Using State-of-the-Art Machine Learning Models for Binary Classification of Road Rutting Severity with an Unbalanced Dataset

Akhil Naseem
Engineering and Mathematics Sciences
University of Western Australia
Perth, Australia

Supervisors

Dr. Ghulam Mubashar Hassan, School of Physics, Maths and Computing, Computer Science and Software Engineering

Dr. Du Huynh, School of Physics, Maths and Computing, Computer Science and Software Engineering

Dr. Mark Reynolds, Head of School, School of Physics, Mathematics and Computing

Word count: 7645

Abstract

Surveying road networks regularly can be an expensive maintenance task. Well maintained roads allow for safe and efficient transportation for vehicles and their passengers. However, for an extensive road network and limited inspection personnel, roads may be not inspected as often as they should be. With this motivation, Main Roads WA aims to automate their road monitoring process to detect signs of a particular type of road defect – rutting. Using current road rutting data, along with visual feeds of the road, this paper presents and evaluates an approach to use computer vision techniques and state-of-the-art machine learning models to predict areas with severe rutting.

Introduction

One of Western Australia's largest assets is its road network, with over $10 billion AUD worth of pavements and surface assets under the management of Main Roads WA. With an estimated 19.7 million vehicles currently using the infrastructure spread across 2.5 million square kilometres, road conditions must be maintained with regular inspections to detect, identify, and repair defects and damages that could affect both the safety and efficiency of motorways. As roads age, the number of pavements to be inspected are expected to increase. The task is usually carried out by field experts making visual on-site inspections based on pavement data collected during survey runs. However, with the limited number of field experts available, Main Roads WA will need to manage their human and financial resources to keep up with the increasing demand for road inspections. Hence the motivation for an innovative solution to automate the manual identification of road defects.

This study aims to report on the practicality and accuracy of using the computing prowess of machine learning algorithms and image processing tools to identify road rutting defects without the need for a field expert to inspect the sites physically. Main Roads WA has deployed survey vehicles with sensors that record periodic rutting levels along the wheel paths of the vehicle, and a visual feed of the road captured by a front-mounted camera. Although this data is available for this study to implement an image processing and machine learning solution, there are limitations of an unbalanced dataset (i.e. roads with minimal rutting far outnumber roads with severe rutting), the lack of a subject matter expert to help understand and verify the accuracy of the ground truth, and the mis-synced irregularities between the sensor data and the visual feed. This study aims to solve the road damage classification and unbalanced dataset problem using a combination of video pre-processing, data sampling, and state-of-the-art image classification machine learning models including – VGG16, ResNet152, and MobileNet. This paper first describes the pipeline of steps taken to extract, identify, and mask areas of the visual feed to the relevant road lane only. Second, it reports on the model evaluations for state-of-the-art classification models. And lastly, makes recommendations for further investigation of the project based on the limitations and results.

Background Information

Road defects - Rutting

Rutting is a form of road defect that refers to the development of excessive channelised deformation in the wheel paths from heavy loads of traffic over a period of time [1]. The severity of the rut is expressed as the maximum vertical displacement in the transverse profile of the road from a reference point measured at 90 degrees to the edge of the road [2]. Traffic loads on roads, especially by heavy freight vehicles, contribute or cause permanent deformation in the form of longitudinal depressions. According to Roberts and Martin, rutting and roughness are key indicators for loss of pavement shape due to such depressions inadvertently causing heaving on road shoulders and edges [3].

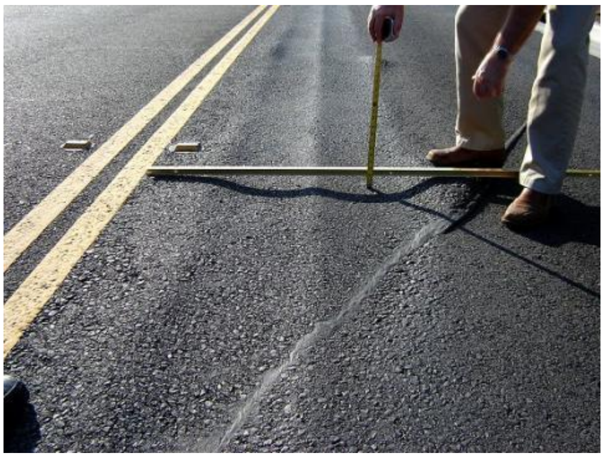


Figure 1 Example of severe road rutting (Quality Engineering Solutions, Inc)

Rutting has direct implications on the road performance (if not as much on the network performance) as it influences vehicle tyre tracking and compromises on safety due to hydroplaning. With the depressions acting as grooves, tyres are pulled towards the rut path as it is steered across the rut. This risk is further aggravated if there is water ponding on the road surface [4]. Without sufficient tread on the tyres, the vehicle will become unresponsive to steering inputs as the tyres do not dispense enough water away from the path. The combination of these three factors lead to the potential loss of vehicle skid resistance, compromising road user safety [2]. Hence the need for regular inspection and repair of road rutting.

Austroads has specified the surveying practices to follow when measuring rutting in its technical information guide [2]. It details that, at a minimum, non-contact measuring methods should sample the entire length of the pavement's left wheel path at a spacing of no greater than 250mm. The samples are then reported as mean rut depth (mm) and standard deviation at intervals of 100m, along with the extent of the rut – the percentage of the length with maximum rut depths in 'bins' (e.g., rut $\leq$ 5 mm, 5 mm $<$ rut $\leq$ 10 mm, 10 mm $<$ rut $\leq$ 15 mm etc). The data must be properly referenced using an established predefined location referencing system. Automatic measurement technologies assisted by vehicle-mounted lasers are capable of measuring transverse road profiles while the host-vehicle travels at normal traffic speeds [2]. For data validation purposes, automatic measurement readings should record a minimum of three points per wheel path, and a minimum of five points if measuring for both wheel path and lane rut depths. Though data can be obtained using automated techniques, they are often less economical for small scale audits. Furthermore, data collection using these methods does not negate the need for field experts to collect information on the type, location and cause of the rutting [2].

State-of-the-Art Machine Learning Models for Classification

Machine learning has become increasingly popular in solving vision-based problems as fundamental algorithms become easily available through open-source libraries and the widespread availability of datasets across a variety of industries. To classify images, machine learning models estimate the concepts and features of an object in an image frame. However, this is often difficult due to different lighting conditions, viewpoints, and the presence of other unlabelled objects [5]. So, the aim of image classification is to determine which class a labelled object belongs to. The natural steps to achieve this result are - selection of the region of interest, feature extraction and classification [6].

Convolutional Neural Networks (CNNs) is one of the main categories of deep neural networks used in image classification tasks. CNNs are capable of classifying faces, cards, street signs and many other types of objects. The underlying block of a CNN is a convolutional layer that performs convolution operations (like edge detection, sharpening, vertical blur etc) on the input image using filters (or otherwise known as kernels), to produce an output map [7]. The CNN activates a filter on areas of the image that it detects to have some specific type of feature. The second type of network layer in a CNN is a pooling layer. Pooling layers are usually placed between convolutional layers, and their purpose is to reduce the number of parameters in the network. The goal is to subsample the input image by sliding windows over the image and only output some aggregate (like maximum or average) for each slide [7]. Hence reducing the size of the input image to the next layer. An activation function is used to map the resulting values of the neural network, which may be negative or greater than one, into a probability between 0 and 1. For a multi-class classification problem, we use the SoftMax activation function. The function returns the probability for an object belonging to each individual class. Many state-of-the-art CNN models have been developed over the years with varying performance benchmarks for different types of classification tasks. Three such popular models are VGG16, ResNet152, and MobileNet.

As CNNs gained popularity, newer models grew both in size and complexity. VGG16 was developed in 2014 with 16 convolutional layers [8]. Although sizeable compared to initial versions of CNNs like LeNet with only seven layers [9], VGG16 became a preferred model due to its very uniform network architecture. The network uses 3×3 convolutional layers, each with stride equal to one. It also uses four 2×2 max pooling layers to reduce the image size by half, with an initial input RGB image of dimensions 224×224 [8]. However, the convolutional layers have very large number of input and output channels. This means the number of parameters is very large and therefore means much slower performance. Hence why even though there exist other models with many more layers like ResNet with 50 layers but only 25 million parameters [10], VGG16 has 138 million [8]. Nevertheless, the advantage to its simplicity makes VGG16 easier to build, code and train, and remains a state-of-the-art model to this day.

In 2015, Kaiming He et. al published a paper in which the team argued that accuracy saturation (also known as degradation) correlated with increasing network depth, is not caused by overfitting alone [10]. As deeper networks are supersets of a shallow network, the paper argues that the extra layers should theoretically be able to learn the identity function and therefore achieve similar accuracies as smaller networks. However, learning identity functions are just as difficult as learning other features. Hence, the paper introduces residual learning in its networks. The proposed model implements very deep networks (up to 152 layers), much fewer channels in its convolutional layers compared to VGG16, with identity mapping shortcuts between layer pairs [10]. Since the identity shortcuts do not introduce any extra parameters and the model layers having fewer filters, ResNet152 has 57% fewer parameters than VGG16, even with 136 more layers. The resulting model does not suffer from the degradation problem, and thus enjoys the accuracy gains from increased depth. ResNet152 scored 5.71% in top-5 error rate for the ImageNet dataset, while VGG16 scored 9.33% [10].

While the general trend for building CNNs have made them deeper like ResNet152, or requiring more compute power like VGG16, MobileNet was introduced as an efficient network architecture that is relatively shallow, with little trade-off in performance. Unlike standard fully connected convolutional layers that convolve on all channels of an input (e.g. RGB channels in an image) as one feature, MobileNet uses depthwise separable layers that convolve on a single input channel at a time before a final channel combination step [11]. The result is that MobileNet spends 95% of its computation time in 1×1 convolutions which also has 75% of the parameters of the 28 layered model [12]. 1×1 convolutions require significantly fewer computations compared to 3×3 layers in VGG16. Additionally, due the relatively shallow depth of the model, regularization and data augmentation are not particularly required, as small models have less trouble with overfitting. Due to its small size, the main application of MobileNet is for deployment on low-powered machines or mobile devices (hence the name). So, to cater to a variety of machine configurations and to reduce latency in real-time applications, MobileNet provides two parameters that help construct even smaller and less computationally expensive models – the width multiplier and the resolution multiplier. The width multiplier thins the network uniformly at each layer, whereas the resolution multiplier sets the input resolution of the network to square images of dimension 224, 192, 160, or 128 pixels [12]. In effect, MobileNet is only one percent lower in accuracy than VGG16 for the ImageNet dataset, whilst being 32 times smaller and 27 times less compute intensive [12].

