# DISTRESS DETECTION OF ROAD IMAGES USING DEEP LEARNING

by

Rishabh Sanjay Goud

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Construction and Facilities Management

Charlotte

2019

Approved by:

_____
Dr. Don Chen (Chair)

_____
Dr. Jake Smithwick

_____
Dr. Barry Sherlock

# ABSTRACT

RISHABH SANJAY GOUD. Distress Detection of Road Survey Images using Deep Learning. (Under the direction of (DR. DON CHEN)

The NCDOT road survey involves capturing images of highway and later processing to determine an overall pavement performance. Terabytes of data is captured even over a small highway section, therefore, there is a need to find better solutions to process these images. This study aims to detect distresses on the collected images using deep learning. Previous studies have utilized some deep learning model to classify road survey images, however, this study utilizes Mask-RCNN to detect the different distresses present on an image and also to classify its type. In addition, previous studies have manually labelled images to develop a labelled dataset, however there is no documented record of this process. Therefore, this research enumerates the steps required to develop a labelled dataset from the raw images and utilize these images to develop several Mask-RCNN models. For labelling purposes, this study testes several software including: GIMP, Adobe Lightroom, Adobe Photoshop, and Whitebox. All the software expect WhiteBox are typically utilized in image manipulation. Whereas, WhiteBox is a geospatial analysis software which is utilized in measurement of lake depths or slope of mountain ranges. This study attempted to apply a WhiteBox for marking of road cracks by treating the cracks as mountain ranges at a small scale. Mask-RCNN model was developed using python with Tensorflow, Keras and opencv2 libraries. Several parameters were changed and a total of five models were tested on training and test road survey images. The results were promising, and further studies need to develop more accurate model to get conclusive results. This study developed the framework of the entire process.

# DEDICATION

I want to first dedicate this thesis to Jyothi and Sanjay Goud my parents with whose unconditional love and support made it possible for me to complete my Master's thesis. They always guided me through the challenges I faced in my academic career. My mom inspired me to continually improve myself and always produce the best results that I could. Also, I would like to wholeheartedly thank Nayonika my sister for her support.

In addition, I would like to thank Deoraj Asane, Shubhanker Shindgikar Yash Tondare, Kunal Patil and Sumedh Yeolekar for supporting me through my Bachelor's degree, without you I would not have been in this position to finish my Master's degree. Also, I would like to thank Bhavana Reddy and Uttara Krishnan for their constants words of encouragement throughout my academic career. All of you have contributed in some way to help me be the man I am today and I thank you for that.

# ACKNOWLEDGEMENTS

I want to give my sincere appreciation to Dr. Don Chen, my advisor for this research project. Dr. Chen helped me achieve my academic and professional goals by always being there for me when I needed him the most. I will forever be thankful and truly grateful for the opportunity to work as his research assistant. Dr. Chen taught me lessons that I can carry with me for the rest of my life.

In addition to my guide, I would like to thank the remaining of my advisor for agreeing to server on my thesis committee, Dr. Jake Smithwick and Dr. Barry Sherlock. Their support throughout my semester was extremely valuable for the completion of this thesis. It was an honor working under their guidance and would recommend them to any graduate student who would want to work with them in the future.

TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1 INTRODUCTION

The purpose of this research is to investigate methods to create a training dataset of pavement distresses for application on deep learning algorithms and use the training dataset to train a selected deep learning algorithm. Pavement Management System (PMS) is a means of maintaining the road network. It consists of three aspects: collecting data on roadway to assess its condition; adopting a maintenance strategy based on the data collected; and improving maintenance and data collection in order to make the PMS more efficient (Han and Kobayashi 2013). Each state faces their unique challenges in terms of weather, traffic and population, therefore the PMS's for all state transportation agencies differ. The challenges arise for state DOT's in terms of saving costs and or time in one or all three aspects mentioned above. Therefore, DOT's are constantly seeking processes to carry out tasks for saving time and cost in road maintenance. This study intends to assist the DOT's to save cost and time with their pavement management system using deep learning.

Government agencies such as state department of transport (DOT's) or Federal highway Administration (FHWA), measures the road conditions in terms of pavement performance. There are several performance indicators to determine the condition of road network, the most commonly used indicators amongst government agencies are: Pavement Condition Index (PCI), Pavement Condition Rating (PCR), International Roughness Index (IRI) and Surface Distress Index (SDI) (Siswoyo and Setyawan 2017).

North Carolina Department of Transportation (NCDOT) uses a Pavement Condition Rating to measure the performance of its roads. This rating requires accurately measuring various

distress types on roadways. Some common road distress types include alligator cracking, longitudinal cracking, transverse cracking, patching, rutting, and pot holes.

Historically, a windshield survey was conducted to manually assess each segment of road to determine which type of distresses existed on the roadway. Windshield surveys generally consist of manually surveying each mile of roadway with a technician memorizing each different distress type and estimating its severity. This leads to many errors due to the difficulty of the survey. Also, this was time consuming and unsafe, as surveyors would have to sometimes drive on shoulder of interstate highways at low vehicle speeds.

Therefore, recently government agencies have shifted to semi-automated survey methods by mounting cameras to survey vehicles (NCDOT 2011). These survey vehicles do not need to be stopped as the camera continuously takes images of the road. The images are stored and later processed to determine distresses type and severity. Currently, North Carolina DOT utilizes a third-party vendor for survey of roads and processing of images. The vendor has developed proprietary software to process the images after surveying in a semi-automated manner. This improves time and costs for surveying roads. However, there are discrepancies with the vendor's results and the actual pavement performance rating.

In the past when processing images used computer vision applications, the images were processed with some filters applied to the image to make the distress more visible to the technician (Sun, Salari et al. 2009). This process was time consuming as the survey vehicle produced a large quantity of images even within a small survey section. To overcome this, various techniques such as filter based, segmentation and thresholding were utilized to process the images (Chambon, Subirats et al. 2009). This process was still time consuming,

and not accurate, but more efficient than manually checking each image for cracks (Oliveira and Correia 2013).

Along with advancing technologies and increase in computational abilities, new methods such as deep learning and machine learning have been applied for image recognition. Machine learning can be described as an algorithm that predicts outcomes based on previously learned data, also known as training data (Mohri, Rostamizadeh et al. 2018). Machine learning is being utilized across all fields when large quantity of data is available. For example, Amazon uses machine learning algorithms to predict what products a consumer might be interested in buying based on search history and previously bought products (Jin, Ho et al. 2009). Even though machine learning proved accurate in predicting results from previously learned data, it was not good in handling large input of datasets (Bengio, 2012). Therefore, deep learning was developed as an evolution from machine learning, to handle more complex datasets. Today, deep learning algorithms can classify multiple forms of complex data to make predictions. Some examples of complex data are: speech recognition, natural language processing, language translation and image recognition (LeCun, Bengio et al. 2015).

In machine learning and deep learning, computer recognizes and stores an image as a matrix of pixel values, consequently each corresponding color on the image has a numerical value associated with it. This matrix of pixel values is converted into an array to be run by deep learning algorithms for pattern recognition in the pixel values. With sufficient number of images, the algorithm learns patterns in the data and can make new predictions on previously unseen data. For example, the algorithm can make predictions

on images of hand-written digits, or it can recognize various distresses on road survey images.

Deep learning algorithms require large quantity of labelled data set, which mark the object on the image. For example, marking and categorizing of various types of distresses such as longitudinal, transverse or alligator crack on road survey images. This laborious process is an essential part of applying deep learning algorithm for image recognition. Currently, there are no publicly available datasets for developing deep learning algorithms to detect different types of distresses on a roadway surface image. CrackNet is a deep learning algorithm that utilizes a manually created dataset by a team of 10 over one year marking 3000 images to near pixel perfect accuracy (Zhang, Wang et al. 2018). However, the labelling method was not provided hence, there is a need to describe methods that can quickly and accurately mark these images to develop deep learning algorithms.

Apparently there is a knowledge gap in terms of creating a training data set of pavement distress images which this study aims to address. This research's contribution is to find ideal labelling methods that can quickly and efficiently mark the images as, deep learning algorithms require large amounts of labelled images for the models to perform accurately. In addition, to utilize the training dataset created in this study for development a robust deep learning algorithm.

## 1.1 Organization of Thesis

This Thesis has been organized into six chapters. Chapter 1 contains the introduction describing the challenges faced by State DOT's. Chapter 2 encompasses the literature review whereas, Chapter 3 includes the research methodology and research objective.

Chapter 4 presents the expected results and discussion and Chapter 5 comprises of the conclusion and recommendations. Appendix I contains the Mask-RCNN Libraries. Appendix II contains the Code of Model. Whereas, Appendix III comprises of the Code of Model Training. Appendix IV encompasses the List of Library Requirements for the Python Mask-RCNN Environment. Appendix V includes the Test Results Without outliers 50E_640RES vs. 50E_960RES. Appendix VI presents Test Results Without outliers 50E_960RES vs. 75E_960RES. Appendix VII contains Test Results Without outliers 50E_640RES Vs. 100E_1024RES. Appendix VII displays Test Results with outliers

# CHAPTER 2 LITERATURE REVIEW

## 2.1 Pavement Management System

AASHTO defines pavement management (PM) as; “the effective and efficient directing of the various activities involved in providing and sustaining pavements in condition acceptable to the travelling public at the least life cycle cost” (AASHTO 1985). According to the Peterson, Pavement Management System (PMS) consists of the five essential elements (Peterson 1987).

1. Pavement surveys related to condition
2. Data base containing the pavement information
3. Analysis method of collected data
4. Decision criteria
5. Implementation

The objective of Pavement Management is to attain the best value of publicly available funds, while providing a safe, comfortable and economic transportation system. To achieve this, the five elements of PMS must be economically feasible. Pavement surveys is the most time consuming and expensive aspect of (PMS). Hence, state agencies are always trying to improve this process.

In the past, surveys were conducted as a road test, where a panel of engineers where selected to ride over specific pavements to judge it on a wide range of conditions. The road sections would be rated on a scale from 1 to 5, very poor being the least and very good being the most. The engineer’s main guideline was whether the road was acceptable or

unacceptable to drive over a long period of time or distance (Finn 1998). This process was time consuming, not accurate, and expensive, therefore, better methods were adapted.

Windshield survey was utilized by state governments with a more robust system. Survey was conducted by trained personnel travelling on speed of 15-20 mph on interstates and major roads with full width shoulders. The raters were provided printouts with beginning and ending of each segments and various information regarding traffic. In addition, several distress types that occurred on the roads were identified and measured in terms of severity (Corley-Lay, Jadoun et al. 2010). The data collection was dependent on the raters and due to this, it had variability.

Due to the limitations of the windshield survey, there was a need for automated high-speed survey that would provide accurate survey results. High speed image collection was available for many years, it is a faster and more economical form of survey compared to the windshield survey. Therefore, NCDOT adopted high speed image collection beginning in 2011 (Mastin 2011). This involves survey vehicles to drive vehicles mounted with cameras and take images while driving. The survey covers the entire width of the lane therefore, no section is missed. The images will be utilized in automated or semi-automated data processing.

## 2.2 Types of Distresses

Pavements encompass many different types of distresses, however, only the distresses that the NCDOT utilizes to calculate a pavement condition rating have been presented. According to the NCDOT 2010 manual (NCDOT 2010), several distresses and there their severity are defined below.

Transverse Cracking

Transverse cracks run perpendicular to the pavement centerline but not over joints in an underlying jointed concrete pavement. It consists of three severity levels (NCDOT 2010):

a) Low Severity

   A sealed crack in a good condition such that the crack width cannot be determined or an unsealed crack with a width less than 0.25 inch.

b) Moderate Severity

   An open, unsealed crack between 0.25 inch and 0.5 inch in width or any crack sealed or unsealed with adjacent transverse cracking within 5-10 feet.

c) High Severity

   An open unsealed crack greater than 0.5 inch in width or any crack sealed or unsealed within 5 feet. Frequent blocking is likely present with areas between 1 and 10 square feet.

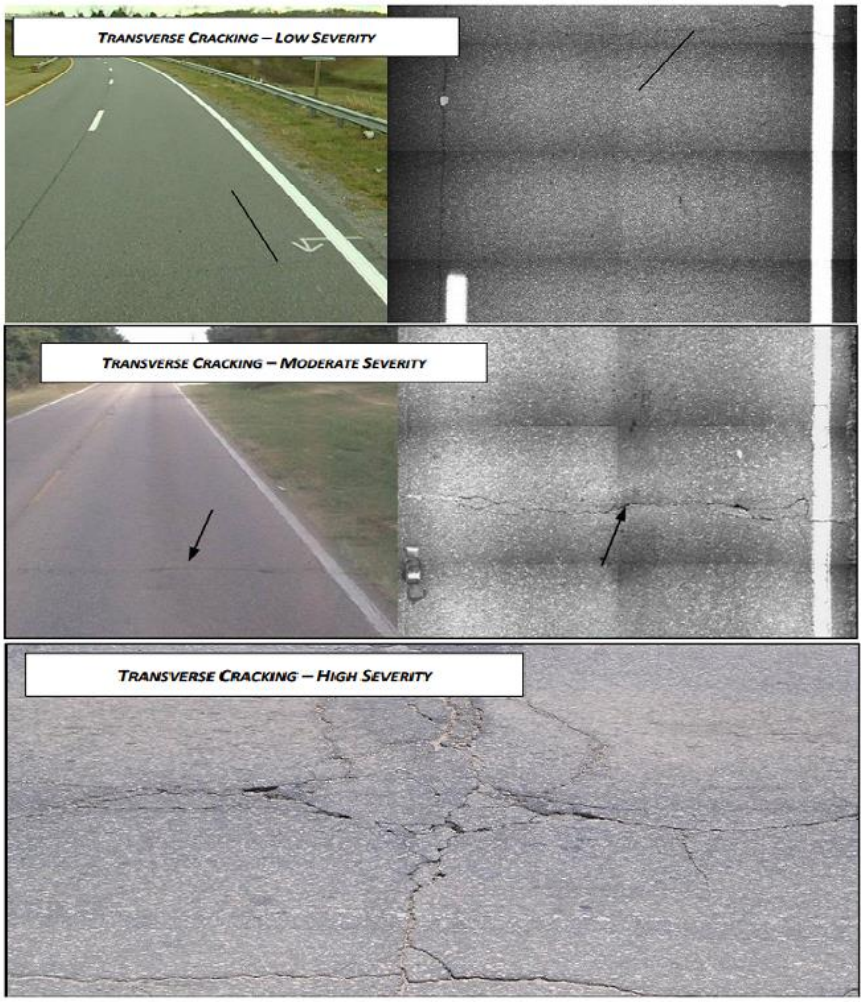



*FIGURE 1: Examples of Transverse Cracking. (Courtesy of NCDOT 2010)*

Longitudinal Cracking – Outside of the Wheel Paths

Longitudinal cracks are those running parallel to the centerline of the roadway, but not over joints in an underlying concrete pavement. However, for rating purposes, only longitudinal cracks outside the wheel paths are counted as longitudinal. It consists of two severity levels (NCDOT 2010):

a. Low Severity

A crack with the sealant in good condition such that the crack width cannot be estimated or a closed crack with width less than 0.25 inch.

b. High Severity

An open unsealed crack or any crack sealed or unsealed with adjacent random cracking.

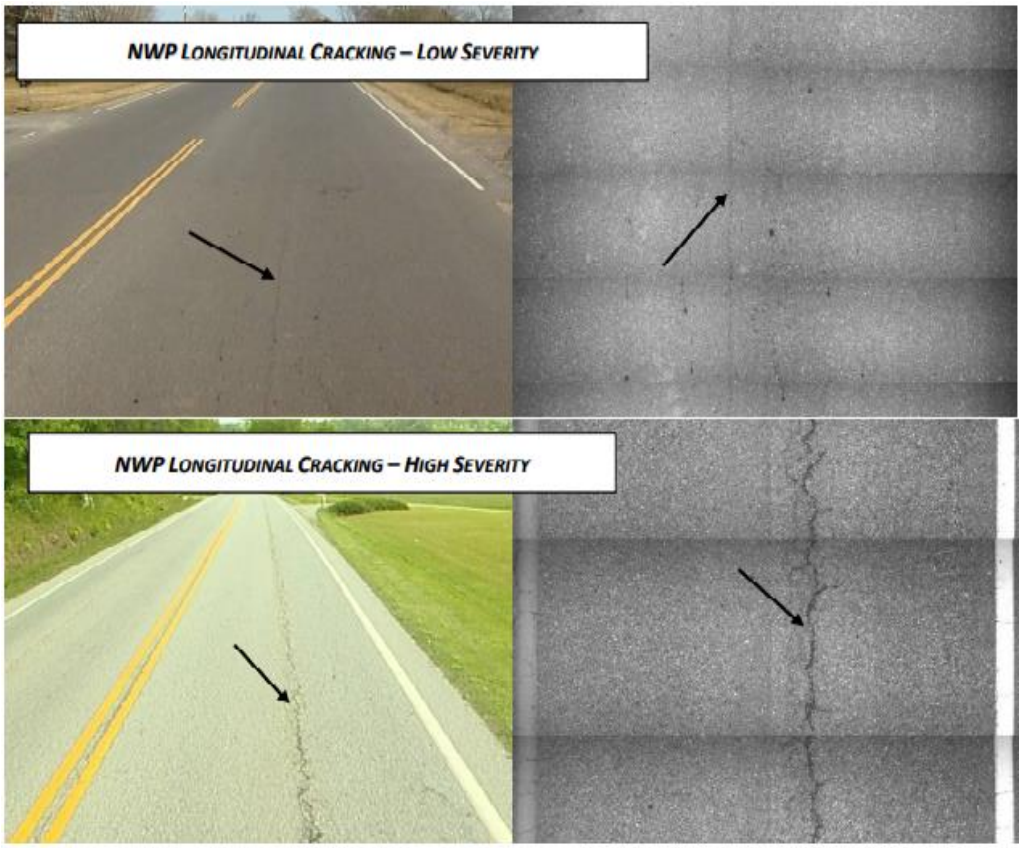



*FIGURE 2: Examples of NWP Longitudinal Cracking. (Courtesy of NCDOT 2010)*

Longitudinal Lane Joint Cracking

NCDOT only classifies lane joint cracking as a distress when the joint has cracked, and it will allow water to penetrate the joint. It consists of two severity levels (NCDOT 2010):

a. Low Severity

A longitudinal paving joint with the sealant in good condition such that the width cannot be estimated or an open unsealed joint.

b. High Severity

The longitudinal paving joint must be cracked with severe spalling or adjacent random cracking to be classified as high severity.

