A Machine Vision System for Detecting Automotive Gear Defects

by

Abdelrahman Allam

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements
for the degree of

Master of Applied Science

in

Engineering + Artificial Intelligence

Guelph, Ontario, Canada

ABSTRACT

A MACHINE VISION SYSTEM FOR DETECTING AUTOMOTIVE GEAR DEFECTS

Abdelrahman Allam

University of Guelph, 2021

Advisor:

Dr. Medhat Moussa

This thesis presents new methods that can be used for the automated quality inspection of automotive gears. Typically, gears are manually inspected at the end of the manufacturing process by trained human inspectors. This process is expensive, somewhat subjective, and suffers from errors due to human fatigue. Automating this process would improve quality and traceability. This would result in fewer rejected gears, thus reducing the overall costs. This thesis investigates multiple approaches ranging from simple image processing techniques to deep learning. The best results were obtained using a hybrid system that combines deep learning with domain knowledge.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr. Medhat Moussa for his invaluable assistance, guidance, and advice, which enabled me to achieve a project of this quality, magnitude, and substance. His mentorship has meant a lot to me and has helped me attain more professional experience.

I would also like to thank my colleagues Cole Tarry, Matthew Veres and Patrick Wspanialy from the Robotics Institute for their support and useful technical assistance. Thanks for always guiding me to obtain better results.

Last but not least, I would like to thank the most important people in my life: my parents, Magdy Allam and Hanan Hassan, and my sister, Hagar, for their unfaltering support. Achieving this without their support would have been more difficult. Finally, I would like to thank all of my family members and friends who helped me in one way or another to finish my degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Gear manufacturing plants around the world face quality issues with their manufactured parts. In order to avoid malfunctions, the parts have to pass quality inspection standards. The current inspection process of an automotive manufacturer located in Guelph, Ontario relies on each manufactured gear being inspected by human inspectors. However, the accuracy of the inspection depends on the ability of the inspector to recognize defects in the gears. In addition, the time spent to perform the process affects the number of gears that can be inspected daily. Therefore, the manufacturer aims to automate the quality inspection of the gears to improve the accuracy and reduce the process time.

An automated inspection system captures multiple images of the manufactured gears as inputs to the system. These images are then inspected for defects by applying a developed algorithm. Finally, the output from the inspection indicates whether the manufactured gear is defective or not.

1.1 Types of Gears

At the aforementioned facility in Guelph, Ontario, four different configurations of helical gears are manufactured for use within automotive power transmissions. These gears have the same overall shape, but differ in their corresponding tooth profiles. These gears share the same total length of 96 mm, a tooth length of 54 mm, and the same outer end shaft diameters (both ends) of 40 mm and 50 mm respectively. In addition, they have the same tapered inner profile, with the profile diameter increasing from 25 mm to 28 mm. Table 1.1 presents the characteristics of the four different types of gears, as stated in the seminal work of Hall [1]

Table 1.1: Characteristics of the four types of gears in the dataset [1].

Gear Type	Number of Teeth	Pitch Diameter (mm)	Helix Angle (Degrees)	Pressure Angle (Degrees)	Major Diameter (mm)	Minor Diameter (mm)
A	22	69.57	23.5	22.5	78.5	62
B	22	66.437	22.5	22.5	75.2	59.5
C	26	73.368	21.25	20	82	66.5
D	22	63.953	21.75	22.5	72.4	57.5

1.2 Types of Defects

Over 100 different types of defects can occur in the gears manufactured by the facility (a full list of the defects is available in Appendix A). These defects occur randomly on the four different types of gears with no specific defects occurring for any type. However, the most common defects that exist on the gears are nick failure and damaged teeth. In Appendix A, these defects have

defect labels of 76 and 55 respectively. The focus of the inspection system is to detect these two defects.

Nick failure defects exist on the edge of the outer circumference of the upper and lower rims of the gear, as shown in Figures 1.1 and 1.2 respectively. The defect creates a high stress area on the rim which can lead to more damage, and in the worst case scenario, gear fractures. Figure 1.3 shows different samples of nick failure defects surrounded by red rectangles for better presentation.

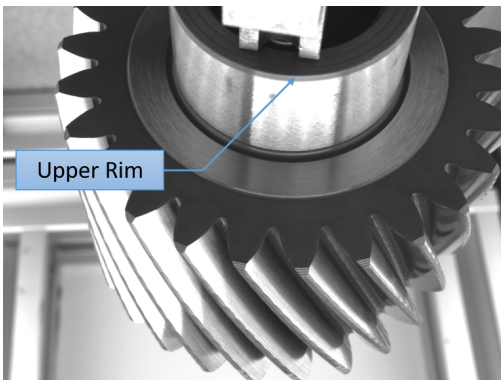


Figure 1.1: Upper rim of the gear.

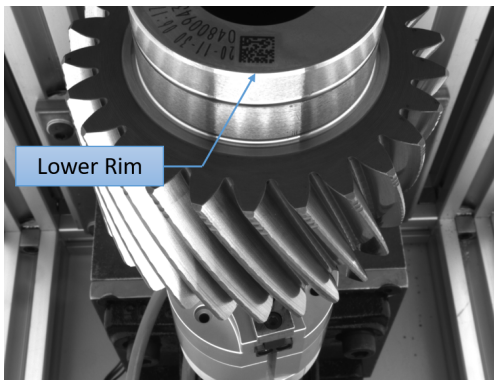


Figure 1.2: Lower rim of the gear.

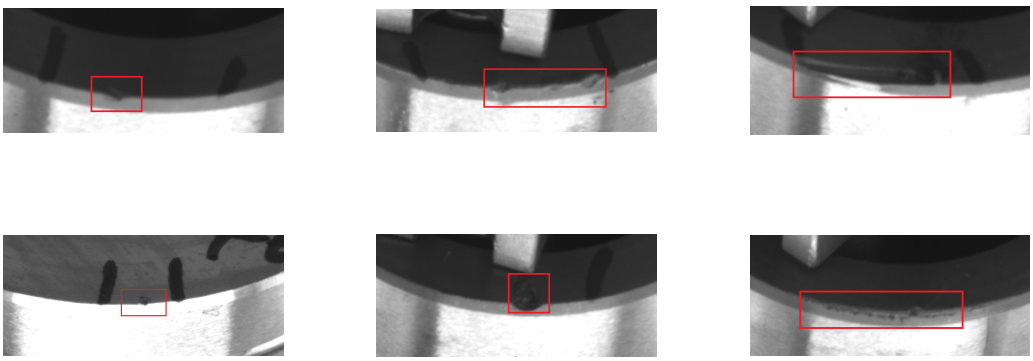


Figure 1.3: Different samples of nick failure defects.

Damaged teeth defects can occur on one or more teeth of the same gear. This defect can also appear in different spots on a tooth. The most common sites in which this type of defect occurs are on the edge of a tooth and along the top land (the topmost surface of the tooth). The existence of a defect on the edge of a gear tooth can cause a misalignment between the gears in the gearbox, which could ultimately damage the gearbox [2]. The first row of Figure 1.4 shows samples of a defect on the edge, while the second row indicates defects which exist along the top land.

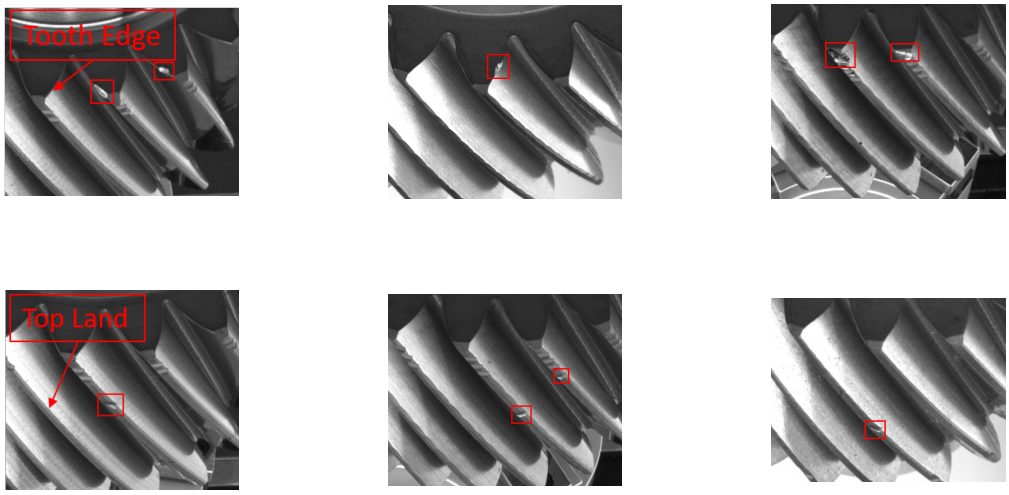


Figure 1.4: Different samples of the damaged teeth defects.

1.3 Thesis Contributions

This thesis has two contributions:

- A new method for detecting nick failure defects on helical gears is developed and tested. The results show excellent accuracy in detecting

nick failure defect.

- A new method for detecting damaged teeth defects on helical gears is developed and tested. The results show excellent accuracy in detecting damaged teeth defects. However, more work is need to reduce instances of false positive.

1.4 Thesis Layout

This thesis is divided into eight chapters. Chapter 2 presents a literature review about vision-based quality inspection systems. Chapter 3 demonstrates the process applied to gear collection. In chapter 4, a classic image processing algorithm is applied to inspect the gear for nick failure defects. In chapter 5, multiple combinations of feature extraction methods and machine learning classifiers are implemented for nick failure defect inspection. In chapter 6, a deep learning model is trained to inspect for nick failure defects with better detection results. In chapter 7, a deep learning model is applied to inspect for damaged teeth on gears. Chapter 8 contains the conclusions of the thesis as well as possible future work.

Chapter 2

Literature Review

Many inspection systems have been installed on factory floors to inspect metal parts produced by manufacturing processes like machining and stamping. Inspection using a machine vision system depends on its ability to distinguish between defective and non-defective manufactured parts. Therefore, various algorithms have been developed to successfully select the defective parts.

2.1 Traditional Image Processing Methods

Various studies have applied traditional image processing methods to the inspection of metals. Moru and Borro [3] developed a calibrated vision system which measures the dimensions of manufactured spur gears. The algorithm that was developed applies subpixel edge detection on the thresholded captured images to define the contours of the gear. The algorithm then calculates the inner and outer diameters of the gear, the number of teeth, and other parameters, such as the pitch diameter, by applying gear equations.

Gadelmawla [4] inspected spur gears by capturing images of the inspected gear positioned on a lighting panel. The illumination assisted in extracting the edges of the gear from the binarized image. The diameter of the gear was then calculated by applying the least square circle method. Liu et al. [5] implemented a multi-camera vision system to inspect bevel gears. The algorithm detected gear defects such as cracks by using a texture analysis of the neighbouring pixels. The diameter of the gear was measured using a circle approximation method.

Borselli et al. [6] developed an algorithm to detect two types of defects in steel products. A Sobel filter and mean brightness binary thresholding were applied separately to the captured image. The algorithm then added the two output images for better edge detection and noise elimination. A fuzzy inference system (FIS) was then applied to classify the type of defect in the image depending on the width and shape of the binarized regions.

Huang and Luo [7] applied Gaussian smoothing in addition to Prewitt eight-directional edge kernels to better detect the edges of the defect. The algorithm then applied morphological operations to the thresholded images for better detection of defects on aluminum strips.

2.2 Classic Machine Learning Methods

Many approaches have identified the defective parts by first extracting features of the defects and then applying machine learning algorithms for classification. Shi et al. [8] proposed and improved on the Sobel algorithm, which operates with eight edge templates to enhance the detection of the edges

of a defect. Next, characteristics of the surface defects on the heavy rails were extracted through the binarization of the area of the defect. A neural network algorithm (NN) was trained and applied to classify the defects.

Wang et al. [9] extracted features of steel surface defects using a histogram of oriented gradient (HOG) in addition to a gray level co-occurrence matrix (GLCM). This method assures the extraction of both the local and global texture features of the defects. A random forest (RF) classifier was applied to the processed images to detect and locate the existence of five different defects.

Wu et al. [10] applied a fast Fourier transform (FFT) to the images of hot rolled strips and extracted features from the center of these images. These images contained information about gray features and geometrical features, which assisted in recognizing defects. Training a neural network with these extracted features resulted in a good classification of the defects as well as a reduction in the detection time.

Liu and Kang [11] extracted feature vectors that depended on the arrangement of grey-level pixels in defective and non-defective areas within images of steel strips. Principal component analysis (PCA) was used to reduce the dimensions of the extracted feature vectors. A feed-forward neural network algorithm was then applied to classify the defective and non-defective parts.

2.3 Deep Learning Methods

In recent years, the use of deep learning methods, such as convolutional neural networks (CNNs), has improved the ability to solve visual inspec-

tion problems in metals [12]. This improvement comes from the fact that CNNs extract features from defects by passing images through a network of filters. As a result, both inspection accuracy and generalization capability have increased when compared to other methods [13].

Zhou et al. [14] proposed a CNN that learns features from both low level and high level pixels for the classification of surface defects on hot-rolled steel sheets. Song et al. [15] proposed an inspection method based on CNN to detect defects on metal screws. As a first check, the captured image was binarized for measuring the contours of the screw. Next, the captured image passed through a CNN for the classification of possible defects.

Wen et al. [16] proposed an algorithm to inspect the circularity of bearing rollers as well as the defects that occur on them. Hough transform method [17] was used to detect the circular contour of the bearing and check for circularity. A CNN then extracted the features of the bearing rollers to classify and locate four defects that commonly occur on bearings.

To improve the feature extraction process, He et al. [18] added a multilevel feature fusion network (MFN) to the feature maps extracted from the CNN. This MFN integrates various hierarchical features into one feature, which contains more information about the location of the defects. A region proposal network (RPN) was included in this algorithm to generate region of interests (ROIs). A detector was then applied to classify and locate defects on these ROIs.

CNNs require a large number of images to create training models in order to acquire good testing results [19]. Some studies have applied different methods to overcome this problem. Yun et al. [20] proposed a data augmentation

method based on a variational autoencoder to create a sufficient amount of training data. The data were then trained with a CNN for the classification of steel surface defects.

Other studies have applied transfer learning as this method uses pretrained models on huge datsets for better feature extraction. Natarajan et al. [21] applied a pretrained CNN with an ImageNet dataset that was used by Deng et al. [22] to detect metal surface defects. Other studies have applied both data augmentation and transfer learning. Zeng et al. [23] applied both methods to detect defects on a steel sheet, while Neuhauser [24] applied these methods to inspect the profiles of extruded aluminum.

While some of the articles reviewed in this survey did use machine vision for quality control of gear production performing automated measurements and inspection, none was applied on helical gears or the type of defects investigated in this thesis. Inspecting helical gears poses different challenges due to their tooth profile.

Chapter 3

The Data Collection Process

Data collection is a vital step in developing a machine learning algorithm. Therefore, a visual data acquisition system was implemented in a previous work by Hall [1] that captured images of the gears that are inspected at the facility. The system was designed to collect images of both defective and non-defective parts. The inspected gear passed through an inspection chamber equipped with a two-camera system (Figure 3.1). These cameras were positioned at both ends of the inspected gear with an approximate distance of 165 mm between the ends and the cameras.

In order to inspect the whole gear, it was mounted on a rotating gripper which allowed the two cameras to fully capture both ends (sections) of the gear. The first camera captured the top section of the gear with an image size of 2048 x 1536 pixels (Figure 3.2). The second camera captured the bottom section with an image size of 2448 x 2048 pixels (Figure 3.3).

Given the fact that defects vary in size, the resolution of the installed cameras was calculated in a previous work by Hall [1] to detect the smallest possible

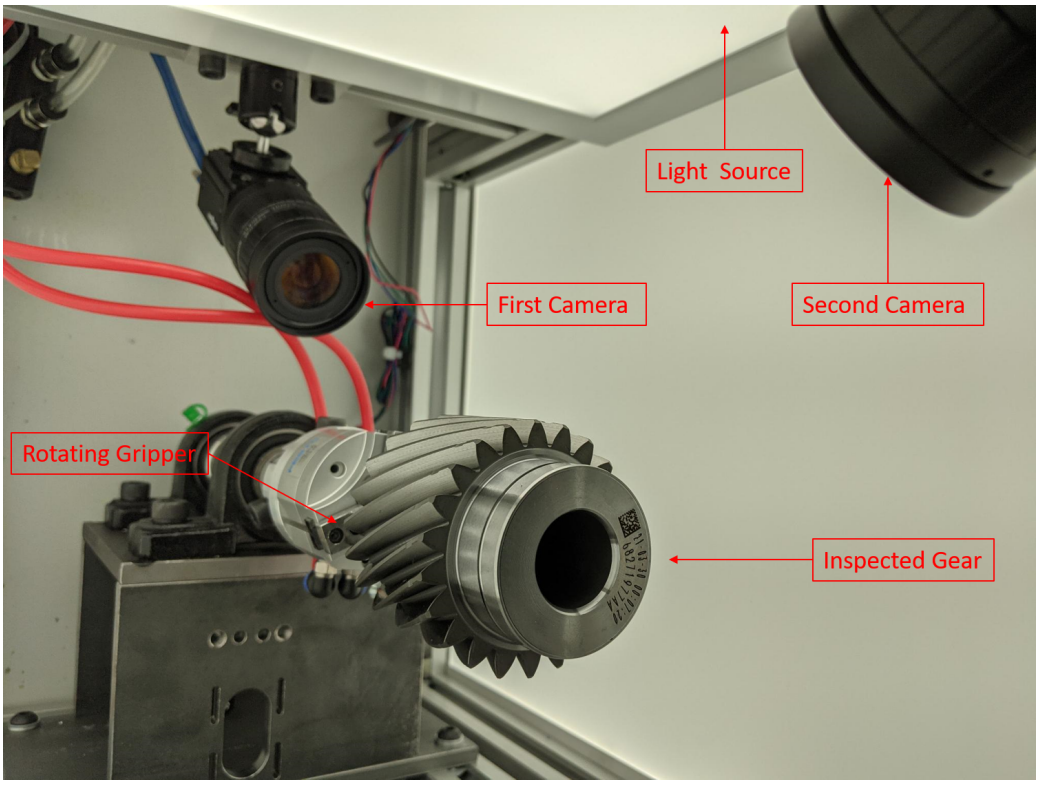


Figure 3.1: Inspection chamber.

defect with a size of 0.42 mm. Therefore, the spatial resolution was set to be 0.05 mm/pixel, which resulted in the smallest defect being covered by around eight pixels. In addition, light panels were installed inside the chamber to ensure the visibility of the defects.

