5.1	MobilenetV2 . . . . .	27
5.1.1	Baseline . . . . .	27
5.1.2	Number of MNV2 . . . . .	29
5.2	SVM KNN . . . . .	31
5.2.1	Number of SVMKNN . . . . .	31
<b>6</b>	<b>Discussion</b>	<b>33</b>
<b>7</b>	<b>Conclusions</b>	<b>35</b>
<b>A</b>	<b>Listings</b>	<b>43</b>
A.1	Code . . . . .	43
A.1.1	Image capture . . . . .	43
A.1.2	Pretrained MobileNetV2 . . . . .	53
A.1.3	SVM . . . . .	53
A.1.4	KNN . . . . .	53

A.2	Project management . . . . .	53
A.3	Test results . . . . .	54
A.3.1	MobilnetV2 code . . . . .	60
A.3.2	SVM and KNN code . . . . .	67



# List of Figures

1.1	Two white . . . . .	2
1.2	Two blue . . . . .	2
1.3	Two green . . . . .	2
2.1	Machine Learning Methods . . . . .	6
2.2	Adaptive Combiner. $x = input, w = weight, b = bias, y = output$ . . . . .	7
2.3	DNN [24] . . . . .	8
2.4	Visual vs convolutional hierarchy[27] . . . . .	10
2.5	Input and Filter [9] . . . . .	10
2.6	Input x filter $\rightarrow$ Feature Map[9] . . . . .	10
2.7	Colours represent different filters [9] . . . . .	10
2.8	<i>Maxpooling</i> [9] . . . . .	11
2.9	Residual block . . . . .	13
2.10	Illustration of camera . . . . .	16
3.1	Normalized depth . . . . .	20
3.2	Normalized depth with ROI . . . . .	20
3.3	Double Normalized depth with ROI . . . . .	20
3.4	White mops avg . . . . .	21
4.1	Caption . . . . .	23
4.2	Marked door . . . . .	25
4.3	Manual feeding . . . . .	25
4.4	Final RGB . . . . .	25
5.1	Baseline RGB . . . . .	28
5.2	Baseline Depth . . . . .	28

5.3	Best white test . . . . .	31
5.4	Best white test metrics . . . . .	31
5.5	Best green test . . . . .	32
5.6	Best green test metrics . . . . .	32
5.7	Best blue test . . . . .	32
5.8	Best blue test metrics . . . . .	32

# List of Tables

5.1	Top results White Mobilnet with <b>PPM1:</b> . . . . .	29
5.2	Top results Green Mobilnet with <b>PPM1:</b> . . . . .	30
5.3	Top results Blue Mobilnet with <b>PPM1:</b> . . . . .	30
5.4	Top results from SVM and KNN with <b>PPM2:</b> . . . . .	31
6.1	Average results . . . . .	33
A.1	Tentative Project plan . . . . .	54





# Chapter 1

## Introduction

### 1.1 Motivation

Nor Tekstil is a company providing industrial laundry services nation-wide in Norway. One branch of their services is delivering clean floor mops to their customers. At the laundry in Drammen, they make use of a machine for packaging mops - this is where Inwatec becomes important. Inwatec is a Danish robotics company, developing advanced software and smart machinery to automate workflows in the industrial laundry industry, and is also the manufacturer of the packaging machine. After the mops are washed and dried, they're packet into a plastic bags. Today this machine is able to count the number of mops going through it, before ending up in a bag. This is done by inserting a high frequency chip in each mop and installing a reader inside the machine. It is not able to classify the type of mop. This can become an issue if the machine is fed with a batch of mops different from the kind it's set to pack. Worst case, the customer is getting a different kind that they ordered. Nor Tekstil has several checkpoints today in their production line where one can find similar challenges. Sorting and counting garments throughout the production line can increase the amount of control variables, which again gives Nor Tekstil the ability to increase the effectiveness of the laundry. Understanding how to automate the task of depth perception is beneficial for all parts involved in this project.

### 1.2 Problem Statement

We want to capture and use RGB-D image data to classify and count mops while being packed by a packaging machine. Is it possible to achieve the same or better accuracy as HF RFID-chip classification (95-97%)? We need to mount a camera at a suitable position and angle. Then interface

with the packaging machine, get a signal from a photo electric sensor when mops are within the field of view of the camera, and capture the image data. Create a large enough dataset and label all images correctly. Create a baseline model, then build models using the most suitable methods for this project after doing extensive research of related work. Compare results and suggest further work. One can think about this classification problems as a set of 2 main tasks. Find out what type of object this is; a white, blue or green mop? When this is established, how many are there in the image?. By design the machine is able to separate mops from a big pile. The exact amount separated varies from one up to four. Lets call this a small pile. When the small pile is transported on the conveyor belt, its not always easy to see for a human how many exactly is in this pile. Imagine you take four mops and toss them on the floor, they will always land differently and sometimes four mops looks like two, one looks like two, and so-on. Each picture below inhabits two mops each - not easy to tell? How do one classify both the type and number of mops?



Figure 1.1: Two white



Figure 1.2: Two blue



Figure 1.3: Two green

**Research question one:** Can classification accuracy of RGB-D data be improved by breaking the problem into two parts?

**Research question two:** Can classical machine learning methods help to improve classification accuracy of depth images?

### 1.3 Constraints

Early on it became evident that there was going to be some constraints on how the topic should and could be researched. Main constraints were:

- Time (four and a half months)
- Access to code from related work
- Type of data the image sensor produce

On the basis of these constraints a couple of important decisions were made along the way in order to deliver a finished product on time. Use transfer learning for baseline model. This minimizes the need for time spent on coding, and gives us a realistic scenario of what method would be used in a real life image classification problem. Now time has been added to focus on testing other methods that hopefully can improve accuracy. Only methods that easily could be tested with the raw data produced was evaluated. This means that for example semantic segmentation was not tested. For semantic segmentation one needs to have a mask for every training image. This is done by manually drawing polygons around the objects to classify [37]. The main intuition when starting this project was that if one wants to find the type and amount, methods that don't use of the volumetric features in an image, will be no better than the state of the art 2D-classification methods. Occlusion of the same type of objects within an image makes it impossible at one point to distinguish between number of objects by only analyzing the 2D properties. As many as possible 3D-classification methods must be evaluated, but, this is not a paper about testing the state-of-the-art methods, it's about testing the methods that is suitable given the before-mentioned constraints.



## Chapter 2

# Background and Related Work

This chapter describes the practical and theoretical foundation of the project. The main focus is on the research topic, and related work. This is a Computer Vision and pattern recognition project, so when researching within a broad topic like this it is necessary to narrow it down to something more specific. There are several sub-categories and what they are should be clear after the end of this analysis. But it even in the beginning it's necessary to narrow it down beyond Computer Vision and pattern recognition. If one thinks of a picture as a two dimensional space, adding depth creates a third dimension. This gives us 3D images as a basis for field to study. Combining this with classification, and type of data produced by the camera, the following material has been created.

## 2.1 Background

### 2.1.1 Interpretability and Explainability

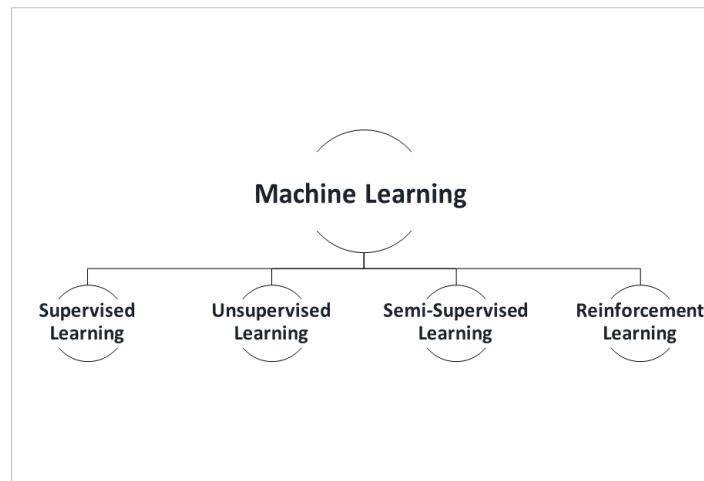
Ethical considerations related to this project is not that obvious. But in a general sense it should be in the back of every researchers mind when creating models that may have the potential to end up in an applications related to AI, or when using and modifying models and methods that are currently being used in AI. According to [21], Interpretability is to what extent one is able to predict the AI's decision, feeding it with a known input. [13] defines it as the ability of a human to understand the result a neural network outputs. Explainability or Explainable AI(XAI) is the methods and processes that allows humans to understand and trust the decisions made by the AI. A useful analogy is if you think about doing a science experiment. Interpretability is that you can see the different parts of the experiment and know if you light a match near the gas nosel, the gas will ignite. Explainability is digging into the chemistry behind it all. According to [13] theres been a surge of work in XAI. We need to fully understand what the model is doing for us to trust it. We

need to be able to do extensive debugging of the models we create in order to better understand them and learn from them. There is a lot of work being done on this, ref [43],[6],[30],[5]

## 2.1.2 Machine learning

As shown in the figure below, there are generally four major types of ML methods. ML is a method

Figure 2.1: Machine Learning Methods



for automating the process of model building. The idea is that a system can learn from data, find patterns, and make decisions with as little human interaction as possible[22]. An effective way to differentiate between the different sub-fields of ML, is to look at it from a data labeling perspective and how they gain insight from data. Supervised learning uses labeled data. Unsupervised uses unlabeled data and Semi-Supervised is a combination of both. Reinforcement learning uses no data, where the algorithm discovers through trial and error, and the goal is like a video game - get the highest score [22].

## 2.1.3 SVM

A SVM in its most basic form is a linear binary classifier that can identify a single boundary between two classes. For multi-class classification one can use a polynomial kernel. According to [7] do a polynomial kernel allow to map the input space into a higher dimensional space that is a combination of products of polynomials. Despite its high computational load, this kernel is frequently applied to data that has been normalized.

## 2.1.4 KNN

K-nearest neighbors algorithm (KNN), is a non-parametric, supervised learning classifier [24], which uses proximity to make classifications or predictions about the grouping of an individual data point. For a classification problem, a class label is assigned on the basis of a vote. The label that is most frequently represented around a given data point is used to determine what class the new observation belongs to, defined by the tuning parameter  $k$ . The following paragraph is taken from [34]: *"One Vs Rest Classifier, also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only  $n$  classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.*

## 2.1.5 Neural Networks

According to [10], Neural networks are based on algorithms that are present in our brain and help in its functioning. A Neural network interprets Numerical patterns which may be present in the form of Vectors. [24] noted that Artificial Intelligence consists of several subfields, and one of them is ML. In ML, there is a subfield called Neural Network (NN). This is the backbone of deep learning. One can think of a NN's as a set of algorithms trying to replicate the human brain's abilities. When designing a network like this, it's classically comprised of four components: inputs, weights, a bias or threshold, and an output, the relationship between them is shown in the figure below.

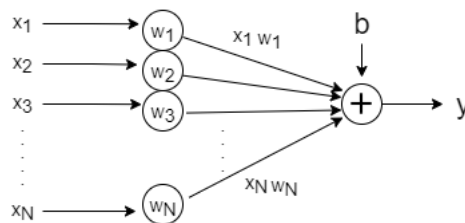


Figure 2.2: Adaptive Combiner.  $x = input, w = weight, b = bias, y = output$

## 2.1.6 Deep Learning

According to [10], Deep learning is a subset of machine learning that gives the system the capability to function like a human brain and imitate patterns that our brain does for making decisions

### 2.1.7 Deep Neural Networks

Based on [24][18], Deep Neural Networks (DNN's) are considered to be networks that are build by 3 or more layers, including input and output. One of the key benefits of deep learning algorithms is that they minimized the need for human interaction. The task of finding features in a dataset is automated. The downside of deep learning algorithms is that they need much more data in order to perform compared to the more classical machine learning algorithms. A popular way to visualize a DNN is shown below.

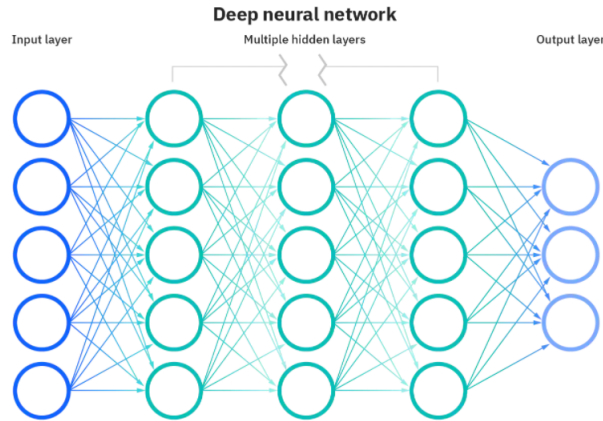


Figure 2.3: DNN [24]



## 2.1.8 Convolution Neural Network

According to [4], before Convolution Neural Network's (CNN's), hand-crafted features were created in order to perform classification. The breakthrough of CNN's is that features are learned automatically from training examples. CNN's proved to be very effective in image classification as the convolution operation captures the 2D nature of images. CNN's has been around for over 20 years, but because of large available data-sets and implementation of CNN on massively parallel GPU'S, its adoption has exploded the past 8-10 years(2016). CNN is based on the human visual cortex and for image and video recognition it's the most common one used [1]. The following subsections will introduce the reader to the different tools and building blocks of a CNN. At the end a precise description of how one can utilize their full potential by combining them, and eventually resulting in something most commonly known as an architecture.