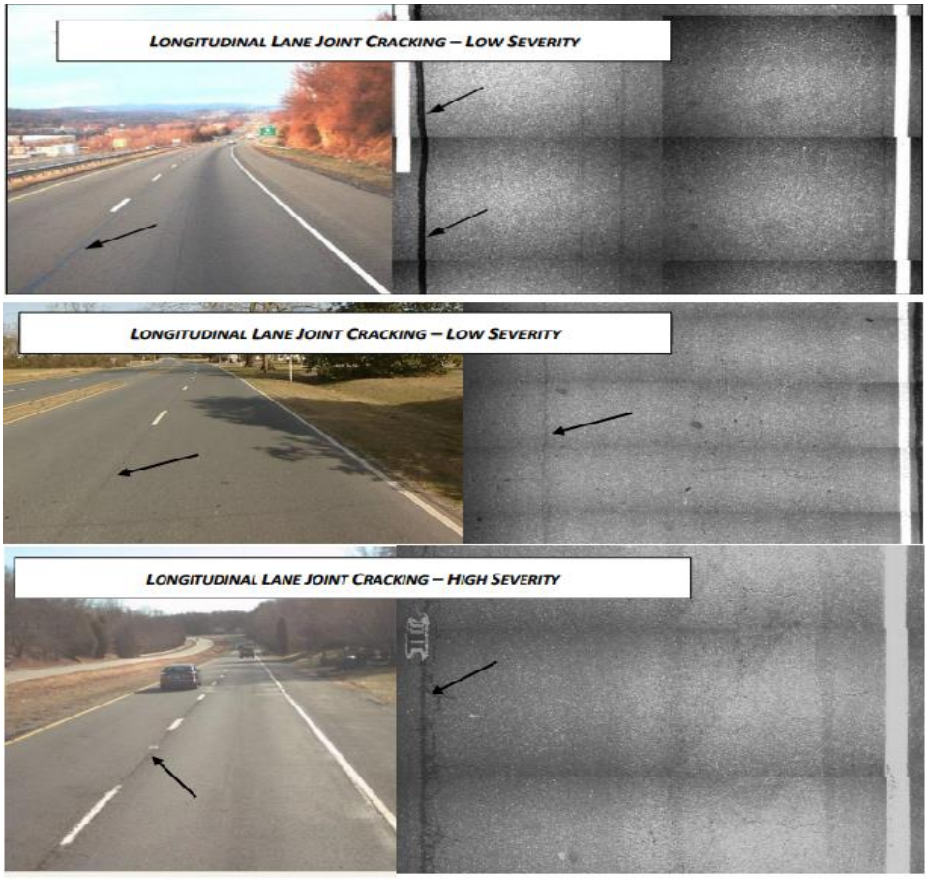



*FIGURE 3: Examples of Longitudinal Lane Joint Cracking. (Courtesy of NCDOT 2010)*

Alligator or Fatigue Cracking

Alligator cracking occurs in areas due to heavy repetitive wheel, NCDOT considers the alligator cracking that are found in wheel path for rating. Typically, alligator cracks begin as longitudinal cracks and develop as more interconnected cracks over time. These

interconnected cracks have pattern similar to chicken wire or alligator hide, hence the term alligator cracking. It consists of three severity levels (NCDOT 2010):

a) Low Severity

A single sealed or unsealed longitudinal crack in the wheel path or an area of cracks with no or few interconnecting cracks with no spalling. Cracks are less than 0.125 inch in width.

b) Moderate Severity

An area of interconnecting cracks forming the characteristic alligator pattern; may have slight spalling. Cracks are typically about 0.25 inch in width

c) High Severity

An area of moderately or severely spalled cracks forming the characteristic alligator pattern. Cracks are typically greater than 0.375 inch in width

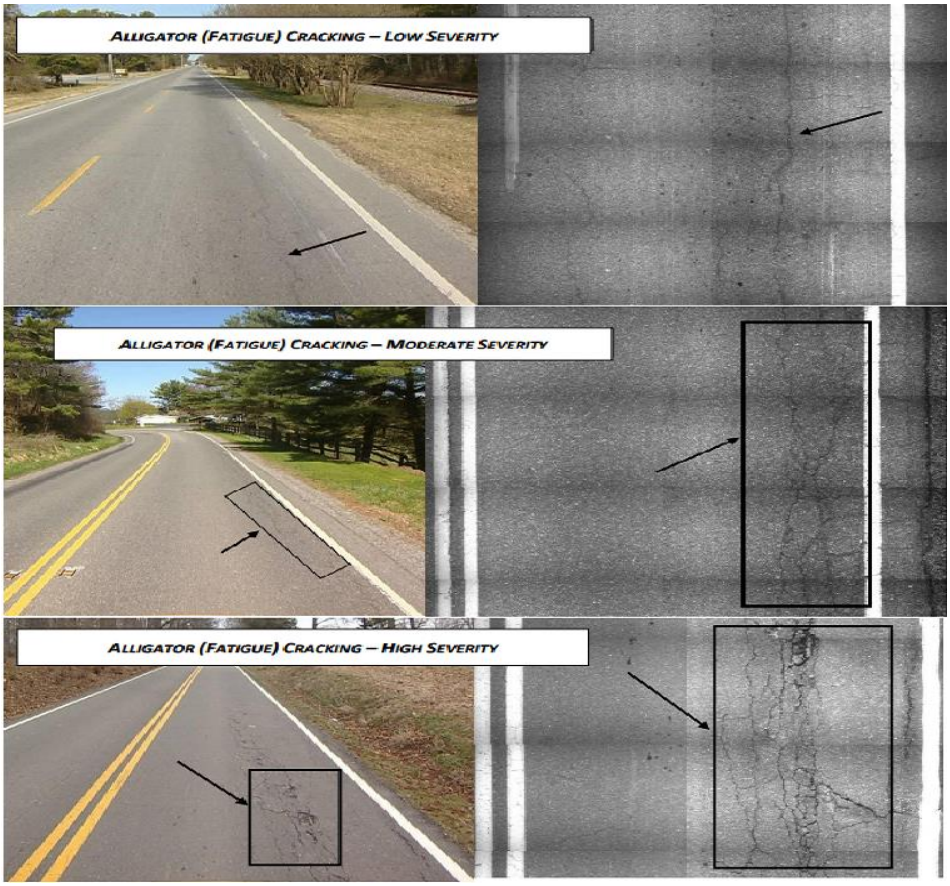



*FIGURE 4: Examples of Alligator Cracking. (Courtesy of NCDOT 2010)*

## Patching

Patches are areas of the pavement surface that have been removed and replaced or where additional material has been placed on the pavement surface to cover cracking or other distress. There are two types of Patching: Wheel path and non-Wheel Path. However, Patching does not have any severity levels. It is only calculated in terms of amount and which is measured in square feet (NCDOT 2010).

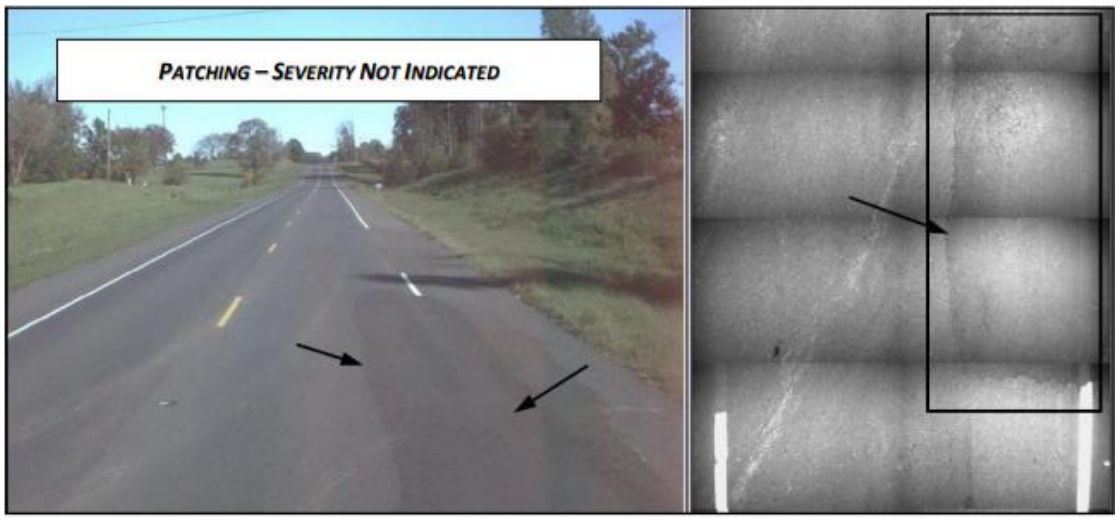



*FIGURE 5: Example of Patching. (Courtesy of NCDOT 2010)*

## 2.3 Pavement Performance

Most State agencies utilize the individual pavement distress data differently, to designate the performance of a roadway. Some performance indicators that state DOT's utilize are International roughness index (IRI), Pavement Condition Rating (PCR) and Pavement Condition Index (PCI) (Siswoyo and Setyawan 2017).

IRI is measured by a road profiler, by which a series of numbers represent the profile of the road way. Road roughness meters or profilers mounted on survey vehicles are utilized to collect data for IRI calculation. The device measures and records displacement of chassis relative to the rear axle per unit distance travelled usually miles (Shafizadeh, Mannering et al. 2002). Washington D.C. Department of transportation measures the IRI of its roads in terms of inches/mile (Arhin and Noel 2014).

North Carolina Department of Transportation (NCDOT) utilizes a Pavement Condition Rating (PCR) to describe the condition of its roadways. Initially, with this approach, each road way is given a score of 100 and each distress is multiplied with a certain weight and

subtracted from 100. A road section's PCR decreases with respect to time and after a certain threshold the section undergoes rehabilitation process (Chen, Cavalline et al. 2014). Fast processing of distress identification would help NCDOT calculate the performance indicator faster.

## 2.4 Data Pre-Processing Techniques

Previously, methods such as filtering, segmentation, wavelet transforms were utilized to detect cracks on images (Oliveira and Correia 2013). Filtering can be described as a method to enhance the contrast in the image to distinguish between pavement distress and image background. Filtering technique is not effective, as it fails to enhance the cracks which are non-continuous or those cracks with width less than 3mm (Sun, Salari et al. 2009).

Although at the time these techniques were utilized to process images, they were eventually deemed time consuming and for the most part inaccurate. These techniques failed to detect crack that were not continuous at a pixel level and the techniques were not accurate at edge detection of the cracks. In essence, these methods could not properly distinguish the pavement distress from the image background. For these reasons, other techniques had to be developed for pavement distress identification in images that could automatically detect road cracks, save time and money.

## 2.5 Machine Learning

Machine learning is defined as, "a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" (Robert 2014). The aim of machine learning is to

allow the computers to learn patterns automatically without human intervention in order to make future predictions. Machine learning algorithms are categorized in two parts: Supervised and Unsupervised.

Supervised Learning is technique in which the computer can apply what is learned in the past to new data by utilizing labeled examples. The labelled examples are known as a training dataset, from which the computer programs learns. Unsupervised learning is utilized when classified or labelled data is not available, the computer program infers patterns through the data set to and finds hidden structure in unlabeled data set.

Machine learning has been utilized in all sectors from farming, supply chain management and cancer detection in medical imagery (LeCun, Bengio et al. 2015). Some of the Machine learning methods that were utilized in cancer detection was: Support vector machine (SVM); Bayesian navies BN, Decision Tree (DT) and Artificial Neural Networks (ANN) (Kourou, Exarchos et al. 2015). From the above mentioned methods artificial neural networks displays the most promise in image recognition and classification.

Artificial neural networks (ANN) models are inspired from the human brain. The ANN model consists of many nodes and synapses which communicate through connecting synapses, similar to the human brain. ANN consists of multiple layers: an input layer where input parameters are processed, hidden layers where the hidden parameters are processed and an output layer to display the results (Kalogirou 2000). Each layer has certain number of neurons and each neuron is a processing unit with a weight associated to it. The process of ANN starts from the summation of input layer by weighted activation of each of the neurons. To summarize, input data is passed through input layer and hidden layer by means of multiplying weights of the neuron and summation of the layer. There are various

algorithms available to train an ANN and selecting the correct one for a particular problem is only through trial and error basis (Ata 2015).

## 2.6 Deep Learning

Deep learning is a subset of machine learning, where the aim in deep learning is to determine more abstract features in higher levels of representation (Bengio 2012). Deep learning neural networks consists of more number of hidden layers with more number of neurons; these are known as deep neural networks (Nguyen, Dosovitskiy et al. 2016). Conventional machine learning was inefficient in processing large amount of data such as speech recognition, or pixel values of high definition images. Deep learning can overcome this drawback by allowing the computer to be fed raw data and automatically discover patterns for detection or classification. For example, an image is represented in the form of an array of pixel values, which is the input layer into the neural network. The first hidden layer typically discovers the presence or absence of edges of objects in the image. The second layer generally discovers a pattern in particular arrangement near the edge of the object. Third layer arranges the patterns into larger combinations to detect familiar objects. This process of discovering key patterns or arranging them in a particular manner is not designed by human engineers, the deep learning algorithm performs this task on its own, and this is the reason it has become a powerful tool in all fields (LeCun, Bengio et al. 2015).

However, deep learning models require large amounts of labelled data set in order to predict highly accurate results. For example, MNIST dataset was one of the first data sets that was created to experiment with deep learning algorithms. It contained 60,000 labelled training images of hand written digits in black and white with an image size of 28 x 28 pixels (Deng 2012). Some deep learning models were able to recognize the digits with up

to 98.13% accuracy. Another example is IBM's facial recognition data set which contains one million labelled images, and their deep learning model has an accuracy of 99.6% in identifying light skinned male subjects, but a 65.3% accuracy in identifying dark skinned female subjects. Despite its advances in image recognition deep learning is not completely accurate in all types of images and it requires large amount of labelled images to predict with high accuracy.

Even though deep learning algorithms have high rate of identifying images accurately in other fields, it is challenging to apply them to pavement distress detection as they require pixel perfect accuracy (Zhang, Wang et al. 2018). The difficulty arises from unavailability of accurately labelled data set to train the deep learning algorithm. One of the first datasets created for deep learning application on pavement images was the Cracktree data set created at Temple University. It contained 500 images of size 99 x 99 pixels that was manually annotated (Zhang, Yang et al. 2016). These images were small in terms of size therefore the cracks on road surface were easily identified as each pixel contained more information of the road crack and road surface. However, this method is not feasible for an automated pavement survey system as the images were taken from a phone. Cracks exist in other civil infrastructure projects as well and need to be identified, in 2017, a study utilized DSLR camera for data collection of 500 images and manually labelled these images (Cha, Choi et al. 2017). Some recent studies for deep neural network applications involved creating two data sets of images containing 2,000 and 3,000 images and each named CrackNet and CrackNet II. These images were manually marked by multiple teams over the period of one year, and the algorithms showed accuracy of 90.13% to 87.63%.

Also, these images were 3D surface images (Zhang, Wang et al. 2017, Zhang, Wang et al. 2018).

### 2.6.1 Mask –RCNN

Deep learning involves many algorithms with applications which have various advantages and disadvantages. These have developed overtime and evolved to more task specific models. One of the first deep learning models was the convolutional neural network (CNN), which was utilized in image classification. Image classification involves classifying what is present on the image into predefined classes. For example, whether the image has a dog or cat, or whether the image has a crack present in it or not. CNN's were utilized in previous studies to classify images with cracks on road survey images (Zhang, 2018). However, the limitation of this model is that it only classifies weather a crack is present on the image or not. Even if it is one single crack or multiple cracks on the same image.

There is more information that is required from a survey to evaluate the performance of the road section than classifying if that section has a crack. For evaluation of pavement performance, the type of crack, the location of the crack, and the width which defines the severity of the crack must extracted from the image. Region Convolutional Neural Network (RCNN) was developed from a CNN, which utilizes region proposals along with CNN's to detect object on an image. The RCNN model detects objects within certain regions and classifies the object within that region (Girshick, 2014). For example, if an image has multiple cracks present on it, the RCNN model detects each crack within a bounding box, in addition, it classifies the crack according to the type of crack.

Mask-RCNN was developed by the Facebook AI Research (FAIR) Team as an evolution from RCNN and CNN. The goal of the Mask-RCNN model is semantic segmentation, which is to classify each pixel on an image into a fixed category. It was developed using the Common Objects in Context (COCO) dataset. Which is a large-scale object detection, segmentation, and labeling dataset developed with collaboration from Microsoft and Facebook (Lin, Maire et al. 2014). The model detects objects within a bounding box and utilizes a mask to cover that object within the bounding box. The goal of this study is to apply this principal in detecting cracks on road survey images and mask the cracks within the bounding box. Further studies can use this to determine the severity level of the cracks by measuring the width of the masks. The Figure 6 describes the framework of the Mask-RCNN model (He, 2017).

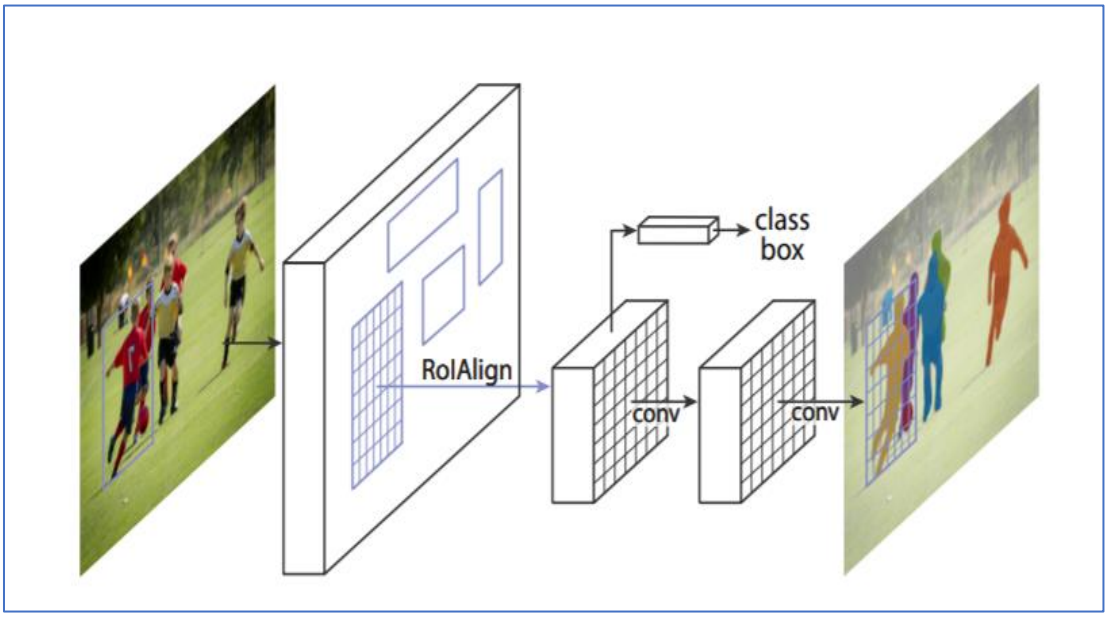



*FIGURE 6: Framework of the Mask-RCNN Model (He, 2017).*

## 2.7 Image Labelling Software

Image annotation is the most arduous and necessary task of applying deep learning for image recognition. It consists of manually selecting each pixel of an object in an image and masking it with a certain color to label the object type present in the image. Most images which have been utilized in deep learning algorithms were either colored images with clearly defined edges of object, or binary images with the object being one color. Due to these reasons, there exist commercial software that can easily annotate images and build data sets. However, in case of road survey images, the cracks are not always easily visible even to the human eye and the boundaries of the cracks are not always clearly defined. Therefore, it is difficult to distinguish between the pixels which can be classified as a crack or non-crack. To the best of the knowledge of this research, there is no software or methods available that describes how a road survey image can be labeled. Therefore, it is the intent of this research to fill that research gap.