Now that we've explored three candidate state-of-the-art models for this paper's task, the next section will discuss approaches in literature for similar road defect detection problems.

Approaches in Literature

An image processing approach to detecting road damage is not new and has been in study over the last two decades. With sensors and cameras becoming more available and inexpensive, we are seeing growth in road damage datasets [13]. Both the availability of data collection hardware, and the advancements in computer vision technologies, have given rise to a handful of papers presenting machine learning solutions for road damage detection. For example, a previous study by Maeda et al. proposed an automated damaged detection algorithm using the MobileNet and InceptionV2 single-shot detectors on a dataset with over 15,000 annotated instances in images of damaged roads captured my smartphones set up on the dashboard of vehicles in Japan [14]. Angulo et al. [15] and Roberts at al. [16] continued the work of Maeda et al. [14] by expanding the dataset with images from Italy and Mexico, but the underlying methodology remained the same, except they also consider the severity of the damages. The different classes of damages within this dataset were still highly imbalanced and therefore the models suffered in very low recall for some classes. Nevertheless, this method could be used as a preliminary survey of public roads before more expensive methods of data collection (like laser sensor data) are used. In a similar fashion, Tedeschi and Benedetto used OpenCV's cascade classifiers on smartphone images of roads to detect fatigue cracks, potholes and transversal/longitudinal cracks with high accuracy, precision, and recall (> 0.72) in real-time [17]. Zhang et al. [18] also used smartphone images for detect presence of damage but does not evaluate its severity. Fan et al. used a CNN to recognise different pavement conditions that worked well even in varying road textures [19]. The paper also suggests a 1:3 sampling ratio to counteract the fact that there are more pixels without cracks than pixels with cracks. Nevertheless, the above studies only measure and detect cracking defects and not deformation defects like rutting.

Other studies like that by Majidifard et al. use the approach of using manually annotated images obtained from Google Street View for classification tasks [20]. Even though the data must be manually annotated, the source provided an abundance of training data and the flexibility of camera orientations for easier image processing. A hybrid model of the YOLO and U-net algorithms was used to develop a comprehensive pavement condition program which output both the class and the severity of the road damage. However, again, the lack of 3D data on the images meant that rutting was not a deformation type the model could be trained to learn.

In a study by Rajab, videos were captured of highway roads in the city of Makkah in the Kingdom of Saudi Arabia which suffered from a lot of rutting. Rajab used the correlation between images of rut regions and a base non-rut region to detect if the current frame contained a rut [21]. The lower the correlation to the base non-rut, the lower the probability of rutting in the current frame. This approach is simple and provides an intuitive plot to recognise locations of rut over a length of pavement. In addition, the difference in correlation also indicates the severity of the rut. Even though, to the best of knowledge when writing this paper, this study was the only one that attempts to detect rutting, it is limited to only four highway sites and lacked quantitative evaluation.

Based on current literature, there is a gap for research in classifying rutting severity. Most papers focus mainly on crack damages and not surface deformations. This could be partly due to the lack of data for rutting examples and hence training a model for rutting detection becomes difficult. The advantage in this proposed thesis is that even though the dataset is unbalanced, it is labelled, and therefore annotations and sampling methods could help the rutting detection challenge.

Problem Statement and Dataset

While previous efforts in literature have been motivated by academic interest and competitions, this paper is motivated by the need for an automated process for detecting road rutting defects in regional Western Australia. Main Roads WA presented the problem of only having one survey vehicle with the necessary sensors and camera technologies to monitor roads under its management. Due to a lacking fleet of survey vehicles to deploy onto its extensive road network, Main Roads WA presented us with the task of using machine learning technologies on dashcam footage to detect road rutting. To facilitate the study, they provided a dataset containing dashcam footage over seven highways and the ground truth rutting values recorded by sensors mounted on the front of the vehicle in a spreadsheet. The next section will describe the specifications of the dataset.

Dashcam Footage

The dataset contains dashcam footage from what seems to be an off-shelf camera mounted on the front of the vehicle. The footage covers different stretches of roads with subtitles describing a coordinate system (described in Straight Line Kilometres, or SLK) for that part of the road. There are 54 videos in total with over 9 hours of footage at five frames a second, equating to 163,690 frames (example of a frame along with corresponding subtitle shown below). Note that not all frames have subtitles, they are mostly only missing at the beginning and end of some videos. Not all videos have unique frames either, i.e. some videos have repeated subsets of coordinates covered in others. There are several irregularities in the video dataset that were noticed, all of which will be described in later sections of this paper. Examples of video frames shown below.

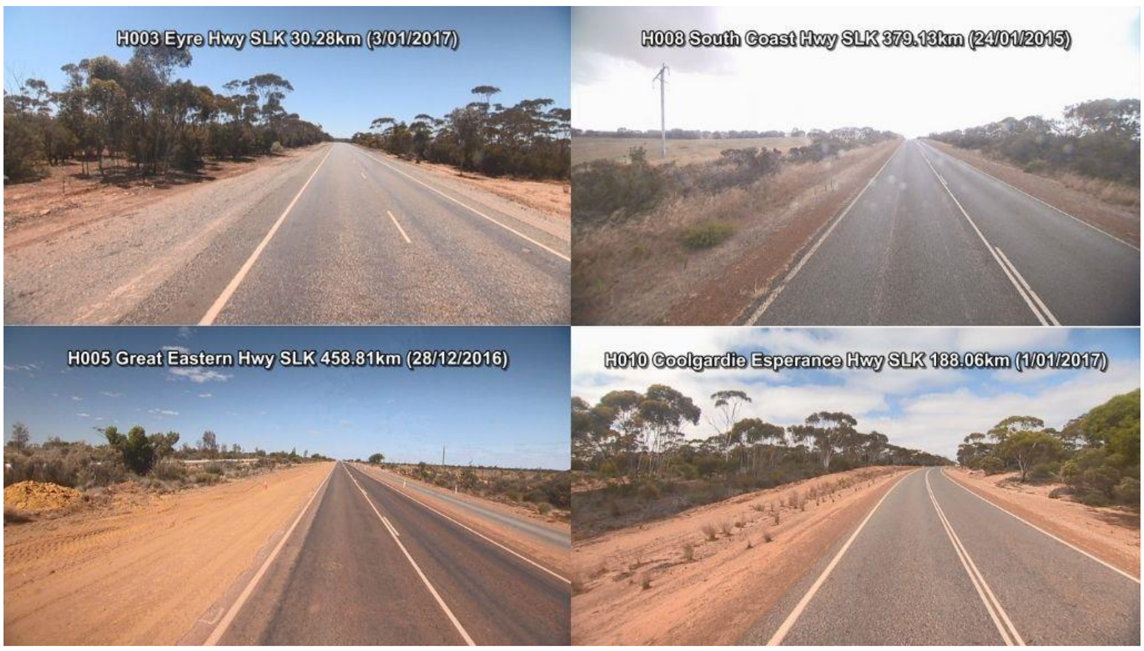

H003 Eyre Hwy SLK 30.28km (3/01/2017)
H008 South Coast Hwy SLK 379.13km (24/01/2015)
H005 Great Eastern Hwy SLK 458.81km (28/12/2016)
H010 Coolgardie Esperance Hwy SLK 188.06km (1/01/2017)

Figure 2 Example of video frames with subtitles (road number, road name, SLK coordinate, survey date)

Ground Truth

Along with the video footage, the dataset also contains a spreadsheet with ground truth rutting values corresponding to the same coordinate system used in the videos. Each entry contains a road coordinate and two corresponding rutting values (one for the inner wheel path and one for the outer wheel path) for each left and right lane (among other information not relevant to the rutting task).

There are 230,096 entries in the spreadsheet (much more than the number of video frames available). The rutting values are given as the average rut depth computed using the straight-edge model, where the length of the straight edge is set to 2 metres. It is unclear what the unit of measurement is, but it is assumed to be in millimetres. The range of rutting depth is between 0.7mm-42.8mm.

Data Imbalance

One of the challenges with the available dataset is the extreme imbalance of good roads and bad roads. For the sake of simplicity, the rest of the paper will consider the rutting values to be binned into 'good' (<15mm) and 'bad' (>15mm). 15mm has been advised by Main Roads WA as a reasonable rutting threshold to distinguish good and bad roads. To this end, there are 226,422 good roads and 3,674 bad roads in the ground truth spreadsheet. That means there are fewer than 1.5% of bad road entries. This number is considerably lower if we consider the fact that not all entries have a corresponding video frame. Given the extremely imbalanced dataset, we will need to employ various mitigation techniques to prevent overfitting when we train a machine learning model. More on this in the methodology.

Methodology

The methodology employed in this study follows the pipeline described in Figure 3 below.

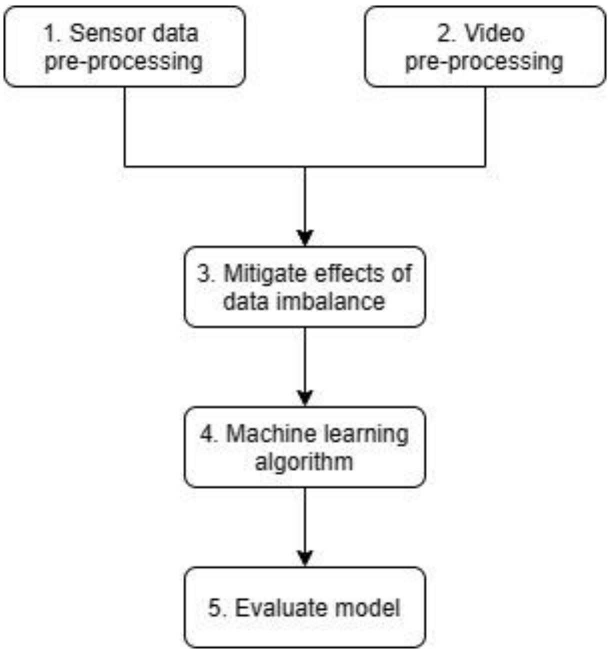

1. Sensor data pre-processing
2. Video pre-processing
3. Mitigate effects of data imbalance
4. Machine learning algorithm
5. Evaluate model

Figure 3 Pipeline of steps in methodology

1. Sensor data pre-processing

For sake of simplicity, instead of having two rutting values for each coordinate lane (i.e. inner wheel path and outer wheel path), we compute and assign one value per coordinate per lane by taking the maximum of the two. See Figure 4 for an illustration. We select only relevant columns from the spreadsheet to what is shown in Table 1.