The capturing camera split the gear into vertical sectors equal to the number of teeth on the gear; therefore, the number of images captured by the cameras depended on the number of teeth on the gear. Three types of the gears had configurations with 22 teeth while the remaining type had 26 teeth. As a result, the system captured 44 and 52 images respectively relative to the number of teeth on the gear (two images per tooth, capturing the top and bottom views).

Figure 3.2: Top view of an inspected gear captured by the first camera.

Figure 3.3: Bottom view of an inspected gear captured by the second camera.

Each of the gears had already been inspected by an inspector at the facility and was given a corresponding defect label. In addition, the defects were most likely to be found between two black lines that were drawn by the inspectors to indicate the exact position of the defect. Normally, manufactured gears would not have these lines.

A database was created to store all the information about the collected gears and to save the ground truth labels. Prior to capturing images of a defective gear, the gear type and the defect label had to be fed into the system. In the case of an inspected, non-defective gear, the only input would be the gear type. Once all the images of a gear were captured, the system was available to receive the next gear.

The data collection process started in January 2020 at the manufacturing facility. A total number of 50,036 images captured from 1095 gears were collected. Of these gears, 816 had 21 different defects, while 279 gears had no defects on them. Figure 3.4 indicates the number of gears collected for different defects.

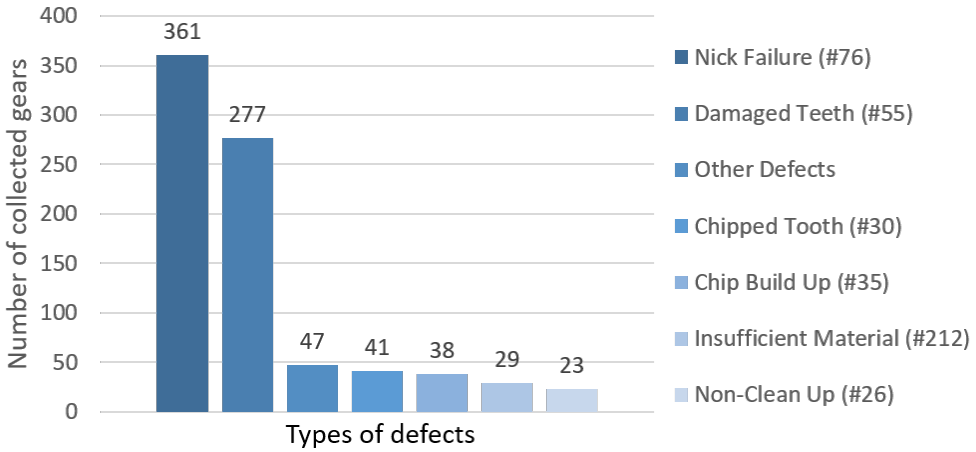


Figure 3.4: Number of gears collected with each defect.

As the inspection in this thesis focused on defects involving nick failure and damaged teeth, more gears containing these defects were collected. This is shown in Figure 3.4. Although more than one defect can be found on a gear, nick failure and damaged teeth did not exist together on any of the gears collected.

Chapter 4

Nick Failure Detection Using a Basic Image Subtraction Algorithm

Nick failure defects tend to have different profiles than the surrounding gear. The hypothesis of this chapter is to use a basic image processing algorithm to detect nick failure defects based on their characteristics. Therefore, the proposed algorithm subtracts the environment and thresholds the image while searching for defects. However, due to a wobble problem within the gripper in the data acquisition process, an alignment step was required prior to applying the algorithm.

Figure 4.1 presents the pipeline for the proposed method. The method began by reading the images of each gear and aligning the pixels in all the images. Next, the mean image of the gear was obtained and intensity values were applied to each image. The full size image was then cropped and the contour areas of the detections in the ROI were calculated and thresholded based on their sizes. After the process had been applied, the output image included the findings.

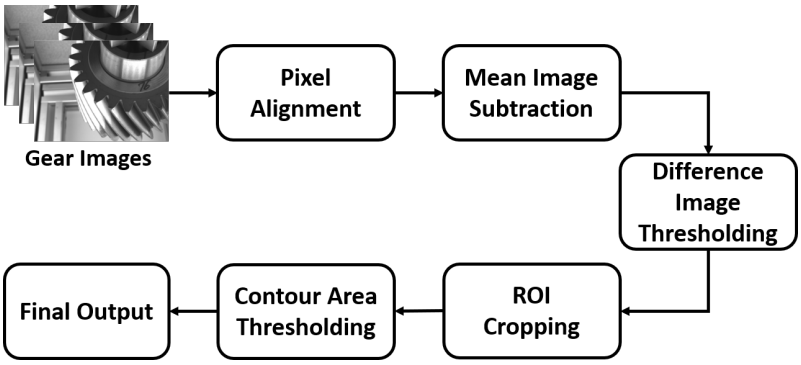


Figure 4.1: Method design pipeline.

4.1 Pixel Alignment

The wobble problem resulted in a shift in the position of the inspected gear in the captured images. The alignment step was therefore necessary to search for a point of interest in the image to perfectly align all images of the gear. As shown in Figure 4.2, the middle rim of the gear exists in all of the images. It was therefore used to detect the point of interest. In order to select the exact rim location, Canny edge detection [25] was applied to the captured image (Figure 4.3).

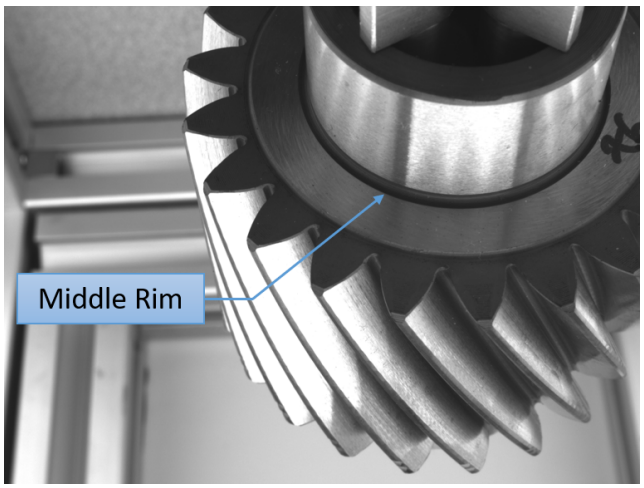


Figure 4.2: Middle rim of the gear.

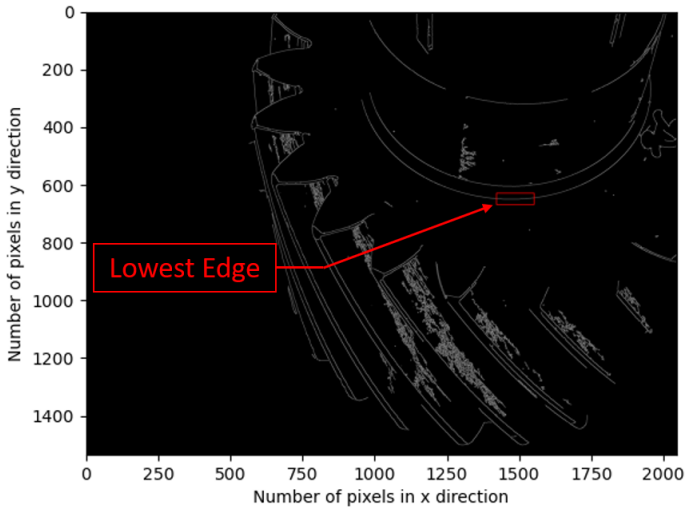
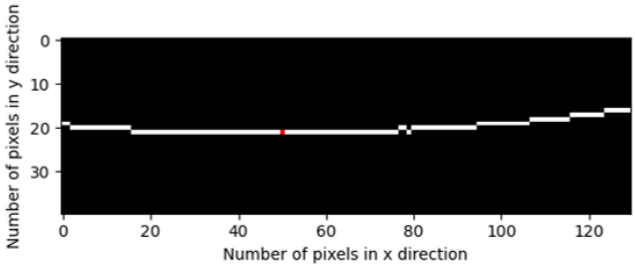


Figure 4.3: Lowest edge of the middle rim in the vertical direction.

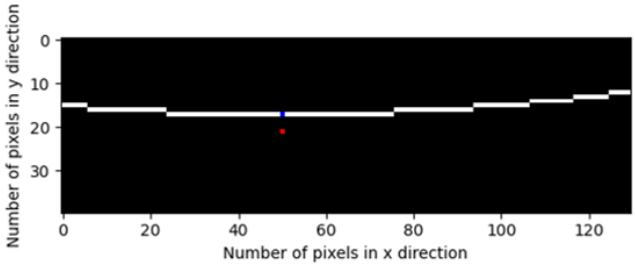
The alignment of the pixels in the vertical (y) direction was applied to the images of the gear. For the first image of the inspected gear, a window was cropped from the full size image using coordinates which ensured that the image contained the lowest edge of the middle rim. The cropped image had a height and width of 40 and 130 pixels respectively. The lowest point of the rim occurred at a pixel number equal to 1470 in the horizontal (x) direction in the full size image (Figure 4.3), and x = 50 on the cropped window (Figure 4.4). The algorithm searched vertically at x = 50 for the pixel number in the y direction that contained the lowest edge of the rim. The y-direction number of that pixel was set as a reference point ($Y_{Reference}$) for the other images of the gear. The shift in the y-direction (Y_{Shift}) of each image of the gear was calculated using Equation 4.1:

$$Y_{Shift} = Y_{Reference} - Y_i \tag{4.1}$$

where (Y_i) is the y-direction value of the lowest edge of the rim in the image,



(a) The reference pixel of the first image is shown in red.



(b) The lowest pixel of the eighth image is shown in blue.

Figure 4.4: Lowest pixels of the rim. In (a) the reference pixel of the first gear image is shown in red, while in (b) the lowest pixel of the eighth image is shown in blue and the reference pixel in red.

and ($Y_{\text{Reference}}$) is the reference point of the first image. Figure 4.4a shows the reference pixel of the first image of the gear in red (y = 21). Figure 4.4b shows the lowest edge pixel of the eighth image in blue (y = 17); therefore, the image was shifted down by four pixels.

4.2 Mean Image Subtraction and Difference Image Thresholding

The next step after alignment was to subtract each image of the gear from the mean image. The mean image contained the mean intensity values of the pixels of all the captured images for one gear. As shown in Figures 4.5 and 4.6, the upper rim of the mean image is sharper after the alignment step.

Figure 4.5: Mean image of the inspected gear before alignment.

Figure 4.6: Mean image of the inspected gear after alignment.

The output difference image included the intensity value differences between the input image and the mean image. Since the difference image contained many changes, intensity values thresholding was applied to reduce the noise. A threshold value of 40 was applied to the difference image in Figure 4.7 to eliminate the noise around the upper rim of the gear. In addition, instead of inspecting the full size image, a region of interest (ROI) was cropped around the upper rim where the nick failure defects occur.

4.3 ROI Cropping

The ROI was cropped using coordinates which ensured that the image contained the defect. As the full rotation of the gear was divided into 22 sectors, and the end shaft diameter was 40 mm, Equation 4.2 was used to calculate the length of each sector:

$$\begin{aligned} \text{Arc Length } &= 2\pi r(\theta/360) \\ &= 2 \times 3.14 \times 20 \times (16.36/360) = 5.7mm \end{aligned} \tag{4.2}$$

where r is the radius of the circle of the end shaft, and θ is the angle subtended by an arc in degrees. Given the fact that the spatial resolution of the camera is 0.05 mm/pixel, the equivalent number of pixels to the arc length was calculated using Equation 4.3 :

$$\begin{aligned} \text{Number of Pixels} &= \frac{\text{Arc Length}}{\text{Spatial Resolution}} \\ &= \frac{5.7mm}{0.05mm/pixel} = 114pixels \end{aligned} \tag{4.3}$$

The width of the ROI was cropped to almost four times wider than the required width to ensure that the defect was found in at least three consecutive images. The ROI was cropped with a height and width of 68 and 475 pixels respectively. Figures 4.7 and 4.8 show the output image with a threshold value of 40 before and after cropping the ROI, respectively.

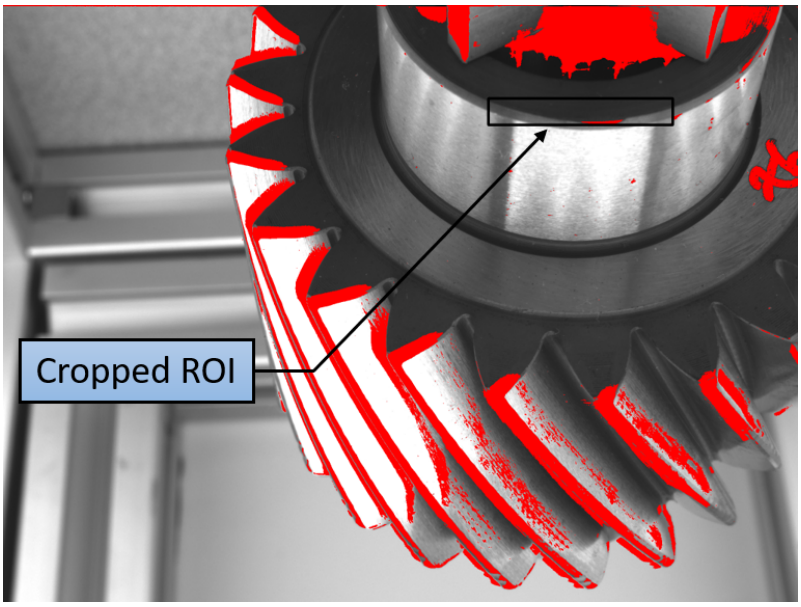


Figure 4.7: The ROI location on the output image.

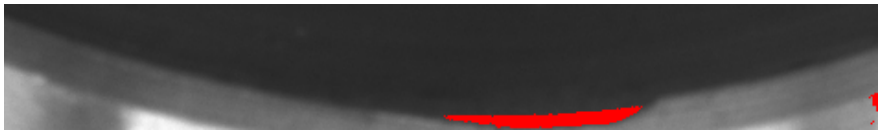


Figure 4.8: The ROI after cropping.

4.4 Contour Area Thresholding and Final Output

The next step was to select the defects from the cropped ROI. However, the thresholded ROI could contain small cluttered areas that might be counted

as defects. Therefore, these cluttered regions were thresholded based on their contour areas and the ROI was converted to a binary image to detect these areas. As seen in Figure 4.9, the binarized ROI contained noise in the white pixels located at the right side of the image. These areas were removed in Figure 4.10 after thresholding the smaller areas; only the bigger areas were considered as possible defects. The final output of the ROI was then determined by the existence of white pixels in the binarized image.

Figure 4.9: Binary image of the ROI before thresholding contour areas.

Figure 4.10: the Binary image of ROI after thresholding contour areas.

4.5 Evaluation of the Method

In order to evaluate the method, each cropped ROI was labeled as either a positive sample or a negative sample depending on the existence of defects in the image. Four metrics were calculated for each cropped ROI after processing and were compared with the ground truth labels. These metrics indicated the numbers of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

In this chapter, a total number of 30 defective gears were chosen as the benchmark test set to evaluate the algorithm. The evaluation took place on

688 images captured by Camera One for those 30 gears. It was noted that the method did not localize the defects but only classified the labeled images.

Precision and recall were calculated to evaluate the performance of the algorithm. Precision refers to the ability of the algorithm to detect images that contain actual defects. It was calculated by dividing the true positives by the sum of the true positives and the false positives (Equation 4.4).

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

Recall refers to the ability of the model to detect all of the images that contain defects. It is calculated by dividing the true positives by the sum of the true positives and the false negatives (Equation 4.5).

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

Precision and recall percentages before and after alignment are shown in Tables 4.1 and 4.2 respectively. This method was applied to intensity threshold values of 40, 50, and 60. In addition, contour area thresholds of 20, 30, 40, and 50 pixels were applied.

(a)

(b)

Figure 4.11: Applying the method on same image before and after alignment. (a) Output image before alignment, and (b) output image after alignment.

Figures 4.11a and 4.11b present examples of the output image of the algorithm when an intensity value threshold of 40 and a contour area threshold of 30 pixels was applied before and after the alignment respectively. The ROI after the alignment showed less noise than before the alignment. As a result, the number of false positives was reduced after the alignment step. This led to an increase in the precision values compared to before the alignment. However, more defects were found by applying the method before the alignment, which led to better recall results.

Table 4.1: Precision and recall percentages before pixel alignment.

Threshold / Contour Area	Thresh.=40	Thresh.=50	Thresh.=60
Area > 20	Precision=33% Recall=77%	Precision=43% Recall=68%	Precision=53% Recall=45%
Area > 30	Precision=34% Recall=72%	Precision=45% Recall=62%	Precision=58% Recall=42%
Area > 40	Precision=36% Recall=70%	Precision=48% Recall=60%	Precision=62% Recall=39%
Area > 50	Precision=38% Recall=67%	Precision=52% Recall=58%	Precision=64% Recall=39%

Table 4.2: Precision and recall percentages after pixel alignment.

Threshold / Contour Area	Thresh.=40	Thresh.=50	Thresh.=60
Area > 20	Precision=41% Recall=76%	Precision=53% Recall=55%	Precision=56% Recall=34%
Area > 30	Precision=42% Recall=66%	Precision=55% Recall=50%	Precision=59% Recall=32%
Area > 40	Precision=45% Recall=64%	Precision=57% Recall=48%	Precision=66% Recall=30%
Area > 50	Precision=46% Recall=62%	Precision=60% Recall=45%	Precision=68% Recall=29%

Applying the method before and after the alignment still resulted in low trade off values between the precision and recall percentages. That was a result of the inability of the algorithm to distinguish between the defective and the non-defective features in the images. Therefore, the algorithms in the next chapters were implemented to overcome the problem of defect recognition.

Chapter 5

Feature Extraction and the Machine Learning Approach

This chapter illustrates how traditional feature extraction was combined with machine learning classifiers to develop and test a method for detecting nick failure defects. Section 5.1 explains the training phase of the machine learning models, while Section 5.2 explains the online inspection phase using these trained models.

5.1 Training Phase for the Method

The training phase pipeline for the method is shown in Figure 5.1. First, the dataset was prepared by labeling the defects on the images as well as extracting the defective and the non-defective patches from the images. Next, three different feature extraction methods were applied to these patches to output feature descriptors. Finally, the descriptors were trained using random forest and SVM machine learning classifiers that were used in method's evaluation.