This research has identified some tools which can be utilized to annotate the road survey images. Some of the tools are free software available for download on the internet and others are commercial software. The following is a description of the various software and its tools utilized in this study

1) GIMP (GIMP 04/07/2019)

GIMP stands for GNU Image Manipulation Program, it is a free software available for download on the internet. It assists users with image retouching, composition and authoring. This software is proposed as a technique to mark road survey images. The software includes a wand tool, which can be utilized in marking multiple pixels of similar value by selecting one pixel. This can be effective to select a pixel on the image where the

crack exists, and the software will automatically select similar pixels within a threshold decided by the user.

2) Adobe Photoshop Lightroom Creative Cloud (Lightroom 02/05/2019)

Lightroom is an image manipulation software distributed by Adobe which is useful in image manipulation and image organization in large quantities. This can be utilized in organizing multiple images with same type of distress efficiently. In addition, Lightroom has image editing capabilities which were tested to mark the distress on the image in this research.

3) Adobe Photoshop Creative Cloud (Adobe 10/12/2019)

Photoshop is another image manipulation software distributed by Adobe, and it has all the capabilities of Lightroom. In addition, it has tools similar to that of GIMP.

i) Magic wand

In Photoshop, Magic wand is a tool similar to the wand tool in GIMP, where the user can select multiple pixels of similar value by clicking on one pixel. The tool selects all pixels in range that is decided by the user. Also, the user has the choice of selecting the pixels that are continuously connected to each other or separated by different value pixels. Since cracks are continuous this study proposes to select all continuous pixels.

ii) Lasso

Within Photoshop there is another tool known as the lasso tool. It allows the user to make a freehand selection around an object on the image, similar to outlining something on a paper with a pen. Also, there are multiple options within the lasso tool which can utilized

in selecting cracks on the image. For example, there are button which will allow the user to add, subtract, or intersect with an existing selection which can be convenient in marking the cracks.

4) Whitebox (Lindsay 1/8/2019)

Whitebox is a software for geospatial analysis and data visualization which was developed for the purposes of terrain analysis. It is a novel approach that was tested to treat the pavement surface as a terrain analysis model. WhiteBox is a software generally utilized in GIS and mapping studies, it helps the users detect valleys and measure lake depths using satellite images. The study proposes to apply this technique to detect cracks in the road survey images, as the road surface can be compared to a mountain range with the cracks being compared to valleys on a smaller scale.

# CHAPTER 3 RESEARCH METHODOLOGY

The methodology for this research comprises of two aspects: identifying the ideal labelling method and developing a robust deep learning algorithm to detect roadway distresses. To accomplish these tasks, raw images need to be obtained from the NCDOT vendor to be utilized for this research. This study proposes to utilize 20 images with multiple cracks on each image. The next step is to determine various methods to label the data which have been discussed in detail further in this chapter. Upon the completion of this process, the images were used to train various deep learning algorithms for image recognition. This process is illustrated in Figure 7 below.

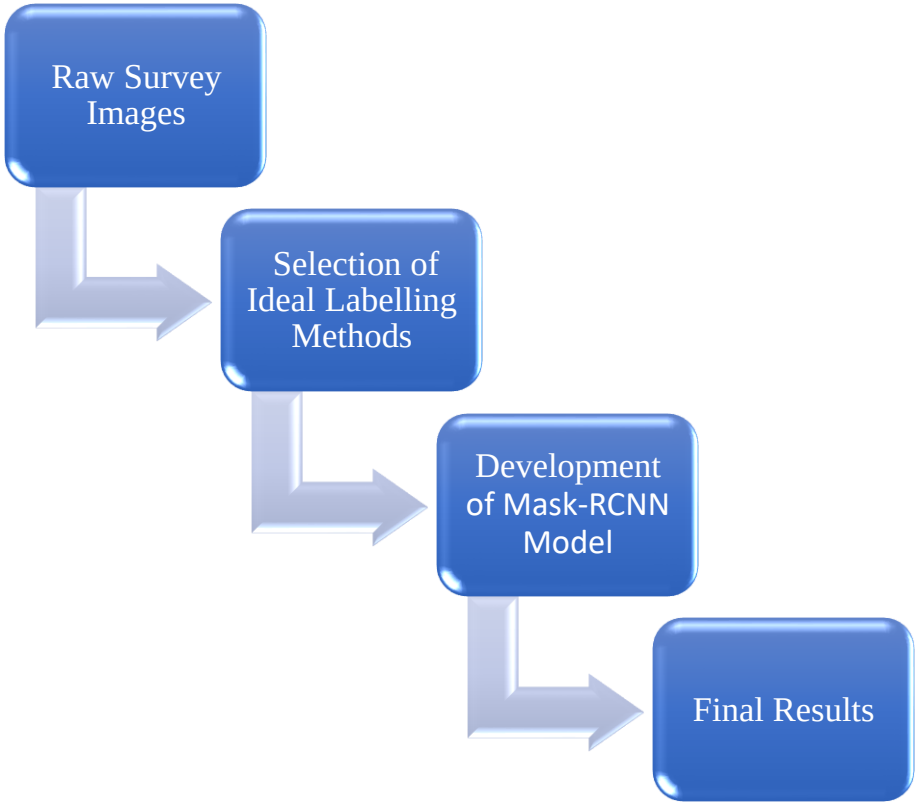



*FIGURE 7: Research Methodology Flow Chart*

## 3.1 Research Objective

The main objectives of this research are as follows:

- Selecting images which have most common surface distresses
- Identifying the most efficient technique in terms of time and ease of labeling the images
- Developing deep learning algorithms to detect distresses

One of the most common distress types in pavement is pavement cracking, and timely repairs of which, can save maintenance costs associated with further deterioration to more extreme conditions such as pot holes. The difficulty in maintaining roads arises from timely identification of deteriorated roadway section and assessing its severity. In order to evaluate the health of roadway, transportation agencies conduct road surveys through a third party. The survey consists of driving a road survey van outfitted with a camera which constantly takes images of the road as the vehicle is being driven at the highway speed. On average the vendor takes 2-3 images per second of surveying the roads and this leads to terabytes of data collected. The data collected from vendor is in the form of JPEG intensity and depth images, post survey data processing of the images is time consuming and hence, there is a need for expediting the processing time of images while maintaining low costs. As mentioned in the literature review, deep learning has been successfully utilized in order to detect distresses on image. Since deep learning require large amounts of annotated data, there is a necessity to find methods to label the images efficiently.

## 3.2 Image Labelling

The workflow illustrates the proposed process of selection of the various software required to annotate the images and the type of images selected for annotation. Four software have been identified which could assist with the annotation of the images: GIMP; Photoshop Lightroom; Photoshop; and Whitebox. The identified software was tested in this study in terms of ease of use, efficiency in marking the cracks, time, and time in generating a binary image. After identifying the best software it was utilized to annotate the distresses on the images and to generate a training dataset.

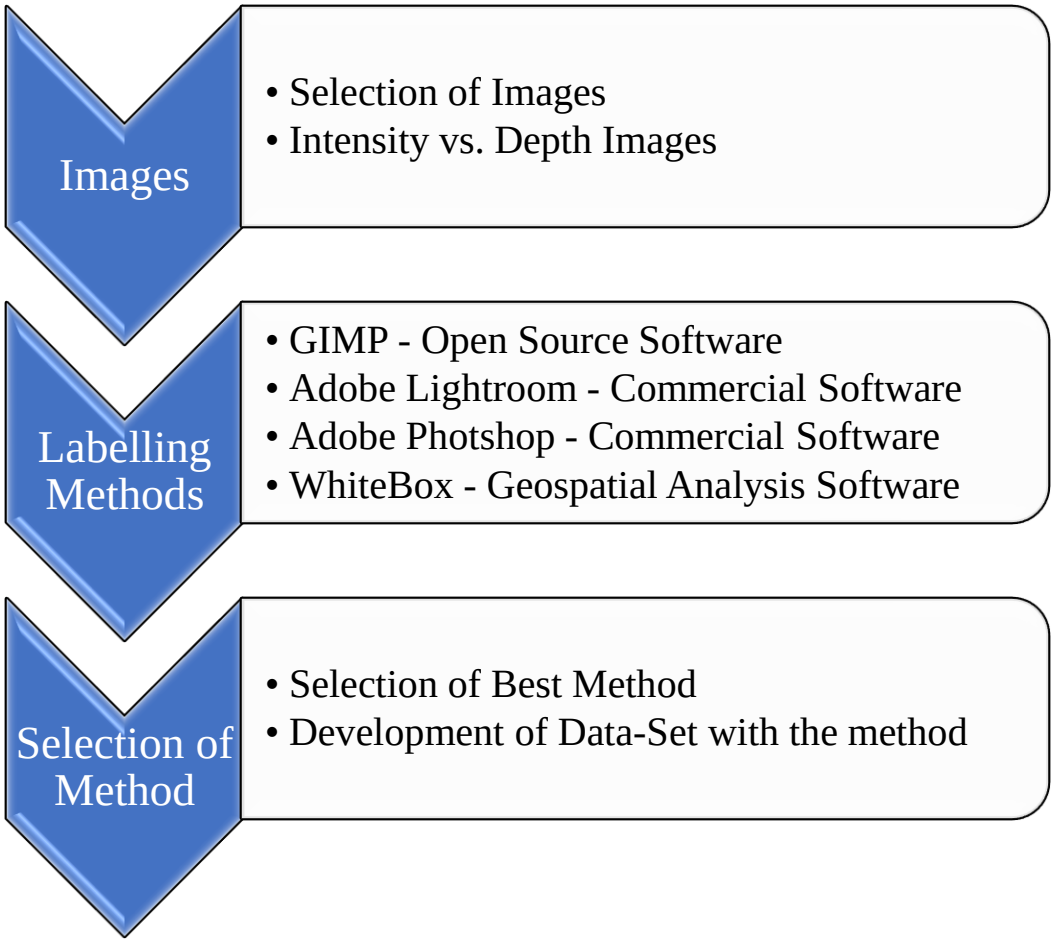



*FIGURE 8: Process of Labelling Road Survey Images*

### 3.2.1 Images

The survey images were greyscale 900 x 1800 pixel resolution, with 2 sets of each survey patch i.e. intensity and depth images. This study utilized intensity images to label and build a data set for the deep learning model. However, there are some distresses which were not clearly visible such as alligator cracking for which depth images were utilized as a reference to label the intensity images. A total of 20 images were selected at random with multiple distresses on each image. The distresses include: longitudinal, transverse, patching and alligator distress. In addition, images with manholes were also selected to avoid misclassification of manhole as a distress. A total of 149 individual distresses were labelled from the 20 selected images.

### 3.2.2 Labelling Software

One of the objectives of this study was to identify the most efficient technique in labelling the images. Therefore, the tools and their pros and cons have been listed below. This study identified four labelling methods to be tested for labelling the images, as previous studies have no enlisted tools or procedures for ideal labelling of cracks.

*GIMP*

This software is very similar to use as the Adobe Photoshop and is equipped with most of the same tools as Photoshop. However, this is an open source software and this study faced a lot of reliability issues associated with software. For example, the software often crashed while labelling and the work remained unsaved, therefore Adobe Photoshop was selected due to its reliability over GIMP.

*Adobe Lightroom*

The Lightroom software is equipped with a magic wand tool, it is utilized in selecting pixels of the same color by selecting on one pixel with a specified threshold. After which, it automatically selects the pixels of the same color within that threshold. Since it is difficult to distinguish a road crack from the surface of the road with human eye, and the pixel values being similar to that of the crack itself this tool was not helpful in annotating the cracks.

*Adobe Photoshop*

Photoshop is a more advanced version of Lightroom and contains all the tools that Lightroom does in addition to which it contains a lasso tool which can be utilized to trace the crack at a pixel level. Therefore, this study determined the lasso tool as the best option available to annotate the data set even though it is a time consuming and laborious process, it is an accurate method.

*Whitebox*

Whitebox is a geospatial analysis tool which is utilized in measurement of watershed volume, depths of valley, or topography of mountains. This study proposed a unique method of using that method in order to “build” the topography of road image so that the cracks could easily be identified and the training data could be built. Due to the evenness of the surface the WhiteBox software output contained a lot of background noise in the image therefore it was difficult to distinguish between crack and road surface. This could be pursued in the future if the background noise is reduced. The figure number 9 displays the best output from the WhiteBox software that was generated.

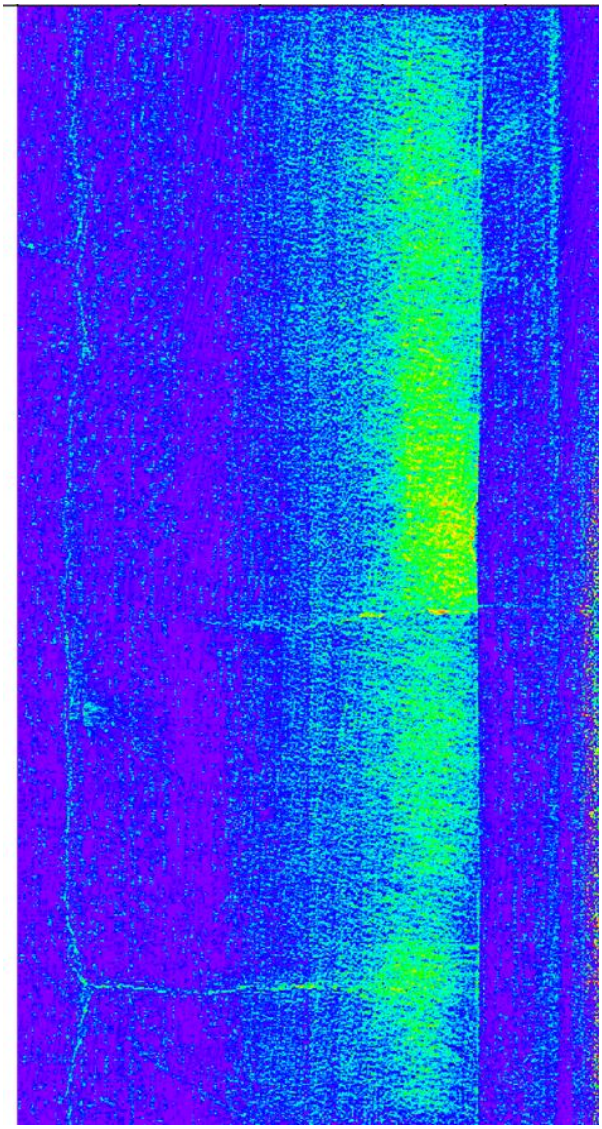


*FIGURE 9: Example output from Whitebox*

### 3.2.3 Process of Labelling the Images

After testing all the above methods, this study identified lasso tool of Adobe Photoshop to label the survey images. This section details the steps of how images were labelled

Steps for Lasso Tool:

1) Import the image into Photoshop

2) Count the number of distresses present on the image and make the same number of layers
3) Identify one distress to labelled
4) Turn off all the layers except one and the background, rename the layer
5) Select the lasso tool
6) Keep the pen size to 0 so that one pixel can be selected
7) Zoom into the identified distress till individual pixels are clearly visible
8) Start selecting the side of pixel till the entire distress is selected
9) Hold alt + backspace key to color the distress in black.
10) Repeat steps 4 through 9 till all the distresses have been colored
11) After all the layers are colored on individual layers deselect all the layers
12) While holding the ctrl key select one layer the selected layer's distress will be highlighted
13) Hold the ctrl + shift + I and the rest of the image will be highlighted
14) While the background is highlighted hold the ctrl + back arrow key and the background will become white while the distress is black
15) Right click the layer and select quick export as PNG and export the image to a folder on the computer
16) Use ctrl + z to undo till no layers are selected
17) Repeat the steps 12-16

With the above steps one image can be labelled the following images show the example of this study's above mentioned method. In essence, the annotation process creates a

mask of each individual distress on image to train the computer algorithm which pixel pattern and color comprises of a crack.

