Aalto University
School of Science
Master's Programme in Systems and Operations Research

Severi Saastamoinen

A Deep Learning Model for Filtering Unusable Visual Data

Master's Thesis
Espoo, December 30, 2022

Supervisors:	Professor Ahti Salo
Advisor:	Mikko Havimo M.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Systems and Operations Research

ABSTRACT OF MASTER'S THESIS

Author:	Severi Saastamoinen		
Title: A Deep Learning Model for Filtering Unusable Visual Data			
Date:	December 30, 2022	**Pages:**	42
Major:	Systems and Operations Research	**Code:**	SCI3055
Supervisors:	Professor Ahti Salo		
Advisor:	Mikko Havimo M.Sc. (Tech.)		

In this Master's Thesis, we study the creation of a classification model using deep learning neural networks for visual data. The purpose of the Thesis was to automatically filter out unusable pictures in defect recognition at a steel plant in order to streamline daily operations.

The model was created as a classification model which tags entire pictures into three categories: "DimPictures", "ReflectionPictures" and "GoodQualityPictures". The data was downloaded from company databases and tagged manually to belong into one of the categories. Thus it was possible to create the model as a supervised machine learning model.

The model performs well against the testing data with a 96% accuracy and performs fast enough to handle incoming datastreams on a batch-by-batch basis without getting overwhelmed. The more layers the underlying neural network had, the faster the model became. It is questionable whether the original training data is enough to account for all real-world pictures, but this can be improved by adding manually misclassified pictures as training data for the model.

This implies that the model performs adequately in handling incoming data on a batch-by-batch basis. The model could benefit from further tweaks to its neural network making it have a more complex layered structure. The model could also use more training data to prevent it from having a potential over-fit indicated by the discrepancy between the training and testing loss functions. However, more training data will become available from running the model on production data and this should address the over-fit issue.

Keywords:	classification model, deep learning, machine vision, digitalization, neural networks, supervised machine learning, sample quality
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Systeemi- ja operaatiotutkimuksen maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Severi Saastamoinen		
Työn nimi: Syväoppimismalli käyttökelvottoman kuvadatan suodattamiseksi			
Päiväys:	30. joulukuuta 2022	**Sivumäärä:**	42
Pääaine:	Systeemi- ja operaatiotutkimus	**Koodi:**	SCI3055
Valvojat:	Professori Ahti Salo		
Ohjaaja:	Diplomi-insinööri Mikko Havimo		

Tässä diplomityössä luodaan luokittelumalli kuvien lajitteluun. Malli luotiin käyttäen apuna syväoppimisneuroverkkoja. Työn tavoitteena oli kehittää malli, joka automaattisesti seuloo käyttökelvottomia kuvia käyttökelpoisten joukosta työajan säästämiseksi terästehtaalla.

Malli rakennettiin luokittelumallina, joka lajittelee kokonaisia kuvia kolmeen kategoriaan: ”DimPictures”, ”ReflectionPictures”ja ”GoodQualityPictures”. Koulutukseen käytetty data ladattiin yhtiön tietokannoista ja lajiteltiin manuaalisesti kuva kerrallaan yhteen näistä kolmesta tietokannasta. Täten datan pohjalta oli mahdollista kehittää valvottu koneoppimismalli.

Malli suoriutui hyvin testausdatasta 96%:n tarkkuudella. Se käsittelee dataa riittävän nopeasti selviytyäkseen tuotannon päivittäisestä datavirrasta satseittain käsitellen dataa nopeammin kuin mitä sitä tulee. Mitä enemmän kerroksia neuroverkossa oli, sitä nopeammaksi laskenta muuttui. On kyseenalaista, riittääkö alkuperäinen koulutusdata ottamaan huomioon todellisen tuotantoympäristön vaihtelun kuvien välillä. Tämä tulee kuitenkin parantumaan, kun mallin väärin luokittelemia kuvia tullaan lisäämään mallin koulutusdataan mallia ajettaessa.

Kaikesta päätellen malli kykenee käsittelemään sille tulevaa dataa riittävän hyvin. Mallia voitaisiin kehittää hyödyntämällä monimutkaisempaa neuroverkon rakennetta ja suurempaa määrää koulutusdataa, joka estäisi mahdollista ylisovitusta, johon ero koulutuksen ja testauksen tappio-funktioiden välillä vihjaa. Koulutusdataa tullaan kuitenkin saamaan lisää mallia ajettaessa, jonka pitäisi ratkaista ylisovitusongelmat.

Asiasanat:	luokittelumalli, syväoppiminen, konenäkö, digitalisointi, neuroverkot, valvottu koneoppiminen, otoksen laatu
Kieli:	Englanti

Acknowledgements

I wish to thank the staff and students of Aalto University, my brother, sister and everyone else who supported me through my studies.
Thank you!

Helsinki, December 30, 2022

Severi Saastamoinen

Contents

Chapter 1

Introduction

This Thesis is focused on creating a classification model for pictures at a production line. The purpose of the model is to filter out pictures that are too dim or have too strong reflections, so that a machine vision model after it can further classify errors in the intelligible pictures of the products.

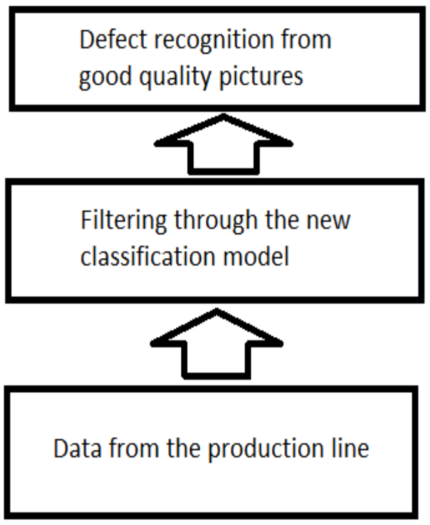

Defect recognition from good quality pictures
Filtering through the new classification model
Data from the production line

Figure 1.1: The flow of image data.

Figure 1.1 shows the flow of the image data. The pictures are taken at the production line, the bad quality ones are filtered out by the model described in this Thesis. Finally an object detection model which already exists recognizes various defects out of the images.

The model was created in Python using deep learning libraries from Tensorflow. Tensorflow is a deep learning library which contains a wide variety of tools that can be used in developing models (Pang et al., 2020). The model is a product of supervised machine learning where the pictures used to train and test the model were taken from former quality control pictures of the production line in the company's databases, after which they were classified manually and taught to the model. The model is to be run on a batch-by-batch basis, where the pictures recognized as good quality pictures are sent forward in a folder to an already existing object detection model for further examination. If the model is spotted classifying pictures incorrectly, those pictures will be labeled and added to the model's training data in order to correct the error.

1.1 Problem statement

The company has had an object detection model in use which automatically recognizes defects in the products on the line. The performance of this model is inhibited by bad quality pictures which are either too dim or have reflections from surrounding lighting on the product which makes the product unrecognizable. The amount of bad quality pictures also increases the workload of employees, as they have to go through them manually to see, if there are legitimate defects in these pictures.

The reasons for taking pictures from coils range from immediate quality control to longer term analysis of defects, their origins and what could be done to mitigate them. This requires the defects to be detected in the first case and this is most efficient to do with an automated system. Furthermore, in order to decide which research and development projects are worth the time, money and effort, it is important to learn what the potential benefit of reducing a specific defect is. Understanding the quantities of various defects and their inflicted losses on the business is therefore important (Breyfogle, 2003, pp. 116-117).

To solve this problem, another model was created to evaluate and filter the pictures before offering the good quality pictures to the object detection model. This should reduce employee workload and ensure that only useful data is stored in the databases.

The solution must also be able to run on other hardware than the one it is created on and be modifiable and understandable by employees with minimal on-boarding.

