# Design and Development of Semi-Automatic Tire Inspection Machine

Henadheera Arachchige Pasan Bhagya Perera

(168671B)

Thesis submitted in partial fulfilment of the requirements for the degree Master of Science in Industrial Automation

Department of Electrical Engineering

University of Moratuwa

Sri Lanka

May 2021

# DECLARATION

---

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the MSc thesis under my supervision.

Signature of the Supervisor: Date:

Prof. A G Buddhika P Jayasekara
Senior Lecturer
Department of Electrical Engineering
University of Moratuwa

## Abstract

Machine learning has become an important and interesting field when addressing complex industrial tasks. In this study, a semi-automatic tire inspection machine for tire retreading industry which uses machine learning techniques is developed.

Most consumers tend to retread their tires because retreading is an economical and an eco-friendly method. As a result, retreading industry is now becoming popular in developed countries as well as developing countries. Initial inspection is the most crucial activity in the retreading process because tire defect identification is performed in this phase. Failure in identification of defects prior to retreading, may cause delamination of a retreaded tire consequently leading to a disastrous accident.

There exist many advanced machines for initial tire inspection. Widely used method in the tire retreading industry is nondestructive defect detection based on X-Ray image processing. This method is very expensive and used by brand new tire manufacturing companies as well as tire retreading companies in developed countries. In addition to that, devices with holography and shearography techniques are used to identify defects which map the tire defects using optical means and are known to be extremely expensive.

Conventional inspection method is being followed by the Sri Lankan tire retreading industry as well as other developing countries such as India, Bangladesh etc. due to its cost effectiveness in replace of extremely expensive advanced machinery. The two main tasks performed in conventional inspection method are visual inspection and hammering test.

Operator carefully observe the worn tire, identify and mark the defects which could be observed through the naked eye under visual inspection. The identified defects can be classified as tire punctures, unwanted metal particles, ply damages, bead damages and side wall damages. Hammering test is carried out to identify the defects which are invisible to naked eye such as inner ply separations, ply damages of a tire, inner canvas damage and small air bubbles in tread area etc. Usually the test is performed by hammering all over the tire tread area using a brass rod and listening to the resulted noise difference by an expertise.

An expert human inspector performs both visual inspection and hammering test. This manual inspection is often associated with inaccurate results and undetected defects due to lack of expertise, causing visual fatigue which results in low efficiency and higher amount of labor costs in local tire retreading industry. Therefore, the main objective of this study is to eliminate the expert human resource from the initial tire inspection process and reduce the complexity of the activity.

In this research, visual inspection activity is trained to a model and Faster RCNN Inception v2 algorithm is used on TensorFlow platform. Image classification and tire defect detection are done with a collection of real-world industrial image data set. These images were captured using four cameras which were having a capacity of 12 mega pixels each. Basically 220 images were trained using a computer. Hammering test activity is trained to a model and YouTube-8M algorithm is used with VGGish

feature extractor on TensorFlow platform. The sound signal was captured via a normal USB microphone. The sound signal was analysed using Audacity open source software and fed as the input to train the model. Unwanted metal particles of the worn tires are detected using a metal detector. In addition to defect identification mechanisms, defect localisation system is developed using a microcontroller and an encoder. Defects which are identified from image processing or sound signal processing or metal detection, location of the defect is identified with respect to a reference point of tire. This is very useful for identifying the exact defect location of tire for the operator.

From the obtained results it can be concluded that, the above image classification and sound signal classification models provide results with a higher level of accuracy. Therefore, the expertise labor which is used to perform the initial inspection process could be replaced by a novice employee. Furthermore the unique and ideal structure of this developed machine is associated with low maintenance cost. As a result, small scale companies would be more comfortable with their existing financial situations when using this semi-automatic tire inspection machine to enhance their throughput of the tire retreading process.

***Keywords*- tire retreading, object detection, sound classification, image processing, deep learning, machine learning, TensorFlow**

# DEDICATION

This dissertation is dedicated to my wife and parents, to whom I can trace my every success to.

# ACKNOWLEDGMENTS

It is my great pleasure to offer my sincerest gratitude to my supervisor, Prof. A G Buddhika P Jayasekara, who has given me the opportunity to pursue my Master degree at the University of Moratuwa and for supporting me throughout my taught course and research with his knowledge and patience whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to his effort and encouragement and without him this research would not have been a success. One could not wish for a better supervisor.

I would also like to thank my progress review committee, Prof. Udayanga Hemapala, Prof. Chandima Pathirana and Prof. Ruwan Gopura for their insightful comments and encouragement, but also for the hard questions brought up which encouraged me to widen my research through various perspectives.

Special gratitude goes to Mr. Danasiri Perera (Proprietor at Pasan Retreaders, Horana, Sri Lanka) for allowing me to implement this machine in his company and supporting me with funding and ideas to align with the company's requirement. My special thanks also goes to the maintenance team of the company for their tremendous support provided through- out the project. I am also grateful to Mr.Youichi Nishizaki (Engineering Manager at Advanced Smart Mobility Inc., Institute of Industrial Science, The University of Tokyo, Japan) for supporting me with deep learning techniques and advising me on this study with new knowledge.

Finally, I thank my wife for her love, patience and understanding. I could not have made this achievement a reality without her confidence, continuous support and sacrifices. I also thank my parents for supporting me throughout my studies.

# TABLE OF CONTENTS

**Declaration** i

**Abstract** ii

**Dedication** iv

**Acknowledgments** v

**Table of Contents** viii

**List of Figures** xii

**List of Tables** xiii

**1 Introduction** 1

1.1 Background . . . . . 1

1.2 Literature Review . . . . . 2

1.2.1 Classification of Defects . . . . . 2

1.2.2 Existing Advanced Machines for Tire Inspection . . . . . 3

1.2.3 Conventional Method of Tire Inspection in Small Companies 6

**2 Design of Semi-Automatic Tire Inspection Machine** **8**

2.1 Objectives of the Design . . . . . 8

2.2 Structure of the Machine . . . . . 9

2.3 Metal Detection . . . . . 10

**3 Defect Detection Using Image Processing** **13**

3.1 Defect Detection and Classification with TensorFlow . . . . . 16

3.1.1 Main steps to create Tire Defect Detection (TDD) model . 17

3.1.2 mean Average Precision - mAP . . . . . 20

**4 Defect Detection Using Sound Analysis** **41**

4.1 Hammering test . . . . . 41

4.2 Sound classification with TensorFlow . . . . . 43

4.2.1 Choosing Tools and a Classification Model . . . . . 44

4.2.2 Training the Model . . . . . 45

4.2.3 Resources, Time, and Accuracy . . . . . 46

4.2.4 Choosing the Model . . . . . 46

4.2.5 Training with balanced dataset . . . . . 47

4.2.6 Training with unbalanced dataset . . . . . 48

**5 Localisation of Defects** **49**

5.1 Defect localisation mechanism . . . . . 49

5.2 Operation of the tire inspection machine . . . . . 52

**6 Results and Discussion** **55**

6.1 Defect identification and classification using image processing . . . 55

6.2 Sound signal classification model . . . . . 57

6.2.1 Tire inspection via sound classification model . . . . . 61

6.3 Tire inspection via metal detection . . . . . 67

6.4 Localisation of Defects . . . . . 68

**7 Conclusions** **70**

**A Appendix** **72**

A.1 Code to create pb2.py files . . . . . 72

# LIST OF FIGURES


1.1 Samples of tire defects (a) tire punctures, (b) unwanted metal particles, (c) ply damages, (d) bead damages and (e) side wall damages 2

1.2 Tire inspection using X-ray technology [26] . . . . . 4

1.3 Side view of advanced tire inspection machine [25] . . . . . 5

1.4 Sample X-ray images of tire defects (a) impurity in tire side wall, (b) bubble in tire side wall, (c) overlap in tire side wall, (d) impurity in tire tread and (e) overlap in tire tread [13] . . . . . 5

1.5 Visual inspection performing by an expert human labour [28] . . . 7

2.1 Structure of the semi-automatic tire inspection machine . . . . . 9

2.2 Metal Detector [27] . . . . . 11

2.3 Metal detector mounted on machine . . . . . 12

3.1 Block diagram of CNN architecture [35] . . . . . 13

3.2 Block diagram of Faster RCNN [35] . . . . . 14

3.3 Inception V2 module [35] . . . . . 14

3.4 Sample images of image dataset . . . . . 15

3.5 Basic procedure of developing the Tire Defect Detection model . . 15

3.6 Detail flow chart of the Tire Defect Detection model via image identification and classification . . . . . 15

3.7 TensorFlow CPU compatible other software versions on MacOS [1] 18

3.8 TensorFlow GPU compatible other software versions on MacOS [1] 18

3.9 Comparison of TensorFlow coco trained models with their speed [1] 20

3.10 Mathematical definitions of Precision, Recall and F1 [32] . . . . . 21

3.11 Flow chart for setting up the Anaconda Virtual Environment . . . 24

3.12 Flow chart for Compiling Protobufs and Running setup.py . . . . 25

3.13 Test image on Jupyter [2, 3] . . . . . 27

3.14 Sample images to train the model . . . . . 29

3.15 Pre defined label class file . . . . . 30

3.16 LabelImg Tool . . . . . 30

3.17 labels updating according to the class map . . . . . 31

3.18 generate_tfrecord.py code updating according to the class map [4] 32

3.19 Flow chart for setting up training . . . . . 34

3.20 View of the Tensorboard while training the model . . . . . 37

3.21 Tested image from the trained model . . . . . 39

3.22 Tested image from the trained model . . . . . 40

3.23 Tested image from the trained model . . . . . 40

4.1 Listening the noise while hammering to worn tire . . . . . 42

4.2 Brass rod which is taken to hammer the tire . . . . . 42

4.3 Basic mechanism of the sound test [9] . . . . . 43

4.4 Sound signal behaviour of defective spot(DS) vs healthy spot(HS) 44

5.1 Encoder mounted to motor rotor shaft . . . . . 50

5.2 Marked reference point . . . . . 50

5.3 Direction of the tire rotation . . . . . 51

5.4 Flow chart of defect localisation for each method . . . . . 52

5.5 Report generated in SD memory card . . . . . 53

6.1 Same point in three consecutive frames] . . . . . 56

6.2 Results from the trained model - Outside_damage [Precision 94%] 56

6.3 Results from the trained model - Outside_damage [Precision 96%] 57

6.4 Sample audio signals of healthy spots . . . . . 59

6.5 Sample audio signals of defective spots . . . . . 60

6.6 Healthy spot- Amplitude vs Time. [wave duration = 79 ms] . . . 61

6.7 Defective spot- Amplitude vs Time. [wave duration = 159 ms] . . 62

6.8 Healthy spot- Sound level(dB) vs Frequency . . . . . 62

6.9 Defective spot- Sound level(dB) vs Frequency . . . . . 63

6.10 Summary of the sound signal behaviour of Healthy spot and Defective spot [9, 16] . . . . . 63

6.11 Healthy spot vs Defective spot . . . . . 64

6.12 Sound signal of healthy spot . . . . . 65

6.13 Result of the sound signal at healthy spot - [Healthy spot: 1.00] . 65

6.14 Sound signal of defective spot . . . . . 66

6.15 Result of the sound signal at defective spot - [Defective spot: 1.00] 66

A.1 Command line code to create pb2.py files . . . . . 72

# LIST OF TABLES

3.1 Performance comparison of object detection algorithms [31] . . . . 36

6.1 Trial data of image processing model . . . . . 58

6.2 Trial data of sound signal analysing model . . . . . 67

6.3 Trial data of metal detection . . . . . 68

6.4 Sample data of defect localisation system . . . . . 69

Chapter 1

# INTRODUCTION

## 1.1 Background

Tire retreading is the procedure of replacing the outer tread area of a tire which is worn out with a new layer of rubber. It is expected that 80% of the material cost of a tire could be saved by following the retreading process. Currently available technologies have enabled to produce high quality tires which has even induced a competition to brand new tires. Usually, retreaded tires are sold at a price which is lower from 30 to 50% of the price of new tires. Therefore, in the present world most consumers tend to retread their tires due to the high cost of new tires and also as an eco-friendly method of recycling [33].

When retreading a worn tire, it is very much important to identify the defects of it during the initial inspection. Because, if a particular defect continues to exist during this retreading process, then it may consequently lead to an accident once it is being installed on a vehicle. More than half of the failures in retreaded tires are as a result of poor initial inspection [5]. Therefore, initial inspection could be highlighted as the most important phase of the retreading process.

An expert human inspector conducts the traditional initial inspection of a tire and mainly tire defects such as unwanted metal particles, bead damages, ply damages, side wall damages and tire punctures etc., are identified. This manual inspection is performed visually and often causes inaccurate results and undetected defects due to lack of expertise, causing visual fatigue which result in

low efficiency and high labor costs.

Most of the developed countries use X-Ray technologies to detect defects via nondestructive defect detection techniques to maintain the quality of their products [13]. However, defect detection machines which use X-Ray technology are very expensive. Hence, being a developing country, Sri Lanka faces difficulties of using this technology in local tire retreading factories owing to its high cost and long payback times.

## 1.2 Literature Review

This section contains a detailed review on tire defects and existing advanced machinery in industry which are used for initial inspection. Towards the end of the chapter, conventional methods being used by small scale companies for initial inspection are reviewed.

### 1.2.1 Classification of Defects

In tire retreading process the initial step is to inspect the worn tire. During this inspection we normally identify and classify the defects such as (a) tire punctures, (b) unwanted metal particles, (c) ply damages, (d) bead damages and (e) side wall damages. Images for each of the mentioned defects are shown in Figure 1.1


Figure 1.1: Samples of tire defects (a) tire punctures, (b) unwanted metal particles, (c) ply damages, (d) bead damages and (e) side wall damages

### 1.2.2 Existing Advanced Machines for Tire Inspection

Retreaded tires are being used for critical applications such as commercial and military aircrafts and also for school buses. Therefore, failure of a retreaded tire due to a delamination may result in a disastrous situation. Many attempts have been made in developing inspection devices which has the capability to detect defects in a tire. For example, devices with holography techniques map the tire defects using optical means and are known to be extremely expensive, highly sensitive for vibration while being a relatively slow method. Shearography is another similar technique which is less sensitive to vibration but very expensive and relatively slow. Defect detection machines which uses high voltage tests are less expensive but are capable of detecting only metal inclusions in the tire tread. Ultra Sound techniques has the ability to detect only air-filled defects and known to be very sensitive to the thickness of the tire tread and therefore are not used to detect defects in aircraft tires. The air injection test is specified by military for tire defect identification but is highly time consuming and known to be less accurate. [34]

Nowadays with the latest technology, industrial products are subjected to automatic quality inspection. This field has become a demanding subject area in image processing as well as computer vision communities with machine learning and deep learning techniques. Currently we can find various methods which are based on different theories, namely texture-based detection method and non-destructive defect detection based on X-Ray image processing. Some existing advanced machines are as follows in the Figure 1.2 and 1.3

Texture feature extraction is used in texture-based detection method. Latif-Amet et al. used the sub band co-occurrence matrices (CM) to characterise texture feature of multi-scale sub-bands of inspected images [13]. Major disadvantage of this method is, having a high computational complexity for larger size images.

Nondestructive defect detection which is based on X-Ray image processing method is the widely used method in the tire retreading industry [10]. This method is also being used by brand new tire manufacturing companies as well. In this method we can identify tire defects clearly as shown in the below Figure 1.4. The main advantage is, we can see the defects which cannot be visualised from the naked eye. Due to the imperfect raw materials which are used in the manufacturing process, tire side wall and bead may contain small bubbles, impurities or overlaps which are not visible to our eyes [14]. Therefore, X-Ray image processing method is a well-suited candidate for tire inspection [11, 13, 21]. However, the main disadvantage of this method is its cost due to the expensiveness of X-Ray technology [8]. Therefore, this method is mainly used by developed countries and large-scale companies. Although this method is famous among tire manufacturing and retreading industry, most of the time developing countries are not capable of using this method due to its high cost. Therefore, majority of the companies use only the visual inspection and hammering test method to check ply damages which are considered as long-established methods.

Figure 1.2: Tire inspection using X-ray technology [26]

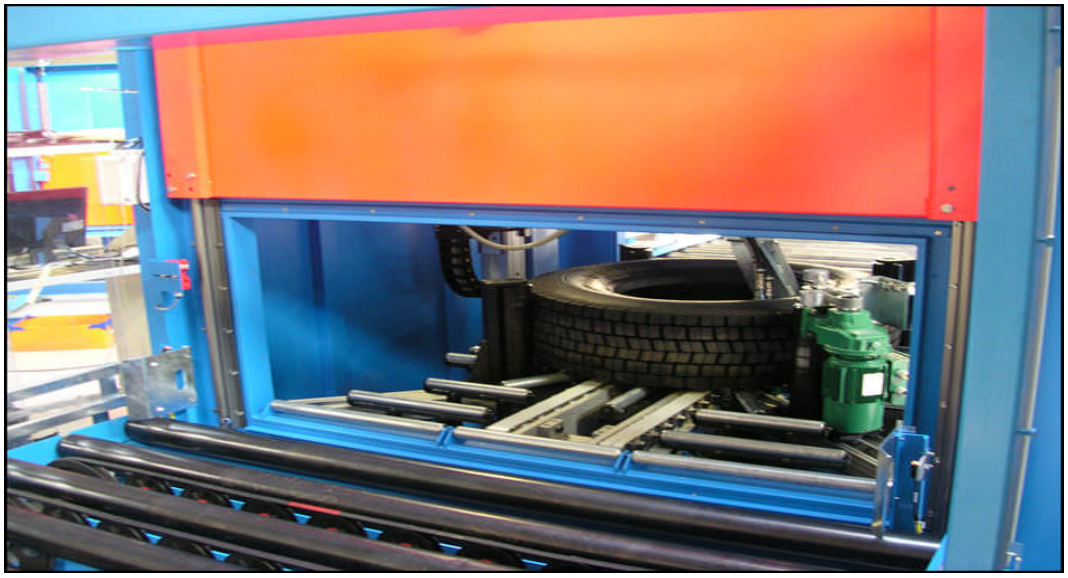


Figure 1.3: Side view of advanced tire inspection machine [25]

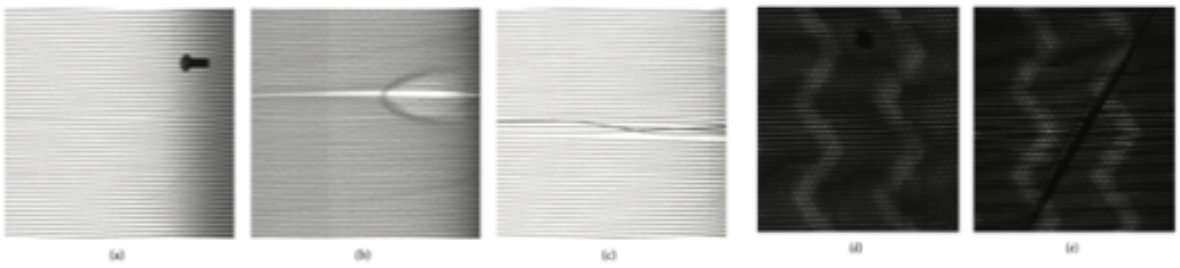


Figure 1.4: Sample X-ray images of tire defects (a) impurity in tire side wall, (b) bubble in tire side wall, (c) overlap in tire side wall, (d) impurity in tire tread and (e) overlap in tire tread [13]

### 1.2.3 Conventional Method of Tire Inspection in Small Companies

Visual inspection and hammering test together is performed in small tire retreading companies to identify tire defects and are known to be the conventional method of tire inspection. These two inspection techniques are cost effective and relatively slow while requires expert human labour to be performed.

