Analysing information content in images of paper straws using image processing, machine learning and deep learning

Graduate work,
Spring and summer of 2022

Written by
Ammie Blomberg
ammieblomberg@hotmail.com

Presented on August 26, 2022

September 5, 2022

Abstract

Continuous production control is important for the production using Tetra Pak equipment, as problems that are found early in the chain of production can minimize the need for waste greatly. In production of paper straws, a possible production control could be to photograph straws during production, analyse these and predict how good the straw is, using the image. If the predicted performance of the straws are steadily decreasing, production can be stopped and control parameters tuned in order to get back to peak performance.

The task described above is built on the assumption that there is information about how good a paper straw is, in a picture. This thesis has the intention of showing whether images alone can be used for production control. Methods such as image processing and the classical machine learning algorithm *Random forest regressor* will be compared to, and used together with, a deep learning option, *Convolutional Autoencoders*.

The results show that an approximate maximum of 10% of the variations in opening force can be explained by the information in the photographs. Best performing was the autoencoder, quite closely followed by image processing at just under 10% of variations explained.

The suggested improvements are many, but can be divided into two categories: further improvements of the models and information extraction, and improvements to increase predictability overall. In the first category, the inclusions of more samples, trying out other models beside Random forest regressor, and examining other network architectures for the autoencoder, are some of the advised options. In the second category, parameters that are deemed important but that cannot be seen in figures are recommended to be added to the data set. These parameters may include details on the glue used to seal the paper straw, or some sort of measure on how sharp the knifes that cut the straw are, to mention two examples.

Sammanfattning

Kontinuerlig processkontroll är en viktig faktor under produktion med utrustning från Tetra Pak. Detta eftersom problem i produktionen uppdagas i ett tidigt skede i processen, och kan tillrättaläggas, och därmed minska behovet av att slänga felproducerade produkter. Ett alternativ för produktionskontroll i fallet av papperssugrör är att fotografera sugrören, analysera bilderna, och prediktera vilken kraft detta sugrör skulle kräva för att göra hål i förpackningen sugröret i ett senare skede monteras på. Om den predikterade kraften blir högre under tid, kan produktionen temporärt stoppas, och kontrollparameterar kan ändras för att återgå till bästa möjliga produktion igen.

Vad som beskrivs ovan är endast möjligt om det finns information om hur stor öppningskraften är, i bilderna av sugrören. Om denna information finns eller ej är i nuläget okänt. Denna rapport avser undersöka om man med endast bilderna kan övervaka produktionen av sugrör. Bildanalys och den klassiska maskininlärningsalgoritmen *Random forest regressor* kommer jämföras med, och användas tillsammans med, ett deep learning alternativ, nämligen *Convolutional Autoencoders.*

Resultatet av studien visar att maximalt 10% av variationen i öppningskraft kan beskrivas med hjälp av informationen i bilderna. Bäst på detta var autoencodern, tätt följd av bildanalysen, som kunde beskriva strax under 10% av variationen.

Förslagen för fortsatt arbete är många, och kan delas in i två tydliga kategorier: åtgärder för att göra extraherandet av information bättre, och åtgärder för att göra predikterbarheten bättre. För första kategorin nämns att ha betydligt fler sugrör med i datasetet, testa andra modeller än Random forest regressor, och att undersöka andra arkitekturer hos autoencodern. För andra kategorin nämns att ha med parametrar som kan ha stor påverkan på predikterbarheten som inte syns i bilderna, i datasetet. Dessa parametrar kan vara detaljer om limmet som används i sugröret, eller något mätvärde som beskriver hur skarpa knivarna som klipper sugrören är, för att nämna två exempel.

Preface

This thesis was conducted at Tetra Pak Lund, from March of 2022 to August of 2022.

Acknowledgements

I'd like to extend my gratitude to my colleagues at Tetra Pak, Sten Sjöström, Jacopo Cavalaglio Camargo Molano, Petra Bååth, Anders Johansson, Andreas Åberg, Zabina Toroczkay and Karolina Andersson for their contributions in the work for this thesis.

I'd also like to extend a big thank you for the great discussion I've had with Alexander Andell and Jurie Germishuys at Combine Control Systems AB, regarding the development of Sympathy for Data, a program that has been key in this work.

Lastly I'd like to thank my supervisor Maria Sandsten at LTH, and my examiner Bo Bernhardsson.

Contents

1 Introduction

Tetra Pak has started developing and producing their own paper straws. One of many goals with the latest iteration of the paper straw, is to design a tip that easily perforates the package that the straw is attached to.

Small unintentional changes in the process during production may produce straws that are not up to standard quality measures. Tetra Pak has post production procedures to quality control their straws, and discard any straws that are not satisfactory. If deviations from what is considered a *good straw* can be found already at production stage, the machine could be stopped, put back into optimal performance, and started again to continue producing *good straws*. Catching imperfections earlier on in the production chain decreases waste, and possibly increases efficiency during production.

What is considered a *good straw* is not understood in terms of the visual attributes of the straw, but rather that a *good straw* is a straw that easily perforates the prepunched hole on top of the package. In the early stages of this thesis project, the question at hand was: *What attributes predict a low required force during perforation?*, but was with time changed to the final question: *Is there enough information about the straw in the images, to (with high accuracy) predict the required force during perforation?*

An engineer that has worked with paper straws for a while has a reasonably good idea about what are important attributes for a straw. Or can at least tell the extremes apart. A straw that is sharp will most likely need less force to perforate the package, compared to a straw that has no tip at all. Intuitively, a rough idea of what is important can be formed, but what are the evidence that these attributes are actually the important ones? Is there something we are missing that may be important? Is this missing thing visible in the images, or is this a parameter outside of the scope of the images?

To find the answer to if we know what attributes of a straw that are important, image processing is used to extract these attributes, and a grid parameter search for a Random forest regressor is used to investigate how well models can predict what force is required to perforate a package using that particular straw. Examples of important features may be what angle the straw is cut at, or its width a set distance down from the tip of the straw.

To find the answer to if enough information about the straw is in the images at all, a convolutional autoencoder is used, to in an unsupervised manner find the important features of the straws. The output from the autoencoder is in a similar fashion as the parameters from image processing used in doing a grid parameter search, to find how good the best models are.

The data consists of force measurements for straws, and photographs of these straws. The photographs are taken from the front, and from the side, against a black background. The images are well lit, and of high resolution (5472 pixels wide, 3648 pixels high).

Before the study of the above mentioned methods were initiated, a Gage R&R study was carried out, to evaluate the influence imposed on the process by the operand and method for photographing the straws. Since this study is not a central

part of this thesis, this will have its own section in the report.

The question about using classical approaches or deep learning approaches is a highly relevant question to ask. While image processing and classical machine learning algorithms have a lot to offer in terms of that they are often quite easy to interpret, they may require a lot of background knowledge of the problem at hand - knowledge that might not exist yet. This knowledge is not necessarily needed when using deep learning. Deep learning can find complex connections within data that we cannot. This is however often at the expense of interpretation - how can we know what was important? How can we better production, if we don't know what went wrong, only that it did go wrong? These concepts are discussed in [3], and acted as a first spark into the interest of comparing image process to deep learning for production control.

2 Theory

This chapter will go over the algorithms, methods and metrics that are used throughout the thesis. How and why these methods are used will be further described in *Method description and Results*, section 3, and *Gage R&R study*, section 4.

2.1 Random forest regressor

This section will introduce the machine learning algorithm *Decision tree*, and further build upon that algorithm to end up with the *Random forest regressor*-algorithm, that will be used extensively in this thesis.

2.1.1 Decision Trees

A *Decision Tree* is a supervised machine learning algorithm, which can be used for both classification and regression. Its name is a good descriptor of how the method works. A simplified example of a Decision Tree can be found in figure 1. The algorithm looks at the presented data, and tries to find statements regarding the available parameters, to separate the samples into the categories presented to the algorithm [11].

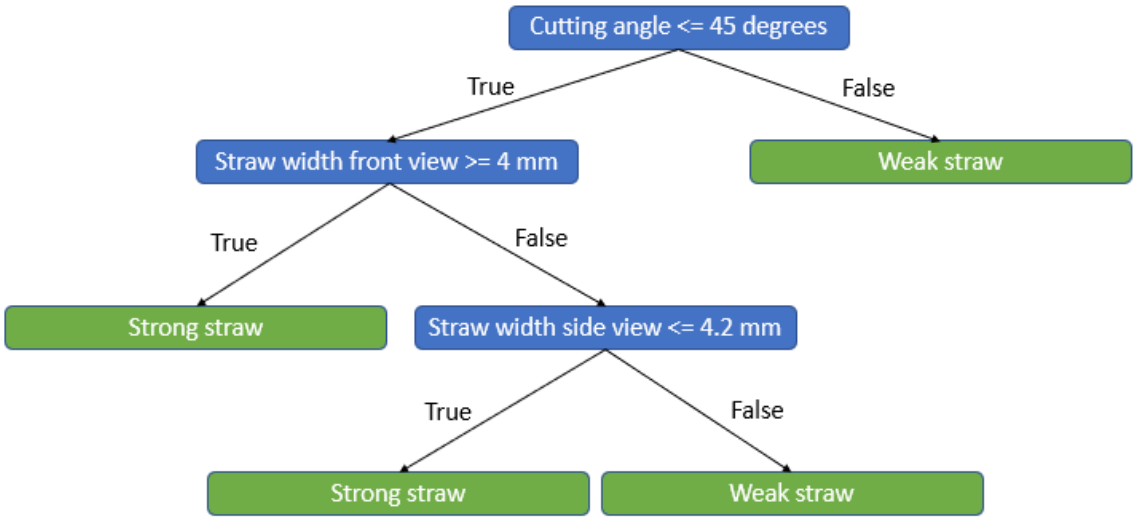

Figure 1: Example of how a Decision Tree using sample data from this work may look, where a sample is to be predicted, is it a *strong straw*, or a *weak straw*? Blue shapes are decision leaves, and green shapes are final leaves. Starting from the top, the algorithm work its way down depending on what criteria the sample satisfy. The two arrows under each decision leaves describe how a decision is made. If the cutting angle is greater than 45 degrees, the straw is predicted to category *weak straw*. If the cutting angle is smaller or equal to 45 degrees, further separation is needed to predict if the sample is of category *weak straw*, or *strong straw*.

The strength that Decision Trees have is that they are intuitive, and follow the logic of how humans think, very well. They are not as abstract in their application

as other machine learning algorithms are. A clear downside to Decision Trees is that they are very inaccurate, they tend to be overfit to the data it is trained on. They are therefore unfortunately often very poor at predicting on unseen data.

The Decision tree model interactions, non-linearities and is insensitive to outliers. It does however not have the ability to transform the input data. Input transformations could for example be to invert the value of a parameter, create linear combinations of parameters, or quotas of parameters.

2.1.2 Random forests

To combat that the Decision tree performs badly on unseen data, multiple sets of decision trees can be used. This ensemble of Decision trees is trained on different subsets of the data, which increases the regularisation (reduces the tendency to overfit) of the method, and increases its predictability on unseen data.

In a Random forest, when a prediction on a sample is to be made, the sample is run through all Decision trees in the forest, and the mean of all predictions, is the final prediction.

Like most machine learning algorithms, Random forests have a range of tunable parameters, that need to be adjusted to the specific problem at hand in order for the algorithm to perform at its best. In the case of Random forests, the central parameters to change are how many trees are in the forest, and the maximum depth of all these trees [4].

The number of trees in the forest is how many Decision trees will be used in predicting the dependent variable. More trees are better than fewer trees in reducing overfitting.

The depth of the trees aids in reducing overfitting of the trees. If the tree is only allowed to go a certain depth, it can only afford to use the most central parameters given to it, in order to find the best split of the data. Having a low number for the maximum depth will most likely give low purity levels for the leaves. The Decision tree presented in figure 1 has a depth of three - the tree uses three instances of parameters (same or different parameters) to divide the data into categories.

Purity is a measure of how good the tree has separated the samples in the leaves. The highest purity corresponds to a *clean* leaf, only one type of sample is present in that leaf. A low purity corresponds to a great deal of mixing of samples in the leaf - the tree has not managed to sort out all the differences between samples. A single Decision tree without any depth restriction will sort the data until all leaves are pure.

2.2 K-means clustering

K-means clustering is an unsupervised clustering algorithm. The K in K-means clustering is the number of clusters the data is divided into, using the algorithm. [9]

The algorithm is initiated by randomly selecting k of the available data points, and have these be the initial centroids, another word for the central points of the k clusters. Each of the data points in the data set is assigned to be part of the cluster that has the lowest distance to that centroid. In other words: the centroid closest to the data point, is the center of the cluster that data point is assigned to

be part of. After all points have been assigned to a cluster, the mean value for each cluster is calculated, and this mean value is now used to reassign clusters to all data points, but it is now the distance to a mean that is the deciding factor, instead of the distance to the centroid. This procedure of finding the mean of each cluster and reassigning memberships to clusters is continued until the difference between one iteration to another is non existent.

Sometimes the initial centroids are placed unfortunately, making the method unable to converge to what the human eye would easily interpret as the correct, or most intuitive, clustering. A way for the algorithm to combat such a problem is to take note of the total variation of the resulting clusters, meaning the variance of the distance between each sample, and its assigned centroid. It re-initiates the k clusters, repositions until no further change, and again takes note of the total variance for the clusters. This is preferably done many times, and the best clustering is the clustering where the variance is divided as evenly over the k clusters as possible, while also being as small as possible overall. An important parameter to evaluate the performance of the clustering, is called *inertia*, and is defined as the negative sum of squared distances of samples to their assigned centroid.

2.3 Grid parameter search

As mentioned, machine learning algorithms may have a great number of parameters that need to be tuned, in order for the algorithm to fit the best model to the application. In the case of a Random forest regressor, two important measures were mentioned in previous section, *maximum depth*, and *number of trees in the forest*, in the case of K-means clustering, it is the number of clusters K.

What the optimal values for these parameters are, are initially unknown. There are no settings that work for every application, since the models are very data dependent. To find the best performing model, a *Grid parameter search* can be performed.

A grid parameter search is a way of strategically test settings, and evaluate the performance of the model when using these settings. The best performing model, is the model that is put forth for further analysis. This is usually done with cross-validation, meaning that the model is trained with the same settings multiple times, but different sets of data, and its performance is evaluated on the data that is not used for building the model. Grid parameter searches can be very time costly, and the least amount of settings tested are preferred to save time. Decreasing the number of cross-validations decreases the total time needed for the search, but at the detriment of reliable results. The metric used to evaluate the performance of a set of models is the mean value of the *Coefficient of determination*, r^2, over all iterations of the cross-validation. The coefficient of determination is defined in section 2.6.

2.4 Autoencoders

An autoencoder is a double-sided artificial neural network, which job is to recreate the input, by first compressing it to a code. It is good at finding complex relations within the data, and compress this into data of lower dimension. Figure 2 shows

a great example of how an autoencoder would encode a data set, compared to the popular algorithm *Principal Component Analysis, PCA*.

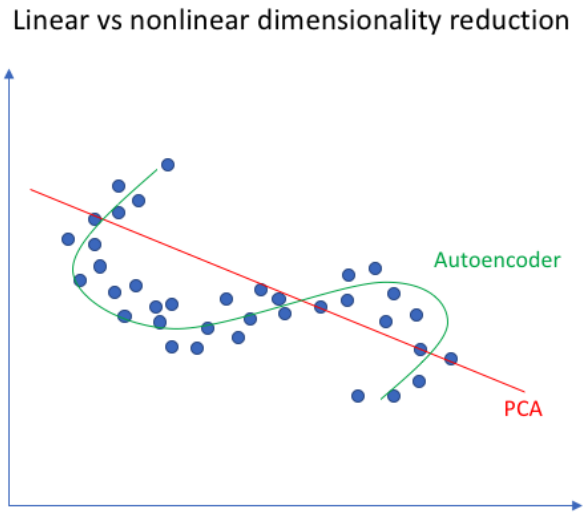

Figure 2: Example of how an autoencoder can find non-linear dimensionality reductions in data, that other linear algorithms (such as Principal Component Analysis, PCA) cannot. The data presented here seemingly only need one dimension to describe it rather accurately using an autoencoder, namely: how far out on the line does the data exist? Image from: [5]

The autoencoder consists of two distinct parts: the *encoder*, and the *decoder*. The encoder takes the image and compresses it in multiple steps, into a smaller amount of numbers, the code. The decoder takes this smaller amount of numbers, the code, and tries to recreate the original image. The size of the code is a hyperparameter you set before training begins, a smaller code implies a greater compression. The code can also be referred to as the bottleneck of the autoencoder. What is trained in an autoencoder are all the weights in the convolutional filters in the encoder, the weights in the fully connected layers in the encoder, and the corresponding weights on the decoder side. Note that the weights do not need to be the same value on the encoder and decoder side.

An Autoencoder has an array of data as input, where as a convolutional autoencoder (CAE for short) has an image as input. What is added to the architecture in figure 3, are convolutional layers before the encoder (to turn the image into an array), and after the decoder (to turn the array into an image). A convolutional autoencoder is a more complex version of an autoencoder, suited for images. Images could be flattened into an array and used in an ordinary autoencoder, but that method is not used here.