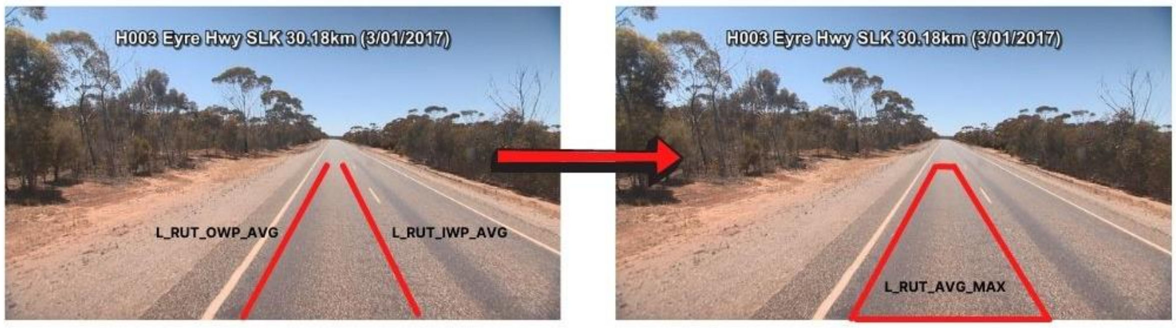

H003 Eyre Hwy SLK 30.18km (3/01/2017)
L_RUT_OWP_AVG
L_RUT_IWP_AVG
H003 Eyre Hwy SLK 30.18km (3/01/2017)
L_RUT_AVG_MAX

Figure 4 Compute and assign one maximum rutting value for left lane

Processed column name	Column name	Description
ROAD_NO	ROAD_NO	Road number
START_SLK	START_SLK	Start straight line kilometre
END_SLK	END_SLK	End straight line kilometre
L_RUT_AVG_MAX	L_RUT_OWP_AVG	The average rut depth of the outer wheelpath of the left lane using a straight edge model set to 2 metres.
	L_RUT_IWP_AVG	The average rut depth of the inner wheelpath of the left lane using a straight edge model set to 2 metres.
R_RUT_AVG_MAX	R_RUT_OWP_AVG	The average rut depth of the outer wheelpath of the right lane using a straight edge model set to 2 metres.
	R_RUT_IWP_AVG	The average rut depth of the inner wheelpath of the right lane using a straight edge model set to 2 metres.

Table 1 Relevant and processed columns from dataset

2. Video pre-processing

The video processing step contains its own pipeline, illustrated in Figure 5.

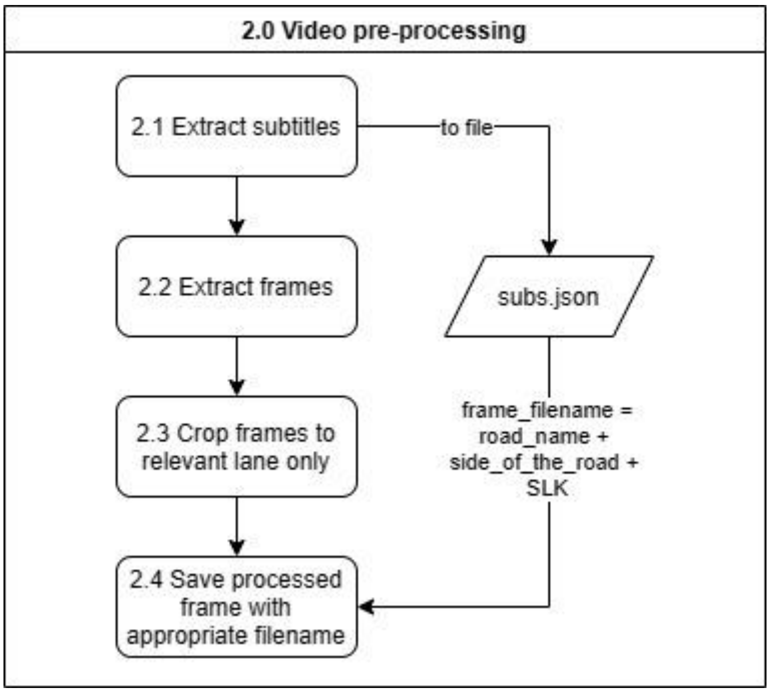

2.0 Video pre-processing
2.1 Extract subtitles
to file
2.2 Extract frames
subs.json
2.3 Crop frames to relevant lane only
frame_filename = road_name + side_of_the_road + SLK
2.4 Save processed frame with appropriate filename

Figure 5 Pipeline of steps for video pre-processing

2.1 Extract subtitles

The subtitles in the video contain information on the coordinate SLK value of each frame, as seen in Figure 4. These subtitles were extracted and saved to a JSON file for later use in naming the frames. Open-source tools like MKVToolNix were used to do the extraction.

2.2 Extract frames

Using the popular OpenCV library in Python3.6, we extract each frame and pass onto the next step in the pipeline before saving it to file.

2.3 Crop frame to relevant lane only

This step has its own sub-processes, illustrated in Figure 6.

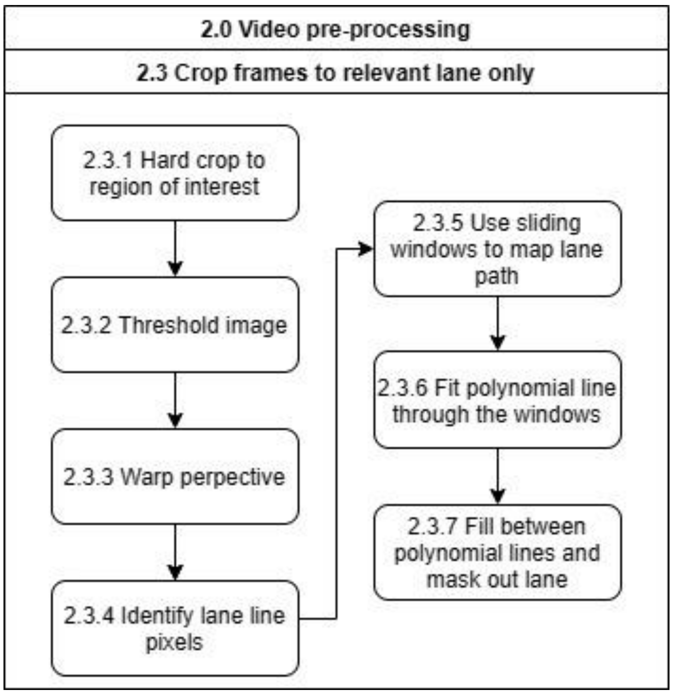

2.0 Video pre-processing
2.3 Crop frames to relevant lane only
2.3.1 Hard crop to region of interest
2.3.2 Threshold image
2.3.3 Warp perpective
2.3.4 Identify lane line pixels
2.3.5 Use sliding windows to map lane path
2.3.6 Fit polynomial line through the windows
2.3.7 Fill between polynomial lines and mask out lane

Figure 6 Pipeline of steps for cropping frame to relevant lane only

2.3.1 Hard crop to region of interest

The first step in processing the frame is to remove all the irrelevant parts in the image like the trees and the sky. This is to prevent the machine learning model from learning features in the image that it shouldn't. To do this, we perform a hard crop on the image with pre-set coordinates that roughly mask out the road. The frames are 1024×576 pixels in size, and our crop is defined by a polygon whose vertices are: bottom-left = [*cols* × 25%, *rows* × 100%], top-left = [*cols* × 50%, *rows* × 45%], bottom-right = [*cols* × 95%, *rows* × 100%], and top-right = [*cols* × 70%, *rows* × 45%], where *cols* and *rows* are the number of columns and rows in the matrix representation of the original image. The crop is implemented as a mask on the original image, filling the rest of the image in black.

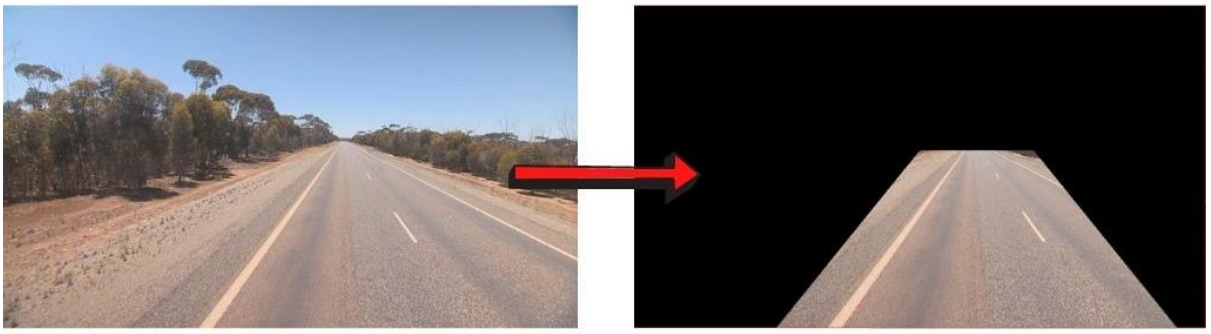


Figure 7 Hard crop on image to mask the region of interest

2.3.2 Threshold image

The next step first involves converting the image from an RGB colour space to HSL. The HSL color space is better than the RGB color space for detecting image issues due to changing lighting conditions such as shadows, glare from the sun, headlights, etc. We want to eliminate all influences that distract from detecting lane lines. For this reason, we use the HSL color space, which divides all colors into hue, saturation, and lightness values.

Then we perform a Sobel Edge detection on the lightness channel of the image to detect sharp discontinuities in the pixel intesities along the x- and y-axis. A sharp change in luminance from one pixel to adjacent pixels indicates a likely edge. We wish to isolate probable lane line edges by detecting the image's strongest edges.

Then, we perform binary thresholding on the saturation channel of the image. A high saturation value means the colour is pure. We expect lane lines to be pure colors and have high saturation, such as solid white and solid yellow. Applying binary thresholding converts an image from colour to one that is filled with 0 (black) and 255 (white) intensity values. Pixels with high saturation values will be set to white (we chose >80 on a scale of 0-255 as our threshold), while everything else will be set to black.