1.2 Structure of the Thesis

The thesis begins with a description of the problem and research environment. This is followed by a discussion on the methods and implementation of the solution. Finally, the effectiveness of the solution, conclusions and discussion of the Thesis are presented.

Finally, the appendix covers material from the project giving additional context.

Chapter 2

Background and Environment

The production takes place at a steel plant where analyzing data is used as a tool for future business decision making. The steel plant produces stainless steel from recycled steel which is melted for raw materials. The products of the plant are shipped either directly to customers or to further production steps at other sites. There are cameras both at the hot rolling mill and cold rolling mill which collect image data for analysis. The work is set in a cloud-environment where machine learning-models are used extensively on data collected from historical production in order to provide better analytics for the decision making of the company. Data collection, model development and usage are carried on a global scale. The model which classifies defects, recognizes the defects from within pictures through machine vision. The model developed adjacent to this Thesis instead classifies pictures in their entirety based on their characteristics. Deep learning is a relatively niche field but there are some libraries for Python that are available.

2.1 Use case

There are cameras at the end of several production lines monitoring the quality of steel. The pictures taken by the cameras are then processed by an object detection model which is already operating on two cold rolling process routes. Many of the pictures, however, do not show the steel surface correctly mostly due to inadequate or excessive lighting.

The pictures affected adversely by unsuitable lighting distract the object detection model during defect recognition and must be taken out of the data set which demands manual work if there is no automatic model to filter the pictures.

2.2 Cold rolling mill

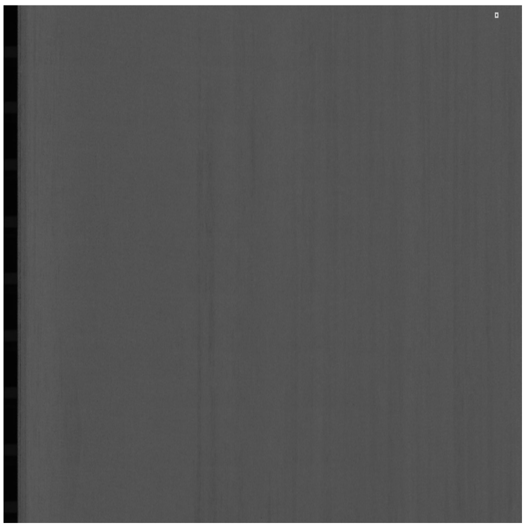


Figure 2.1: Defectless coil.

The physical production line from which the pictures are taken has long coils of steel being run through it as straight bands, where pictures are taken regularly to inspect the surface quality. A machine vision model then interprets these pictures and keeps count of various defect types to track developments in overall steel quality and sends notifications to those concerned, when something severe is spotted. Figure 2.1 shows an example of a product without defects as pictured by the camera system. The pictures are taken in various sizes and from various parts of each coil, but before being given to the model each of the pictures is resized to 180x180. The coils in the pictures are typically two kilometers long, 1.5 meters wide and 2mm thick, although the dimensions can vary for each individual coil.

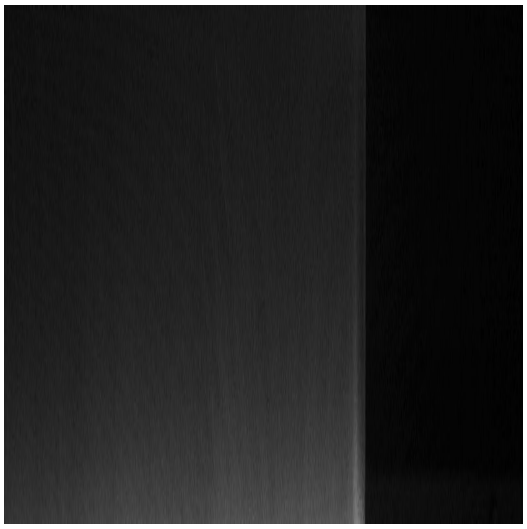


Figure 2.2: An excessively dim picture.

The picture in Figure 2.2 is too dark to be used for recognizing defects. It is not currently known why this is. This could be due to light being reflected away from the camera as the coil shakes slightly during its course through the line.

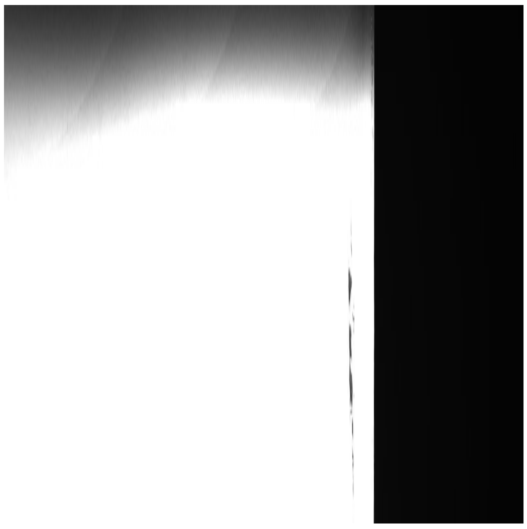


Figure 2.3: A picture which has been burnt through by reflected light.

Figure 2.3 shows a picture which has been burnt through by too much light. Such pictures need to be filtered out from the rest of the pictures going onward to defect recognition.

The production line itself where the data is collected from is a unified rolling, annealing and pickling line with annealing, pickling and cold rolling combined into a single structure, where the production line runs for the length of over a kilometer uninterrupted. The line is very modern and almost completely automated, requiring only a small crew on a normal work day. During annealing the product is treated with high temperatures to reduce hardness to make it more malleable and to increase ductility (Wu and Fan, 2020, p. 14). Pickling is treatment with acid which removes carbon layers from the surface of the steel after heat treatment (Eagleson, 1994, p. 834). In cold rolling, the product is thinned down to its target thickness while its total length increases. Unlike hot rolling, this is done without the need to heat the coil to high temperatures (Degarmo et al., 2003, pp. 384-408).

The quantities of steel flowing through the line are in the hundreds of tons daily. The models have to be fast enough to process approximately one million pictures daily. The line is stopped only for maintenance purposes. Therefore, it is easy to find training material for the model. However, when looking for specific types of pictures, it is possible that there are only few examples to be found. This may not be a problem because, by definition, they are not a large part of the material flow, do not affect the accuracy of the model to a large degree and thus are not a priority.

The camera is located above and below the coil as it runs through the line and the setting is illuminated by lights above the coil on its sides. The quality of the picture depends on the camera receiving a correct amount of light. If the light is reflected badly, the resulting picture can be either too dim or burnt through by the brightness.

Figure 2.4 shows the setting in which the pictures are taken at the end of the line. The lights on the sides illuminate the coil in order to enable the camera above to take pictures of the coils as the steel runs through. Problems with lighting can be caused by the coil shaking during its run through the line and reflections can occur, if camera lights hit the surface in a bad angle. Other causes for reflections or dimness could be the varying amount of ambient light around the facility during different times of day or different steel grades reacting differently to the same amount of light.

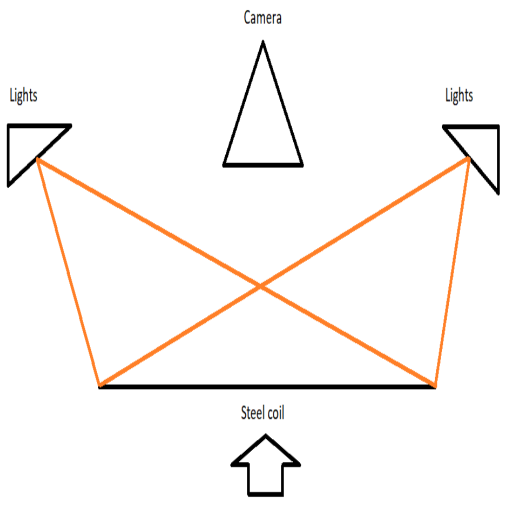


Figure 2.4: A schematic diagram of the setting where the pictures are taken.