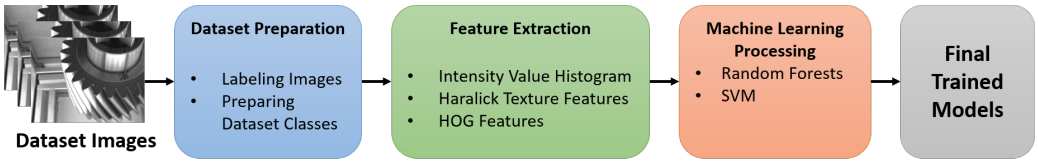


Figure 5.1: Pipeline of the method's training phase.

5.1.1 Dataset Preparation

This step prepared the dataset images for feature extraction. Labeling the images was the first step followed by preparing the dataset patches. This included exporting and resizing the labeled cropped images. Moreover, rotations of the resized images were applied to achieve better training results.

5.1.1.1 Labeling the Images

As the first step in the dataset preparation process, labeling played a crucial role in defining what a defect looks like. The main purpose of this process was to collect samples of defective areas in preparation for training the model. The COCO annotator tool seen in Brooks [26] was used to perform the labeling process.

As mentioned previously, an inspector indicated the presence of a nick failure defect by surrounding it with black lines. Given the fact that these black lines have features similar to some defects, they might be falsely classified as defects by the system. Therefore, these lines were also labeled to avoid the misclassification of gears while testing the model.

Defects and black line categories were considered as two different items to be labeled in the dataset, Different types of labeling tools were used to label the

dataset. Bounding boxes were used to put a perimeter around the defect, while a brush tool was used to mark the black lines. A gear with a nick failure defect between two black lines is shown before and after labeling in Figures 5.2 and 5.3 respectively. After labeling the whole dataset, a JSON file was exported. This file contained the annotation details regarding each image. These details included the image name, the image size, the label segmentation coordinates, and the categories included in the image.

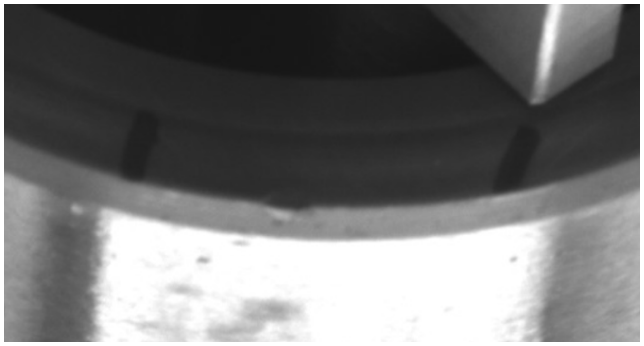


Figure 5.2: Nick failure defect and black lines before labeling.

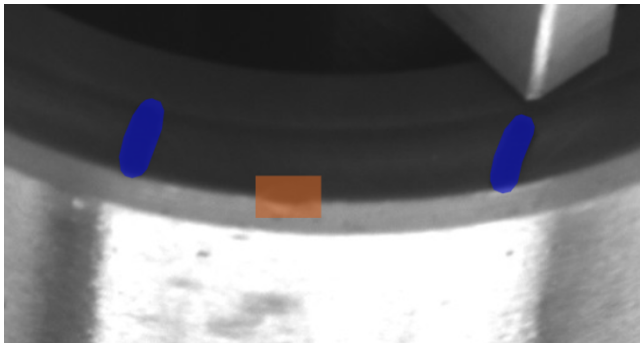


Figure 5.3: Nick failure defect and black lines after labeling.

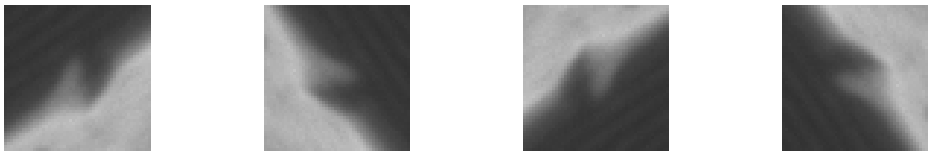
5.1.1.2 Preparing Dataset Classes

In order to distinguish between defective and non-defective areas in images, features had to be extracted from both areas; hence, patches of cropped images were created for defective (positive class) and non-defective (negative class) instances. A positive class indicated the presence of a nick failure defect within the cropped images. These cropped images were generated from full images using the annotation coordinates obtained from the exported JSON file.

The average height and width of the cropped images of the defect were 30 and 41 pixels respectively. As the feature extraction process includes methods which are sensitive to the pixel distribution in the image, all cropped images must be the same size. These images were therefore resized to 50 x 50 pixels to ensure that the defects were covered from all directions. These 50 x 50 pixel images were considered as the positive class patches. Figure 5.4 shows different examples of a a positive class training images.

Figure 5.4: Examples of positive class cropped images.

Given the fact that a nick failure defect can be found at different orientations, positive class cropped images were rotated by 90, 180 and 270 degrees. Adding these rotated cropped images to the dataset of the original cropped images improved learning about defect features and characteristics. Figure 5.5 displays rotated cropped images of a nick failure defect on one of the gears.



(No Rotation) (90° Rotation) (180° Rotation) (270° Rotation)

Figure 5.5: Rotated images of a nick failure defect.

A negative class indicated that the cropped images were free of defects. These images were cropped randomly from the full size images. For greater training accuracy, more images were cropped from the top area of the gear, specifically from the edge of the upper rim where the defect occurs. Figure 5.6 displays examples of negative class.

Figure 5.6: Examples of negative class cropped images.

5.1.2 Feature Extraction

Instead of using the image pixels directly, feature descriptors were extracted and used in training the machine learning models. Due to the variety of nick failure defect shapes and sizes, different features were extracted from the positive class cropped images. Consequently, the same features were extracted from the negative class cropped images. Three different feature extraction methods were applied to both the positive and negative class cropped images. These three methods included the intensity value histogram, the Haralick texture features, and the histogram of oriented gradients (HOG).

5.1.2.1 Intensity Value Histogram

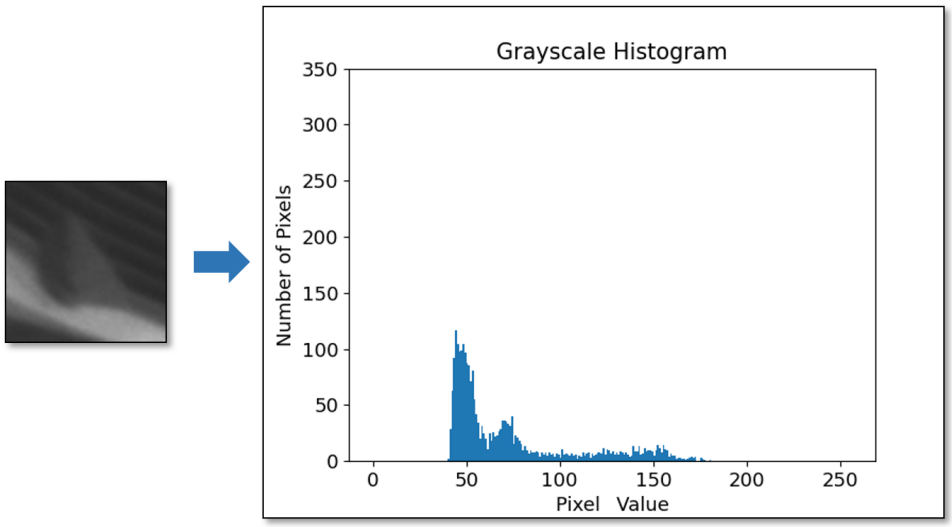
A histogram carries the intensity information of an image by counting the number of pixels at each grayscale pixel value. These values range from 0 to 255, where 0 has weakest intensity (darkest) and 255 has the strongest intensity (lightest).

A defect produces a change in the intensity values of the edge. This results in a defective edge having a different intensity distribution from that of a non-defective edge. Figure 5.7 presents two different nick failure defects and the resulting histograms.

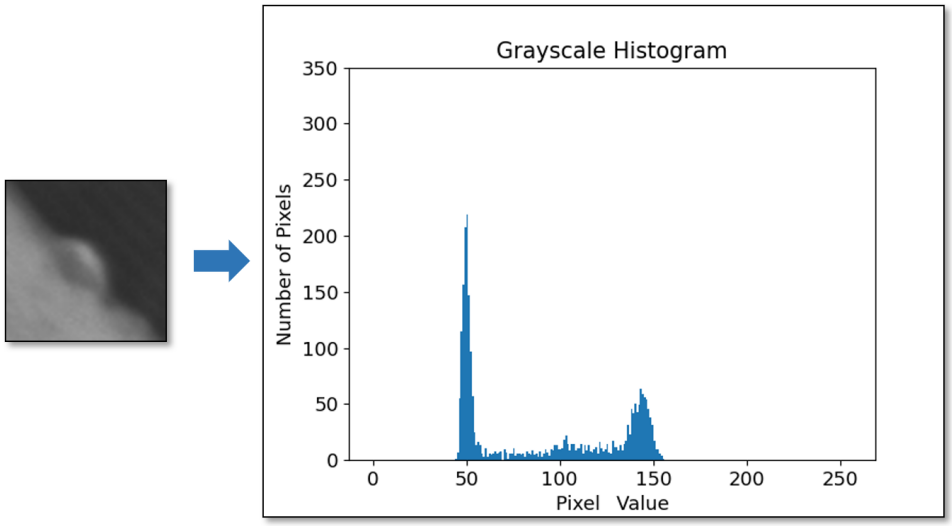
The negative class cropped images consisted of different shades of gray depending on the region from which they came. Cropped images from the edge of the rim have histogram values which are close to that of the defects. However, images chosen randomly from other areas in the full image have different histograms from that of the defect. Figure 5.8 presents two non-defective cropped images in addition to their histograms.

As shown in Figure 5.7, a nick failure defect produces a sudden change in the edge of the rim along the circumference. This change creates different intensity values in the pixels associated with the defect compared to those of the surrounding pixels. Both histograms in the figure have a low number of pixels, roughly between the intensity values of 50 and 150.

Although Figure 5.8a shows a cropped negative image from the edge of the rim, the intensity value distribution is different from that of the defective one. The two peaks existing between the intensity values of 50 and 150 indicate the existence of a large number of pixels at the edges. Specifically, they refer

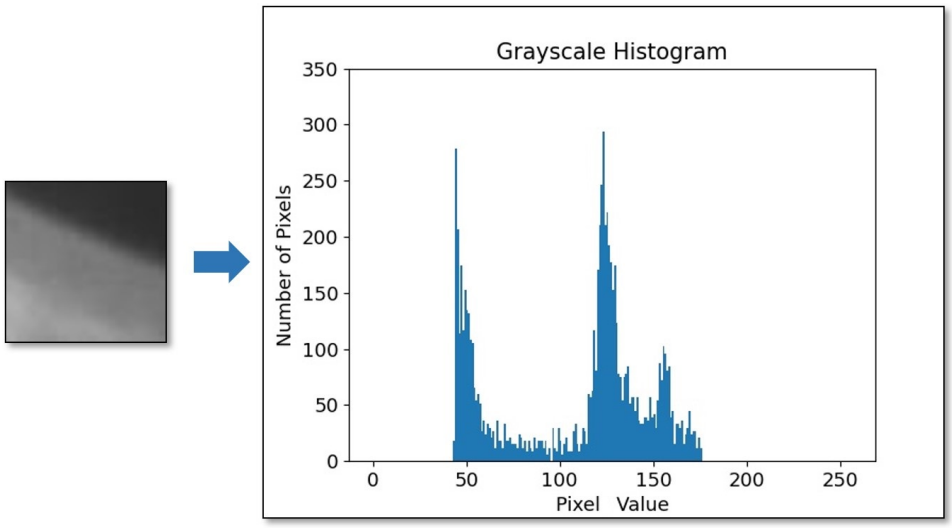


(a) First example of positive class image.

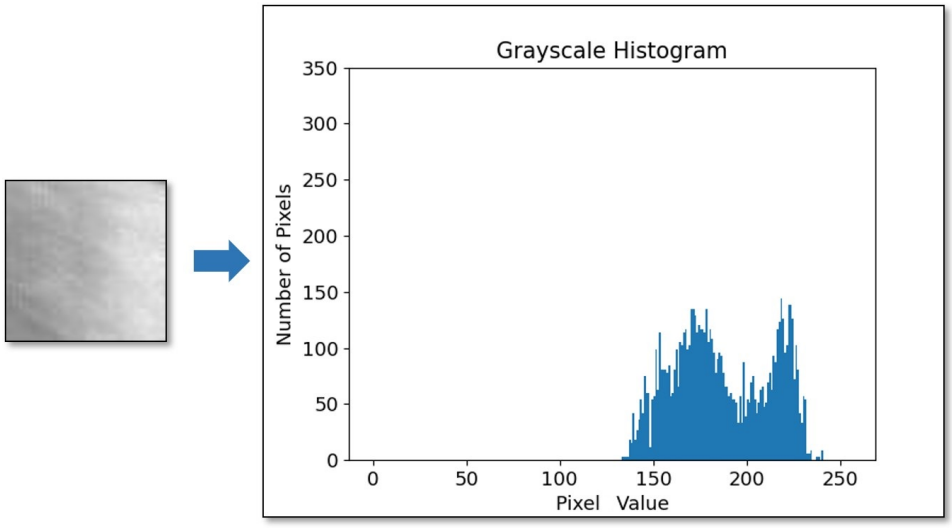


(b) Second example of positive class image.

Figure 5.7: Histograms for two different positive class images with different defect shapes and sizes.



(a) First example of negative class image.



(b) Second example of negative class image.

Figure 5.8: Histograms for negative class images. The image in (a) shows a histogram that was generated from a negative class cropped edge image and has a distribution close to that of a defect, while (b) shows a histogram of a negative class cropped image chosen from an area other than the rim of the gear.

to continuous edges without any interrupting patterns. These peaks did not occur on the images of the defect because the pixel numbers were more evenly distributed over the intensity values. In Figure 5.8b, a random negative class cropped image was selected from a brighter area of the gear. These pixels were distributed over the larger (brighter) intensity pixel values rather than the smaller (darker) intensity values.

5.1.2.2 Haralick Texture Features

To enhance the identification of the characteristics of a defect, the correlations between neighboring pixels were computed. The existence of a nick failure defect on the gear changed the texture between the area of the defect and that of its surroundings. This change was noticeable as the defect caused the texture of the gear to go from a smooth surface to a rough one. Similar to the eyes of a human inspector, Haralick texture features [27] allowed the system to recognize the difference in the texture at the edge of a gear.

A gray level co-occurrence matrix (GLCM) prepared the image for extracting the Haralick features. The matrix recorded how often the intensity values of two adjacent pixels were repeated in an image. The square matrix $\mathbf{G}$ in Equation 5.1 has a size of N_g x N_g, where N_g is the number of gray intensity levels in the image. Each element $[i, j]$ of the matrix $\mathbf{G}$ refers to the number of times that a pixel value of i appears next to a pixel value of j. The matrix $\mathbf{G}$ is then normalized by dividing the value of each element by the sum of all of the element values to create a probability matrix. Figure 5.9 presents the process of preparing the GLCM matrix for extracting the Haralick features.

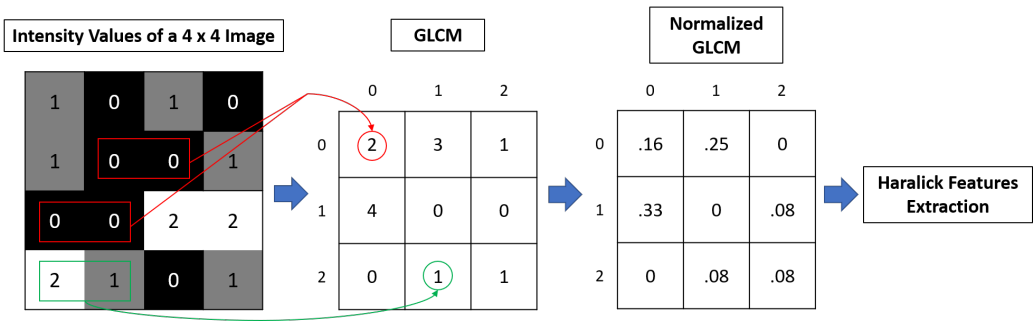


Figure 5.9: The process of constructing a GLCM and extracting the Haralick features. This figure shows an example of a 4 x 4 image with grayscale intensity values from 0 to 2. The GLCM was generated using horizontal direction adjacency. The neighbouring pixel is to the right of the reference pixel. GLCM is then normalized to present the probability of the appearance of two pairs of adjacent pixels. The Haralick features were then extracted from the normalized GLCM.

$$\mathbf{G} = \begin{bmatrix} p(1,1) & p(1,2) & \cdots & p(1,N_g) \\ p(2,1) & p(2,2) & \cdots & p(2,N_g) \\ \vdots & \vdots & \ddots & \vdots \\ p(N_g,1) & p(N_g,2) & \cdots & p(N_g,N_g) \end{bmatrix} \tag{5.1}$$

Since a pair of adjacent pixels can appear in more than one direction in a 2D image, a GLCM matrix was generated for four directions of adjacency. Figure 5.10 indicates vertical, horizontal, right, and left diagonal adjacency directions.

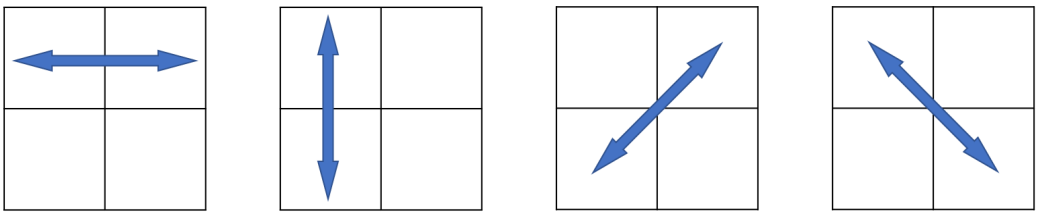


Figure 5.10: Four directions of 2D adjacency between pixels.

The Haralick texture features were then extracted from matrices in each of the four directions. For each direction, thirteen different features were computed from the GLCM matrix of the image. The output of each direction was a 1 x 13 vector which created a 4 x 13 matrix combined for the four directions. In order to reduce the computational work, an average vector size of 1 x 13 was calculated from the 4 x 13 matrix. This vector contained feature information in four directions, which made it invariant to rotation.