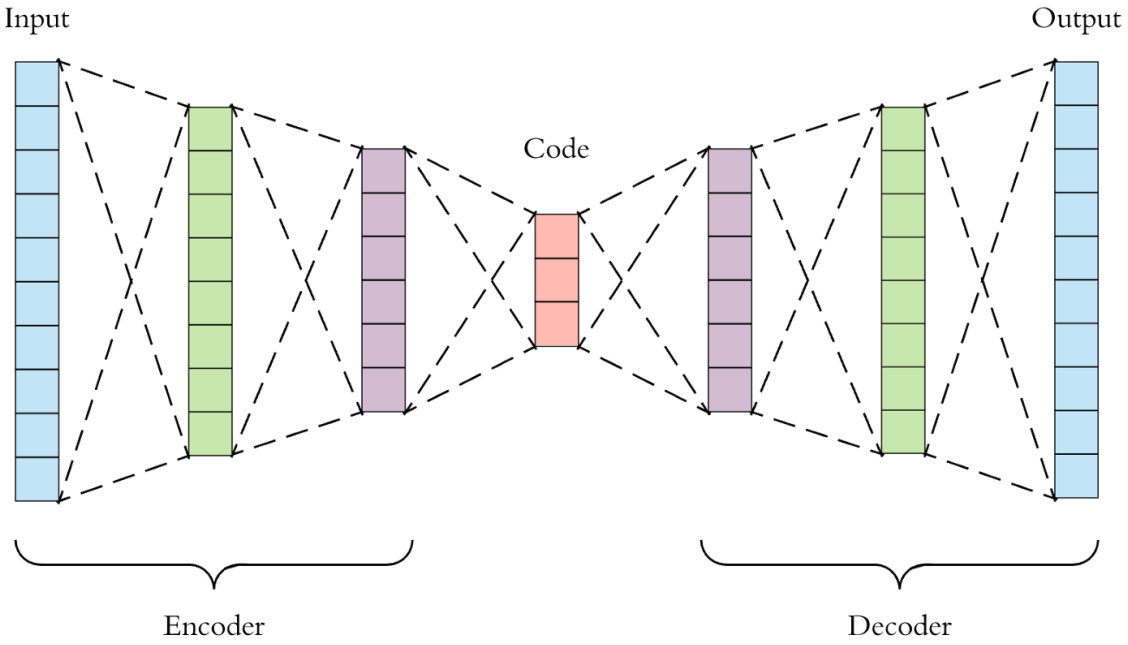

Figure 3: Overview of how an autoencoder looks. The input to the left is what is sought to be compressed. The left side of *Code* is the encoder, and the right side is the decoder. The network is trained to reproduce the input. The encoder and decoder are each others reverse visually, although this is not necessary by design. Image from [1].

2.5 Mean-squared error loss

A loss function for training the autoencoder is called *Mean-squared error loss*, short: *MSELoss*, and is a pixel by pixel comparison between the original image, and the reconstructed image.

The Mean-squared error loss is defined in equation (1), where $\hat{y}_i$ is the predicted value, y_i is the correct value, and i runs over all pixels in the image. [10]

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{1}$$

2.6 Coefficient of determination, r^2

To determine the accuracy of the obtained models, the *coefficient of determination* will be used, otherwise known as r^2. r^2 is defined in equations (2) to (5). [2] It is a measure of how much of the variations in the dependent variable can be explained by the independent variables. In the equations below, $\vec{y} = [y_1, ..., y_n]$ are the true values, and $\vec{f} = [f_1, ..., f_n]$ are the predicted values.

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2}$$

$$SS_{res} = \sum_i (y_i - f_i)^2 \tag{3}$$

$$SS_{tot} = \sum_i (y_i - \bar{y})^2 \tag{4}$$

$$\mathrm{r}^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{5}$$

$\bar{y}$ is the mean of the observed data, SS_{res} is the sum of squares of residuals, and SS_{tot} is the total sum of squares. The model predicts the values for f_i, meaning that there is nothing hindering SS_{res} from being bigger than SS_{tot}, and thereby creating an arbitrarily big negative value for r^2, corresponding to an arbitrarily bad model. A perfect model has an r^2 value of 100%.

2.7 Gage R&R study

Gage R&R is short for Gage Repeatability and Reproducibility. It is a study conducted to investigate if the variations found during testing, are due to true variations in the test sample, or due to the operand during measurement, or how the measurement was done. [6]

A set of operands is asked to follow the same directions, and asked to repeat the process at a later date. The study uses different operands at different dates to examine how well the instructions and method enable reproducibility, and repeatability, in the measured parameter(s).

3 Method description and results

This section covers the main work of the thesis. It begins with a description of the clustering that was conducted to find what straws to include in data set 2. It will describe the two data sets, with figures and tables. After the data sets are introduced, the measures taken to improve the performance of the autoencoder and its results are presented. The final autoencoder is used to create a code, which is used in three separate tests to evaluate the information content. After that, the image processing version is presented, where the parameters are shortly described, and then investigated for information content.

3.1 Clustering

In order to find the different *types* of straws that exist, data from image processing of 1348 unique straws was clustered, using K-means clustering. Two grid parameter searches were conducted: one coarse, and one finer building upon the result from the coarse parameter search.

3.1.1 Coarse grid parameter search

3.1.1.1 Method description

A *Grid parameter search* was conducted to find the optimal number of clusters. The parameter search was focused on the number of clusters, as opposed to also including parameters like how many times the centroids are to be placed, how many iterations each run is allowed to conduct, et cetera.

An initial test where K was odd integers from 3 up to 25, (3, 5, 7, ..., 23, 25) was performed. Centroids were initiated 10 000 times for each number of clusters, and iterated a maximum of 1000 times. Cross-validation was used, with validation done 20 times for each cluster-trial. This initial parameter search is intended to find a range for where the optimal value for K should be.

3.1.1.2 Results

The initial parameter search using odd integers from 3 to 25, yields the test score plotted against value for K, as found in figure 4. The sharp increase in inertia for $K = 5$, gives rise to a suspicion that the optimal value for K is somewhere in that region. Increasing the amount of clusters further, seem not to increase inertia notably.

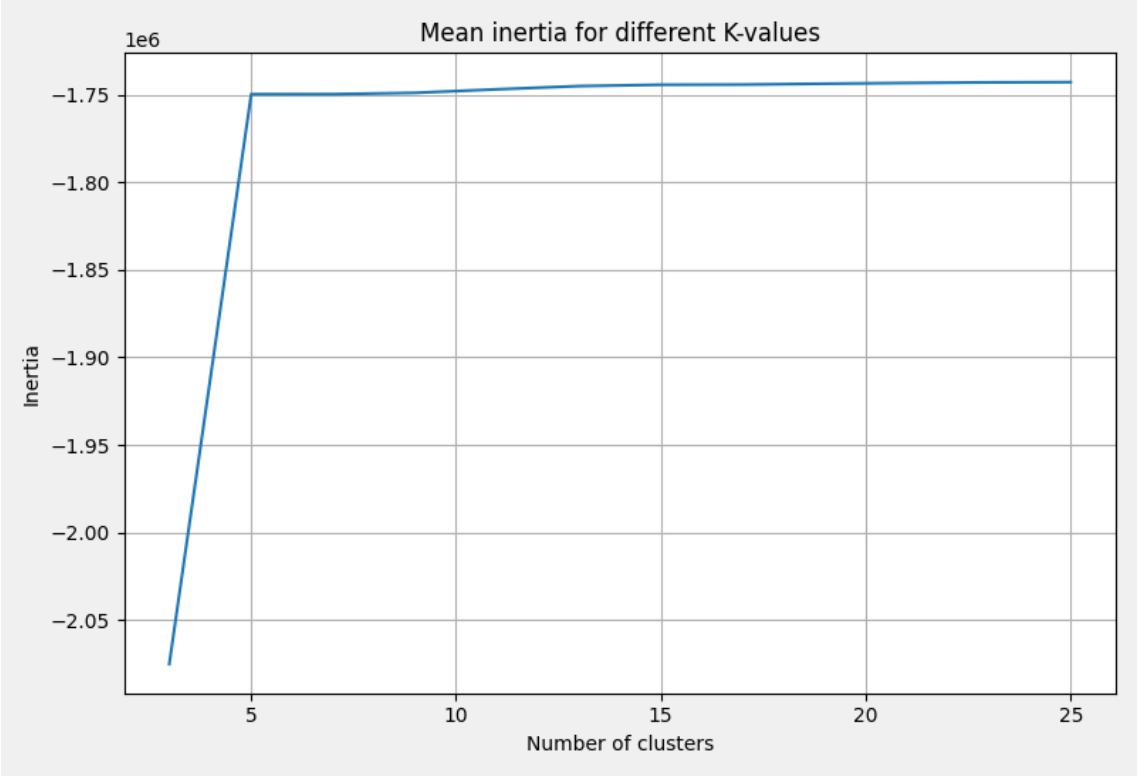

Figure 4: Results of the grid parameter search for the K-means clustering algorithm, where the number of clusters are odd integers from 3, up to and including 25. "Mean inertia" refers to the mean value of inertia, for the 20 runs of cross-validation. Notice the sharp increase in inertia for $K = 5$, and notice also how the inertia doesn't increase notably for higher values of K.

3.1.2 Fine grid parameter search

3.1.2.1 Method description

The results from the initial test showed that the optimal value for K should be somewhere between 3 and 5. A second grid parameter search was conducted, this time on integers from 3 up to and including 10. Centroids were initiated 10 000 times for each number of clusters, and iterated a maximum of 1000 times. Cross-validation was used, with validation done 20 times for each cluster-trial. The second parameter search was to conclude in greater detail, what the optimal value for K is.

An equal amount of straws from each cluster were randomly chosen, and sent for opening force analysis.

3.1.2.2 Results

The second parameter search using integers from 3 up to and including 7, can be found in figure 5.

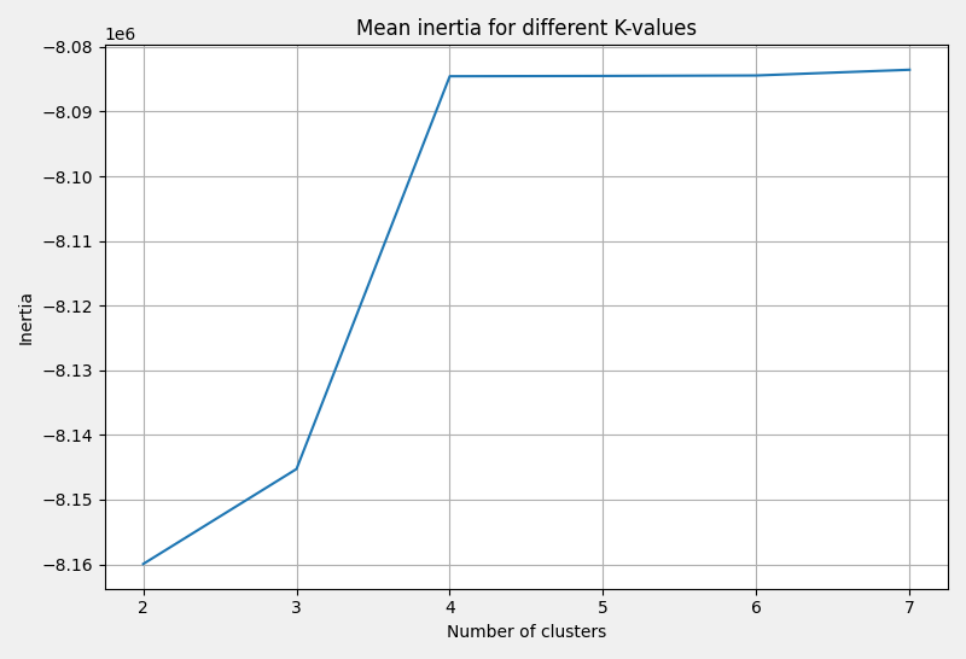

Figure 5: Results of the grid parameter search for the K-means clustering algorithm, where in this case the number of clusters are integers from 3, up to and including 7. "Mean inertia" refers to the mean value of inertia, for the 20 runs of cross-validation. The sharp increase in inertia for $K = 4$ suggests that the optimal value for K is four. In this figure, as well as in figure 4, no big changes in inertia are present for higher values of K.

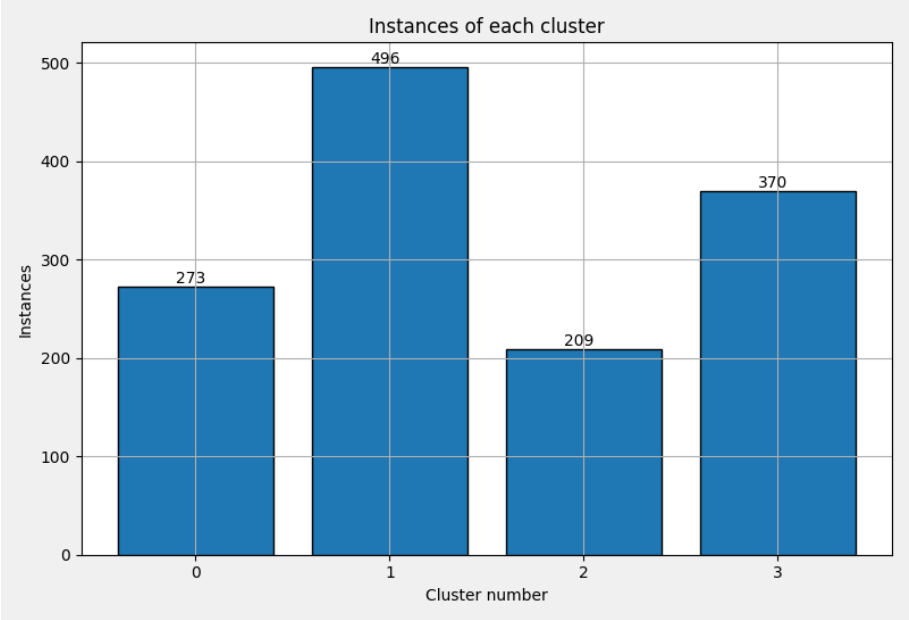

Figure 6: The resulting cluster division, using K-means clustering, where $K = 4$. The number of straws in each cluster can be found in the figure, with instances ranging from 209, to 496.

The separation of straws into the four clusters, yields a division into categories, as can be seen in figure 6.

3.2 Data sets

A description of how the required force is measured can be found in appendix, section B. The force measurements were conducted by colleagues at Tetra Pak, not by myself. One important detail of the measurements is the direction of the pre-punched hole in the package, the details are left for the section of text mentioned above, but what is important to mention is that the material has different properties depending on which way it is perforated, and two directions of interest in the material are named MD (machine direction), and CD (cross direction). As can be seen from this section, the difference is non-negligible, and whether the acquired force is measured in MD or in CD is important to include with the rest of the independent variables, in order for a model to have a fair shot at accurately predicting the opening force.

For each measured force, there is one picture of the straw taken from the front, and one picture of the same straw taken from the side. The method of photographing the straws is further detailed in section A in appendix.

3.2.1 Data set 1

Data set 1 consists of 160 samples. The force measurements are plotted in histograms in figure 7, and statistics divided per MD/CD and both together, can be found in table 1. The upper figure shows forces measured in CD, the bottom figure shows forces acquired while measuring in MD. The values for MD are higher for min, mean and max, but has a lower standard deviation, than CD has. This data set has been available since May 6th 2022, and is used for the improvements of the autoencoders, as well as in the final autoencoder, and image processing. The straws that are pictured were produced at the end of April 2022.

Category	**Min**	**Mean**	**Max**	**StDev**
MD+CD	4.47	8.97	16.81	2.45
MD	5.69	9.75	16.81	2.23
CD	4.47	8.19	15.92	2.41

Table 1: Statistics over data set 1, divided for measurements MD/CD, and both together. Notice how min, mean and max are higher for MD, where as standard deviation is higher for CD.

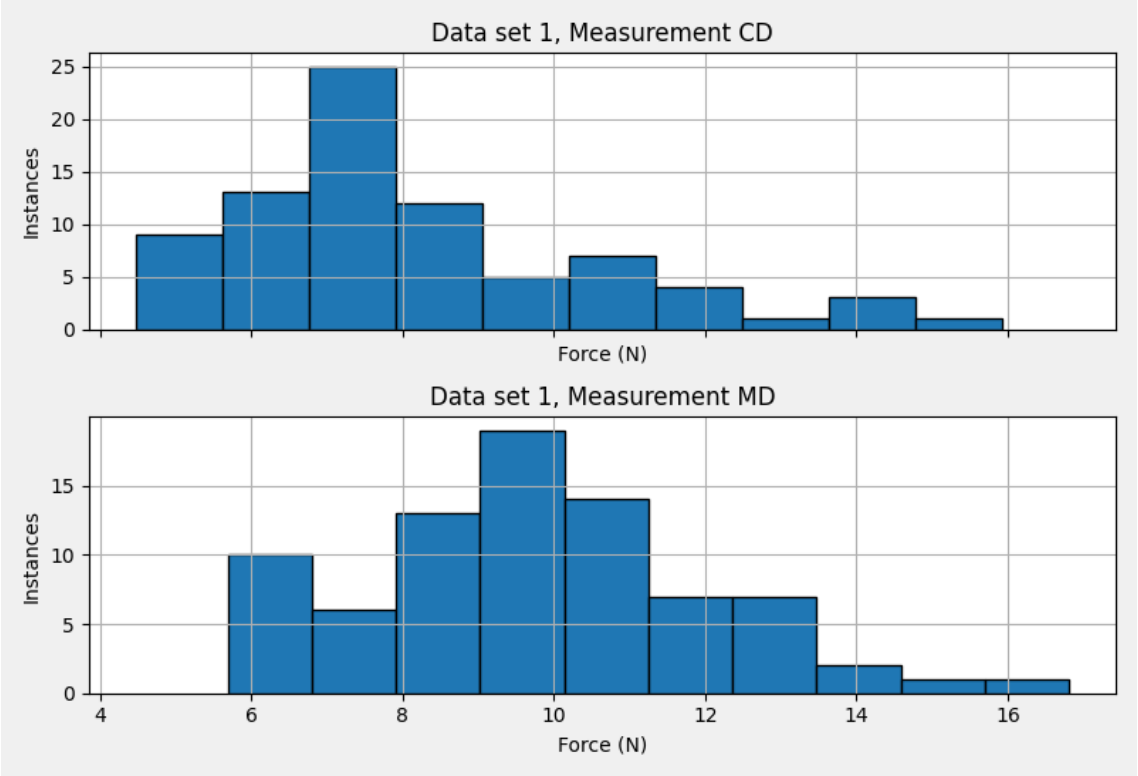

Figure 7: Histograms showing force distributions for measurements in CD versus MD, for data set 1. Notice that the lowest value for CD, is lower than the lowest value for MD, and that the highest value for MD is higher than the highest value measured in CD. Notice that this is a non-normal skewed distribution.

3.2.2 Data set 2

Data set 2 consists of 157 samples. Three of the 160 samples were damaged upon arrival to the lab, and could not be force tested. Histogram of the force measurements for data set 2 can be found in figure 8. Statistics (min, mean, max and standard deviation) can be found in table 2. This data set has been available since August 2nd 2022, and is used in the final autoencoder, and image processing. These straws were produced at the end of June 2022.

Category	**Min**	**Mean**	**Max**	**StDev**
MD	5.19	8.58	17.24	1.85

Table 2: Statistics over data set 2.