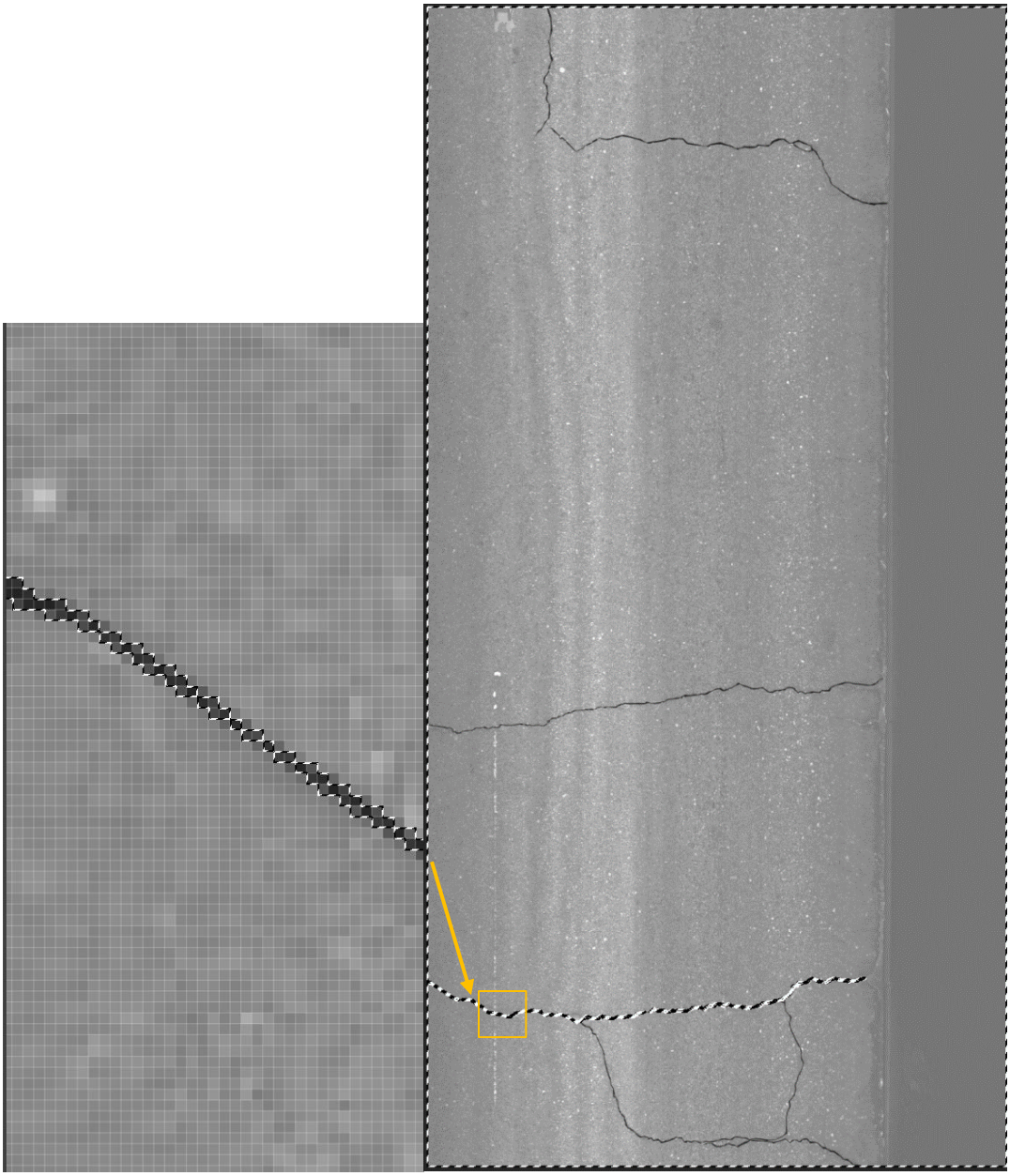


*FIGURE 10: Example of Step 8 and Step 13*

Figure 10, on the left, displays a zoomed window the crack which is visible and is traced almost pixel to pixel, this will help the deep learning algorithm recognize patterns in the

image for it to detect the cracks. This method has been utilized to annotate all the 20 images utilized further in this study for the deep learning model.

## 3.3 Data Pre-Processing

Data pre-processing consists of utilizing the exported image from Photoshop to convert it into a usable data which can be applied to the deep learning algorithm. As the deep learning algorithms requires a .json file format for data input, data pre-processing is done in order to generate a .json format of the labelled images. This is explained in detail further in this chapter and Figure 11 illustrates the process.

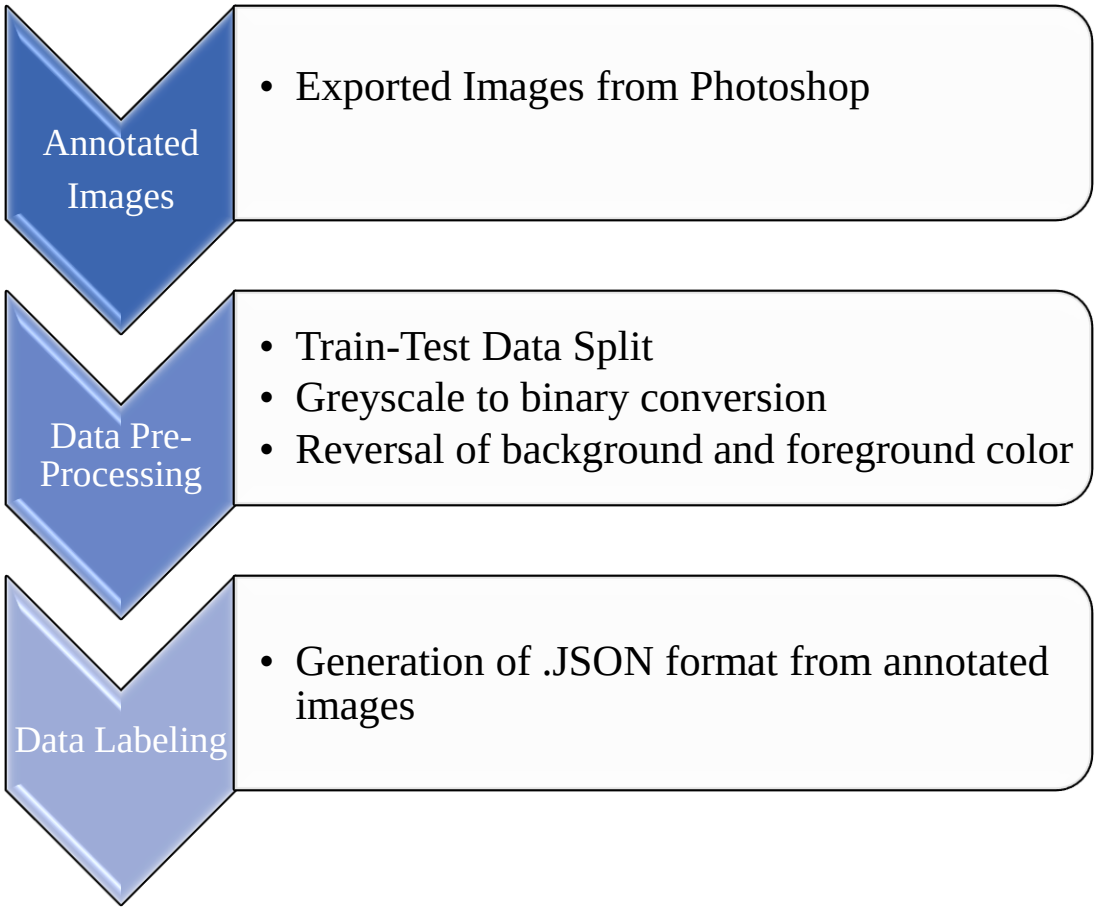



*FIGURE 11: Data Pre-Processing workflow*

### 3.3.1 Annotated Images

The images from Photoshop are exported as a .PNG format and are saved into one folder known as the annotations folder. Each roadway survey image contains multiple distress, however, each annotated image only contains one of the distress whereas all the rest are hidden. These individual distress images are utilized to train the algorithm on individual distress types. This study refers to the Coco method of labelling images. For example, the Figure 12 below shows the format required to label the images.

```
-train
 |
 └──annotations
 |     | <image_id>_<object_class_name>_<annotation_id>.png
 |     | ...
```

*FIGURE 12: Annotations Method (Courtesy of patrickwasp.com)*

In Figure 12, the <image_id> represents the image identification number given to the image at the time of survey which contains the time stamp of when and where the image was photographed. For example, 10000000223Cintensity is one of the 20 image ID's utilized in this study. <object_class_name> is utilized for all the classes present in a particular image. This study comprises of five object classes also known as the various distresses present on an image which are listed below:

- Longitudinal
- Transverse
- Patching
- Manhole
- Alligator

The <annotation_id> represents the number of same object classes present on a single image, for example, if an image contains 4 different longitudinal distresses then the annotation ID is used to distinguish between each of the longitudinal distresses.

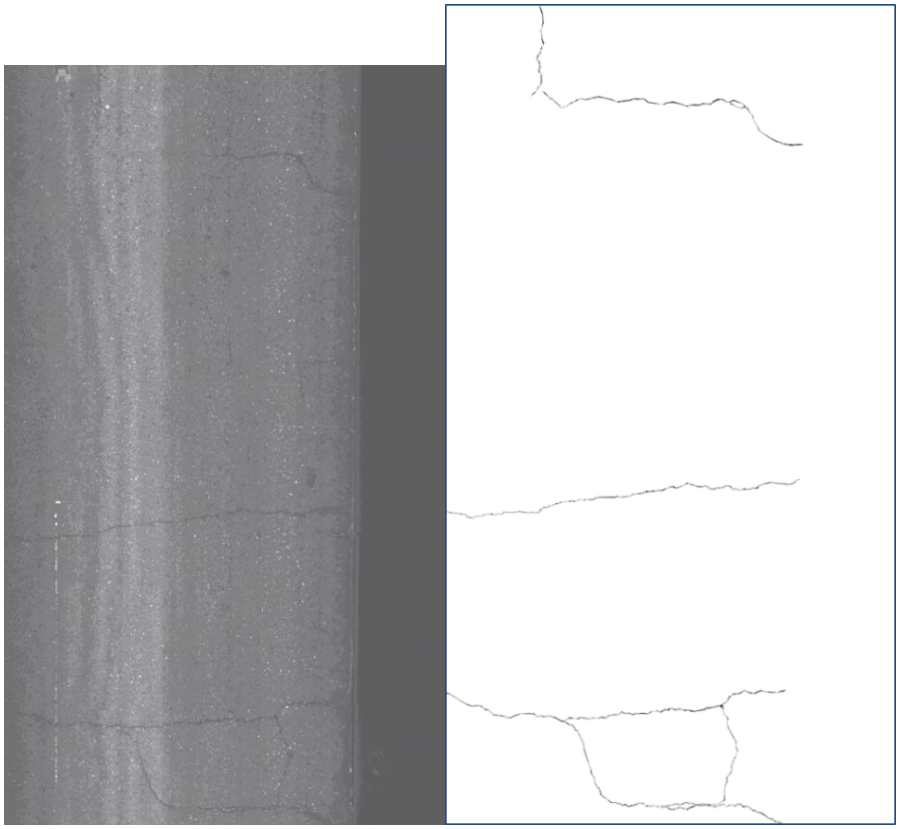


*FIGURE 13: Annotations Method Courtesy of patrickwasp.com*

Figure 13 shows the original image and the combined annotated image, whereas Figure 14 shows the individual annotated distress image. Each of the individual image is saved in the COCO format as 10000003421Cintensity_transverse_1 and 10000003421Cintensity_transverse_2 respectively.

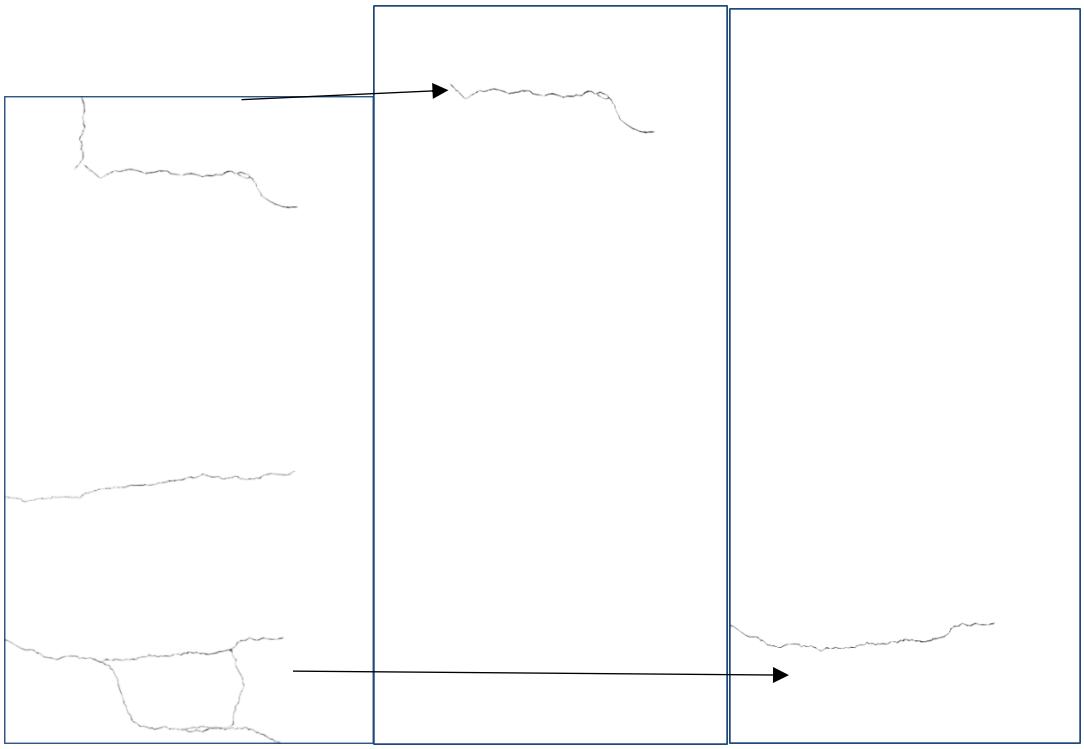


*FIGURE 14: Annotated image with individual annotations*

### 3.3.2 Data Pre-Processing

Mask-RCNN models require a .json (java script object notation) file as an input in addition to the labelled images, .json files are explained further in section 3.4 of this chapter. Generation of .json files require several steps which are enlisted below.

Data pre-processing comprised of three steps:

i) Test-Train Data Split

ii) Greyscale to binary conversion

iii) Reversal of Background and Foreground colors

Test-Train Data Split

The first step involved splitting the data set into training and testing for the deep learning algorithm. This study utilized 20 images, the training and testing was split into 15 and 5 respectively. The 15 training images comprised of 115 individual distress images, and the 5 test images comprised of 34 individual distress images. Due to small data set this study utilized training images for validation purposes.

Greyscale to Binary Conversion

The output of image from Photoshop is not a binary image, instead it is a grayscale image. Each pixel in a greyscale image have a value in the range from 0-255, 0 being complete black and 255 being complete white. This represents the intensity of the color that the pixel is displaying, this can be better explained in terms of "greyness" i.e. the intensity of the grey that exists in any given pixel. Photoshop exports the black and white images in a greyscale format. However, the Coco dataset requires the images to be binary in order to create masks for the deep learning algorithm i.e. each pixel must be 0 and 1, 0 representing black and 1 representing white.

Due to these greyscale images there was a requirement for conversion from greyscale to binary. This is can be done manually by converting each greyscale image in Photoshop to binary which would have been labor intensive and time consuming. This study utilized the openCV library in order to loop through the entire annotated directory and convert the greyscale image to a binary image and to save it in a different directory.

There are several algorithms which convert greyscale image to binary image. Through research it was concluded that Otsu's binarization is the most commonly utilized algorithm

for this purpose. The openCV library contains all the necessary functions required to execute Otsu's binarization algorithm. The two required functions are:

I) THRESH_BINARY

II) THRESH_OSTU

The Figure 15 shows the sample code required to execute this algorithm.

```
1  import cv2
2  import os
3
4  def prepare_images(path, inv=False):
5
6      for i in os.listdir(#path):
7          filepath = os.path.join(path,i)
8          # print(filepath)
9          if os.path.splitext(i)[-1]==".png":
10
11             #Read image
12             img = cv2.imread(filepath,0)
13
14             # Otsu's thresholding
15             ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
16
17             #Invert image if required
18             if inv:
19                 th2 = cv2.bitwise_not(th2)
20
21             #Save image
22             cv2.imwrite(filepath,th2)
23
24 prepare_images#("filepath")
25
```

*FIGURE 15: Sample Code of Otsu Binarization*

Reversal of Background and Foreground Color

Binary images are utilized to generate a JSON file for the M-RCNN Model, JSON files are explained in depth further in this chapter. However, there were several errors while generating a JSON file, and one of the solutions was to inverse the binary image, i.e. the background of the image must be black and the distress must be white. Line 19 in Figure 15 above shows the function which was utilized to inverse the image color before saving

the image in a new directory. Figure 16 shows the output of Photoshop and the inverse image.

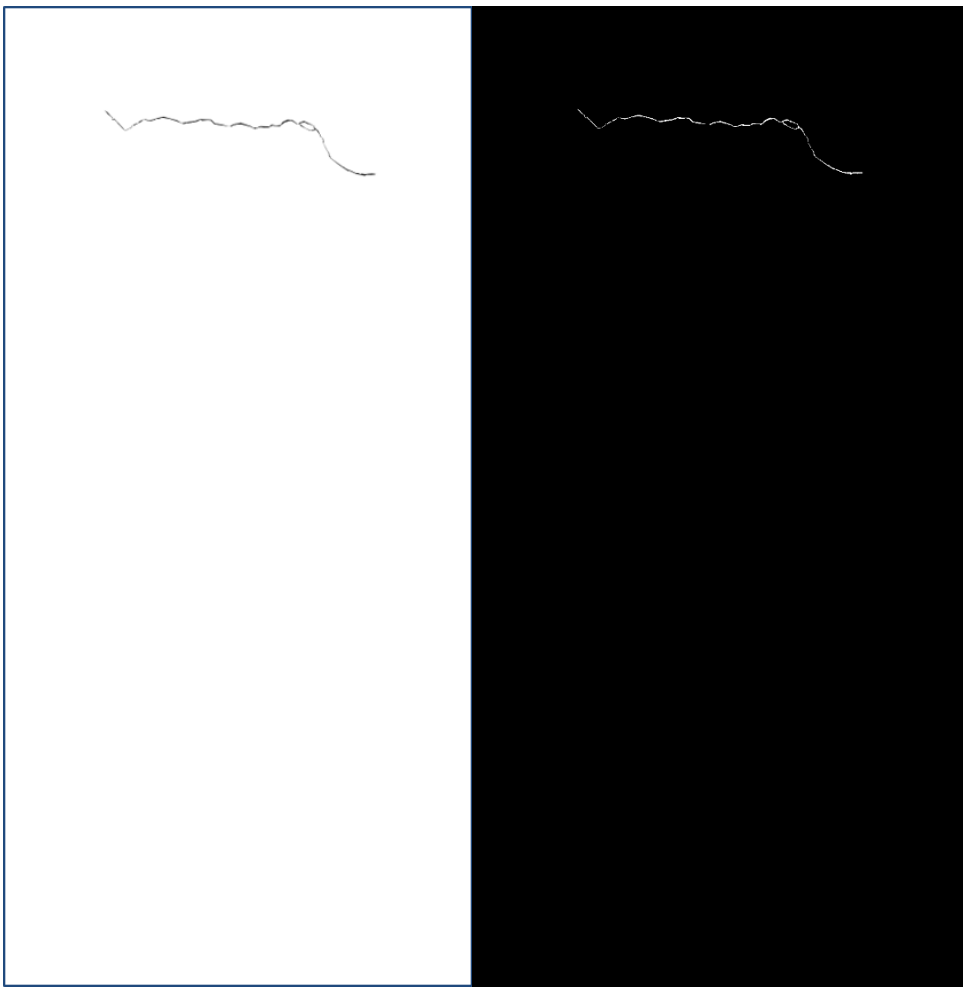


*FIGURE 16: Inverse of Images*

The conversion of greyscale to binary and inverse of image was done separately on the train images and test images, so as to make it easier to keep track of all the images.