5.1.2.3 Histogram of Oriented Gradients

The histogram that was extracted in Section 5.1.2.1 only contained information about the magnitude of the intensity values, which helped to recognize edges and shapes. However, a histogram of oriented gradients (HOG) was used in this section to generate more information about the defects that were seen in the images. HOG is a feature descriptor that is based on generating histograms that represent both the gradient magnitude and the direction of the pixels in the image [28].

Three steps have to be applied to an image to extract the HOG feature descriptor. The pipeline in Figure 5.11 explains the process applied to a 50 x 50 pixel image of a nick failure defect. First, an image gradient was generated from the input image in both the x and y directions. The gradient of the two directions was combined using Equation 5.2. The direction of the gradient was then calculated using Equation 5.3.

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{5.2}$$

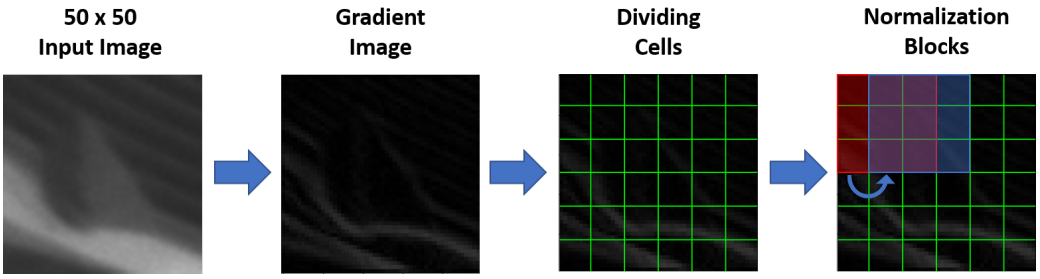


Figure 5.11: Pipeline of a HOG feature descriptor extraction.

$$\theta = \arctan 2\,(G_y, G_x) \tag{5.3}$$

Second, the image was divided into 8 x 8 pixels per cell. A HOG descriptor was constructed using the gradient magnitude $|G|$ and direction θ. Although the image's orientation angles ranged from 0 to 360 degrees, the descriptor used unsigned values, which led to values from 0 to 180 degrees. The angles were divided into nine bins for the final output of the histogram's orientation. Magnitude values are the weights attached to these orientation bins.

Finally, block normalization occurred to output a descriptor which was independent of the illumination changes. Each block consisted of 3 x 3 cells and each cell was shared between multiple blocks. Therefore, the normalizations of the cell were different between the blocks; hence, the cell exists multiple times in the final descriptor with different normalizations.

The output of the process was a HOG feature descriptor which contained magnitude and orientation information about each image. The HOG descriptor was applied to patches of both positive and negative classes of the dataset. Figure 5.12 shows two nick failure defects after extracting the HOG features, while Figure 5.13 shows two examples of images with no defects.

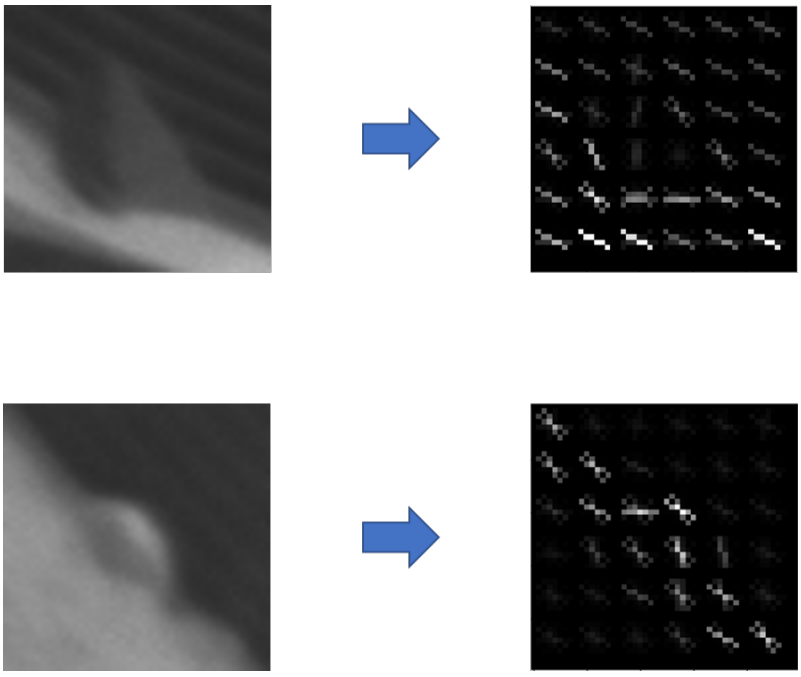


Figure 5.12: Visualization of the HOG features on positive class images.

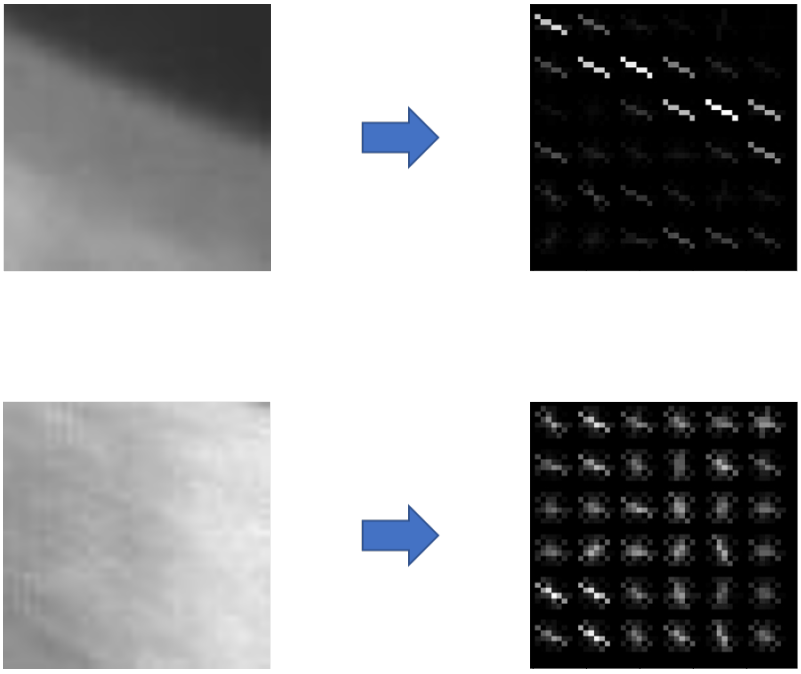


Figure 5.13: Visualization of the HOG features on negative class images.

As seen in Figure 5.13, the edge of the top image, which contains no defects, is represented with similar orientation angles, which creates a pattern of straight parallel lines. However, the two defects in Figure 5.12 construct a disturbance to the straight line pattern of the edge. This occurred due to variations in the orientation angle values of the cells. Randomly selected negative class images generated entirely different patterns than the images of the rim of the gear (bottom image in Figure 5.13).

5.1.2.4 Preparing the Features for Training

Two steps were applied to each image in the dataset in preparation for the training process. First, the intensity value histogram, the Haralick texture features, and the HOG feature were extracted separately in the form of feature descriptors. Second, a label that represented the class characteristics of the image was assigned to each feature descriptor. As the dataset contained only positive and negative classes, labels of 1 and 0 were assigned to them respectively.

A dataframe was then created for each feature extraction method that was applied to the dataset. The columns of the dataframe represent image identification numbers, feature descriptors, and class labels, whereas the rows of the dataframe include the number of images in the dataset.

5.1.3 Machine Learning Process

The training of the machine learning classifiers took place after the extraction and preparation of the features for the positive and negative classes

were done. In the training process, classifiers learned the descriptors of the various features that were extracted from the images. The variety of nick failure defect shapes and sizes in the dataset increased the probability that the classifiers would recognize a defect. The variety improved the ability of the classifiers to generalize and select nick failure defects that they had not encountered previously. Random forests and support vector machines were the two supervised machine learning classifiers used in this section. Both methods were used as binary classifiers, placing the input images into either defective or non-defective classes.

5.1.3.1 Random Forests

A random forest classifier [29] is a supervised machine learning method which consists of multiple decision trees. By combining these decision trees, the weakness of using them individually is avoided and a more powerful classifier is created. As explained in a paper by Breiman [30], the generalization of a forest relies on the correlation strength between the decision trees as well as their own strength.

The word random in the term "random forests" is a representation of how the training datasets are chosen. This randomness is applied using the bootstrap method. A bootstrapped dataset contains randomly selected samples from the original dataset. Each decision tree in the forest is then trained using a random bootstrapped dataset. This means that one sample can be used multiple times in training different trees. As a result of randomly choosing the training datasets, the random forests generate non-biased outputs.

The training direction of a tree goes from the first node to the last leaf in a top-down manner. A random forest classifier with 100 decision trees was used in the applied algorithm. The minimum number of samples required to split a node was selected to be two. Hence, the maximum depth of each tree was decided by either reaching a pure leaf (all samples at the node have the same labels) or until no further splits were possible (less than two samples).

The output of a random forest classifier depends on the vote of the majority of the decision trees in the forest. Figure 5.14 presents the process of training the dataset with a random forest classifier.

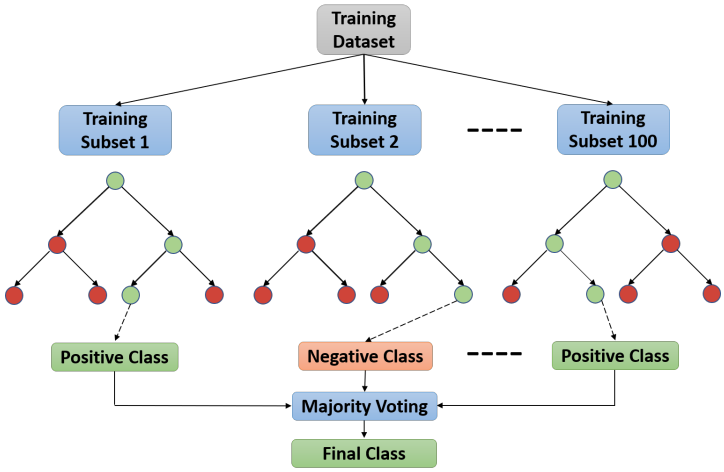


Figure 5.14: Random forest architecture applied to the dataset.

5.1.3.2 Support Vector Machines

A support vector machine (SVM) [31] is the second supervised machine learning classifier used in this chapter. The main task of the algorithm is to create margins between classes in the dataset depending on the feature values. Regarding the dataset in this work, binary classification was applied between the positive and negative classes.

In order to separate two classes, feature points must be linearly separable. However, for more complex datasets, classes cannot be separated using linear margins. Instead, kernels are used to transform non-linearity into linearity. Kernels map the low dimensional data into a higher dimension where it can be separated linearly [31].

Patches of positive and negative classes around the upper rim of the gear had similar features. As this is considered to be the main area where nick failure defects occur, the feature points created non-linear margins between the two classes; hence, a radial basis function (RBF) kernel was applied to create a linear margin.

5.1.3.3 Trained Dataset

A combination of feature extractions was paired with RF and SVM classifiers to determine the best method of identifying nick failure defects within the gears. Training and test sets were constructed from the dataframes that were created for each feature extraction method. An example of this method involved a dataframe that was created by combining both a histogram and the Haralick features, which were trained together. This combination added texture features to the intensity value features of the defects. Training took place on 80% of the dataset images while testing was applied to the other 20%. In order to compare this method with the other approaches in this work, the benchmark test set of 30 gears was reserved for testing these algorithms.

Table 5.1 indicates the total number of images trained with the different models as well as the number of images for the positive and negative classes. All the trained patches had a size of 50 x 50 pixels. The positive class dataset

Table 5.1: Number of patches per model.

Model Name	Hog+SVM	Hog+RF	Histogram+RF	(Histogram +Haralick) +RF
Positive patches	12,236	12,236	12,236	12,236
Negative patches	32,818	32,818	32,818	32,818
Total patches	45,054	45,054	45,054	45,054

included non-rotated cropped images together with images that had been rotated by 90, 180, and 270 degrees. The negative class dataset consisted of patches selected from the full size image of the gear. The focus was on the rim of the gear, which is where nick failures occur the most.

5.2 Inspection Phase for the Method

Once the learning step was completed, the trained classifiers were tested on images that they had not encountered previously. As the test set images had not been seen by the classifiers during the training process, they were used to evaluate the trained models. The pipeline in Figure 5.15 presents the process of inspecting a gear, beginning with capturing the images and ending with printing the predicted class. In this section, the trained classifiers are applied to the test set images in preparation for a performance evaluation.

5.2.1 Preprocessing of the Inspected Images

The pipeline was applied to the test set of gears by processing one gear at a time. The inspection process began with reading the full size images that had

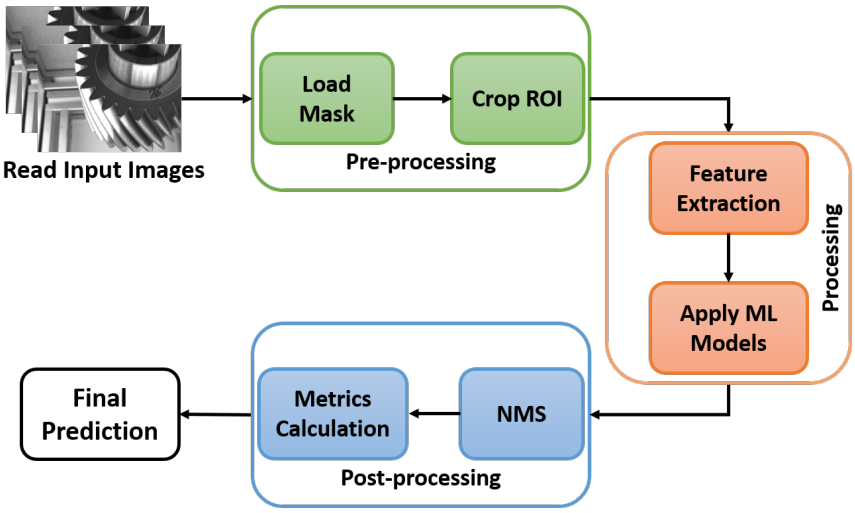


Figure 5.15: Pipeline for the online inspection phase.

been captured for one gear. For each image, the algorithm read the grayscale image as well as the mask assigned to it. This mask was generated from the exported COCO annotations file. For images showing defects, the mask included the background category as well as the segmentation coordinates of both the nick failure defect and the black line marker categories. The masks for the images that did not show any defects included only the background category. Both black line and background categories are considered negative classes, while the nick failure category is the only positive class; hence, the mask for each image represented the ground truth of the image. Figure 5.16 shows a full size test image of a defective gear, while Figure 5.17 presents the corresponding mask for that image. As seen in Figure 5.17, the defect, background, and black line categories have different pixel values in the mask.

Instead of inspecting the full size image of the gear, a region of interest (ROI) was inspected. This region highlighted the area where the nick failure defect occurred. By inspecting only a ROI, the inspection time required for each image was reduced. In addition, this reduced the existence of the false

Figure 5.16: Full size test image of the gear.

Figure 5.17: Full size annotations mask for the test image of the gear.

positive instances that affected the evaluation of the model. The selected ROI focused on the upper rim of the gear, specifically where the defect was clearly visualized (shown in Figure 5.18). The width and height of this area were found to be 460 x 95 pixels respectively.

Since the pixels were shifting between the images in chapter 4, cropping the ROI using the locations of the pixels would not be precise; hence, Hough circles [17] were used to overcome that problem. Hough circles detect circles within the image. As all gears share the middle rim, the Hough circles were tuned to detect it.

In order to crop the ROI highlighted in Figure 5.18, two steps had to be followed. First, a Canny edge detector was applied to threshold the image. This step eliminated the noise that could create Hough circles in the image. Second, a trial and error process was applied to find the range of the Hough circle diameters in which the diameter value of the middle rim was located. After detecting the Hough circle of the middle rim, the x and y coordinates for the center of the circle were calculated. Figure 5.19 indicates the Hough circle in the middle rim of the gear as well as the center of the middle rim.

Given that the distance between the middle rim and the upper rim (where the nick failure defect occurs) is constant in all the gears, an offset value was added to the center of the circle to determine the ROI window. The same ROI coordinates were used to crop the mask of the annotations. Once the ROI of the mask was ready, thresholding was applied to check the different categories in the mask. Figure 5.20 presents the cropped grayscale ROI, while Figure 5.21 shows the corresponding cropped mask for that ROI.

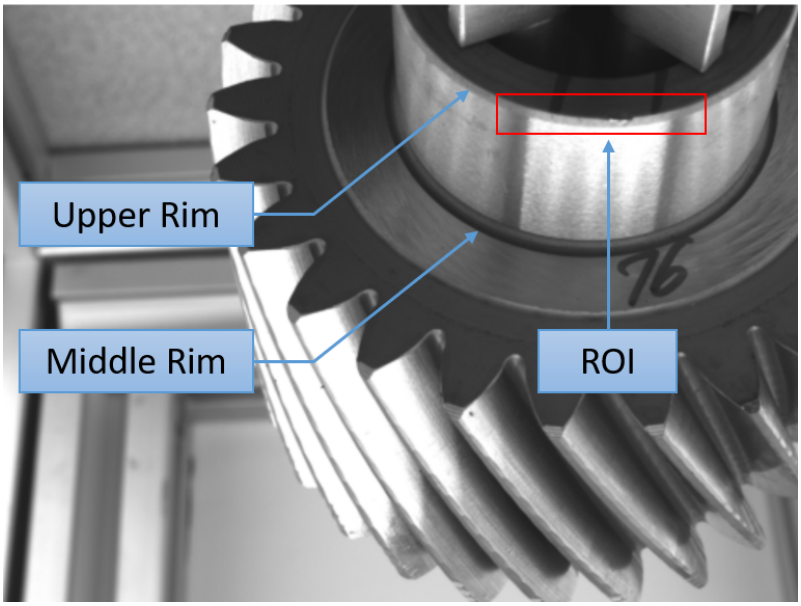


Figure 5.18: The ROI location on the upper rim of the gear.