Visual inspection is carried out over the outside of the tire as well as inner side of the tire. During this process, operator observe, identify and mark the defects which were explained in the above classification of defects section [5]. Below Figure 1.5 shows how an operator performs the visual inspection.

During the hammering test, brass rod is used to hammer all over the tread area of the tire. In this case, operator's experience matters more than the visual inspection. While hammering the tire, operator listens to the resulted sound of the tire to identify ply damages or weak areas which are not visible to naked eye. If these ply damages are not identified in the initial inspection phase, small air bubbles may develop with time during the process of retreading or when the vehicle is running on the road. These ply damages may lead to catastrophic accidents when vehicle is running at a high speed. After identifying all areas of defects via conventional method of tire inspection (visual inspection and hammering test), tires will be repaired in the repairing section prior to tire buffing activity while worn tires without defects would be taken directly into the buffing activity.

Figure 1.5: Visual inspection performing by an expert human labour [28]

Chapter 2

# DESIGN OF SEMI-AUTOMATIC TIRE INSPECTION MACHINE

---

In this research, I have introduced a novel method to identify defective tires accurately and classify the defects to reduce the complexity of identification of tire defects. The main objectives of the design are as follows.

## 2.1 Objectives of the Design

Main objective of this research is to eliminate the requirement of expertise knowledge of human labor from the initial inspection process in the tire retreading industry. In order to achieve this, below objectives are addressed.

1. Developing a model to detect surface defects on a tire. The developed defect detection techniques have focused on image recognition and I have used faster RCNN (Regional Convolution Neural Network) inception v2 model to improve the image classification speed on TensorFlow with deep learning techniques.

2. Developing a method to identify internal ply damages which are not visible to naked eyes. Audacity software is used for noise recognition and to analyse the sound signal obtained as the outcome of hammering test performed on the tire.

3. Identifying the unwanted metal particles using a metal detector.

4. Localisation of defects which were identified via image processing, sound

signal classification and metal detection.

5. Designing and implementing a suitable structure for tire inspection machine which can hold all the above features in a simple and an ideal manner for local tire retreading industry.

The model developed in this study is more economical and affordable for tire retreading industry in developing countries such as Sri Lanka.

## 2.2 Structure of the Machine

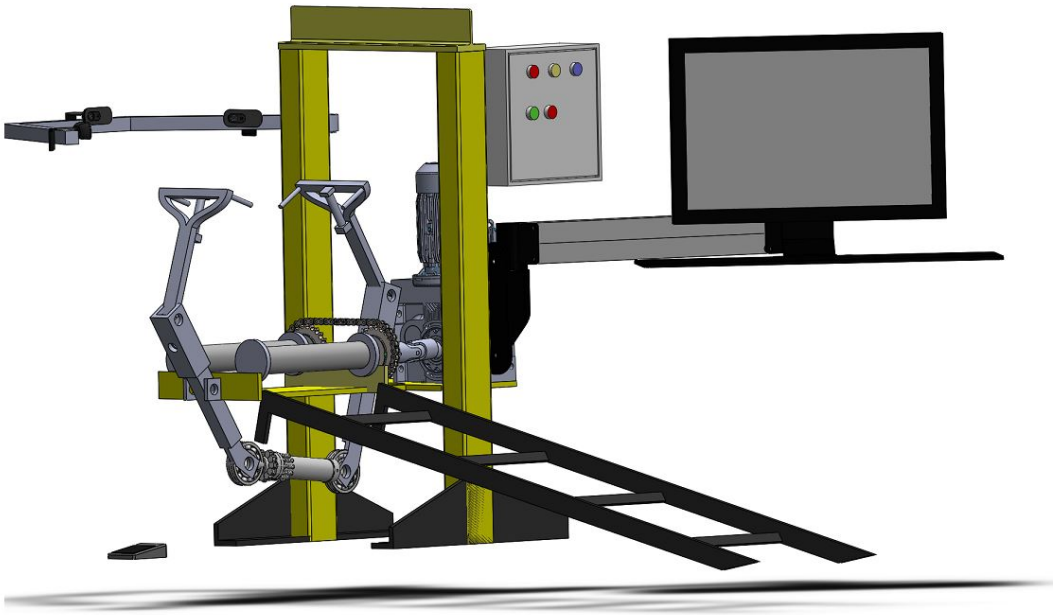


Figure 2.1: Structure of the semi-automatic tire inspection machine

This research is carried out by obtaining image data from worn tires and audio data when hammering on worn tires while doing the ordinary initial inspection process of tire retreading in a local company, Sri Lanka. I have designed a special machine structure with four cameras as shown in the above Figure 2.1. This structure is having the ability to grab the useful data from the cameras and microphones when the tire is rotating. Not only that, it is very much convenient

to proceed with the hammering test using this machine, when compared to the conventional method. In addition to these activities, it is very easy to observe the defect detection through the monitor which is very much conveniently visible to the operator.

Initially, I have taken a sample image dataset of 220 images from a 12-megapixel camera and sound signals were taken from built in microphone in the computer. Sound signals were analysed by using an open source software called Audacity Soft

Visual inspection can be done up to a certain level while developing a defect classification model via image object detection using machine learning techniques and tools such as TensorFlow.

Hammering test can be done by analysing the sound signal of the hammering noise. To perform the test, we use a brass rod and the hammering is done by a human labor same as in the conventional method of tire inspection. As I mentioned previously, the analysing of the sound signal is carried out from the Audacity Software and signal is fed to a sound classification model which is developed by TensorFlow. Therefore, we can clearly identify the difference between a sound signal of a healthy spot and a defective spot in a tire.

In addition to these two methods I have introduced a metal detector which is fine tuned for tires to identify the unwanted additional metal particles which usually comes with the worn tires. This metal detector consists of an induction proximity sensor. These sensors are fine tuned up to 6cm distance to sense.

### 2.3 Metal Detection

Usually worn tires may consist with unwanted metal particles such as nails, pins and etc. Therefore we must remove unwanted additional metal particles

during the initial inspection process. The importance of removing the metal particles is preventing probable damages which might occur for machines in next stages of the retreading process. I have used an inductive proximity sensor to identify these particles. Basically 90 percent of tires used for retreading are canvas tires which are having canvas inside the ply area while the other amount is steel wired tires. Higher amounts of canvas tire usage in tire retreading industry is due to their reusability. With the suggested metal detection method in this study, we can only proceed with canvas tires. The inductive proximity sensor is as below in Figure 2.2 and the way of mounting to main structure is shown in Figure 2.3.


Figure 2.2: Metal Detector [27]

Figure 2.3: Metal detector mounted on machine

Chapter 3

# DEFECT DETECTION USING IMAGE PROCESSING

Developing a model for image identification and classification of defective tires is not convenient with different tire tread patterns. But here as I mentioned previously, as the first step of collective image dataset, I have taken a small dataset of 220 images to initialise the model. Defect detection is developed using Faster RCNN (Region Convolutional Neural Network) inception V2 model in COCO models which is developed based on TensorFlow [18].

The modified version of Fast RCNN is called Faster RCNN. The main difference between the two is that, selective search for generating Regions of Interest is used in Fast RCNN while "Region Proposal Network" (RPN) is used by Faster RCNN. Figure 3.1 illustrates the architecture of CNN. It consists of an input layer, convolution layer, pooling layer, fully connected layer, and an output layer.

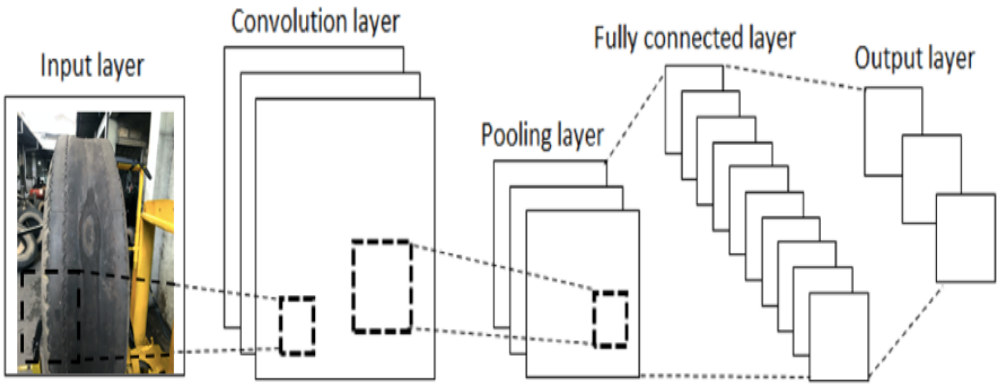



Figure 3.1: Block diagram of CNN architecture [35]

The reason behind selecting the Faster RCNN algorithm can be elaborated by comparing the similar algorithms with their limitations. Below table 3.1 shows the difference of Faster RCNN algorithm compared to other algorithms. Faster RCNN model diagram is shown in below Figure 3.2. Some sample data images are shown in Figure 3.4.

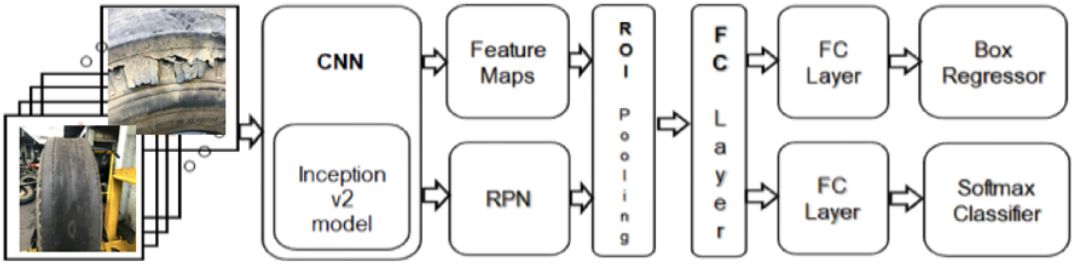



Figure 3.2: Block diagram of Faster RCNN [35]

The features of the proposed region in an image is extracted using the convolutional neural network loaded with the Inception V2 filter banks and the architecture of the inception V2 model is as in the Figure 3.3

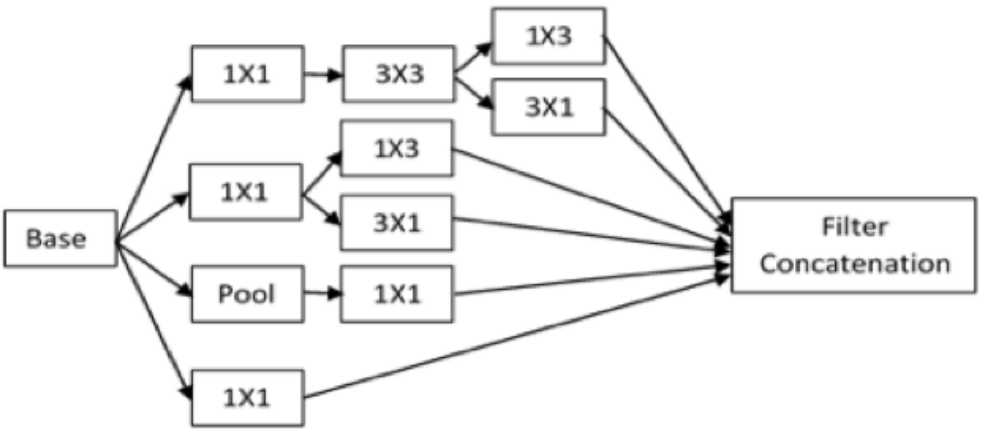



Figure 3.3: Inception V2 module [35]

When making the model for visual inspection I have considered all the defects which are in the classification of defects section. Steps and the basic procedure of developing an image object detection model which I followed to make the model are shown in Figure 3.5 and Figure 3.6

Figure 3.4: Sample images of image dataset

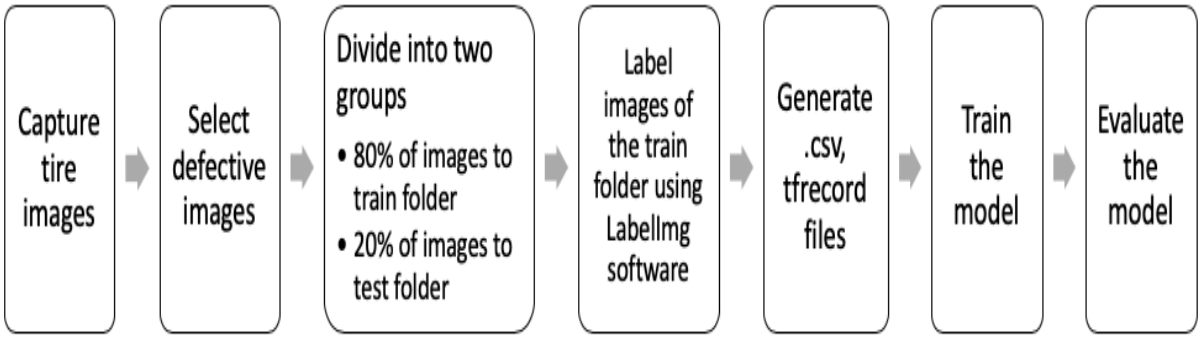



Figure 3.5: Basic procedure of developing the Tire Defect Detection model

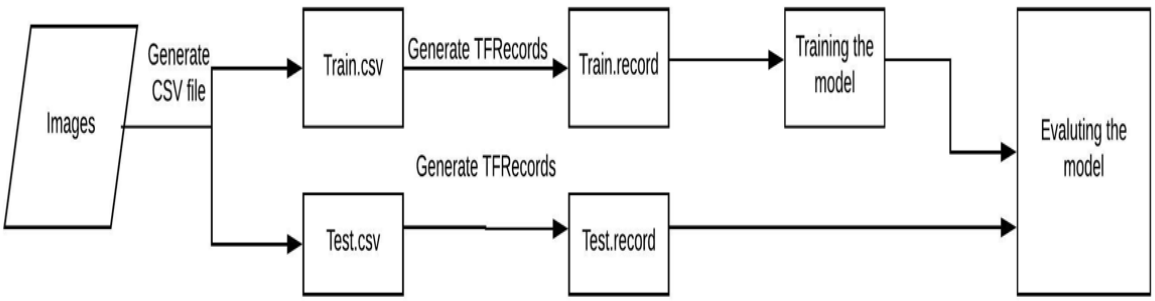



Figure 3.6: Detail flow chart of the Tire Defect Detection model via image identification and classification

The issue of selective search is solved by Faster R-CNN through restoring with Region Proposal Network (RPN). Initially, feature maps are extracted from the image input utilizing ConvNet. Next, these maps are passed across a RPN from which object proposals are returned. Received maps are classified in the next step and finally prediction of bounding boxes are done.

The steps followed in a Faster R-CNN algorithm when detecting objects in an image can be summarised as below.

1. Get an input image first and then pass it on to the ConvNet where feature maps would be generated for the image
2. Try out Region Proposal Network (RPN) on the above feature maps and obtain object proposals
3. ROI pooling layer is applied to weigh down the size of all the proposals to one similar size
4. At last, all the proposals are passed to a layer which is fully connected to classify and bounding boxes are predicted for the image

### 3.1 Defect Detection and Classification with TensorFlow

In this research, TensorFlow (CPU) platform is used in MacOS. For further developments this CPU version can be upgraded to GPU version for better performance. In that case we can train our model eight times faster than the CPU version. The basic steps which I followed to develop the model can be stated as follows. [3]

1. Installing Anaconda
2. Creating the defect detection model directory and anaconda virtual environment

3. Collecting and labelling images

4. Training data creation

5. Creating the label map and setting up the training

6. Training of the model

7. Exporting and analysing the inference graph

8. Testing the model

### 3.1.1 Main steps to create Tire Defect Detection (TDD) model

**1. Installing Anaconda**

Terminal in MacOS is used with bash shell to install Anaconda and the necessary software. For Anaconda installation, we can use "pip install anaconda" command in Terminal prompt.

Since I am using TensorFlow CPU version, CUDA and CUDNN is not a requirement to install. But If we are using TensorFlow GPU, then it is essential to install versions; CUDA and CUDNN which are compatible with TensorFlow version. I have shown below which version of CUDA, CUDNN and Python are required for TensorFlow CPU and GPU versions in Figure 3.7 and Figure 3.8. The red colour box of Figure 3.7 shows the version which I have used in this research to make the model.

**2. Creating the defect detection model directory and anaconda virtual environment**

It is required to use the particular directory structure which is supplied in GitHub repository of it for the TensorFlow Object Detection Application Programming Interface (API). Other supplementary Python packages and specific

CPU

| Version | Python version | Compiler | Build tools |
|---|---|---|---|
| tensorflow-2.2.0 | 3.5-3.8 | Clang from xcode 10.1 | Bazel 2.0.0 |
| tensorflow-2.1.0 | 2.7, 3.5-3.7 | Clang from xcode 10.1 | Bazel 0.27.1 |
| tensorflow-2.0.0 | 2.7, 3.5-3.7 | Clang from xcode 10.1 | Bazel 0.27.1 |
| tensorflow-2.0.0 | 2.7, 3.3-3.7 | Clang from xcode 10.1 | Bazel 0.26.1 |
| tensorflow-1.15.0 | 2.7, 3.3-3.7 | Clang from xcode 10.1 | Bazel 0.26.1 |
| tensorflow-1.14.0 | 2.7, 3.3-3.7 | Clang from xcode | Bazel 0.24.1 |
| tensorflow-1.13.1 | 2.7, 3.3-3.7 | Clang from xcode | Bazel 0.19.2 |
| tensorflow-1.12.0 | 2.7, 3.3-3.6 | Clang from xcode | Bazel 0.15.0 |
| tensorflow-1.11.0 | 2.7, 3.3-3.6 | Clang from xcode | Bazel 0.15.0 |
| tensorflow-1.10.0 | 2.7, 3.3-3.6 | Clang from xcode | Bazel 0.15.0 |

Figure 3.7: TensorFlow CPU compatible other software versions on MacOS [1]

GPU

| Version | Python version | Compiler | Build tools | cuDNN | CUDA |
|---|---|---|---|---|---|
| tensorflow_gpu-1.1.0 | 2.7, 3.3-3.6 | Clang from xcode | Bazel 0.4.2 | 5.1 | 8 |
| tensorflow_gpu-1.0.0 | 2.7, 3.3-3.6 | Clang from xcode | Bazel 0.4.2 | 5.1 | 8 |

Figure 3.8: TensorFlow GPU compatible other software versions on MacOS [1]

additions to the variables; PATH and PYTHONPATH are also required with some additional setup commands for everything to configure properly and to train or run the object detection model.