Next, to isolate the yellow and white colour values (which are typical hues found in lane lines), we perform binary thresholding on the red channel of the hard-cropped RGB image. And since pure white has RGB value (255,255,255) and pure red is (255,255,0), both have high red channel values. So, to generate the binary image, we set our red channel threshold with >120 to be white, and all other pixels set to black.

In our last step, we perform a bitwise AND operation between our two binary images to achieve a final image with thresholding to show pixels that have high saturation and high intensity of white or yellow colours. The resulting image is shown in Figure 8.

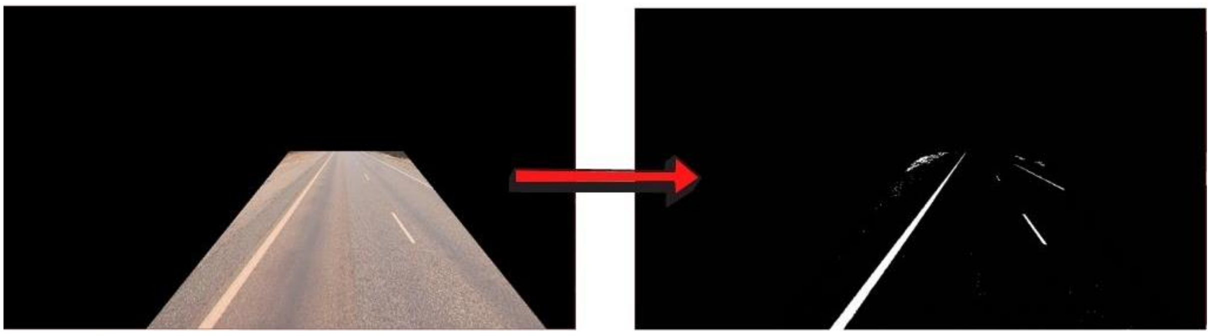


Figure 8 Thresholding image based on high saturation and high red channel values

2.3.3 Warp perspective

Not all roads are straight and have vanishing points in the centre of the image. Many roads have curvatures and must be accounted for if we want to mask out the lane properly. To achieve this, we first warp the perspective of the dashcam footage to a bird’s eye view. We do this by specifying a polygon region of interest, whose vertices we use to warp from a trapezoidal perspective to a rectangular perspective. It is now visually easier to identify the curvature of the road in the next couple of steps.

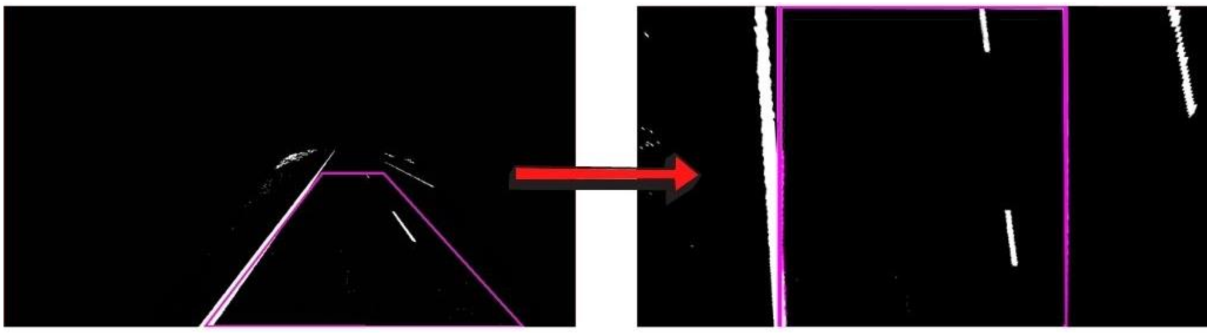


Figure 9 Warp perspective of threshold image to bird's eye view

2.3.4 Identify lane-line pixels

The continuous white pixels on some range of x-coordinates in the warped image represent lane lines. We can map areas of the image that have high concentrations of white pixels using a histogram in Figure 10. Ideally, there will a left and right peak, on either side of the vertical centre of the image. These peaks correspond to the lane lines.

After we calculate the peaks, we record the x-coordinate of the peaks and pass them to the next step of the process – sliding windows for white pixel detection.

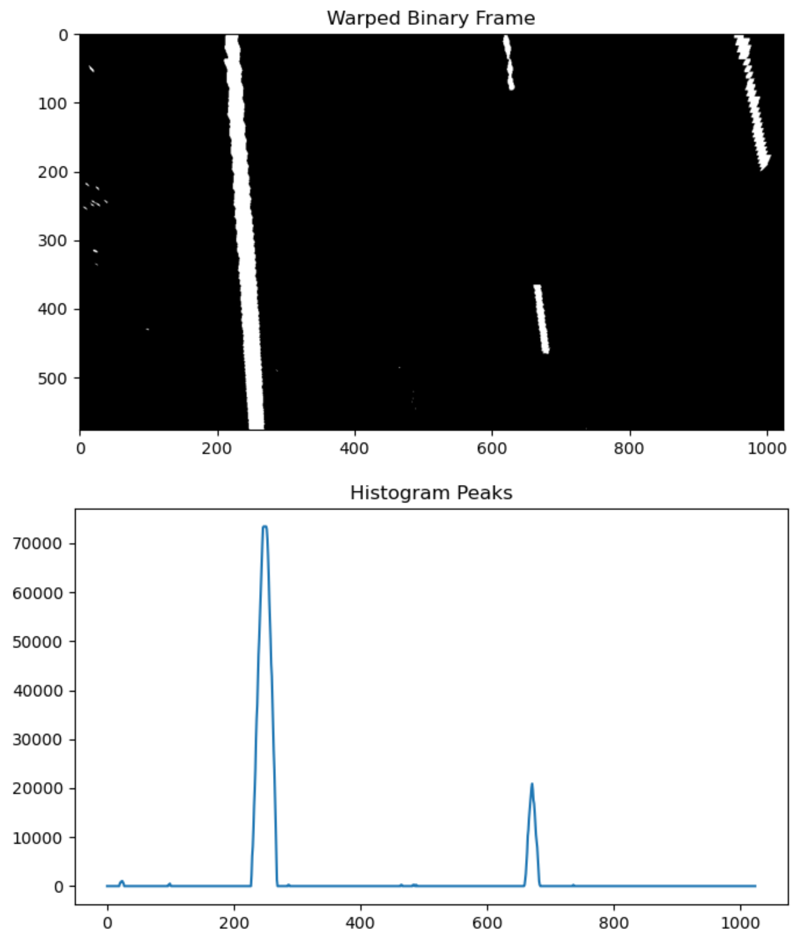


Figure 10 Histogram of white pixels in warped threshold image

2.3.5 Use sliding windows to map lane path

The next step is to use a sliding window technique where we start at the bottom of the image at the x-coordinates of the histogram peaks computed in the previous step. We scan all the way to the top of the image. Each time we search within a sliding window, we add all white coloured pixels within it (denoting probable lane lines) to a list. The mean position of the white pixels in a window becomes the centre of the following sliding window if there are enough of them (i.e. at least some fraction of the width of the image, we chose 1/25th).

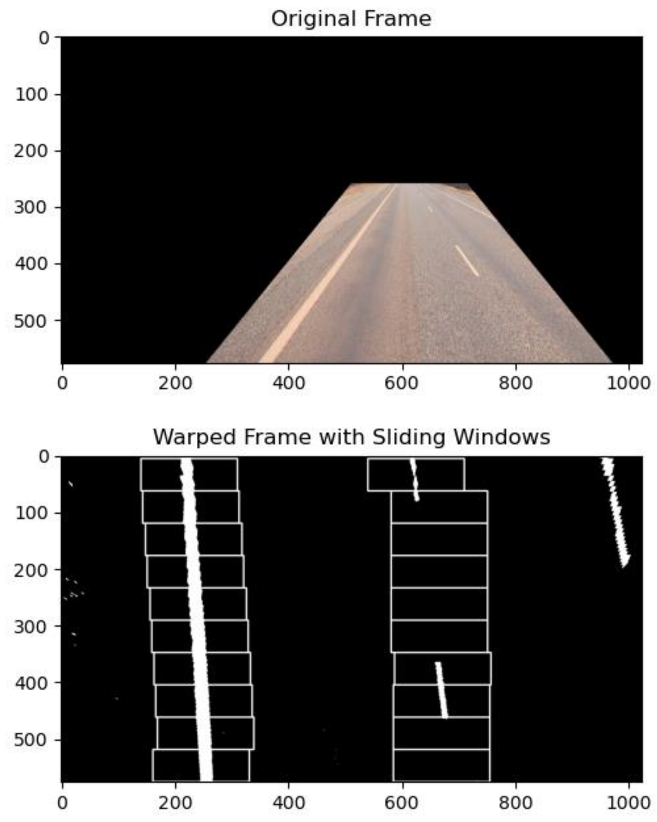


Figure 11 Sliding windows over warped threshold image to detect white pixels

2.3.6 Fit polynomial line through windows

Once we have identified the pixels that correspond to the left and right lane lines, we draw a polynomial best-fit line through the centre of each sliding window. The polynomials represent our best guesses for lane lines. If a good polynomial fit cannot be made, then we use the same mask as the previous frame, as it is likely that the orientation of the lane may not have changed much over just one frame.

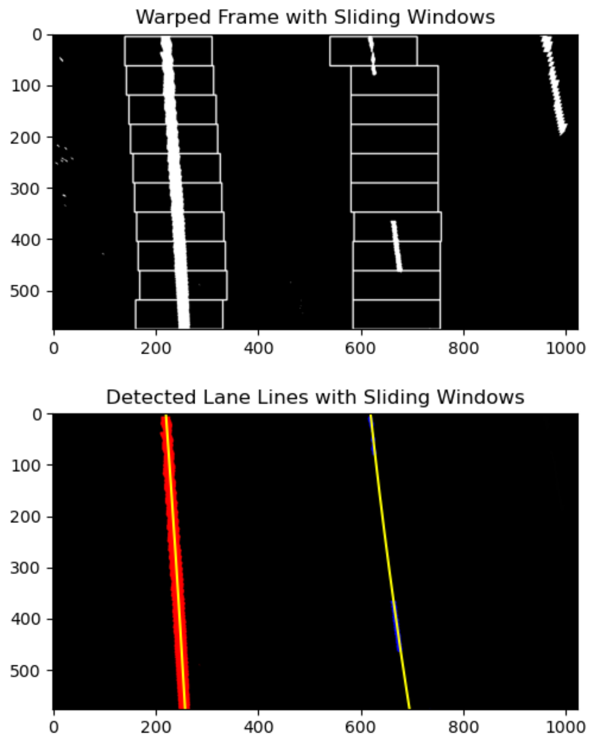


Figure 12 Fit polynomial line through the centre points of every sliding window

2.3.7 Fill between polynomial lines

Lastly, we fill in a polygon overlay that fits between the polynomial lines. This overlay is then finally used to mask out the lane in Figure 13, and the final cropped and masked image shown in Figure 14.