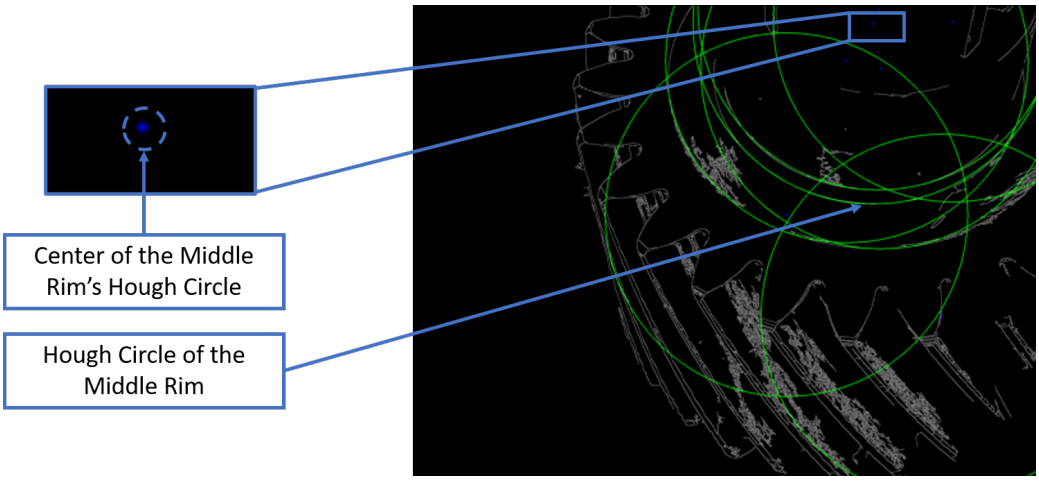


Figure 5.19: Middle rim detection using Hough circles.

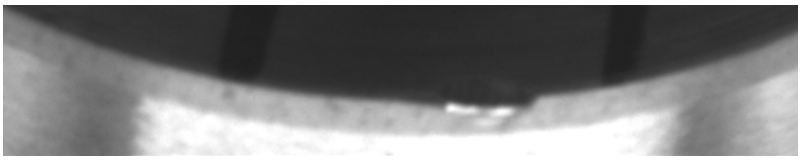


Figure 5.20: ROI for and image of the gear.

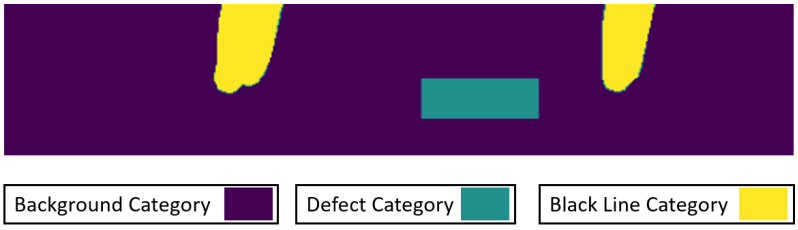


Figure 5.21: ROI mask for an image of the gear.

5.2.2 Feature Extraction and Applying ML Models

As the features were extracted and trained using 50 x 50 pixel images, the inspection needed to occur on an image of the same size. Therefore, the 460 x 95 pixel ROI image was scanned by applying a 50 x 50 pixel sliding window as demonstrated in Figure 5.22. This window acted as the bounding box for the prediction. The window began by sliding horizontally with a step of ten pixels between each movement. Once the first horizontal line was covered for inspection, the window shifted by ten pixels in the vertical direction to scan the next line. The procedure ran until the ROI was fully inspected. At each sliding step, the algorithm extracted the features and predicted an output class for the 50 x 50 inspected bounding box.

Since the classification output is binary, the prediction applied to the classifier was either a positive class (Label 1) or a negative class (Label 0). Positive class refers to the existence of a defect while negative class indicates the existence of the background or black line markers.

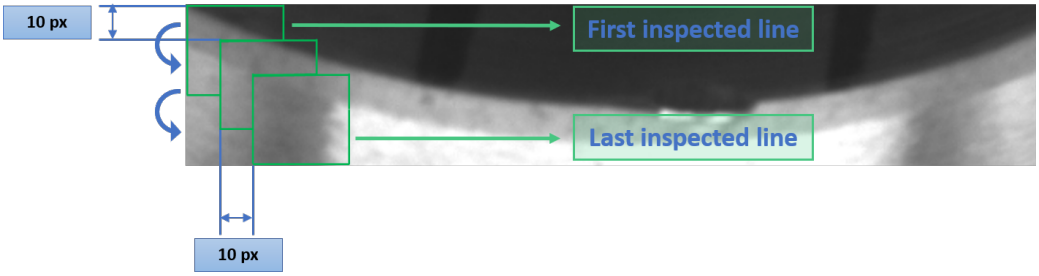


Figure 5.22: Inspection using a 50 x 50 sliding window.

An intersection over union (IOU) method was applied to calculate the intersection between the ground truth bounding boxes and the prediction bounding boxes. The IOU value refers to how close the predicted box is to the ground truth box. In this section, the IOU was set to be greater than zero to increase the probability of detecting the defects. Figure 5.23 shows how the IOU was computed. The numerator contained the area where both the ground truth and the prediction boxes overlapped. The denominator contained the area of the union between the two boxes.

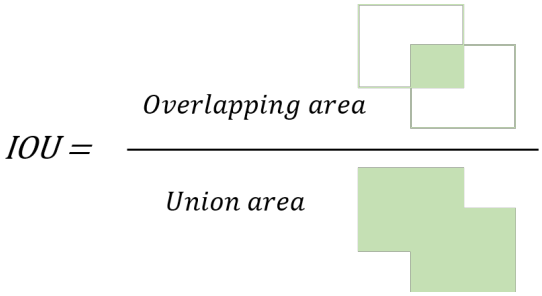


Figure 5.23: Intersection over union computation.

Since the algorithm could mistakenly predict the black lines as a positive class, a checking step had to be applied. If the classifier predicted a 50 x 50 pixel bounding box as a positive class, the coordinates of this bounding box were compared with the corresponding coordinates in the mask. In the case of an intersection with a black line segment, the algorithm turned

this prediction from a positive class to a negative class. However, if the intersection was with a segment from the defect, the prediction would still count as a positive class. Figure 5.24 shows the labeled ground truth defect in the area indicated by the green box, while the blue boxes indicate the prediction of the classifier.

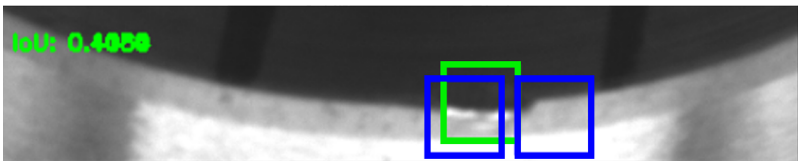


Figure 5.24: Prediction of the classifier on the ROI image.

5.2.3 Post-Processing of the Inspected Images

Once the sliding window for the classifier had inspected the whole ROI image, the final output class of the image had to be calculated. The number of detections as well as the four metrics computed in Chapter 4 had to be generated for each image. Once again, these metrics are true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). They were calculated to be compared with the ground truth labels during the evaluation of the models.

In order to have an accurate count of the metrics, non-maximum suppression (NMS) was applied. NMS eliminates extra prediction boxes from around the defect leaving only one. The NMS threshold was set to choose the prediction bounding box with the largest IOU within the defect. Figure 5.25 shows the ROI image after applying NMS.

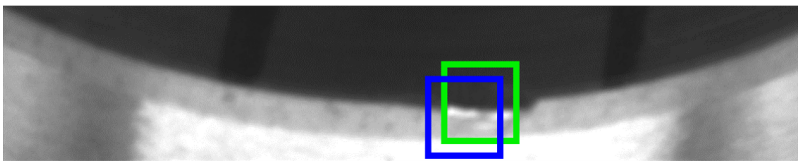


Figure 5.25: Non-maximum suppression applied to the prediction boxes.

5.3 Evaluation of the Method

The evaluation of a model is an indication of how well it performs on previously unseen images. This process requires applying the inspection pipeline seen in Section 5.2 to all the test set images. As most of the defects were captured by Camera One and only a few were captured by Camera Two, the models were evaluated using the images from Camera One for the preliminary results. The performance of the four models was evaluated using 688 images from 30 different defective gears.

Precision and recall were calculated to evaluate the performance of the models. Table 5.2 presents the precision and recall from the four different models applied to the dataset.

Table 5.2: Precision and recall percentages from the evaluation.

Model Name	HOG+SVM	HOG+RF	Histogram +RF	(Histogram +Haralick) +RF
Precision	35%	28%	10%	5%
Recall	80%	50%	20%	27%

As seen from Table 5.2, the model that included the histogram and the Haralick features scored the lowest precision and recall results. These low percentages indicate that the extracted features were not accurately defining the characteristics of the defects; therefore, many true defects were missed

during the inspection. In addition, many instances that have similar features to defects were incorrectly detected as true positives. However, HOG features scored better results than the histogram and Haralick features. This indicates that by extracting the intensity values of the defects as well as the directions, the features of the defects become more distinct. Although close precision values were scored by applying HOG features with both RF and SVM, the SVM classifier had a better recall.

Although the HOG+SVM model scored well on recall, the precision value of this model was close to that of the others. This indicates the challenge of having many instances in the image that contain features which are similar to the defects. Moreover, low precision can also result if the defects are labeled inaccurately. Multiple examples that were labeled by the human inspectors were difficult to detect due to their dissimilarity to actual defects.

Chapter 6

Deep Learning Approach for Nick Failure Defects

Deep learning is widely used in many object recognition applications. Due to its detection robustness and speed, it is implemented in daily industrial quality inspection systems. In this chapter, this approach is applied to nick failure defects.

6.1 Description of the Method

The training and recall pipelines of the deep learning model that was used are shown in Figure 6.1. The dataset images were first preprocessed by labeling the nick failure defects. In the training pipeline, labeled training images were loaded. Next, faster region-based convolutional neural networks (Faster R-CNN) [32]which are an improved version of R-CNN [33], were applied. Faster R-CNN extracts multiple regions of interest from full size images to detect the objects. Features are extracted by applying a feature backbone network

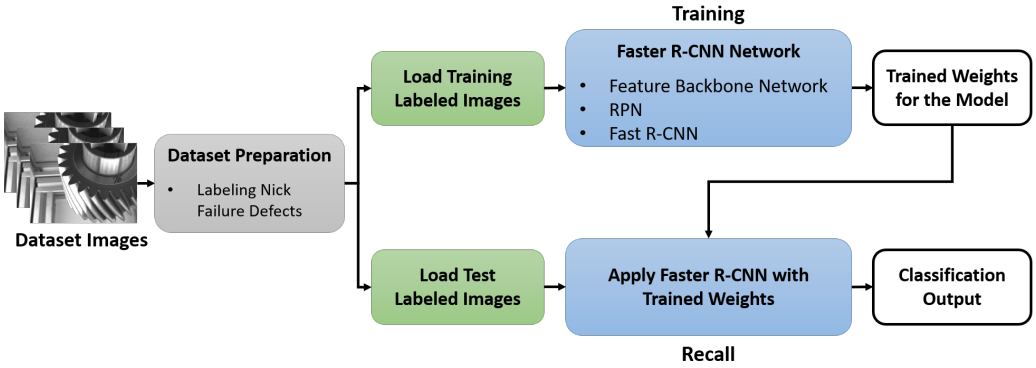


Figure 6.1: Pipeline of the deep learning approach for nick failure defect.

to the images. Fast R-CNN is then used for the binary classification of the defects. Finally, the network outputs the trained weights for the model. In the recall pipeline, labeled test images were fed to the network. The network extracted the features of the objects in the images and applied the trained weights to classify the possible nick failure defects.

6.1.1 Dataset Preparation

The dataset preparation for the deep learning model started with labeling the nick failure defects on the images. The network only required annotations for the positive class instances as it considered other instances on the image to be in the negative class. Therefore, a COCO annotator tool [26] was used to annotate 5633 nick failure defects on 2500 images. These images had been taken by both cameras and were generated from 255 defective gears. Some of the defective gears in the dataset did not have markers surrounding the defects; hence, some instances were labeled as true defects without actually being marked. All the bounding boxes of the annotated defects were saved in a JSON file to be imported to the network as the ground truth boxes.

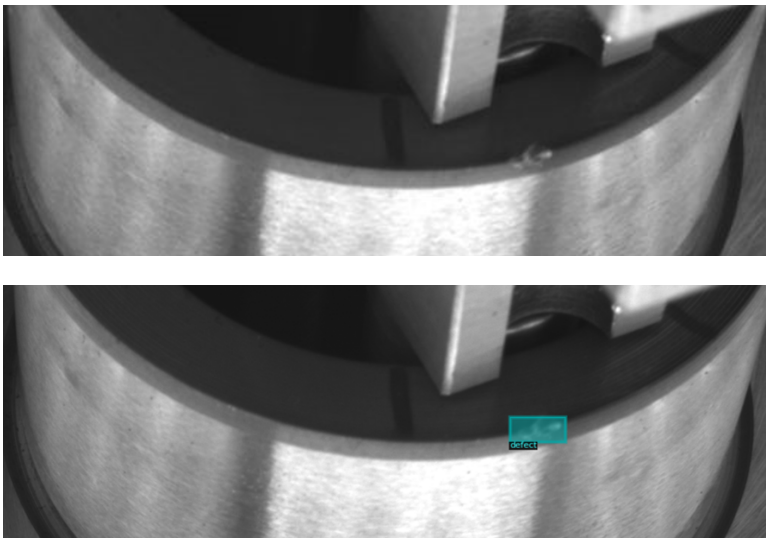


Figure 6.2: A visualization of the nick failure defect bounding boxes on an image from the first camera.

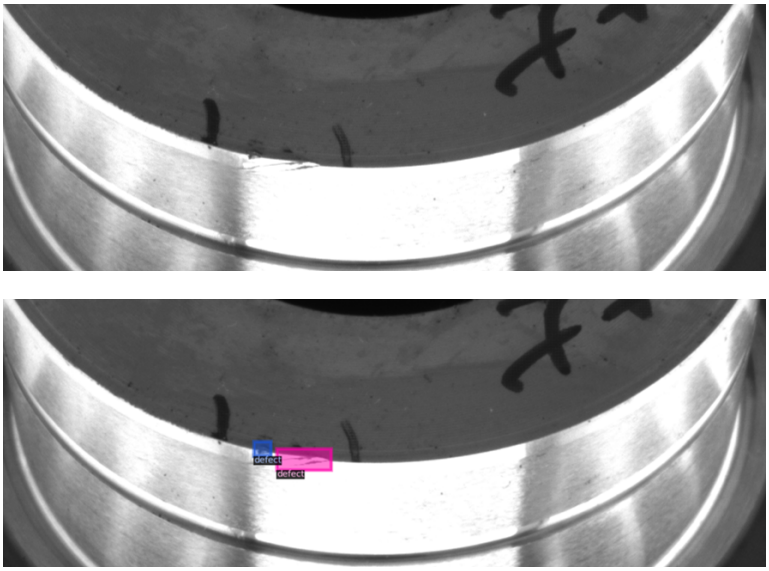


Figure 6.3: A visualization of the nick failure defect bounding boxes on an image from the second camera.

Figures 6.2 and 6.3 show the ground truth annotations on images from Cameras One and Two respectively. Images in both figures were cropped for better presentation.

Dataset images from both Cameras One and Two with sizes of 2048 x 1536 pixels and 2448 x 2048 pixels, respectively, were fed to the network. However, these images did not need to be resized as the network resized the images to different scales while processing.

6.1.2 Faster R-CNN Deep Learning Network

The Faster R-CNN model illustrated in Figure 6.4 consists of three different networks. First, a feature backbone network extracts the features from the images. Second, a region proposal network (RPN) [32] proposes regions of interest on the images. Third, the Fast R-CNN [34] is used for object classification. A Detectron2 framework [35], which was developed by Facebook Research AI, was used to implement the model.

6.1.2.1 Feature Backbone Network

In Chapter 5, features were extracted by manually choosing particular methods that extracted specific information from the images. In the deep learning approach, multiple filters are applied automatically to the images with no prior selection. The applied model uses feature pyramid networks (FPN) [36] to extract features at multiple scales. FPN consists of bottom-up and top-down pathways. A bottom-up pathway uses a residual neural network (ResNet) [37] with 50 layers for feature extraction. ResNet consists of five convolution blocks that each contain a different number of convolution layers. These convolution blocks apply a filter to the input images and down-sampled its dimensions with a stride of two at each block resulting in scales of 1/2, 1/4, 1/8, 1/16, and 1/32 for the five blocks respectively. However, the top-

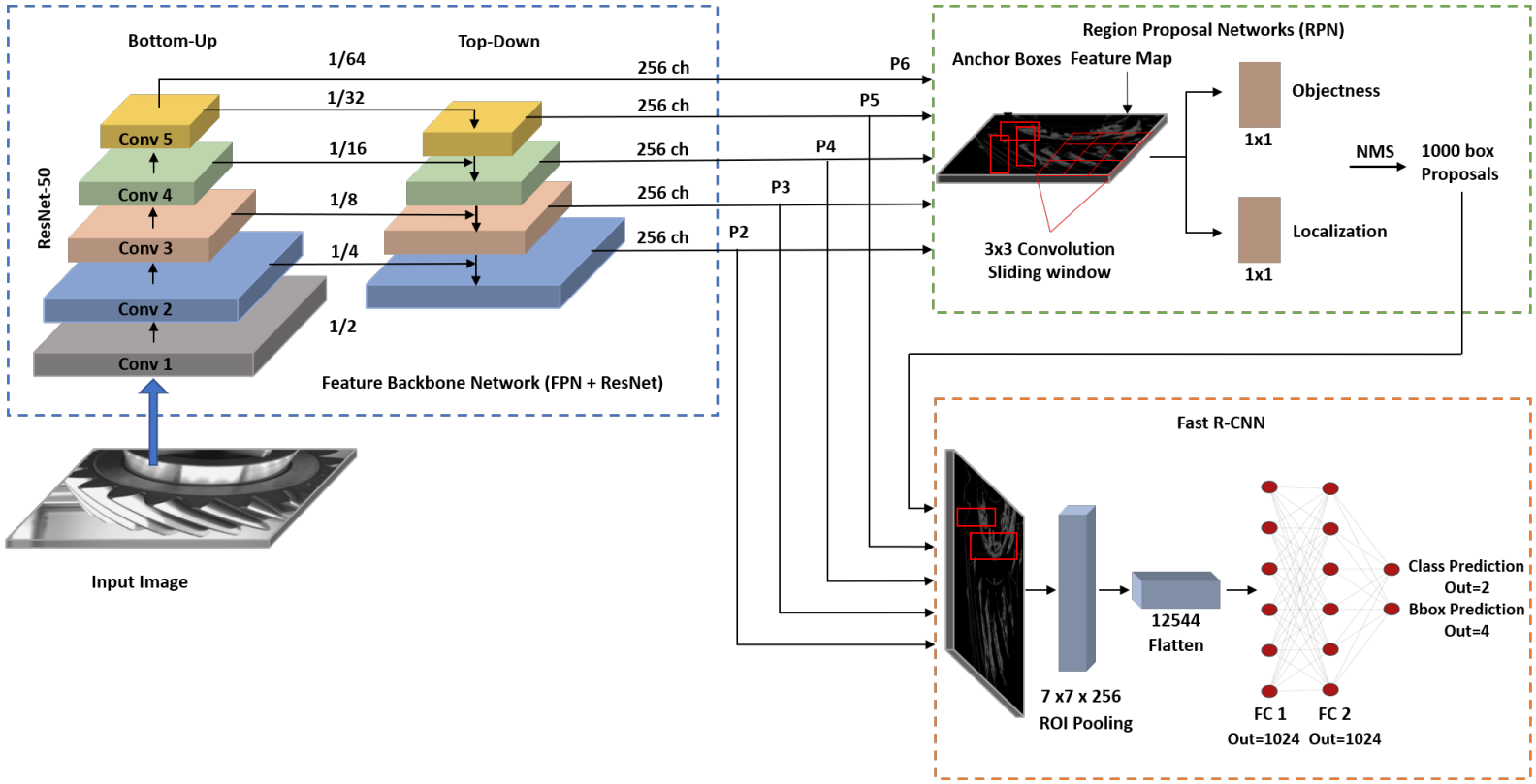


Figure 6.4: Faster R-CNN Deep Learning Network.

down process is not applied to the first convolution block because it has large dimensions. The output features of the remaining four convolution blocks pass through a 1 x 1 lateral convolution layer and return 256 channel feature maps, which are then fed to the top-down pathway. These feature maps are fed to an up-sampler and are also merged with the lateral convolution layer of the previous convolution block. The result of this addition is applied to a 3 x 3 convolution layer, which also keeps the 256 channel feature maps. The output of the top-down pathways is then P5, P4, P3, and P2 feature maps. In addition, a max pooling layer with a kernel=1 and a stride of two down-sample P5 features from a 1/32 scale to a 1/64 scale results in the output of a P6 feature map. Feature maps from P2 to P6 with 256 channels each are then fed to the RPN network. In addition, feature maps from P2 to P5 are fed to the Fast R-CNN network.