### 2a. Downloading TensorFlow Object Detection API repository from GitHub

The following link gives the TensorFlow Object Detection API repository from GitHub which I used to develop the defect detection model.

https://github.com/tensorflow/models

A folder is created directly in /Desktop/Github/ and it was named as "tensorflow1". The full framework of TensorFlow object detection exists in the working directory, while my training data, training images, configuration files, trained classifier, and everything else that are required for the object detection classifier.

After downloading the zip file, "models-master" folder is extracted directly into the directory of /Desktop/Github/tensorflow1 which was created previously. Then renamed the "models-master" as "models".

This research utilized TensorFlow (CPU) v1.15 on MacOS and the GitHub commit of it in TensorFlow Object Detection API.

### 2b. Downloading Faster-RCNN-Inception-V2-COCO model from TensorFlow's model zoo

Many models used for object detection including pre-trained classifiers with particular architectures of neural network are provided by TensorFlow's model zoo. Models like SSD-MobileNet contains an architecture that permits faster detection but results in lower accuracy where as models like Faster-RCNN is capable of providing slower detection associated with a larger accuracy. [2, 3]

Faster-RCNN-Inception-V2 model was chosen for my detector to maintain higher accuracy throughout the operation. Even-though this model is slow compared to other models, it is very much important to maintain the higher level of accuracy to catch all the defects in the worn tire. Below Figure 3.9 shows some models with their speed.

| Model name | Speed (ms) | COCO mAP[^1] | Outputs |
|---|---|---|---|
| ssd_mobilenet_v1_coco | 30 | 21 | Boxes |
| ssd_mobilenet_v1_0.75_depth_coco ☆ | 26 | 18 | Boxes |
| ssd_mobilenet_v1_quantized_coco ☆ | 29 | 18 | Boxes |
| ssd_mobilenet_v1_0.75_depth_quantized_coco ☆ | 29 | 16 | Boxes |
| ssd_mobilenet_v1_ppn_coco ☆ | 26 | 20 | Boxes |
| ssd_mobilenet_v1_fpn_coco ☆ | 56 | 32 | Boxes |
| ssd_resnet_50_fpn_coco ☆ | 76 | 35 | Boxes |
| ssd_mobilenet_v2_coco | 31 | 22 | Boxes |
| ssd_mobilenet_v2_quantized_coco | 29 | 22 | Boxes |
| ssdlite_mobilenet_v2_coco | 27 | 22 | Boxes |
| ssd_inception_v2_coco | 42 | 24 | Boxes |
| faster_rcnn_inception_v2_coco | 58 | 28 | Boxes |
| faster_rcnn_resnet50_coco | 89 | 30 | Boxes |
| faster_rcnn_resnet50_lowproposals_coco | 64 | | Boxes |

Figure 3.9: Comparison of TensorFlow coco trained models with their speed [1]

### 3.1.2 mean Average Precision - mAP

mean Average Precision (mAP) for Object Detection is compared in the above table with other trained models. Average precision (AP) is a famous metric used to measure the accuracy of object detectors such as Faster R-CNN, SSD etc. Average precision calculates the mean of precision value for recall value over 0 to 1 [32].

### Precision and Recall

Precision also known as positive predictive value and measures how much accurate the predictions are, that is the fraction of appropriate instances among the retrieved total instances.

Recall also called sensitivity measures how good you find all the positives, that is the fraction of appropriate instances which were retrieved. As an example, 80 percent of probable positive cases can be found in top K predictions.

Below Figure 3.10 shows their mathematical definitions:

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

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

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$TP$ = True positive

$TN$ = True negative

$FP$ = False positive

$FN$ = False negative

Figure 3.10: Mathematical definitions of Precision, Recall and F1 [32]

Proportion of TP is Precision ( Eg. Precision = 2/3 = 0.67 )

Proportion of TP out of the possible positives is Recall ( Eg. Recall = 2/5 = 0.4 )

By comparing the above stats, we can select which model is the most suitable for object detection classifier to be trained on. SDD-MobitelNet model is more suitable to be used in a situation where the object detector is used on a device which has considerably smaller computational power like smart phone or Raspberry Pi. In a situation where the detector is running on a decently powered desktop PC or laptop, RCNN models are preferable. As I am using a computer,

I have chose Faster RCNN Inception V2 COCO model.

Below included link can be used to download the model.

http://download.tensorflow.org/models/object_detection/

faster_rcnn_inception_v2_coco_2018_01_28.tar.gz

The downloaded folder is moved into below directory in my Github repository

~/Desktop/Github/tensorflow1/models/research/object_detection

This repository which was downloaded contains all the images, .csv files, annotation data, and TFRecords which are required when training "Pinochle Deck" playing card detector. But in my research I completely converted it to Tire Defect Detector by doing following changes to the Github repository. Python scripts are also held by it which are utilized when generating training data. Repository contains scripts that are used to test out object detection classifier on videos, webcam feed or images.

Following changes have been made to create the Tire Defect Detector model.

Deleted the following files without deleting the folder,

1. All files in \ object_detection \ images\ train and\ object_detection\ images\ test.
2. The "test_labels.csv" and " train_labels.csv" files in \object_detection\images.
3. All files in\ object_detection\ training.
4. All files in \object_detection\ inference_graph.

### 2c. Setting up new Anaconda virtual environment

New virtual environment named "tensorflow1" is created in the command terminal that pops up by providing the command as shown below flow chart in Figure 3.11

TensorFlow does not require packages such as 'pandas' and 'opencv-python', but they are employed in order to create TFRecords in Python scripts and also to work with videos, images and webcam feeds [6, 17, 19].

### 2d. Configuring PYTHONPATH environment variable

Next, create a PYTHONPATH variable which points to the /models, /models/research, and /models/research/slim directories. The following command can be made the PYTHONPATH to these three locations.

(tensorflow1) ~/Github$ export PYTHONPATH=(models folder path):(research folder path):(slim folder path)

Path can be easily added by drag and drop the folder to terminal.

(tensorflow1) export PATH=$PATH:PYTHONPATH

Each time when exiting the virtual environment of "tensorflow1", the PYTHONPATH variable gets resent and it is required to set it up again. "echo %PYTHONPATH% can be used to check whether it has or not been reset.

### 2e. Compiling Protobufs and running setup.py

These Protobuf files are used for model configuration and parameter training by TensorFlow. The following steps are done as shown below flow chart in Figure 3.12

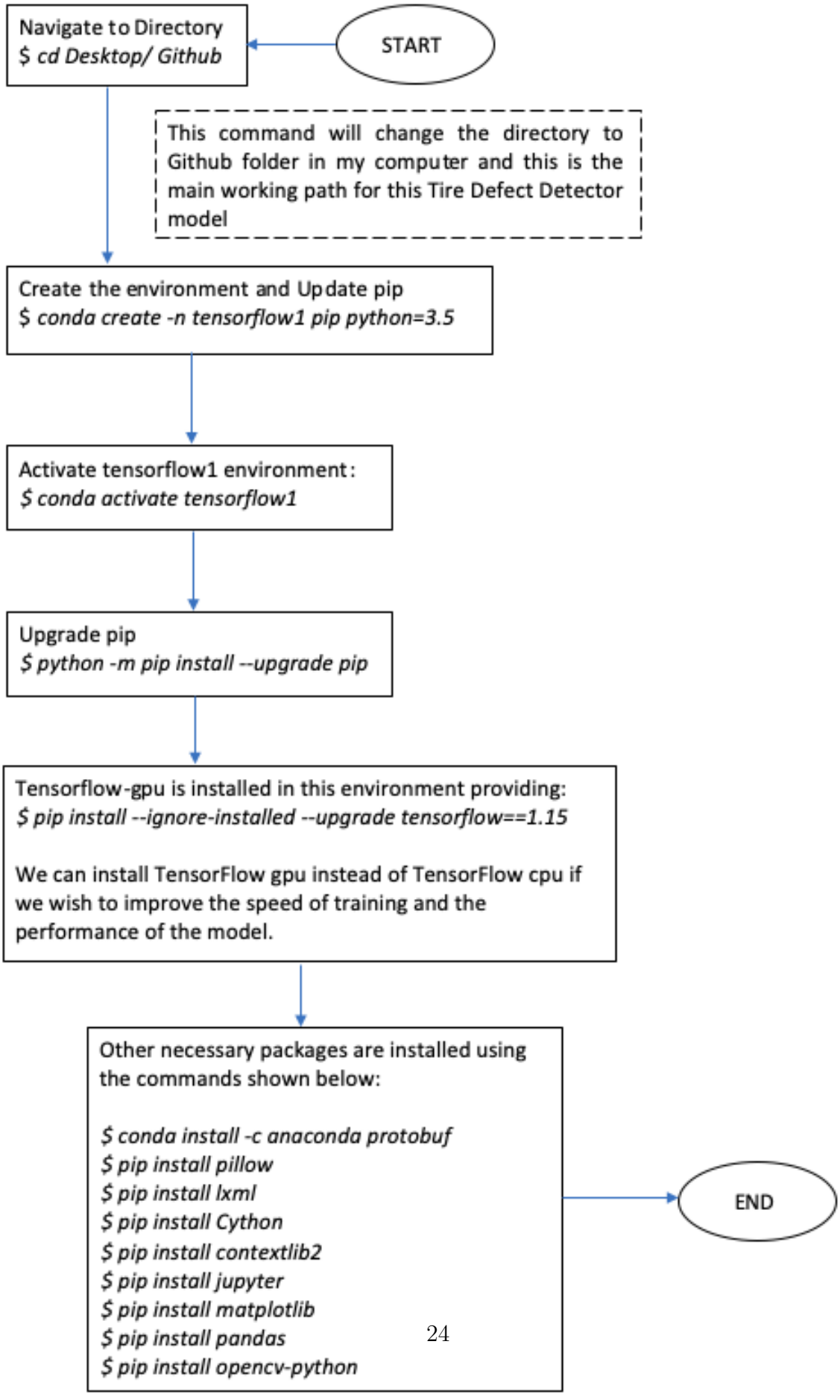





Figure 3.11: Flow chart for setting up the Anaconda Virtual Environment

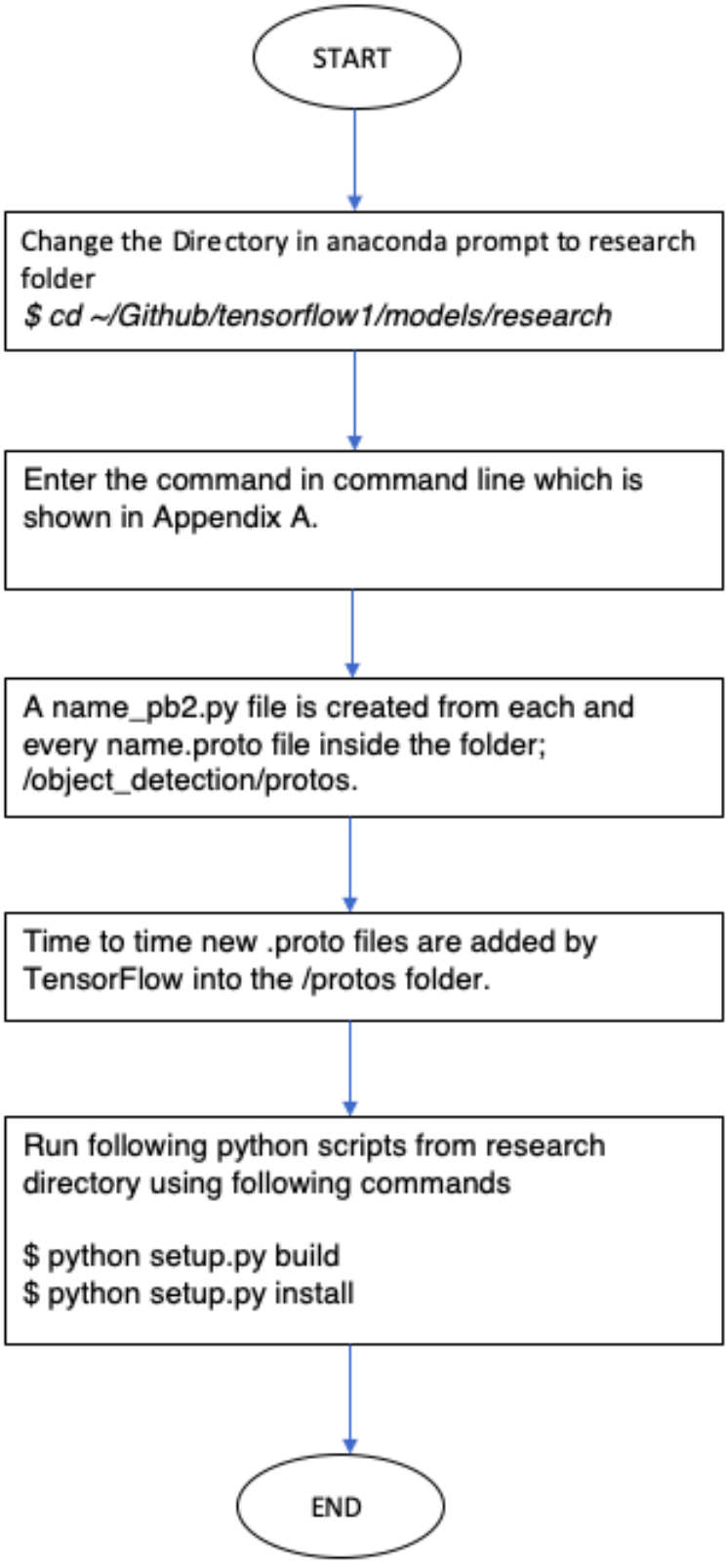



Figure 3.12: Flow chart for Compiling Protobufs and Running setup.py

### 2f. Testing TensorFlow setup to confirm whether it works

At this stage, the TensorFlow Object Detection Application Programming Interface (API) is prepared to employ pre-trained models or train new models for object detection. First, it is needed to test it out to confirm whether the installation is properly working. In order to achieve this, launch the object_detection_tutorial.ipynb script with Jupyter. From the /object_detection directory,

(tensorflow1) ~/Github$ cd tensorflow1/models/research/object_detection

The above command changes the working directory into object_detection

(tensorflow1) ~/object_detection$ jupyter notebook object_detection_tutorial.ipynb

The script can be opened in the default web browser using the above code. This allows to step one section at a time through the code. By clicking on the "Run" button which exists in the upper toolbar, it is possible to step through every section. When the "In [ * ]" text populates with a number next to the section (e.g. "In [1]", we can come to a conclusion as the section has completed running.

Once stepping throughout the script is completed, we should be able to see two images which are labeled at the bottom part of the page. In case if we see the below Figure 3.13, then we can come to a conclusion as everything is functioning well. The bottom section will appear with any errors which have been encountered if it is not functioning properly.

### 3. Collecting and labelling images

As the TensorFlow Object Detection API is ready to be used, images should then be delivered which are employed to train a new detection classifier.

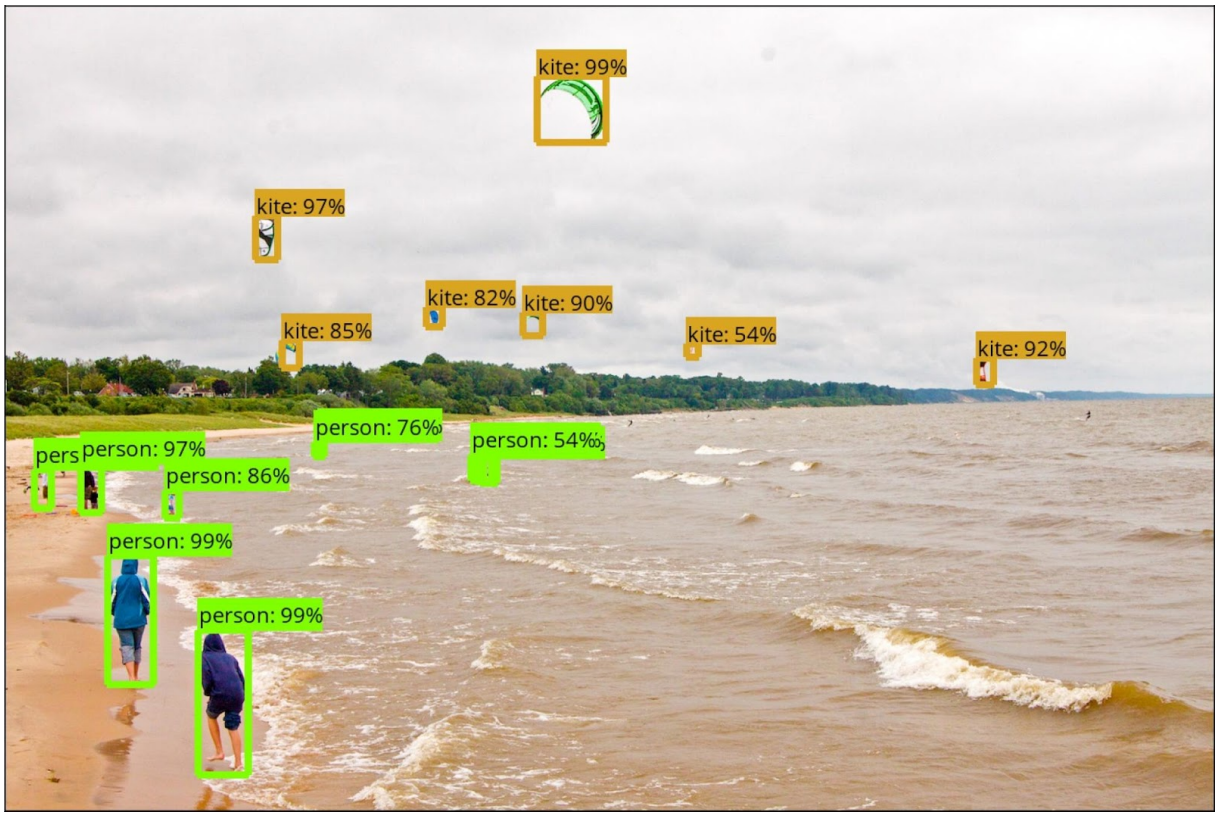



Figure 3.13: Test image on Jupyter [2, 3]

3a. Gathering Images

Hundreds of images of a particular object are required by TensorFlow to train a good detection classifier. It is necessary for the training images to occupy random objects in company with desired objects while having different types of backgrounds along with various lighting conditions to train a robust classifier. It is necessary to have several images in which the desired object is overlapped with some other objects, partially obscured or appear only halfway in the image.