### Neuron

Neurons in CNN's can be seen as a combination of an adaptive combiner (shown in 2.3) and activation function[32]. They are thought of as the building block of the CNN layers. This representation will come in handy later in this section.

### Biology and math

[29] and [33] noted that Convolutional Neural Networks (CNN) is inspired by the animal visual cortex, more specifically, the discovery of the way neurons are connected with a certain pattern in a cat's brain [17] as cited in [27]. The neurons cover a part of the visual field, tiled together by overlapping each other. Neurons respond to stimuli through simple and complex cells, and a convolutional operation can mathematically replicate this [27, p.2] [33, p.4]. Figure 2.4 shows the relationship between the visual system and base convolutional operations.

On the left, we can see that the simple cells have a preferred location in an image that causes activation, and further the complex cells react to activation of the simple cells[27]. On the right, we can see the filtering process that we up to now have called convolutional operation.

### Convolutional layer

The core building blocks of a CNN is the convolutional layers. This is where the feature extraction is happening. A filter also known as a kernel, with a specific size is applied to the input, and the output is considered to be the feature represented as weights[42]. Lets define a feature we wanna

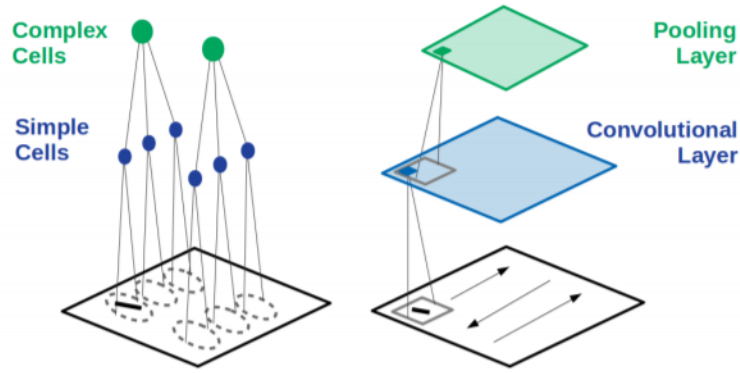


Figure 2.4: Visual vs convolutional hierarchy[27]

search for in an input, represented as a 3x3 matrix. This is applied throughout the input with a set of predefined rules for movement. It slides a window over the inputs to find where the feature matches and map this into another matrix. This is called the feature map. Figure 2.5 visualizes both input and filter. Figure 2.6 shows how its mapped onto a feature map.

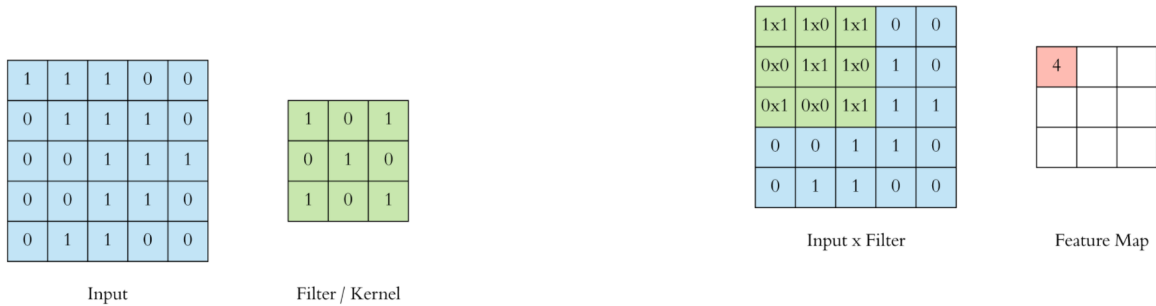


Figure 2.5: Input and Filter [9]

Figure 2.6: Input x filter → Feature Map[9]

This is done for as many filters that are applied in the network.

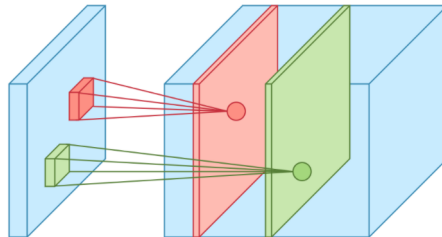


Figure 2.7: Colours represent different filters [9]

### 2.1.9 Padding and Stride

How much the window slides is called stride. In figure 2.8 a stride of 2 is performed, with a pooling window of 2x2 .

#### Pooling

[9] and [20] noted that pooling is performed by down-sampling each feature map. By reducing the height and width, keeping the depth intact, training time is shortened. *Maxpooling* is the most common type of pooling. This is yet another convolutional operation that slides a window over the inputs and maps the max value into another matrix.

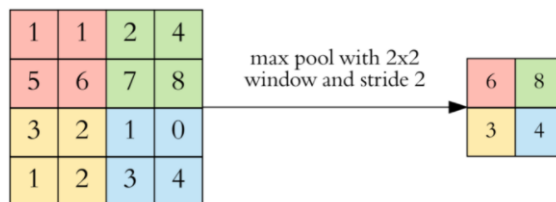


Figure 2.8: *Maxpooling* [9]

#### Batch Normalization

[23] as cited in [31, p.111-113] explains how Batch Normalization helps the network train faster and achieve higher accuracy. This is done to every layer in a neural network. A vector is grabbed before going through a layer, then each component of the vectors in the batch is normalized by subtracting the mean and dividing by the standard deviation.

#### Activation Function

The purpose of applying the activation function is to increase the non-linearity in images[38], [20]. In image classification, one of the most effective functions is the rectified linear units (ReLU).  $f_{Relu}(h_{i,k}) = \max(0, h_{i,k})$ . This is a function that sets all negative values as zero, and positive values equals as the input. [20] explains that one of the keys to recent success in NN is the use of ReLU and that it's generally a better solution than the conventional Sigmoid-like units. This is why we have used this activation function, as described in the Methods section later in this report.

## Dropout

Dropout is one of the most common method for preventing overfitting in DNN's [9, section 4.1] [31, p45]. As described in [9, section 4.1] [31, p45-46], dropout works by "dropping" or disabling neurons. [9] noted that dropout is analogous to a company with an expert in finance being away from work for a period of time each week. This forces the coworkers that were dependent on this expert to learn something new and work with people they usually wouldn't. This makes the company more dynamic and raises the overall competency level of the employees. The parameter for tuning the amount of dropout is called dropout-rate. If the dropout rate  $p = 0.2$ , 20 percent of the neurons are being disabled. If a neuron is disabled, its input and output are inactive for the rest of the training step but can be active the next.

## Loss function and gradient decent

The loss function is used for gradient descent. In simple terms we need to know the error when predicting an image - how far away from the desired output are we?. If  $y$  is the desired output and  $x_i$  is the input from the previously connected neuron, what is the rate of change in  $y$  with respect to  $x_i$ . We can check this by doing the equivalent to gradient descent by taking the derivative of the output with respect to the input. In simple words, the loss is used to calculate the gradients. And gradients are used to update the weights of the network. This is how a CNN is trained. In the experiments done in this thesis, the loss is calculated using the "sparse categorical cross entropy" function from Tensorflow.

$$J(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2.1)$$

Where  $w$  are the model parameters - weights of the neural network.  $y_i$  is the true label, and  $\hat{y}_i$  is the predicted label.

### 2.1.10 Residual Networks

When searching for "Image classification" on IEEE, the most cited publication, with over 50,000 citations, and used in over 180 patents, is the paper; "Deep Residual Learning for Image Recognition" [19]. Clearly our solution space contains this paper and technology related to it. In 2016 [19] proved the ResNet can go deep, 152 layers, 8x deeper than VGG nets but still having lower complexity. With the ResNet model they achieved 3.57 percent error on the ImageNet test set[19]. This result won 1st place on the ILSVRC 2015 classification task[19]. Moving forward the popularity of ResNet's grew. A ResNet is a CNN with residual blocks. In a CNN each layer feeds

into the next, but in a network with residual blocks, each layer feeds into the next layer and directly into layers that are 2 to 3 blocks away. It's skipping. See figure below. Given input  $x$ ,  $F(x)$  is the

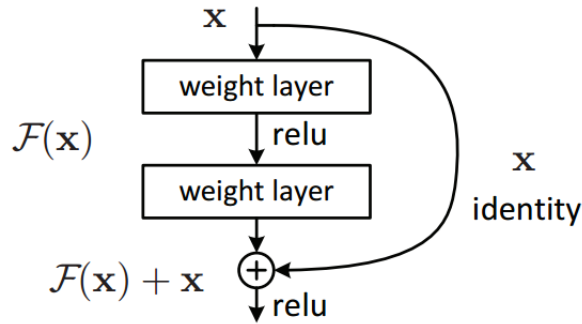


Figure 2.9: Residual block

result of  $x$  after it has been passed through two convolutional layers, with relu between them. The identity of  $x$  is then added to  $F(x)$  given the output as  $F(x) + x$ . This means the next layer learns new information and at the same time retain knowledge learned in previous layers[19].

### 2.1.11 Transfer Learning

The main idea is that one can transfer knowledge from one model to another. If a model has learned a set of features by being trained on a large dataset, the intuition is that it could serve a good model of the visual world. [25] as cited in [2], noted that by utilizing deep and complex CNNs but freezing their layers, thereby decreasing the trainable parameters and allowing for knowledge transfer from training on large image datasets. Specifically one can use a pre-trained model as is, or perform transfer learning by only training the top layers of the pre-trained model, and the newly-added classifier layers. By doing this, we're allowing to "fine-tune" feature representations in the pre-trained model in order to make them more relevant for the specific task [2]. The intuition behind transfer learning for image classification [2] is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

### 2.1.12 Stereo Camera Sensing

One of the most recent published surveys on RGB-D datasets, [28] starts off by classifying different sensors able to obtain depth information into the following categories: Structured Light, Time-of-Flight (TOF), Light Detection and Ranging (LIDAR), and Stereo Camera Sensing (SCS). We will

focus on the latter. To create a depth map, one can use two or more image sensors or lenses [28]. A straightforward strategy to measure depth from two or more cameras is Triangulation. The Triangulation idea is the same as applied in Structured Light sensors, but using a camera instead of a projector. The idea is that finding the position of a pixel in the image plane of camera A projected from a point P in the space, and the position of a pixel projected by the same point P in camera B, it is possible to find the depth of that point in a scene with the intrinsic parameters of the camera. After finding both lines projected in both cameras from point P, it is only necessary to know the distance between the two cameras (baseline distance) and internal parameters of the cameras to know the depth of the point P. A limitation of this strategy occurs when the point of interest has no texture. For instance, it is practically impossible to determine which point of a smooth painted wall observed in the image projected by camera A is equivalent to the image projected by camera B. Therefore, it is difficult to determine a point's depth with acceptable accuracy without the correspondence of the pixels in both image planes. Recently, Deep Learning based methods have tried to address this limitation, increasing the accuracy of the estimation [27]. Examples of such types of sensors include light field cameras and ZED cameras.

## 2.2 Related Work

Can I make the case that something is missing in the current literature? I have different vague ideas about how to solve the problem with a quite elementary overarching algorithm, comprising of one model for determining the type, and then depending of the type chose 1 out of three models determining the amount. The most work using depth information has been focused on semantic segmentation and object recognition.

### 2.2.1 RGB-D

After the emergence of low cost sensors like Microsoft's Kinect and Intel's Realsense, the number of RGB-D datasets has grown, and as a consequence so has the number of methods for classifying the data. Early methods for classifying RGB-D combines CNN and Recurrent Neural Network (RNN) [36] as cited in [15]. The methods is to learn low level translation invariant features with the CNN for both RGB and depth images, but do it separately. Then the learnt features is fed into multiple RNNs to map out more high level features. This method proved to reduce computation time. A similar approach is done by [11], who are training two CNNs on RGB and depth, and then fuse them together creating one fusion stream, and leaving one stream from each

network untouched. By doing this, one can learn the weights of the fusion and the independent features from the input. In [16], Gupta explores the idea of stacking the depth channel onto a CNN architecture, but with modifications. They represent each pixel using horizontal disparity, pixel height above ground, and angle between normals and gravity, known as HHA. These three features can be represented as a 3 channel image and therefore make use of the feature extracting skills of a CNN. [41] assumes that pixels on the same 3D-plane tend to share the same class. They have developed a generic model, Z-ACN can be applied to all applications such as classification, segmentation, and object detection. [14] shows that by applying a 5x5 and 3x3 encoding filters to the depth image and normalizing the original data, one can create a 3-channel input for an CNN, which performs better than just replicating the original depth channel into 3 identical ones. An advantage of this encoding method is that the pre-processing filters can be easily incorporated as part of the CNN. This can be achieved by adding a new convolutional layer to the depth path of the RGB-D CNN. Others have tried to take the volumetric approach by modifying the kernel to instead of having the size of  $w, h, l$  as input, has a  $k, k, d$  and outputs  $w, h, m$ . Each stride traverses the grid similar to a 2D convolution operator and increases depth by a given stride once each 2D plane has been convolved [15].