## 3.4 Data labelling .JSON Files

JSON stands for JavaScript Object Notation and it is an integral part of Mask-RCNN (M-RCNN) algorithms. Primarily, it is a data-interchange format which is utilized across

multiple programming languages (json.org). Mask-RCNN is a deep learning algorithm which is explained in the literature review chapter. Mask-RCNN requires .json file for storage of image ID and all the image's information within one file. For example, one .json file can store information of all the images present in the directory and the information present within each image i.e. the co-ordinate information of each object/distress present in the image. In other words, .json file "tells" the computer on which image the object is located, the class of the object and the location of that object on the image.

```
INFO = {
    "description": "Example Dataset",
    "url": "https://github.com/waspinator/pycococreator",
    "version": "0.1.0",
    "year": 2018,
    "contributor": "waspinator",
    "date_created": datetime.datetime.utcnow().isoformat(' ')
}

LICENSES = [
    {
        "id": 1,
        "name": "Attribution-NonCommercial-ShareAlike License",
        "url": "http://creativecommons.org/licenses/by-nc-sa/2.0/"
    }
]

CATEGORIES = [
    {
        'id': 1,
        'name': 'square',
        'supercategory': 'shape',
    },
    {
        'id': 2,
        'name': 'circle',
        'supercategory': 'shape',
    },
    {
        'id': 3,
        'name': 'triangle',
        'supercategory': 'shape',
    },
]
```

```
CATEGORIES = [
    {
        'id': 1,
        'name': 'longitudnal',
        'supercategory': 'shape',
    },
    {
        'id': 2,
        'name': 'transverse',
        'supercategory': 'shape',
    },
    {
        'id': 3,
        'name': 'manhole',
        'supercategory': 'shape',
    },
    {
        'id': 4,
        'name': 'marking',
        'supercategory': 'shape',
    },
    {
        'id': 5,
        'name': 'alligator',
        'supercategory': 'shape',
    },
    {
        'id': 6,
        'name': 'patching',
        'supercategory': 'shape',
    },
]
```

*FIGURE 17: Example of JSON notation*

In Figure 17 on the left is an example of the different classes of figures that are utilized to generate a .json file; on the right, displays the figure classes this study utilized to generate a .json file.

## 3.5 Mask-RCNN Model

Several factors influence the application of images on the Mask-RCNN model. For example, the processing capabilities of the computer, the programming language which is utilized and the parameters of the Mask-RCNN model itself. This section details all the components which were utilized in order to develop a deep learning Mask-RCNN model. Figure 18 illustrates all the components that were required for this process.

3.5.1 Computer Properties

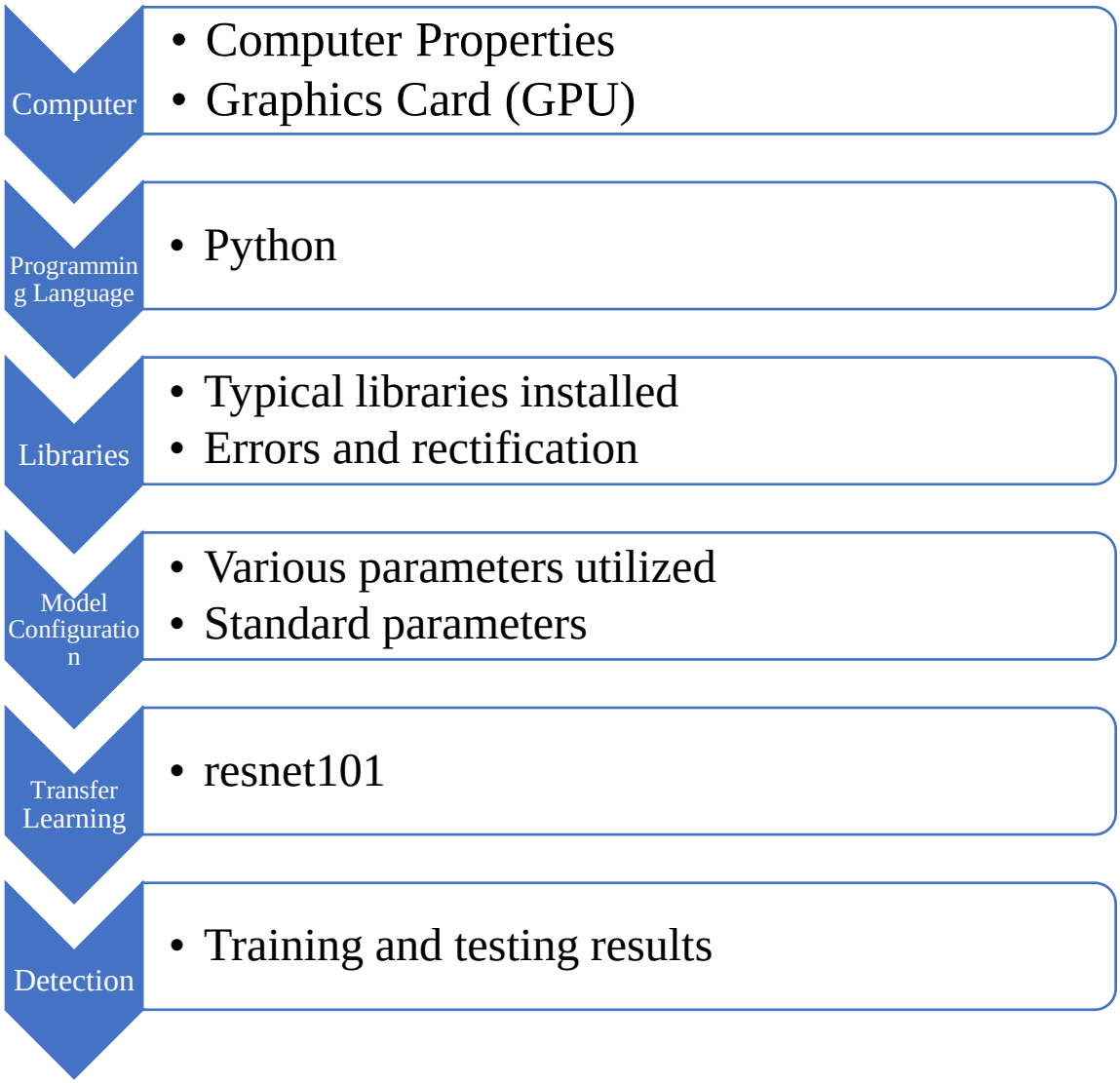



*FIGURE 18 Workflow of MRCNN Model*

The model utilized a windows 10 operating system using an alienware laptop with the following system configuration:

Processor: Intel® Core(TM i7-7700HQ CPU @ 2.80GHz 2.80 GHz

Installed Memory (RAM): 16.0 GB (15.9 GB usable)

System Type: 64-bit Operating System, x64based processor

Graphics Card Name: NVIDIA GeForce GTX 1060

Chip Type: GeForce GTX 1060

Total Memory 14.175 GB

The system configuration is important to note as the deep learning algorithms are computational intensive programs, and are difficult to implement on computer system with less processing capabilities. The processing speed of the computer is directly related to the time it takes for the execution of the deep learning algorithm.

### 3.5.2 Programming Language

This study utilized python programming language to create the M-RCNN model due to its ease of use in coding and the availability of vast libraries which support deep learning models. This study utilized python from anaconda with a python version of 3.6.7.

### 3.5.3 Libraries

Although anaconda is pre-installed with most libraries, deep learning models require the usage of special libraries created for deep learning which must be installed. The libraries include:

i) Tensorflow GPU: Library created by Google in 2016 for open source which assist programmer with building deep learning model

ii) Keras: Modified version of the Tensorflow library which is also utilized in building a deep learning neural network

iii) OpenCV2: Is an open source library which is utilized in the field of computer vision. This library was also required by the research in order to perform the object detection operation and binarization of images.

### 3.5.4 Errors and Debugging

The deep learning Mask-RCNN model is dependent on more than just the three libraries mentioned above and due to these dependencies, there are a lot of errors which occur in the code when the functions from these libraries are called. Therefore, a solution was to create a python environment where older versions of libraries can be downloaded and utilized within that environment. A complete list of all the libraries and version of libraries utilized have been provided in the Appendices.

### 3.5.5 Model Configuration

Figure 19 below is part of the code this study utilized to define the parameters for the M-RCNN model. Some of the parameter which were changed for better performance was image size and steps per epoch and epochs. The parameters are explained in depth in the next chapter.

## Configuration

```
image_size = 960
rpn_anchor_template = (2, 4, 8, 16, 32) # anchor sizes in pixels
rpn_anchor_scales = tuple(i * (image_size // 16) for i in rpn_anchor_template)

class ShapesConfig(Config):
    """Configuration for training on the shapes dataset.
    """
    NAME = "shapes"

    # Train on 1 GPU and 2 images per GPU. Put multiple images on each
    # GPU if the images are small. Batch size is 2 (GPUs * images/GPU).
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

    # Number of classes (including background)
    NUM_CLASSES = 1 + 6  # background + 3 shapes (triangles, circles, and squares)

    # Use smaller images for faster training.
    IMAGE_MAX_DIM = image_size
    IMAGE_MIN_DIM = image_size

    # Use smaller anchors because our image and objects are small
    RPN_ANCHOR_SCALES = rpn_anchor_scales

    # Aim to allow ROI sampling to pick 33% positive ROIs.
    TRAIN_ROIS_PER_IMAGE = 100

    STEPS_PER_EPOCH = 600

    VALIDATION_STEPS = STEPS_PER_EPOCH / 20

config = ShapesConfig()
config.display()
```

*FIGURE 19: Code for Model Configuration*

Image size is the resolution of the image for which the model has to be trained, for example in the model above, image size is set to 960 which means the training images are fed into the network with a size of 960 x 960 pixels. Region Proposal Network (RPN) is the size of the regions on the image on which the model is trained. For example, the size 2 denotes a 2 x 2-pixel matrix which detects all objects within a 2 x 2 grid on the image. For this study, the size of the RPN were kept constant at 2, 4, 8, 16, and 32. Steps per epoch is another parameter which can be changed to determine how often the model updates its weights. This is explained in more detail later, however for this study steps per epoch were changed

to determine a more accurate model. Although, this study only tested some parameters there is a list of parameter, which can be changed to build a more accurate model. The Figure 20 lists all the parameters which can be changed. These parameters can be changed in future studies to develop more accurate models.

```
Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.7
DETECTION_NMS_THRESHOLD        0.3
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 1
IMAGE_MAX_DIM                  960
IMAGE_META_SIZE                19
IMAGE_MIN_DIM                  960
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [960 960   3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0,
'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           shapes
NUM_CLASSES                    7
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (120, 240, 480, 960, 1920)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                600
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           100
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               30.0
WEIGHT_DECAY                   0.0001
```

*FIGURE 20: List of Parameters*

### 3.5.6 Transfer Learning

Transfer learning is a concept from deep learning where a previously trained deep learning model is utilized to initialize weights for a new model to be trained on so that the new model is not initialized with weights of zero. This facilitates programmer to quickly build a model which is accurate without having to start from zero (Bengio 2012) . This study utilized resnet101 as the initial network for transferring of its weights. This is a network created by Microsoft research team and tested on the COCO dataset (He, Zhang et al. 2016).

The results and findings of these experiments are presented in the next chapter.

# CHAPTER 4 RESULTS AND DISCUSSION

## 4.1 Results

There were a total of five different models which were utilized for the training and testing of the images. The development of these five models are summarized in the table below. The first column consists of the name given to this model by this study, and the epoch column summarizes the number of epoch which were executed for each individual model. For example, in the first model a total of 30 epochs were executed by the algorithm. The other columns describe the resolution, learning rate, steps and training time. These terms are explained further below (Table 1).

Table 1: Model configuration and description

| Model Config | Epoch | Resolution | Learning Rate | Steps | AUG | Training Time(hr.) |
|---|---|---|---|---|---|---|
| 30E_1536Res_200S | 30 | 1536 | 0.001 | 200 | 0 | 5 |
| 50E_640Res_400S | 50 | 640 | 0.001 | 400 | 0 | 8 |
| 50E_960Res_600S | 50 | 960 | 0.001 | 600 | 0 | 9 |
| 75E_960Res_600S | 75 | 960 | 0.001 | 600 | 50% | 13 |
| 100E_1024Res_400S | 100 | 1024 | 0.001 | 400 | 0 | 15 |

**Epoch**

One epoch can be described as one full forwards and backwards iteration of the entire dataset through an artificial neural network. One epoch is not sufficient to train the entire neural network as the network calculates the weights in nodes and adjusts the weight of each node after each epoch to make a better prediction. However, accuracy is not solely dependent of the number of epochs a model is trained on (Bengio, Louradour et al. 2009).

**Resolution**

Human vision requires high resolution images to detect object on an image, for example, the crack on a road survey image requires high resolution to be viewed by a human eye. However, in the case of deep learning, the neural network determines patterns in similar value pixels, therefore it does not require high resolution image to make a prediction (LeCun, Bengio et al. 2015). In addition, the higher the resolution of the image the more time the algorithm will take to be trained. Due to this, the resolution can be changed while training the model to determine better accuracy and reduce the training time.

**Learning Rate**

Learning Rate is a parameter in the deep learning algorithm which controls the adjustment of weights of the neural network. Also, learning rate controls the amount of time it takes to train the network. For example, the lower the value the more time it takes to train the model and vice versa. In addition, a high learning rate can make a model less accurate, therefore the learning rate should be ideal (Smith 2017).

**Steps per Epoch**

Steps in the neural network are utilized in order to update and fine tune the weights within the neural network. It can be described as the number of times the performance of the model will be evaluated within one epoch. Steps help develop a more accurate model, however increasing the number of steps does not necessarily increase the accuracy of the model as the accuracy depends on more factors. Steps can be described as a checkpoint which are utilized to train the model and gather information on the performance (Xin, Ma et al. 2018).

**Augmentation**

Deep learning models requires large annotated training, validation, and testing data in order to predict with high accuracy. However, when there is less training data image augmentation can be utilized as method to artificially build training images. In most deep learning algorithms image augmentation is used to mirror the image to build a larger training data. This study utilized image augmentation to mirror the images left to right.

**Training Time**

This study utilized an Alienware computer with a 12GB graphics processing unit (GPU) with a 16GB RAM. However, deep learning and image recognition is a highly computationally intensive program. Therefore, each model required more than a few hours of training time which must be accounted for, as a more powerful computer can produce faster results.

**Training and Testing Images**

All the images were divided into training and testing images; however, all the images were utilized to test the models. Since the sample size of the images for this study was small, the

training images were utilized for validation purpose. A total of 15 images were utilized for training and 5 images for testing the model, with a total of 149 individual annotated distress images.

**Bounding Box and Masking**

Region proposals are generated by the M-RCNN to determine which region contains an object, there are multiple region proposals that are generated within one prediction. However, the region proposal where the object is most likely to exist on the image is predicted. These region proposal are known as bounding boxes, i.e. the area in which the cracks exist (Dai, Li et al. 2016). Within these bounding boxes the M-RCNN algorithms masks the object simultaneously as the bounding boxes are being classified (He, Gkioxari et al. 2017).

**Bounding Box and Masking Results**

The results are comprised of two aspects in this study: first the correct number of bounding boxes correctly identified from the original image and second the average percentage of correct masking within the correctly identified bounding boxes. For example, image 10000000029C_intensity in model 50E_960Res, all the bounding boxes were correctly identified but only an average of half was masked therefore; the bounding box result was 100% and the masking average score was 55%. The Figure 21 below depicts these results.

As observed below the original image contains 3 distresses with bounding boxes and the model predicts all three distresses accurately. However, it masks one of the longitudinal bounding boxes with 100% accuracy whereas it does not mask the transverse bounding

box more than 5%. Therefore, the overall average masking of the correctly identified bounding boxes has been recorded at 55%

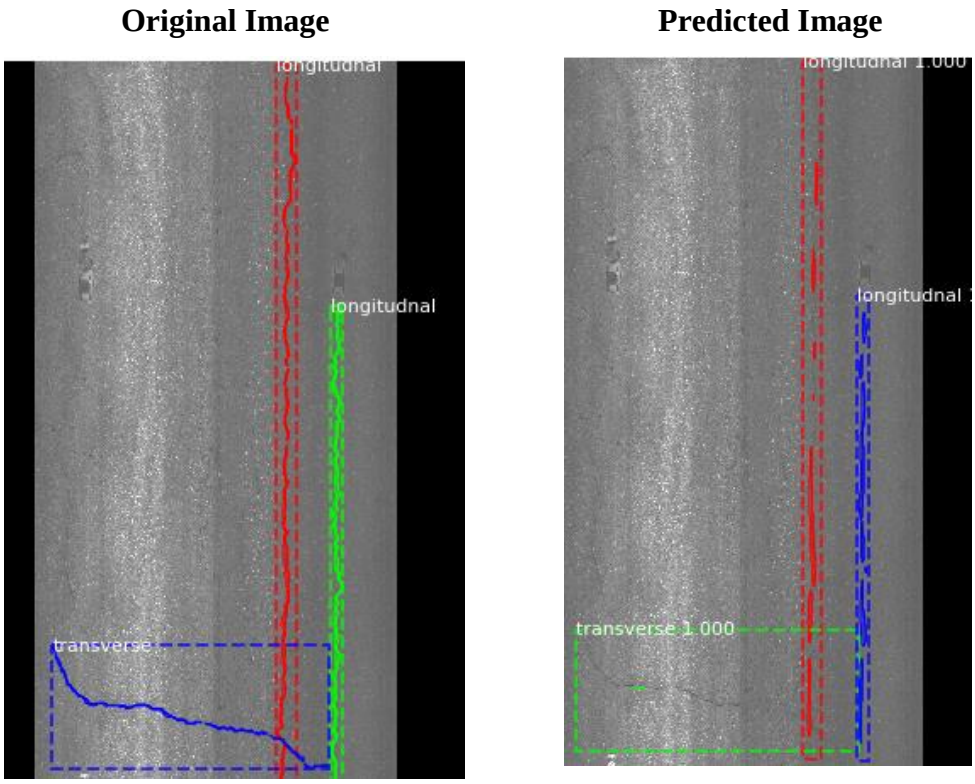



*FIGURE 21: Original and predicted images*

**Outliers**

Due to the small size of labelled data there were two images which were omitted from the test results as outliers to present a more accurate result. This study contained only one labelled alligator cracking which was utilized in testing the model. However, the model was not trained in detection of alligator cracking. Therefore, the results of the alligator cracking have been omitted in the final results. The other image which was omitted from test results had a background significantly different from all the train and test images, hence the cracks were not being detected.