In my classifier, six different defect types, namely; Bead damages, Previous Repair, Ply damages, Outside damage, Inside puncture and impurities to be detected . To capture these images, I have used a 12 mega pixel webcam to capture about 40 images from each defect type, with many different non-desired objects appearing in the images. I recommend to have minimum 200 pictures overall. In this model, I have employed 220 images to train Tire Defect Detector

(TDD) and the sample images used to train the model are shown in Figure 3.14

Images should not be too large. Furthermore, they should be usually smaller than 200KB, and the resolution of images must not exceed 720x1280. Larger the image sizes are, longer it would consume for the classifier to be trained. I have reduced the image size when using larger images.

After getting all the images which are needed, 20% should be moved to repository's /object_detection/ images/ test directory, and the left 80% to the / object_detection / images/ train directory. It is necessary to confirm that there exist a diversity of images in two of the directories; /test and /train.

3b. Labelling Images

This is process of labelling the desired objects in each image.LabelImg open source software can be found in Github and is a prominent and a convenient tool for labelling images.

LabelImg Github link is as below.

https://github.com/tzutalin/labelImg

LabelImg software can be run by using the following command. Defects types should be saved in this pre defined class file as shown in the Figure 3.15

python3 labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE PATH]

Then labelling procedure must be done for both of the /images/train and /images/test directories.

LabelImg creates a .xml file that holds label data of every image. Using these .xml files, TFRecords are generated which acts as an input to the training model. After each image is labelled and saved, there will be a .xml file for every image inside the relevant image directory.

Figure 3.14: Sample images to train the model

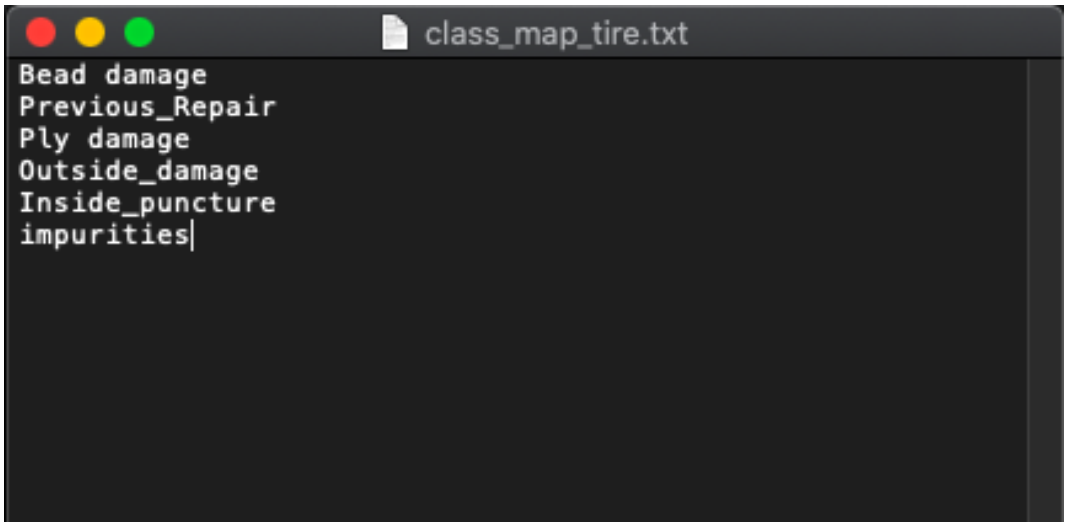



Figure 3.15: Pre defined label class file

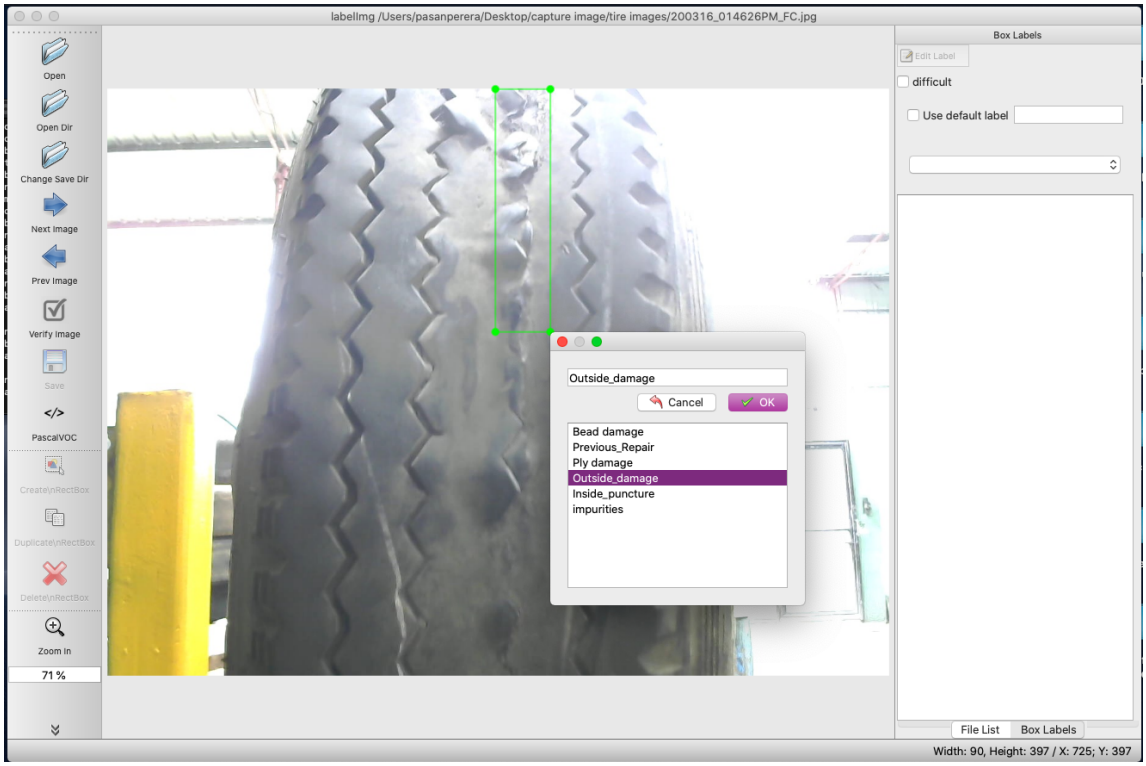



Figure 3.16: LabelImg Tool

## 4. Training data creation

With the labelled images, we can now produce TFRecords which act as one of the input data for the training model. In order to achieve this, convert .xml files into .csv format and generate .tfrecord file by using generate_tfrecord.py

The image .xml data is first used in creating .csv files which include all the necessary data for the training and testing of images. The below shown command from the /object_detection folder is used in the Anaconda command prompt

(tensorflow1) ~/object_detection$ python xml_to_csv.py

Two separate .csv files are produced to train labels and test labels as train_labels.csv and test_labels.csv file in the /object_detection/images folder.

After that, I have updated the generate_tfrecord.py file in a text editor as shown in the figure 3.12. According to the label map, every object is assigned a name and an ID number. The very same method of number allocation is used when setting up the labelmap.pbtxt as shown in Figure 3.17

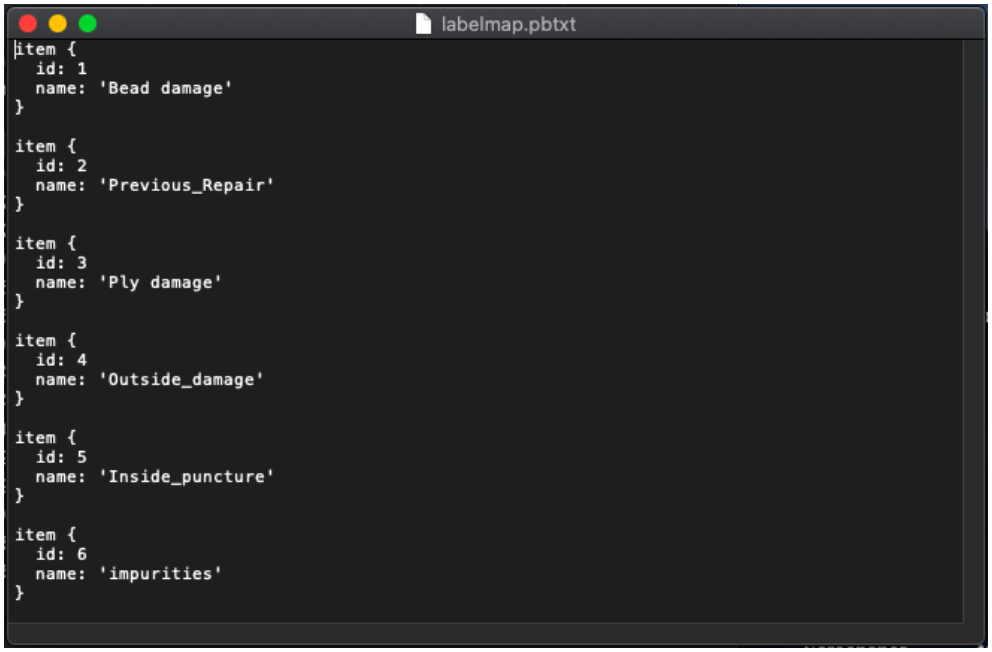



Figure 3.17: labels updating according to the class map

I have created the below shown code in generate_tfrecord.py and these label classes must be the same as label class map. The updated code is shown in Figure 3.18

```
29
30 # TO-DO replace this with label map
31 def class_text_to_int(row_label):
32     if row_label == 'Bead damage':
33         return 1
34     elif row_label == 'Previous_Repair':
35         return 2
36     elif row_label == 'Ply damage':
37         return 3
38     elif row_label == 'Outside_damage':
39         return 4
40     elif row_label == 'Inside_puncture':
41         return 5
42     elif row_label == 'impurities':
43         return 6
44     else:
45         return 0
46
```

Figure 3.18: generate_tfrecord.py code updating according to the class map [4]

Then tfrecord files are generated by using following commands from /object_detection folder:

python generate_tfrecord.py –csv_input= images/train_labels.csv –image_dir= images/train –output_path= train.record python generate_tfrecord.py –csv_input= images/test_labels.csv –image_dir= images/test –output_path= test.record

This create two record files, namely; train.record file and test.record file in /object_detection which are employed in training the new defect detection model.

### 5. Creating the label map and setting up the training

Creating the label map and setting up the training configuration file are the final steps to be followed before training the model.

5a.Creating the label map

Label map provides information to the trainer about each object; their class names and the class ID numbers according to the mapping. Here I have used a text editor when producing a new file. It is then saved as labelmap.pbtxt in the ~/tensorflow1/models/research/object_detection/ training folder.

ID numbers of the label map must be identical to which are created in the generate_tfrecord.py file as shown in Figure 3.17 and Figure 3.18

5b. Setting up training

As the last step, the object detection training pipeline should be set up. This defines what parameters and which model are used for training.

Next, go to ~tensorflow1/models/research/object_detection/samples/configs, copy the faster_rcnn_inception_v2_pets.config file into the /object_detection/ training directory. After that, the file which has a text editor should be opened. Some modifications are necessary to be done in the .config file, predominantly adding the file path to the training data and replacing the number of classes as well as examples.

Below flow chart in Figure 3.19 is shown modifications have been performed to the faster_rcnn_inception_v2_pets.config file.

**6. Training of the model**

In order to begin training, following command is used from the /object_detection directory.

python train.py –logtostderr –train_dir = training –pipeline_config_path = trainingfaster_rcnn_inception_v2_pets.config

Next, TensorFlow will initialise the training if everything has been configured

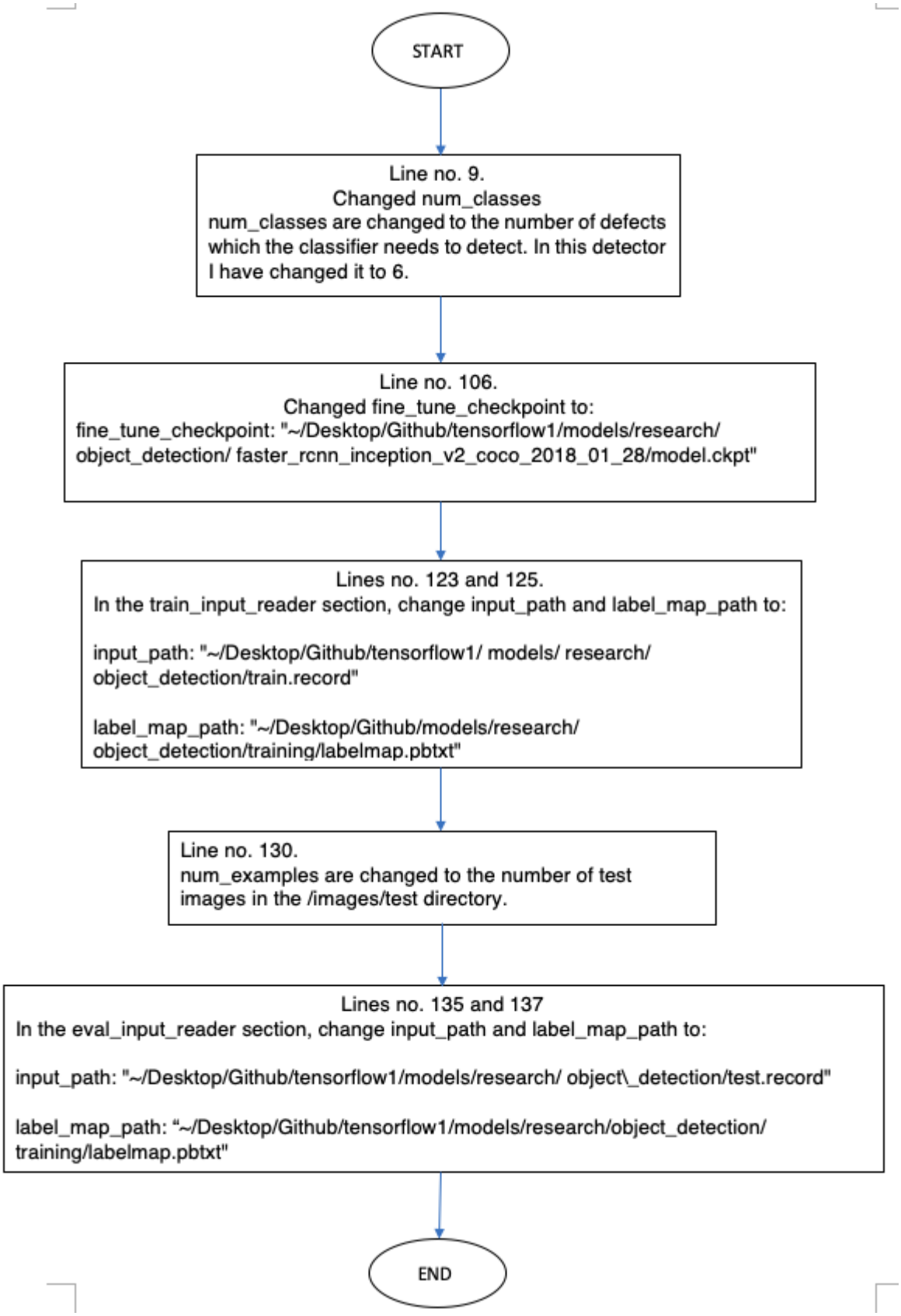



Figure 3.19: Flow chart for setting up training

correctly. The initialisation process can usually consume up to 30 seconds prior to starting the actual training.

The loss is reported in each step of the training. It usually starts with a higher loss and gradually decreases its value to a lower level of loss as the training progresses. It started at a value of 3.0 and quickly dropped lower than 2.5 during the training which I conducted on the Faster-RCNN-Inception-V2 model. I suggest to allow the model to train till the loss falls lower than 0.05 for which it would take around 8 hours or in other words 60,000 steps or more.

To view the Tensorbord, below command can be entered to a new terminal after activating the Anaconda tensorflow1 in the directory of ˜/tensorflow1/ models/ research/ object_detection.