## 2.2.2 Semantic Segmentation

Depth provides additional geometric information that can benefit an RGB semantic segmentation model [26] as cited in [3]. We define the semantic segmentation task as follows: given an input RGB image  $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ , the objective is to produce an output semantic segmentation map  $\mathbf{S} \in \mathbb{R}^{H \times W \times C}$ , where  $C$  is the number of semantic classes. In other words, for each of the  $\mathbf{H} \times \mathbf{W}$  pixels of an RGB image, the semantic segmentation task produces a probability distribution over  $C$  categories. In an RGB-D context, a depth map  $\mathbf{D} \in \mathbb{R}^{H \times W}$  is available in addition to the RGB input so as to enhance the accuracy of the predicted segmentation map.

## 2.3 Tools

### 2.3.1 Realsense software development kit

Realsense software development kit (Realsense SDK 2.0) from Intel, supports a wide range of programming languages and development platforms[8]. With one of the supported languages we can apply settings on the camera. It's also possible to perform a variety of useful post and pre-processing operations, on both color and depth-image. Examples of settings and processing

are; framerate, resolution, filtering, shutter speed, exposure etc.

### 2.3.2 Realsense D435

The Intel RealSense depth camera D435 [8] is a stereo camera, offering depth in addition to regular RGB. It's wide field of view is perfect for applications such as robotics or augmented and virtual reality, where seeing as much of the scene as possible is vitally important. With a range up to 10m, this small form factor camera can be integrated into any solution with ease, and comes complete with our Intel RealSense SDK 2.0 and cross-platform support.

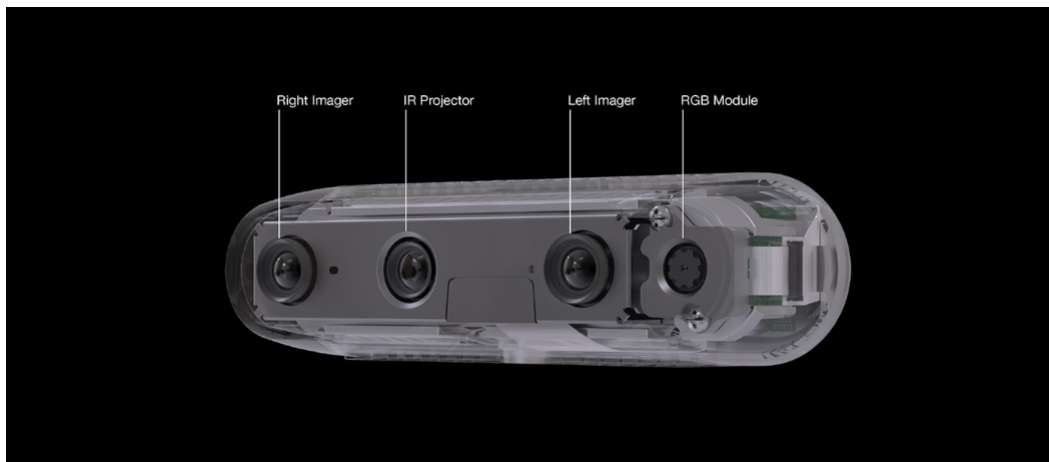


Figure 2.10: Illustration of camera

Here are some specs from [8]

- Range .3 m to 3 m
- Depth Field of View (FOV)  $87^\circ \times 58^\circ (\pm 3^\circ)$
- Depth Accuracy  $<2$  percent at 2 m
- Depth Stream Output Resolution Up to  $1280 \times 720$
- Depth Stream Output Frame Rate Up to 90 fps
- RGB Frame Resolution  $1920 \times 1080$
- RGB Sensor FOV  $69^\circ \times 42^\circ$

Note that the FOV are different for the RGB and depth sensors. This will be useful to remember later in this report.



### **2.3.3 C++ and Python**

Inwatec has explained that they use C++ because of the speed. When having to do computationally demanding task in matter of milliseconds, C++ is outperforming other languages like Python. So when Inwatec allready had developed a software for image capturing, the choice was easy. Python was used for all other tasks in this project. It allows for the use of popular machine learning libraries like Tensorflow and Keras to tune the pre-trained model, scikit to create the SVM and KNN, and other useful tools like Matplotlib, Numpy and OpenCV for data processing and visualisation. All code used in this project can be found in A.1.



# Chapter 3

## Methodology

This chapter presents the chosen methodologies for investigating the research question. A taxonomy was created in order to map out key features in the field of RGB-D as early as possible. We investigated features like number of citations, methods used, dataset used, Due to the limited size of the dataset, a set of good strategies for optimisation had to be considered. In situations similar to this, where the dataset is not large enough to feed the network and achieve satisfying results, three methods are commonly proposed.

- Transfer learning [2],[39],[44], [12].
- Preprocessing [40], [35].
- Select a less complex architecture, with less trainable parameters [2].

All these methods will be considered and the next subsections describe them.

### 3.1 Transfer learning

To create a baseline model with transfer learning is a strategy that assumes that the pre-trained model gives a good indication of what accuracy a state of the art model can achieve on the custom RGB-D dataset. MobilNetV2's [https://tfhub.dev/google/tf2-preview/mobilenet\\_v2/feature\\_vector/4](https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4) feature extractor was loaded with keras to create a model without the dense layer, enabling us to add and train the top layers. The model needs 3 channels (RGB), 224x224 pixels as input to work. We train on RGB and depth separately. When training on depth data we need to create 3 instances of the depth image, one for each channel.

## 3.2 Preprocessing

How can we make the pretrained model improve accuracy by introducing depth data? We will test of a number of preprocessing methods (PPM), marked as **bold** text. **PPM1**: swap one of the color channels with the depth channel, with the following combinations: RGD,RDB,DGB,DDD. **PPM2**: select Depth-ROI based on measurements on maximum and minimum height of all object in the dataset. Set a threshold for accepted values, and change all values outside to a value within Depth-ROI. All three figures shows the same three depth images. Top has one mop, middle has two mops, and lowest has three mops. Figure 3.2 and 3.3 uses different preprocessing methods.



Figure 3.1: Normalized depth

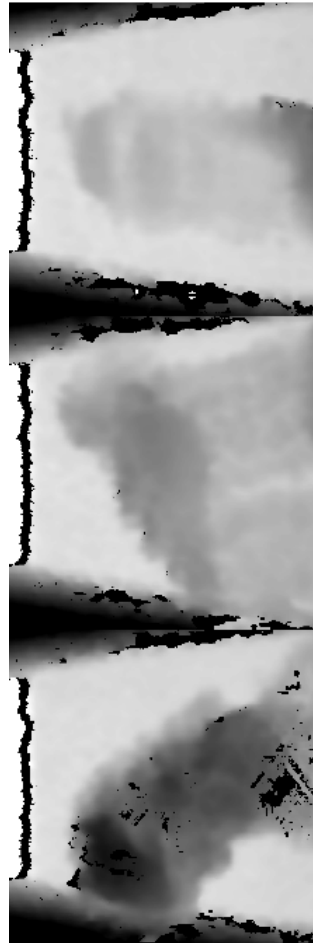


Figure 3.2: Normalized depth with ROI

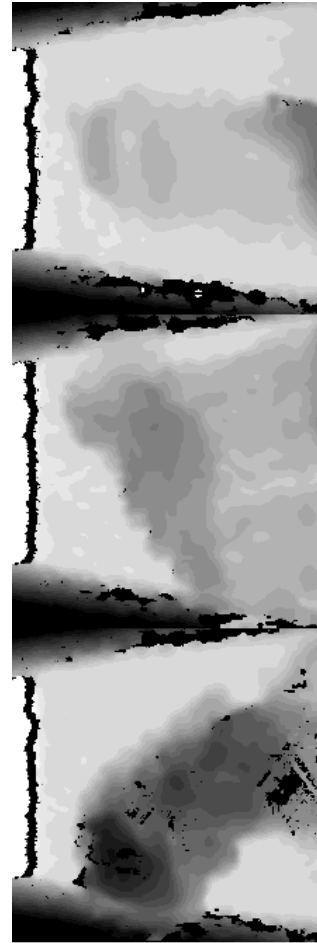


Figure 3.3: Double Normalized depth with ROI

### 3.3 SVM and kNN

The use of a less complex model could be beneficial considering our small dataset, and there is hope. Especially considering the use of SVM and KNN for classification of depth image. By calculating the sum of all depth values in an depth image, the hypothesis is that there will be a significant difference in distribution depending on how many mops that are located inside the FOI of the camera. Either the models finds this feature themselves, or we compute the sum of points, add it as a feature and let the models do the rest. Both methods will be tried. The figure below shows the average value of a pixel per class.

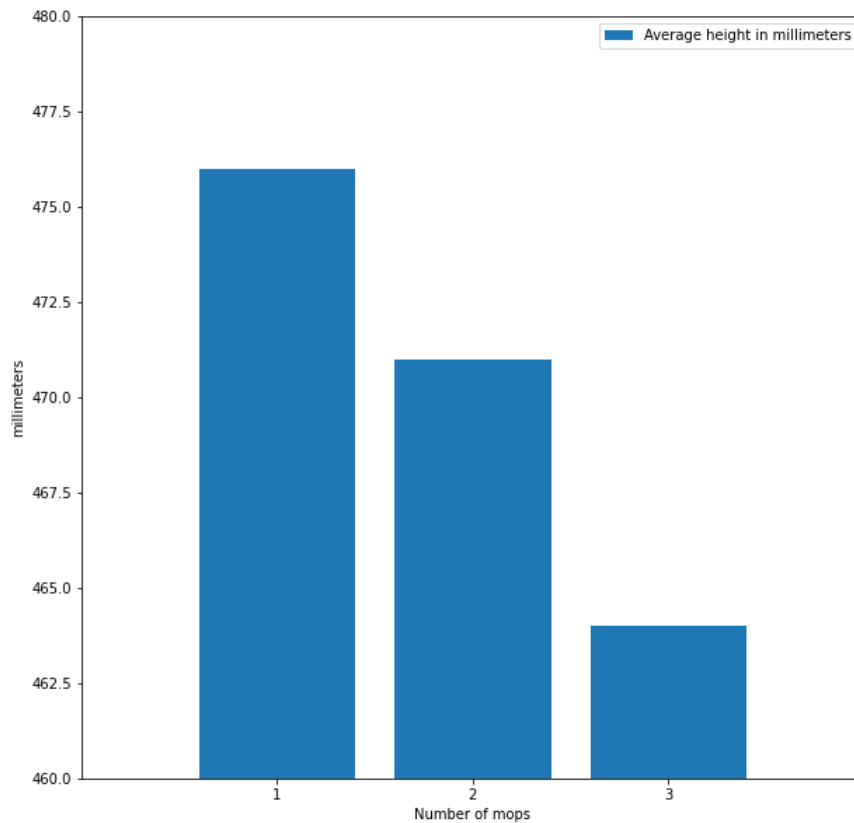


Figure 3.4: White mops avg

### 3.4 Experiments

All experiments will follow a structure based upon a test procedure, designed by combining all of the previously proposed methods Visualizing the data after applying a method is useful for gaining a deeper understanding of the results.

### **3.5 Data collection**

After each test, all metrics data are stored in an Excel sheet in A.3.

### **3.6 Data analysis**

First we need to inspect the dataset. Different tools are used to visualize all RGB and depth images in order to gain insight into any key takeaways from each class. Smaller pre-tests with visualisation of each step of preprocessing must be done in order to find mistakes before running through thousands of images. When all test is finished, most common key metrics can be analysed with scatter plot, bar plot, confusion matrix, and basic sorting functions in Excel.

# Chapter 4

## Implementation

This is where I describe what I actually did in my research, like field studies, experiments, implementations, media productions, interviews, etc.

### 4.1 Hardware

As mentioned in 1.2 hardware needed to be installed in order to begin image capturing. The picture below shows the machine, where the blue marked box is a custom made steel bracket for attaching the camera. The conveyor belt transporting the mops is tilted at an angle, meaning that the bracket needed to match the same angle in order to get useful depth data. After attaching the camera to the bracket, the camera is connected to the AI-pc, which is connected directly to the PLC running the machine.