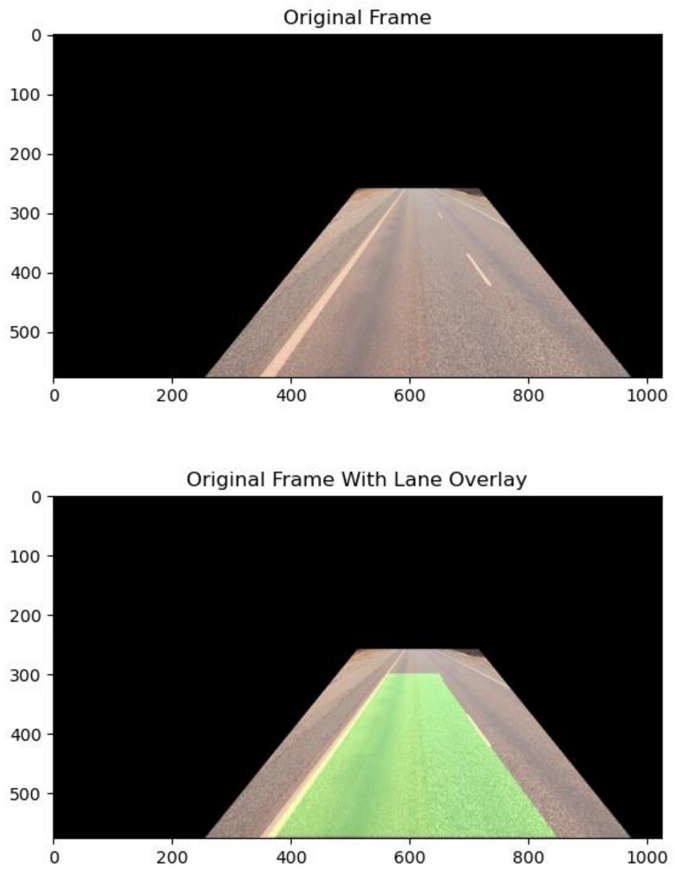


Figure 13 Polygon overlay between polynomial lines representing lane lines

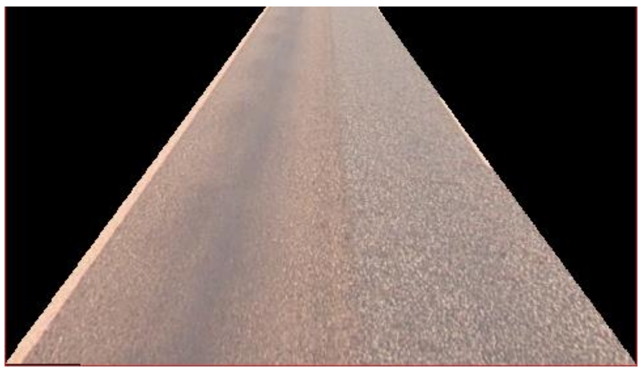
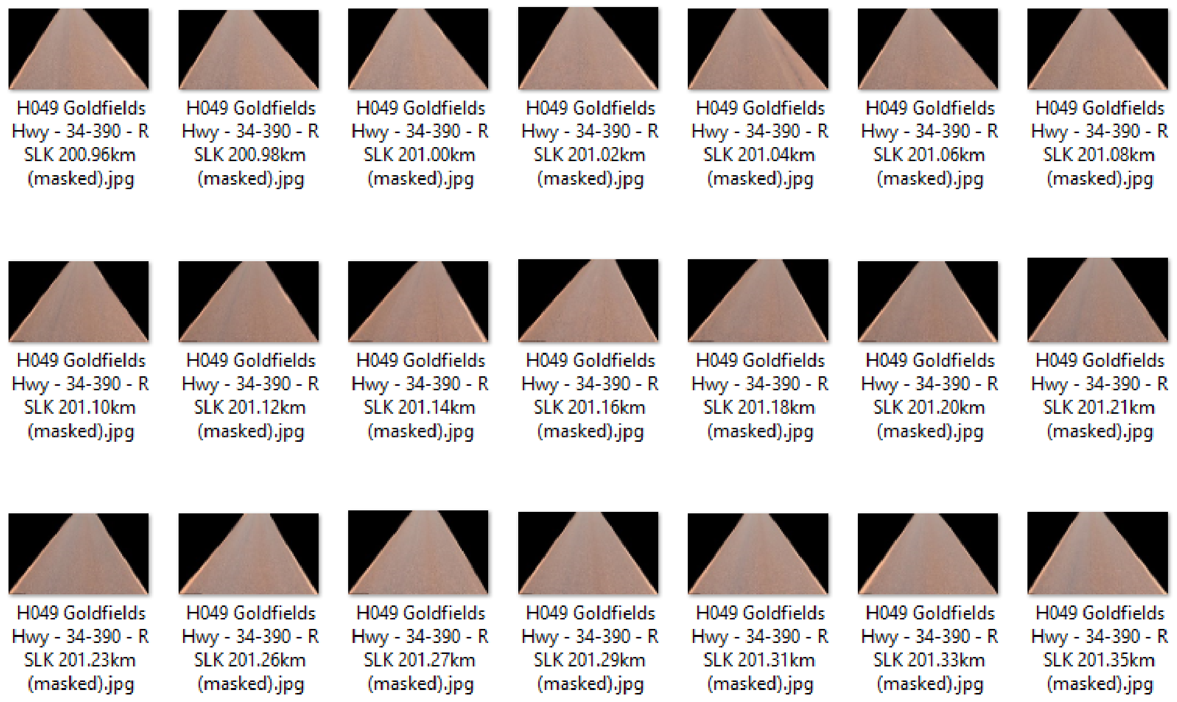


Figure 14 Final masked and cropped lane

2.4 Save processed frame with appropriate filename

The last step of the image pre-processing pipeline is to use the extracted subtitles JSON file to match the frame with its corresponding subtitle that describes the road number, road name, whether left (L) or right (R) lane, and its SLK coordinate.



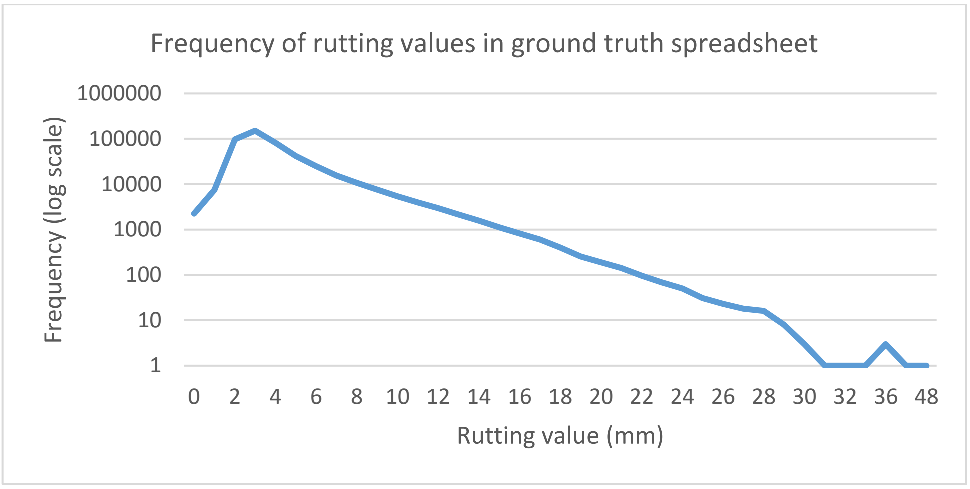
H049 Goldfields Hwy - 34-390 - R SLK 200.96km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 200.98km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.00km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.02km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.04km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.06km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.08km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.10km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.12km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.14km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.16km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.18km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.20km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.21km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.23km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.26km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.27km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.29km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.31km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.33km (masked).jpg
H049 Goldfields Hwy - 34-390 - R SLK 201.35km (masked).jpg

Figure 15 Examples of saved pre-processed images with correct identifier filenames

3. Mitigate effects of data imbalance

As explained earlier, the dataset is extremely unbalanced, which is far from ideal for a machine learning task. Therefore, some mitigation strategies have been implemented.

The distribution of rutting values is shown in Figure 16. Recollecting the threshold of 15mm for classifying the severity of rutting, less than 1.5% of roads are considered bad. We will therefore need to undersample the good roads. To do this, we choose to remove roads with rutting in range 2.9mm-14.9mm, so that we are ignoring the middle chunk of good roads and only consider the extreme ends of rutting severity. By making the good and bad roads as distinct as possible, we will be able to better train a machine learning model to learn the features of what makes a very good or very bad road.



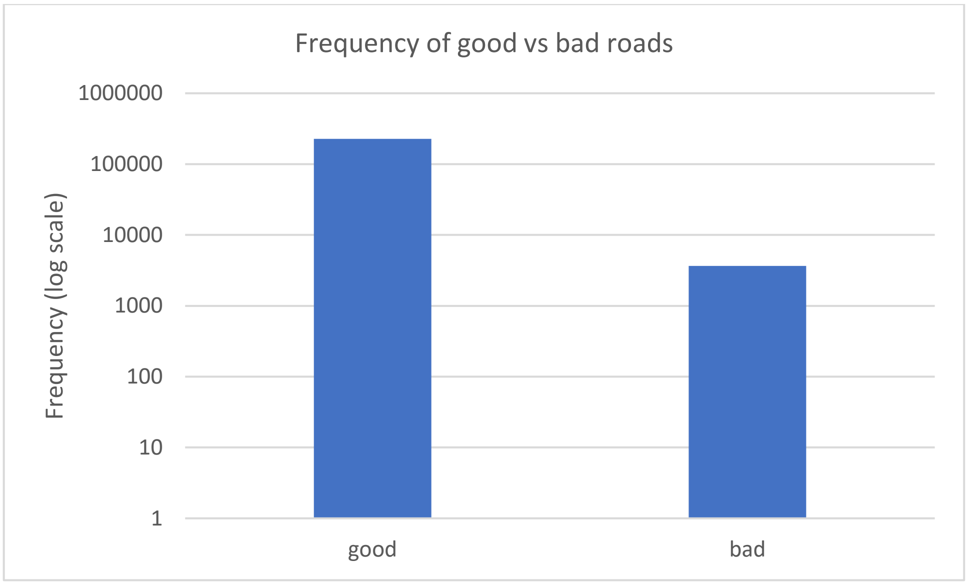


Figure 16 Imbalanced distribution of rutting values in the Main Roads WA dataset

Additionally, not all entries in the spreadsheet have corresponding frames. Plus, there were several anomalies in the image pre-processing step which were manually removed (see section on limitations of methodology for more details). So, after considering the undersampling, the missing frames, and the removed anomalies, the final distribution of our dataset is shown in Figure 17, where we now have 15% bad roads.