(tensorflow1) ˜/object_detection$ tensorboard –logdir=training

Next, a webpage would be created on the local machine that could be opened through a web browser. Information will be displayed on the TensorBoard as graphs which provide an idea regarding the progression of the training. Loss graph can be emphasized as the most important graph out of all that highlights the total loss of the training model over time. As shown in below Figure 3.20, the advancement of the training job can be viewed using TensorBoard.

| Algorithm | Features | Prediction time / image | Limitations |
|---|---|---|---|
| CNN | First, Image is divided into multiple regions. Each region is then classified into different classes. | – | Many regions are required to predict accurately which result in higher computation time. |
| RCNN | Selective search is used to create regions. Around 2000 regions are usually extracted from every image. | 40-50 seconds | As a result of passing each region separately to the CNN, higher computation time is resulted. Three different models are used to make predictions. |
| Fast RCNN | Every image is proceeded to the CNN once only. Then the extraction of feature maps are done. In order to create predictions, selective search is utilized on these maps. All of the three models which are used in R-CNN are combined together. | 2 seconds | As a result of selective search being slow, higher computation time is consumed. |
| Faster RCNN | Method of selective search is replaced with region proposal network (RPN) and this shapes the algorithm into a much faster one. | 0.2 seconds | Object proposal consumes time and as a result of having many different systems which work consecutively, performance of the system is depending mainly on the performance of the previous system. |

Table 3.1: Performance comparison of object detection algorithms [31]

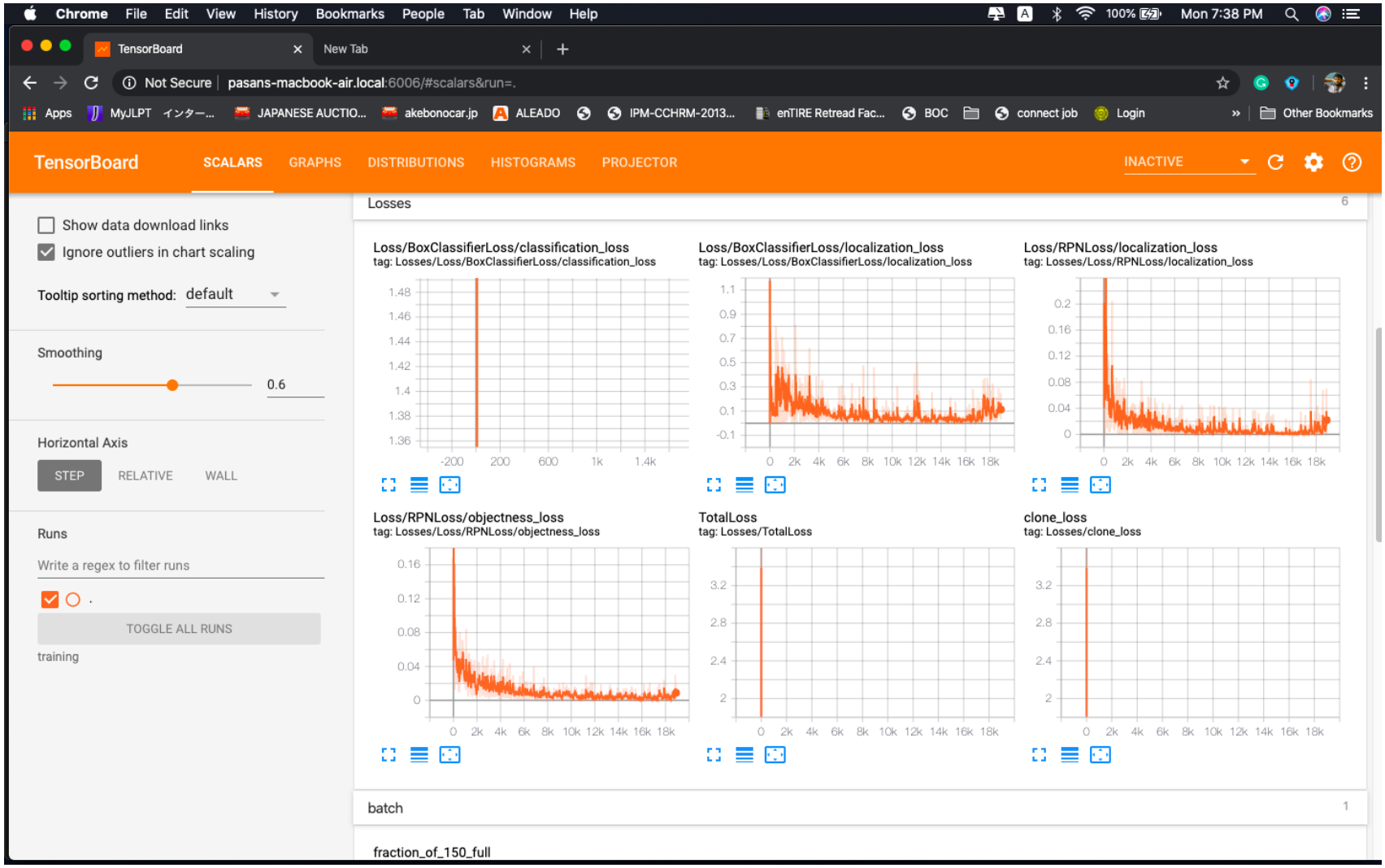



Figure 3.20: View of the Tensorboard while training the model

Checkpoints are created every five minutes occasionally so as to save the progress of the training. The training process could be terminated at anytime by pressing Ctrl+C during training of the model. Generally, I terminate the training immediately after a checkpoint is saved. Once the training is terminated, it could be restarted later from the checkpoint which was last saved. The checkpoint with a highest number of steps is utilized when generating the frozen inference graph [20].

**7. Exporting and analysing the inference graph**

Once the training is completed, generating the frozen inference graph is the final most step which needs to be followed. To achieve this, from the /object_detection folder, below command is issued, in which "XXXX" in "model.ckpt-XXXX" must be replaced in the training folder using the highest-numbered .ckpt file.

(tensorflow1) ~/object_detection$ python export_inference_graph.py –input_type image_tensor –pipeline_config_path training/ faster_rcnn_inception_v2_pets.config –trained_checkpoint_prefix training/ model.ckpt-XXXX –output_directory inference_graph

A frozen_inference_graph.pb file in the /object_detection/inference_graph folder is then produced. Object detection classifier is contained in the .pb file.

**8. Testing the model**

Now, that the Tire Defect Detection Classifier is completely ready to be executed, a Python script which was written by myself was used to test it out on a webcam feed, video or an image [7, 22].

It is necessary to type "idle3" to run any python script in the Anaconda Command Prompt at a time of "tensorflow1" virtual environment is activated.

Some tested images are as shown in below Figure 3.21, Figure 3.22 and Figure 3.23

Figure 3.21: Tested image from the trained model

Figure 3.22: Tested image from the trained model

Figure 3.23: Tested image from the trained model

Chapter 4

# DEFECT DETECTION USING SOUND ANALYSIS

## 4.1 Hammering test

Heirtzler et al. (2002) have invented a tire defect detection system which includes a support structure for receiving a tire; an actuator for impacting the tire, a microphone for receiving the generated sound signal and a computer responsive to the microphone. This computer in the system is programmed to compare the properties of the resulting sound wave with the stored properties indicative of a defect in order to determine whether a defect exists in the tire [34].

However, the practice of the small scale tire retreading companies in developing countries is to perform a hammering test using a brass rod instead of an actuator and listen and differentiate the resulted sound signal wave of a healthy spot and a defective spot using expertise human labour in replace of an advanced computer system.

Hammering test is a famous practice among the local tire retreading industry and mostly this method is used as an economical testing method to identify hidden damages such as inner ply separations, ply damages of the tire, small air bubbles in tread area and etc. Basically in this test, the operator strikes to surface of the tire from a brass rod to all over tread area of the tire. The ordinary activity is shown in below Figure 4.1 and the using hammer is shown in Figure 4.2. When hammering with an even force, healthy spot generates a clear sound signal and this sound signal consist of the same wave length. But when operator strikes to

the defective spot, the signal gets distorted and the wavelength differs from the previous wavelength of the healthy spot. The sound signal of the defective spot is completely depending on the defect condition and the size of the defect [9, 15]. The basic mechanism of the sound test can be described as shown in Figure 4.3. As an example, if the defect is a ply damage in the middle of the tread area, then the size of ply loosen area would be affect for the sound signal of the defective area.

Figure 4.1: Listening the noise while hammering to worn tire

Figure 4.2: Brass rod which is taken to hammer the tire

In this research, I have monitored and analysed the existing hammering method and captured the signals of defective spots and the healthy spots. These signals

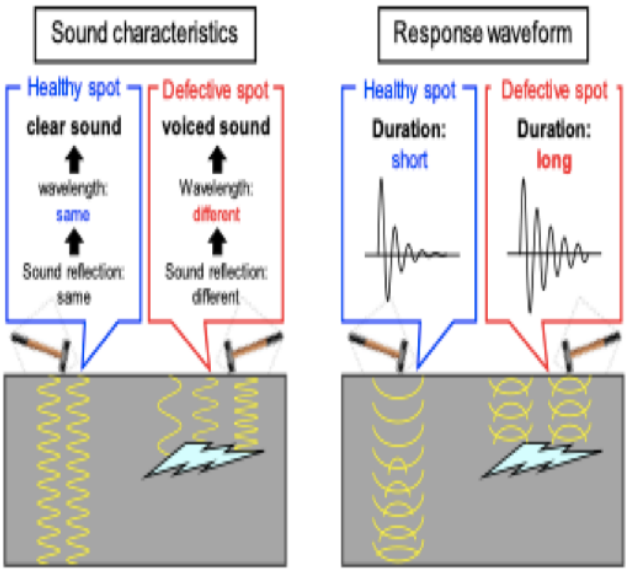



Figure 4.3: Basic mechanism of the sound test [9]

were captured from the USB microphone and Audacity software was used to analyse the audio signal. Audio signal was sampled at a 44100Hz sampling rate and the data were taken to a spectrogram [16]. Sound level (dB) vs frequency(Hz) in Figure 4.4 shows a clear difference in the defective spot compared to a healthy spot on same tire.

## 4.2 Sound classification with TensorFlow

At present, there exist various types of services as well as projects used for human speech recognition such as Google's Speech API and several others. Although such kind of applications are capable of recognising speech to text with a considerable amount of good quality, none of them has the ability to determine discrete sounds which are captured by the microphone [23, 30]. In my research area, the main problem is classifying the sound from defective spot and the healthy spot. Normally experienced labour identifies the sound difference with their experience. But my problem is, if this activity is supposed to be carrying by a novice labour, then the defects identification would be really hard and there will be a possibility

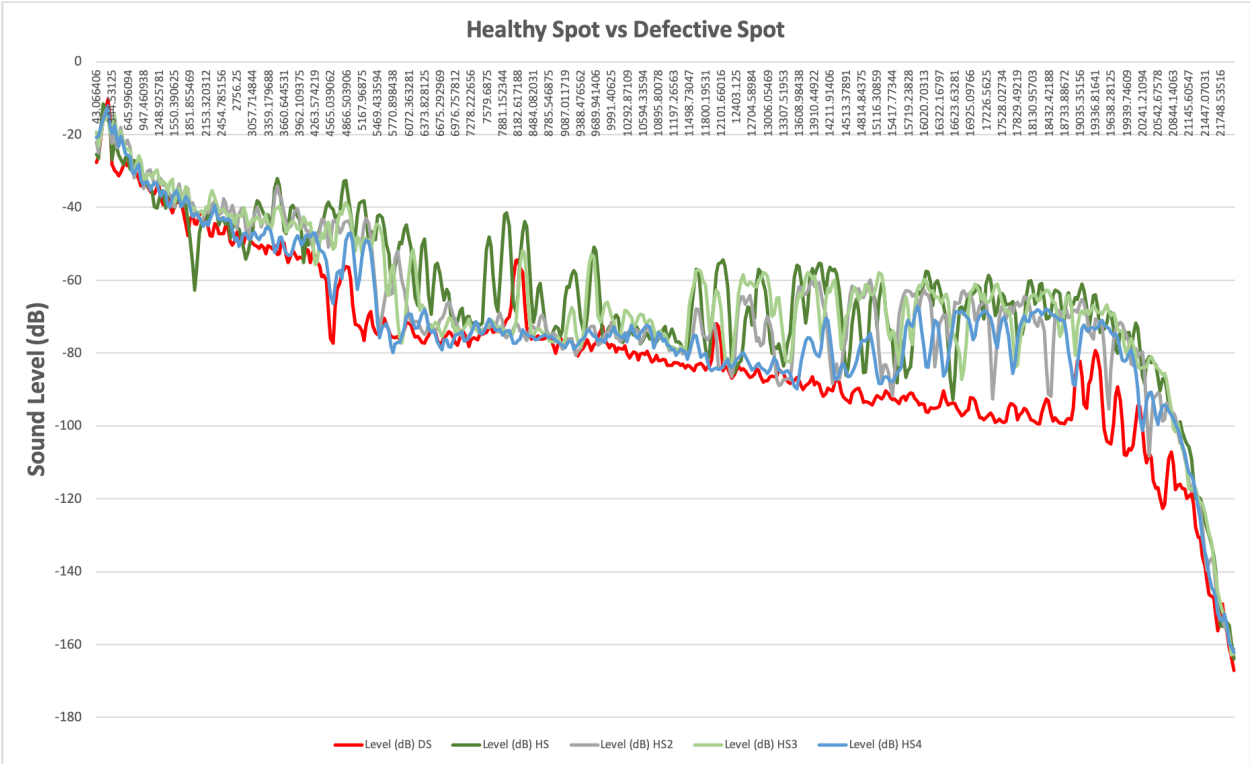



Figure 4.4: Sound signal behaviour of defective spot(DS) vs healthy spot(HS)

to mis diagnose the defective spot. Therefore if we can make a model to identify the difference of the both sound signals, then the problem would be solve up to a certain extend.

In order to achieve the above mentioned proposed solution, I investigated and analysed the difference of both the sound signals. Then, I decided to build a model which has the ability to classify two discrete sounds with the use of machine learning algorithms. In this chapter I have explained the types of tools which I used, the challenges I faced and how I overcame them, procedure followed when training the TensorFlow model and the method of running it in open source software.

### 4.2.1 Choosing Tools and a Classification Model

Initially, I had to chose a software which has the capability to work with neural networks similar to how I did in above Tire Defect Detection model. The very

first appropriate solution which I came across was Python Audio Analysis. A good training dataset is the main problem when it comes to machine learning. Therefore to get a good training dataset, a good quality sound analysing software was a compulsory requirement. After some research, I found Audacity open source software which can record and analyse my hammering sound signals according to my requirement.

Below problems were encountered by myself after performing some testing.

pyAudioAnalysis was not flexible enough and a wide variety of parameters were not considered while some of them were calculated on the fly.

Google AudioSet was the next option which I came across. Google AudioSet is developed based on Youtube video segments which are labeled and is able to download in two formats: [24]

1. CSV files which contains ID of YouTube video, one or more labels, start time and end time of each segment.

2. Audio features which are extracted and stored as record files in TensorFlow

The above mentioned features have compatibility with YouTube-8M models. TensorFlow VGGish model is also offered as a feature extractor by the Google AudioSet [30]. A larger portion of my requirements were covered by this solution and therefore, this was the best choice and most suitable solution for my research.

### 4.2.2 Training the Model

The next piece of work was to understand how the interface of the YouTube-8M works. The interface is designed to work with both video and audio. Although it contains number of sample classes which is hardcoded, the library appeared to be moderately flexible. Therefore, it was slightly modified to pass the number of

classes as a parameter. In my study I only used two classes (defective spot and healthy spot).

YouTube-8M has the ability to work with two types of data, namely; frame features and aggregated features. As I have mentioned before, Google AudioSet has the ability to produce data as features. By researching a bit further I found that features exist in the frame format. Next requirement was to select the model needed to be trained.

### 4.2.3 Resources, Time, and Accuracy

As I have mentioned earlier, GPUs are a more appropriate option than CPUs for machine learning. But in this research I have trained this model by using only TensorFlow CPU.

In my research, it really didn't matter about the training time. I must highlight the fact that, to produce a preliminary decision regarding the selected model as well as its accuracy, around 1 hour of training was sufficient. I wanted to achieve the largest probable accuracy. But, in order to train a model which is further complex and provides better accuracy, it was required to have more RAM to fit it in, video RAM in the case of GPU.

### 4.2.4 Choosing the Model

YouTube-8M dataset which is published by Google AI includes more than 6 million YouTube videos of 2.6 billion audio. Furthermore, the dataset consists of visual features along with more than 3,700 of visual entities which are associated on an average of 3.0 labels per video. Every video was decoded until the first 360 seconds at 1 frame-per-second. Then, features were extracted through a pre-trained model. Basically this model allows video classification and I have changed this model to an audio classifier while eliminating the video inputs. Mainly there

are two types of YouTube -8M models depending on the running location and the models are as follows [30].

1. Running on local computer
2. Running on Google's Cloud Machine Learning Platform

Since my dataset was only including audio files and lesser capacity, I chose to run on my local computer. When running on the local computer it trains at a slower rate and for this model it was sufficient and it took about 1 hour to train the model. If the model supposed to be fed with video, then the local machine must have higher capacity in RAM and TensorFlow GPU to train faster. However, the training data existed in the frame format and therefore, it was necessary to use frame-level models for this model. Even-though this model allows a higher number of classes, I have used only two classes (defective and healthy) in this study.

### 4.2.5 Training with balanced dataset

In a balanced training dataset must be have at-least minimum 5 seconds duration each. The Audacity software is getting the microphone signal at a sampling rate of 44100 Hz and once the digital audio files fed into the model in .wav format, this will resample at a sampling rate of 16000 Hz.

The training command appears as shown below:

python train.py –train_data_pattern= /path_to_data/ audioset_v1_embeddings/ bal_train/*.tfrecord –num_epochs=100 –learning_rate_decay_examples =400000 –feature_names= audio_embedding –feature_sizes =128 –frame_features –batch_size= 512 –num_classes= 527 –train_dir= /path_to_logs –model =ModelName

### 4.2.6 Training with unbalanced dataset

Initially, I made my dataset only with the transient sound signal. wave duration of each sound signal was less than 2 seconds. Therefore it was not enough for training the model and always tend to finish the training without taking much time ( less than 2 minutes ). Then the result was also not accurate. Therefore to get a balanced train , the dataset must be very clear with having a minimum duration of 4 seconds to 5 seconds for each sample.

Chapter 5

# LOCALISATION OF DEFECTS

## 5.1 Defect localisation mechanism

In order to eliminate human labor, it is necessary to automatically recognise the location of defects which are identified via image processing, sound signal classification and metal detection. To achieve this goal, I have used a rotary encoder to motor shaft which enable the identification of the location of a defect with respect to a reference point (in other words, starting point of rotation). This process is called defect localisation in the tire inspection system. Below Figure 5.1 illustrates how the rotary encoder is mounted to the motor rotor shaft.

This encoder output signal is taken to a micro controller (Arduino Uno). Tire rotation and the rotating speed (rpm) of the tire are calculated and it is saved in the SD memory card. SD card saves these data in .csv file in every second.

In this application, tire rotation speed is constant and it is fine tuned to the optimum level which is 1.5 rpm. (40 seconds per round)

Starting reference point is the location in which the machine is set to Auto mode and the start button to switch ON is pressed. Then micro controller takes the signal as the reference point and the rotation of the tire is calculated in degrees with respect to the starting point. Starting point is marked by the operator and it is shown in Figure 5.2.