6.1.2.2 Region Proposal Networks

In the deep learning approach, a full size image was inspected. Region proposal networks enhanced the search process by sliding a window over the feature maps. Feature map levels P2 to P6 were imported to the network one after the other. For each feature level, a 3 x 3 convolution filter was applied followed by two separate 1 x 1 convolutions for box regression and box classification.

At each sliding window position, a number of proposals were predicted. Anchors were generated with different scales and aspect ratios. An anchor was considered a foreground class ("1") if it had an intersection over union (IoU) greater than 0.7 with regards to the ground truth bounding box. However,

if the IoU of the anchor was less than 0.3, it was considered as a background ("0"); other than that, it would be ignored ("-1").

The majority of the anchors proposed by the network are considered background class. Training the network with more images of the background class than the foreground class will result in labeling imbalance, and it would be hard for the network to learn the foreground class. Therefore, the network balances the two classes by randomly selecting labels from the background class and reduce the number of its labels to be equal to the foreground class.

Objectness indicates whether the network has predicted an object or not, while localization is the prediction of the dimensions of the bounding box. Objectness and localization losses were calculated for training. Binary cross entropy loss was calculated for the objectness loss in the foreground and background classes, while the localization loss was calculated by the L1 loss for the foreground class only. Next, the top scored anchors from the five scales of the feature maps (P2 to P6) were chosen. Finally, a non-maximum suppression was applied to select only 1000 proposal boxes.

6.1.2.3 Fast R-CNN Network

Fast R-CNN uses the outputs of both the RPN and FPN networks for box classification and regression. ROI pooling cropped the rectangles that were proposed by the RPN from the feature maps at the different scales produced by the FPN. The resulting feature map was 7 x 7 x 256, where 7 x 7 is the ROI size and 256 is number of channels. The feature maps were then flattened to 12,544 channels and fed to two fully connected layers, which had an output of 1024 features. After that, the final box prediction was obtained

by two layers. One layer made the output for the object class (two classes, defect and background), and the other made the output for the bounding box dimensions (four coordinates). During training, L1 loss and softmax cross entropy loss functions were used for the localization and the classification losses respectively.

6.1.2.4 Training Parameters

In the training process, only one foreground class was used: the inspected nick failure defect. The model adjusted its weights after feeding the network with batches of four images. The optimizer used in the training was a stochastic gradient descent (SGD) with a learning rate of 0.01. The maximum number of iterations for training the model was 10,000.

6.2 The Preliminary Evaluation of the Method

The purpose of the preliminary evaluation was to test the ability of the deep learning model to recognize the features of the defects. As a result, the evaluation only considered the images of the defect rather than an inspection of the whole gear. The same labeled defects used in Chapter 5 were used in this section for the preliminary evaluation. As the deep learning network requires a lot of data to improve its learning, 90% of the dataset images were used for training. Moreover, a k-fold cross-validation was applied to allow for a better evaluation. In a k-fold cross-validation, the whole dataset is divided into equal sets of images. At each fold, the model is trained with a number of k-1 sets and tested on one set. This method ensures the ability of the model

to generalize images which have not been seen previously.

In this chapter, a ten-fold cross validation was used to divide the dataset into a 90% training set and a 10% test set at each fold. The dataset contained 225 defective gears with a total of 2500 images of defects. Among these images were 5633 defects generated from both Cameras One and Two. The metrics used in the evaluation of the model were precision and recall.

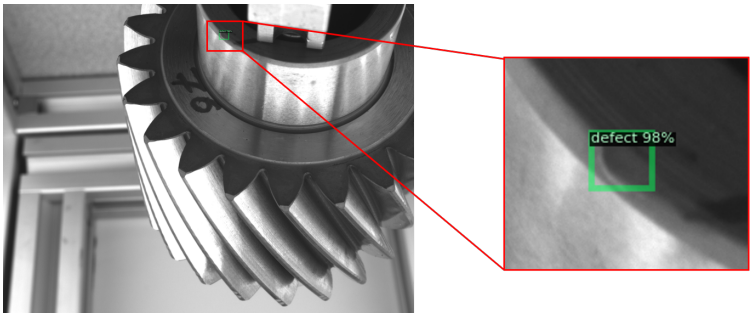


Figure 6.5: The prediction score of the nick failure defect on the Camera One image.

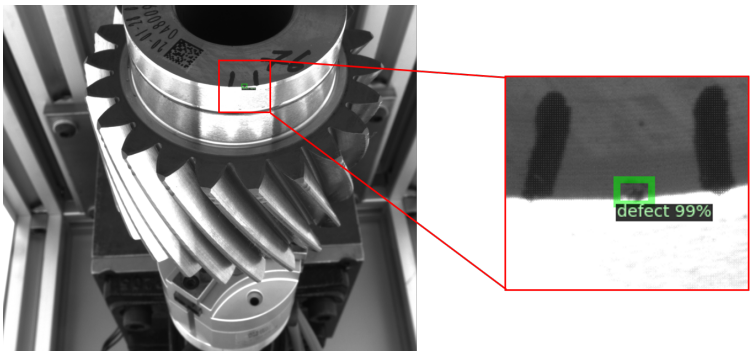


Figure 6.6: The prediction score of the nick failure defect on the Camera Two image.

The prediction score indicated the percentage of the similarity between a detected object and a valid defect. In order for the model to consider a detected object as a defect, its prediction score had to be more than a specific prediction threshold. That threshold created a trade-off relationship between

precision and recall. Setting a low threshold number allowed the model to predict more true defects, although more false positives were also counted. If a higher threshold number was set, it might eliminate some of the false positives, but it would also miss some of the true defects. Figures 6.5 and 6.6 show examples of the prediction scores when detecting defects in the images from Camera One and Camera Two respectively. As shown in both figures, the whole image was inspected instead of only a cropped region of interest.

In order to have a better understanding of the performance of the model, the evaluation took place on different thresholds. Figure 6.7 presents the average precision and recall values of the 10-fold cross validation. The values were generated at prediction thresholds ranging from 75% to 95% in increments of 5% at each step. Precision scored the highest value of 95% at a threshold of 95%, while the lowest value was 80% at a threshold of 75%. Recall scored the highest value of 55% at a threshold of 75% and the lowest value of 26% at a threshold of 95%.

The figure also shows that there is a trade-off between the precision and recall values over the different thresholds. Although the model was able to eliminate many false positives, which increased the detection precision, it was unable to detect many of the ground truth defects; it missed 45% of these defects at even the lowest prediction threshold. Since this detection accuracy is low, the next section illustrates the improvements made to the model.

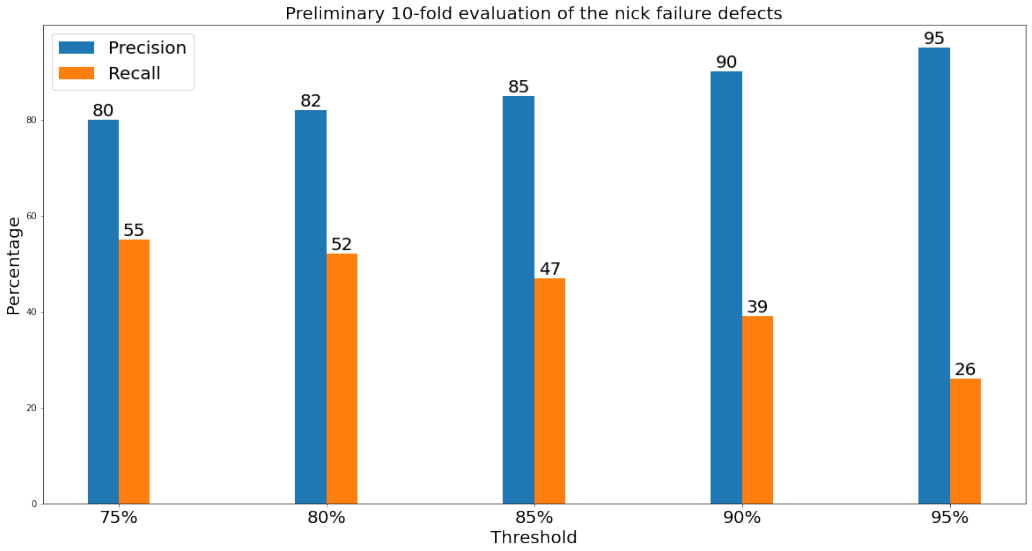


Figure 6.7: The preliminary average precision and recall for nick failure defects using 10 folds.

6.3 Final Evaluation of the Method

The preliminary testing in Section 6.2 resulted in good precision percentages but poor recall results. In this section, an attempt is made to rectify the performance of the model. An enhancement of the recall results was performed by adjusting the defect labeling and feeding the network with more images for training purposes. Instead of using the labels in Section 6.1.1, the labels were modified to ignore small defects based on feedback from the industrial partner.

The evaluation in this section was applied using three different aspects. First, the images of the defects on the gears in the dataset were evaluated by applying a cross validation method. Second, an evaluation was made regarding images of defects from a 30 gear benchmark test set. Third, an evaluation was performed on both the images of the gears from the test set and from a

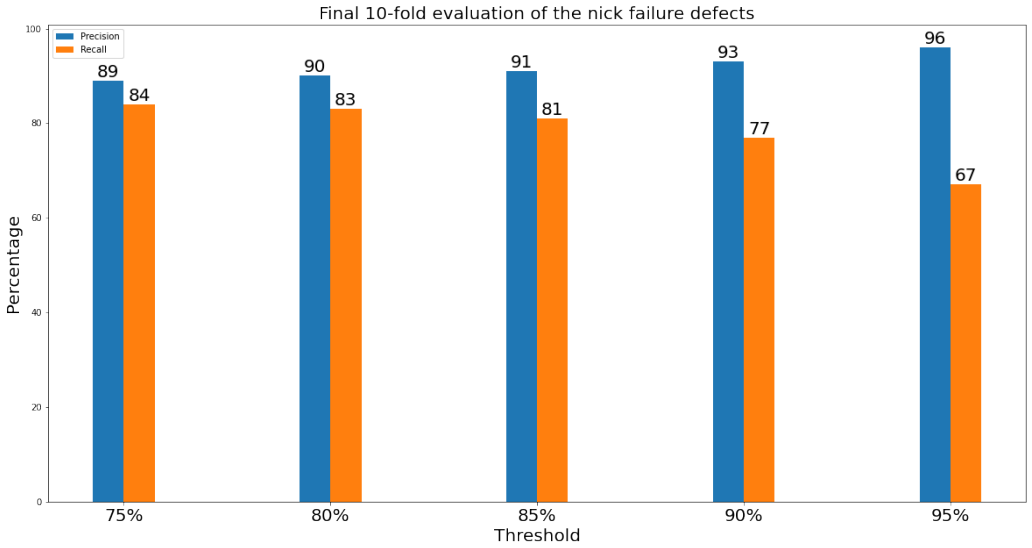


Figure 6.8: Nick failure final average precision and recall for 10 folds.

test set of 100 non-defective gears.

6.3.1 Updated Cross Validation Results for Images of the Defects

The same evaluation method was applied to the images of the defective gears after adjusting the labeling and adding more labeled images. In this dataset, a 10-fold cross validation method was applied to 2909 images of defects captured by Cameras One and Two. These images were generated from 301 defective gears which included 4096 annotated defects. This added 76 more gears with 409 images to the dataset used in the preliminary evaluation. Figure 6.8 shows the final average precision and recall values for the evaluation.

As shown in Figure 6.8, there was a noticeable improvement in the average precision and recall values with more enhancement shown in the latter. At

a prediction threshold of 75%, the model predicted 84% of the ground truth defects and eliminated more false positives with a precision of 89%.

6.3.2 Evaluation of the Images of the Defects from a Benchmark Test Set

Instead of applying one of the models trained in the 10-fold cross validation, a new model was trained to reserve the 30 defective benchmark gears for testing. Testing on these benchmark gears enhanced the comparison between different algorithms. The new model was trained on 2609 images of defects. These images were generated from 271 defective gears out of the total 301 gears used in the 10-fold cross validation. The test set of 30 defective gears consisted of 300 images with 441 annotated defects.

Figure 6.9 shows the precision and recall percentages that resulted from applying the trained model to the 300 images. At a prediction threshold of 75%, the model scored a precision of 88% and a recall of 75%. However, if the prediction threshold was increased, the recall percentages decreased.

6.3.3 Whole Gear Inspection and Industrial Validation

Thus far, the methods have been used to inspect the images and a comparison of their ability to detect defects has been made. In an industrial setting, if the system detects a defect on one image of the gear, the entire gear is sent for manual inspection. In this case, even if several defects were detected, the result would be the same. Therefore, it is important to evaluate the performance of the system in terms of an industrial inspection. This section

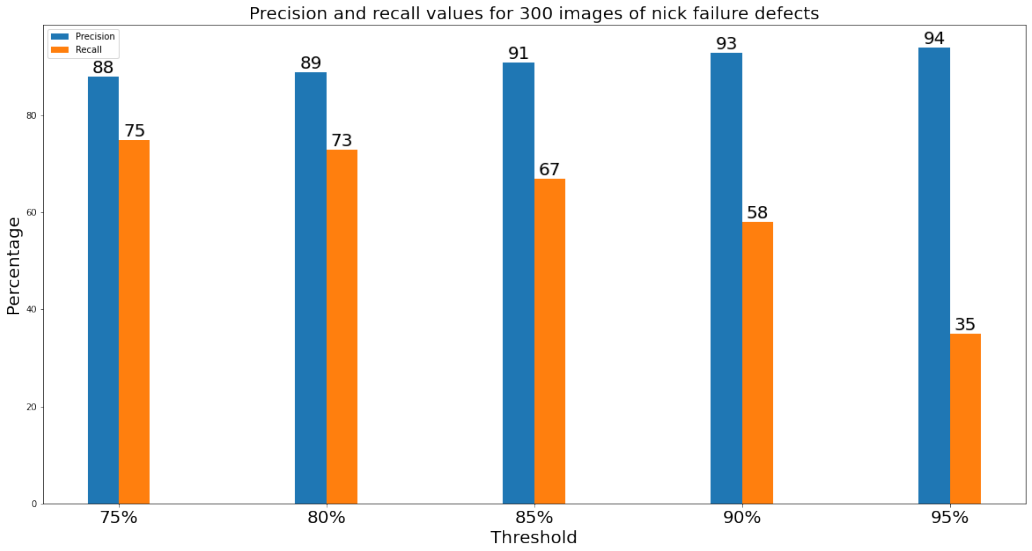


Figure 6.9: Precision and recall values for 300 images of nick failure defects.

therefore evaluates the performance of the model on images of the whole gear. This is done by applying two constraints:

1. The number of images of the defect per gear. This is referred to as the number of images constraint.

2. The number of consecutive images of the defect per gear. This is referred to as the number of consecutive images constraint.

Applying both constraints increases the recall of defective gears and decreases the misclassification of non-defective gears. Although the images of the non-defective gears do not contain defects, it is possible for the model to detect false positives. Accordingly, some of the non-defective gears would be classified as defective and sent for a further manual inspection. For this reason, whole gear inspection was applied to non-defective gears collected from the manufacturing plant to evaluate the accuracy of the model with regards to false positives.

This method was evaluated utilizing the same 30 defective benchmark gears used in Section 6.3.2 as well as 100 non-defective gears. The pipeline in Figure 6.10 illustrates the inspection process for one gear. After all images of the gear were captured by the cameras, the images were fed to the algorithm for inspection. An inspection occurred on all full size images of the gear by applying the deep learning model for the nick failure defect. Next, the number of images constraint and the number of consecutive images constraint were applied separately to images of the whole gear. Finally, the gear was classified as either defective or non-defective based on the output of the constraints.

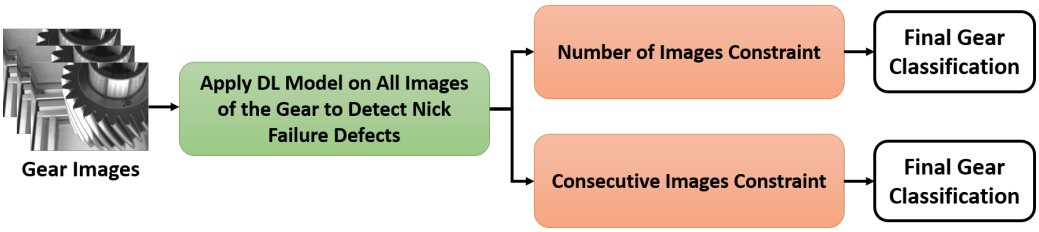


Figure 6.10: Whole gear inspection pipeline for nick failure defects.