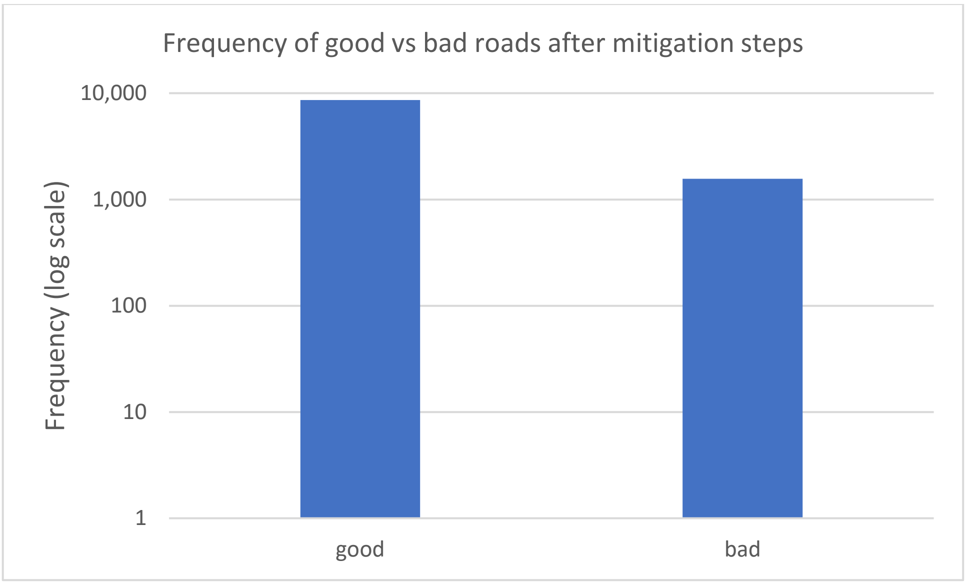


Figure 17 Distribution of good vs bad roads after data imbalance mitigation steps

The sampling method and binary class binning helped, but it is still highly imbalanced. But, we can modify the current training algorithm to consider the skewed distribution of the classes. This can be achieved by giving different weights to both the majority and minority classes. The difference in weights will influence the classification of the classes during the training phase. The purpose is to penalize the misclassification of the minority class by giving it a higher class-weight than the majority class. We compute the class-weight as follows:

$$w_{class} = \frac{total\ number\ of\ samples}{number\ of\ classes\ \times\ number\ of\ samples_{class}}$$

$$w_{good} = \frac{10{,}184}{2\ \times\ 8{,}614} \sim 0.59$$

$$w_{bad} = \frac{10{,}184}{2\ \times\ 1{,}570} \sim 3.24$$

4. Machine Learning

The aim of this paper is to report on the effectiveness of using state-of-the-art machine learning models in the rutting detection task. For this reason, we have chosen VGG16 for its advantages in architecture uniformity, ResNet152 for its significantly deep network, and MobileNet for its moderately shallow and less compute intensive architecture. For a benchmark baseline, we also use a simple CNN model with 10 layers. All models have been implemented using pre-packaged Keras applications.

	SimpleCNN	**VGG16**	**ResNet152**	**MobileNet**
No. of layers	10	16	152	28
No. of params	0.05M	134M	60M	4.2M
Unique characteristic	Simple and small model, baseline benchmark	Uniform network architecture	Very deep network with moderate no. of params	Specifically designed for devices with low computing power

Due to the lack of local hardware with a GPU, we used the free tier of Google Collaboratory for our machine learning tasks. Even still, GPU usage is restricted to one active instance and requires interactive use (i.e. may time-out if no user activity detected within roughly 90 minutes). The types of GPUs that are available and amount of memory allocated in Collaboratory vary over time. The GPUs available in Collaboratory include Nvidia K80s, T4s, P4s and P100s. There is no way to choose what type of GPU you can connect to in Collaboratory at any given time.

Due to these reasons, the following parameter decisions were made.

- Input image to be resized to 224×224 pixel dimensions to save on memory and reduce feature space
- Number of epochs limited to 10, due to restricted runtime limits
- Loss function set to binary cross entropy
- Optimizer set to Adam with default learning rate of 0.001

All models used the same train, validation, and test sets split in ratio 80:10:10.

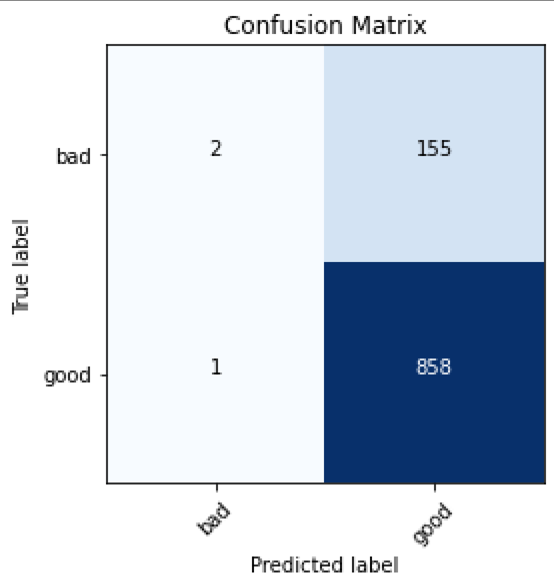
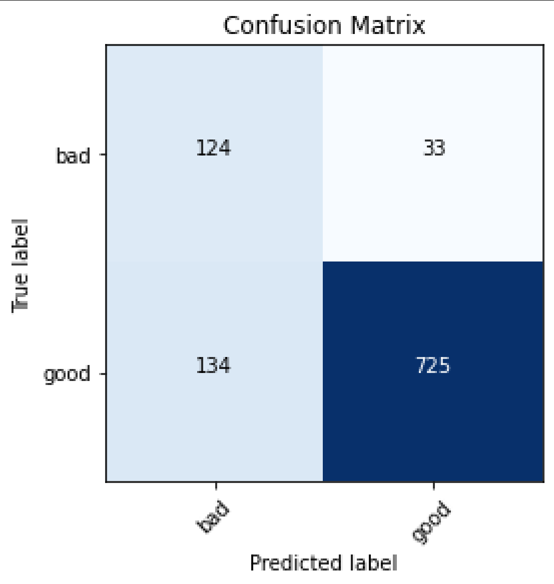
5. Results – Model Evaluation

Classification report

Model	**Accuracy**	**Class**	**Precision**	**Recall**	**F1-score**
Simple CNN	0.85	Good	0.85	1.00	0.92
		Bad	0.67	0.01	0.03
VGG16	0.85	Good	0.96	0.84	0.90
		Bad	0.48	0.79	0.60
ResNet152	0.85	Good	0.93	0.70	0.80
		Bad	0.31	0.73	0.43
MobileNet	0.84	Good	0.96	0.84	0.90
		Bad	0.48	0.83	0.61

Table 2 Classification report of Simple CNN, VGG16, ResNet152, and MobileNet on the Main Roads WA dataset

Confusion Matrix

Simple CNN	**VGG16**
	

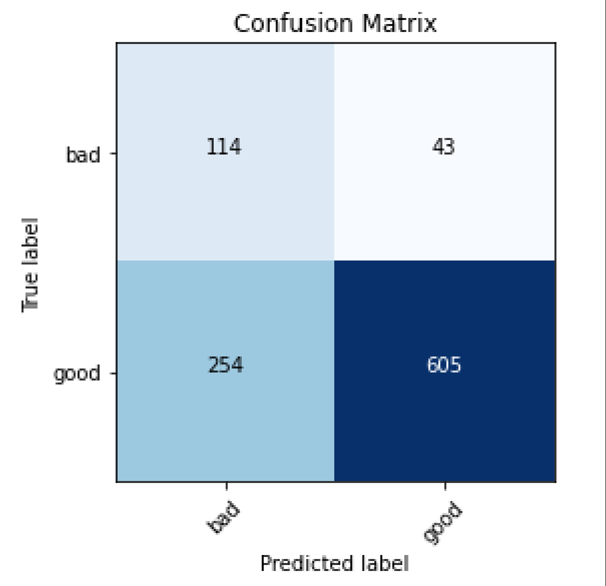
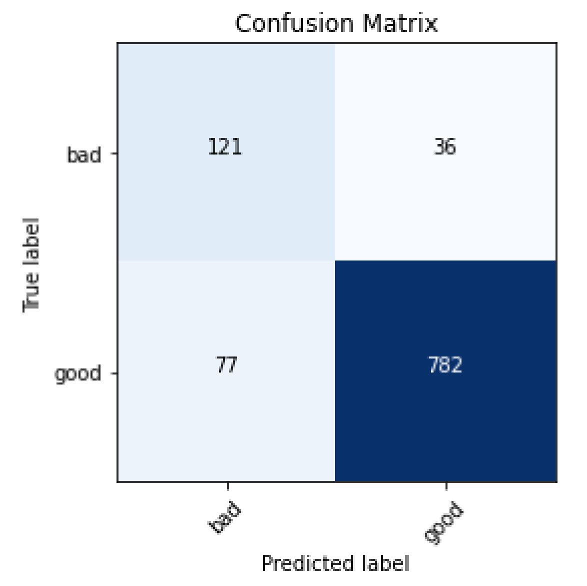
ResNet152	MobileNet
	

Table 3 Confusion matrices for Simple CNN, VGG16, ResNet152, and MobileNet on the Main Roads WA dataset