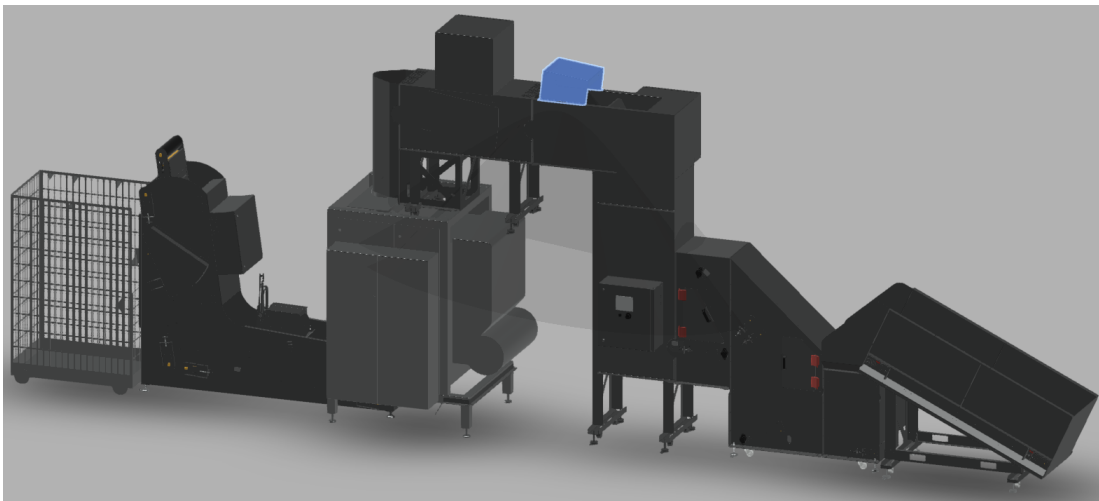


Figure 4.1: Caption

## 4.2 Image capturing

After installation of necessary hardware, learning how to use the software and camera for image capturing, several attempts were made to start building a RGB-D dataset. This part of the project was most time consuming. Little to no prior knowledge about RGB-D data, how to use the equipment, what data to capture, and making sure the quality was good made this a long process split into two major parts.

### 4.2.1 Auto-labeling

Initially an attempt to perform auto-labeling was done. The PLC receives a signal when mops is within the field of view of the camera, this signal is then registered by the image capturing software running on the AI-PC, and triggers the software to capture data. Since its a constant stream of data, one has the option to manually calibrate which frame to save. Either exactly at the time the sensor was triggered or x amount of frames before. When the HF antenna then reads the chips within the reading range, the PLC recvies the tag numbers of the present mops. The idea is that one can know exactly how many mops that where within the camera's field of view at the time of capturing - given that the calibration is done correctly, and the HF-reader gives the correct data. After several tests it proved to be a solution that was to imprecise, and manual labeling was the better option. At the end of this process we learned that the data being captured was not of such a quality that one could make use of both RGB and depth images together. The field of view of the depth image is greater than the RGB image. If we want to be able to use the proposed methods, both the depth and RBG images needs to be aligned, so that the field of view after image capturing is the same for both images. Lets say you wanna merge RGB and depth features for a region in a 224x224 pixel image. Depending on what image one is observing, the same region in the images does not represent the same regions in the real world. This makes the problem much more difficult and was avoided.

### 4.2.2 Manual labeling

By bypassing the first separation stage of the machine, one can feed the mops manually, providing full control of how many mops that are captured by the camera. Watching live what the image that was captured look like, is a extra step for making sure the quality of the dataset is as good as possible. Image 4.2,4.3 ,and 4.4 illustrates where the machine is fed manually, how it looks after they're placed on the conveyor belt, and finally a screenshot of the image taken when the mops



reaches the cameras field of view.



Figure 4.2: Marked door



Figure 4.3: Manual feeding

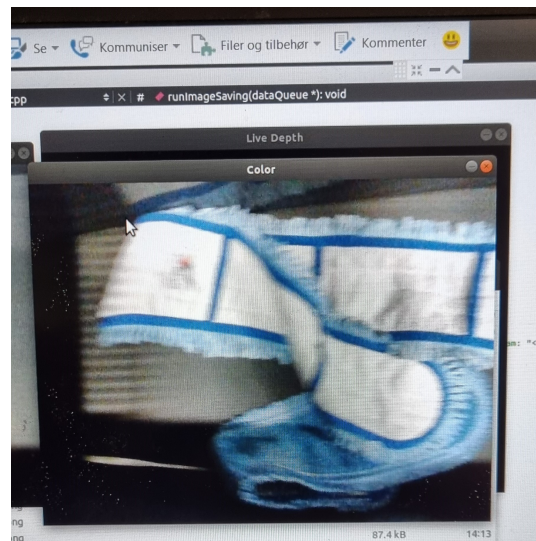


Figure 4.4: Final RGB

Started with configuring the image capture software to store the following images in a specified folder on the AI-pc, then started the software, and fed the machine with the amount and type corresponding to the configuration. When this step is done, the same procedure is repeated until all types (Blue, Green, White), and amount of each type(1,2,3) was stored inside 9 different folders. It took about 30 hours to capture around 2300 images, feeding the machine 4500 mops. Adding a

new class of 4 mops to each type could enable us to simulate a classification scenario that's similar to live classification, but it would require feeding the machine with 3000 new mops, and there was simply no more time.

### **4.3 Model development**

Since Inwatec provided a good framework for capturing images, the main focus was on development of software for testing the different machine learning methods. Originally the plan was to run all testing on the AI-PC. But after a couple of weeks with struggling to get all libraries to work, versions to match, and other challenges related to the latter, the decision was made to switch to Google Colab. Benefits like access from everywhere, scalable data resources, all libraries work imiteatly, no need to install GPU software on local machine - the list goes on.

# Chapter 5

## Results

In this chapter the results from all test performed will be presented. The use of tables and confusion matrices will be used to present the data.

### 5.1 MobilenetV2

#### 5.1.1 Baseline

We tested on both RGB and depth images separately. Figure 5.1 and 5.2 display the results in form of two confusion matrices, one for each data type. We attempt to classify all 9 classes. There's are a total of 2347 images, test size is set to 20 percent with the train test split from [34]. Random state is set to 42, so the test is reproducible. The baseline tests produce the best results after 30 epoch, after testing 10,20, and 30 epochs.

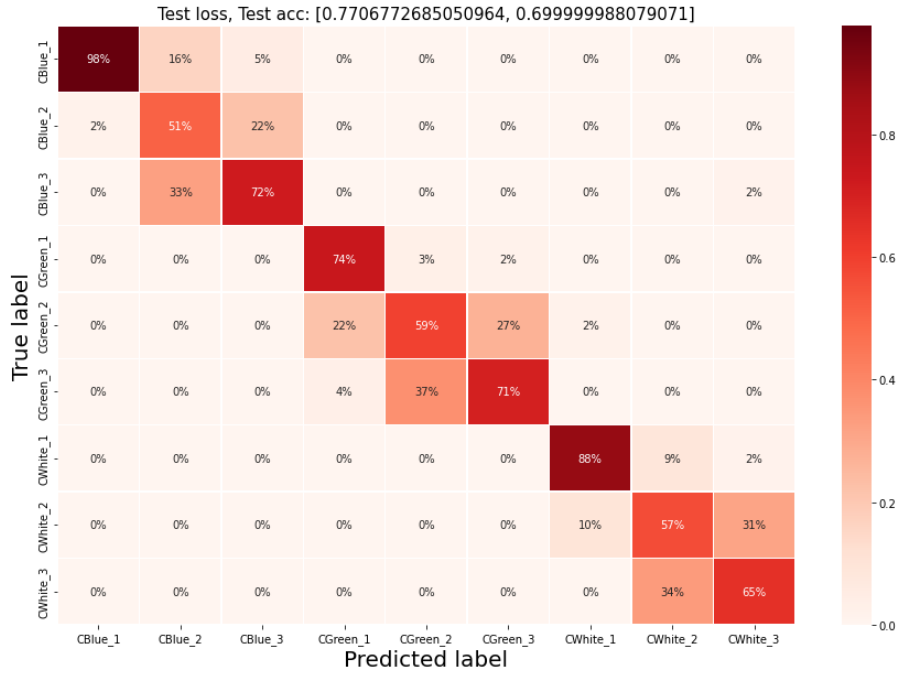


Figure 5.1: Baseline RGB

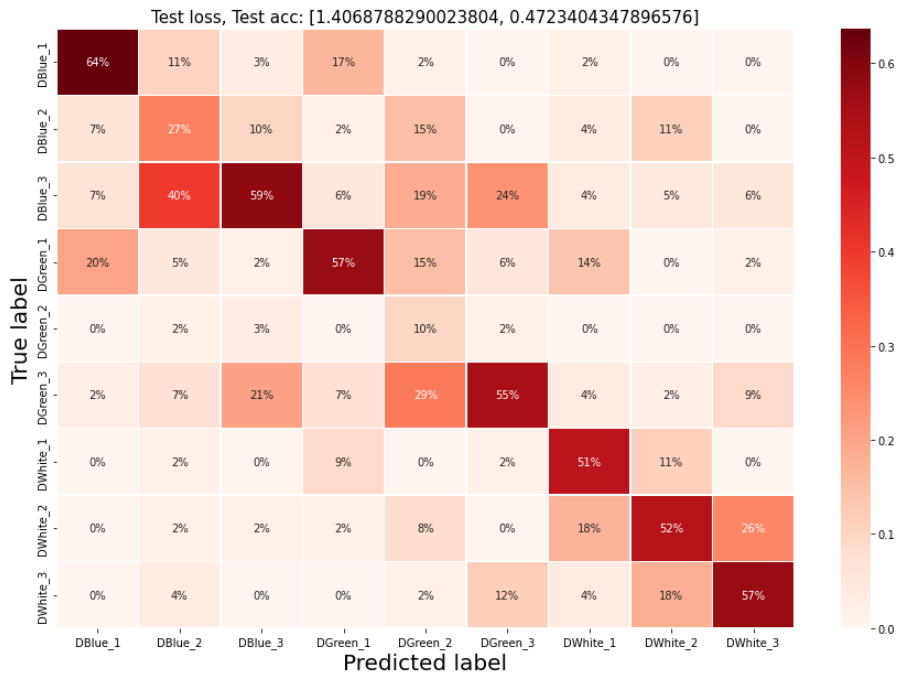


Figure 5.2: Baseline Depth

### 5.1.2 Number of MNV2

After creating a baseline model, an attempt to split the problem into two domains, type and number classification. We tried classifying only the type of mop (color), total of 3 classes, Blue, green and white. Test size is 20 percent with random state set to 42. After 7 epochs the model classified with 100 percent accuracy and a loss of 0.007. So the focus from here it to classifying only the number of mops, total of 3 classes, 1,2 and 3 mops. Test size is 20 percent with random state set to 42. Table 5.1 shows the results from only classifying number of white mops with **PPM1**: swap one of the color channels with the depth channel, with the following combinations: RGD,RDB,DGB,DDD. The same procedure was done for both green 5.2 and blue 5.3 mops.

Table 5.1: Top results White Mobilnet with **PPM1**:

Type of test	Acc	Model	Pretrained	dtype	Epochs	Gaus filter	DEPTH ROI
White_RGB	80.3	Mobilnet	True	float	30	False	False
White_RGD	79.6	Mobilnet	True	float	10	False	False
White_RGD	79.6	Mobilnet	True	float	30	False	False
White_DGB	78.4	Mobilnet	True	float	30	False	False
White_RGB	77.7	Mobilnet	True	float	10	False	False
White_DDD	74.5	Mobilnet	True	float	30	False	False
White_RDB	73.9	Mobilnet	True	float	10	False	False
White_RGG	73.3	Mobilnet	True	float	10	False	False
White_RDB	70.1	Mobilnet	True	float	30	False	False
White_DDD	69.4	Mobilnet	True	float	10	False	False

Table 5.2: Top results Green Mobilnet with **PPM1**:

Type of test	Acc	Model	Pretrained	dtype	Epochs	Gaus filter	DEPTH ROI
Green_RGB	71.8	Mobilnet	True	float	10	False	False
Green_RGB	71.8	Mobilnet	True	float	30	False	False
Green_DDD	70.5	Mobilnet	True	float	30	False	False
Green_RDB	69.2	Mobilnet	True	float	10	False	False
Green_RDB	68.6	Mobilnet	True	float	30	False	False
Green_RGD	66.1	Mobilnet	True	float	10	False	False
Green_DGB	63.5	Mobilnet	True	float	10	False	False
Green_DDD	63.5	Mobilnet	True	float	10	False	False

Table 5.3: Top results Blue Mobilnet with **PPM1**:

Type of test	Acc	Model	Pretrained	dtype	Epochs	Gaus filter	DEPTH ROI
Blue_RGD	75.6	Mobilnet	True	float	10	False	False
Blue_RGB	73.9	Mobilnet	True	float	30	False	False
Blue_RGB	70.7	Mobilnet	True	float	10	False	False
Blue_RDB	70	Mobilnet	True	float	10	False	False
Blue_DGB	69.4	Mobilnet	True	float	10	False	False
Blue_DDD	64.9	Mobilnet	True	float	10	False	False

## 5.2 SVM KNN

In this section the results from using SVM and KNN with **PPM2**: is presented. For the SVM a polynomial kernel is used. For the KNN, a One Vs rest classifier is used and tested from 0-20 k.