2.3 Software development and deployment environment

The cloud environment in which the analytics is run is a collection of third party applications which run on Python. The models are developed in a separate environment before being deployed into the production environment. Developed models are containerized and adjacent folders are mounted to enable their usage reliably outside of the the hardware on the premises or virtual machines where they were originally developed and tested. Once a model has been developed, its performance if supervised and corrected by adding additional training data if necessary. Using cloud computing services introduces costs to the company which makes it necessary to plan ahead how the model is trained and run.

2.4 The accuracy of classification results

A classification model will inevitably lead to false positives and false negatives. Overall, their existence will not be critical for the operation of the model as long as they remain rare exceptions.

The model is supposed to filter out unusable pictures from the usable ones so that a false positive would be the model removing a usable good quality picture. The effects of this depend highly on the picture. For example, if the picture depicts normal non-defective steel, the consequences of the error would be practically irrelevant, since there is nothing to classify in the picture. If instead the picture depicts holes in the steel, the holes would remain undetected. The consequences of this might mean defective material being treated in the next phases of development or the defects even reaching customers. Both of which would cause unnecessary costs for the company.

False negatives would cause bad quality pictures to make it through to error recognition. At worst this causes false positives to be raised and at best the bad quality pictures accumulate into the company's databases without raising errors forcing human researchers to shift through them to get the usable material.

As long as the amount of both types of errors stays low (e.g. under 5% of classifications being wrong), the problems caused by the errors remain negligible (Adams and Essex, 2014, pp. 435-446). An individual error is unlikely to have severe ramifications in terms of used working hours or direct financial cost.

There is also an option of enabling manual inspection and classification for artificial intelligence classifications with a low confidence rating. This, however, would require substantially more working hours from employees and thus cause additional costs for the company. Also, the time needed to provide such functionality in a user-friendly fashion would be a significant bottleneck in time allocation.

Chapter 3

Methods

The methods used to solve the problem are to implement a deep learning model in Python. Deep learning seeks to imitate the activity of the human nervous system by building its own virtual neural network which seeks to imitate human learning (Marblestone et al., 2016). When building a deep learning model, one does not build the model directly but instead the neural network. The model trains itself within the network based on training data through trial and error. During the creation of the neural network for the model Tensorflow and Keras will be used. Tensorflow and Keras are libraries which contain useful functions when programming deep learning models (Nandy and Biswas, 2018, p. 129). The model is a classification model created via supervised learning. It was most practical to complete the project as a classification model classifying entire pictures, because the ultimate goal was not to recognize individual features and objects within pictures, but classify the pictures themselves for further use.

Supervised learning requires there to be ready categories with tagged examples of each category (Mohri et al., 2012). The model receives the pictures and learns from the similarities in each category, instead of letting the model itself determine from incoming data those pictures that are similar to each other, as is the case in unsupervised machine learning (Hinton and Sejnowski, 1999).

The solution in this Thesis is a deep learning model, because the technology enables the creation of machine vision models in a straightforward manner. The model uses supervised instead of unsupervised learning, because it is possible to tag sufficient amounts of training and testing data and there are also specific categories that are desired for the data, “good” and “bad” quality.

In addition to classification models, there are clustering and regression models. Clustering models are unsupervised machine learning models in which the observations are grouped into clusters based on their properties. No previous labels are utilized and the label of each observation is the cluster it is assigned to. The closer two observations are, the more similar they are. This can be applied to pictures by splitting the picture into segments and clustering them (Bewley and Upcroft, 2013, pp. 1-2). This is not very useful, because there is historical data which can be tagged to different labels manually. In addition, the properties that the pictures should be tagged according to are fairly specific. For example, the model is not supposed to classify the pictures based on whether or not they contain holes.

Regression models attempt to establish relationships between different data characteristics in order to predict a label. This form of supervised machine learning is used mostly to determine the dependency of a property of the data in relation to others. It would be impractical to try to use it for image classification (Cook and Weisberg, 1982, pp. 313-361).

Hyperparameters are parameters which govern the creation of the model. Because the parameters of the model are created within the neural network without human interference, hyperparameters are a method to fine-tune the parameters of the model indirectly. The learning rate measures how much the model corrects itself during training to minimize its loss function. A higher learning rate means that the model will approach an optimal solution faster with less training, but will find it difficult to arrive at the best possible solution due to an inability to fine-tune parameters near the optimal values. A low learning rate, on the other hand, means that it will likely take a longer training for the model to provide good results increasing computing costs, but the ultimate result will be of better quality unless the algorithm gets stuck in the loss functions local minima (Buduma and Locascio, 2017, p. 21).

The batch size determines how many pictures are taken into a single training step, between which the weights of the parameters are updated. A smaller batch size can result in a slower training, whereas larger batch sizes are limited by the computer graphics processing unit the training is done on.

An epoch is a series of steps that train through the entire dataset. The number of epochs used during training is not fixed in this Thesis and thus choosing the correct number of epochs is at the data scientist's discretion. A large number of epochs can make the training needlessly long. The improvements made to the model usually stall after a certain amount of training depending on e.g. the shape of the neural network, hyperparameters and dataset quality.

As hyperparameters, the model will use a batch size of 64 which is recommended in literature alongside with the size of 32 (Kandel and Castelli, 2020). The number of epochs used for training will be determined empirically during model development based on how many epochs it takes for the model's training loss to stop decreasing and the accuracy of the model to stop improving. This is because there is no definite way of determining the best number of epochs outside of empirical tests (Sinha et al., 2010). It would be possible to let the code determine this with a conditional stop for model training when the training loss has not decreased in several consecutive epochs, but it should not be necessary for training an individual model, as it is possible to let the model train for several excess epoch to be certain of optimal training results. This leads to higher computing costs, but ensures the development of a slightly better model.

3.1 Data augmentation

The validation data has the issue that it is taken from the same few coils that the training data was taken from. Thus, taking into account how most pictures of the same coil look very similar and the bad quality pictures tend to concentrate on specific coils, there is a danger of overfitting the data for only specific coils or steel grades (Shorten and Khoshgoftaar, 2019).

To solve this, it is possible to use data augmentation. In Figure 3.1, there are examples of pictures generated by data augmentation from a single picture.

Data augmentation modifies a picture by modifying different parameters of a picture to create variation within the data set. The goal is to compensate for small sample sizes and take in to account slightly different angles and varying degrees of brightness in the real world.

This can be done by shifting the picture upwards or sideways within the frame, flipping the picture around its horizontal or vertical axis, rotating the picture around its center, zooming in or out of it to make the object look larger or smaller, modifying the brightness of the picture or doing a shear transformation to the picture in which one edge of the picture is moved up while the other one is moved down. These variations can be done with commands from Keras.

Out of these possibilities rotation seems to be of limited use, because practically speaking all pictures will have the steel coil running in the same angle.

Figure 3.2 shows an example of how the Randomflip function can be used to augment the dataset. It flips a picture around to make the dataset more robust.