ROC Curve

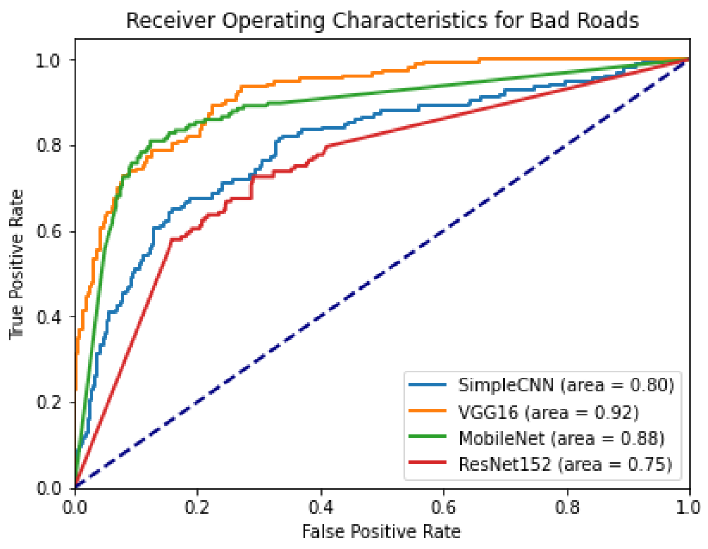


Figure 18 ROC curves for bad roads

Precision-Recall Curve

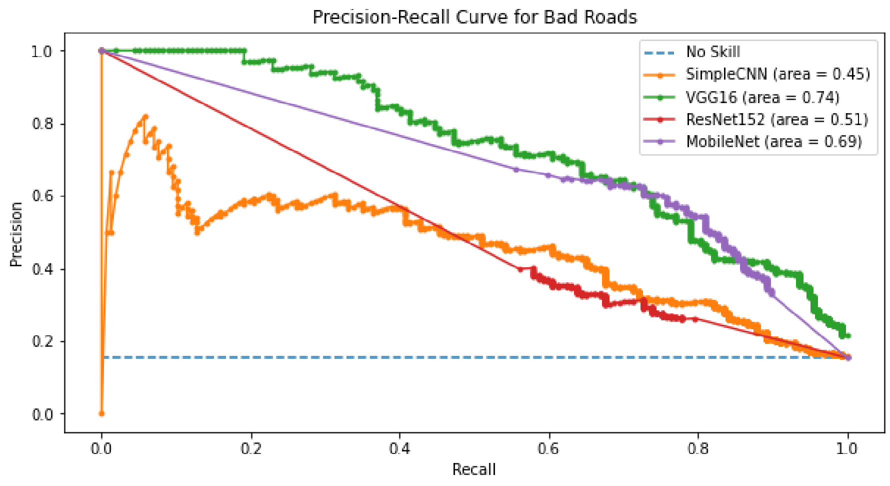


Figure 19 Precision-recall curve for bad roads

Discussion

How well did the models do?

Accuracy, precision, and recall

All models achieved more than 84% in accuracy. However, this is because the model was predicting most of the good roads correctly, and since they are the overwhelming majority class, accuracy is deceivingly high. We need to evaluate the models on other metrics.

We are more interested in predictions of bad roads. Precision helps makes sure what is predicted as a bad road is in fact a bad road. Whereas recall helps makes sure we are not missing out on any bad roads. The Simple CNN model fared the worst with only 1% recall. MobileNet performed better than VGG16 and ResNet152 by 4% and 10% respectively in the recall score for bad roads with 83%. In terms of precision, the Simple CNN scored the highest with 67%, but this is because it predicted only three instances as a bad road. So its precision score is meaningless. Among the others, VGG16 and MobileNet tied with 48%, but ResNet152 did not fare equally with only 31% precision for bad roads. Based on the precision and recall, the evaluation indicates that models that have are moderately deep and having fewer parameters perform better than models with many layers or significantly more parameters.

Confusion matrices

The confusion matrices are easy to understand visual representations of the model predictions against the ground truth. The Simple CNN performed poorly, which predicted only three roads as bad (and one of which was an incorrect prediction). VGG16 and MobileNet incorrectly predicted 33 and 36 true bad roads as good roads respectively, out of a total of 157 true bad roads. ResNet152 scored worse with 43 incorrect predictions in the same quadrant. ResNet152 surprisingly also had significantly higher incorrect predictions of true good roads as bad roads. It predicted the greatest number of bad roads, and is the only model to get more than twice as many bad road predictions incorrectly than correctly. This is more evidence suggesting that network depth and finer feature learning may not be suitable for road rutting detection tasks.

ROC curve

Two diagnostic tools that help in the interpretation of probabilistic forecast for binary classification problems are ROC (Receiver Operating Characteristic) curves and Precision-Recall curves. ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds. The areas under the curve (AUC) of the ROC curve can be used as a summary of the model skill. A skilful model will, on average, assign a higher probability to a randomly chosen true positive label than a negative label. This is what we mean when we say that the model has skill. Generally, skilful models are represented by curves that bow up to the top left of the plot. A no-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases, and is represented by a diagonal line from the bottom left to the top right of the ROC plot and has an AUC of 0.5. The two axes, True Positive Rate (TPR) and False Positive Rate (FPR) are calculated as follows.

$$FPR = \frac{FP}{TN + FP}$$

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

Since we are interested in predictions of bad roads, Figure 18 shows the ROC curves for bad roads for all our models. According to the curves, VGG16 is the most skilful with the highest AUC of 92%. At lower probability thresholds, it performs similarly to MobileNet. However, at greater thresholds, VGG16 has greater true positive rates. If we consider which model bows to the top left corner the most, i.e. which model has a prediction threshold that lowest false positive rate and the highest true positive rate, then MobileNet triumphs, with VGG16 in a close second. The other models are significantly far apart from these two on the ROC plot.

Although, SimpleCNN performed better with a greater AUC than ResNet152, this is an example of how ROC curves can present an overly optimistic view of a model's performance if there is a large skew in the class distribution. In our imbalanced Main Roads WA dataset, we have a high number of true negatives (good roads). So, the x-axis in the ROC curve (False Positive Rate), that has True Negatives in its denominator, will be a very small number, which pushes the ROC curves to the left side and misleadingly increase the ROC AUC value. Additionally, the reason SimpleCNN outperformed ResNet152 is because it only predicted three bad roads in total and only one of them was incorrect. So consequently, scored highly on the true positive rates. Hence, using ROC curves for evaluating our models, albeit intuitive, can be deceptive and lead to incorrect interpretations of the model skills. Therefore we turn to the precision-recall curve in the next section.

Precision-recall curve

Precision-recall (PR) curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds. Unlike ROC curves, typically, the most skilful model is the one that bows towards the top right of the PR plot. PR curves does not use True Negatives and hence is a suitable technique for evaluating models trained on imbalanced datasets. The equations to compute precision and recall are shown below.

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

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

According to the PR curves, VGG16 scored the highest AUC with 74%, followed by MobileNet with 69%. SimpleCNN once again performed better than the ResNet152, which continues to be misleading due the very few positive predictions for bad roads made by SimpleCNN, and consequently skewing the curve higher up the PR space. We expected the results to be different to the ROC curve, given the advantage of PR curves not being susceptible to misleading conclusions to class imbalances. However, the ranking of models based on AUC remains unchanged as that reported by the ROC curve – (1) VGG16, (2) MobileNet, (3) SimpleCNN, and (4) ResNet152.

Limitations to the methodology

Robustness of Image Pre-processing

The methodology pipeline presented in this paper has several limitations, even before we get to the machine learning step. Firstly, the lane masking process is not robust against changing lighting conditions, presence of other straight white line anomalies, heavy shadows, or if the lane lines are faded or covered in dirt. Sometimes the lane lines are completely missing. Other times, depending on the time of day it the frame was captured, the overall colour and hue of the lane lines are much more orange than white, due to the changing shine of sunlight. Because of these changing conditions, the histogram peaks do not always correspond to concentration of white pixels from lane lines, or the sliding windows go on a tangent because it detected white pixels on overexposed parts of the road caused by direct sunlight. And in some cases, the initial hard-cropped image did not mask the road well when there is severe curvature on the road. Some examples of poorly masked images are shown in Figure 20. Nevertheless, manual attempts were made to ignore such outputs to the best of abilities.

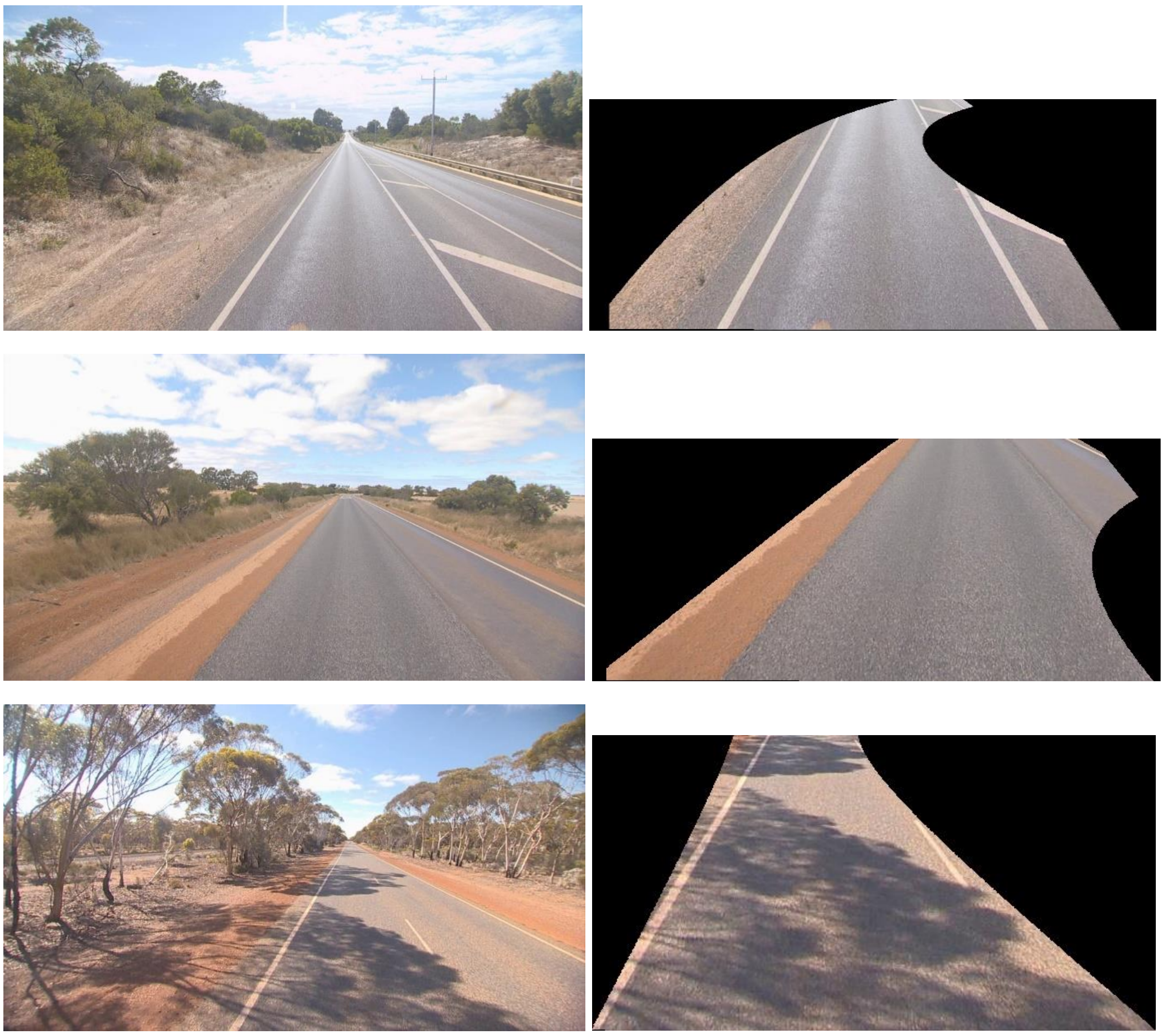


Figure 20 Examples of poorly masked road lane images; the first pair shows the influence of concentration of white pixels due to overexposed parts of the image, the second shows the absence of lane lines, and the third shows the influence of heavy shadows.