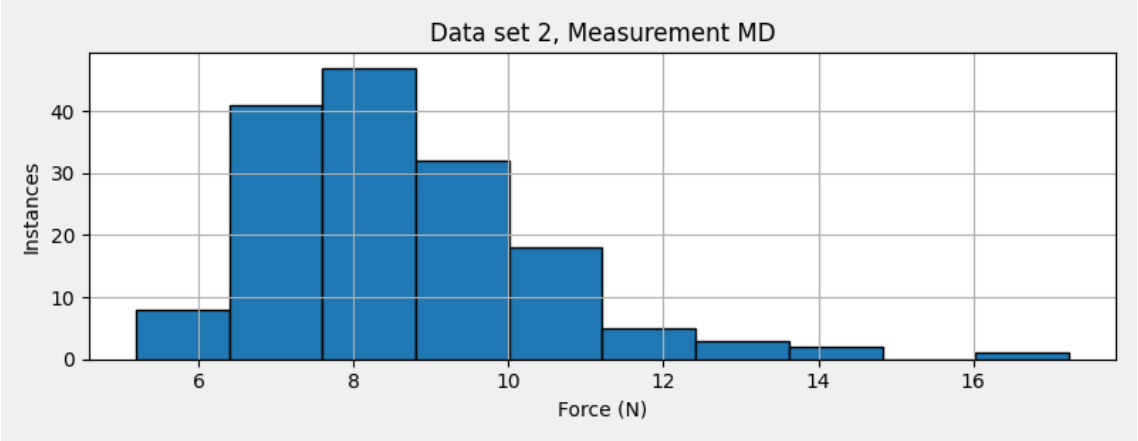

Figure 8: Histogram showing force distributions for measurements in MD, for data set 2. Notice that this is a non-normal skewed distribution.

3.3 Improving performance of Convolutional Autoencoders

The downside of image processing is that previous knowledge about the process is needed: I need to know what could be important to predict opening force. That is not an issue when working with deep learning alternatives, like a convolutional autoencoder.

Four different sets of autoencoders will be used. One autoencoder which we will refer to as the benchmark, this is the standard autoencoder without any performance improvement measures. To improve the performance of the autoencoder, the contrast in the images will be improved and used for a second set of autoencoders. The third set of autoencoders will have random images included in the data set, to further improve performance. The last autoencoder is a combination of set two and three. Each autoencoder will be described in turn, with method and result following for each autoencoder. What is important to look at when evaluating the performance of an autoencoder, is how well it reconstructs an image, not only images it has been trained on, but also images of a kind it has never encountered before. Autoencoders that have been trained on front images will therefore be asked to reconstruct front images, but also side images. Beside these reconstructions, the training history will also be included for each autoencoder.

The presentation of the results will follow a format, best described by figure 9.

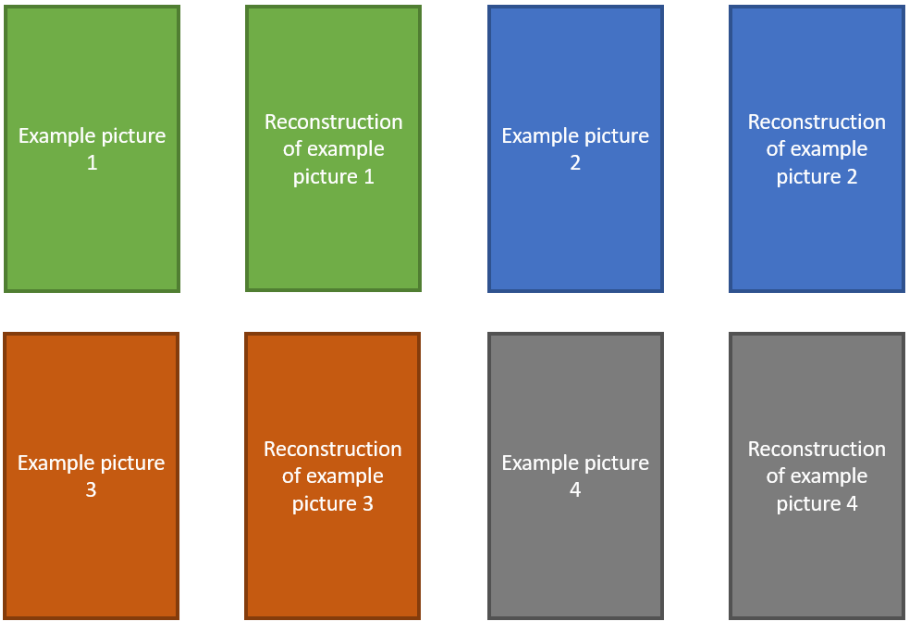

Figure 9: Descriptive image showing how to interpret the figures showing the results from the autoencoders. The images are to be analysed in pairs, where the left image is the image the autoencoder is asked to reconstruct, and the image on the right is what the autoencoder outputs. These example images can be images of straws, or random images added to the data set.

3.3.1 Benchmark autoencoder

3.3.1.1 Method description

As a benchmark to compare further development on, the 160 straws in data set 1 were used to train two autoencoders. One autoencoder was trained on front images, and one autoencoder was trained on side images. Both autoencoders had latent-space dimension 20, with two convolutional layers in the encoder, followed by two fully connected layers with 28 nodes, and 24 nodes. Encoder and decoder were each others inverses. The autoencoders were trained with a learning rate of 0.5, for 300'000 epochs each. The loss function was the mean-squared error loss (MSELoss), and the criterion used was Stochastic Gradient Descent (SGD). The models were all trained on a Tetra Pak issued laptop, with GPU support. The definition of the autoencoder can be found in Appendix, section C. Different learning rates were used in the early stages, but set to 0.5 after a while, as that was a good compromise between a too aggressive learning rate, and a learning rate that required the models to train for too long of a time. Other loss functions, criteria or latent-space dimensions were not studied.

3.3.1.2 Results

Examples of the reconstruction of images using the first approach to convolutional autoencoders can be found in figure 10 for front images, and 11 for side images. In each of these two figures, the network is asked to reconstruct on the type of images it was trained on: the autoencoder trained on front images is asked to reconstruct

front images, and the autoencoder trained on side images is asked to reconstruct side images.

Notice that the reconstructions are fairly accurate when it comes to the outline of the straw, where as the details within the straw, mainly where you can see the inside of the straw in the front images, are not well reconstructed at all. The side reconstructions have little to no contrast in them, they are just white blobs.

In figure 12 you can see the autoencoder trained on front images, reconstruct side images. In figure 13 you find the opposite situation: the autoencoder trained on side images, reconstruct front images.

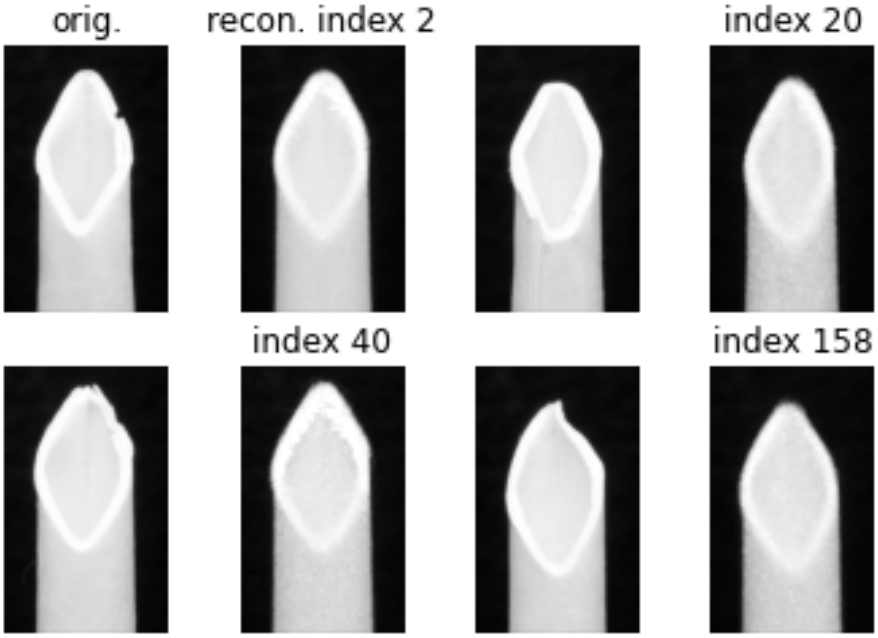

Figure 10: Benchmark autoencoder trained on front images, reconstructing front images. Notice that the outline of the straws are reconstructed well, overall. The convex part of the straw on the lower right is not reconstructed, and no details in the straw are reconstructed.

The reconstructed images from the autoencoder trained on another orientation of the straw, still show a very clear resemblance to the orientation it is trained on. The autoencoder trained on front images reconstruct front images even though it is given a image taken from the side, and the same goes for the autoencoder trained on images taken from the side, when asked to reconstruct images taken from the front.

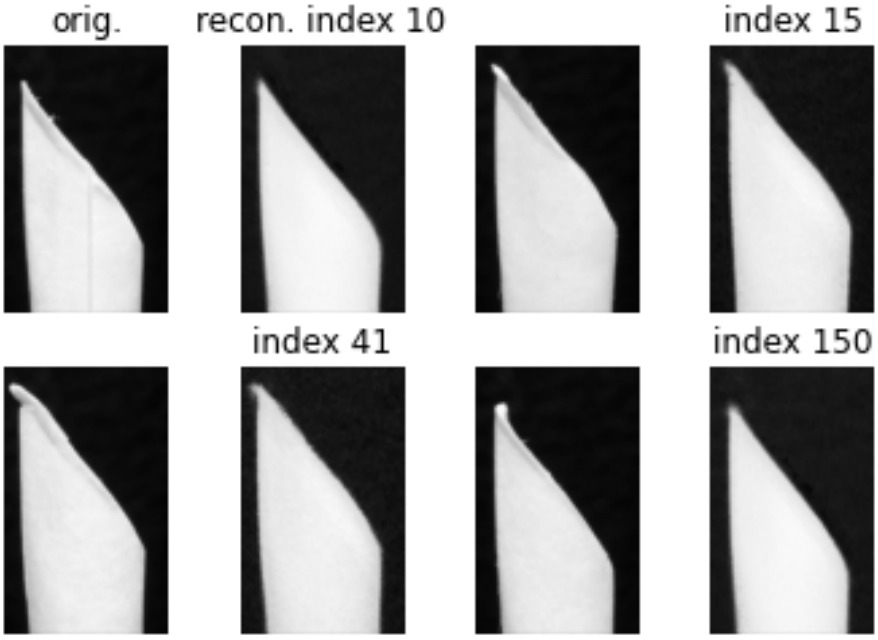

Figure 11: Benchmark autoencoder trained on images side, reconstructing images side. The outline of the straws are reconstructed in a varying capacity: the bend in the tip in the lower left straw is reconstructed, but the inward bend on the straw to the lower right is not. The seam that can be seen in the straw on the upper left is not reconstructed either.

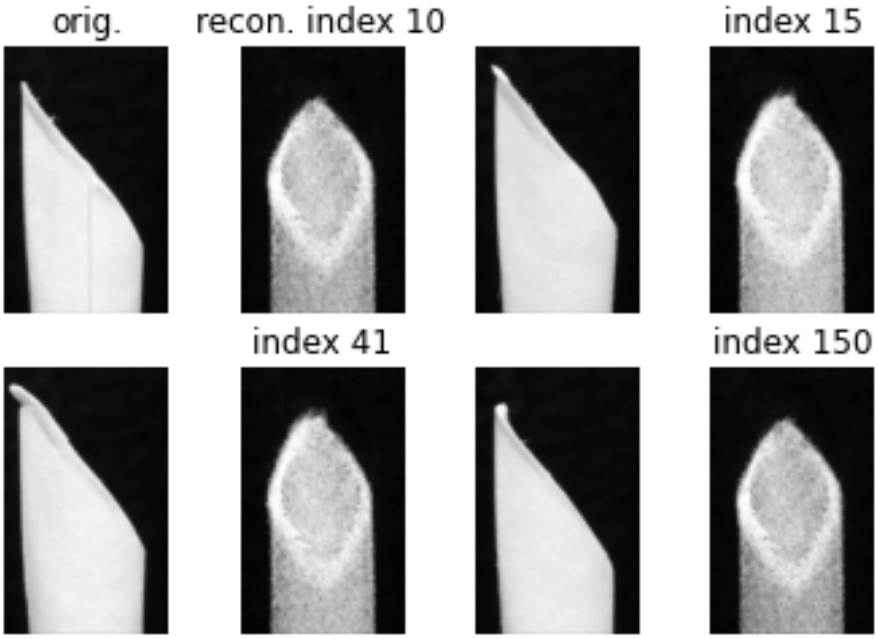

Figure 12: Benchmark autoencoder trained on front images, reconstructing side images. All reconstructed straws look like images of straws taken from the front.

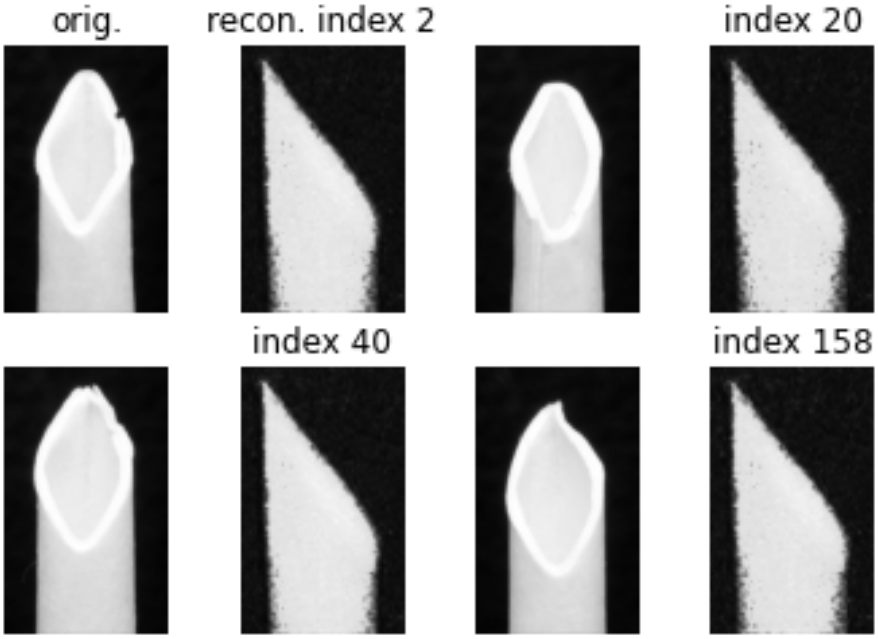

Figure 13: Benchmark autoencoder trained on side images, reconstructing front images. All reconstructed straws look like images of straws taken from the side, similar to the opposite situation in figure 12.

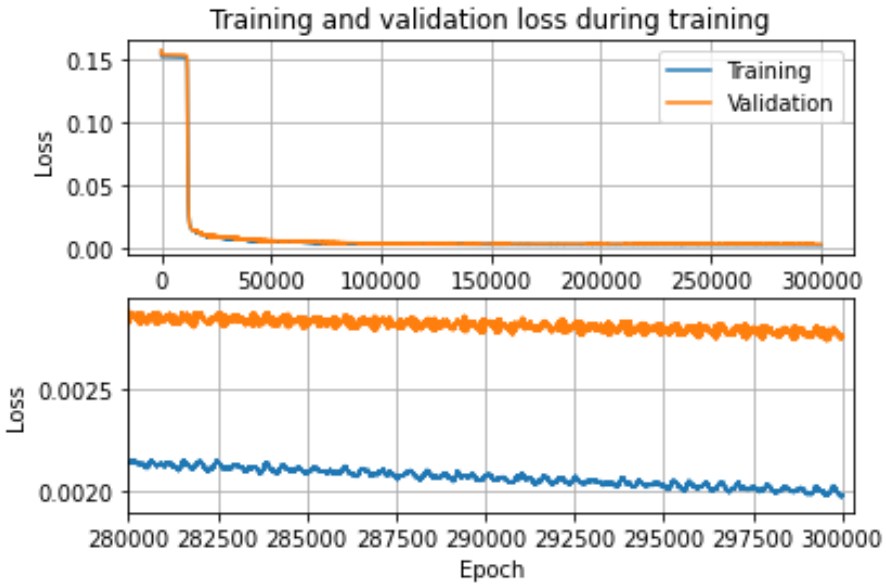

Figure 14: Training history of benchmark autoencoder trained on front images. Top figure is history from start to finish, where as the bottom figure is showing only the last 20'000 epochs of training. Both training and validation loss is decreasing over the last 20'000 epochs of training. No global minimum for validation loss has been reached.

Training history for the autoencoder trained on front images can be found in figure 14, and figure 15 for the autoencoder trained on side images. No global minimum

for validation loss was reached in any of the two trainings, and the training and validation loss is decreasing over the last 20'000 epochs of training.

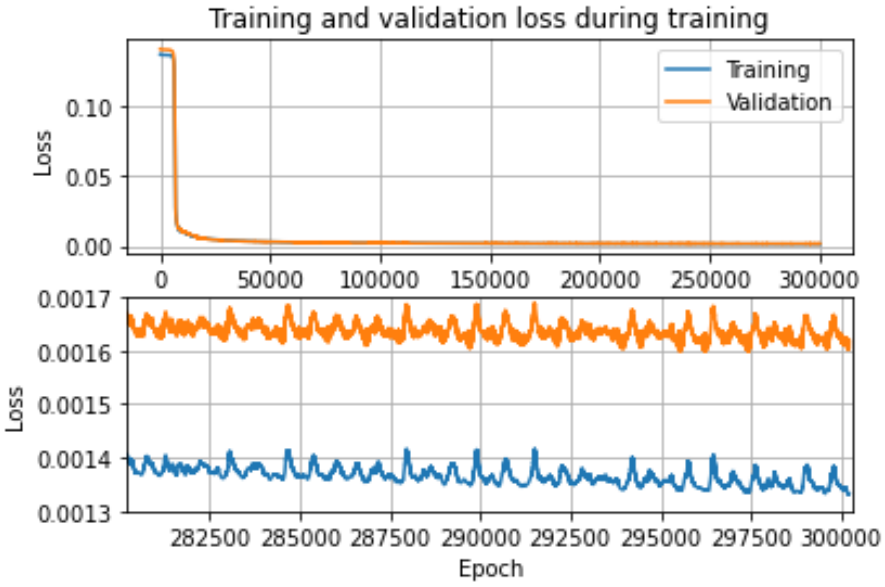

Figure 15: Training history of autoencoder trained on side images. Top figure is history from start to finish, where as the bottom figure is showing only the last 20'000 epochs of training. Notice that the trend for the training loss is still negative, during the last 20'000 epochs, no global minimum for validation loss has been reached.