6.3.3.1 Applying the Deep Learning Model to All Images of the Gear

The deep learning model that was used to inspect the images of the defects in the benchmark test set in Section 6.3.2 was applied to the images of the benchmark gears. The model created an output for the number of images of defects per gear. The constraints were then applied.

6.3.3.2 Number of Images Constraint

The number of images constraint considered a gear to be defective if a specific number of images were predicted to have defects. The constraint was applied to different setting values, which ranged from detecting a defect on only one image to detecting a defect on four or more images of the gear. Some of the false positives on the non-defective gears would thus be eliminated by increasing the constraint value.

6.3.3.2.1 Applying the Constraint to the Defective Gears

The false negative percentages obtained by applying the deep learning model to the 30 defective gears are presented in Table 6.1. The columns in the table show the constraint values for the number of images which must display the defects to consider the gear as defective. The rows in the table present the different prediction thresholds used to determine if an image displays a defect.

Table 6.1: False negative percentages of 30 gears containing nick failure defects after applying the number of images constraint.

Prediction Threshold	Number of Images >= 1	Number of Images >= 2	Number of Images >= 3	Number of Images >= 4
75%	0%	0%	0%	0%
80%	0%	0%	0%	0%
85%	0%	3.3%	3.3%	3.3%
90%	3.3%	6.7%	6.7%	6.7%
95%	6.7%	13.3%	20%	23.3%

For example, the value of 3.3%, which is at the intersection of "Number of Images >= 2" and the "Prediction Threshold" of 85%, refers to the percentage of the defective gears that were not predicted by the model. As the number of images constraint decreased, the percentage of false negatives also decreased.

6.3.3.2.2 Applying the Constraint to Non-Defective Gears

The false positive percentages for the 100 non-defective gears used for this evaluation are shown in Table 6.2. The false positive percentage decreased with an increase in the number of images required to classify a gear as defective. However, increasing this number led to a reduction in the recall percentage of the defective gears. To solve this problem, another constraint was tested on the gears in next section.

Table 6.2: False positive percentages for 100 non-defective gears after applying the nick failure model and the number of images constraint.

Prediction Threshold	Number of Images >= 1	Number of Images >= 2	Number of Images >= 3	Number of Images >= 4
75%	93%	73%	65%	52%
80%	91%	65%	57%	43%
85%	79%	57%	42%	34%
90%	64%	38%	29%	25%
95%	28%	18%	17%	10%

6.3.3.3 Consecutive Images Constraint

Many false positives occurred in random images of the gear although they did not occur in the previous or the next images. Therefore, a consecutive images constraint was added to consider a gear defective if it had a minimum number of two consecutive images with defects. This constraint was applied using values that ranged from two to four consecutive images.

6.3.3.3.1 Applying the Constraint to Defective Gears

As shown in Table 6.3, the percentage of false negatives increased compared to the previous constraint results in Section 6.3.3.2.1. For example, the false negative percentage of 3.3% at the intersection of "Number of Consecutive Images >= 3" and a "Prediction Threshold" of 80% increased. This was in comparison to a result of 0% for the previous constraint with same prediction threshold and number of images. This increase indicates that the model was able to detect a defect in a gear in more than or equal to three images but that these images were not consecutive.

Table 6.3: False negative percentages after applying the consecutive images constraint to 30 gears with nick failure defects.

Prediction Threshold	Number of Consecutive Images >= 2	Number of Consecutive Images >= 3	Number of Consecutive Images >= 4
75%	0%	0%	3.3%
80%	0%	3.3%	6.7%
85%	3.3%	3.3%	10%
90%	6.7%	13.3%	16.7%
95%	20%	26.7%	36.7%

6.3.3.3.2 Applying the Constraint to Non-Defective Gears

The consecutive images constraint aided in the elimination of false positives on the non-defective gears. As a result, a lower number of non-defective gears would be sent for further manual inspection. As shown in Table 6.4, the false positive percentages of the 100 non-defective gears dropped noticeably. For instance, at the intersection of "Number of Images >= 3" and a "Prediction Threshold" of 80%, the false positive rate dropped from 57% to 36% when the consecutive images constraint was applied. The reduction in the percentage demonstrates that many of the false positives that were randomly detected by the model did not occur in consecutive images.

Table 6.4: False positive percentages after applying the nick failure model and the consecutive images constraint to 100 non-defective gears.

Prediction Threshold	Number of Consecutive Images >= 2	Number of Consecutive Images >= 3	Number of Consecutive Images >= 4
75%	56%	40%	33%
80%	48%	36%	28%
85%	41%	33%	20%
90%	33%	18%	14%
95%	17%	11%	8%

6.3.3.4 Discussion

The deep learning model that was trained to detect nick failure defects scored good results after adding the post-processing constraints. However, these

constraint values must be chosen according to the quality inspection target of the manufacturer. By decreasing the prediction score threshold and the number of consecutive images, most of the defective gears were detected; however, more non-defective gears were classified as defective and sent for further inspection. Nevertheless, all the gears that are currently being produced are inspected by the manufacturer; hence, reducing the number of gears that require inspection result in a faster inspection process.

The best results were generated by setting the prediction threshold and the number of consecutive images to 75% and three images or more, respectively. These inspection parameters led to the detection of all 30 of the defective gears. In addition, only 40 out of the 100 non-defective gears were classified as defective. Therefore, instead of manually inspecting 100 non-defective gears, the system dropped the number by 60%.

Chapter 7

Deep Learning Approach for Damaged Teeth

The second defect that is considered in this work is damaged teeth. As the deep learning model provided a good rate of detection for nick failure defects, it was also used to inspect for damaged teeth. In this chapter, the same deep learning network that was applied to the tests in Chapter 6 was adjusted to inspect for damaged teeth on the gears.

7.1 Description of the Method

A Faster R-CNN [32] deep learning network was applied to both the training and recall pipelines for the damaged teeth defects are shown in Figure 7.1. First, the damaged teeth were labeled on the images of the gears and split into training and test sets. In the training phase, training images were loaded and fed to the Faster R-CNN network. The network chose different regions of interest to automatically extract features from the images. Classification was performed by applying a Fast R-CNN network to the regions of interest. Finally, the training phase network provided an output for the trained

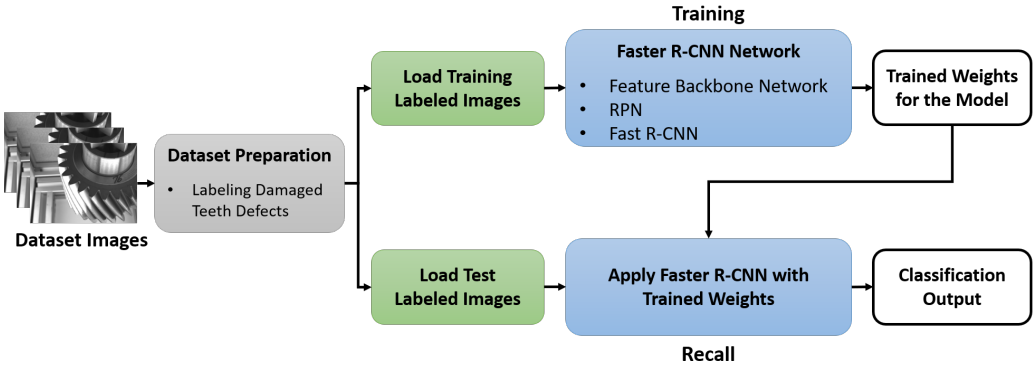


Figure 7.1: Pipeline of the deep learning approach for damaged teeth defect.

weights. In the recall phase, a Faster R-CNN network was applied to the test images to extract the features of the image. Next, the network used the trained weights for the model to classify the extracted features into either a damaged teeth defect or a non-defect.

7.1.1 Dataset Preparation

The COCO annotator tool [26] was used to label damaged teeth by surrounding them with bounding boxes. The network used these bounding boxes as the positive class, whereas the negative class patches were chosen automatically from other parts of the gear. All bounding boxes were saved in a JSON file to be imported to the network as the ground truth boxes. A total number of 3172 defects were annotated in 1711 images. These images were generated from 193 defective gears. This large number of annotated defects was the result of capturing the top and bottom sections of each tooth using the two cameras. Occasionally, the same defect was labeled twice as each camera would capture an image of the same tooth. Figures 7.2 and 7.3 show examples of the labeled defects from the first and second cameras respectively.

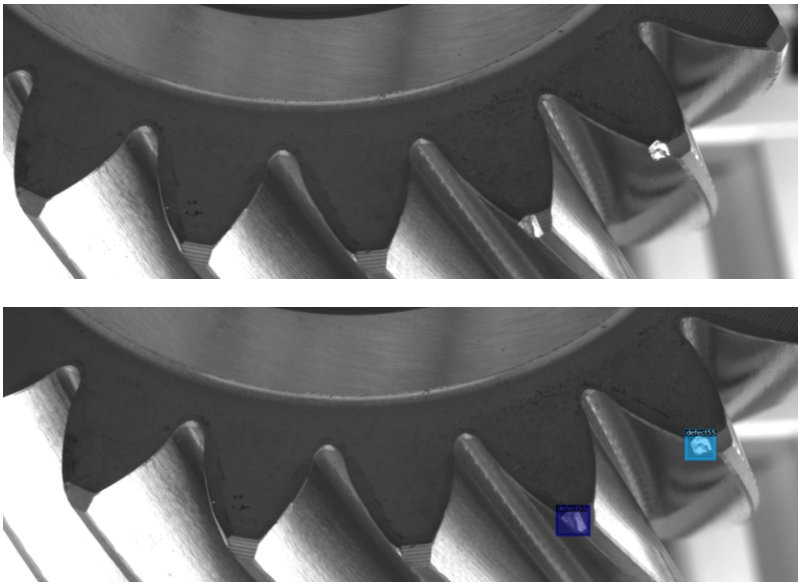


Figure 7.2: Bounding box visualization of the damaged teeth using an image from the first camera.

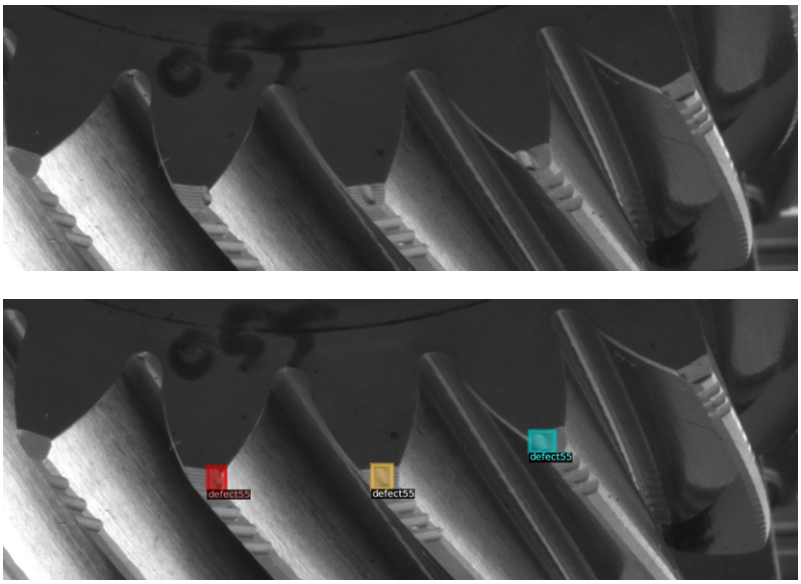


Figure 7.3: Bounding box visualization of the damaged teeth using an image from the second camera.

7.1.2 Faster R-CNN Deep Learning Network

The Faster R-CNN deep learning network consists of the same three networks seen in Section 6.1.2. First, a features backbone network (FPN) [36] extracts features from images at multiple scales. A residual neural network (ResNet) [37] then extracts these features automatically and randomly using 50 layers. Second, a region proposal network (RPN) [32] proposes regions of interest on the full size image. The network slides a window over the feature maps generated from the FPN to classify objects in the image into foreground, background, or ignored classes. Finally, a Fast R-CNN [34] network uses feature maps and regions of interest generated from FPN and RPN respectively for bounding box classification and regression.

The network was trained using only one foreground class, which was the damaged teeth. The model adjusted its weights after feeding the network with batches of four images. Stochastic gradient descent (SGD) was used as an optimizer with a learning rate of 0.01. The network was trained with 10,000 iterations.

7.2 Evaluation of the Method

The evaluation took place using three different phases. First, cross validation was applied to evaluate the method on all the images of the damaged teeth. Second, an evaluation was performed on the images of the defects from a 30 gear benchmark test set. Finally, images of the whole gears were evaluated in the sets of the 30 defective gears and the 100 non-defective gears.

7.2.1 Cross Validation Results for the Images of the Defects

The method was evaluated using a 10-fold cross validation to ensure the ability of the model to perform well on different random test sets. The dataset of 1711 images of defects was used for the 10-fold validation. At each fold, 90% of the images of the defects were used for training while 10% of the images of the defects were used for testing. The training and test sets contained 1540 and 171 images respectively.

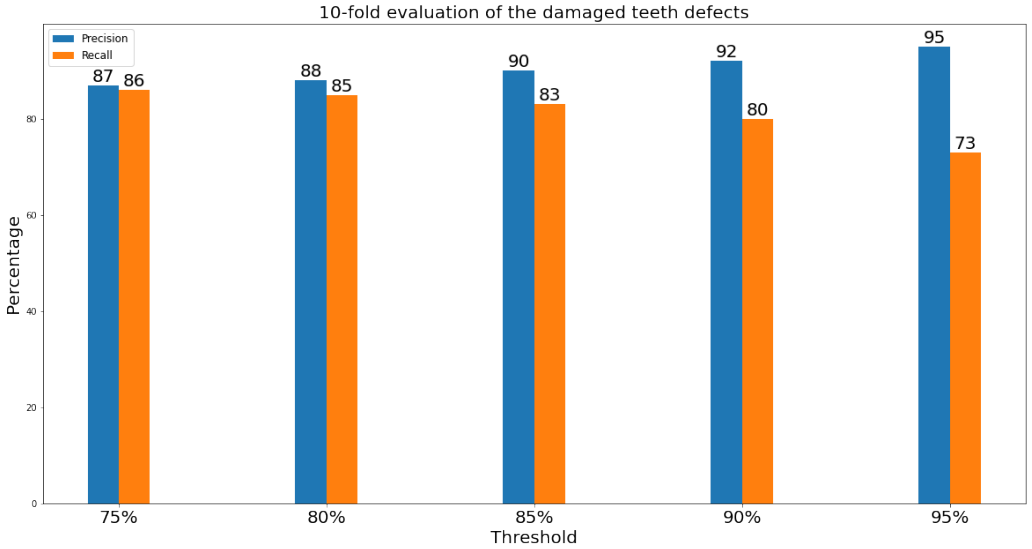


Figure 7.4: Average precision and recall of the 10 folds for the damaged teeth defects.

As shown in Figure 7.4, the model scored well on both precision and recall. The prediction score refers to the percentage of similarity between an object that was detected on the image and a true defect. At a prediction threshold of 75%, the model was able to detect 86% of the defects and maintain a precision of 87%. However, when the prediction threshold was increased, some of the defects were missed but more false positives were eliminated.

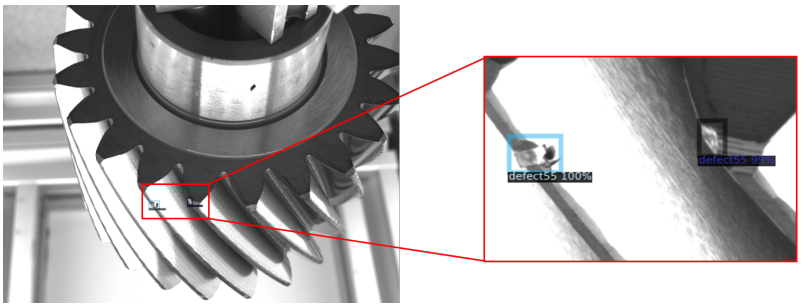


Figure 7.5: Prediction score for a damaged teeth defect on the Camera One image.

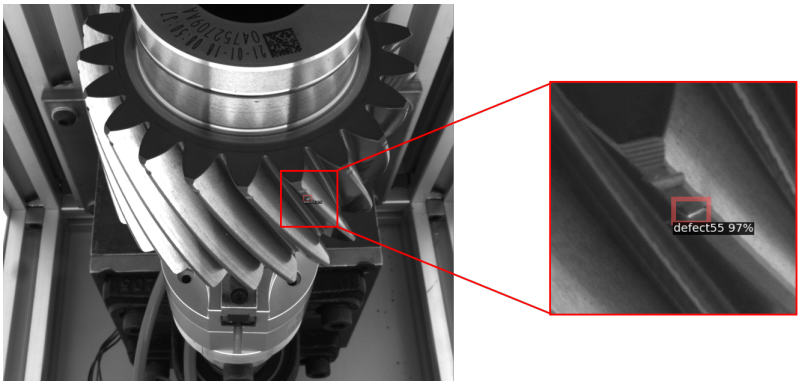


Figure 7.6: Prediction score for a damaged teeth defect on the Camera Two image.

Figures 7.5 and 7.6 present examples of the damaged teeth defects that were detected by Cameras One and Two respectively.

7.2.2 Evaluation of Images of the Defects in a Benchmark Test Set

The damaged teeth model was trained using 1405 images of the defects extracted from 163 gears out of the total of 193 gears used for the 10-fold cross validation. The remaining 30 defective gears were reserved for testing. There were 306 images of the defects from these gears with a total of 554 damaged

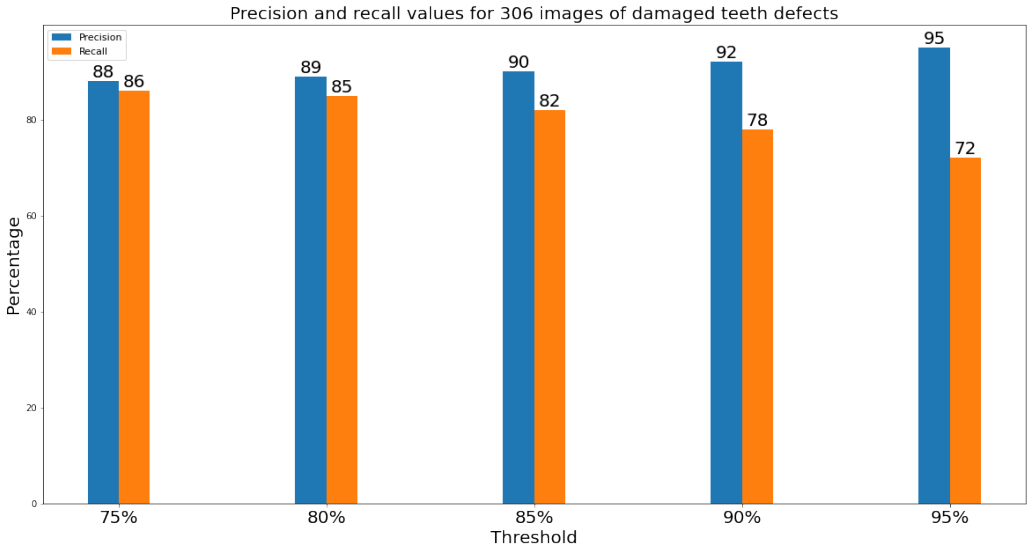


Figure 7.7: Precision and recall values for 306 images of damaged teeth defects.

teeth defects. The precision and recall of the trained deep learning model are shown in Figure 7.7. At a prediction threshold of 75%, the model scored well: 88% and 86% for precision and recall respectively. The model also showed good precision and recall percentages when the prediction score was increased.