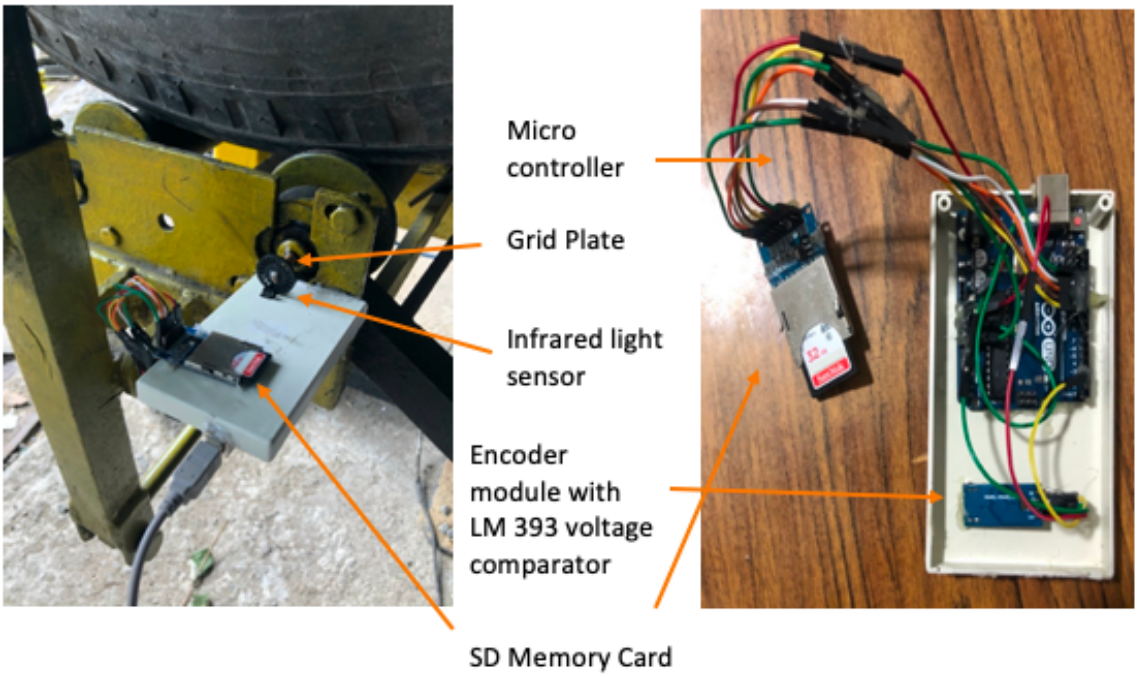



Figure 5.1: Encoder mounted to motor rotor shaft

Figure 5.2: Marked reference point

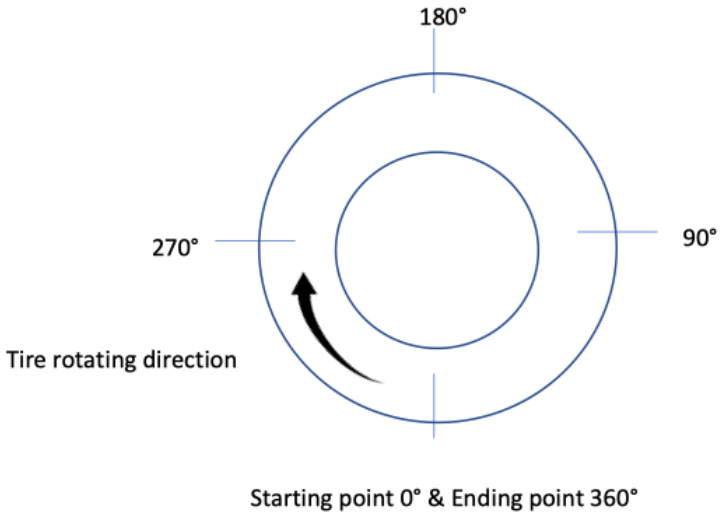



Figure 5.3: Direction of the tire rotation

Tire defects are identified under three different methods in this tire inspection machine as discussed in previous chapters. They are as follows.

1. Image processing
2. Sound signal analysis
3. Metal detection

The defect localisation process for each of those methods is illustrated below in the flow charts in Figure 5.4.

If there is a defect identified by any model (from image processing or sound analysis or metal detection), then the defect will save in the third column of REPORT.csv file in SD card with the tire rotation with respect to the reference point as shown in below Figure 5.5.

Least count measurement of the defect localisation system is 9 degrees. Therefore, the results obtained for the defect localisation is associated with an accuracy of + or - 2.5%.

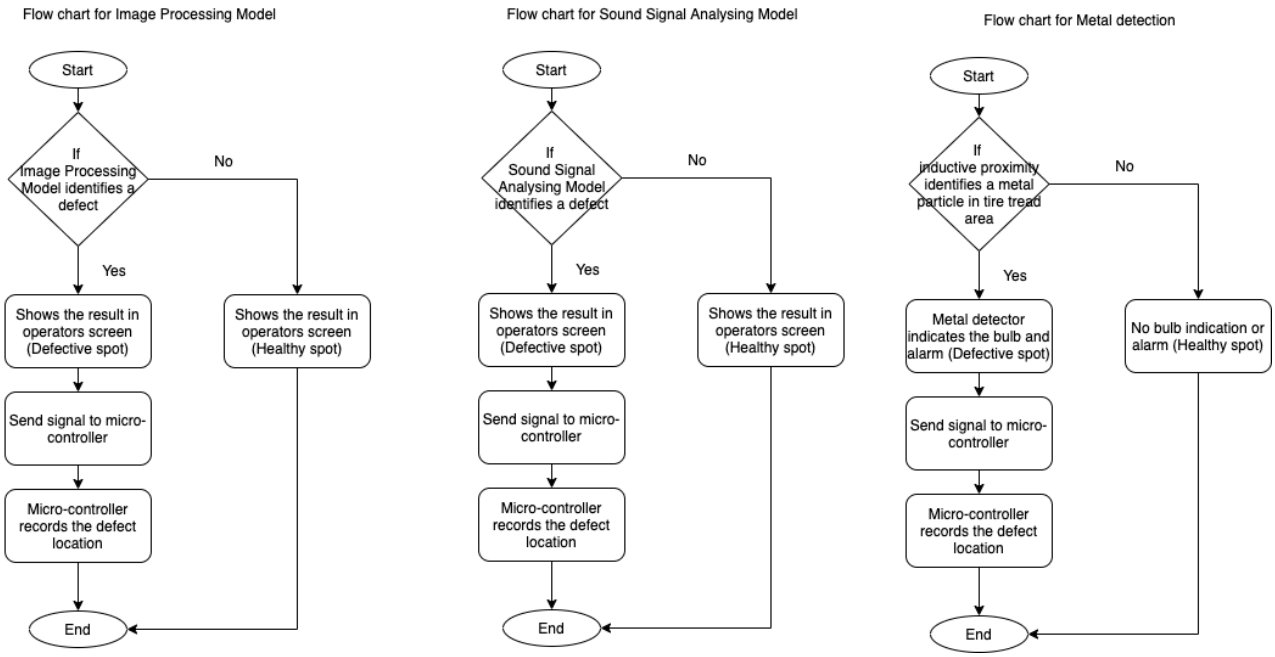
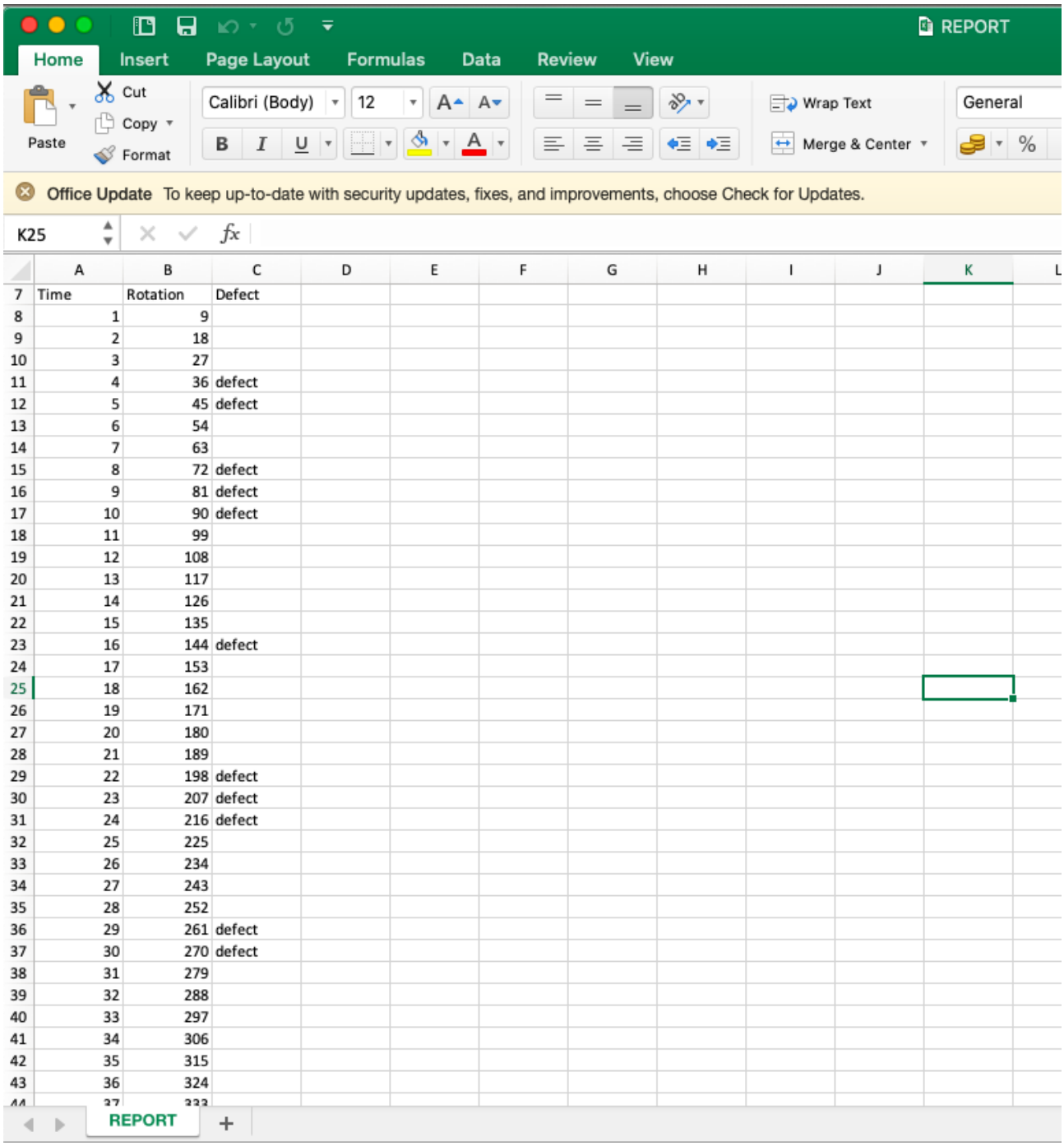



Figure 5.4: Flow chart of defect localisation for each method

To enhance the accuracy and the reliability of the defect localisation system, an industrial heavy-duty rotary encoder can be used for the motor rotor shaft instead of proto-type encoder.

## 5.2 Operation of the tire inspection machine

Once you decide under which method the tire which is placed on the inspection machine has to be tested (image processing, sound signal analysis or metal detection), you can select the method on the computer. Then operator press the switch on button, and the micro controller takes the signal from the encoder as the starting location as the reference point. Then the tire starts to rotate and the defects under the selected method are identified and the location of the defects with respect to the starting reference point are recorded. The type of the identified defect with location in degrees are visible to the operator in the computer screen. The rotation takes place until the operator is satisfied with inspection and stops the machine.



REPORT

Home Insert Page Layout Formulas Data Review View

Cut Copy Format Paste

Calibri (Body) 12

Wrap Text General

Merge & Center

Office Update To keep up-to-date with security updates, fixes, and improvements, choose Check for Updates.

K25

| | A | B | C |
|---|---|---|---|
| 7 | Time | Rotation | Defect |
| 8 | 1 | 9 | |
| 9 | 2 | 18 | |
| 10 | 3 | 27 | |
| 11 | 4 | 36 | defect |
| 12 | 5 | 45 | defect |
| 13 | 6 | 54 | |
| 14 | 7 | 63 | |
| 15 | 8 | 72 | defect |
| 16 | 9 | 81 | defect |
| 17 | 10 | 90 | defect |
| 18 | 11 | 99 | |
| 19 | 12 | 108 | |
| 20 | 13 | 117 | |
| 21 | 14 | 126 | |
| 22 | 15 | 135 | |
| 23 | 16 | 144 | defect |
| 24 | 17 | 153 | |
| 25 | 18 | 162 | |
| 26 | 19 | 171 | |
| 27 | 20 | 180 | |
| 28 | 21 | 189 | |
| 29 | 22 | 198 | defect |
| 30 | 23 | 207 | defect |
| 31 | 24 | 216 | defect |
| 32 | 25 | 225 | |
| 33 | 26 | 234 | |
| 34 | 27 | 243 | |
| 35 | 28 | 252 | |
| 36 | 29 | 261 | defect |
| 37 | 30 | 270 | defect |
| 38 | 31 | 279 | |
| 39 | 32 | 288 | |
| 40 | 33 | 297 | |
| 41 | 34 | 306 | |
| 42 | 35 | 315 | |
| 43 | 36 | 324 | |

REPORT

Figure 5.5: Report generated in SD memory card

A comparison between the results obtained through manual operation by a novice employee and automatic defect detection and localisation by the developed tire inspection machine for all three defect detection methods are included under Chapter 6.

Chapter 6

# RESULTS AND DISCUSSION

## 6.1 Defect identification and classification using image processing

When starting the tire inspection under image processing method, the point at which the machine is set to Auto mode and switch ON is pressed is marked and taken as the reference point. Then micro controller takes the signal as the reference point and tire rotation is started. The defects which are identified via

In addition, I have fixed a laser light for each camera and it helps the operator to identify the exact defect location of tire from the computer screen. Therefore, once the model identifies any defect, then localisation system identifies the location of the defect with respect to reference point. So, it is made easy to the operator with automatic identification and localisation of defects.

When finding the optimum speed of the the tire rotation, I have considered the frame capturing speed and the tire rotation speed. Therefore, to minimised the error of capturing the defects, I have fine tuned the rotational speed where frame can capture any point of tire in three frames. The Figure 6.1 shown in below shows how a single point of tire captures in consecutive frames. The Optimal speed is calculated as 1.5 rpm ( 40 seconds per round ).

The detected spots are shown in Figure 6.2 and Figure 6.3.

A trial was conducted on several tires which were consisting surface defects


Figure 6.1: Same point in three consecutive frames]


Figure 6.2: Results from the trained model - Outside_damage [Precision 94%]


Figure 6.3: Results from the trained model - Outside_damage [Precision 96%]

and they were examined by an expertise . I have tested for several tires which are having surface defects. Those defects were normally identified by the expertise. Trial data are as follows in below table 6.1.

This image identification and classification model was evaluated by getting samples from 10 tires and there were 24 considerable defects such as outside damage, impurities, bead damage and ply damages. From these 10 tires, 19 defects were correctly identified by the image processing model. Therefore this model is having a 79.16% accuracy according to the collected sample dataset. This accuracy can be improved by continuously training with more variety of defect and non defect complex images.

## 6.2 Sound signal classification model

In general sound test is operated by manual work and each striking power is usually different. But in this analysis, we have tried to strike with the same

| No. | Sample | Tire size | No. of defects (Inspected by expertise) | No. of defects identified by novice employee | Time taken for inspection (mm:ss) | No. of defects identified by image processing model | Time taken for inspection (mm:ss) |
|---|---|---|---|---|---|---|---|
| 1 | Tire 1 | 9.00-20 | 3 | 2 | 3:30 | 2 | 04:30 |
| 2 | Tire 2 | 9.00-20 | 2 | 2 | 3:00 | 1 | 04:30 |
| 3 | Tire 3 | 9.00-20 | 3 | 2 | 3:00 | 2 | 04:30 |
| 4 | Tire 4 | 9.00-20 | 4 | 3 | 3:30 | 3 | 04:30 |
| 5 | Tire 5 | 9.00-20 | 2 | 1 | 2:30 | 2 | 04:30 |
| 6 | Tire 6 | 9.00-20 | 1 | 1 | 3:00 | 1 | 04:30 |
| 7 | Tire 7 | 9.00-20 | 2 | 1 | 2:30 | 2 | 04:30 |
| 8 | Tire 8 | 9.00-20 | 3 | 1 | 3:00 | 2 | 04:30 |
| 9 | Tire 9 | 10.00-20 | 2 | 1 | 3:00 | 2 | 04:30 |
| 10 | Tire 10 | 10.00-20 | 2 | 2 | 3:30 | 2 | 04:30 |
| | Total | | 24 | 16 | | 19 | |
| | Accuracy of novice employee | | | | — | 66.7% | |
| | Average time taken for inspection (novice employee) | | | | — | 03:03 | mm:ss |
| | Accuracy of developed model | | | | — | 79.2% | |
| | Average time taken for inspection (developed model) | | | | — | 04:30 | mm:ss |

Table 6.1: Trial data of image processing model

power to keep a consistent impact all over the tire and here I have assumed that all strikes have the same power [12]. The waveforms of a healthy spot and defective spot are shown in below from Figure 6.6 to Figure 6.7. These wave durations are calculated up on the wave settling time and this wave settling level is taken where signals is maintaining its amplitude below + or - 5%.

This samples are taken from the same tire at two different spots which are in healthy and defective. Audio signals of these samples have analysed from Audacity Software and the output graph show clearly about the difference of two signals. Sample audio signals of healthy spots and defective spots are shown in Figure 6.4 and Figure 6.5.