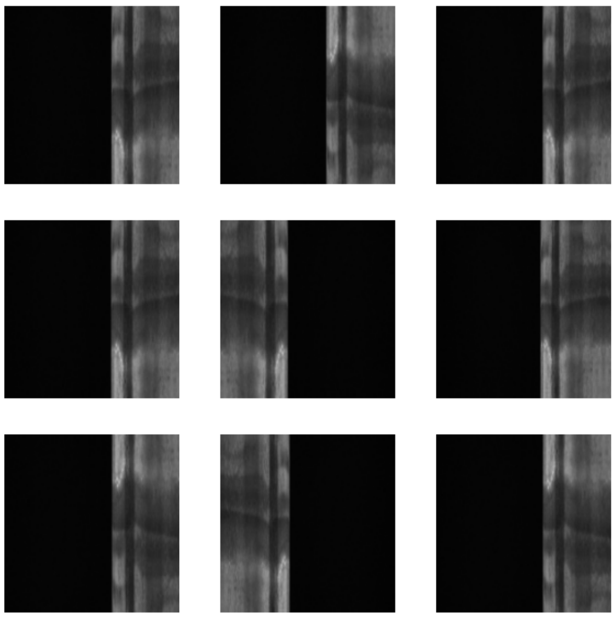


Figure 3.1: Pictures generated by data augmentation from the data to prevent overfitting.

Figure 3.3 is an example of how the Randomrotation function can be used to augment the dataset. It rotates the pictures randomly within given parameters This is not very useful in the project, however, because all pictures in the data display the coil moving vertically in relation to the camera. Therefore, Randomrotation will not be used in training.

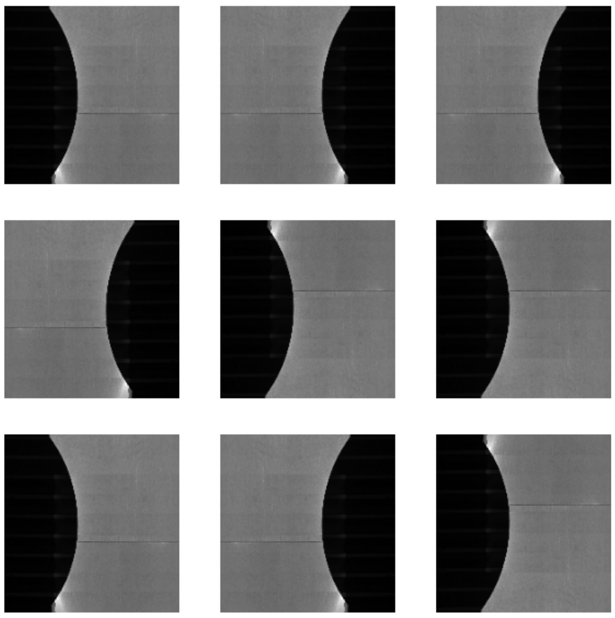


Figure 3.2: An example of the use of Randomflip.

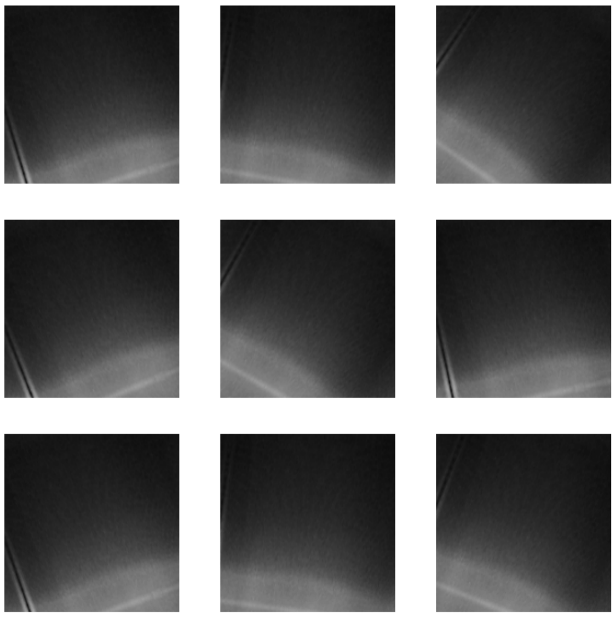


Figure 3.3: An example of the use of Randomrotation.

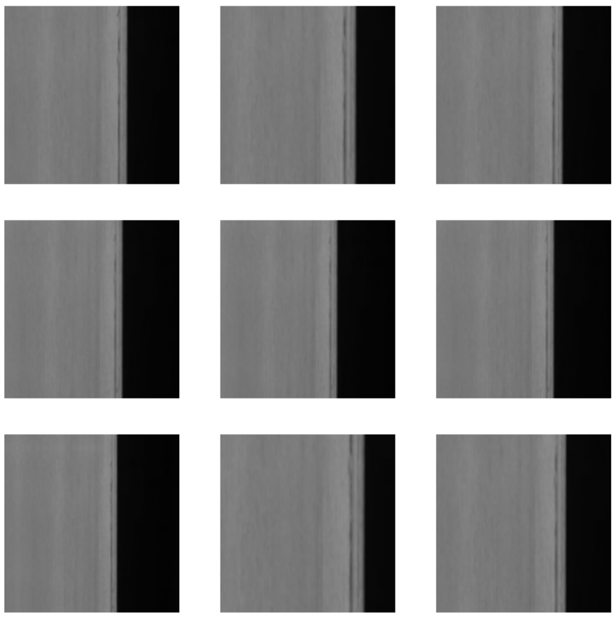


Figure 3.4: Pictures zoomed with Randomzoom.

The pictures in Figure 3.4 are created by zooming into original pictures within the training data set. The usefulness of these pictures in training the model to recognize new data varies, but many of them are similar to real pictures taken at different parts of the coil.

Figure 3.5 shows cropped pictures which look similar but offer slight variation and are thus useful for the augmentation.

Ultimately, the functions Randomzoom, Randomflip, Randomcrop were chosen to be used in augmenting the data. When several of these functions are added together, it is possible to combine them in the same generated picture, thus further increasing the amount of pictures available.

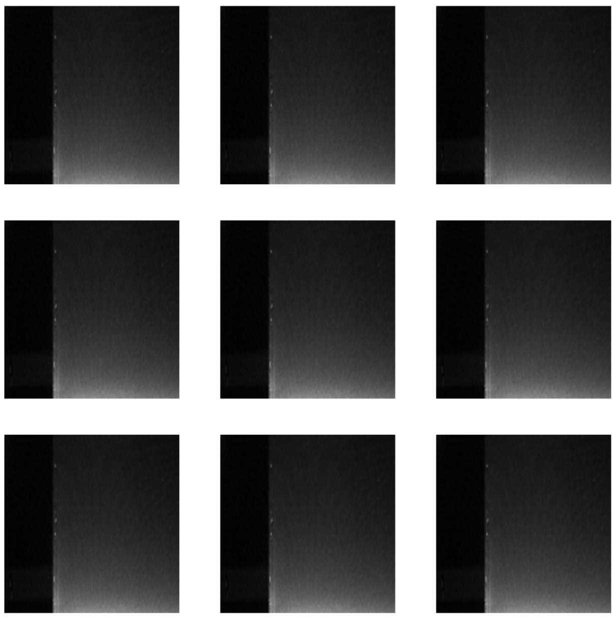


Figure 3.5: Randomly cropped pictures.

Chapter 4

Implementation

The project was successfully completed as a sequential Keras model. The largest change to the initial plans was to divide the bad quality pictures into two separate categories. This was done to avoid confusion while training the model. Since dim and reflective pictures are hard to distinguish from each other, it makes sense to create two separate tags for bad quality pictures, one for pictures which are too dim and the other for pictures with severe reflections. This makes it easier for the model to tell these very different looking pictures apart from each other, making the overall classification process more reliable. It is worth noting that the model's ability to distinguish between reflective and dim pictures was not relevant to the main purpose of the model which was to keep both of these groups separate from the good quality pictures.