3.3.2 Increase contrast in straw

3.3.2.1 Method description

The error function used, MSELoss, is a pixel by pixel comparison between the original image, and the reconstructed image. This makes the difference between predicting black (pixel value of 0) and white (pixel value of 1) big, giving a big value for the loss, making the model very good at telling a black pixel from a white pixel. However, this does also mean that the difference between predicting almost white (pixel value of 0.8) and white (pixel value of 1) is much smaller. The model will therefore find it more rewarding to tell black from white, rather than almost white from white.

The images of the straws are good in the sense that it is rather clear what is important in the image. The background is very dark, compared to the light foreground that holds the straw. The details in the image of the straw are however differences on the scale of almost white and white, which the loss function used does not favor.

One way to combat this would be to change the loss function into one that is more sensitive for higher pixel values. This would require a lot of coding on my part. Another way would be to increase the contrast in the straw, making the differences in details bigger in terms of pixel values. It is the increasing of contrast that will be done in this thesis, as that is more time efficient than coding a new loss function.

An example of how a straw may look after the contrast increase can be seen in figure 16.an be seen in figure 18.

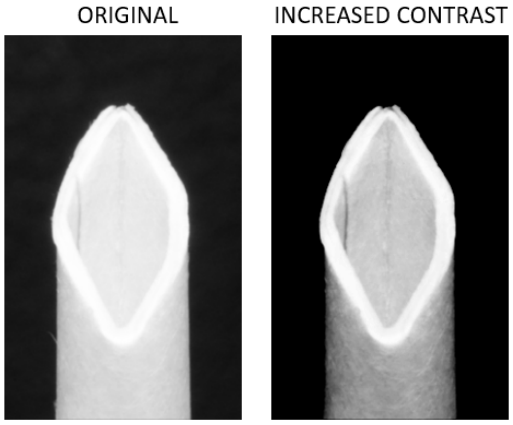

Figure 16: Example of how the increase of contrast looks. To the left is the original image of the straw, and to the right is the same straw but with increased contrast. Notice that the details in the straw are more clear, while the overall shape of the straw is still the same as before enhancement.

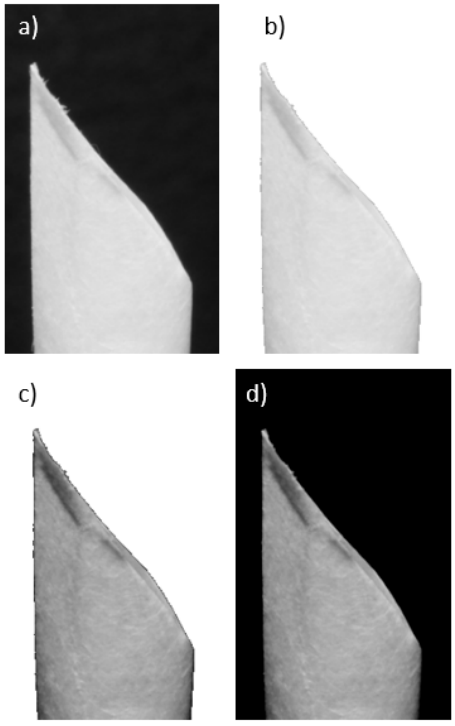

Figure 17: Four images showing the intermediate steps of increasing the contrast of the straw. a) is the original image. b) shows all pixels with a value above 0.6, on a white background. c) shows the straw after normalisation, and d) shows the finished image of the straw, after the contrast increase is finished.

The increase in contrast was done using a program called *Sympathy for Data*, which is described in more detail in section 3.5. Please refer to figure 17 when

following the text. All pixels in the original image (image a) is the original image) that have a value above 0.6 will be used in the contrast increase, as only pixels depicting the straw have such a high pixel value. The pixels that do have a pixel value higher than 0.6, are pictured in b), on a white background. Image b) is normalised, meaning that instead of having values between 0.6 and 1, these are stretched over the interval 0 to 1. The resulting image is c). The normalised values for the straw are put on a black background, to finalise the contrast increase, and the final result can be found in d). The changes in pixel values can be seen in figure 18.

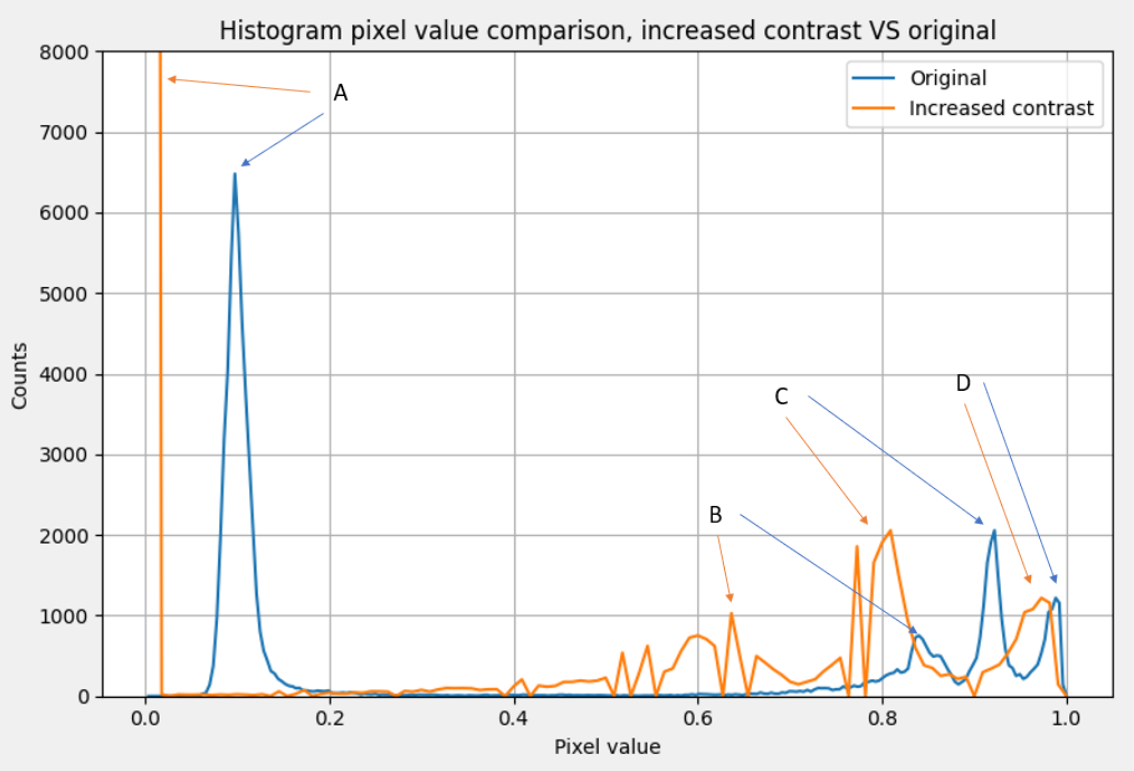

Figure 18: Plot showing the distribution of pixel values in the original image, compared to the increased contrast version of the same image. Notice how peak A has been moved from around 0.1 to 0 (this is the background), while peaks B to D have been spread out over a larger interval of pixel values (these are the details in the straw).

When it comes to the autoencoders, the network architecture, learning rate and maximum epochs are the same as described in section 3.3.1.1, the benchmark autoencoders. The only difference is that the data set now is of images of straws with increased contrast.

3.3.2.2 Results

Examples of reconstructing the same set of straws as in previous section, can be seen in the following figures: figure 19 for front images, and figure 20 for side images. In each of these two figures, the network is asked to reconstruct on the type of images

it has already seen and been trained on. The autoencoder trained on front images is asked to reconstruct front images, and the autoencoder trained on side images is asked to reconstruct side images.

In both figures there are more details in the straws after reconstruction, as compared to the reconstructed images from the benchmark autoencoders, see figure 10 and 11. There is a dampening of the features in the straw, most easily seen in figure 20 on the straw on the lower right: the crease under the cut is reconstructed, but at a much milder rate. The tip of the straw on the lower left in the same figure is less well reconstructed, as compared to the benchmark autoencoder, figure 11.

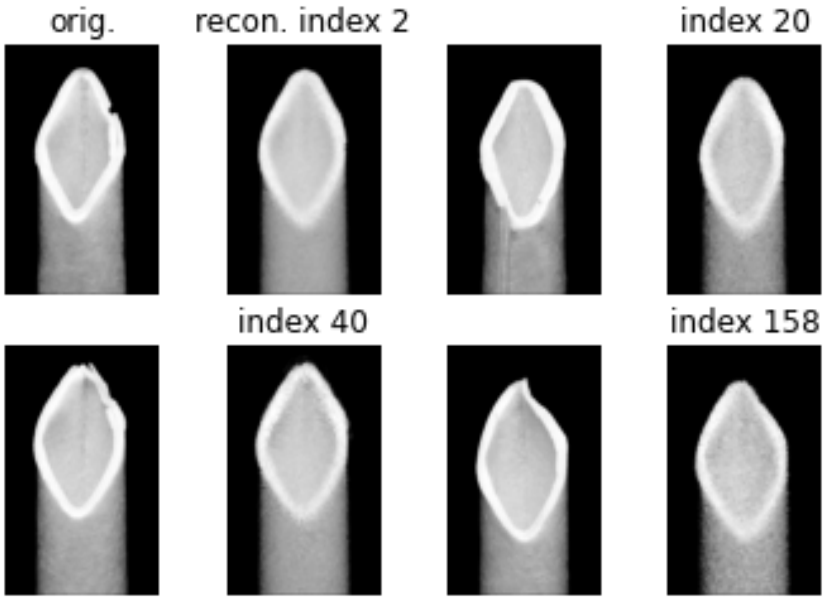

Figure 19: Autoencoder trained on front images where contrast has been increased, reconstructing front images. The ellipse formed by the cut is clearly reconstructed, in all images. The creases on the outline of the lower left straw are not reconstructed, but the convex bend in the outline of the straw on the lower right is reconstructed minimally, only showing a small bend in the corresponding place.

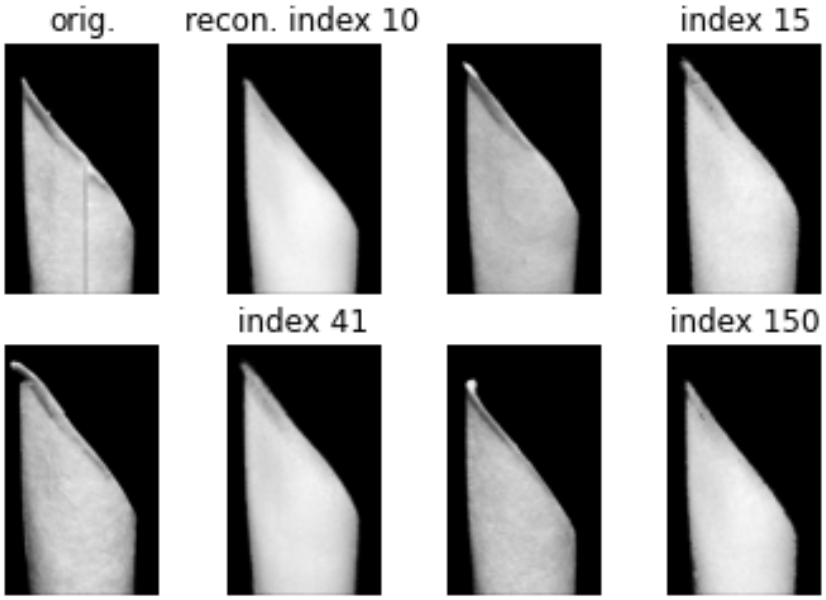

Figure 20: Autoencoder trained on side images where contrast has been increased, reconstructing side images. The outline of the straw is reconstructed well, with the exception of the inward bent tip in the lower right. The vertical line in the upper left straw is not reconstructed. The creases under the cut are for the most part reconstructed, but at a much milder rate than in the original images.

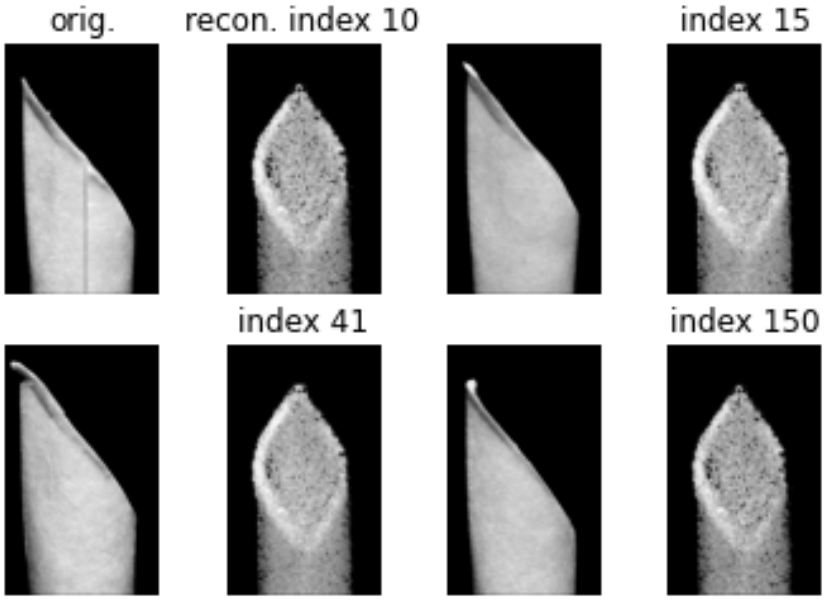

Figure 21: Autoencoder trained on front images where contrast has been increased, reconstructing side images. The reconstructed images all look like images of straws taken from the front.

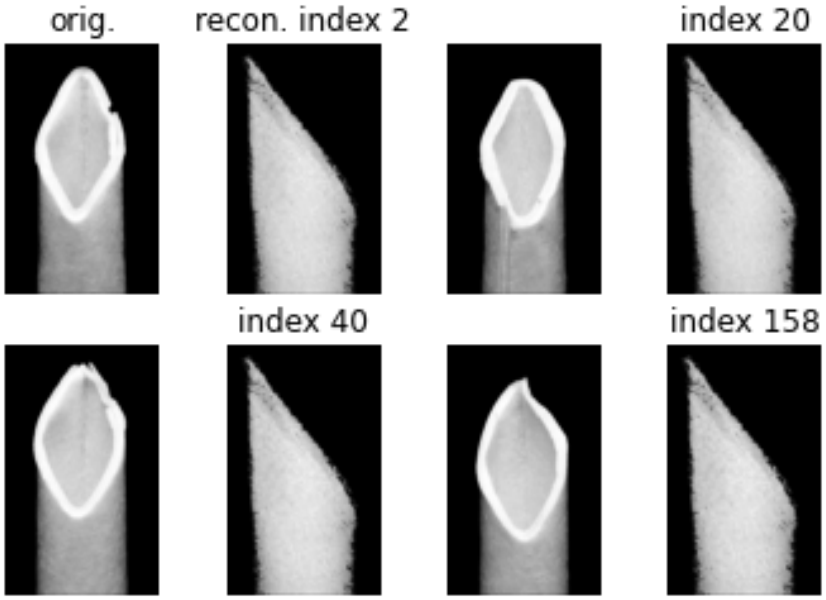

Figure 22: Autoencoder trained on side images where contrast has been increased, reconstructing front images. The reconstructed images all look like images of straws taken from the side.

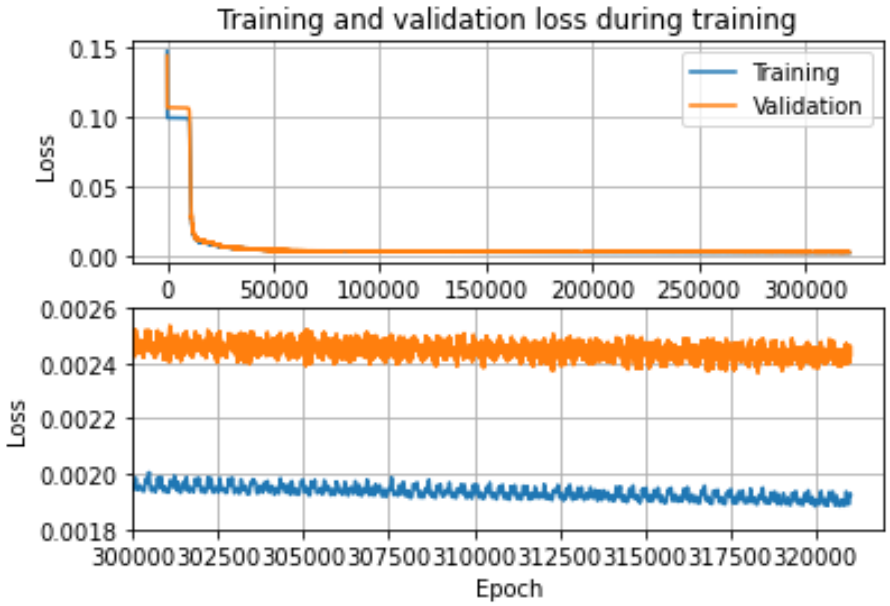

Figure 23: Training and validation loss history, for the encoder trained on side images with increased contrast. Top figure shows the entire history, from epoch 0 to epoch 322'000, whereas the bottom figure shows history from epoch 300'000 to 322'000. Notice that the overall trend of the loss is still decreasing over the last 22'000 epochs.

In figure 21 you find examples of where the autoencoder trained on front images reconstruct side images, and vice versa in figure 22. The same errors as the bench-

mark autoencoders, figure 12 and 13 can be seen here: the reconstructed images look like the straws the autoencoder has been trained on, not what it has been asked to reconstruct.

Figure 23 plots the training and validation loss history, for autoencoder trained on increased contrast side images. See figure 24 for front images. These models took approximately 14 hours each to train.