Healthy spot has less duration and less amplitude compared to the defective signal which is having a ply damage in tire tread area. In conventional method,

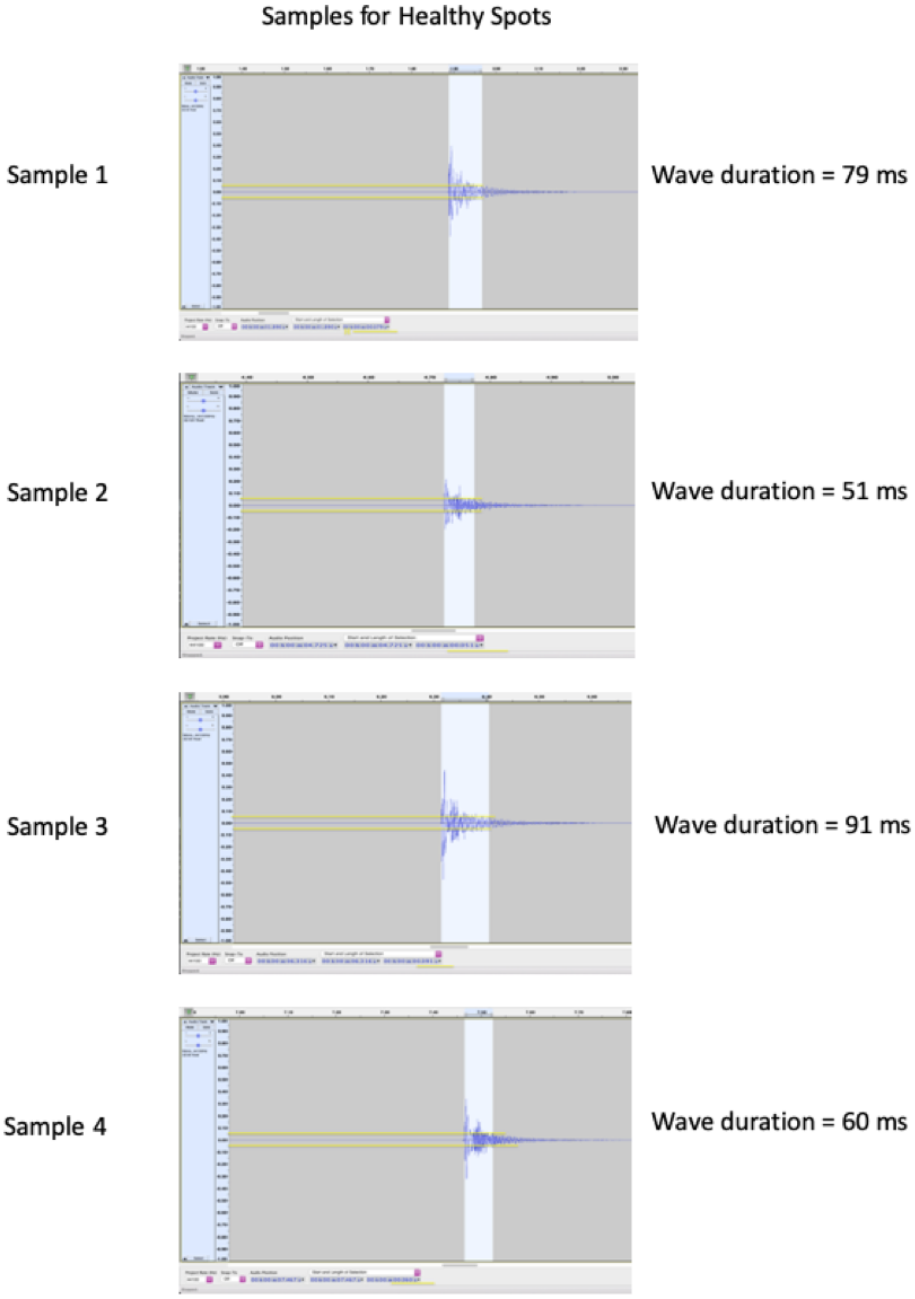



Figure 6.4: Sample audio signals of healthy spots

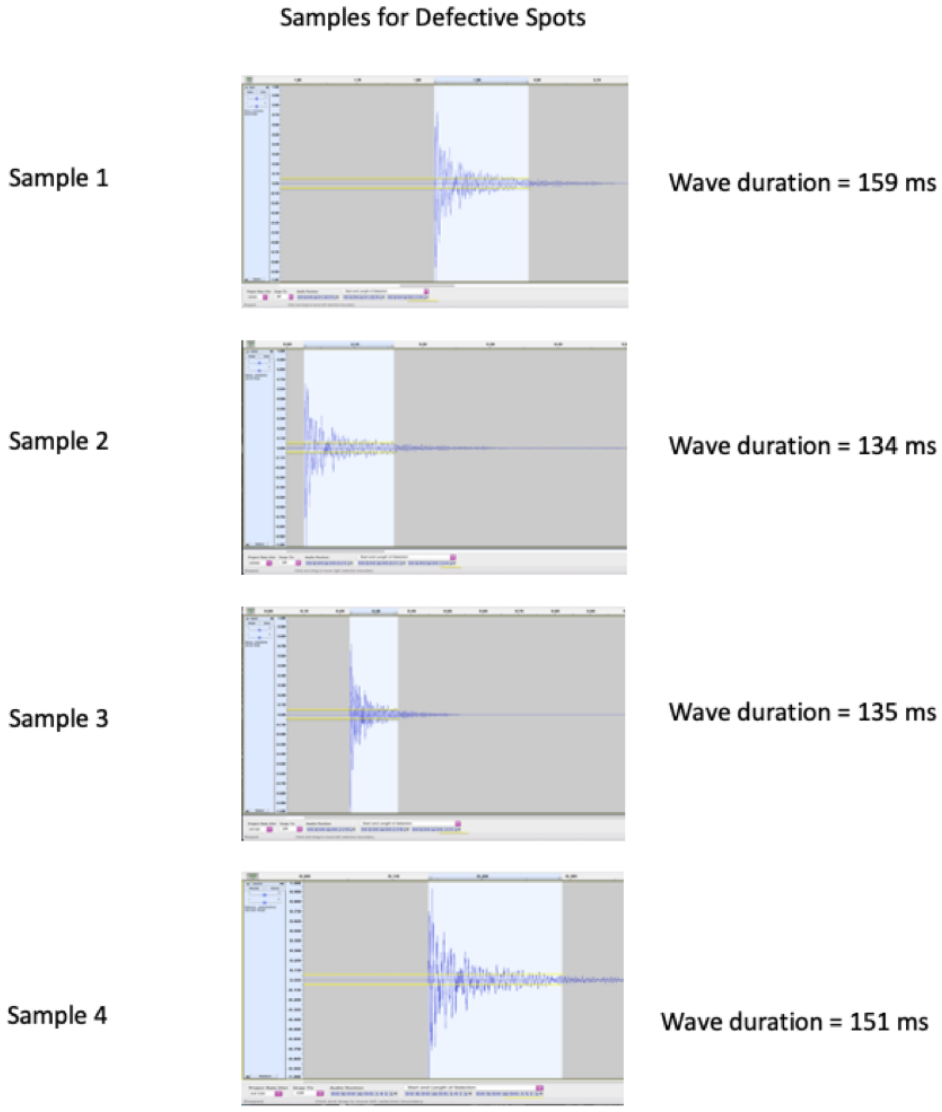



Figure 6.5: Sample audio signals of defective spots

this difference is observed by an expert human resource.

Figure 6.6 shows the sound signal of a healthy spot and it has a duration of 79ms. Sound signal of the defective spot shows in Figure 6.7 and it has duration of 159ms. Therefore we can clearly observe the difference between the sound signals of a healthy spot and a defective spot of a same tire. Wavelengths and the signal duration completely differs from tire to tire since worn tires are having different thicknesses of the tire tread area. But the main observation was, when striking on any defective tire which has a ply damage shows a higher time duration at their ply loosen or damaged spots compared to their healthy spots. Not only in time domain, even in frequency domain these results show a clear difference as shown in Figure 6.8 and Figure 6.9

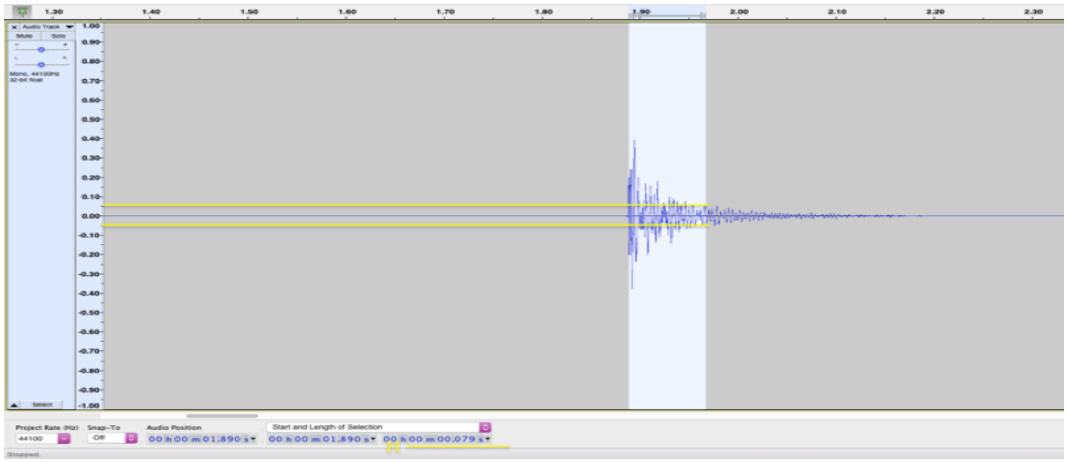


Figure 6.6: Healthy spot- Amplitude vs Time. [wave duration = 79 ms]

Findings can be summarised as in below Figure 6.10 to get a better understanding about the sound signal behaviour

### 6.2.1 Tire inspection via sound classification model

Another sample tire is shown below in Figure 6.11 and this sample tire was diagnosed by the developed sound classification model. Captured sound signals were analysed by Audacity software and the fed into the sound classification model in the format of .wav files. The sample test files are having a time duration of

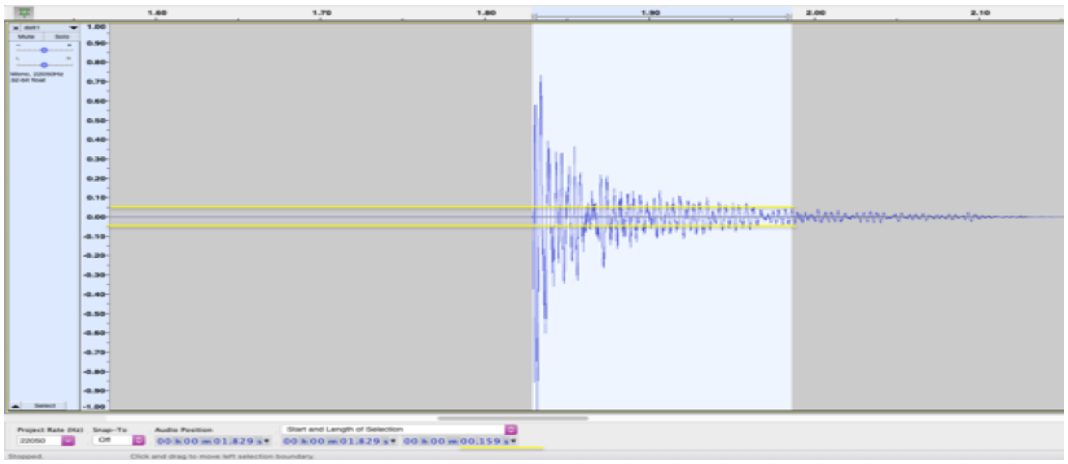


Figure 6.7: Defective spot- Amplitude vs Time. [wave duration = 159 ms]

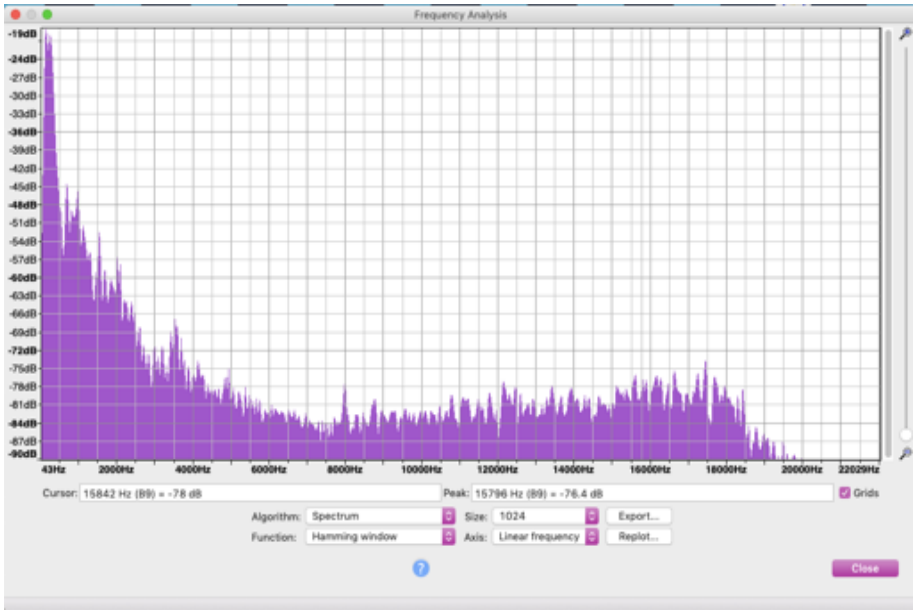



Figure 6.8: Healthy spot- Sound level(dB) vs Frequency

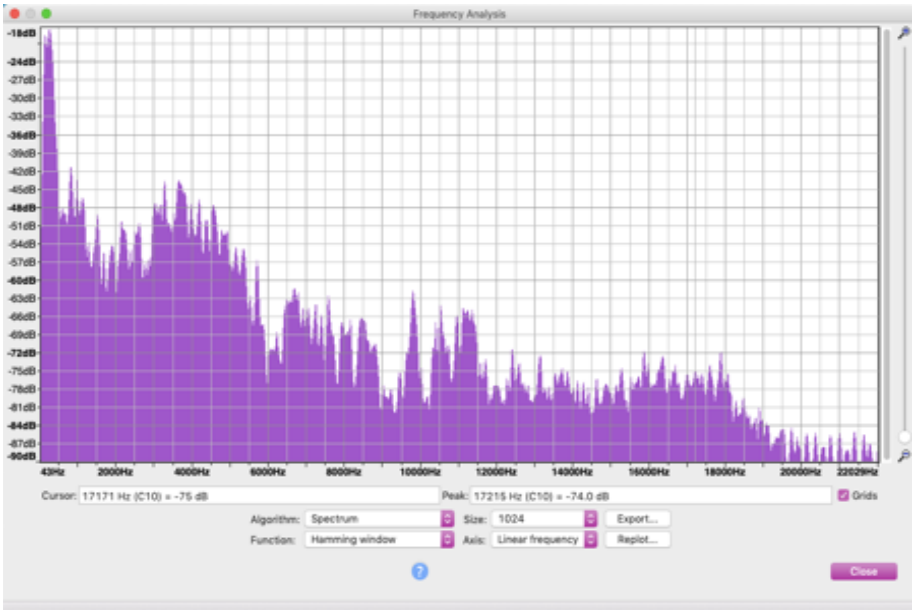



Figure 6.9: Defective spot- Sound level(dB) vs Frequency

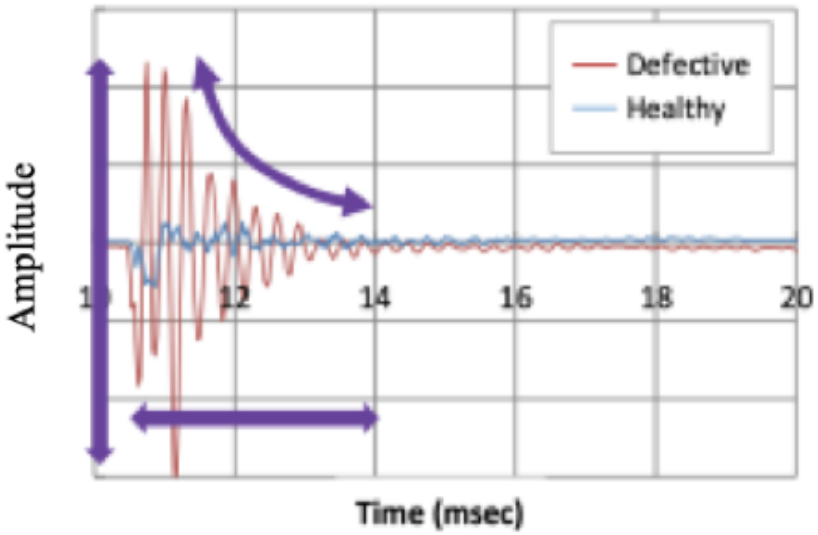



Figure 6.10: Summary of the sound signal behaviour of Healthy spot and Defective spot [9, 16]

(a) Healthy spot (b) Defective spot

Figure 6.11: Healthy spot vs Defective spot

approximately 5 seconds each. The result of the both signals are shown with their accuracy level in Figure 6.13 and Figure 6.15

This model was evaluated by getting samples from 10 tires and there were 14 considerable defects such as ply damages, tread separation and uneven tear with weak tread area. Trial data are as follows in below table 6.2.

From these 10 tires, 10 defects were correctly identified by the sound classification model. Therefore this model is having a 71.42% accuracy according to the collected sample dataset. This accuracy can be improved by continuously training with more variety of sound signals.

Therefore the research study can come to a conclusion where, all internal defects in tread area such as ply damages, ply separation, tread separation, air bubbles and weak tread area can generate a sound signal which has a higher time duration with its amplitude compared to its healthy spots. Not only that, as in the above Figure 6.9 , some considerable frequency difference with high frequency

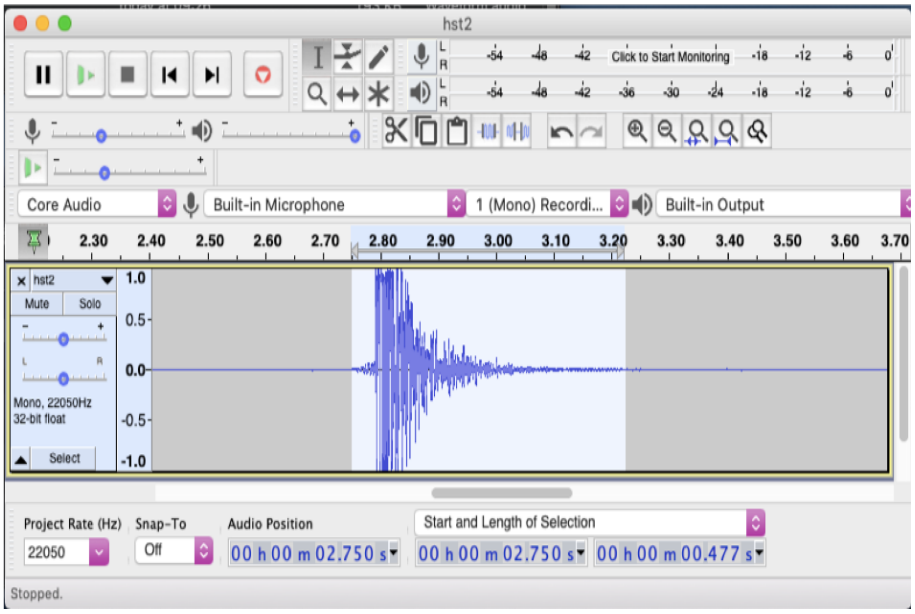



Figure 6.12: Sound signal of healthy spot

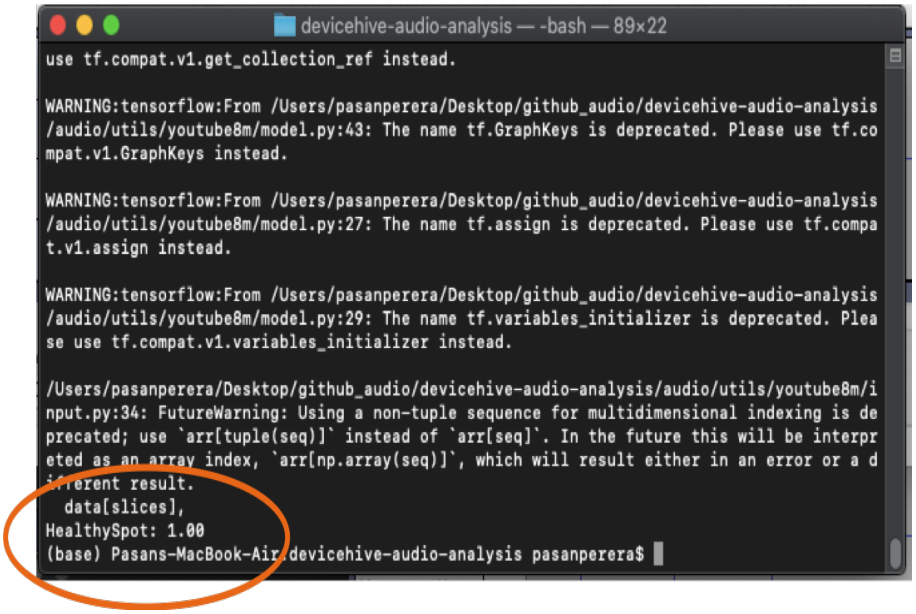



Figure 6.13: Result of the sound signal at healthy spot - [Healthy spot: 1.00]