7.2.3 Whole Gear Inspection and Industrial Validation

The evaluations in Sections 7.2.1 and 7.2.2 were only performed on images of the defects. However, if the system detected a defect on one image of the gear in an industrial setting, the whole gear is sent for a further manual inspection. Therefore, the evaluation that is performed in this section simulates the inspection of the whole gears that is performed in an industrial setting. This is achieved by applying two constraints:

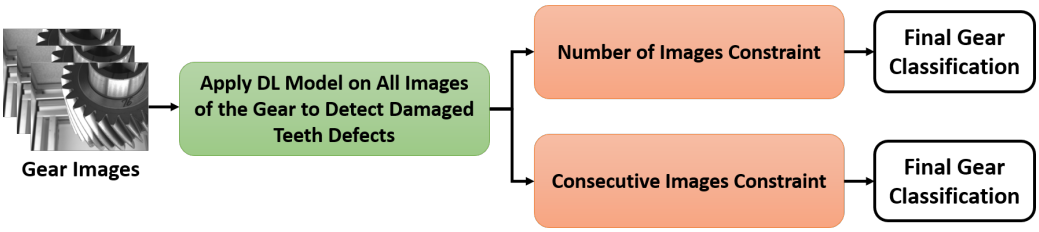


Figure 7.8: Whole gear inspection pipeline for damaged teeth defects.

1. The number of images of the defect per gear. This is referred to as the number of images constraint.

2. The number of consecutive images of the defect per gear. This is referred to as the number of consecutive images constraint.

The pipeline in Figure 7.8 explores the inspection process of the whole gear for the damaged teeth defect. This pipeline was applied to the benchmark test set of 30 defective gears that was used in Section 7.2.2 as well as the 100 non-defective gears inspected in Chapter 6. The process began by importing images of a gear to the network. An inspection was performed on all full size images of the gear by applying the deep learning model for the damaged teeth defect. Next, the constraints regarding the number of images and the number of consecutive images were applied separately to the images of the whole gear. Finally, the gear was classified as either defective or non-defective based on the output of the constraints.

7.2.3.1 Applying the Deep Learning Model to All Images of the Gear

Each image of the gear was inspected by applying the deep learning model which was used to inspect the images in the benchmark test set in Sec-

tion 7.2.2. For each gear, the model provided an output for the number of images of the defects. The constraints were then applied in the next step.

7.2.3.2 Number of Images Constraint

Applying the number of images constraint assisted in eliminating some of the false positives that appeared on the non-defective gears. The constraint considered a gear to be defective if it detected defects in a specific number of images from the gear. The constraint values ranged from detecting a defect in only one image to detecting four or more images.

7.2.3.2.1 Applying the Constraint to the Defective Gears

As shown in Table 7.1, the model was able to detect all 30 of the defective gears by applying the different constraint values and thresholds, with the exception of one element in the table. At a "Prediction Threshold" of 95% and "Number of Images >= 4", only 3.3% of the 30 defective gears were not detected. However, to accurately evaluate the model, an inspection was also performed on the non-defective gears.

7.2.3.2.2 Applying the Constraint to the Non-Defective Gears

Table 7.2 presents the false positive percentages when the model was applied to 100 non-defective gears. Although the model scored well regarding the false negative results on the defective gears, the false positive percentages for the non-defective gears was high. Therefore, the consecutive images constraint was applied in next section to reduce the false positive percentages.

Table 7.1: False negative percentages for the 30 gears containing damaged teeth defects after applying the number of images constraint.

Prediction Threshold	Number of Images >= 1	Number of Images >= 2	Number of Images >= 3	Number of Images >= 4
75%	0%	0%	0%	0%
80%	0%	0%	0%	0%
85%	0%	0%	0%	0%
90%	0%	0%	0%	0%
95%	0%	0%	0%	3.3%

Table 7.2: False positive percentages for 100 non-defective gears after applying the damaged teeth model and the number of images constraint.

Prediction Threshold	Number of Images >= 1	Number of Images >= 2	Number of Images >= 3	Number of Images >= 4
75%	94%	82%	71%	63%
80%	91%	75%	64%	57%
85%	83%	70%	59%	50%
90%	78%	60%	48%	39%
95%	57%	45%	30%	26%

7.2.3.3 Consecutive Images Constraint

The majority of the detected false positives were random instances which had high prediction scores for defects. Many of these false positives occurred on only one image and did not appear on the neighbouring images. Therefore, a consecutive images constraint was applied. This constraint considered a gear to be defective if there was a minimum of two consecutive images of the

defect. The number of consecutive images used in this process ranged from two images to four or more images.

7.2.3.3.1 Applying the Constraint to the Defective Gears

As shown in Table 7.3, the model was able to detect defective gears at high prediction thresholds and a number of consecutive images greater than or equal to three. However, at a number of consecutive images greater than or equal to four, the percentage of false negatives increased compared to the results of the previous constraint.

Table 7.3: False negative percentages found by applying the consecutive images constraint to 30 gears with damaged teeth defects.

Prediction Threshold	Number of Consecutive Images >= 2	Number of Consecutive Images >= 3	Number of Consecutive Images >= 4
75%	0%	0%	6.7%
80%	0%	0%	10%
85%	0%	0%	10%
90%	0%	0%	13.3%
95%	0%	3.3%	23.3%

7.2.3.3.2 Applying the Constraint to the Non-defective Gears

In order to obtain a more precise evaluation of the constraint, it was applied to the 100 non-defective benchmark gears. As seen in Table 7.4, there is a noticeable drop in the false positive percentages compared to the previous constraint results. For example, when a 75% prediction score threshold and a

number of images greater than or equal to three was applied to the model using the previous constraint, the resulting false positives were 71%. However, when the same threshold level with a constraint of three or more consecutive images was applied to the model, the percentage of false positives decreased to 49%.

Table 7.4: False positive percentages when applying the damaged teeth model and the consecutive images constraint to 100 non-defective gears.

Prediction Threshold	Number of Consecutive Images >= 2	Number of Consecutive Images >= 3	Number of Consecutive Images >= 4
75%	69%	49%	34%
80%	60%	44%	32%
85%	57%	40%	28%
90%	48%	34%	24%
95%	36%	24%	14%

7.2.3.4 Discussion

The implemented deep learning model showed high precision and recall percentages when detecting damaged teeth defects in the images. Adding the consecutive images constraint to the model eliminated many of the false positives and scored a high recall on the defective gears. The best results were obtained by setting the prediction threshold and the number of consecutive images to 90% and three images or more, respectively. The system was able to detect all 30 of the defective gears. In addition, only 34 out of the 100 non-defective gears were classified as defective. Instead of manually inspecting 100 non-defective gears, the system dropped the number by 66%.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Various algorithms were developed to inspect manufactured gears for defects. The image subtraction pipeline that was implemented in Chapter 4 showed good recall results regarding the nick failure defect but detected many false positives. Applying that pipeline to the industrial setting would result in many gears being sent for manual inspection. In Chapter 5, the algorithms that were implemented to inspect the gears for nick failure defects using feature extraction and machine learning classifiers showed low precision results. This was the result of the models' inability to distinguish between the characteristics of true and false defects.

However, the deep learning model that was developed in Chapters 6 and 7 was able to extract good features for nick failure and damaged teeth defects, resulting in good precision and recall percentages. Therefore, a further step was taken to evaluate the whole gear instead of only the images of the defects. The consecutive images constraint that was applied to the model eliminated

many of the false positives on the gears. The applied constraint could be set with different values for nick failure and damaged teeth defects. As shown in Table 8.1, the best detection results for nick failure defect were obtained by setting the number of consecutive images to three or more and the prediction score threshold to 75%. Regarding the damaged teeth defect, the best results were generated by setting the same number of consecutive images constraint and prediction score threshold of 90%.

Table 8.1: The consecutive constraint parameters that generated the best results for both nick failure and damaged teeth defects.

Defect Type	Prediction Score Threshold	Number of Consecutive Images	False Negative Percentage	False Positive Percentage
Nick Failure	75%	3	0%	40%
Damaged Teeth	90%	3	0%	34%

There are three sources of errors that could be the reason for the false positive percentages that are shown in Table 8.1. First, the ground truth labeling done by the human inspectors could contain errors due to lack of expertise and human fatigue. Second, some errors could occur during the labeling step using the annotator software. Finally, errors could also exist in the training phase of the deep learning network.

Finally, instead of sending 100% of the non-defective gears for inspection by human inspectors, the model dropped the number to 60% in case of the nick failure defect, and to 66% for the damaged teeth defect. In addition, all gears containing defects were successfully classified as being defective.

8.2 Future Work

The deep learning model that was implemented could be expanded to include an inspection for other defects that exist on the manufactured gears. It would be necessary to capture images of these defects to adjust the model for this inspection. Moreover, by feeding the model with more images of each defect, the performance of the model could be increased.

Bibliography

[1] G. Hall, "A data collection system for future automation applications," Master's thesis, College of Engineering, University of Guelph, 2020, Available: http://hdl.handle.net/10214/17892.

[2] V. Panwar and S. Mogal, "A case study on various defects found in a gear system," *International Research Journal of Engineering and Technology*, vol. 3, no. 3, pp. 425–429, 2015.

[3] D. K. Moru and D. Borro, "A machine vision algorithm for quality control inspection of gears," *The International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 105–123, Jan. 2020.

[4] E. Gadelmawla, "Computer vision algorithms for measurement and inspection of spur gears," *Measurement*, vol. 44, pp. 1669–1678, Nov. 2011.

[5] R. Liu, D. Zhong, H. Lyu, and J. Han, "A bevel gear quality inspection system based on multi-camera vision technology," *Sensors*, vol. 16, p. 1364, Aug. 2016.

[6] A. Borselli, V. Colla, M. Vannucci, and M. Veroli, "A fuzzy inference system applied to defect detection in flat steel production," in *International Conference on Fuzzy Systems*, (Barcelona, Spain), pp. 1–6, IEEE, July 2010.

[7] X.-Q. Huang and X.-B. Luo, "A real-time algorithm for aluminum surface defect extraction on non-uniform image from CCD camera," in *2014 International Conference on Machine Learning and Cybernetics*, (Lanzhou, China), pp. 556–561, IEEE, July 2014.

[8] T. Shi, J.-y. Kong, X.-d. Wang, Z. Liu, and G. Zheng, "Improved Sobel algorithm for defect detection of rail surfaces with enhanced efficiency and accuracy," *Journal of Central South University*, vol. 23, pp. 2867–2875, Nov. 2016.

[9] Y. Wang, H. Xia, X. Yuan, L. Li, and B. Sun, "Distributed defect recognition on steel surfaces using an improved random forest algorithm with optimal multi-feature-set fusion," *Multimedia Tools and Applications*, vol. 77, pp. 16741–16770, July 2018.

[10] G. Wu, H. Zhang, X. Sun, J. Xu, and K. Xu, "A bran-new feature extraction method and its application to surface defect recognition of hot rolled strips," in *2007 IEEE International Conference on Automation and Logistics*, (Jinan, China), pp. 2069–2074, IEEE, Aug. 2007.

[11] G.-W. Kang and H.-B. Liu, "Surface defects inspection of cold rolled strips based on neural network," in *2005 International Conference on Machine Learning and Cybernetics*, (Guangzhou, China), pp. 5034–5037 Vol. 8, IEEE, 2005.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[13] Z. Liu, B. Yang, G. Duan, and J. Tan, "Visual defect inspection of metal part surface via deformable convolution and concatenate feature

pyramid neural networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 9681–9694, Dec. 2020.

[14] S. Zhou, Y. Chen, D. Zhang, J. Xie, and Y. Zhou, "Classification of surface defects on steel sheet using convolutional neural networks," *Materials and Technology*, vol. 51, pp. 123–131, Feb. 2017.

[15] L. Song, X. Li, Y. Yang, X. Zhu, Q. Guo, and H. Yang, "Detection of micro-defects on metal screw surfaces based on deep convolutional neural networks," *Sensors*, vol. 18, p. 3709, Oct. 2018.

[16] S. Wen, Z. Chen, and C. Li, "Vision-based surface inspection system for bearing rollers using convolutional neural networks," *Applied Sciences*, vol. 8, p. 2565, Dec. 2018.

[17] P. V. Hough, "Method and means for recognizing complex patterns," Dec. 18 1962. US Patent 3,069,654.

[18] Y. He, K. Song, Q. Meng, and Y. Yan, "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 1493–1504, Apr. 2020.

[19] Q. Luo, X. Fang, J. Su, J. Zhou, B. Zhou, C. Yang, L. Liu, W. Gui, and L. Tian, "Automated visual defect classification for flat steel surface: A survey," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 9329–9349, Dec. 2020.

[20] J. P. Yun, W. C. Shin, G. Koo, M. S. Kim, C. Lee, and S. J. Lee, "Automated defect inspection system for metal surfaces based on deep learning and data augmentation," *Journal of Manufacturing Systems*, vol. 55, pp. 317–324, Apr. 2020.

[21] V. Natarajan, T.-Y. Hung, S. Vaikundam, and L.-T. Chia, "Convolutional networks for voting-based anomaly classification in metal surface inspection," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, (Toronto, ON), pp. 986–991, IEEE, Mar. 2017.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, (Miami, FL), pp. 248–255, IEEE, June 2009.

[23] W. Zeng, Z. You, M. Huang, Z. Kong, Y. Yu, and X. Le, "Steel sheet defect detection based on deep learning method," in *2019 Tenth International Conference on Intelligent Control and Information Processing (ICICIP)*, (Marrakesh, Morocco), pp. 152–157, IEEE, Dec. 2019.

[24] F. M. Neuhauser, G. Bachmann, and P. Hora, "Surface defect classification and detection on extruded aluminum profiles using convolutional neural networks," *International Journal of Material Forming*, vol. 13, pp. 591–603, July 2020.

[25] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679 – 698, 12 1986.

[26] J. Brooks, "COCO annotator." `https://github.com/jsbroks/coco-annotator/`, 2019.

[27] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 610–621, Nov. 1973.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, (San Diego, CA, USA), pp. 886–893, IEEE, 2005.

[29] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278–282, IEEE, 1995.

[30] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, Sept. 1995.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *arXiv:1506.01497 [cs]*, Jan. 2016.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv:1311.2524 [cs]*, Oct. 2014.

[34] R. Girshick, "Fast R-CNN," *arXiv:1504.08083 [cs]*, Sept. 2015.

[35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[36] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *arXiv:1612.03144 [cs]*, Apr. 2017. arXiv: 1612.03144.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), pp. 770–778, IEEE, June 2016.

Appendix A

List of Gear Defects

Defect Label	Defect Description	Defect Label	Defect Description	Defect Label	Defect Description
1	Broken Tap	37	Missed Operation/Missed Component	73	Profile Error
2	Broken Drill	38	Program Error	74	Index Error
3	Broken Insert	39	Wrong Material	75	Pitch Runout
4	Broken Tool (other)	40	Machine Failure	76	Nick Failure
5	Dimension O/S	41	Incorrect Offset	77	Grinding Burn
6	Dimension U/S	42	Assembly Error	78	Gauge Failure
7	Diameter O/S	43	Part Misloaded	79	Tooth Action Error
8	Diameter U/S	44	Circularity	80	Lead Taper
9	Thread O/S	45	Chatter	81	Shallow/Step on Root
10	Thread U/S	46	Out of Round	82	Uneven Ground Teeth Flanks
11	Stripped Thread	47	MOB Gears O/S	83	Taper Diameter
12	Thread Missing	48	MOB Gears U/S	84	Total Composite Error
13	New Set Up Scrap	49	Misloaded by Robot	85	Machine Twice
14	Tool Change	50	Obsolete Parts	86	Push Out Failure
15	Out of Position	51	Power Failure	87	Damaged Hob Cutter
16	Cylindricity	52	Excessive Rust Preventive	88	Damaged Grinding Wheel
17	Flatness	54	Broach Tears	89	TE Error
18	Runout	54	Broach Shears	90	Burr in Root
19	Straightness	55	Damaged Teeth	91	Chip in Root
20	Parallelism	56	NCU Gear Teeth	92	Chamfering Failure
21	Perpendicularity	57	Bore Position	93	Drill Change
22	Concentricity	58	Hole Out of Position	94	Clamping Failure
23	Surface Finish	59	Material Distortion	95	Oxidization
24	Rust	60	Insufficient Milling	96	Failed Flow Test
25	Burrs	61	Chip Off Hardened Surface	97	Bar Code Failure
26	Non-Clean Up	62	Hob Twice	98	Change Over
27	False Cut	63	Misalignment	99	E-Stop
28	Insert Change	64	Misassembled	100	Thermal Damage
29	Tip/Root Diameter	65	Damaged by Robot	101	Lobe Lineal
30	Chipped Tooth	66	Fixture Mark (s)	102	Matrix/Laser
31	Tool Marks	67	Press Mark (s)	103	Telesis
32	Sensor/Tool Alarm	68	Leaker-Failed Leak Test	104	Angularity
33	Handling Damage	69	Coolant Failure	105	Circle Base Diameter
34	Crack(s)	70	Spline O/S	106	Scratch on Bore
35	Chip Build Up	71	Spline U/S	212	Insufficient Material
36	Incorrect Revision	72	Lead Error		

Appendix B

Algorithms

The algorithms that were implemented in the different pipelines found in Chapters 4, 5, 6, and 7 are available at:

`https://github.com/Allam95/Visual-Inspection-Algorithms`