The hyper-parameters of the model were not experimented with to a large degree. However, this did not seem necessary, as the model performed sufficiently with minimal adjustments. It would be a worthwhile topic of further research to explore and document how various hyper-parameters affect the accuracy and speed of the model. This could be fine-tuned by having another artificial intelligence tool to tune the hyper-parameters. However, this would add an additional layer of computing and time required to complete the Thesis would have increase significantly. Also, the costs caused by using 3rd party cloud computing services would have multiplied beyond acceptable levels.

4.1 Hyper-parameters and structure

Layer (type)	**Output Shape**
rescaling1 (Rescaling)	(None, 180, 180, 3)
conv2d (Conv2D)	(None, 180, 180, 16)
maxpooling2d (MaxPool-ing2D)	(None, 90, 90, 16)
conv2d1 (Conv2D)	(None, 90, 90, 32)
maxpooling2d1 (MaxPool-ing2D)	(None, 45, 45, 32)
conv2d2 (Conv2D)	(None, 45, 45, 64)
maxpooling2d2 (MaxPool-ing2D)	(None, 22, 22, 64)
flatten (Flatten)	(None, 30976)
dense (Dense)	(None, 128)
dense1 (Dense)	(None, 3)

Table 4.1: The structure of the neural network.

Table 4.1 outlines the shape of the neural network. The first row shows the input layer with its dimensions, each row below that is subsequent hidden layer until the last row which is the output layer. The structure of the neural network was made with the hidden layers expanding in size towards the center and contracting again towards the output layer. This is a simple structure which enables the model to separate the picture into smaller sections, recognize them and combine them towards a greater whole. The model trains itself several times through the neural network during training. At the end of each run through the network, the model changes itself based on whether it improved or not.

The model was trained with a batch size of 64 and 50 epochs. The learning rate was set to 0.001. The number of epochs was determined empirically by training the model with more epochs until training progress visibly stopped. For all test runs there was a drop off in accuracy gain and loss function minimization slightly before the 50th epoch. The learning rate of 0.001 was found sufficiently small to provide a confidence of over 95%. Decreasing the learning rate runs the risk of increasing time required for retraining to unacceptable amounts.

4.2 Training data

The training data consisted of real-world pictures taken from coils at the end of the rolling, annealing and pickling line with digital cameras. The data was found by manually browsing databases, finding a coil which had either reflections or dim photos taken from it. All pictures of the found coil were then downloaded to a laptop and sorted manually into "GoodQualityPictures", "DimPictures" and "ReflectionPictures". This process was also repeated to coils that were found to only have good quality pictures taken from them and picture folders of coils chosen at random. The goal was to have a dataset which would not be biased towards a specific type of product type or coil, since by the nature of the database, it was easier to take all pictures representing one coil at once.

It was also important to balance each tag with roughly equally many pictures in the data, because if one tag would have magnitudes more training data, it would cause issues due to over-fitting. The balancing was done after the data was collected by deleting pictures. This made it possible to avoid overfitting while having a representative sample of data to train the model. The exact volume of data required for training a classification model is hard to determine. For this purpose, there is no strict rule, but classification models often start working properly with hundreds of data pieces per tag.

The amount of data was increased as long as there was a tangible improvement in model performance. Ultimately, there were 1038 Pictures tagged with "DimPictures", 722 tagged "GoodQualityPictures" and 296 tagged "ReflectionPictures". The imbalance between the tags is still acceptable, although there are clearly fewer "ReflectionPictures" than other tags. The imbalance is due to how rare it is to have reflections in the pictures that make them unintelligible.

There is a degree of subjectivity involved in tagging pictures, because it is up to the tagger to decide which picture belongs to which tag. This can make judging close calls problematic, because the opinion of the tagger can differ from what defect classification software is able to understand. This problem can be mitigated by consulting senior staff about the software's capabilities and having a single individual do all the tagging, so that the results are consistent and should average out within a larger dataset towards an acceptable interpretation of the tags.

It is not known why some pictures are burned through by reflections. This may be due to certain steel grades having more reflective surfaces or possibly thinner coils shaking more, causing light to reflect towards the camera. The latter explanation seems plausible, because pictures depicting folded coils often had reflections on them also. This indicates that the angle the steel affects the picture quality and since the coils angle towards the camera changes if the coil shakes significantly, there may be a direct relation between the two.

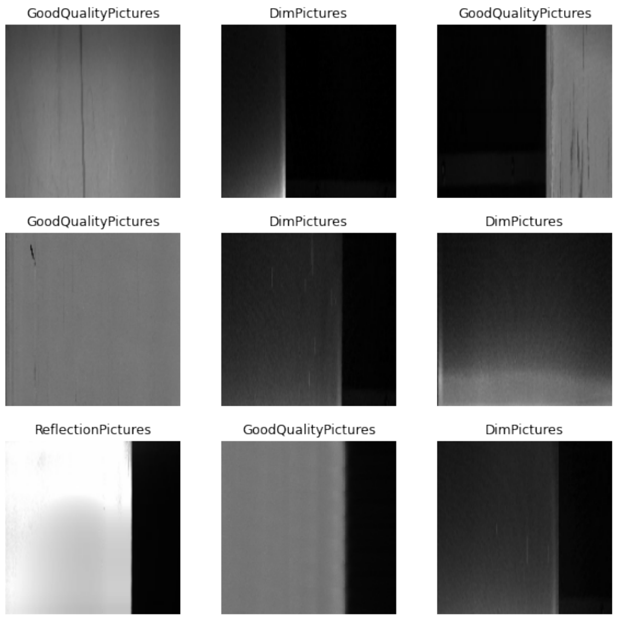


Figure 4.1: Examples of tagging data for training.

Figure 4.1 shows examples of pictures and tagging them. The tag reads above each picture.

4.3 Model without data augmentation

The initial attempt at creating the model involved composing and training the model directly from the data. A share of 80% of the data was used for training and the remaining 20% for validation.

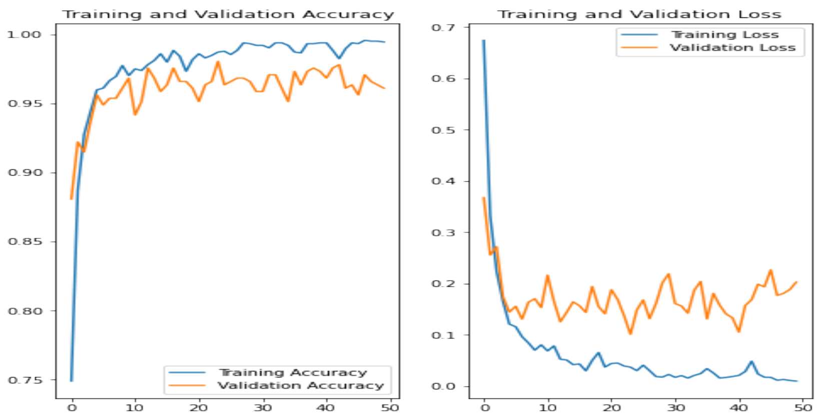


Figure 4.2: Performance of the model with training and testing data.

Figure 4.2 displays the development of the models accuracy and loss function. Validation is the most interesting aspect, because it shows how the model would perform in a setting with completely new data it has not been trained with.

The validation of the model resulted in an accuracy of 96.1%. The accuracy is based on using the validation data on the model and calculating how many of the pictures the model classified correctly regardless of class. The results are satisfactory, as the model performs well against the testing data. The quality of the data is concerning however, because there is a danger of the model being overfitted. This hypothesis is also supported by the fact, that the validation loss curve performs slightly worse that the training loss curve. Ideally these two should be aligned, but the discrepancy indicates that additional data could be useful in training.

4.4 Model with data augmentation

With the model augmented it is marginally more accurate that before at 96.5%. There is still a minor but consistent difference between the training and validation loss curves which persists despite adding more training data and augmenting the data for overfitting. The training has enough epochs to provide all improvements to model accuracy that are expected to manifest themselves realistically. In addition, with data augmentation, the validation loss function starts to worsen after about 30 epochs. This may be due to the model getting overfitted for augmented data that does not correspond to reality.