### 5.2.1 Number of SVMKNN

Table 5.4: Top results from SVM and KNN with **PPM2**:

<i>Type of test</i>	<i>Acc</i>	<i>Model</i>	<i>dtype</i>	<i>DEPTH ROI</i>	<i>ROI VALUE</i>	<i>NORM Val</i>	<i>k-value</i>
<b>D_White</b>	<b>90.4%</b>	<b>SVM</b>	<b>int</b>	<b>True</b>	<b>400-550</b>	<b>0-10</b>	
D_White	86.6%	KNN	int	True	400-550	0-20	2
<b>D_Green</b>	<b>87.8%</b>	<b>SVM</b>	<b>int</b>	<b>True</b>	<b>400-550</b>	<b>0-10</b>	
D_Green	80.1%	KNN	int	True	400-550	0-10	4
<b>D_Blue</b>	<b>85.3%</b>	<b>SVM</b>	<b>int</b>	<b>False</b>	<b>400-550</b>	<b>0-5</b>	
D_Blue	72.0%	KNN	int	True	400-550	0-20	16

Following is the confusion matrix and metrics report from the best results from each color type, marked with bold in table 5.4. Numbers from 0-2 indicate number of mops-1. Meaning 0 is actually the class with 1 mop.

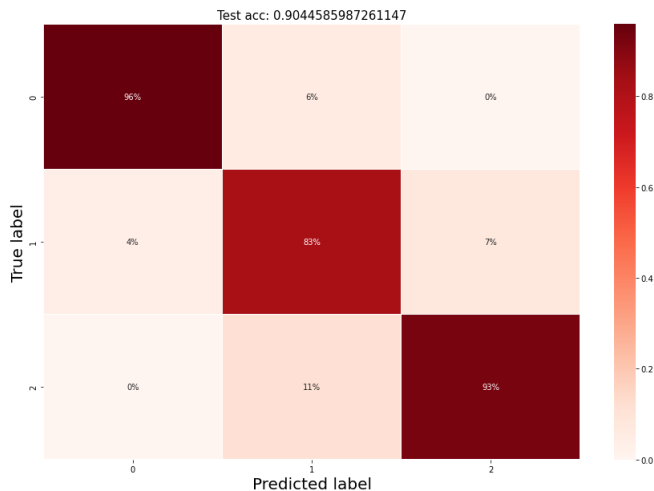


Figure 5.3: Best white test

	precision	recall	f1-score	support
0	0.94	0.96	0.95	49
1	0.88	0.83	0.85	53
2	0.89	0.93	0.91	55
accuracy			0.90	157
macro avg	0.90	0.91	0.90	157
weighted avg	0.90	0.90	0.90	157

Figure 5.4: Best white test metrics

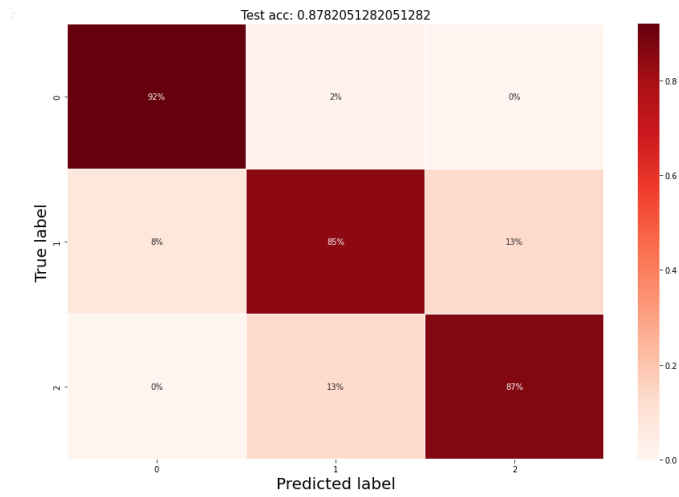


Figure 5.5: Best green test

	precision	recall	f1-score	support
0	0.98	0.92	0.95	50
1	0.80	0.85	0.83	53
2	0.87	0.87	0.87	53
accuracy			0.88	156
macro avg	0.88	0.88	0.88	156
weighted avg	0.88	0.88	0.88	156

Figure 5.6: Best green test metrics

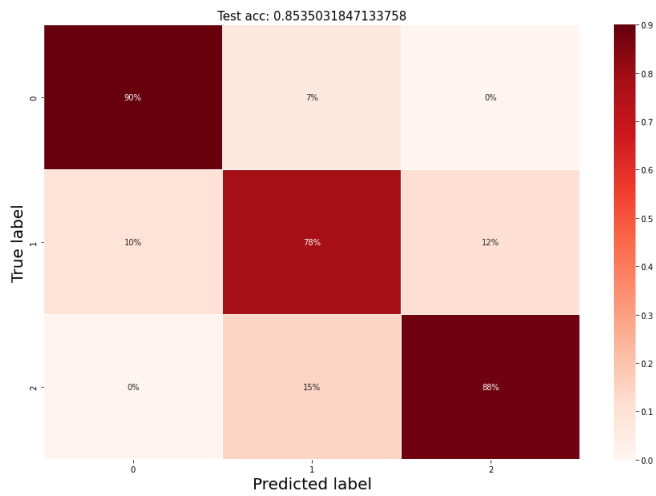


Figure 5.7: Best blue test

	precision	recall	f1-score	support
0	0.92	0.90	0.91	50
1	0.80	0.78	0.79	55
2	0.85	0.88	0.87	52
accuracy			0.85	157
macro avg	0.86	0.86	0.86	157
weighted avg	0.85	0.85	0.85	157

Figure 5.8: Best blue test metrics



# Chapter 6

## Discussion

First of all, it became evident quite early on in the testing process that the pretrained model would perform excellent when classifying only the colour type, and that's when we decided to make it a 2 step classification problem. If one can predict the colour 100% of the time for all color images, then we had to focus on determining the best method for classifying the number. Intuitively an engineering approach to it seemed most promising. By using the depth images could possibly yield the best results if one could capture the volumetric feature in each depth image. We compare the average of the top results from the pretrained model and the top results from the SVM & KNN test in figure 6 SVM shows a significant increase in accuracy compared to both KNN and

<i>Average Acc</i>	<i>Model</i>
<b>87.9%</b>	SVM
79.6%	KNN
75.9%	MobilNetV2

Table 6.1: Average results

MobilNetV2. In this scenario where we used MobilenNetV2 as a baseline model, we managed to answer both the research questions. Lets revisit them. **Research question one:** Can classification accuracy of RGB-D data be improved by breaking the problem into two parts? It looks like it, we have shown a significant boost of accuracy, compared to solving the problem as one task.

**Research question two:** Can classical machine learning methods help to improve classification accuracy of depth images? The results from both KNN and SVM shows a better result than with the pretrained model. In the literature it seems like the volumetric approaches perform better when trying to classify RGB-D data, and so do our methods.



## Chapter 7

# Conclusions

This report has shown the implementation of how to capture image data in a industrial environment, then build a dataset with a total of 9 classes. The quality of the images varies, and the size of the dataset is not more than about 260 images per class. So when reading the results it should not be interpreted as evidence with solid confidence, but more as a proof of concept. We answered both of the research question and visualized the results in the form of the most common way in the literature; Confusion matrix and tables with precision, recall, f1-score support, accuracy, macro average and weighted average. For further work it would be interesting to try and build a larger dataset with higher quality of images. Testing semantic segmentation for RGB-D data, depth aware CNN and other volumetric methods could help improve the accuracy.



# Bibliography

- [1] Shrestha. Ajay and Mahmood. Ausif. 'Review of Deep Learning Algorithms and Architectures'. In: *IEEE Access* 7 (2019), pp. 53040–53065. DOI: 10.1109/ACCESS.2019.2912200.
- [2] Ioannis D. Apostolopoulos and Mpesiana Tzani. 'Industrial object, machine part and defect recognition towards fully automated industrial monitoring employing deep learning. The case of multilevel VGG19'. In: *CoRR abs/2011.11305* (2020). arXiv: 2011.11305. URL: <https://arxiv.org/abs/2011.11305>.
- [3] Sami Barchid, José Mennesson and Chaabane Djéraba. 'Review on Indoor RGB-D Semantic Segmentation with Deep Convolutional Neural Networks'. In: *2021 International Conference on Content-Based Multimedia Indexing (CBMI)*. 2021, pp. 1–4. DOI: 10.1109/CBMI50038.2021.9461875.
- [4] Mariusz Bojarski et al. 'End to End Learning for Self-Driving Cars'. In: *CoRR abs/1604.07316* (2016). arXiv: 1604.07316. URL: <http://arxiv.org/abs/1604.07316>.
- [5] Mariusz Bojarski et al. 'Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car'. In: (2017). DOI: "1704.07911".
- [6] Mariusz Bojarski et al. 'VisualBackProp: efficient visualization of CNNs'. In: (2017). arXiv: 1611.05418 [cs.CV].
- [7] Jair Cervantes et al. 'A comprehensive survey on support vector machine classification: Applications, challenges and trends'. In: *Neurocomputing* 408 (2020), pp. 189–215.
- [8] Intel Corporation. *Build it your way*. <https://www.intelrealsense.com/sdk-2>. 2022 (Online; accessed 29-04-2022).
- [9] Arden Dertat. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>. 2021 (Online; accessed 14-05-2021).

- [10] Ask Any Difference. *NLLLOSS*. <https://askanydifference.com/difference-between-deep-learning-and-neural-network/>. 2022 (Online; accessed 20-03-2022).
- [11] Andreas Eitel et al. 'Multimodal Deep Learning for Robust RGB-D Object Recognition'. In: *CoRR* abs/1507.06821 (2015). arXiv: 1507.06821. URL: <http://arxiv.org/abs/1507.06821>.
- [12] Lihao Ge et al. 'Robust 3D Hand Pose Estimation in Single Depth Images: From Single-View CNN to Multi-View CNNs'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 3593–3601.
- [13] Leilani H. Gilpin et al. 'Explaining Explanations: An Overview of Interpretability of Machine Learning'. In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. 2018, pp. 80–89. DOI: 10.1109/DSAA.2018.00018.
- [14] Radhakrishnan Gopalapillai et al. 'Convolution-Based Encoding of Depth Images for Transfer Learning in RGB-D Scene Classification'. In: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21237950. URL: <https://www.mdpi.com/1424-8220/21/23/7950>.
- [15] David Griffiths and Jan Boehm. 'A review on deep learning techniques for 3D sensed data classification'. In: *CoRR* abs/1907.04444 (2019). arXiv: 1907.04444. URL: <http://arxiv.org/abs/1907.04444>.
- [16] Saurabh Gupta et al. 'Learning Rich Features from RGB-D Images for Object Detection and Segmentation'. In: *CoRR* abs/1407.5736 (2014). arXiv: 1407.5736. URL: <http://arxiv.org/abs/1407.5736>.
- [17] Hubel. D. H. and Wiesel. T. N. 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*'. In: vol. 160. 1. 1962 (Online; accessed 09-05-2021), pp. 106–154.
- [18] Larry Hardesty. *Explained: Neural networks*. <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>. 2017 (Online; accessed 07-05-2021).
- [19] Kaiming He et al. 'Deep Residual Learning for Image Recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.
- [20] Ide. Hidenori and Kurita. Takio. 'Improvement of learning for CNN with ReLU activation by sparse regularization'. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 2684–2691. DOI: 10.1109/IJCNN.2017.7966185.
- [21] IBM. *Explainable AI*. <https://www.ibm.com/watson/explainable-ai>. 2021 (Online; accessed 24-11-2021).

- [22] SAS Institute INC. *Machine Learning - What it is and why it matters*. [https://www.sas.com/en\\_us/insights/analytics/machine-learning.html#machine-learning-today-world](https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-today-world). 2021 (Online; accessed 05.11.2021).
- [23] Sergey Ioffe and Christian Szegedy. 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: <http://proceedings.mlr.press/v37/ioffe15.html>.
- [24] Eda Kavlakoglu. *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?* <https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>. 2020 (Online; accessed 09-05-2021).
- [25] Simon Kornblith, Jonathon Shlens and Quoc V. Le. 'Do Better ImageNet Models Transfer Better?' In: *CoRR* abs/1805.08974 (2018). arXiv: 1805.08974. URL: <http://arxiv.org/abs/1805.08974>.
- [26] Seungyong Lee, Seong-Jin Park and Ki-Sang Hong. 'RDFNet: RGB-D Multi-level Residual Feature Fusion for Indoor Semantic Segmentation'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4990–4999. DOI: 10.1109/ICCV.2017.533.
- [27] Grace W. Lindsay. 'Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future'. In: *Journal of Cognitive Neuroscience* (Feb. 2020 (Online; accessed 09-05-2021)), pp. 1–15. ISSN: 0898-929X. DOI: 10.1162/jocn\_a\_01544. eprint: [https://direct.mit.edu/jocn/article-pdf/doi/10.1162/jocn\\_a\\_01544/1888650/jocn\\_a\\_01544.pdf](https://direct.mit.edu/jocn/article-pdf/doi/10.1162/jocn_a_01544/1888650/jocn_a_01544.pdf). URL: [https://doi.org/10.1162/jocn%5C\\_a%5C\\_01544](https://doi.org/10.1162/jocn%5C_a%5C_01544).
- [28] Alexandre Lopes, Roberto Medeiros de Souza and Hélio Pedrini. 'A Survey on RGB-D Datasets'. In: *ArXiv* abs/2201.05761 (2022).
- [29] Matsugu. M et al. 'Subject independent facial expression recognition with robust face detection using a convolutional neural network'. In: *Neural networks : The official journal of the International Neural Network Society*. 2003 (Online; accessed 01-05-2021), pp. 5–6. DOI: 10.1016/S0893-6080(03)00115-1.
- [30] Keisuke Mori et al. 'Visual Explanation by Attention Branch Network for End-to-end Learning-based Self-driving'. In: *2019 IEEE Intelligent Vehicles Symposium (IV)* (2019), pp. 1577–1582.