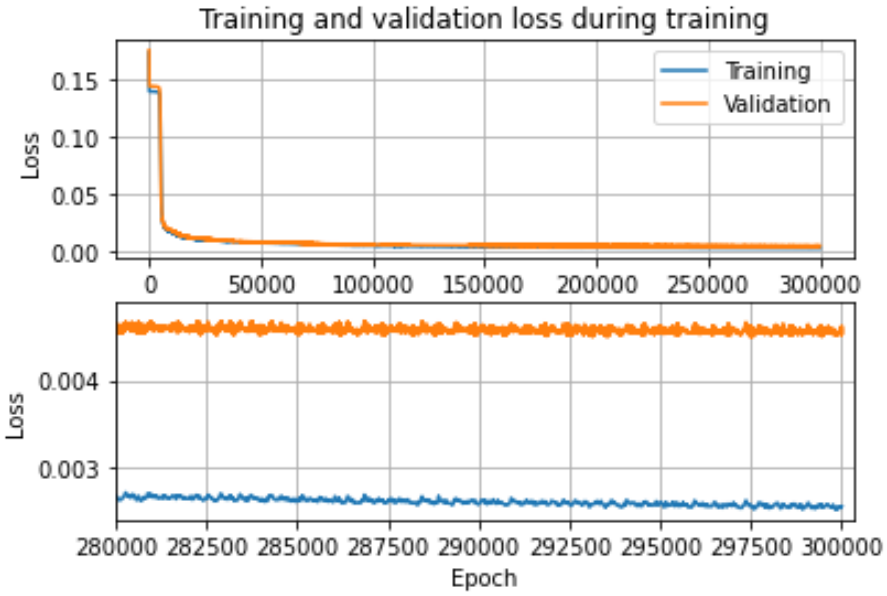

Figure 24: Training and validation loss history, for the encoder trained on front images with increased contrast. Upper figure shows training from start to finish, and bottom shows only the last 20'000 epochs. Validation loss and training loss are decreasing over the last 20'000 epochs.

3.3.3 Mixing in random images in training

3.3.3.1 Method description

If the only thing the autoencoder sees are images very similar to each other, it is favorable for the autoencoder to reconstruct very similar images, independently of what the input is. This is called *bias*. If the network is shown something it is not trained on (like a horse, when it is trained on images of straws), the output should not ideally look like anything special, since a horse is not familiar to the network. If the network truly has figured out the important features of a straw, it should also have a good understanding that a horse is not a straw, and should therefore not be reconstructed as such. What is required from the network is not to be able to reconstruct a horse into a horse when it has been trained on straws, but rather to not reconstruct a horse into a straw.

A way of reducing bias, is by introducing random images into the training. This is done by having the autoencoder train on all 160 images of straws taken from the side, together with the 160 images of straws taken from the front, together with 10% extra images picturing something else. This forces the network to learn what a straw actually is, and how to differentiate it from a cat, a horse, or a tire for example.

If the reduction of bias has been successful, every image not depicting a straw, will not resemble a straw at all after reconstruction.

The network has the same architecture, loss function, learning rate and maximum epochs as described in section 3.3.1.1.

3.3.3.2 Results

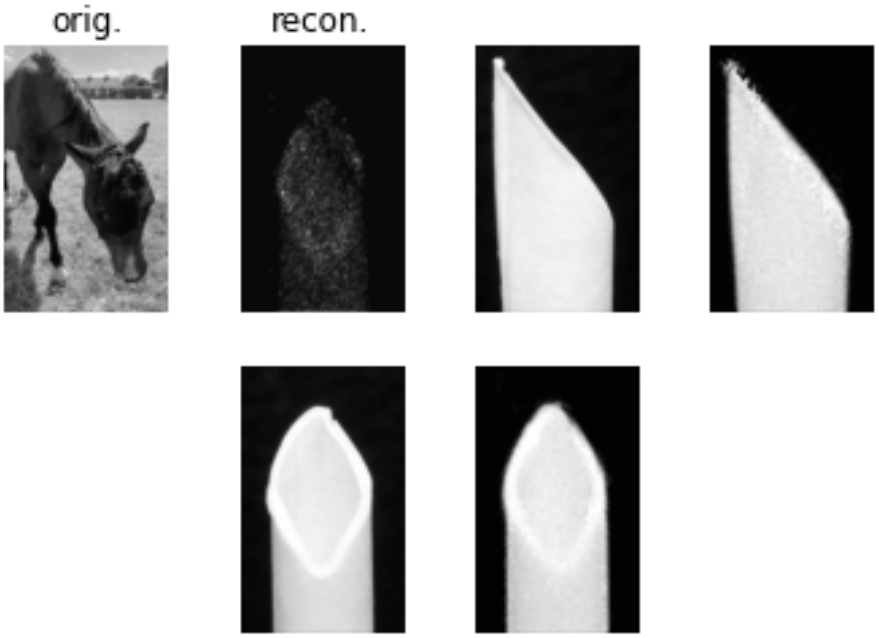

Figure 25: Reconstructions from the autoencoder with mixed images. Upper left is an example of an image of something other than a straw. Upper right is an image of a straw from the side, and below is from the front. Notice that the image of the horse looks partly like a straw, but not very clearly. The actual images of straws are reconstructed fairly well, but the details in the straws are not reconstructed at all.

Examples of reconstructed images can be seen in figure 25, where you can see reconstructions of images of straws, as well as reconstruction of an image of a horse. Notice that the reconstruction of the horse do not look like a straw.

Training history for the autoencoder trained with images of front, side and random things can be found in figure 26. The validation loss has a global minimum around epoch 75'000.

Figure 26: History of training and validation loss for autoencoder trained with mixed images. Notice specifically the global minimum for *validation* data set around epoch 75'000.

3.3.4 Combining increased contrast with random images

3.3.4.1 Method description

For the final autoencoder, the results from the previous two sets of autoencoders were used: the images were increased in contrast, and there were 10% random images added to the data set. The data set used were both data set 1, and data set 2.

The network architecture, loss function, learning rate and maximum epochs were the same as described in section 3.3.1.1.

3.3.4.2 Results

The final autoencoder reconstructing front images can be found in figure 27, side images in figure 28, and random images in figure 29. Training history can be found in figure 30. Training of this network took close to three days.

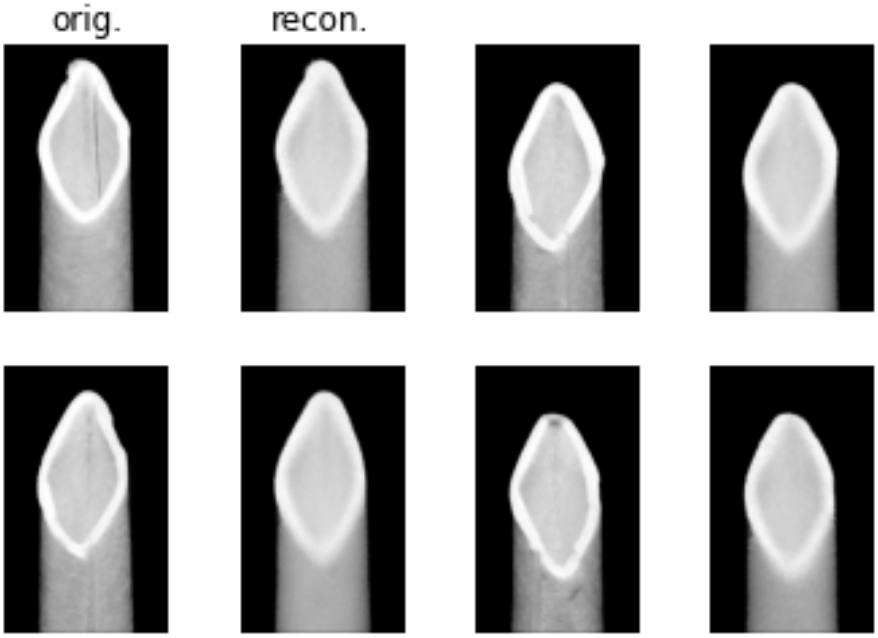

Figure 27: Autoencoder after improvements, reconstructing front images. Notice that the reconstruction of the upper left straw has captured the almost round tip. The outline of the straws are reconstructed very well. There is contrast between the inner and outer wall of the straw, but details such as the seam in the upper left straw is not reconstructed.

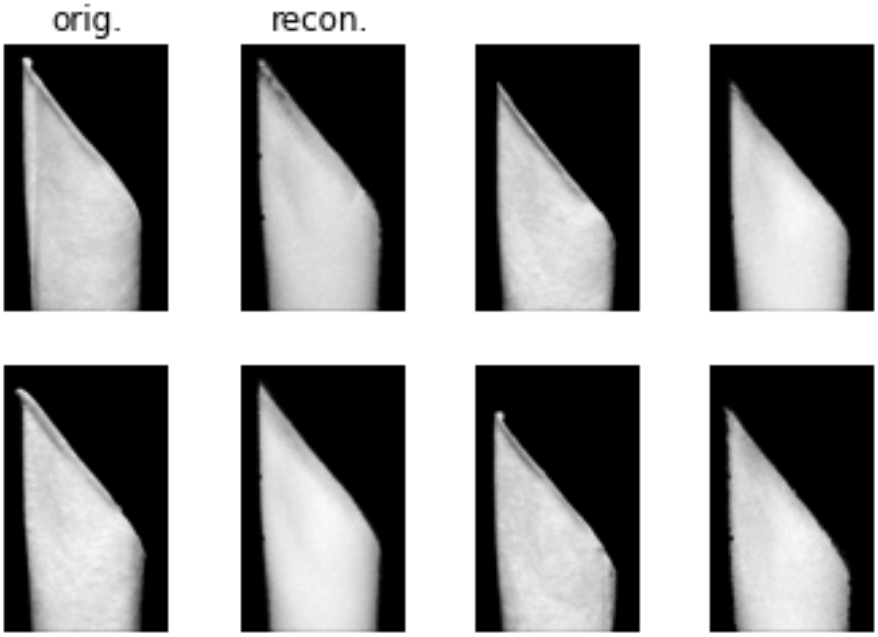

Figure 28: Autoencoder after improvements, reconstructing side images. Notice that some of the reconstructed straws have gotten indents on the sides, which the original straws do not. Some details in the straws can be seen in the reconstructions, but they are very damped compared to the originals.

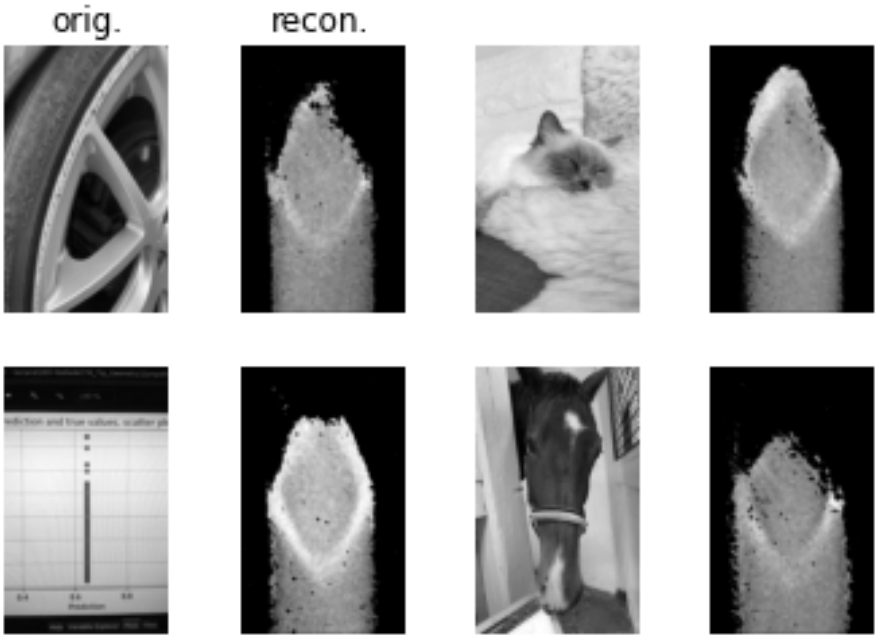

Figure 29: Autoencoder after improvements, reconstructing random images. Notice that each of the reconstructions look similar to a straw from the front.

Figure 30: Training history for the final autoencoder. Training was continued for a total of 300'000 epochs, but a global minimum in validation loss was found at about epoch 237'500.

As can be seen in figure 27, the reconstructions of front images capture the outer shape of the straw very well. The rounded tip on the upper left straw is reconstructed, and the ovality of the lower left straw is also reconstructed. The ellipse

that the cut creates is quite clear, but details such as the seam on the inside of the upper left straw is lost during reconstruction.

For the reconstructions of side images, figure 28, there are indentations in the side of the reconstructed straws, that are not present in the original images. The outward bent tip of the straw on the lower left is not reconstructed, and not the inward bent tip on the straw on the lower right. Some creases under the cut, mainly in the upper left straw, are reconstructed, but the vast majority of details are lost.

The reconstructions of random images, figure 29, show quite a bit resemblance of front images.

For the training of the model, see figure 30, training was continued until a minimum for validation loss was found, at about epoch 237'500.

3.4 Predicting opening force using *code*

The main object of the analysis using autoencoders was to draw conclusions about if there is enough information in the images to connect an image of a straw to the force required to perforate a package, using the straw depicted in the photograph.

Data set 1 and data set 2 together hold 317 straws, which in turn are depicted in 634 images. Each image is via the autoencoder turned into an array of 20 values. Each straw therefore has in turn 40 values. The 40 values for all 317 straws were gathered, and tested in three separate tests, as described below. For a visual representation of data division between tests, see figure 31. Refer to section 3.2 for a description of the difference between MD and CD.

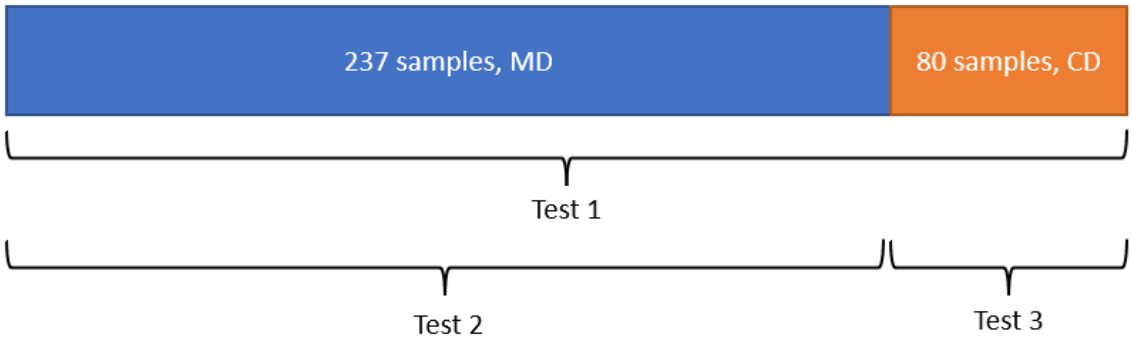

Figure 31: Visualization of what data is used for each of the three tests. Notice that the sample size for test 3 is smaller, than the sample sizes for test 1 and test 2.

Test 1: The purpose of Test 1 is to deduce if the information from the autoencoder can be used to predict the opening force, accurately, using both MD and CD measured straws.

Test 2: The purpose of Test 2 is to deduce if MD is a better measurement to predict on, compared to CD. There are 237 straws that are measured in MD.

Test 3: The purpose of Test 3 is to deduce if CD is a better measurement to predict on, compared to MD. There are 80 straws measured in CD.

Not only will these tests tell about the presence of information in the images, it will also compare the measurements MD and CD to one another.

3.4.1 Test 1 (MD and CD)

3.4.1.1 Method description

A grid parameter search to find the optimal settings for Random forest regressor for test 1 was conducted. Maximum depth was allowed five values between 3 and 20, linearly distributed. Number of trees in the forest was five values between 1'000 and 100'000, linearly distributed.

Cross-validation was done five times per model. All available data was used for training and testing, in a scheme pictured in figure 32. The data separated into 80% training, and 20% testing five times, and each of the models was given a test score, on how they performed on their test data. No extra batch of final testing data was put aside from the available data, to test all models on.

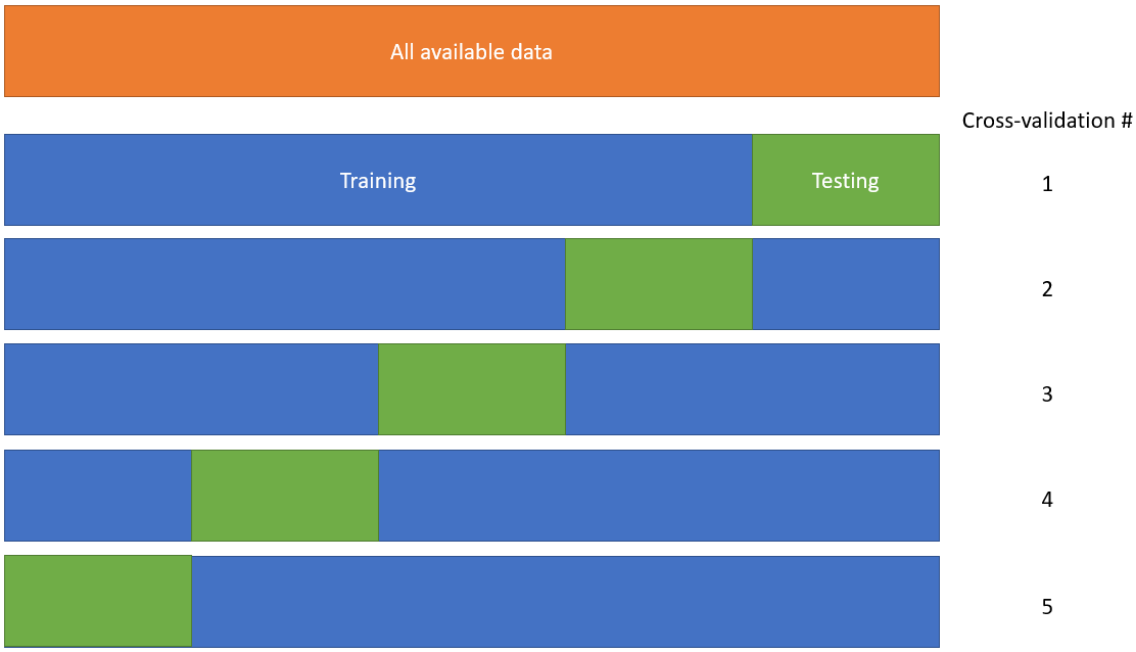

Figure 32: Figure showing the sampling of the data for the grid parameter search. All available data for each test was used for both training and testing, in a scheme pictured in this figure. Cross-validation was used five times, meaning that 20% of the data was put aside five times over for each of the models. Blue is the data for training, and green is the data for testing.

3.4.1.2 Results

Results from the grid parameter search for a Random forest regressor to test 1 can be found in figure 33. The set of models with the best mean test score was a model with maximum depth of 7, and 1000 estimators. These settings had a mean test score of 10.08% over the five cross-validations, and a training score of 80.12%. The five individual test scores were 6.45%, 14.13%, 8.71%, 20.46% and 0.67%, giving a standard deviation of 7.55%.