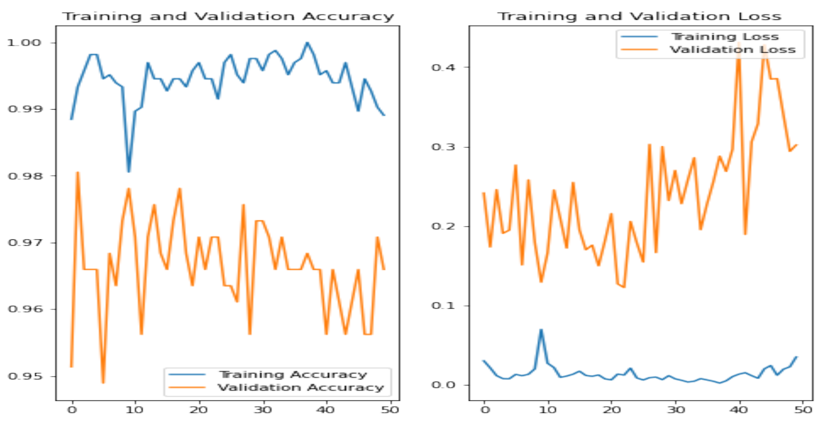
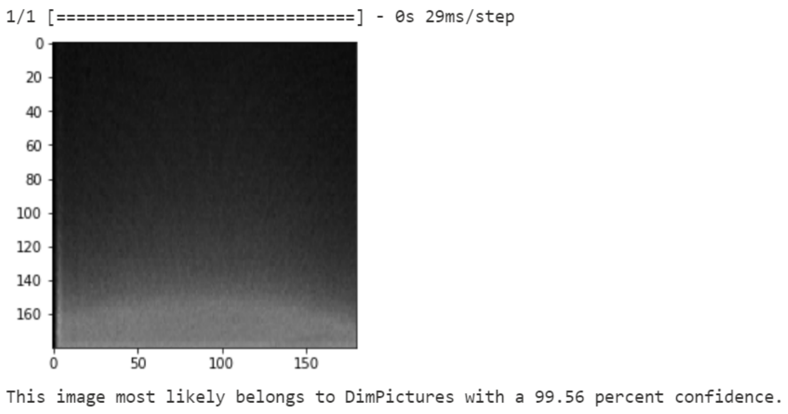


Figure 4.3: Performance of the model after data augmentation.

Figure 4.3 displays the performance of the model after augmenting the training data. The model does not experience significant changes to its accuracy from the amount of epoch it is trained through. Regardless, the models validation accuracy is marginally better than it was without data augmentation.

4.5 Model speed and performance

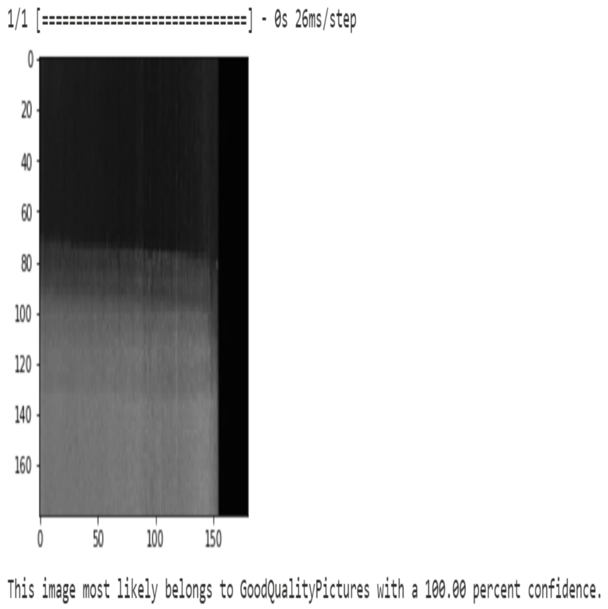
The model's performance averages out at roughly 26ms per measured picture. The speed should be clearly under 10ms per picture for the model to perform effectively against the incoming flow of data. This is unlikely to be a problem, however, because model development was done on a virtual machine without graphics processing unit or an efficient central processing unit in order to cut costs. The virtual machine had 4 processor cores and 28GB random access memory. The servers in which the model will be ultimately run will have more powerful hardware to handle the classification.


```
1/1 [==============================] - 0s 29ms/step
```

```
This image most likely belongs to DimPictures with a 99.56 percent confidence.
```

Figure 4.4: An example of a dim picture used as testing data.

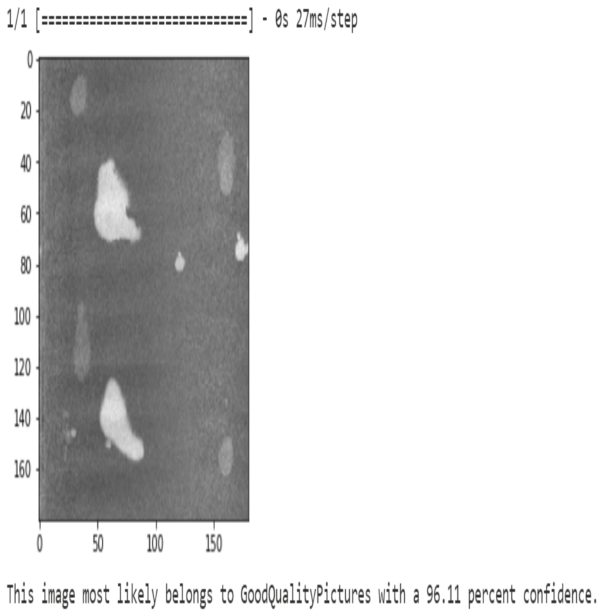
The picture in Figure 4.4 is too dim to be used. It is impossible recognize if there are defects on the coil or not, thus the model was trained with different examples of dim pictures such as this for it to be able to recognize them.


```
1/1 [==============================] - 0s 26ms/step
```

```
This image most likely belongs to GoodQualityPictures with a 100.00 percent confidence.
```

Figure 4.5: An example of a good picture with pigment used as testing data.

There is dark pigment on the coil in figure 4.5. This can easily be confused as poor lighting. Therefore it is important to have representation of such cases also in the training data.

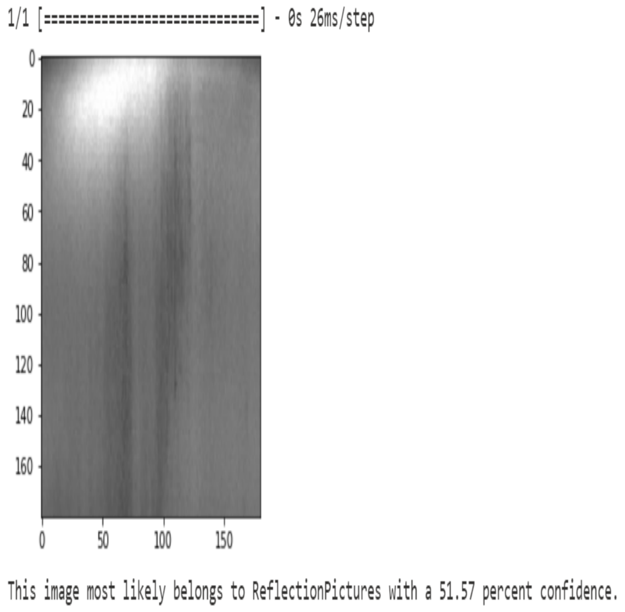


```
1/1 [==============================] - 0s 27ms/step
```

```
This image most likely belongs to GoodQualityPictures with a 96.11 percent confidence.
```

Figure 4.6: An example of a good picture with inconsequential reflections used as testing data.