- [31] Nikhil Buduma Nithin Buduma. *Fundamentals of Deep Learning, 2nd Edition*. O'Reilly Media, Inc, 2021. ISBN: 9781492082163.
- [32] Rabah Nory, Mustafa Aljumaili and Nezar Ismat. 'Fire Detection Using Convolutional Deep Learning Algorithms'. In: *AUS* 26 (Apr. 2019), pp. 441–448. DOI: 10.4206/aus.2019.n26.2.53.
- [33] P.Ongsulee. 'Artificial intelligence machine learning and deep learning'. In: *2017 15th International Conference on ICT and Knowledge Engineering (ICT KE)*. 2017 (Online; accessed 10-05-2021), pp. 1–6. DOI: 10.1109/ICTKE.2017.8259629.
- [34] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [35] Max Schwarz, Hannes Schulz and Sven Behnke. 'RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), pp. 1329–1335.
- [36] Richard Socher et al. 'Convolutional-Recursive Deep Learning for 3D Object Classification'. In: NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 656–664.
- [37] Abhiroop Talasila. *Generating Image Segmentation Masks — The Easy Way*. <https://towardsdatascience.com/generating-image-segmentation-masks-the-easy-way-dd4d3656dbd1>. 2022 (Online; accessed 25-04-2022).
- [38] SuperDataScience Team. *The Ultimate Guide to Convolutional Neural Networks (CNN)*. <https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn>. 2018 (Online; accessed 14-05-2021).
- [39] Eleftherios Trivizakis et al. 'Extending 2-D Convolutional Neural Networks to 3-D for Advancing Deep Learning Cancer Classification With Application to MRI Liver Tumor Differentiation'. In: *IEEE Journal of Biomedical and Health Informatics* 23 (2019), pp. 923–930.
- [40] Sebastien C. Wong et al. 'Understanding data augmentation for classification: when to warp?' In: *CoRR* abs/1609.08764 (2016). arXiv: 1609.08764. URL: <http://arxiv.org/abs/1609.08764>.
- [41] Zongwei Wu et al. 'Depth-Adapted CNN for RGB-D cameras'. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Nov. 2020 (Online; accessed 20-04-2022).
- [42] Zharfan Zahisham, Chin Poo Lee and Kian Ming Lim. 'Food Recognition with ResNet-50'. In: *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAJET)*. 2020, pp. 1–5. DOI: 10.1109/IICAJET49801.2020.9257825.



- [43] Wenyuan Zeng et al. 'End-To-End Interpretable Neural Motion Planner'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)*, pp. 8652–8661.
- [44] Qilin Zhang et al. 'Can Visual Recognition Benefit from Auxiliary Information in Training?' In: *ACCV*. 2014.



# Appendix A

## Listings

### A.1 Code

#### A.1.1 Image capture

##### Data sharing source file

```
#include "Data_Sharing.h"

//initialise atomic variables
std::atomic_int g_sensorID(0);
std::atomic_int g_RFID_ID(0);
std::atomic_int g_numberOfChips(0);
std::atomic_int g_currentProgram(0);
std::atomic_bool g_sensorTriggered(false);

ImageData_struct dataQueue::getData(int id)
{
    //Get lock to block other thread from accessing
    boost::mutex::scoped_lock lock(mux_data);

    //data to be returned
    ImageData_struct returndata;
    returndata.pictureID = -1; //-1 means ID not found

    //look through queue
    for (int i=0; i<data.size();i++)
        if (data[i].pictureID==id)
            //if the id matches the one we are looking for
```

```

    {
        returndata = data[i]; //copy data
        break; //exit loop
    }

    int lastPoppedID=-1;
    while (data.size()>0 && lastPoppedID!=returndata.pictureID)
    //pop stuff from queue until we pop the element
    //we were looking for; if element not found,
    //picutreID=lastPopppedID so we do not eneter loop
    {
        lastPoppedID = data[0].pictureID;
        //copy the last removed id
        data.pop_front();
        //remove element from queue
    }

    //return element
    return (returndata);
}

void dataQueue::addData(ImageData_struct dataIn)
{
    //Get lock to block other thread from accessing
    boost::mutex::scoped_lock lock(mux_data);
    //add data to end of queue
    data.push_back(dataIn);
}

```

## Data sharing header

```

#define DATA_SHARING_H

#include <atomic>
#include <deque>

///#include <librealsense2/rs.hpp> // Include RealSense if we add
//realsense objects to the ImageData struct
///#include <opencv2/opencv.hpp> // Include OpenCV if we add OpenCV
//objects to the ImageData struct

```

```

#include <boost/thread/mutex.hpp>

//shared variables
extern std::atomic_int g_sensorID,g_RFID_ID,g_numberOfChips, g_currentProgram;
extern std::atomic_bool g_sensorTriggered;

//structure that holds data we send bethween threads
struct ImageData_struct
{
    //picture ID
    int pictureID;
    //plus whatever data to be saved
};

//class implementing a data queue bethween threads
class dataQueue{
    private:
        boost::mutex mux_data;
        std::deque<ImageData_struct> data;
    public:
        ImageData_struct getData(int id); //get data with ID, and delete
                                           //elements from the queue up to that ID
        void addData(ImageData_struct dataIn); //add data to end of queu
};

#endif

```

## Image capture source file

```

#include "Image_Capture.h"

#include <iostream>
#include <boost/thread.hpp>

void runImageCapture(dataQueue * output)
{
    int prev_sensorTriggered = -1;
    while (true)
    {
        //Connect to camera, start grabbing pictures

```

```

while (true)
{
    //get frame from camera

    //check for rising edge of signal
    if (prev_sensorTriggered == 0 && g_sensorTriggered == 1)
    {
        std::cout<<"Capture_new_picture_with_ID:_"<<g_sensorID<<std::endl;

        //maybe process frames

        //Object to add to queue
        ImageData_struct newData;

        //put data in object
        newData.pictureID = g_sensorID;

        //add object to queue
        output->addData(newData);
    }
    //save previous value
    prev_sensorTriggered=g_sensorTriggered;

    //Sleep
    boost::this_thread::sleep(boost::posix_time::milliseconds(3));
}
}
}

```

## Image capture header

```

#ifndef PLCCOMS_H
#define PLCCOMS_H

#include "../Data_Sharing/Data_Sharing.h"

class plcCommunication{
private:

    //PLC data types to C++

```

```

//SINT -> char
//USINT -> unsigned char
//INT -> short
//UINT -> unsigned short
//DINT -> int
//UDINT -> unsigned int
//BOOL -> bool
//REAL -> float
//STRING[n] -> char[n+1]
//All strings in
//Automation studio reserve
//an extra character
//for the null character
//The order in the send/receive
//structures must
//match that on the PLC
struct PLCDataIn
{
    unsigned int sensorID=0;
    bool sensorTriggered=false;
    unsigned int RFID_ID=0;
    short numberOfChips=0;
    bool heartbeat=false;
    short currentProgram = 0;
};

struct PLCDataOut
{
    bool heartbeat=false;
};

void readMessage(PLCDataIn pin);

public:
    void RunUdpCom();
    plcCommunication();
};

#endif // PLCCOMS_H

```

Image saving source file

```

#include "Image_Saving.h"

#include <boost/thread.hpp>
#include <iostream>

void runImageSaving(dataQueue * input)
{
    int prevID=-1;

    while (true)
    {
        //wait here for new ID
        while (prevID == g_RFID_ID)
            boost::this_thread::sleep(boost::posix_time::milliseconds(3));

        //save previous ID
        prevID = g_RFID_ID;

        //get image from queue
        ImageData_struct dataToSave = input->getData(g_RFID_ID);

        if (dataToSave.pictureID >=0)
            //If data found in queue, save it
            {
                std::cout<<"Save_picture_with_ID:
                _____" <<g_RFID_ID<<"_Number_of_chips :
                _____" <<g_numberOfChips<<"_Current_program :
                _____" <<g_currentProgram<<std::endl;

                //Save dataToSave based on g_numberOfChips and g_currentProgram
            }

    }
}

```

### Image saving header

```

#ifndef IMAGE_SAVING_H
#define IMAGE_SAVING_H

```



```

#include "../Data_Sharing/Data_Sharing.h"

void runImageSaving(dataQueue * input);

#endif

```

PLC com source file

```

#include "PLC_Communication.h"

#include <boost/asio.hpp>
#include <boost/array.hpp>
#include <boost/thread.hpp>

#include <iostream>

using boost::asio::ip::udp;

plcCommunication::plcCommunication()
{
    //initilize communication
}

void plcCommunication::RunUdpCom()
{
    PLCDataIn pin;
    PLCDataOut pout;

    while (true)
        try
        {
            //Open socket listening on port 9869
            boost::asio::io_service io_service;
            udp::socket socket(io_service, udp::endpoint(udp::v4(), 9869));
            while (true)
            {
                //When something connects
                udp::endpoint remote_endpoint;
                boost::array<char, 512> recv_buf;
                boost::system::error_code error;
                //get data
            }
        }
}

```

```

        int bytes_transferred = socket.receive_from(boost::asio::buffer(recv_buf),
            remote_endpoint, 0, error);
        //copy data in input structure
        if (sizeof(pin) >= bytes_transferred)
            memcpy(&pin, recv_buf.data(), bytes_transferred);
        //exit in case of error
        if (error)
            break;
        //read message in input structure
        readMessage(pin);

        //prepre outoyt message
        pout.heartbeat = pin.heartbeat;

        //send it back
        boost::array<char, sizeof(pout)> send_buf;
        memcpy(send_buf.data(), &pout, sizeof(pout));
        boost::system::error_code ignored_error;
        socket.send_to(boost::asio::buffer(send_buf),
            remote_endpoint, 0, ignored_error);
    }
    socket.cancel();
    socket.close();
}

catch (std::exception& e)
{ //in case of error, show it, sleep 10 ms and try again
    std::cout << e.what() << std::endl;
    boost::this_thread::sleep(boost::posix_time::milliseconds(10));
}
}

void plcCommunication::readMessage(PLCDataIn pin)
{
    //Copy over variables to shared data points
    g_sensorID = pin.sensorID;
    g_numberOfChips = pin.numberOfChips;
    g_RFID_ID = pin.RFID_ID;
    g_currentProgram = pin.currentProgram;
    g_sensorTriggered = pin.sensorTriggered;
}

```

PLC com header

```

#ifndef PLCCOMS_H
#define PLCCOMS_H

#include "../Data_Sharing/Data_Sharing.h"

class plcCommunication{
    private:

        //PLC data types to C++
        //SINT -> char
        //USINT -> unsigned char
        //INT -> short
        //UINT -> unsigned short
        //DINT -> int
        //UDINT -> unsigned int
        //BOOL -> bool
        //REAL -> float
        //STRING[n] -> char[n+1] //All strings in
        //Automation studio reserve an extra character
        //for the null character
        //The order in the send/receive structures must
        //match that on the PLC
        struct PLCDataIn
        {
            unsigned int sensorID=0;
            bool sensorTriggered=false;
            unsigned int RFID_ID=0;
            short numberOfChips=0;
            bool heartbeat=false;
            short currentProgram = 0;
        };

        struct PLCDataOut
        {
            bool heartbeat=false;
        };

        void readMessage(PLCDataIn pin);

public:

```

```

    void RunUdpCom();
    plcCommunication();

};

#endif // PLCCOMS_H

```

## CMakeLists

```

project(ImageCapture)

cmake_minimum_required(VERSION 2.8)

SET(CMAKE_BUILD_TYPE Debug)

# Save the command line compile commands in the build output
set(CMAKE_EXPORT_COMPILE_COMMANDS 1)

include(CheckCXXCompilerFlag)
CHECK_CXX_COMPILER_FLAG("-std=c++14" COMPILER_SUPPORTS_CXX14)
CHECK_CXX_COMPILER_FLAG("-std=c++11" COMPILER_SUPPORTS_CXX11)
CHECK_CXX_COMPILER_FLAG("-std=c++0x" COMPILER_SUPPORTS_CXX0X)
if(COMPILER_SUPPORTS_CXX14)
    set(CMAKE_CXX_STANDARD 14)
elseif(COMPILER_SUPPORTS_CXX11)
    set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -std=c11")
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
elseif(COMPILER_SUPPORTS_CXX0X)
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++0x")
endif()

find_package(OpenCV REQUIRED)

find_package(Boost COMPONENTS system filesystem regex date_time thread chrono REQUIRED)
include_directories(${Boost_INCLUDE_DIRS})

SET(SOURCES main.cpp Data_Sharing/Data_Sharing.cpp
Image_Capture/Image_Capture.cpp
Image_Saving/Image_Saving.cpp
PLC_Communication/PLC_Communication.cpp)
SET(HEADERS Data_Sharing/Data_Sharing.h
Image_Capture/Image_Capture.h