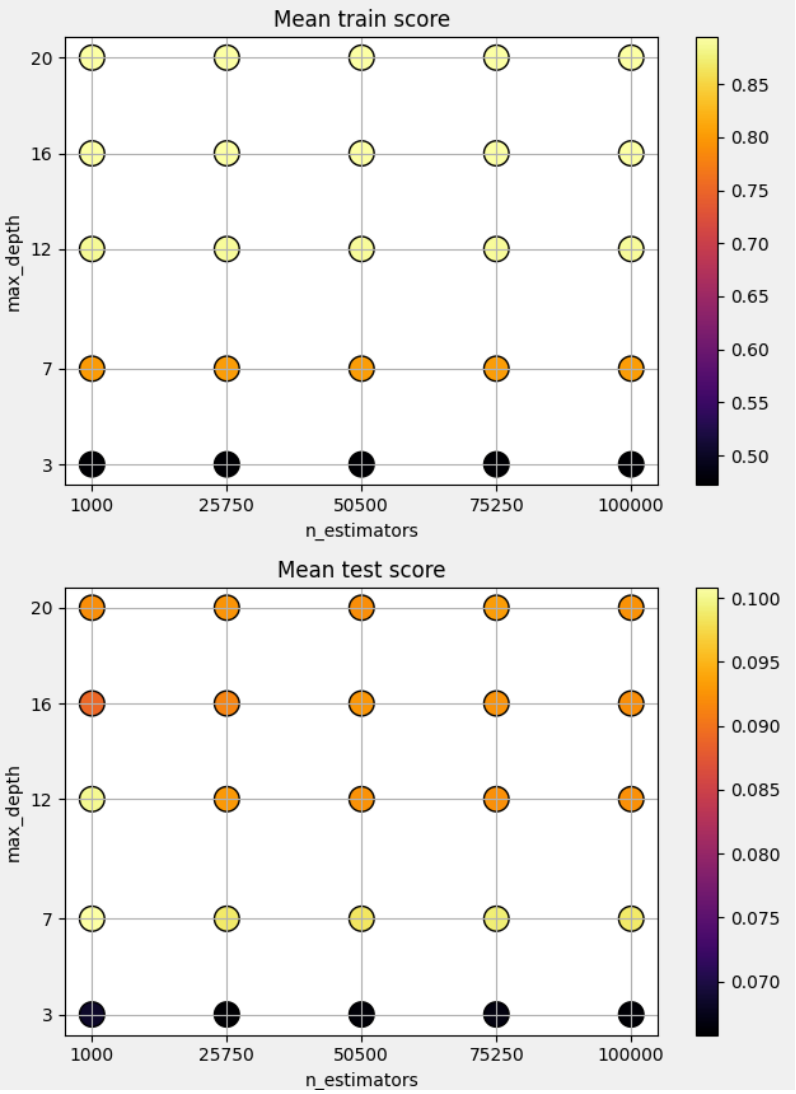

Figure 33: Plot showing the result from the grid parameter search for test 1. The upper plot shows the mean score for models during training, and the bottom plot shows the mean score for models during testing. The best model settings gave a mean test score of 10.08%.

3.4.2 Test 2 (only MD)

3.4.2.1 Method description

The parameters tested for this model was the same as in the grid parameter search for test 1, see section 3.4.1.1. The sampling for cross-validation is also the same as in mentioned section.

3.4.2.2 Results

Results from the grid parameter search for a Random forest regressor to test 2 can be found in figure 34. The best performing models were models with maximum depth of 7, and 75'250 trees in the forest. These models had a mean test score

of -0.0635, and a mean training score of 81.56%. The five individual models in the cross-validation had test scores of -0.1801, -0.0491, 9.16%, -0.1698 and -0.01, yielding a standard deviation of 11.4%.

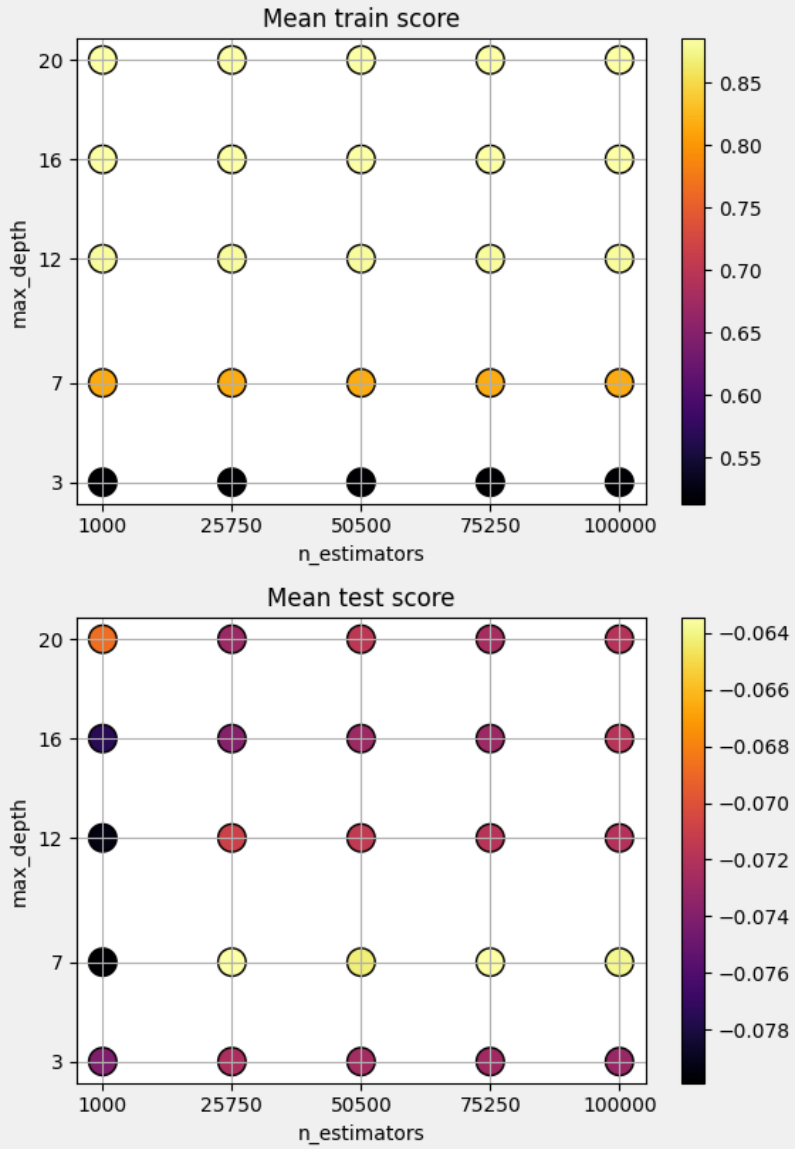

Figure 34: Scatter plots visualising the results from the grid parameter search for test 2. Upper plot shows the mean score for models during training, and bottom plot shows the mean score for models during testing. The best model settings gave a mean test score of -0.0635.

3.4.3 Test 3 (only CD)

3.4.3.1 Method description

The parameters tested for this model was the same as in the grid parameter search for test 1, see section 3.4.1.1. The sampling for cross-validation is also the same as in mentioned section.

3.4.3.2 Results

Results from the grid parameter search for a Random forest regressor to test 3 can be found in figure 35. The best model is given by a model of max depth 16, and 25'750 trees in the forest, as this is the model with the highest mean score during testing, -0.2909, and 90.58% for training. The five models had a testing score of -0.5104, -0.4138, -0.6497, 19.12% and -0.0718, which yields a standard deviation of 30.74%.

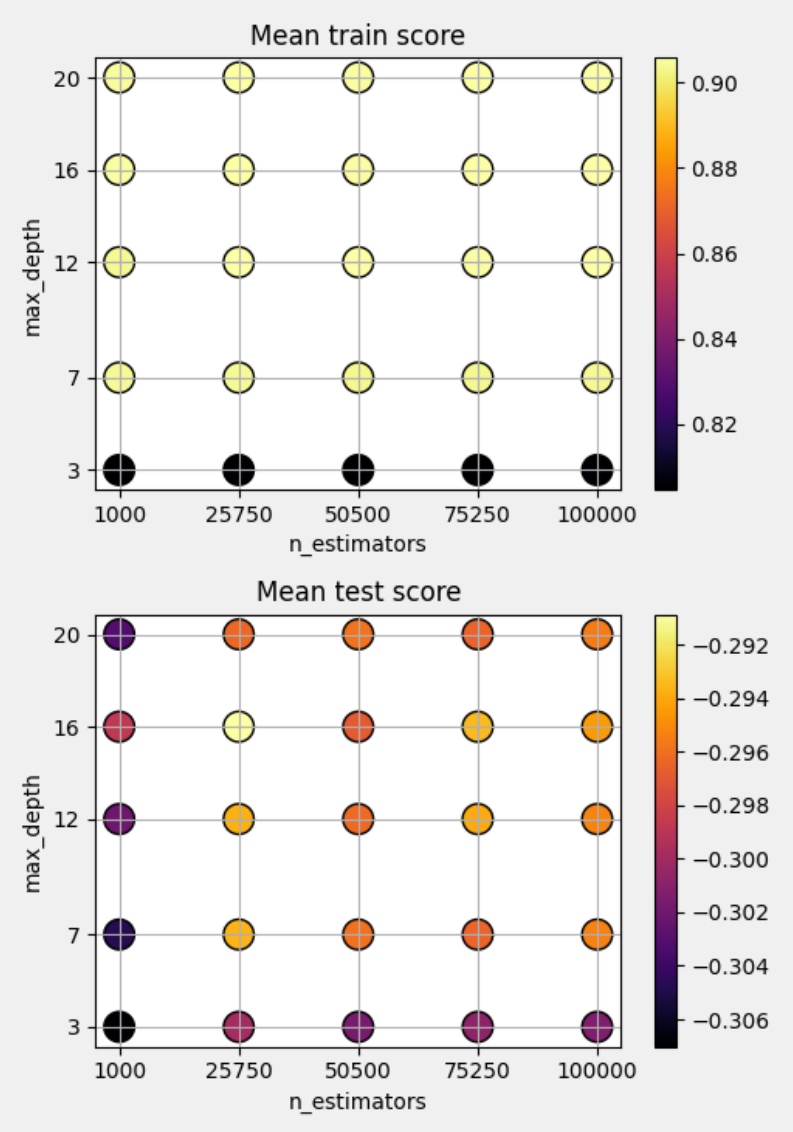

Figure 35: Scatter plots showing the results from the grid parameter search for a Random forest regressor for test 3. Upper plot shows the mean score for models during training, and bottom plot shows the mean score for models during testing. The best model settings gave a mean test score of -0.2909.

3.4.4 Summary of results, predicting opening force using *code*

A table format summary of the results presented in the sections above, can be found in table 3.

Table 3: Table summarizing the resulting values for the coefficient of determination for the three tests. The standard deviation refers to the standard deviation for the test score, over the five cross-validations. The best performing test is test 1.

Test number	**Mean r^2 training (%)**	**Mean r^2 testing (%)**	**StDev (%)**
1	80.12	10.08	7.55
2	81.56	-6.35	11.4
3	90.58	-29.09	30.74

3.5 Image processing and Random forest regressor

The image processing is done on grey scale of the images, in a software called *Sympathy for Data*, which is an open-source platform for building and running data analytics and data science applications [8]. Sympathy for Data is a Python based program, and the image processing algorithms available in Sympathy for Data, are implementations of *Scikit-image*, which is a collection of algorithms for image processing in Python [7].

Both images of each straw (front and side) are analysed separately, and the information from the two images are then merged together, to form a set of parameters for each straw. The parameters gathered using image processing that will be used in the following sections, can be found described in words in table 4.

The methods used to extract these parameters were already implemented and used in previous projects with paper straws. Upon me starting this thesis, some fine-tuning and changing of methods were conducted, but the vast majority of the methods were already in place before the start of this master thesis. The methods will not be described in detail, as the methods themselves are not a central part of this project. Please note that the exact details of these parameters are not important for the understanding of the results. What is important to bring forward is that these parameters cover what one would think are the important features of a paper straw, in relation to its perforation ability.

The parameters that show difference in straw width at different heights relative to the tip, is created to aid the Random forest regressor, as it cannot transform parameters, as mentioned in its theory section. The sharpness of the tip is related to the difference in width at different heights, and that is what these parameters are meant to capture.

Table 4: Short description of the many parameters extracted using image processing.

Parameter name	**View**	**Description**
Cutting_angle(deg)	Side	The angle the straw is cut. This is the angle created from the horizontal, up to the top of the cut.
StrawWidth(mm)	Both	Width from one side of straw to the other side. If measured from front, *frontview* is added to name. If measured from side, *sideview* is added to name. Added to the name is also how far down from the top the measurement is taken. If it is measured 5.0 mm from the tip, *_5.0mm_from_tip* is added to the name. Width is measured in millimeters. Straw width is measured every millimeter from the tip, down to 8 mm.
Cutting_angle_mid(deg)	Side	The angle the straw is cut. This is the angle created from the horizontal to between 25% of width and 75% of width of the straw.
S_delta_y(mm)	Side	Describes the shape of the cut of the straw.
Cutting_angle_0-25%(deg)	Side	The angle the straw is cut. This is the angle created from the horizontal to 25% of the width, measured from the tip.
Cutting_angle_0-10%(deg)	Side	The angle the straw is cut. This is the angle created from the horizontal to 10% of the width, measured from the tip.
diff_StrawWidth(mm)	Both	Difference between the width at two points. If measured from front, *front* is added to name. If measured from side, *side* is added to name. Added to the name is also how far down from the tip the two measurements are taken.
TM_crushed_tip_intensity	Side	Template matching, a measure of how well a template of a crushed tip fits into the sample picture.
x_seam(mm)	Front	Horizontal position of seam, if any, in relation to the tip of the straw. See figure 36.
y_seam(mm)	Front	Vertical position of seam, if any, in relation to the tip of the straw. See figure 36.
Diameter_tip_model(mm)	Front	Approximated value for a theoretical circle in the top 1 mm of the straw.

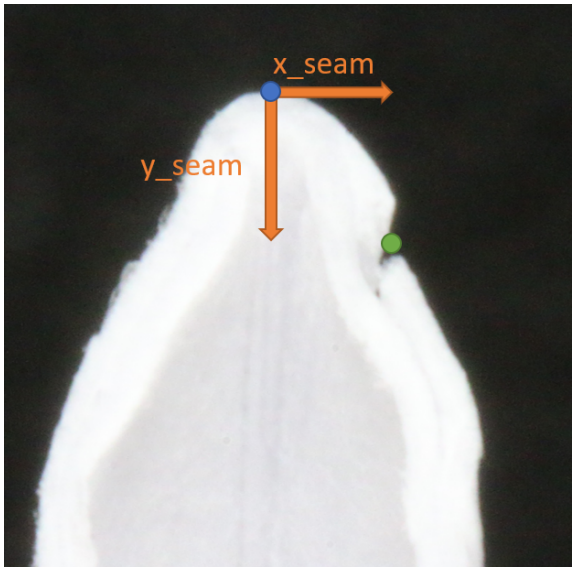

Figure 36: Definition of the parameters *x_seam* and *y_seam*. Blue dot marks the tip of the straw, and green dot marks the seam on the straw.

3.5.1 Method description

The parameters tested for this model was the same as in the grid parameter search for test 1, see section 3.4.1.1. The sampling for cross-validation is also the same as in mentioned section.

3.5.2 Results

Best performing set of models had a maximum depth of 16, and 1000 estimators. Those models had a mean test score of 9.78%. The five individual models had test scores of 19.13%, 8.82%, 23.66%, -0.0459 and 1.9%, yielding a standard deviation of 11.7%. A figure showing the result from the grid parameter search can be found in figure 37.

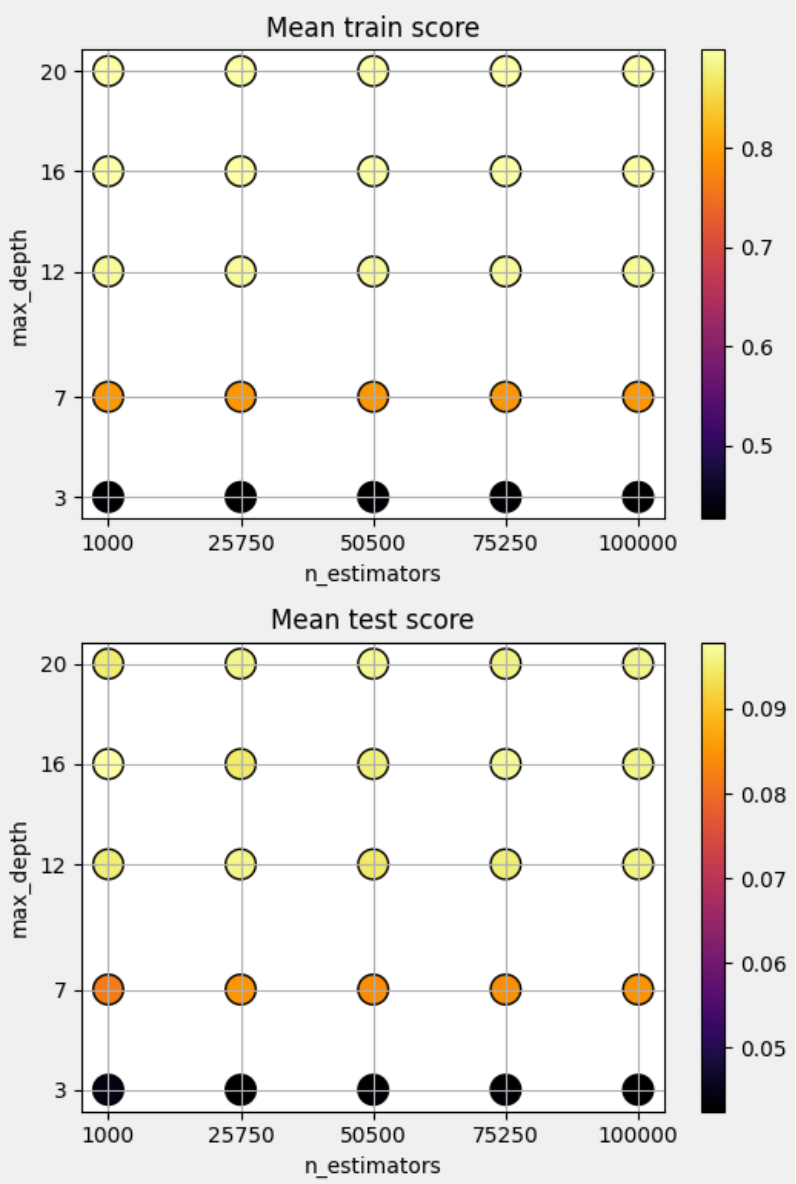

Figure 37: Results from the grid parameter search for the features extracted using image processing. Upper plot shows the mean score for models during training, and bottom plot shows the mean score for models during testing. The best performing set of models had a mean test score of 9.78%.