Figure 4.6 shows actual reflections on the coil, but they do not prevent one from seeing whether or not the coil has defects. Therefore, it is better to classify this type of picture as good quality.



```
1/1 [==============================] - 0s 26ms/step
```

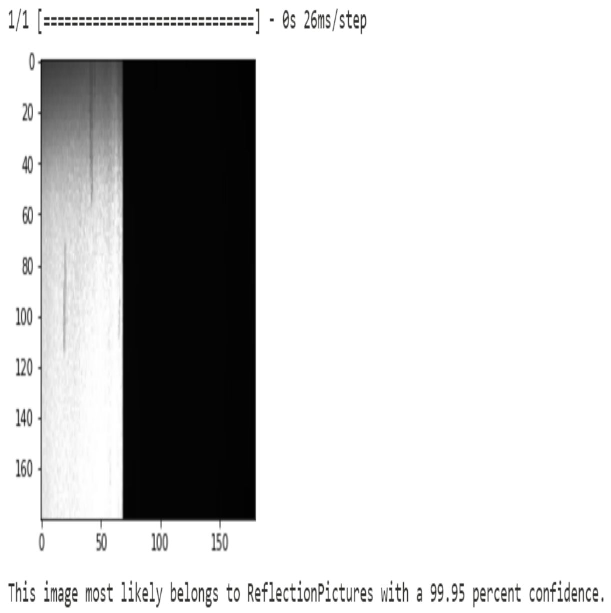
```
This image most likely belongs to ReflectionPictures with a 51.57 percent confidence.
```

Figure 4.7: An example of a mild reflection used as testing data.

Figure 4.7 shows a picture which could arguably be classified as good quality or as a reflection picture. Here the picture is classified as a reflection picture with a low confidence of 51.57%. The model evidently had difficulty deciding, should the picture belong to "ReflectionPictures" or "GoodQualityPictures". The end result depends on human interpretations during classification of the training data. If a specific type of error is overwhelmingly less preferred than the other, it is possible to set a condition in the code to count pictures in specific classes only if the confidence supersedes a certain confidence.

The reflection in Figure 4.8 is an example of a picture rendered the coil and its defects unrecognizable. The picture is thus unusable for the purposes of defect recognition and classification.

Ultimately, the folders containing the model's data were mounted in order for them to be referred to in code outside of the workspace the model was created on. The model was containerized and sent to a cloud storage, where it could be used further. The containerization had to be done with another virtual machine which supported using a graphics processing unit. After this, the model will be retrained with pictures it has failed to recognize and adapted to meet changing needs.



```
1/1 [==============================] - 0s 26ms/step
```

```
This image most likely belongs to ReflectionPictures with a 99.95 percent confidence.
```

Figure 4.8: An example of a picture made useless by reflections used as testing data.

Chapter 5

Evaluation

The development of the model succeeded in that the model provides the results it is supposed to with a good confidence. The model was containerized, sent to cloud data storage and is ready to be used on any hardware required. The speed of the model was lacking, but this is likely due to the hardware it was trained and tested on.

The model has some confusion in borderline cases between classes. This is due to human interpretation while classifying the data. There are also some anomalies between the "DimPictures" and "ReflectionPictures" classes, as many pictures have a very dark lighting with reflections in the middle. This is not a very serious problem, however, because the primary concern of the model is to separate the "GoodQualityPictures" class from the others. If the other two classes mix together, it should not interfere with the performance of the model.

The loss function of the testing remained higher than that of the training function. This indicates that the model could use more training data. However, appropriate training data for reflections specifically is hard to come by and the model seems to perform sufficiently regardless of the lack of training data. The issue with data can be mended over time by adding pictures the model has misidentified to the training data correctly tagged. This of course would require a manual review of the data regularly, which would increase the required working hours.

It would have been interesting to further explore and test various hyperparameters and structures for the neural network used in training the model. However, the time used to retrain the model with increasingly complex neural networks would have been lost and it is uncertain if this would have resulted in sufficient improvements in the model to justify the time allocation beyond the initial experimentation and study.

Chapter 6

Discussion

It would have been possible to have an option to manually review classifications made by the model with a lower confidence (e.g. $< 70\%$). However, this would require working hours from staff to be able to understand the environment the artificial intelligence is built in. This time could be used for other projects elsewhere and thus a manual solution with large amounts of data could prove expensive due to its opportunity cost. Manual inspection will still be carried out so that once an error in classification is spotted further on in the pipeline, the pictures can be tagged and added to the training data of the model.

It was surprising how relatively straightforward it is to create a deep learning model with a self constructed neural network. There are many vendors offering similar services to companies, but it seems possible to circumvent many expensive fees by creating the software by yourself. Having these capabilities in-house makes it also possible to maintain and adapt software without fear of vendor lock-in or accumulating costs, once the infrastructure has been built in the first place.

It would have been interesting to experiment how a classification model changes its behaviour when the underlying neural network is modified by adding more layers or nodes to existing layers, for instance. It would be interesting to find out especially how the performance in terms of speed is affected by a more detailed neural network and which features specifically have the most significant effect.

While this is interesting, time constraints forced this experimentation to be minimal in this Thesis. This would be interesting and feasible to research, when there is a ready-made working model provided at the start of the project and the sole goal is to conduct experiments to it by modifying its hyper-parameters and neural network structure, running it and comparing results between runs.

In the literature review for this Thesis, no clear cut rule was found on what the size of the training dataset should be for each class. The closest to this was to have the datasets for each class within the same scale of magnitude and that the identification of some classes functions well with a few hundred samples while others require more. This knowledge comes mostly from anecdotes from experienced data scientists instead of rigorous scientific research and is in stark contrast to, e.g., linear regression, where a $1 : 10$ ratio is widely considered ideal between the number of explanatory variables and training data (Hair et al., 2010, pp. 573-574). Having a baseline amount of required data would simplify the initial steps of the development process significantly.

It would be interesting to examine data from other parts of the production process in order to identify which factors cause the quality of the pictures to deteriorate. Possible causes are the surfaces of certain product types reflecting light differently and thinner coils possibly shaking more while moving through the line. This would be interesting, because knowing this would help focus the development efforts on mitigating unacceptable quality among the most susceptible pictures.

Chapter 7

Conclusions

The purpose of this Thesis was to create a classification model to sort out entire pictures in different classes between usable and unusable ones for defect recognition. The project environment was a cloud service handling data from a steel plant, where an already existing artificial intelligence classifies defects in the pictures into further categories. The project was successful and the model provides accurate classifications in a reasonably short amount of time.

The model was based in a deep learning artificial intelligence environment which was developed with Python using Tensorflow and Keras libraries. Augmented data was added to the training data which was a collection of modified pictures based on the real data in order to reduce overfitting. This provided a small increase in overall model accuracy and resulted in there being smaller differences in model performance throughout different training epochs.

When the underlying structure of the neural network used to train the model was modified to contain more layers, the model became faster as a general trend. This was unexpected and quite interesting. It would be an interesting topic of further study to explore how the shape of the neural network affects the resulting model in more detail.

The model may make erroneous classifications, but this can be improved over time by adding pictures that were manually noticed to be wrongly classified into the training data of the model and retraining the model to recognize the new type of pictures.

The main difficulty in developing the model was in finding a good representative data set to train the model with, because most pictures containing reflections precisely come from a limited number of specific coils. As a result, the training data is biased towards the features of these specific coils, such as surface characteristics caused by steel grades, the specific parts of the coil the pictures are taken from and coil width. This casts some doubts on the validation accuracy. The problem might not be that relevant though, as certain coils have more of a tendency to have reflections and if these coils are represented in a proportionate manner, the results should reflect reality, because the over represented coils stem from the majority of coils that have reflections in the first place.