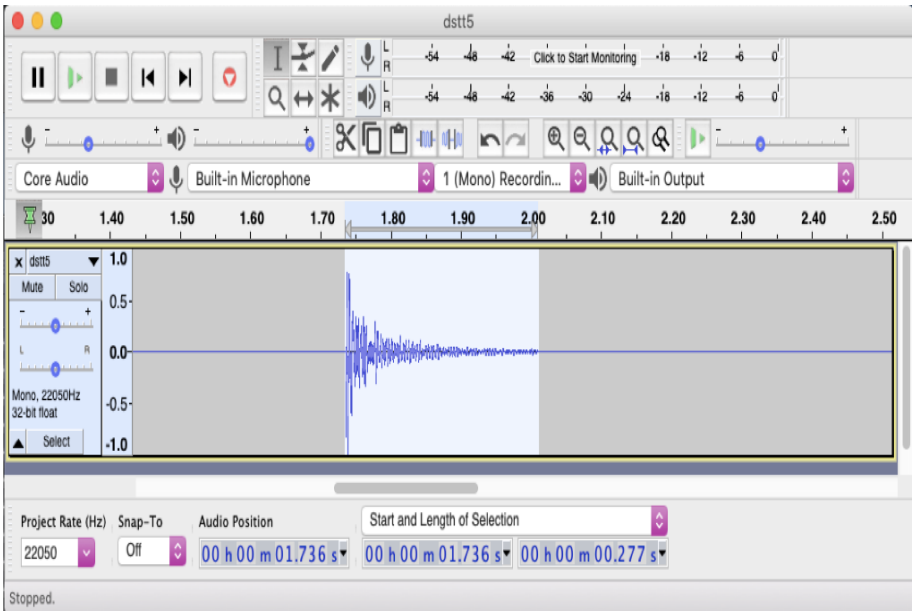



Figure 6.14: Sound signal of defective spot

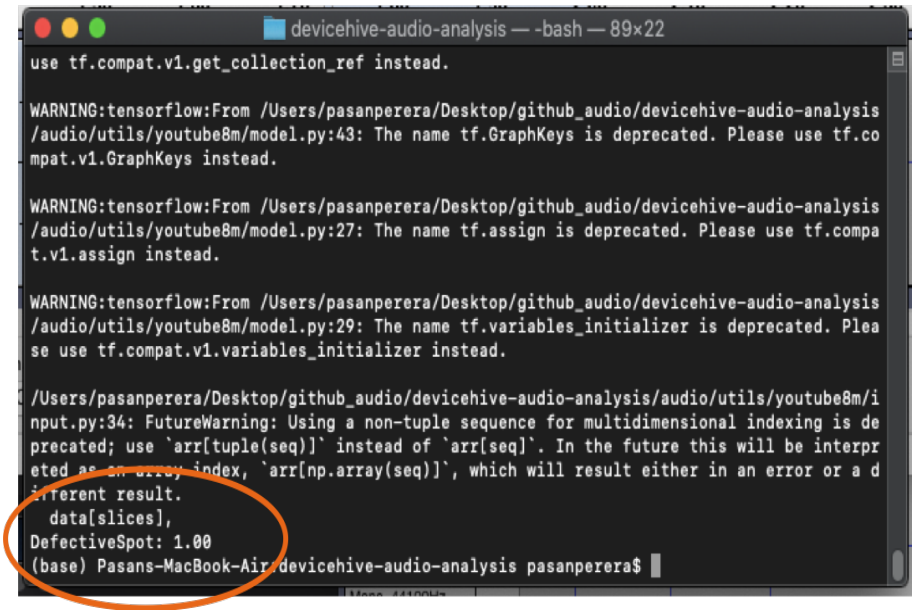



Figure 6.15: Result of the sound signal at defective spot - [Defective spot: 1.00]

| No. | Sample | Tire size | No. of defects (Inspected by expertise) | No. of defects identified by novice employee | Time taken for inspection (mm:ss) | No. of defects identified by sound signal analysis model | Time taken for inspection (mm:ss) |
|---|---|---|---|---|---|---|---|
| 1 | Tire 1 | 9.00-20 | 1 | 0 | 3:00 | 1 | 05:30 |
| 2 | Tire 2 | 9.00-20 | 1 | 0 | 3:00 | 0 | 05:30 |
| 3 | Tire 3 | 9.00-20 | 1 | 1 | 3:00 | 1 | 05:00 |
| 4 | Tire 4 | 9.00-20 | 2 | 1 | 3:30 | 1 | 05:30 |
| 5 | Tire 5 | 9.00-20 | 1 | 0 | 2:30 | 1 | 05:30 |
| 6 | Tire 6 | 10.00-20 | 1 | 1 | 3:00 | 1 | 05:30 |
| 7 | Tire 7 | 10.00-20 | 2 | 1 | 3:30 | 1 | 04:30 |
| 8 | Tire 8 | 10.00-20 | 1 | 0 | 3:30 | 1 | 05:00 |
| 9 | Tire 9 | 10.00-20 | 2 | 1 | 3:30 | 1 | 05:30 |
| 10 | Tire 10 | 10.00-20 | 2 | 1 | 3:00 | 2 | 05:30 |
| | Total | | 14 | 6 | | 10 | |
| | Accuracy of novice employee | | | | — | 42.9% | |
| | Average time taken for inspection (novice employee) | | | | — | 03:09 | mm:ss |
| | Accuracy of developed model | | | | — | 71.4% | |
| | Average time taken for inspection (developed model) | | | | — | 05:18 | mm:ss |

Table 6.2: Trial data of sound signal analysing model

in frequency spectrum can be observed compared to its healthy spots.

### 6.3 Tire inspection via metal detection

This model was evaluated by getting samples from 10 tires and there were 10 considerable defects such as unwanted metal particles and nails in tread area. Trial data are as follows in below table 6.3.

Therefore this model is having a 100% accuracy according to the collected sample dataset.

| No. | Sample | Tire size | No. of defects (Inspected by expertise) | No. of defects identified by novice employee | Time taken for inspection (mm:ss) | No. of defects identified by metal detector | Time taken for inspection (mm:ss) |
|---|---|---|---|---|---|---|---|
| 1 | Tire 1 | 9.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| 2 | Tire 2 | 9.00-20 | 1 | 0 | 1:30 | 1 | 02:00 |
| 3 | Tire 3 | 9.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| 4 | Tire 4 | 9.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| 5 | Tire 5 | 9.00-20 | 1 | 0 | 1:30 | 1 | 02:00 |
| 6 | Tire 6 | 9.00-20 | 1 | 0 | 1:30 | 1 | 02:00 |
| 7 | Tire 7 | 9.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| 8 | Tire 8 | 9.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| 9 | Tire 9 | 9.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| 10 | Tire 10 | 10.00-20 | 1 | 1 | 1:30 | 1 | 02:00 |
| | Total | | 10 | 7 | | 10 | |
| | Accuracy of novice employee | | | | — | 70.0% | |
| | Average time taken for inspection (novice employee) | | | | — | 01:24 | mm:ss |
| | Accuracy of developed model | | | | — | 100.0% | |
| | Average time taken for inspection (metal detector) | | | | — | 02:00 | mm:ss |

Table 6.3: Trial data of metal detection

### 6.4 Localisation of Defects

Defect localisation facility is developed for all three defect detection methods. Therefore whenever a defect detects from any detection method (Image processing or sound signal or metal detection) the microcontroller sends signal to save the defect location in the SD memory card. As I mentioned in chapter 5, the defect localisation system has a least count measurement of 9 degrees. Therefore, the results obtained for the defect localisation is associated with an accuracy of + or - 2.5%. Sample data is show in below table 6.4.

| Time (in seconds) | Rotation with respect to reference point (in degrees) | Remarks |
|---|---|---|
| 0 | 0 | |
| 1 | 9 | |
| 2 | 18 | |
| 3 | 27 | |
| 4 | 36 | defect |
| 5 | 45 | defect |
| 6 | 54 | |
| 7 | 63 | |
| 8 | 72 | defect |
| 9 | 81 | defect |
| 10 | 90 | defect |
| 11 | 99 | |
| 12 | 108 | |
| 13 | 117 | |
| 14 | 126 | |
| 15 | 135 | |
| 16 | 144 | defect |
| 17 | 153 | |

Table 6.4: Sample data of defect localisation system

Chapter 7

# CONCLUSIONS

In this study, a tire inspection machine for tire retreading industry is developed. The developed inspection machine eliminates the complexity of the inspection activity and the requirement of expert labor as well.

The summary of the contribution and conclusions are as below.

- Defect detection model is developed using faster RCNN (Regional Convolution Neural Network) algorithm with inception v2 coco feature extractor. This model shows more than 79% accuracy according to the sample data.
- Model developed for sound identification and classification to identify the internal ply separations and damages which are not visible to naked eyes. This model is having more than 71% accuracy according to the sample data.
- According to this research, defective spots due to ply damages are having a longer duration in their sound signals when compared to the sound signals at healthy spots.
- Sound signal differs from tire to tire depending on the tread quality of the worn tire.
- With this machine, unwanted metal particles can be easily identified by inductive proximity sensors. Earlier these metal particles were diagnosed during the visual inspection and some particles were not visible to the naked

eye due to their size. But with this solution, it is possible to identify even smaller metal particles.

- Designed and implemented a unique structure to this machine which can hold all the feature in an ideal manner.

- The developed machine has the capability of identification and classification of defects which were identified during the visual inspection and classification of sound signals when performing the hammering test. From the obtained results it can be concluded, the above models provide considerable information with a higher level of accuracy to the operator. Therefore the expertise labor who is carrying out the initial inspection process could be replaced by a novice employee. After implementing this machine, it was noticed that most of the doubtful decisions were transformed into clear decisions.

Therefore, this developed mechanism would be a good upgrade for small scale companies who are not capable of affording expensive inspection technologies such as nondestructive defect detection based on X-Ray image processing method. Moreover, using this machine would help in reducing the human labor consumed for initial inspection phase of the whole tire retreading process. Since there is no much maintenance cost associated with the developed machine, small scale companies would be more comfortable with their existing financial situations when using this semi-automatic tire inspection machine.

Appendix A

# APPENDIX

---

## A.1 Code to create pb2.py files

Command line code to create pb2.py files in chapter 3 , 2e. Compile Protobufs and run setup.py section

```
protoc --
python_out=. ./object_detection/protos/anchor_generator.proto ./object_detection/p
rotos/argmax_matcher.proto ./object_detection/protos/bipartite_matcher.proto ./obj
ect_detection/protos/box_coder.proto ./object_detection/protos/box_predictor.proto
./object_detection/protos/eval.proto ./object_detection/protos/faster_rcnn.proto .
/object_detection/protos/faster_rcnn_box_coder.proto ./object_detection/protos/gri
d_anchor_generator.proto ./object_detection/protos/hyperparams.proto ./object_dete
ction/protos/image_resizer.proto ./object_detection/protos/input_reader.proto ./ob
ject_detection/protos/losses.proto ./object_detection/protos/matcher.proto ./objec
t_detection/protos/mean_stddev_box_coder.proto ./object_detection/protos/model.pro
to ./object_detection/protos/optimizer.proto ./object_detection/protos/pipeline.pr
oto ./object_detection/protos/post_processing.proto ./object_detection/protos/prep
rocessor.proto ./object_detection/protos/region_similarity_calculator.proto ./obje
ct_detection/protos/square_box_coder.proto ./object_detection/protos/ssd.proto ./o
bject_detection/protos/ssd_anchor_generator.proto ./object_detection/protos/string
_int_label_map.proto ./object_detection/protos/train.proto ./object_detection/prot
os/keypoint_box_coder.proto ./object_detection/protos/multiscale_anchor_generator.
proto ./object_detection/protos/graph_rewriter.proto ./object_detection/protos/cal
ibration.proto ./object_detection/protos/flexible_grid_anchor_generator.proto
```

Figure A.1: Command line code to create pb2.py files

# BIBLIOGRAPHY


[1] "models/detection_model_zoo.md at master tensorflow/models." [online] Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/ g3doc (accessed Jul. 02, 2020).

[2] "tensorflow/models: Models and examples built with TensorFlow." [online] Available at: https://github.com/tensorflow/models (accessed Jul. 02, 2020).

[3] "EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10: How to train a TensorFlow Object Detection Classifier for multiple object detection on Windows." [online] Available at: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10 (accessed Jul. 02, 2020).

[4] M. Lutz and D. Ascher, Oreilly_Learning_Python, no. March. 1999.

[5] Bandag Incorporated, "Truck Tire Retreading" J. Chem. Inf. Model., vol. 53, no. 9, pp. 1689–1699, 2013.

[6] S. Mallick and D. Ph, "Computer Vision Resources," 2015, [Online]. Available: https://github.com/kjw0612/awesome-deep-vision.

[7] B. A. Meier, Python GUI Programming Cookbook - Second Edition. 2017.

[8] M. Examples, "X-ray sCMOS camera C12849 series," vol. 2048, no. V, pp. 30–31.

[9] Y. W. Lim, "Building Information Modelling for Building Energy Efficiency Evaluation," 6th Annu. Int. Conf. Archit. Civ. Eng. (ACE 2018), no. Ace, pp. 423–426, 2016, doi: 10.5176/2301-394X.

[10] E. O. F. Usage, "C9750-27fcc,-27fcd," pp. 20–21.

[11] Y. Zhang, X. Cui, Y. Liu, and B. Yu, "Tire defects classification using convolution architecture for fast feature embedding," Int. J. Comput. Intell. Syst., vol. 11, no. 1, pp. 1056–1066, 2018, doi: 10.2991/ijcis.11.1.80.

[12] A. Yamashita, T. Hara, and T. Kaneko, "Hammering test with image and sound signal processing," Nihon Kikai Gakkai Ronbunshu, C Hen/Transactions Japan Soc. Mech. Eng. Part C, vol. 72, no. 3, pp. 772–779, 2006, doi: 10.1299/kikaic.72.772.

[13] Q. Guo, C. Zhang, H. Liu, and X. Zhang, "Defect Detection in Tire X-Ray Images Using Weighted Texture Dissimilarity," J. Sensors, vol. 2016, no. Cm, 2016, doi: 10.1155/2016/4140175.

[14] Q. Guo and Z. Wei, "Tire defect detection using image component decomposition," Res. J. Appl. Sci. Eng. Technol., vol. 4, no. 1, pp. 41–44, 2012.

[15] S. N. Mokhatar et al., "A Fundamental Hammering Sound Test to assess the Degree of Deterioration in Reinforced Concrete Structure," IOP Conf. Ser. Mater. Sci. Eng., vol. 431, no. 12, 2018, doi: 10.1088/1757-899X/431/12/122013.

[16] C. A. Rosales, H. J. Lee, W. Kim, and C. K. Park, "Decision matrix analysis of impact sounding test method to determine interlayer condition of concrete bridge deck," J. Sensors, vol. 2017, 2017, doi: 10.1155/2017/8516319.

[17] B. Singh, P. Rajiv, and M. Chandra, Mastering Opencv With Pratical Examples. 2015.

[18] D. A. Teich and P. R. Teich, "PLASTER: A Framework for Deep Learning Performance Whitepaper sponsored by NVIDIA," no. May, 2018.

[19] A. Zelinsky, Learning OpenCV—Computer Vision with the OpenCV Library (Bradski, G.R. et al.; 2008)[On the Shelf], vol. 16, no. 3. 2009.

[20] N. Buduma, Deep Learning DESIGNING NEXT-GENERATION ARTIFICIAL INTELLIGENCE ALGORITHMS. 2016.

[21] X-scanimaging.com. 2014. X-ray U-shape Line-scan Camera Series. [online] Available at: https://x-scanimaging.com/wp-content/uploads/2018/08/xu8800seriesrev1p4.pdf (Accessed Jul. 03, 2020).

[22] B. Meier, Python GUI programming cookbook: over 80 object-oriented recipes to help you create mind-blowing GUIs in Python. .

[23] "Building an audio classifier with your own dataset." Youtube.com. 2021. [online] Available at: https://www.youtube.com/watch?v=eTy6JWFk9Fc (Accessed Jul. 03, 2020).

[24] "AudioSet." [online] Available at: https://research.google.com/audioset/download.html (accessed Jul. 03, 2020).

[25] "X-Ray machine." https://www.directindustry.com/prod/cyxplus/product-64486-1315841.html.

[26] CyXplus - Non Destructive Testing. n.d. X-Ray Inspection Machines - CyXplus - Non Destructive Testing. [online] Available at: https://www.cyxplus.fr/portfolio/x-ray-inspection-machines/ (accessed Jul. 10, 2020).

[27] "Metal detector." https://www.directindustry.com/prod/shenzhen-unisec-technology-co-ltd/product-181386-2293335.html.

[28] "Tire inspection machine." [online] Available at: http://retreadingtech.com/about-370.html. (accessed Jul. 10, 2020).

[29] "Faster RCNN." https://www.researchgate.net/publication/334987612_Real-Time_Diseases_Detection_of_Grape_and_Grape_Leaves_using_Faster_R-CNN_and_SSD_MobileNet_Architectures/figures?lo=1.

[30] "Sound-Classification-With-Tensorflow-8209Bdb03Dfb @ Medium.Com." [Online]. Available: https://medium.com/iotforall/sound-classification-with-tensorflow-8209bdb03dfb.

[31] "Summary of the Algorithms." [online] Available at: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/. (accessed Jul. 1, 2020).

[32] "Precision-and-Recall @ Developers.Google.Com." [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[33] Debo L.G., Van Wassenhove L.N. (2005) Tire recovery: the RetreadCo case. In: Flapper S.D.P., van Nunen J.A., Van Wassenhove L.N. (eds) Managing Closed-Loop Supply Chains. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-27251-8_11

[34] "TIRE DEFECT DETECTION SYSTEM AND METHOD", United States Patent, US 6,381547 B1, Heirtzler et al., Apr. 30, 2002

[35] Rubin Bose S, Sathiesh Kumar V, "Hand Gesture Recognition Using Faster R-CNN Inception V2 Model", July 2019