4 Gage R&R study

A prerequisite for a study like the one conducted in this thesis, is that the images used are consistent with the process it is showing. Although a Gage R&R study is not what the thesis is about, a Gage R&R study was conducted at the start of the thesis, in order to rule out that the variance found is due to reproducibility and repeatability issues.

I was one of the participants, and I was also the one to do the image analysis of the straws, and conclude the results in figures. The two other operands were colleges of mine at Tetra Pak.

4.1 Method description

Three operands where asked to follow the instructions given, to set up the camera, and then take photos from the side, and front, of ten straws. All operands photographed the same ten straws. This was done twice for each operand, with at least one day between sessions. A description of the camera setup can be found in appendix, section A.

The photographs of the straws were analysed using image processing, extracting the same parameters that will be of importance in this thesis. The extracted attributes were compared to one another, to find what standard deviation was introduced from the different operands, and if that is of size comparable to the process variance. No gage R&R was conducted on the opening force measurements.

The parameters that were examined further were *Cutting angle* and *difference in straw width in front view, 3 mm from tip VS w mm from tip.*

4.2 Results

The two parameters used to evaluate the variance introduced by the operand and method of photographing images, will be presented using images showing how the values for the parameter varied for the ten straws depending on operand and day, as well as the size of the variance. The two parameters are *Cutting angle* and *difference in straw width in front view, 3 mm from tip VS w mm from tip.*

The two main types of graphs that will be used will tell about the reproducibility for each part and operand in the study, and the repeatability for each part and operand. Part refers here to straw, and operand is the person setting up the camera and taking the pictures.

The reproducibility graph plots the parameter against the parts. The different colored markers correspond to the three different operands.

The repeatability graphs consist of two separate plots. The plot to the left shows the values per operator, and how the ranges between the operators vary. The right plot shows the values per part, instead of per operator.

4.2.1 Cutting angle

Graphs showing reproducibility for the parameter *Cutting angle (deg)* can be found in figure 38. Repeatability figures can be found in figure 39.

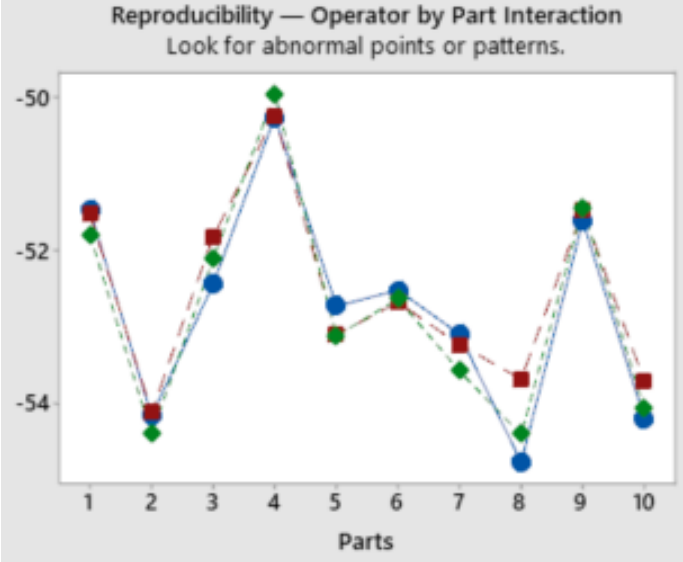

Figure 38: Graph showing reproducibility per part, for the parameter *Cutting angle (deg)*. Notice that the vast majority of variance in the values, are between parts and not operands.

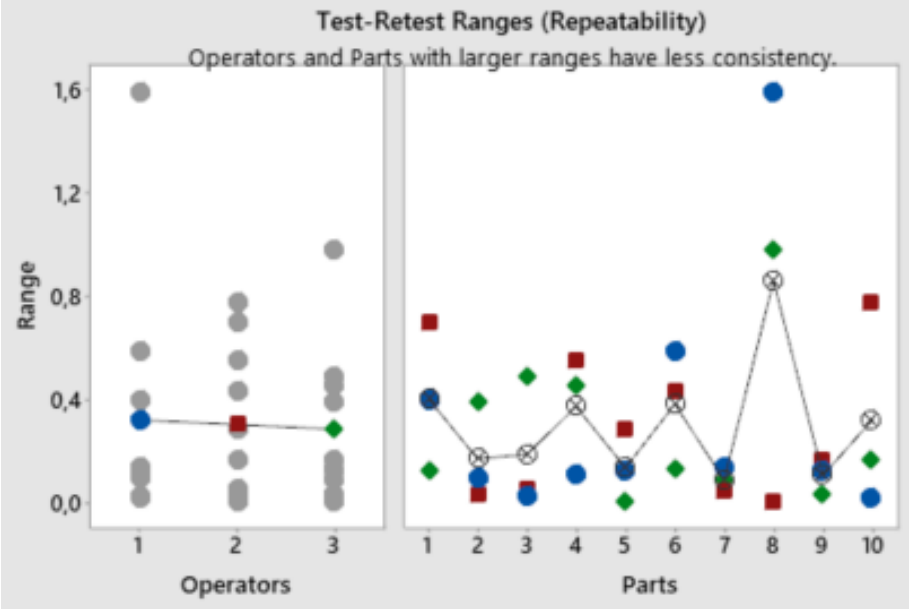

Figure 39: Graph showing repeatability per part, for the parameter *Cutting angle (deg)*. Notice how there is a big difference in measured values for straw number 8.

As can be seen in figure 38, the variance from part to part is much bigger than the variance between operators. As can be seen in figure 39, the ranges look very much alike for all 10 parts, with the exception of part 8 which has an abnormal measured value for *cutting angle(deg)*.

4.2.2 Straw width, front view, difference 3 mm from tip VS 2 mm from tip

Graph for reproducibility can be found in figure 40, and repeatability in figure 41.

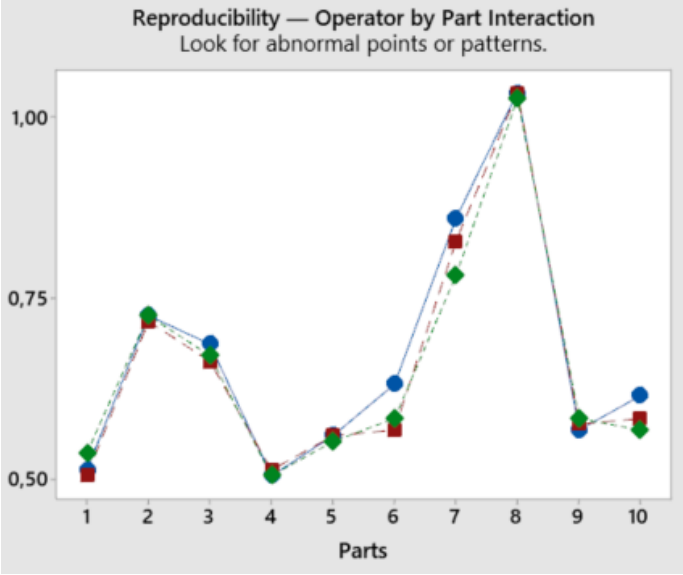

Figure 40: Graph showing reproducibility per part, for the parameter *Straw width, front view, difference 3 mm from tip VS 2 mm from tip.* The variance between samples is much greater than the variance for each sample.

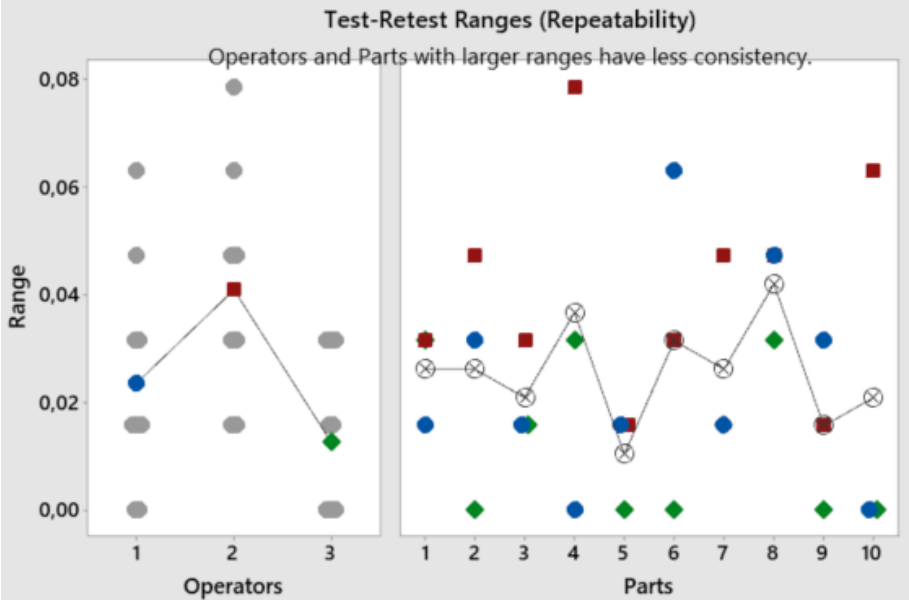

Figure 41: Graph showing repeatability per part, for the parameter *Straw width, front view, difference 3 mm from tip VS 2 mm from tip.* No part or no operator seem to be under-performing.

As can be seen in figure 40, the variance between samples is far bigger than the

variance between the three operators. In figure 41, the ranges for the different parts and operators look very alike.

4.3 Discussion

The results from the Gage R&R study are satisfactory. In both parameters, the vast majority of the variation is shown to come from the process itself, and not the measurements. This is satisfactory since the measurements conducted in this thesis would not be reliable, if the biggest difference from one measurement to another is not the difference in paper straw, but who took the pictures, and what day the pictures were taken.

5 Discussion

This section will be divided into the following subcategories:

1. Autoencoders
2. Predicting opening force using *code*
3. Comparing approaches: image processing and autoencoder

The subsection on autoencoders covers the improvements that were made of the performance of the autoencoder. The *Predicting opening force using code* discusses the results from the three tests where the output from the autoencoder is used. *Comparing approaches: image processing and autoencoder* compare the results between how good a model using input from image processing is, compared to a model using input from an autoencoder.

5.1 Autoencoders

This section will discuss the measures that were taken to improve the performance of the autoencoder.

5.1.1 Reconstruction of images without any improving measures

What can easily be seen from the figures where the autoencoder trained on front images has to reconstruct front images, figure 10, is that the network is rather good at capturing the most clear parts of the shape of the straw. Details such as the seam for image with index 2 is lost, as well as the clearly non-symmetrical tip for the straw in image index 158. The details on the straw are however to a great extent lost. The ellipse that the cut of the straw makes is barely visible, and no indents or similar in the straws are recreated. The reasoning for this is (as previously mentioned in the method section) that the loss function used is a pixel by pixel comparison between the input and the reconstructed image. The differences in pixel values between background (pixel values close to zero), and pixel values for straw (values close to one), are of a much greater importance for the model to decrease the loss, than a difference in shade of white is. This is the reason why the contrast in the images are increased as a next step, and the network is retrained. The increase in contrast is to punish the model more for not learning differences in details on the straw.

What can be seen from the figures where the autoencoder trained on front images is asked to reconstruct side images, figure 12, is that the reconstructed images look like front images of straws. Not only do they look like front images, they also look pretty much exactly the same. This reveals a big bias in the network. The network has not learned what is important about a front image of a straw, instead it overfits to the data it was trained on. To combat this, images not depicting straws are added to the training data set, as described in the method section, and further discussed in a section below.

Much like what can be seen from the autoencoder trained on front images, can be seen from the autoencoder trained on side images. Where the autoencoder is asked

to reconstruct side images, figure 11, the background is fairly well captured, where as the details in the image are lost. Any creases or indents in the straw are lost between original and reconstructed image. Where the side-trained autoencoder is asked to reconstruct front images, figure 13, the reconstructed images look like very similar side images. There is also a big bias in this network.

Regarding the training of the networks, see figure 14 for the autoencoder trained on front images, and figure 15 for the autoencoder trained on side images, the over all look is the same. What I would like to draw extra attention to are the last 20'000 epochs of training. The loss for both training data set and validation data set is decreasing. This would suggest that the networks are not necessarily done training - further training could reduce both losses without overfitting being an issue. Training was stopped at about 300'000 epochs due to the extended amount of time required to train these networks. If more computer power was available, or more time, the models could have been trained for a longer time, with possible greater reward.

5.1.2 Reconstruction of images after contrast increase

By looking at figure 20, where the autoencoder trained on side images with increased contrast, is asked to reconstruct side images, and comparing it to the corresponding reconstructions without increased contrast, figure 11, there are far more details in the figure for increased contrast, compared to the standard images. The outer shape of the straw is in most aspects well represented (same as when no contrast increase had been performed), but the major difference is in the presence of details on the straw. Shadows created by creases in the paper close to the cut are visible, while shadows created by the seam (index 10) are not reconstructed.

The corresponding conclusions can be drawn where the autoencoder trained on front images are reconstructing front images, see figure 19. The ellipse resulting from the cut is far clearer reconstructed here, than in the version without contrast increase, see figure 10. The outline of the straw is quite well reconstructed. The inward bend close to the tip in the lower right straw is somewhat reconstructed, and the splitting of the paper in the lower left straw is reconstructed a bit. The crease on the inside of the paper in the upper right straw is not present in the reconstruction, and neither is the seam in the upper right straw. The increase in contrast has given the results that were asked for, but not to the extent that one would hope for. Some details are captured better, some are still not visible.

The figures showing autoencoder trained on front images reconstructing side images (figure 21 and vice versa (figure 22) show the same bias as the images without contrast increase. This is not strange, as the contrast increase is not meant to decrease bias.

Similar to with the autoencoders trained on the original images, the training loss and validation loss is still decreasing over the last 20'000 epochs of training, see figure 24 for front, and figure 23 for side. Due to limitations in the time available, these networks could have been further trained but were not.

5.1.3 Reconstruction of images after mixing in random images

Studying the graph for validation data set during training, figure 26, the network is starting to overfit to training data somewhere after epoch 75'000. Since the intermediate models are saved, a model trained to only 90'000 epochs is used to make the figures found in the result section.

From figure 25, it shows that this network suffers from the same problem that the networks trained on original images (without any contrast increase) do, there is a lack of details on the straw, but they replicate the shape of the straw pretty well, both from the front and from the side. This is not strange, as there is no difference between the images of the straws used to train the networks of original images, and the images of straws for this model.

However, what is also seen from figure 25 is that images of things that are not straws, do not reconstruct into straws. There is resemblance of a straw, but it is not as clearly a straw, compared to networks trained without random images, see figure 22 and figure 13, for example. This is what was intended with the inclusion of random images, concluding that this was indeed a good method. Including a bigger fraction of random images could further reduce the bias.

The clear downside of this network is that the network never reached the level of details in the straw reconstructions, as models trained on the original images managed. This is however a problem not easily managed with the sample size at hand. One way to improve the reconstruction would be to have more images, which was not an option for this thesis.

5.1.4 Reconstruction of images after performance improvements

Training of the network lasted for about three days, and spanned 300'000 epochs. A global minimum in validation loss was found around epoch 237'500, suggesting overtraining had begun after that. The network used is therefore the network trained to 237'500 epochs. The entire training history up to epoch 300'000 is unfortunately lost due to an overwriting of the model during capturing of images for the report.

The reconstruction of front images, figure 27, shows some details in the straws being captured to some degree, like the bend in the upper left side, of the straw to the upper left. Other details in the straw, such as the seams in the lower left straw, is completely left out.

In the reconstruction of side images, figure 28, the network has managed to capture some of the shadowing around the tip, as can be seen in the lower left straw. There are however some straws (upper left and lower right) that have indents in the left side of the straw in the reconstructed images. These are not present in the original images. The bend of the tip in the lower left image has not been captured in the reconstructed image.

The reconstructions of the random images, see figure 29, show quite clear signs of bias. Not only is there a quite clear straw in each of the images, but there is also quite clearly a front straw in all reconstructions. The reason for this is unclear to me, as there is an equal amount of front images as there are straw images in the data set presented to the model. One reason might be that when the model separated the images into training and validation, it might have accidentally used

more front images than side images to train on. This might also be the reason why the reconstructions look similar to straws, there might have been to few random images in the training data set. If more time would have been available, I would have made sure that the training data set was balanced, and not left to chance.

5.2 Predicting opening force using *code*

Table 3 summarises the best mean training and test score for the three tests. The best test score was achieved in test 1, where both MD and CD measurements were used. Second best was for test 2 where only MD was used, and worst mean test score was found for test 3, where only CD was used. The same order of performance is accurate for the standard deviations for the test score, with test 1 having the lowest standard deviation, and test 3 uncontested having the biggest standard deviation.

The mean test score for test 1, 10.08%, is not good, but it does show that there is some valuable information in the images. The performances for test 2 and 3 are however incredibly bad: -6.35% with a standard deviation of 11.4% for test 2, and -29.09% with a standard deviation of 30.74%. The reason for the big difference in performances is not super clear. One possible reason might be that the five iterations of cross-validation was not enough. Maybe ten iterations would have been better. No good reasoning was behind the initial choice of five iterations, so it might as well had been ten. Doing ten iterations would have taken twice as long, which would have been a little too long given the time available for this analysis.

Another possibility might be that the difference between MD and CD is very big, and easy to model, but the differences between samples within MD and within CD are harder to model correctly. That would explain the big difference in performance.

Worth noting is also that there was only 80 samples in test 3, which could influence the results badly.

5.3 Comparing approaches: image processing and autoencoder

The best set of models for predicting opening force given parameters gathered from image processing, had a mean test score of 9.78%. The corresponding autoencoder-generated values, were 10.08%. There is a slight difference in these values, preferring the autoencoder-generated values. Not only is the autoencoder approach better in mean test score, but it also has a lower standard deviation: 7.55% for the autoencoder, compared to 11.7% for the image processing approach.