Bibliography

R. Adams and C. Essex. *Calculus A Complete Course, 8th edition.* Pearson, Toronto, 2014.

A. Bewley and B. Upcroft. *Advantages of Exploiting Projection Structure for Segmenting Dense 3D Point Clouds.* Queensland University of Technology, Brisbane, Australia, 2013.

F. Breyfogle. *Implementing Six Sigma: Smarter Solutions Using Statistical Methods, 2nd Edition.* Wiley, Austin, Texas, 2003.

N. Buduma and N. Locascio. *Fundamentals of Deep Learning : Designing Next-Generation Machine Intelligence Algorithms.* O'Reilly, Sebastopol, California, 2017.

R. Cook and S. Weisberg. Criticism and influence analysis in regression. *Sociological Methodology*, 13:313–361, 1982.

E. Degarmo, J. Black, and R. Kohser. *Materials and Processes in Manufacturing, 9th edition.* Wiley, Austin, Texas, 2003.

M. Eagleson. *Concise Encyclopedia Chemistry.* De Gruyter, Berlin, 1994.

J. Hair, W. Black, B. Babin, and R. Anderson. *Multivariate Data Analysis 7th Edition.* Pearson, Hoboken, New Jersey, 2010.

G. Hinton and T. Sejnowski. *Unsupervised Learning: Foundations of Neural Computation.* MIT Press, Cambridge, Massachusetts, USA, 1999.

I. Kandel and M. Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6:312–315, 2020.

A. Marblestone, G. Wayne, and K. Kording. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10: 94, 2016.

M. Mohri, A Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning.* MIT Press, Cambridge, Massachusetts, USA, 2012.

A. Nandy and M. Biswas. *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python.* Apress, 2018.

B. Pang, E. Nijkamp, and Y.N. Wu. Deep learning with tensorflow: A review. *Journal of Education and Behavioral Statistics*, 45(2):227–248, 2020.

C. Shorten and T. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(60):1–48, 2019.

S. Sinha, T. Singh, V. Singh, and A. Verma. Epoch determination for neural network by self-organized map (SOM). *Computational Geosciences*, 14: 199–206, 2010.

H. Wu and G. Fan. An overview of tailoring strain delocalization for strength-ductility synergy. *Progress in Materials Science*, 113:100675, 2020.

Appendix A

Additional pictures

Below are further examples of the data used to train the model and the types of pictures that the model is expected to handle. The white rectangles indicate tagged defects which are supposed to be identified from the pictures determined to be of sufficient quality by software outside the scope of this project. These pictures are meant to provide additional context to the project and its environment.

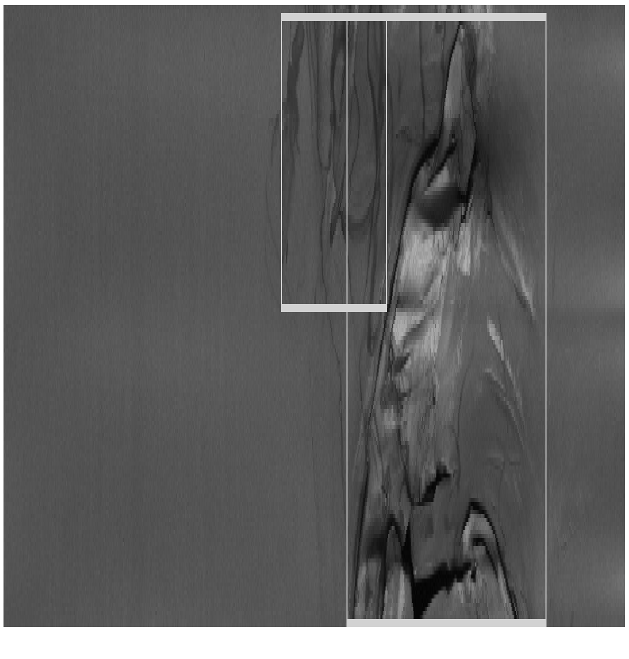


Figure A.1: A coil with a fleck and a hole on it.

The coil in Figure A.1 has a fleck and a hole in it. The severity of various defects depends on the product. A fleck might not be serious if the product is intended to be in the internal side of machinery, for instance, where it cannot be spotted. The hole, however, is always a severe defect which causes the specific part of the coil to be taken off and scrapped.

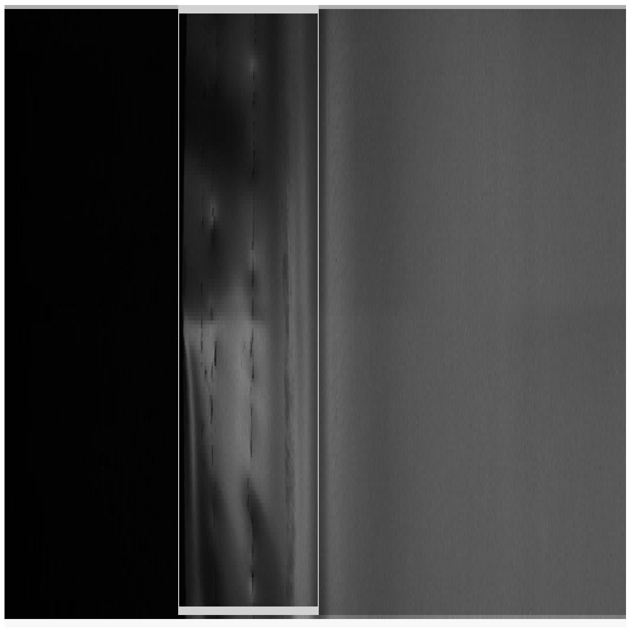


Figure A.2: A coil with a fold on it.

The coil in Figure A.2 has folds on its left side. This could be due to the coil hitting something during transportation. The lighting of the coil gets dimmer near the fold due to light not being reflected from the sloped surface properly.

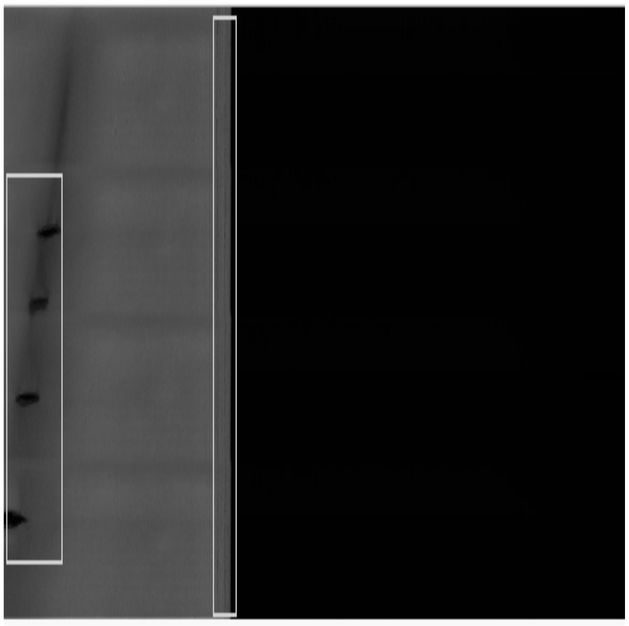


Figure A.3: A coil with holes and scratches.

In Figure A.3 there is a hole and light scratches at the right edge of the coil. Such scratches are very common and often the edges of coils are cut and the planned material for the saleable products is taken from the center.

The scratches in Figure A.4 can be caused by rollers during cold rolling when the roller comes into direct contact with the coil. The resulting traces are largely aesthetic, but can cause issues if the material was supposed to end up e.g. on the outer surface of a car.

Figure A.4: A coil with stain scratches.