4.2 Sensitivity Analysis

Sensitivity analysis examines how the accuracy of any model can be examined with regards to variations within its input parameters (Pianosi, Beven et al. 2016). In essence, the study can focus on changing the parameters described above in order to make a more accurate model. Therefore, models which share similar input parameters can be compared to determine which variables the models accuracy is dependent on. This can be further utilized to fine tune the model to make a more accurate model.

## 4.3 Cases Compared

As part of the sensitivity analysis, the results have been compared pairwise utilizing two pairs of models which share similar parameters. This was done for a sensitivity analysis in order to conclude which parameters were performing better in the model. This result presents three pairwise comparisons of the models tested:

i) **50E_640Res_400S Vs. 50E_960Res_600S**

ii) **50E_960Res_600S Vs. 75E_960Res_600S**

iii) **50E_640Res_400S Vs. 100E_1024Res_400S**

The Results of these comparisons have been presented below.

## 4.4 DISCUSSION

### 4.4.1 50E_640Res_400S vs. 50E_960Res_600S

These two models are similar due to having run the same number of epochs. However, they are comparable due to having two different parameters i.e. the first model differs in resolution and the number of steps per epoch to the second model. In the first model the image resolution was 640 x 640 pixels where as in the second model it was 960 x 960

pixels, hence the name 50E_640Res_400S vs. 50E_960Res_600S. In addition, both the models have different number of steps per epoch, the first model contains 400 steps whereas the second model contains 600 steps per epoch. The summary of the results is presented in Figure 22. Model 50E_960Res_600S performs 20% better in terms of accuracy of detecting distresses on training images therefore a case could be made that model with 960Res and 600 steps performs is more accurate. However, both models performed similarly for the test images, therefore a conclusive claim cannot be made that one model is better than the other in testing on new images. In addition, in terms of masking both models performed similarly with training and test images. It was concluded that model 50E_960Res_600S performs best in terms of accuracy in predicting distresses on the training images when compared with all other models. However, this is contradictory to the comparisons in test images as all the models perform similar when presented with unseen images. Since the number of epochs were the same and only the resolution and number of steps were increased.

### 4.4.2 50E_960Res_600S vs. 75E_960Res_600S

The main difference between these two models is the increase in the number of epochs and the image augmentation. The first model consisted of 50 epochs and no image augmentation whereas the second model comprised of 75 epochs and 50% image augmentation. All other aspects between the two models were the same. Figure 23 displays the summary of the results. The model 50E_960Res_600S performed 20% better in terms of accuracy than the model which contained 75 epochs. However, the accuracy of the 75E_960Res_600S model is better in testing with unseen images.

### 4.4.3 50E_640Res_400S vs. 100E_1024Res_400S

These model epochs, and resolution were doubled while keeping the same number of steps for both the models. It must be noted that the computer program attained max memory usage at 100 epoch and 1280 x 1280 resolution, therefore the image size was reduced to 1024 x 1024 resolution. The model 50E_640Res_400S performed far better than the 100E_1024Res_400S model in terms of training and testing. Figure 24 summarizes the results of these two models.

## 4.5 Rationale

Studies show accuracy results vary with change in image resolution. Since, most studies in this field have image dataset comprising of several thousand images, the image size is considerably reduced to process the large amount of data. Some studies reduced image size to 128 x 128 pixels in CNN models whereas RCNN models were built using 227 x 227 pixel size (Sharif Razavian, Azizpour et al. 2014). The objectives of each model were different in both: the studies CNN involves object classification whereas, RCNN involved object detection (Girshick, Donahue et al. 2014). Therefore it can be concluded that accuracy can be increased or decreased with change in image resolution. For the purposes of this study, three different resolution of images were tested, and 960 x 960 image resolution proved to be the most accurate.

Most deep learning models perform several 100,000s epochs on large image dataset utilizing super computers. In one study, The Mask-RCNN model consisted of 180,000 epochs, 30000 steps with 135,00 training images from the COCO dataset (He, Gkioxari et al. 2017). However, in the case of this study the maximum number of epochs was 100.

Depending on the available size of the data set increasing the number of epoch causes overfitting of the model which yields less accuracy (Gerfo, Rosasco et al. 2008). Therefore the model must consist of an adequate number of epochs to avoid overfitting. For this study, 50 epochs proved to be the most accurate model along with 960 resolution and 600 steps.

In future studies, with larger training and testing datasets, different parameters should be tested in order to develop a more accurate model. This study provides details of the five models that were evaluated from the dataset created by this study. However, future studies will have to test parameters that would give accurate results by tweaking the model with trial and error.

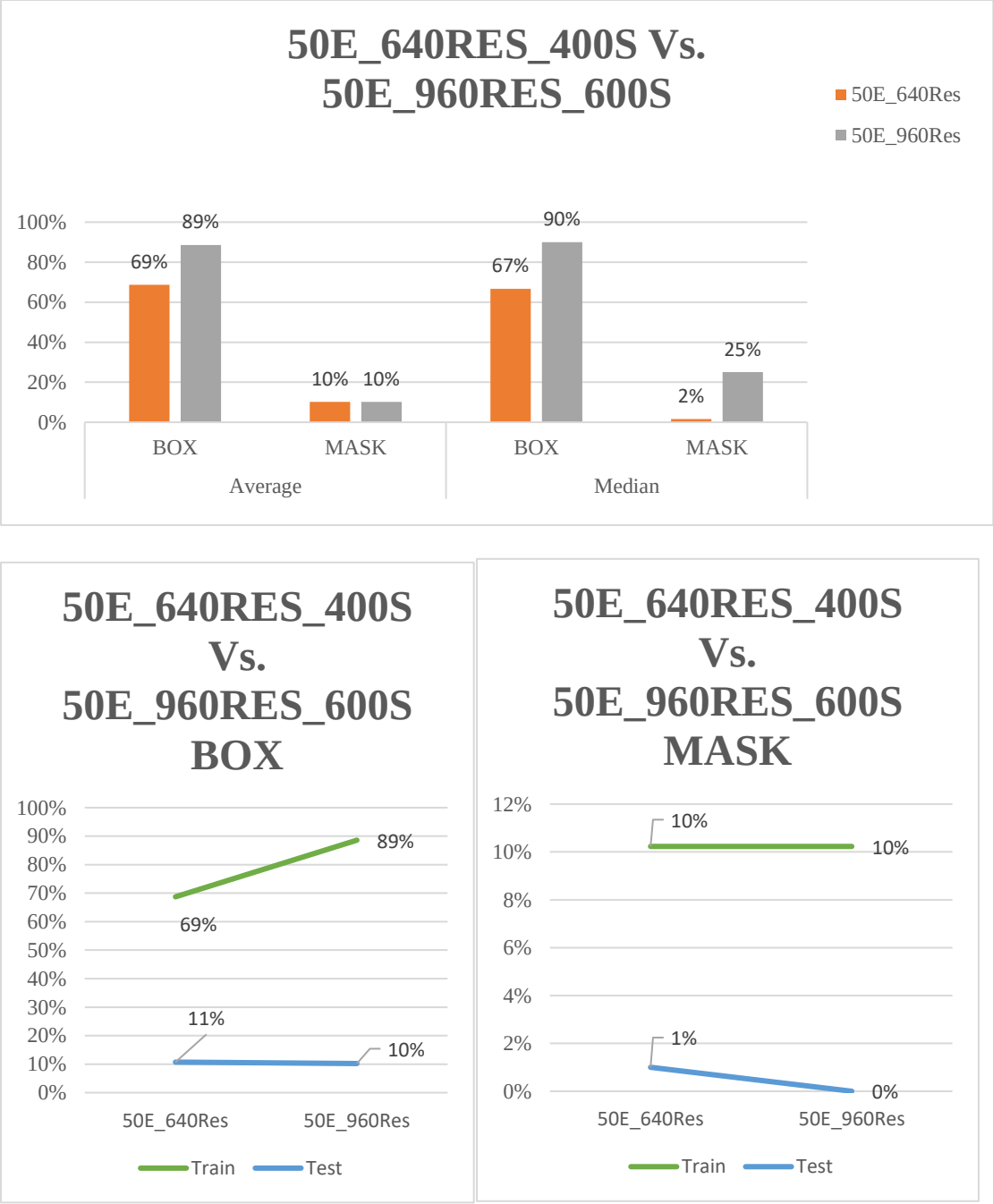



*FIGURE 22: Summary of 50E_640Res_400S vs. 50E_960Res_600S*

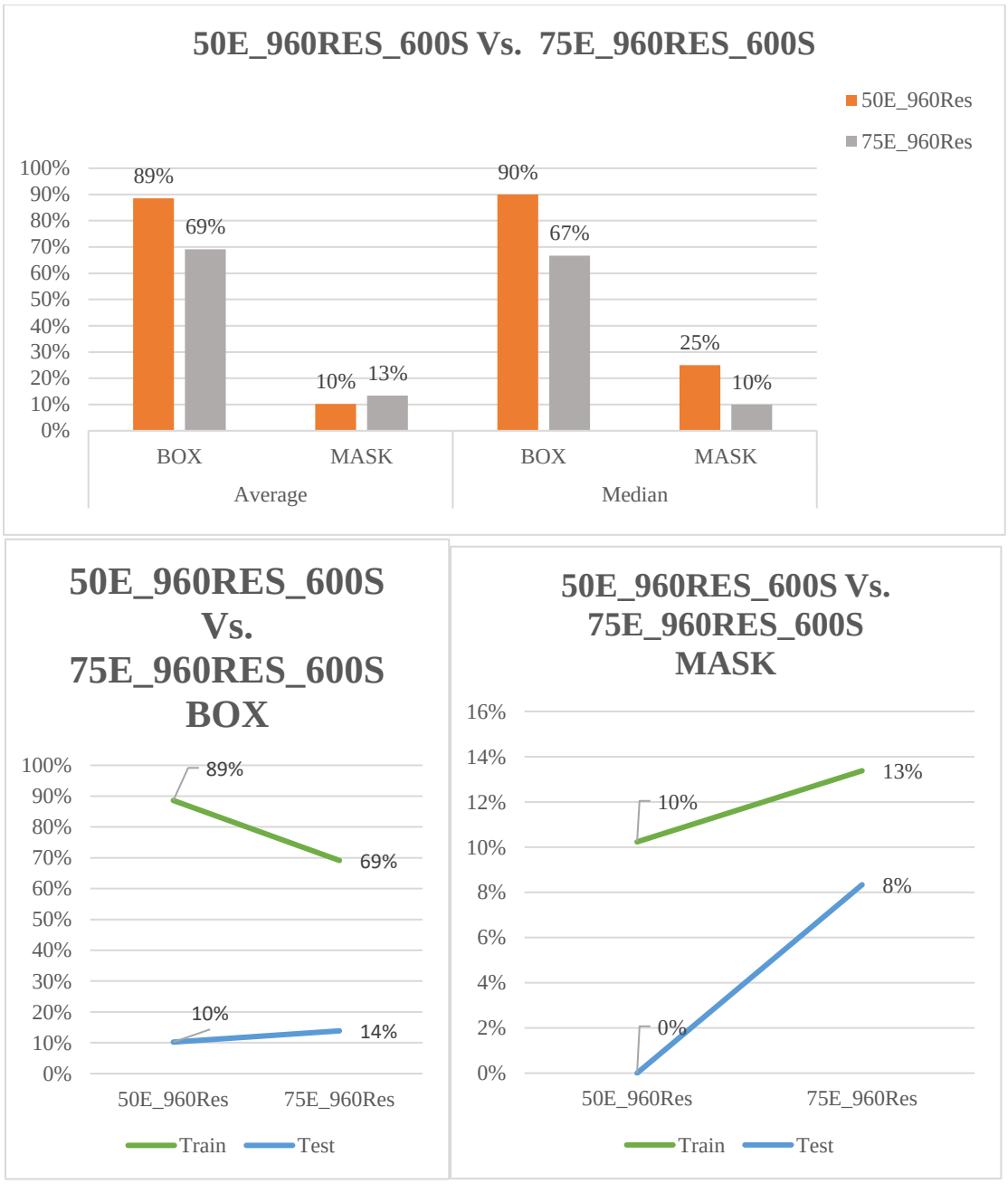
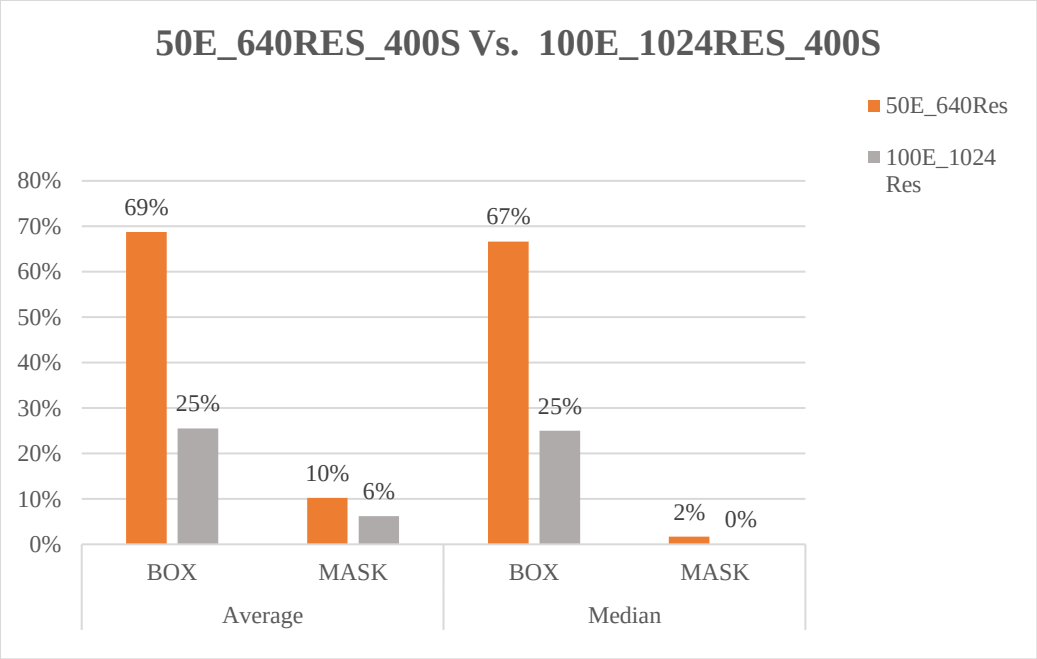
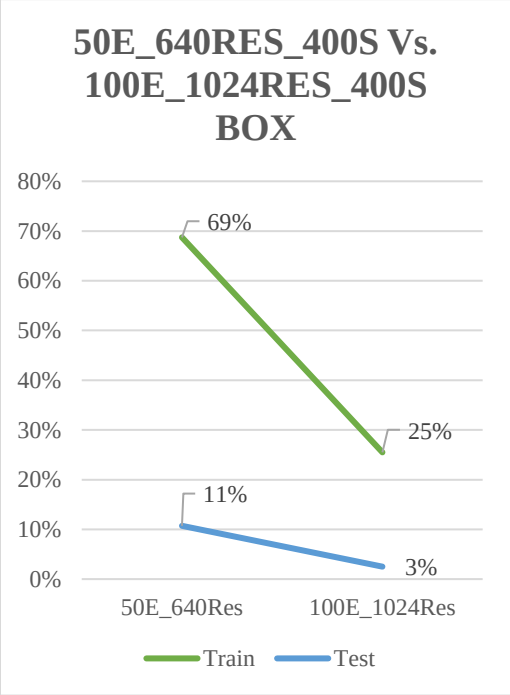



*FIGURE 23: Summary of 50E_960Res_600S vs. 75E_960Res_600S*



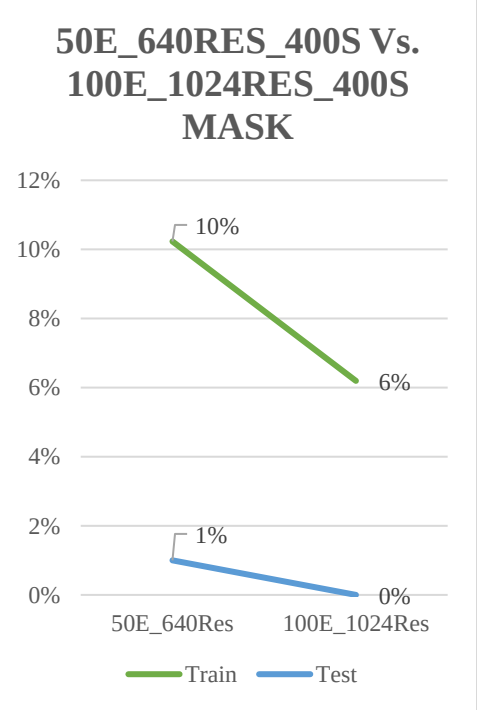




*FIGURE 24: Summary of 50E_960Res_600S vs. 100E_1024Res_400S*

# CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Conclusions

This study tested various tools to annotate images for the deep learning model. The various tools were GIMP, Adobe Photoshop and Lightroom, and WhiteBox. This study concludes that Photoshop is the most efficient tool to annotate images in terms of time, and ease of use. The Photoshop key board short cuts were utilized to generate black and white masks of the image with ease. Therefore Adobe Photoshop is a viable tool in annotating road survey images to build a training, validation and test dataset. Also, Whitebox can be a tool to determine existence of cracks on the road ways, however, further studies maybe required.

The 149 individual distress images that were generated from Adobe Photoshop were applied to develop various Mask-RCNN models. The different parameters of the models included epochs, image resolution, and steps. The list of various models is presented below, it is named according to the parameters changed within the model.

1. 30E_1536Res_200S
2. 50E_640Res_400S
3. 50E_960Res_600S
4. 75E_960Res_600S
5. 100E_1024Res_400S

The description of these models have been explained in the previous chapter. After performing a sensitivity analysis of these algorithms, it was concluded that model 50E_960Res_600S proved to be the best amongst all developed models in terms of training accuracy of 89%. Models 50E_640Res_400S and 75E_960Res_600S are tied for the second best at 69% accuracy in training images. The other two models were not accurate

more than 25% in training therefore have not been mentioned here. Also, all the models performed similarly to test images with an average accuracy of about 12%. Deep learning models require large data in order to predict with high accuracy and 35 images for testing is not sufficient to develop an accurate model. Therefore, more time must be utilized to build a larger dataset to improve the performance of the Mask-RCNN model. This research provides a proof of concept for application of Mask-RCNN for distress detection of road survey images

After development of larger dataset, future studies can utilize the results from this study as a reference for the development of the deep learning model. The model 50E_960Res_600S performed best amongst all the other models and future models can start training the labelled dataset with these parameters. This study built a framework for which a Mask-RCNN model can be utilized to predict distresses on road survey images. Further studies will require a larger dataset to develop more accurate models. In addition, with more accurate masking the severity of the cracks can be determined.