In the case of image processing, and how well the extracted parameters can explain the variation in measured force, a lot of knowledge is needed from the engineer deciding the parameters that are to be extracted. Since the parameters decided to extract in this case did not fully manage to explain the relationship between straw and required force, a conclusion one might want to draw, is that there is not enough information about the straw in the image, to be able to deduce what force would be required if that straw was to perforate a package. Drawing that conclusion at this stage would be wrong, since part of the model is a person, or a group of people, doing educated guesses on what would be important to measure in the image. The

group might be very right in all their guesses, but misses one vital parameter that could be extracted. Or the group catches all information that exist in the image, but there is vital information that is missing in the images. An image of a straw can not tell the humidity in the paper of the straw, it can not tell the quality of the glue that glued the two paper strips together in the making of the straw.

In the case of using a deep learning method to find the important features of the straw, the human part of the process disappears, and one of the possible hinders to drawing the conclusion that not enough information is present in the images, can be removed. The twenty values for each image created by the autoencoders could be seen as features of the straw. For a human, knowing what angle the straw is cut at, and the width at certain distances from the tip, is enough to recreate a similar image of the straw at hand. The autoencoders do the same, but the important values for the autoencoder might not be what we as humans think are the important parameters. Given that the autoencoders together with the Random forest regressor fail to produce overall better models than the image processing with Random forest regressor does, may tell us that the parameters extracted using image processing are covering the vital parts of the straw, and captures the information available in the images. Another possibility is that an autoencoder with latent-space dimension of 50 would possibly produce a better model, where we then can draw the conclusion that there is more information in the images, than the image processing methods extract. The question then becomes: What are these features that we do not already understand are important for the prediction of the required force?

There are things that can affect the measured force, that can not be captured in a image of the straw. Previously mentioned are the humidity of the paper in the straw, and the glue used to produce the straw. There are also other parameters that can introduce variance in the testing of the force. For example how the barrier the paper straw perforate vary in thickness, or in stiffness. This can not possibly be seen in an image of the straw used to perforate the package. If the acquired models in this thesis could explain 100% of the variations between samples, that would suggest that the factors mentioned above do not have any influence at all over the process, which is not reasonable what so ever. In this discussion it is also worth mentioning that the two data sets that are used in this thesis were produced at two different times. Data set 1 was produced at the end of April 2022, and data set 2 at the end of June 2022. A lot of the parameters that are not visible in the images can over this period be vastly different, which lowers the predictability for the combined data set. If these two data sets were analysed separately, these differences may not introduce any variance, because these parameters, although not visible in the images, are not different between samples, in each of the data sets. For example: the glue used is maybe very important, but the glue is the same for all samples in data set 1, while the glue used for data set 2 is from another batch, with somewhat different properties.

The gage R&R study shows that the act of cutting the straws to length, marking them with numbers, placing them in the photo rig, rotating them correctly, placing focus on the correct spot and taking the photos also introduce variance to the entire process. The variance was concluded to be sufficiently small for the method to be reliable, but the process still introduces variation to the models trained in this thesis.

The choice to use Random forest regressors as the machine learning algorithm was originally built upon the reasoning that the results about what parameters are the most important would be as easily interpreted as possible. As is mentioned in the introduction, this was the initial question for the thesis, which in time turned into the present question. The reason to use Random forest regressors do not hold anymore, as easy interpretation no longer is top priority. It might very well be that the problem of modelling the force is not as straight forward as the Random forest regressor would like for it to be. It might be that inverses of parameters need to be multiplied by other parameters, and that is simply not something the Random forest regressor can do. There are however many machine learning algorithms out there, that may unlock the complexity of this problem far better than the Random forest regressor.

6 Conclusion

This thesis was initiated with a gage R&R study, that showed that the method for photographing straws is good enough that the major part of the variance found, is due to variance in the process, as compared to the method itself.

An autoencoder was used for compressing the images of the straws, to later be used with a Random forest regressor, to deduce the amount of information available when using deep learning. Improvements such as increasing contrast in images and including random images not picturing straws increased the performance of the autoencoder.

The final autoencoder was used to parameterise the straws, which later was used as data for a Random forest regressor, in three different tests. These three different tests compared the performance of models using data from both MD and CD, to models using only CD, and only MD. It is very apparent that the performance of MD and CD together is by far outperforming models with only MD, or only CD. The combination of MD and CD could explain 10.08% of the variations, where as the ones for only MD or only CD had negative values for r^2.

The results from the autoencoder and Random forest regressor were compared with parameters extracted using image processing, used together with a Random forest regressor. Here it was not as clear what the better setup is, but the scale is tipping towards the autoencoder, given that it had marginally higher mean test score (10.08% compared to 9.78%), and a lower standard deviation (7.55% compared to 11.7%).

6.1 Improvements of this thesis and further work

Many things could have been done differently, or in greater detail than was possible for this thesis. Most of what is to come is already mentioned scattered about the thesis, but they will be collected and summarized here.

First regarding the autoencoders: the architecture of the autoencoder could have been changed, to see how this affects the predictability. These changes could be how many convolutional layers that should be in the networks, how deep these convolutional layers should be, how many fully connected layers should follow these convolutional layers, and how big the latent-space should be. These changes in architecture are very time consuming, but could increase the performance. The most obvious parameter for performance impact is the latent-space dimension. Comparing the most extreme case of having only one value for each image, to having 50 values per image, makes it easy to draw the conclusion that this is an important parameter. Maybe 20 isn't the best, maybe 50 would yield much better result.

Beside the change in architecture of the network, the loss function could have been changed. Measures were taken to work around the shortcomings of the MSEloss. Another option would have been to investigate other possibilities for loss functions, or even invented one specific for this implementation.

Moving on from the autoencoder, and now focusing on the grid parameter searches: a very important parameter, namely the number of cross-validations, could have been increased. In the discussion section, an increase to ten iterations was men-

tioned. This could absolutely be a better option, and give more reliable results. Another improvement would be to include more parameters in the grid parameter search. As mentioned when introducing the settings, only two parameters were included in the search, as these are the most important for performance. Including more parameters would without a doubt increase the amount of time needed for a search, but it could increase the performance of the models overall. Besides the already mentioned measures, a finer grid than what was used, could have been tested. This would also increase time spent on each search.

What would greatly help the training of the autoencoders would be more samples. 317 samples (that turn into 634 images since each sample is photographed twice) is on the lower side of what is required when working with deep learning algorithms.

One point of variance that could not possibly be captured in an image of a straw, is the variance in the properties of the barrier in the pre-punched hole. Since the force measurements were conducted on real packages, the variance of the barrier is naturally included in the measurement of the force. To remove this variance, the force measurement could be replaced with a stiffness measurement. The straw would then be pressed against a hard surface, and the stiffness of the straw would be recorded, as opposed to the force. The stiffness does not necessarily encompass what a costumer is looking for when trying to open a package with the straw, which is a clear downside for this type of measurement.

A similar setup with three different tests could have been conducted with the data from image processing. There is no good reason for not doing this, other than that time did not allow for it, and that it was not high on list of priorities. Testing the same for image processing as for the autoencoder could give a confirmation (or raise further suspicion) that the differences within MD and CD are far harder to model, than the differences between them.

The parameters from image processing could also be combined with the parameters from the autoencoder, to see if the data compliment each other, and give better results for predictability.

As was mentioned in the theory section on Decision trees and Random forest regressors, these algorithms are not able to conduct any transformations to the input data. One thing that could be examined would be how other machine learning algorithms perform on the same data, or a multilayer perceptron, that are able to transform input data.

In regards to the clustering that was conducted, other algorithms than the K-means clustering could have been examined. Different clustering methods are better fit for different sorts of data, and K-means clustering may not have been the most appropriate algorithm for the setting.

A Camera setup

Each paper straw is photographed in a side view, as well as a front view. Examples of how these images look can be found in figure 42.

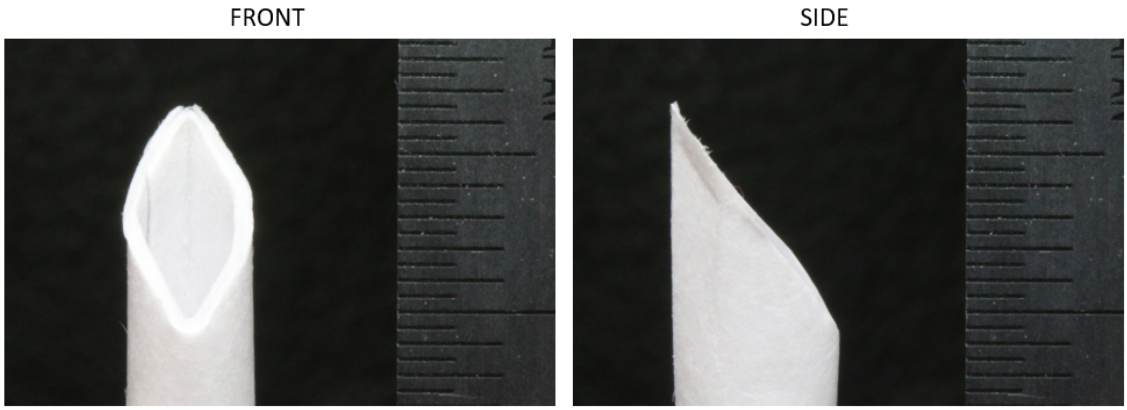

Figure 42: Examples of how the original images of a straw looks. The left image is a straw photographed from the front, and to the right is that same straw but now rotated 90 degrees, showing the cut of the straw. All images include a ruler to the right in the image.

The images of the paper straws are taken in a climate-controlled room, where the temperature is 23 degrees Celsius, and the relative humidity is 50%. This is due to that the straw tips can vary in shape depending on humidity.

The camera used was a Canon EOS 70D, or similar model. The lens is a Canon EFS 60 mm, with a Canon extension tube EF25 II.

The camera rig is specifically built for this setup, and can be found in figure 43. The rig allows the camera to be mounted at the same distance to the specimen each time the camera is set up. This makes the difference between images taken at separate times little to non.

Figure 43: Photography of the camera rig used to take photos of the paper straws.

The camera is set to manual focus, ISO of 6400, shutter speed of 1/25 s, and depth of field of f/32. These settings were found by maximizing the depth of field to make the depth of focus as big as possible, which is important in image analysis, as the out of focus of important parts of specimen makes the analysis inaccurate. Shutter speed and ISO were adjusted to find good lightning conditions. The focus is set on the ruler, which in turn makes the focus be in the middle of the specimen, since the ruler and specimen are at an equal distance to the camera. Focus is set at the ruler to the right in the image.

The specimens are cut to a length of 5.5 cm. Each straw gets a straw number/straw ID written on them, to enable backtracking from an image of a straw, to that specific straw. The specimen is mounted on the metal pin as seen in figure 43, close to the ruler. The operand is prompted to not touch the tip of the straw during handling. After all straws are cut and numbered, the process of taking the images is started. Each specimen is photographed from the front, and from the side. Note that there is nothing on the rig that ensures that the straw is rotated correctly, neither for the front images or the side images. It is subjective to the operand exactly the rotation for the straw to be *front* and *side*.

The photo number, straw number, and if the image is a side image or front image is written in a photo log, and put together with the images in a folder on a shared server space.

B Force measurement

Before measuring the force, the paper straws are cut to just below the corrugated bend, so only the straight part of the straw down to the tip is used. The straws are then mounted on a metal pin through a hole along the diameter of the straw, that is attached to a load cell. A package with a pre-punched hole is put directly under the tip of the straw, and aligned perfectly from the middle of the straw, down to the hole. Notice that each individual straw, is tested on a new unique package. Variations introduced by the package will therefore be included in the measurement.

The material on the package being perforated is not isotropic. It has different properties depending on which direction relative to how it was created, it is perforated. One direction used for testing is Machine direction (short: MD), another one is Cross direction (short: CD). Depending on which direction the test is conducted in (MD or CD), the resulting force will be different. Given that the properties of the straw is changed after it has perforated a package, the same straw cannot be tested in MD as well as in CD.

The data gathered from the test is how the force on the straw varies with displacement. What is used in this thesis is the peak force in the first peak. See figure 44 for a visual of what these graphs may look like, and where the *first peak* value is found.

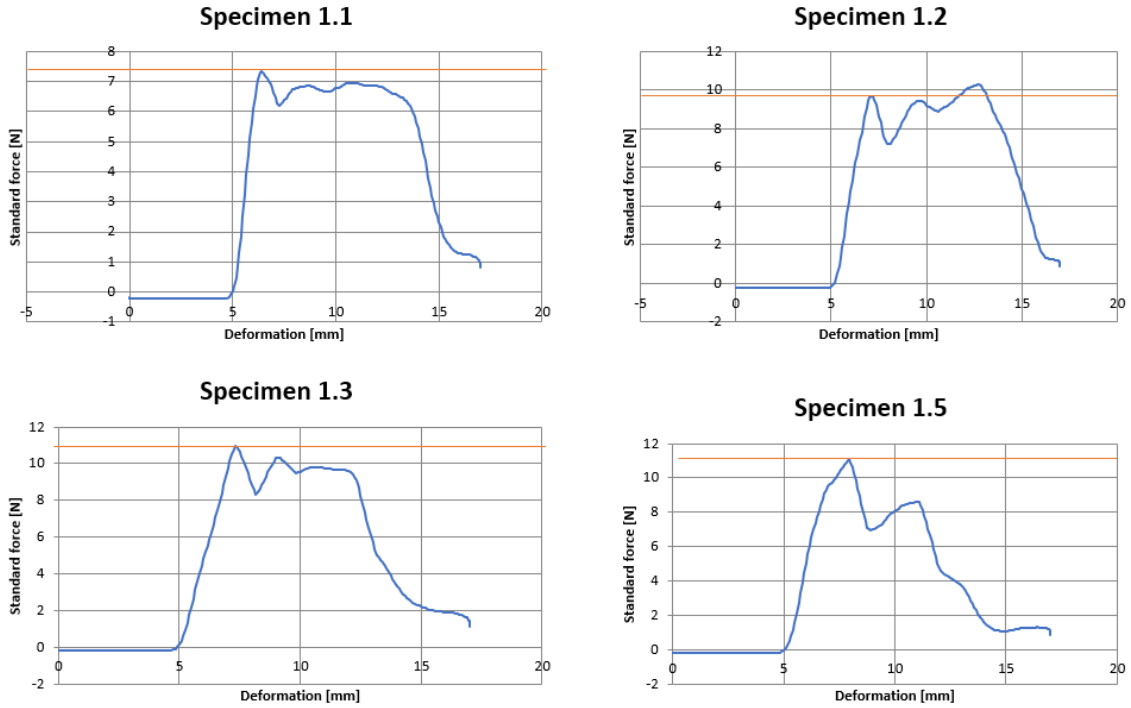

Figure 44: Obtained force VS displacement graphs for four different straws. These graphs are parametrised, and the value used is the peak force in the first peak. This value is marked with an orange line in all four graphs.

C Autoencoder code

```
### Encoder definition
class Encoder20(nn.Module):
    def __init__(self):
        super().__init__()

        ### Convolutional section
        self.encoder_cnn = nn.Sequential(
            nn.Conv2d(1, 8, 3, stride=2, padding=1),
            nn.ReLU(True),
            nn.Conv2d(8, 16, 3, stride=2, padding=1),
            nn.ReLU(True)
        )

        ### Flatten layer
        self.flatten = nn.Flatten(start_dim=1)
        ### Linear section
        self.encoder_lin = nn.Sequential(
            nn.Linear(81536, 28),
            nn.ReLU(True),
            nn.Linear(28, 24),
            nn.ReLU(True),
            nn.Linear(24, 20)
        )

    def forward(self, x):
        x = self.encoder_cnn(x)
        x = self.flatten(x)
        x = self.encoder_lin(x)
        return x

### Decoder definition
class Decoder20(nn.Module):
    def __init__(self):
        super().__init__()
        self.decoder_lin = nn.Sequential(
            nn.Linear(20, 24),
            nn.ReLU(True),
            nn.Linear(24, 28),
            nn.ReLU(True),
            nn.Linear(28, 81536),
            nn.ReLU(True)
        )

        self.unflatten = nn.Unflatten(dim=1,
```

```
        unflattened_size=(16,91,56))

        self.decoder_conv = nn.Sequential(
            nn.ConvTranspose2d(16, 8, 3, stride=2,
            padding=1, output_padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(8, 1, 3, stride=2,
            padding=1, output_padding=1)
        )

    def forward(self, x):
        x = self.decoder_lin(x)
        x = self.unflatten(x)
        x = self.decoder_conv(x)
        x = torch.sigmoid(x)
        return x

class AutoEncoder20(nn.Module):
    def __init__(self):
        super().__init__()

        self.encoder = Encoder20()
        self.decoder = Decoder20()

    def forward(self, X):
        encoded = self.encoder(X)
        decoded = self.decoder(encoded)
        return decoded, encoded

### Redefine loss to fit for an autoencoder
class AutoEncoderNet(NeuralNetRegressor):
    def get_loss(self, y_pred, y_true, *args, **kwargs):
        decoded, encoded = y_pred
        loss_reconstruction = super().get_loss(decoded, y_true, *args, **kwargs)
        return loss_reconstruction

### Define the network
net = AutoEncoderNet(
        AutoEncoder20,
        lr=0.5, warm_start = True, device = 'cuda'
    )
```

References

[1] Applied deep learning: Part 3, Autoencoders. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. Ac-

cessed: 2022-08-17.

[2] Coefficient of determination. https://en.wikipedia.org/wiki/Coefficient_of_determination. Accessed: 2022-08-10.

[3] Deep Learning vs. Traditional Computer Vision. https://arxiv.org/ftp/arxiv/papers/1910/1910.13796.pdf. Accessed: 2022-08-17.

[4] Hyperparameter Tuning the Random Forest in Python. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit\-learn-28d2aa77dd74. Accessed: 2022-07-24.

[5] Image of Autoencoder. https://www.jeremyjordan.me/autoencoders/. Accessed: 2022-07-24.

[6] iSixSigma Gage R&R. https://www.isixsigma.com/dictionary/gage-rr/. Accessed: 2022-05-23.

[7] Scikit-image, image processing in Python. https://scikit-image.org/. Accessed: 2022-07-24.

[8] Sympathy for Data, Combine Control Systems AB. https://www.sympathyfordata.com/. Accessed: 2022-07-24.

[9] Understanding K-means Clustering in Machine Learning. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1. Accessed: 2022-08-10.

[10] Understanding the 3 most common loss functions for machine learning regression. https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine\-learning-regression-23e0ef3e14d3. Accessed: 2022-08-10.

[11] What is a decision tree? https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1. Accessed: 2022-07-24.