```

```

Image_Saving/Image_Saving.h
PLC_Communication/PLC_Communication.h )

add_executable ( ImageCapture      main.cpp
                ${SOURCES}
                ${HEADERS}
                )

TARGET_LINK_LIBRARIES( ${PROJECT_NAME}
                      ${Boost_LIBRARIES}
                      ${OpenCV_LIBS}
                      realsense2
                      -lboost_thread
                      )

```

### A.1.2 Pretrained MobileNetV2

### A.1.3 SVM

### A.1.4 KNN

## A.2 Project management

After a topic where chosen, a tentative project plan was created. See table A.1 for visualisation of the first version created. Further development of this plan was heavy influenced by a method taught at Electronics Engineering at OsloMet, called "Work Breakdown Structure" (WBS). The goal when using this method is to break down a project into the smallest possible activities. The idea is that a project as a single thing is not solvable, but a single activity is. So by mapping each possible activity related to this type of project, one should be able to deliver a solution. A WBS serves as a great framework when creating project plan. This project plan and agreement contract was signed by all involved parties in late 2021. It ensures that everybody had an idea of how the outline of this project looked and got an idea of the time frame and schedule. Primary supervisor and master-student agreed on weekly meetings in early January. They met with Nor Tekstil and Inwatec in early February to discuss expectations and other relevant topics to ensure that they had the same understanding of what the following month would look like, and what to expect when the thesis is finished. As figure A.1 implies, it was a strategy to do a month of theoretical preparations before starting any practical activities. This is not meant as a way to become an expert in the related fields, but this being a short thesis, time is not a luxury. The idea is that time can be saved by doing some

Table A.1: Tentative Project plan

<b>Practical activities</b>	<b>Week number</b>
Sign agreement	43
Install camera at site in Drammen	6
Capture 2D/3D data at site	7-10
Labeling of dataset	10
Visit Inwatec in Denmark	11
Create models	11-14
Test model on dataset	14-15
Model tuning	16-17
Test model on site	16-17
<b>Theoretical activities</b>	
Map strength and weaknesses	2-3
Finish WBS/detailed project plan	3-4
Risk analysis	3-4
Create structure for thesis	3-4
Litterature review and taxonomy	4-6
Write Introduction and background	6-15
Write remaining part of thesis	16-19
Revision	17

well taught out preparations in order to crystallize the idea of what one should focus on by the time image capturing, and model development is on the agenda.

### A.3 Test results

Type of test	Acc	Model	Pretrained	dtype	Epochs	Gaus filter	DEPTH ROI	ROI VALUE	NOR
DDD_All images	64	Mobilnet	True	int	10	False	False		
DDD_Blue	64	Mobilnet	True	int	10	False	False		
RGD_Blue	63	Mobilnet	True	int	10	False	False		
RDB_Blue	66.8	Mobilnet	True	int	10	False	False		
DGB_Blue	71.9	Mobilnet	True	int	10	False	False		
						False	False		
RGB_All_types	61.3	Mobilnet	False	float	10	False	False		
						False	False		
RGB_White	80.3	Mobilnet	True	float	30	False	False		
RGD_White	79.6	Mobilnet	True	float	10	False	False		
RGD_White	79.6	Mobilnet	True	float	30	False	False		
DGB_White	78.4	Mobilnet	True	float	30	False	False		
RGB_White	77.7	Mobilnet	True	float	10	False	False		
RGD_Blue	75.6	Mobilnet	True	float	10	False	False		
DDD_White	74.5	Mobilnet	True	float	30	False	False		
RGB_Blue	73.9	Mobilnet	True	float	30	False	False		
RDB_White	73.9	Mobilnet	True	float	10	False	False		
RGG_White	73.3	Mobilnet	True	float	10	False	False		
RGD_All_types	72.4	Mobilnet	True	float	30	False	False		
RGB_Green	71.8	Mobilnet	True	float	10	False	False		
RGB_Green	71.8	Mobilnet	True	float	30	False	False		

DGB_All_types	71.7		Mobilnet	True	float	10	False	False	
DGB_All_types	71.1		Mobilnet	True	float	10	False	False	
RGB_Blue	70.7		Mobilnet	True	float	10	False	False	
DDD_Green	70.5		Mobilnet	True	float	30	False	False	
RDB_White	70.1		Mobilnet	True	float	30	False	False	
RDB_Blue	70		Mobilnet	True	float	10	False	False	
RDB_All_types	69.6		Mobilnet	True	float	30	False	False	
DGB_All_types	69.4		Mobilnet	True	float	30	False	False	
DGB_Blue	69.4		Mobilnet	True	float	10	False	False	
DDD_White	69.4		Mobilnet	True	float	10	False	False	
RDB_Green	69.2		Mobilnet	True	float	10	False	False	
DDD_All_types	0.453567677		Mobilnet	True	float	10	False	False	
RGB_All_types	68.7		Mobilnet	True	float	30	False	False	
RDB_Green	68.6		Mobilnet	True	float	30	False	False	
DDD_All_types	0.463567677		Mobilnet	True	float	20	False	False	
RGB_All_types	68.3		Mobilnet	True	float	10	False	False	
DDD_All_types	0.472340435		Mobilnet	True	float	30	False	False	
RDB_All_types	66.8		Mobilnet	True	float	10	False	False	
RGD_Green	66.1		Mobilnet	True	float	10	False	False	
DDD_Blue	64.9		Mobilnet	True	float	10	False	False	
DDD_Green	63.5		Mobilnet	True	float	10	False	False	
DGB_Green	63.5		Mobilnet	True	float	10	False	False	



DDD_Blue	68.2%		Mobilnet	True	int	30	False	True	400-550,500,500	0-3
DDD_Green	67.9%		Mobilnet	True	int	30	False	True	400-550,500,500	0-3
DDD_White	70.7%		Mobilnet	True	int	30	False	True	400-550,500,500	0-3
DDD_White	73.5%		Mobilnet	True	int	10	False	True	400-550,400,550	0-20
DDD_White	77.1%		Mobilnet	True	int	30	False	True	400-550,400,550	0-20
DDD_White	74.5%		Mobilnet	True	int	30	False	True	400-550,400,550	0-255
<b>Type of test</b>	<b>Acc</b>		<b>Model</b>	<b>dtype</b>	<b>DEPTH ROI</b>	<b>ROI VALUE</b>	<b>NORM Val</b>	<b>k-value</b>		
D_White	90.4%		SVM	int	True	400-550,500,500	0-10			
D_White	90.4%		SVM	int	True	400-550,500,500	0-20			
D_White	90.4%		SVM	int	True	400-550,500,500	0-30			
D_White	89.8%		SVM	int	True	400-550,500,500	0-40			
D_White	89.8%		SVM	int	True	400-550,500,500	0-50			
D_White	88.5%		SVM	int	True	400-550,500,500	0-5			
D_White	86.6%		KNN	int	True	400-550,500,500	0-20	2		
D_White	86.6%		KNN	int	True	400-550,500,500	0-30	2		
D_White	86.6%		KNN	int	True	400-550,500,500	0-40	2		
D_White	86.6%		KNN	int	True	400-550,500,500	0-50	2		
D_White	85.4%		KNN	int	True	400-550,500,500	0-10	2		
D_White	83.4%		KNN	int	True	400-550,500,500	0-5	2		
<b>Type of test</b>	<b>Acc</b>		<b>Model</b>	<b>dtype</b>	<b>DEPTH ROI</b>	<b>ROI VALUE</b>	<b>NORM Val</b>	<b>k-value</b>		

D_Green	87.8%	SVM	int	True	400-550	0-10		
D_Green	87.8%	SVM	int	True	400-550	0-20		
D_Green	87.8%	SVM	int	True	400-550	0-50		
D_Green	87.2%	SVM	int	True	400-550	0-40		
D_Green	87.2%	SVM	int	True	400-550	0-255		
D_Green	86.5%	SVM	int	True	400-550	0-30		
D_Green	82.7%	SVM	int	True	400-550	0-5		
D_Green	80.1%	KNN	int	True	400-550	0-10	4	
D_Green	80.1%	KNN	int	True	400-550	0-20	4	
D_Green	80.1%	KNN	int	True	400-550	0-40	4	
D_Green	79.5%	KNN	int	True	400-550	0-30	4	
D_Green	79.5%	KNN	int	True	400-550	0-255	4	
D_Green	78.8%	KNN	int	True	400-550	0-50	4	
D_Green	76.9%	KNN	int	True	400-550	0-5	4	
<b>Type of test</b>	<b>Acc</b>	<b>Model</b>	<b>dtype</b>	<b>DEPTH ROI</b>	<b>ROI VALUE</b>	<b>NORM Val</b>	<b>k-value</b>	
D_Blue	85.4%	SVM	int	True	400-550	0-5		
D_Blue	84.1%	SVM	int	True	400-550	0-20		
D_Blue	83.4%	SVM	int	True	400-550	0-30		
D_Blue	83.4%	SVM	int	True	400-550	0-40		
D_Blue	83.4%	SVM	int	True	400-550	0-50		
D_Blue	83.4%	SVM	int	True	400-550	0-255		

D_Blue	82.2%	SVM	int	True	400-550	0-10		
D_Blue	72.0%	KNN	int	True	400-550	0-20	16	
D_Blue	72.0%	KNN	int	True	400-550	0-40	10	
D_Blue	71.3%	KNN	int	True	400-550	0-30	10	
D_Blue	71.3%	KNN	int	True	400-550	0-50	10	
D_Blue	70.7%	KNN	int	True	400-550	0-5	12	
D_Blue	70.7%	KNN	int	True	400-550	0-10	7	
D_Blue	70.1%	KNN	int	True	400-550	0-255	3	
<b>Model</b>	<b>Type of test</b>	<b>Acc</b>	<b>Loss</b>	<b>Epochs</b>				
MobilNetV2	Baseline_RGB	70.0%	0.771	30				
MobilNetV2	Baseline_Depth	47.2%	1.407	30				
Model	Type of test	Acc	Loss	Epochs				
MobilNetV2	What_number_all	72.6%	0.654	30				
Model	Type of test	Acc	Loss	Epochs				
MobilNetV2	What_type	100.0%	0.007	7				

**A.3.1 MobilnetV2 code**

```
pip install opendatasets
```

[+ Code](#)[+ Text](#)

```
import numpy as np
import cv2
import pathlib
import PIL.Image as Image
import os
import opendatasets as od
import seaborn as sn
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
import tensorflow_hub as hub

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from google.colab.patches import cv2_imshow
from scipy.ndimage.filters import gaussian_filter

physical_devices = tf.config.list_physical_devices('GPU')
print("Num GPUs:", len(physical_devices))

    Num GPUs: 0

from google.colab import drive
drive.mount('/content/gdrive')
data_dir = 'gdrive/MyDrive/Masterprosjekt/Color_tester_all'

os.listdir(data_dir)
data_dir = pathlib.Path(data_dir)
classes = os.listdir(data_dir)
classes.sort()
number_of_cl = len(os.listdir(data_dir))

print(number_of_cl)
print(classes)

#Depending on what folders to use for, comment out the one you dont need
my_images_dict = {

    #'MOP1': list(data_dir.glob(classes[0]+'/*')),
    #'MOP2': list(data_dir.glob(classes[1]+'/*')),
    #'MOP3': list(data_dir.glob(classes[2]+'/*')),
    'MOP4': list(data_dir.glob(classes[3]+'/*')),
    'MOP5': list(data_dir.glob(classes[4]+'/*')),
    'MOP6': list(data_dir.glob(classes[5]+'/*')),
    #'MOP7': list(data_dir.glob(classes[6]+'/*')),
    #'MOP8': list(data_dir.glob(classes[7]+'/*')),
```

```

    ##'MOP9': list(data_dir.glob(classes[8]+'/*')),

}

#This is where you indicate the true label
my_labels_dict = {

    ##'MOP1': 0,
    ##'MOP2': 1,
    ##'MOP3': 2,
    'MOP4': 0,
    'MOP5': 1,
    'MOP6': 2,
    ##'MOP7': 0,
    ##'MOP8': 1,
    ##'MOP9': 2,

}

IMAGE_SHAPE = (224,224) #Set the image size to fit th model
X, y = [], []
#Create arrays for images and labels
#read images with cv2 and run through all folders
#Save images as floats for optimal results
for my_name, images in my_images_dict.items():
    for image in images:
        img = cv2.imread(str(image),cv2.IMREAD_UNCHANGED)
        resized_img = cv2.resize(img,IMAGE_SHAPE)
        X.append(resized_img)
        y.append(my_labels_dict[my_name])

X = np.array(X,dtype='float32')
y = np.array(y,dtype=np.int32)

X.shape

(785, 224, 224, 3)

##Run this if depth data is to be loaded
data_dir = 'gdrive/MyDrive/Masterprosjekt/Depth_tester_all'

os.listdir(data_dir)
data_dir = pathlib.Path(data_dir)
classes = os.listdir(data_dir)
classes.sort()
number_of_cl = len(os.listdir(data_dir))
print(number_of_cl)
print(classes)

#Indetical to color loader above