### 5.2 Recommendations

This study provides several recommendations which are listed below:

- **Creation of Larger Dataset**

This study recommends building a larger dataset for training, validation and testing of road survey images. For comparison purposes, CrackNet II contained two thousand labeled images which required a team of 10, eighteen months to annotate (Zhang, Wang et al. 2018). Whereas, this study comprised of 149 individual distress images for training, validation and testing. Even with a limited dataset the training results prove it is possible

to build a highly accurate model with sufficient training data. However, due to the limited sample size the test results were inaccurate therefore, future studies should spend more time in creating larger data set to identify road distresses on survey images.

Some challenges faced in this study was the false classification of road markings as distress type. Therefore, this study recommends annotating road marking in order to train the model and prevent misclassification.

- **Changing Parameters**

This study recommends to change the parameters of the model to test the performance of various parameters. This research only experimented with changing epochs, steps and image resolution within the deep learning model. However, figure number 20 lists the various parameters that can be changed which could help in building a more accurate model. Therefore, this study recommends to evaluate more models with different parameters.

- **Thermal Imaging**

It is difficult to detect cracks on the image due to the similarity in the color of the crack and the background. A possible solution to the background problem is changing the method of data collection from a high resolution camera to thermal imaging. Thermal imaging is utilized across numerous fields for crack detection, for example detection of cracks in weld joints on bridges. Therefore, with the help of thermal imaging, it will be easier to label a data for training as the color of the crack would be different from the color of the road as the distress would act as black body giving a warmer heat signature than surface of theroad (Broberg 2013). This would also help in developing a more accurate deep learning model.

# REFERENCES

https://www.ncdot.gov/divisions/highways/Pages/maintenance-operations.asp

https://medium.com/syncedreview/ibm-builds-the-worlds-largest-facial-image-dataset-to-battle-bias-in-ai-1b4967f332ba

https://www.expertsystem.com/machine-learning-definition/

Arhin, S. A. and E. C. Noel (2014). Predicting Pavement Condition Index from International Roughness Index in Washington, DC.

Ata, R. (2015). "Artificial neural networks applications in wind energy systems: a review." Renewable and Sustainable Energy Reviews 49(534-562).

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. Proceedings of ICML Workshop on Unsupervised and Transfer Learning.

Bengio, Y., et al. (2009). Curriculum learning. Proceedings of the 26th annual international conference on machine learning, ACM.

Broberg, P. (2013). "Surface crack detection in welds using thermography." NDT & E International 57: 69-73.

Cha, Y. J., et al. (2017). "Deep learning-based crack damage detection using convolutional neural networks." Computer-Aided Civil and Infrastructure Engineering 32(5): 361-378.

Chambon, S., et al. (2009). Introduction of a wavelet transform based on 2D matched filter in a Markov Random Field for fine structure extraction: Application on road crack detection. Image Processing: Machine Vision Applications II, International Society for Optics and Photonics.

Chen, D., et al. (2014). "Development and validation of pavement deterioration models and analysis weight factors for the NCDOT pavement management system." Rep. No. FHWA/NC/2011-01, Federal Highway Administration (FHWA), Washington, DC.

Corley-Lay, J., et al. (2010). "Comparison of flexible pavement distresses monitored by North Carolina department of transportation and long-term pavement performance program." Transportation Research Record: Journal of the Transportation Research Board(2153): 91-96.

Dai, J., et al. (2016). R-fcn: Object detection via region-based fully convolutional networks. Advances in neural information processing systems.

Deng, L. (2012). "The MNIST database of handwritten digit images for machine learning research [best of the web]." IEEE Signal Processing Magazine 29(6): 141-142.

Finn, F. (1998). "Pavement management systems--Past, present, and future." Public Roads 62(1).

Gerfo, L. L., et al. (2008). "Spectral algorithms for supervised learning." Neural Computation 20(7): 1873-1897.

GIMP (04/07/2019). "GNU Image Manipulation Program." 4/14/2019, from https://www.gimp.org/.

Girshick, R., et al. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition.

Han, D. and K. Kobayashi (2013). "Criteria for the development and improvement of PMS models." KSCE Journal of Civil Engineering 17(6): 1302-1316.

He, K., et al. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition.

He, K., et al. (2017). Mask r-cnn. Proceedings of the IEEE international conference on computer vision.

Jin, W., et al. (2009). OpinionMiner: a novel machine learning system for web opinion mining and extraction. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM.

Kalogirou, S. A. (2000). "Applications of artificial neural-networks for energy systems." Applied energy 67(1-2): 17-35.

Kourou, K., et al. (2015). "Machine learning applications in cancer prognosis and prediction." Computational and structural biotechnology journal 13: 8-17.

LeCun, Y., et al. (2015). "Deep learning." nature 521(7553): 436.

Lightroom, A. (02/05/2019). "Adobe Photoshop Lightroom." from https://www.adobe.com/products/photoshop-lightroom.html.

Lin, T.-Y., et al. (2014). Microsoft coco: Common objects in context. European conference on computer vision, Springer.

Lindsay, D. J. (1/8/2019). "Welcome to the Whitebox GAT Project."

Ma, C., et al. (2009). Pavement cracks detection based on FDWT. Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on, IEEE.

Mohri, M., et al. (2018). Foundations of machine learning, MIT press.

Nguyen, A., et al. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. Advances in Neural Information Processing Systems.

Oliveira, H. and P. L. Correia (2013). "Automatic road crack detection and characterization." IEEE Transactions on Intelligent Transportation Systems 14(1): 155-168.

Peterson, D. E. (1987). Pavement management practices.

Pianosi, F., et al. (2016). "Sensitivity analysis of environmental models: A systematic review with practical workflow." Environmental Modelling & Software 79: 214-232.

Robert, C. (2014). Machine learning, a probabilistic perspective, Taylor & Francis.

Shafizadeh, K. R., et al. (2002). A statistical analysis of factors associated with driver-perceived road roughness on urban highways, Citeseer.

Sharif Razavian, A., et al. (2014). CNN features off-the-shelf: an astounding baseline for recognition. Proceedings of the IEEE conference on computer vision and pattern recognition workshops.

Siswoyo, D. P. and A. Setyawan (2017). "The Evaluation of Functional Performance of National Roadway using Three Types of Pavement Assessments Methods." Procedia Engineering 171: 1435-1442.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE.

Sun, Y., et al. (2009). Automated pavement distress detection using advanced image processing techniques. Electro/Information Technology, 2009. Eit'09. IEEE International Conference on, IEEE.

Xin, D., et al. (2018). "How Developers Iterate on Machine Learning Workflows--A Survey of the Applied Machine Learning Literature." arXiv preprint arXiv:1803.10311.

Zhang, A., et al. (2017). "Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network." Computer-Aided Civil and Infrastructure Engineering 32(10): 805-819.

Zhang, A., et al. (2018). "Deep Learning–Based Fully Automated Pavement Crack Detection on 3D Asphalt Surfaces with an Improved CrackNet." Journal of Computing in Civil Engineering 32(5): 04018041.

Zhang, L., et al. (2016). Road crack detection using deep convolutional neural network. Image Processing (ICIP), 2016 IEEE International Conference on, IEEE.

# APPENDIX I MASK-RCNN LIBRARIES

## Mask R-CNN ¶

This notebook shows how to train a Mask R-CNN object detection and segementation model on a custom coco-style data set.

[1]:
```
import os
import sys
import random
import math
import re
import time
import numpy as np
import cv2
import matplotlib
import matplotlib.pyplot as plt
import imgaug

sys.path.insert(0, 'D:\\rishabh\\deep-learning-explorer\\mask-rcnn\\libraries')
sys.path.insert(0,'D:\\rishabh\\pycococreator')
from mrcnn.config import Config
import mrcnn.utils as utils
import mrcnn.model as modellib
import mrcnn.visualize as visualize
from mrcnn.model import log
import mcoco.coco as coco
import mextra.utils as extra_utils

%matplotlib inline
%config IPCompleter.greedy=True

HOME_DIR = 'D:\\rishabh\\deep-learning-explorer'
DATA_DIR = os.path.join(HOME_DIR, "data/shapes")
WEIGHTS_DIR = os.path.join(HOME_DIR, "data/weights")
MODEL_DIR = os.path.join(DATA_DIR, "logs")

# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(WEIGHTS_DIR, "mask_rcnn_coco.h5")
# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)
```

```
Using TensorFlow backend.
```

# APPENDIX II CODE OF MODEL

## Model

```
model = modellib.MaskRCNN(mode="training", config=config, model_dir=MODEL_DIR)
```

```
inititalize_weights_with = "coco"  # imagenet, coco, or last

if inititalize_weights_with == "imagenet":
    model.load_weights(model.get_imagenet_weights(), by_name=True)

elif inititalize_weights_with == "coco":
    model.load_weights(COCO_MODEL_PATH, by_name=True,
                       exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                                "mrcnn_bbox", "mrcnn_mask"])

elif inititalize_weights_with == "last":
    # Load the last model you trained and continue training
    model.load_weights(model.find_last()[1], by_name=True)
```

```
# Image Augmentation
# Right/Left flip 50% of the time
augmentation = imgaug.augmenters.Fliplr(0.5)
```

APPENDIX III CODE OF MODEL TRAINING

# Training

Training in two stages

## Heads

Only the heads. Here we're freezing all the backbone layers and training only the randomly initialized layers. To train only the head layers, pass layers='heads' to the train() function.

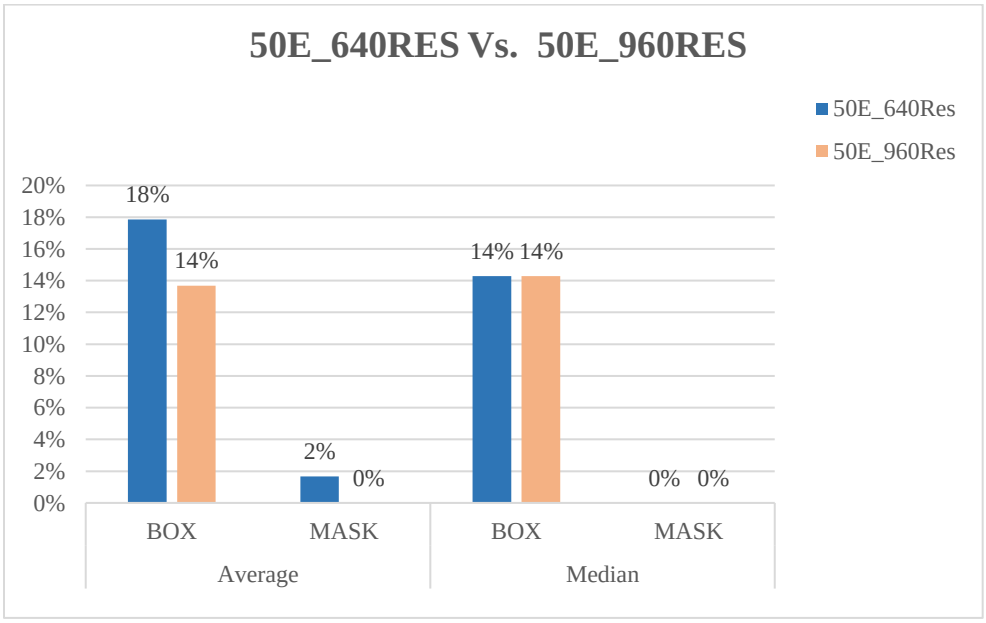
```
model.train(dataset_train, dataset_validate,
            learning_rate=config.LEARNING_RATE,
            epochs=50,
            layers='heads')
```

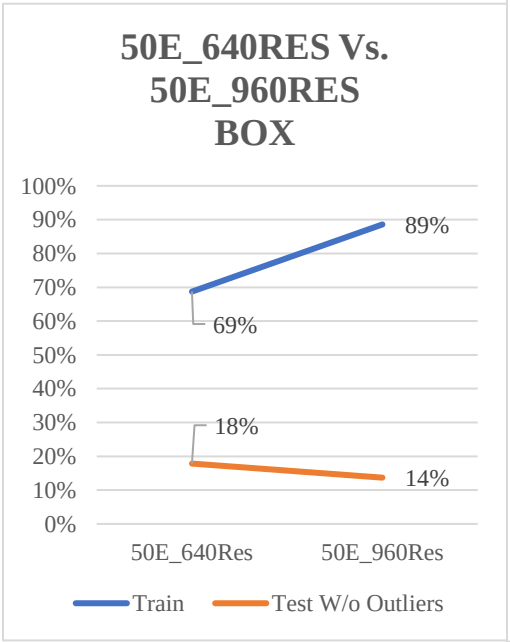
# APPENDIX IV LIST OF LIBRARY REQUIREMENTS FOR THE PYTHON MASK-RCNN ENVIRONMENT

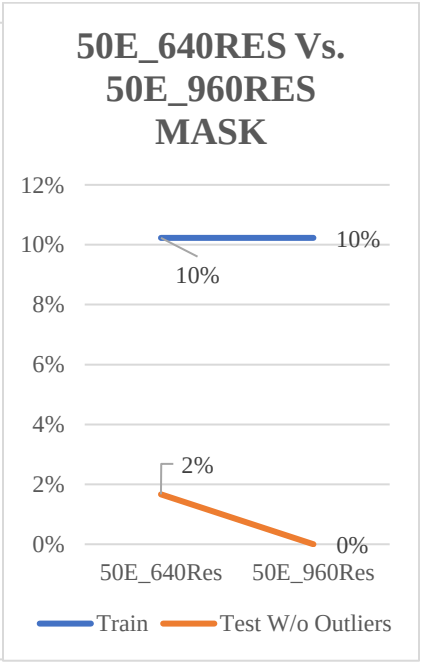
absl-py==0.7.0
alabaster==0.7.12
anaconda-client==1.7.2
anaconda-project==0.8.2
asn1crypto==0.24.0
astor==0.7.1
astroid==2.1.0
astropy==3.1
atomicwrites==1.2.1
attrs==18.2.0
Babel==2.6.0
backcall==0.1.0
backports.os==0.1.1
backports.shutil-get-terminal-size==1.0.0
beautifulsoup4==4.6.3
bitarray==0.8.3
bkcharts==0.2
blaze==0.11.3
bleach==3.0.2
bokeh==1.0.2
boto==2.49.0
Bottleneck==1.2.1
certifi==2018.11.29
cffi==1.11.5
chardet==3.0.4
Click==7.0
cloudpickle==0.6.1
clyent==1.2.2
colorama==0.4.1
comtypes==1.1.7
conda==4.6.7
conda-build==3.17.6
contextlib2==0.5.5
cryptography==2.4.2
cycler==0.10.0
Cython==0.29.2
cytoolz==0.9.0.1
dask==1.0.0
datashape==0.5.4
decorator==4.3.0
defusedxml==0.5.0
distributed==1.25.1
docutils==0.14
entrypoints==0.2.3
et-xmlfile==1.0.1
fastcache==1.0.2
filelock==3.0.10
Flask==1.0.2
Flask-Cors==3.0.7
gast==0.2.2
gevent==1.3.7
glob2==0.6
greenlet==0.4.15
grpcio==1.19.0
h5py==2.8.0
heapdict==1.0.0
html5lib==1.0.1
idna==2.8
imageio==2.4.1
imagesize==1.1.0
imgaug==0.2.8
importlib-metadata==0.6
ipykernel==5.1.0
ipython==7.2.0
ipython-genutils==0.2.0
ipywidgets==7.4.2
isort==4.3.4
itsdangerous==1.1.0
jdcal==1.4
jedi==0.13.2
Jinja2==2.10
jsonschema==2.6.0
jupyter==1.0.0
jupyter-client==5.2.4
jupyter-console==6.0.0
jupyter-core==4.4.0
jupyterlab==0.35.3
jupyterlab-server==0.2.0
Keras==2.2.4
Keras-Applications==1.0.7
Keras-Preprocessing==1.0.9
keyring==17.0.0
kiwisolver==1.0.1
lazy-object-proxy==1.3.1
libarchive-c==2.8
llvmlite==0.26.0
locket==0.2.0
lxml==4.2.5
Mako==1.0.7
Markdown==3.0.1
MarkupSafe==1.1.0
matplotlib==3.0.2
mccabe==0.6.1
menuinst==1.4.14
mistune==0.8.4
mkl-fft==1.0.6
mkl-random==1.0.2
mock==2.0.0
more-itertools==4.3.0
mpmath==1.1.0
msgpack==0.5.6
multipledispatch==0.6.0
nbconvert==5.4.0
nbformat==4.4.0
networkx==2.2
nltk==3.4
nose==1.3.7
notebook==5.7.4
numba==0.41.0
numexpr==2.6.8
numpy==1.15.4
numpydoc==0.8.0
odo==0.5.1
olefile==0.46
opencv-python==4.0.0.21
openpyxl==2.5.12
packaging==18.0
pandas==0.23.4
pandocfilters==1.4.2
parso==0.3.1
partd==0.3.9
path.py==11.5.0
pathlib2==2.3.3
patsy==0.5.1
pbr==5.1.3
pep8==1.7.1
pickleshare==0.7.5
Pillow==5.3.0
pkginfo==1.4.2
pluggy==0.8.0
ply==3.11
prometheus-client==0.5.0
prompt-toolkit==2.0.7
protobuf==3.7.0
psutil==5.4.8
py==1.7.0
pycocotools==2.0
pycodestyle==2.4.0

pycosat==0.6.3
pycparser==2.19
pycrypto==2.6.1
pycurl==7.43.0.2
pyflakes==2.0.0
Pygments==2.3.1
pygpu==0.7.6
pylint==2.2.2
pyodbc==4.0.25
pyOpenSSL==18.0.0
pyparsing==2.3.0
PySocks==1.6.8
pytest==4.0.2
pytest-arraydiff==0.3
pytest-astropy==0.5.0
pytest-doctestplus==0.2.0
pytest-openfiles==0.3.1
pytest-remotedata==0.3.1
python-dateutil==2.7.5
pytz==2018.7
PyWavelets==1.0.1
pywin32==223
pywinpty==0.5.5
PyYAML==3.13
pyzmq==17.1.2
QtAwesome==0.5.3
qtconsole==4.4.3
QtPy==1.5.2
requests==2.21.0
rope==0.11.0
ruamel-yaml==0.15.46
scikit-image==0.14.1
scikit-learn==0.20.1
scipy==1.1.0
seaborn==0.9.0
Send2Trash==1.5.0
Shapely==1.6.4.post1
simplegeneric==0.8.1
singledispatch==3.4.0.3
six==1.12.0
snowballstemmer==1.2.1
sortedcollections==1.0.1
sortedcontainers==2.1.0
Sphinx==1.8.2
sphinxcontrib-websupport==1.1.0
spyder==3.3.2
spyder-kernels==0.3.0
SQLAlchemy==1.2.15
statsmodels==0.9.0
sympy==1.3
tables==3.4.4
tblib==1.3.2
tensorboard==1.13.0
tensorflow==1.12.0
tensorflow-estimator==1.13.0
termcolor==1.1.0
terminado==0.8.1
testpath==0.4.2
Theano==1.0.3
toolz==0.9.0
tornado==5.1.1
tqdm==4.28.1
traitlets==4.3.2
typed-ast==1.1.0
unicodecsv==0.14.1
urllib3==1.24.1
wcwidth==0.1.7
webencodings==0.5.1
Werkzeug==0.14.1
widgetsnbextension==3.4.2
win-inet-pton==1.0.1
win-unicode-console==0.5
wincertstore==0.2
wrapt==1.10.11
xlrd==1.2.0
XlsxWriter==1.1.2
xlwings==0.15.1
xlwt==1.3.0
zict==0.1.3