```

```

my_images_dict = {

    ##'MOP1': list(data_dir.glob(classes[0]+'/*')),
    ##'MOP2': list(data_dir.glob(classes[1]+'/*')),
    ##'MOP3': list(data_dir.glob(classes[2]+'/*')),
    'MOP4': list(data_dir.glob(classes[3]+'/*')),
    'MOP5': list(data_dir.glob(classes[4]+'/*')),
    'MOP6': list(data_dir.glob(classes[5]+'/*')),
    ##'MOP7': list(data_dir.glob(classes[6]+'/*')),
    ##'MOP8': list(data_dir.glob(classes[7]+'/*')),
    ##'MOP9': list(data_dir.glob(classes[8]+'/*')),

}

my_labels_dict = {

    ##'MOP1': 0,
    ##'MOP2': 1,
    ##'MOP3': 2,
    'MOP4': 0,
    'MOP5': 1,
    'MOP6': 2,
    ##'MOP7': 2,
    ##'MOP8': 2,
    ##'MOP9': 2,

}

X_d, y_d = [], []
IMAGE_SHAPE = (224,224)
for my_name, images in my_images_dict.items():
    for image in images:

        img = cv2.imread(str(image), cv2.IMREAD_UNCHANGED) #Read data
        resize_img = cv2.resize(img,IMAGE_SHAPE) #Resize to fit model
        resize_img[np.where(resize_img>550)]=550 #Tuning parameter
        resize_img[np.where(resize_img<400)]=400 #Tuning parameter
        resize_img = cv2.normalize(resize_img,resize_img,0,20,cv2.NORM_MINMAX)
        #tuning paramter
        resize_img = cv2.normalize(resize_img,resize_img,0,255,cv2.NORM_MINMAX)
        #This always needs to be done in order to have RGB values between 0-255

        img = cv2.cvtColor(img,cv2.COLOR_GRAY2RGB)#Make 3 channels
        img = cv2.resize(img[:, :, :],IMAGE_SHAPE) #resize to fit model
        img[:, :, 2]=resize_img #Copy depth image to r channel
        img[:, :, 1]=resize_img #Copy depth image to g channel
        img[:, :, 0]=resize_img #Copy depth image to b channel
        X_d.append(img)

X_d = np.array(X_d,dtype='int16')

```

```

#Inspect image folder
print(X.shape)
print(X_d.shape)
print(np.max(X_d))
print(np.unique(X_d))

#compare depth and color images
cv2_imshow(X_d[600])
cv2_imshow(X[600])

#This is only used if one wanna merge depth and color channels
#X[:, :, :, 0]=X_d[:, :, :, 0] b
#X[:, :, :, 1]=X_d[:, :, :, 1] g
#X[:, :, :, 2]=X_d[:, :, :, 2] r

#Remember tuning parameters test_size, number of classes and nr of epochs

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=None)

X_train_scaled = X_train / 255 #Scaling values to tensor size
X_test_scaled = X_test / 255 #Scaling values to tensor size

feature_extractor_model = "https://tfhub.dev/google/tf2-preview/mobilenet\_v2/feature\_vector
pretrained_model_without_top_layer = hub.KerasLayer(
    feature_extractor_model, input_shape=(224,224,3), trainable=False)

my_num_of_classes = 3

model = tf.keras.Sequential([
    pretrained_model_without_top_layer,
    tf.keras.layers.Dense(my_num_of_classes)
])
model.summary()

model.compile(
    optimizer="adam",
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['acc'])

model.fit(X_train_scaled, y_train, epochs=10)

print("Evaluate on test data")
results = model.evaluate(X_test_scaled,y_test)
print("test loss, test acc:", results)

predicted = model.predict(X_test_scaled)
predicted = np.argmax(predicted, axis=1)

print(predicted.shape)
print(y_test.shape)

```



```
print("test loss, test acc:", results)

    test loss, test acc: [0.5251584649085999, 0.7707006335258484]

nr_of_cl = np.unique(y_test)

column_names = np.unique(y_test)
row_names = np.unique(y_test)
dim_mat = len(np.unique(y_test))

matrix = np.zeros((dim_mat,dim_mat), dtype=np.int32 )
df = pd.DataFrame(matrix, columns=column_names, index=row_names)
df

#run through predictions and compare
for x in range(len(predicted)):
    pred = predicted[x]

    valid = y_test[x]

    df.loc[pred,valid] = df.loc[pred,valid]+1

df

#print out colored confusion matrix
fig, ax = plt.subplots(figsize=(15,10))
ax=sn.heatmap(df/np.sum(df), annot=True, linewidths=.5, fmt='.0%', cmap='Reds')
plt.title('Test loss, Test acc: {}'.format(results), fontsize = 15)
plt.xlabel('Predicted label', fontsize=20)
plt.ylabel('True label', fontsize=20)
plt.show(ax)
figure = ax.get_figure()
figure.savefig('Confusion_results_Merged_mobilnet_3.png', dpi=400)
```

---

✓ 0s completed at 7:22 AM



### A.3.2 SVM and KNN code

```
pip install opendatasets
```

```
import numpy as np
import cv2
import pathlib
import PIL.Image as Image
import os
import opendatasets as od
import seaborn as sn
import matplotlib.pyplot as plt

import tensorflow as tf
import tensorflow_hub as hub
import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from google.colab.patches import cv2_imshow
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import train_test_split

from google.colab import drive
drive.mount('/content/gdrive')
data_dir = 'gdrive/MyDrive/Masterprosjekt/Depth_tester_all'

os.listdir(data_dir)
data_dir = pathlib.Path(data_dir)
classes = os.listdir(data_dir)
classes.sort()
number_of_cl = len(os.listdir(data_dir))
print(number_of_cl)
print(classes)

#Depending on what folders to use for, comment out the one you dont need
my_images_dict = {

    'MOP1': list(data_dir.glob(classes[0]+'/*')),
    'MOP2': list(data_dir.glob(classes[1]+'/*')),
    'MOP3': list(data_dir.glob(classes[2]+'/*')),
    ##'MOP4': list(data_dir.glob(classes[3]+'/*')),
    ##'MOP5': list(data_dir.glob(classes[4]+'/*')),
    ##'MOP6': list(data_dir.glob(classes[5]+'/*')),
    ##'MOP7': list(data_dir.glob(classes[6]+'/*')),
    ##'MOP8': list(data_dir.glob(classes[7]+'/*')),
    ##'MOP9': list(data_dir.glob(classes[8]+'/*')),
```

```

}

#This is where you indicate the true label
my_labels_dict = {

    'MOP1': 0,
    'MOP2': 1,
    'MOP3': 2,
    ##'MOP4': 1,
    ##'MOP5': 2,
    ##'MOP6': 3,
    ##'MOP7': 6,
    ##'MOP8': 7,
    ##'MOP9': 8,

}

X, y = [], []
IMAGE_SHAPE = (224,224)
for my_name, images in my_images_dict.items():
    for image in images:
        img = cv2.imread(str(image), cv2.IMREAD_UNCHANGED)
        img = cv2.resize(img,IMAGE_SHAPE)
        img[np.where(img>550)]=550 #tuning parameter
        img[np.where(img<400)]=400 #tuning parameter
        img = cv2.normalize(img,img,0,5,cv2.NORM_MINMAX)
        #tuning parameter
        img = cv2.normalize(img,img,0,255,cv2.NORM_MINMAX)
        #tuning parameter
        resized_img = img
        resized_img = resized_img.flatten()
        X.append(resized_img)
        y.append(my_labels_dict[my_name])

#KNN MODEL
# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state=42)
#Settings and fit training data to SVM
svc = SVC(kernel='poly',gamma='auto')
svc.fit(X_train, y_train)

    SVC(gamma='auto', kernel='poly')

predictions = svc.predict(X_test)

print("Accuracy on data is",accuracy_score(y_test,predictions))

```


```

nr_of_cl = np.unique(y_test)

column_names = np.unique(y_test)
row_names = np.unique(y_test)
dim_mat = len(np.unique(y_test))

matrix = np.zeros((dim_mat,dim_mat), dtype=np.int32 )
df = pd.DataFrame(matrix, columns=column_names, index=row_names)
df

```

	0	1	2	
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

```


for x in range(len(predictions)):
    pred = predictions[x]

    valid = y_test[x]

    df.loc[pred,valid] = df.loc[pred,valid]+1

```

df

	0	1	2	
0	45	4	0	
1	5	43	6	
2	0	8	46	

```

#Confusion matrix for SVM results
fig, ax = plt.subplots(figsize=(15,10))
ax=sn.heatmap(df/np.sum(df), annot=True, linewidths=.5, fmt='.0%', cmap='Reds')
plt.title('Test acc: {}'.format(accuracy_score(y_test,predictions)), fontsize = 15)
plt.xlabel('Predicted label', fontsize=20)
plt.ylabel('True label', fontsize=20)
plt.show(ax)

```

```

#Metrics from SVM
print(classification_report(y_test,predictions))

```

	precision	recall	f1-score	support
0	0.92	0.90	0.91	50
1	0.80	0.78	0.79	55
2	0.85	0.88	0.87	52
accuracy			0.85	157

macro avg	0.86	0.86	0.86	157
weighted avg	0.85	0.85	0.85	157

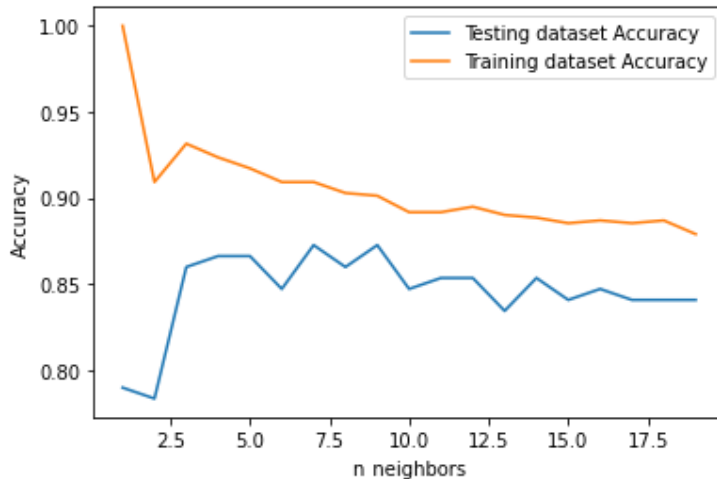
```
#Knn model
neighbors = np.arange(1, 20)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):

    knn = OneVsRestClassifier(KNeighborsClassifier(n_neighbors=k))
    knn.fit(X_train, y_train)
    train_accuracy[i] = knn.score(X_train,y_train)
    test_accuracy[i] = knn.score(X_test,y_test)

# Generate plot
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
print(test_accuracy)
```



```
[0.78980892 0.78343949 0.85987261 0.86624204 0.86624204 0.84713376
0.87261146 0.85987261 0.87261146 0.84713376 0.85350318 0.85350318
0.8343949 0.85350318 0.84076433 0.84713376 0.84076433 0.84076433
0.84076433]
```

```
print("Best Accuracy : ",test_accuracy[np.argmax(test_accuracy)])
print("k-value : ", np.argmax(test_accuracy)+1)
```

```
knn = OneVsRestClassifier(KNeighborsClassifier(n_neighbors=np.argmax(test_accuracy)+1))
knn.fit(X_train, y_train)
```

```
predictions_KNN = knn.predict(X_test)
```

```
#Metrics from KNN
print(classification_report(y_test,predictions_KNN))
```

	precision	recall	f1-score	support
0	0.85	0.96	0.90	49
1	0.81	0.81	0.81	53
2	0.96	0.85	0.90	55
accuracy			0.87	157
macro avg	0.88	0.88	0.87	157
weighted avg	0.88	0.87	0.87	157

```
nr_of_cl = np.unique(y_test)

column_names = np.unique(y_test)
row_names = np.unique(y_test)
dim_mat = len(np.unique(y_test))

matrix = np.zeros((dim_mat,dim_mat), dtype=np.int32 )
df = pd.DataFrame(matrix, columns=column_names, index=row_names)
df
```

	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

```
for x in range(len(predictions)):
    pred = predictions_KNN[x]

    valid = y_test[x]

    df.loc[pred,valid] = df.loc[pred,valid]+1

df
```

	0	1	2
0	47	8	0
1	2	43	8
2	0	2	47

```
#Confusion matrix FOR KNN
fig, ax = plt.subplots(figsize=(15,10))
ax=sn.heatmap(df/np.sum(df), annot=True, linewidths=.5, fmt='.0%', cmap='Reds')
```



```
plt.title('Test acc: {0}'.format(accuracy_score(y_test,predictions_KNN)), fontsize = 15)
plt.xlabel('Predicted label', fontsize=20)
plt.ylabel('True label', fontsize=20)
plt.show(ax)
#figure = ax.get_figure()
#figure.savefig('Confusion_results_Merged_mobilnet_3.png', dpi=400)
```

---

✓ 0s completed at 10:05 AM