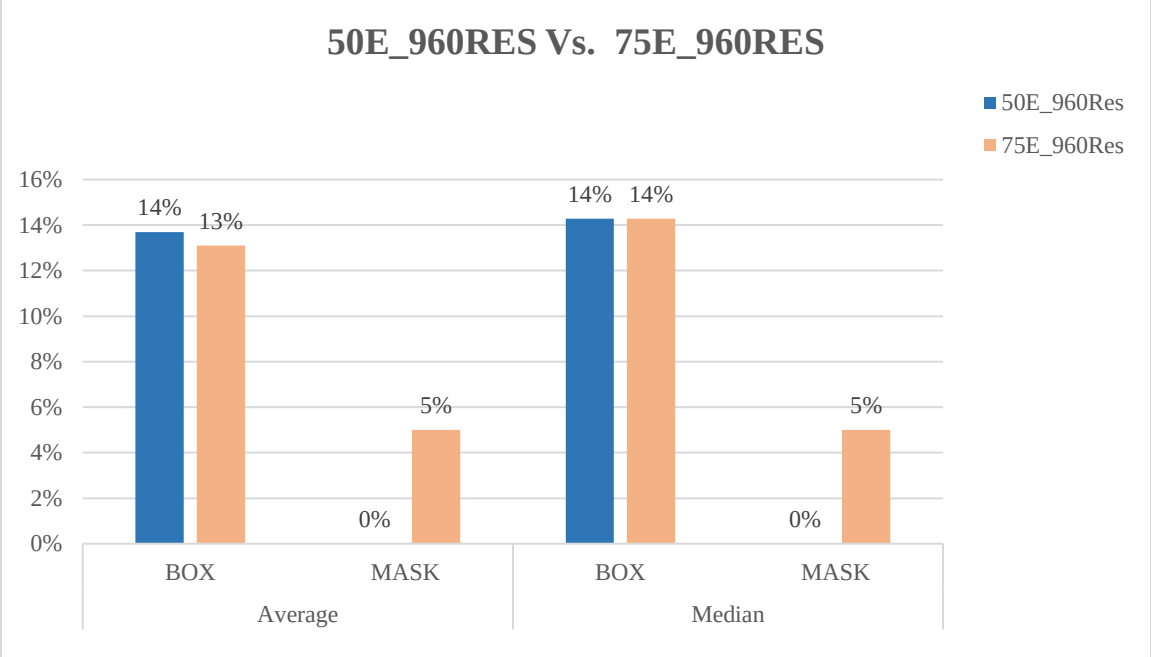
APPENDIX V TEST RESULTS WITHOUT OUTLIERS 50E_640RES VS. 50E_960RES

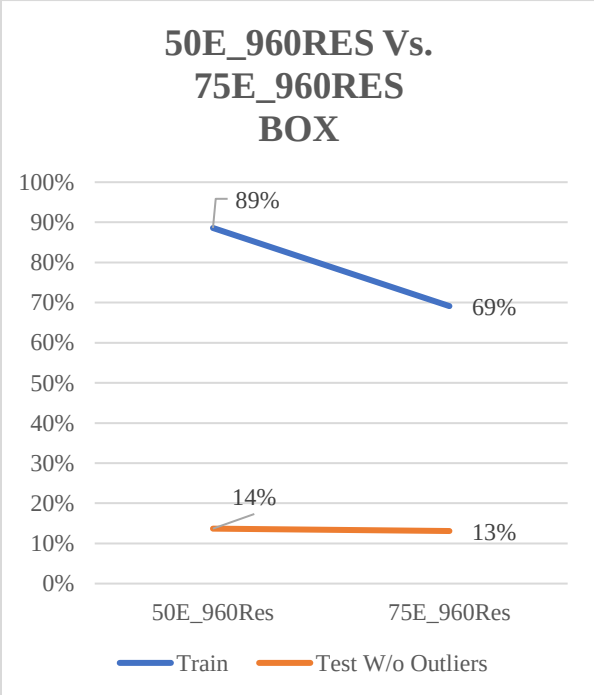


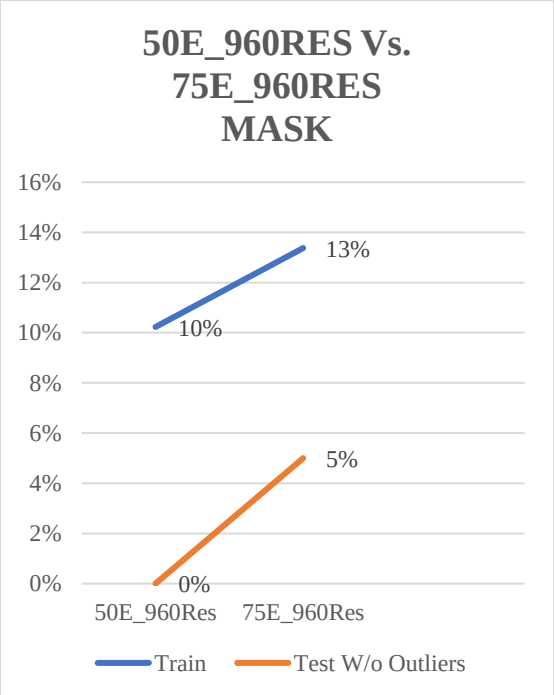



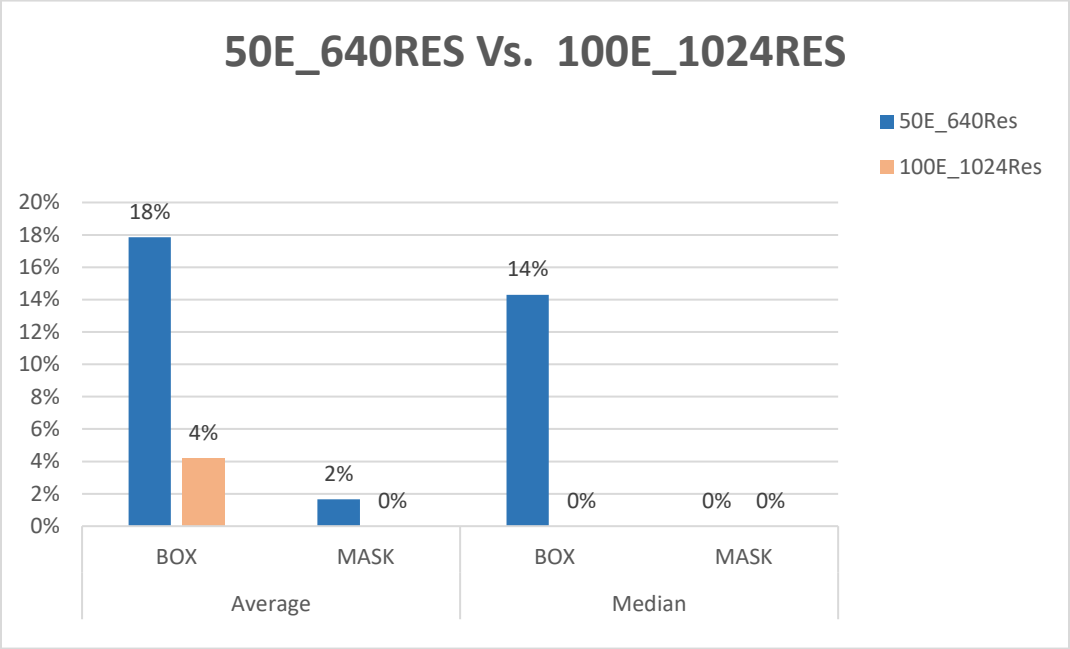
APPENDIX VI TEST RESULTS WITHOUT OUTLIERS 50E_960RES VS. 75E_960RES

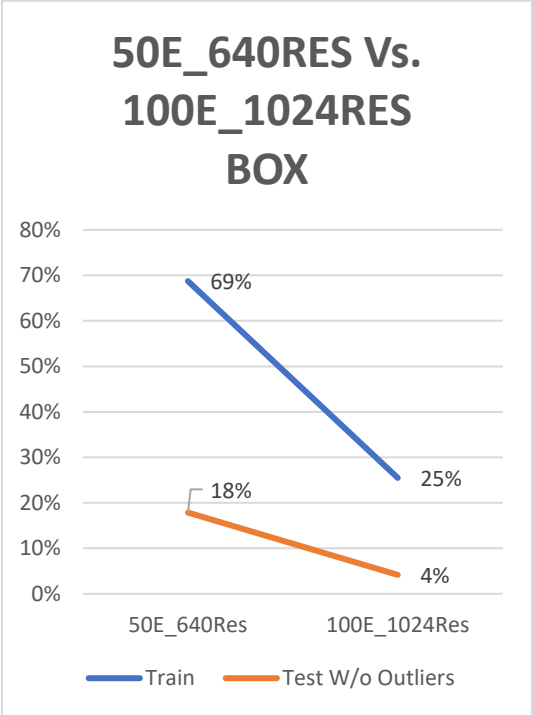


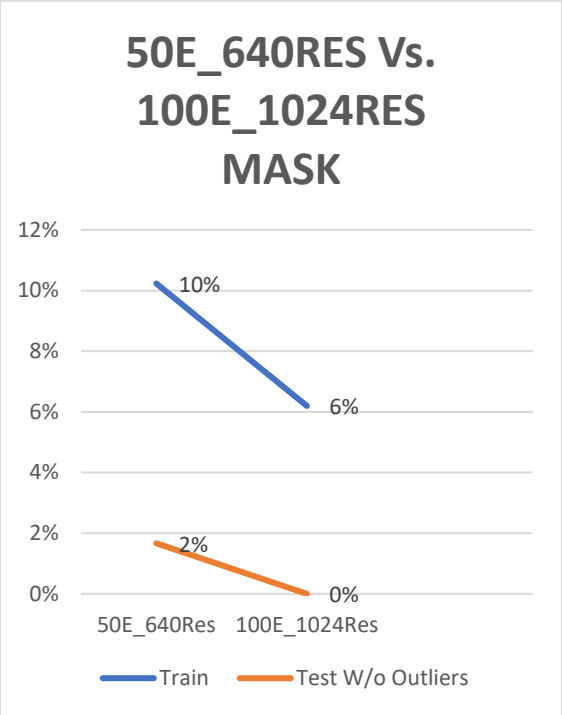



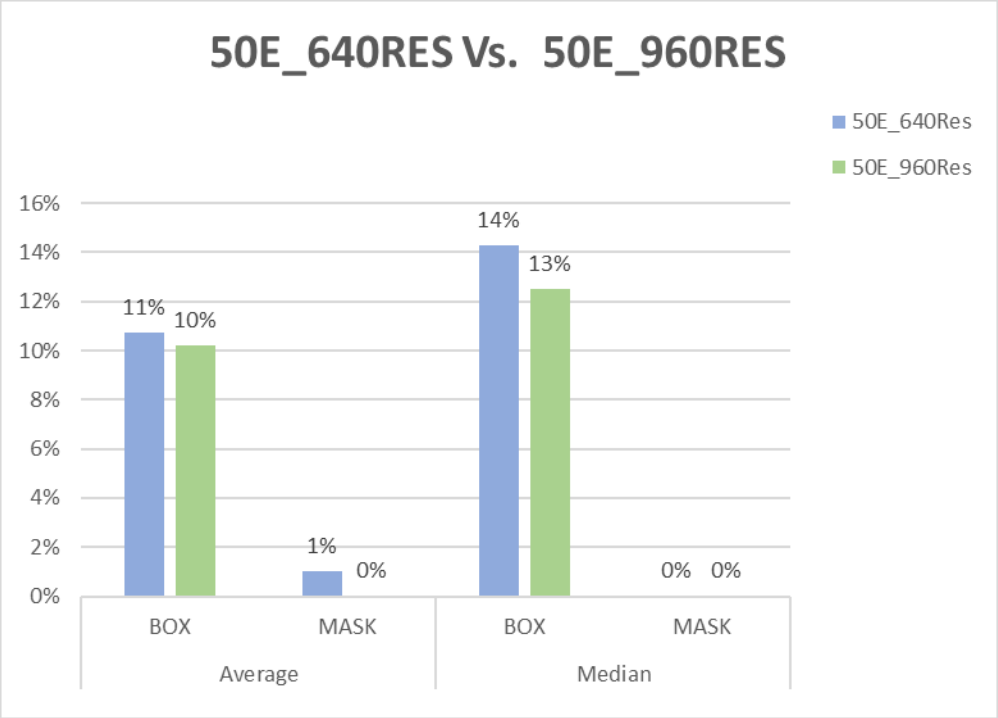
APPENDIX VII TEST RESULTS WITHOUT OUTLIERS 50E_640RES VS. 100E_1024RES

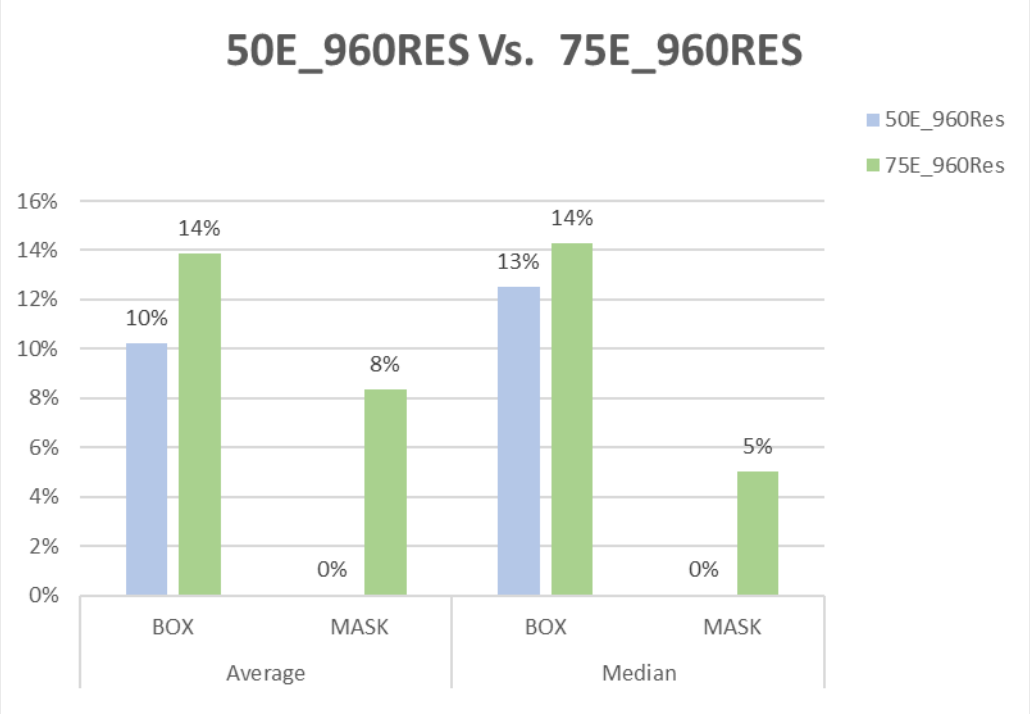


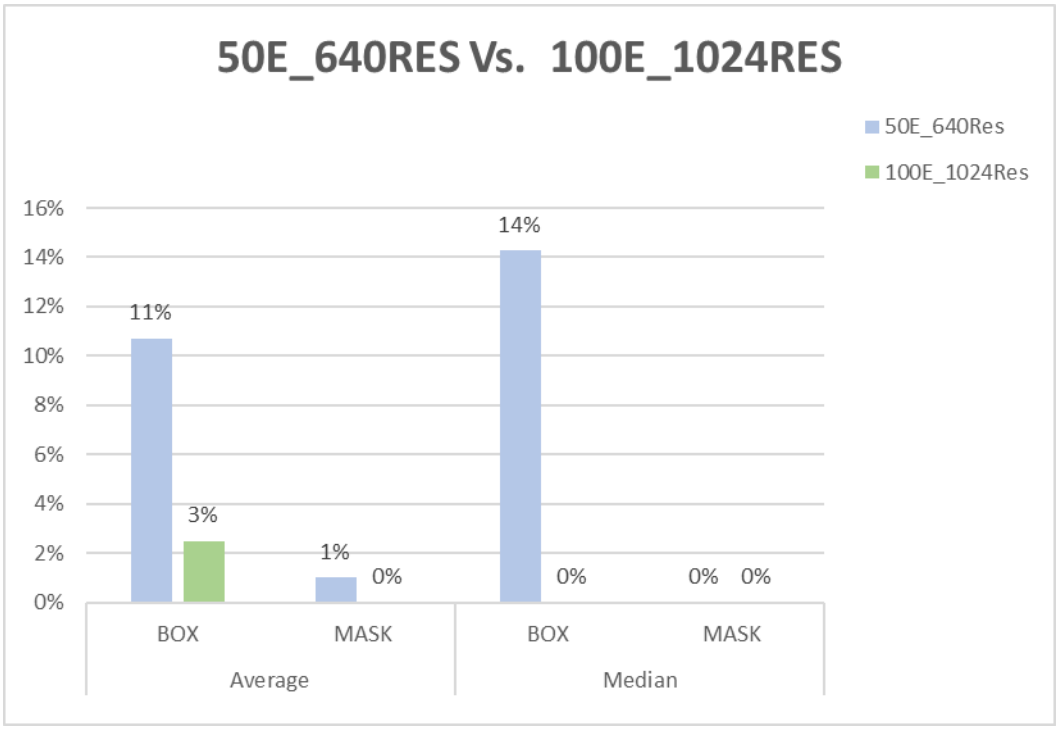



APPENDIX VII TEST RESULTS WITH OUTLIERS




50E_640RES Vs. 100E_1024RES
50E_640Res
100E_1024Res
16%
14%
12%
10%
8%
6%
4%
2%
0%
11%
3%
1%
0%
14%
0%
0%
0%
BOX
MASK
BOX
MASK
Average
Median