Lack of hyperparameter tuning and limited computing power

The methodology did not consider tuning the machine learning models. Several model parameters were trained with default values like the optimization function and the learning rate. The input images were also compressed into 224×224 images, which not only reduced the available resolution but also changed the aspect ratio of the original 1024×576 pixel dimensions. The models used in this study were implemented using pre-packaged Keras applications, which are optimized for the ImageNet dataset (hence the image size compressions). Ideally, we would want to input higher resolution images, and therefore would want the input layers to accept the full original image dimensions. The models were also only trained for 10 epochs due to the limited usage limits on Google Collaboratory. This also limited the prospect of using more sophisticated methods of fine-tuning like grid-searching for optimum class weights, learning rates, steps per epoch, and optimization functions.

Inconsistencies in the dataset

Besides the limitations to the methodology, the dataset itself has several inconsistencies that negatively influence the accuracy of the ground truth. The observed inconsistencies have been described below.

Inconsistent repeat frames

Several videos cover the same road coordinates. However, not all the frames with the same coordinates match location wise. What looks like a location in one frame, does not look the same in the other, even if they share the same SLK coordinate. An example of this is shown in Figure 21. Given that rutting values are reported every 20 metres, and that only area of roads closest to the camera have high resolution, even a margin of error of a few metres can lead to incorrect labelling.

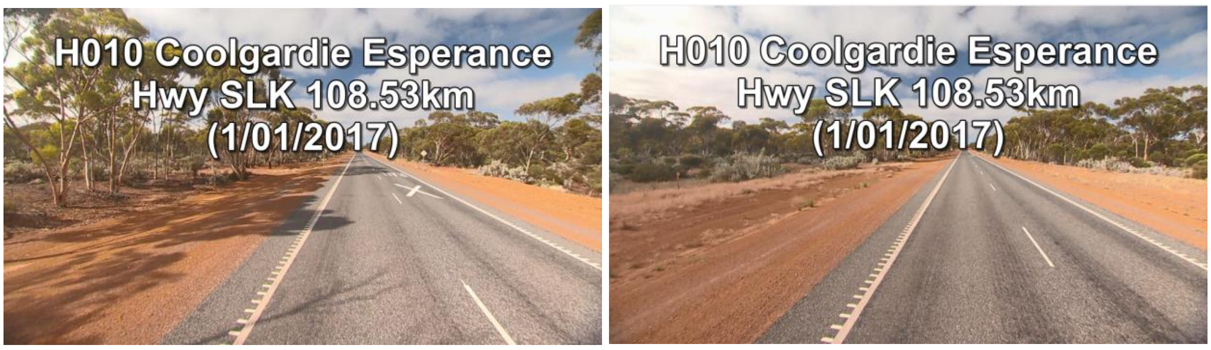
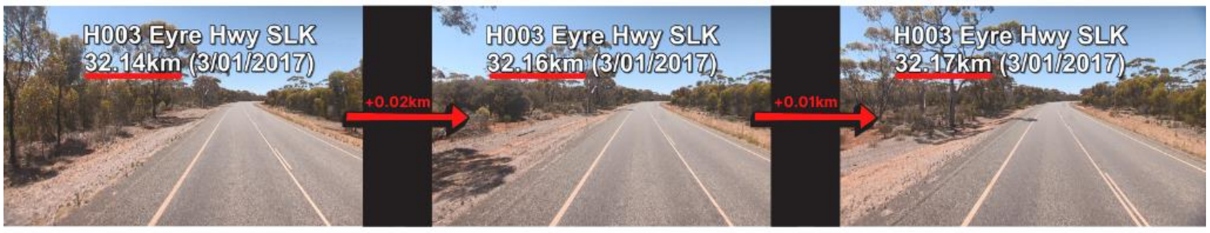

H010 Coolgardie Esperance Hwy SLK 108.53km (1/01/2017)
H010 Coolgardie Esperance Hwy SLK 108.53km (1/01/2017)

Figure 21 Frames from different videos with the same SLK do not match in location

Recording intervals

The spreadsheet reports rutting values in even-numbered 20 metre SLK intervals. However, in the videos, the SLK often skips in and out 10 metres, which desyncs the matching with the spreadsheet record. An example is shown in Figure 22. The example shows how the spreadsheet continued to record data every 0.02 SLK, but one of consecutive video frames in this example skip only 10 metres, desyncing every other frame that comes after it.

And since the rutting depth values observed in the spreadsheet are not smooth rise and falls, it is possible to have many incorrect labels of good and bad roads.


H003 Eyre Hwy SLK 32.14km (3/01/2017)
+0.02km
H003 Eyre Hwy SLK 32.16km (3/01/2017)
+0.01km
H003 Eyre Hwy SLK 32.17km (3/01/2017)

ROAD_NO	START_SLK	END_SLK	L_RUT_AVG_MAX	R_RUT_AVG_MAX
H003	32.12	32.14	7.5	6.5
H003	32.14	32.16	7	9.8
H003	32.16	32.18	3.8	9.7
H003	32.18	32.20	2.7	4.9

Figure 22 Desynced SLKs between spreadsheet and video frames

Camera orientation and video capturing technique

The camera is affixed to the front of the survey vehicle facing the direction of travel. However, this is not the most ideal method to capture images of road conditions. Due to the forward-facing orientation, only the part of the road closest to the camera has high resolution. This means that the pixel information for learning features about the road condition is limited to a small part of the image only, as annotated in Figure 23. The higher up the image, the lower the quality. Therefore, since it is unknown how the rutting value is computed (i.e. whether it is a single point measure, or an average of 20 metres), it is unclear which part of the image we should train the machine learning models on.

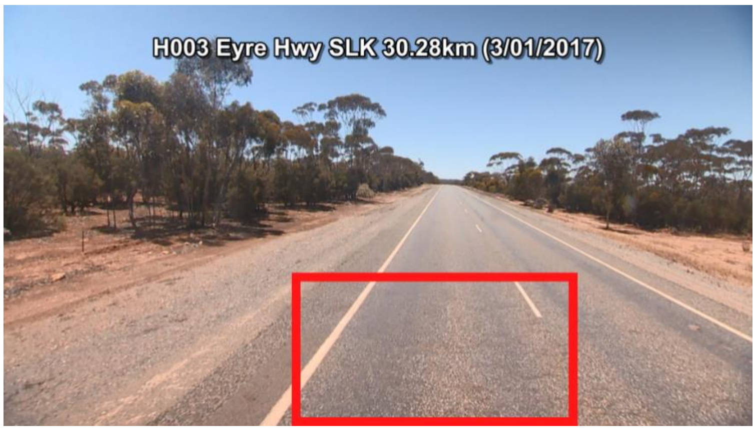


Figure 23 Area of road with highest resolution in a video frame

Conclusion

This paper presented a methodology to use ground truth road rutting depth values along with corresponding video footage of the road to classify rutting severity. The dataset, provided by Main Roads WA, is highly imbalanced with very few examples of bad roads (less than 1.5%). The suggested methodology pipeline involved extracting image frames from video footage and masking only the relevant road lane areas to feed into machine learning models. In this paper, we attempted to train one baseline model and three state-of-the-art models: a simple CNN, VGG16, ResNet152, and MobileNet. To mitigate the effects of class imbalance, we used heavy undersampling techniques to remove a large number of instances from the majority class that had medium rutting values. This also helped keep the good and bad roads as extremely different as possible, making learning rutting severity feature easier during training. The models were evaluated on different metrics with each drawing to similar conclusions. The classification reports and confusion matrices suggest that the best performing model was MobileNet and SimpleCNN the worst, whereas the ROC and PR curves suggest VGG16 as the best and ResNet152 the worst. Since we are more concerned that we do not miss predicting any potential bad roads, recall is the metric that we want to maximize. MobileNet performed the best with 83% recall for bad roads.

However, evaluating on imbalanced datasets is deceiving as there is just not enough data of the positive class to test on. The issue is further aggravated with limitations to the lane masking methodology, where several images had to be discarded to the poor robustness of the masking lanes algorithm, and with limitations to the dataset, where there exist several inconsistencies in the recorded data.

With continued dialogue with Main Roads WA about recommendations for surveying methods, there is much scope to solve this problem of automating the detection of rutting in roads. Further investigation into this problem can be facilitated if Main Roads WA are able to provide further data from future surveys, using techniques informed by the needs of computer vision and machine learning. This includes orienting the camera down towards the surface of the road with an affixed light source that will light up the captured frame evenly, mitigating the effects of sunlight. This would remove the need for a robust lane detection algorithm and provide visual data that need very little manipulation. Another recommendation is to seek consultation from a subject matter expert to learn features of the road, if not the rutting itself, that might suggest the severity of rutting (e.g. heaving on the edge of the road caused by lateral forces from heavy vehicles that also cause rutting). Apart from the suggestions for data collection, it is recommended that the machine learning process be fine-tuned to suit the task and to run for longer epochs. This will allow shifting the optimizations of the models from the ImageNet dataset towards the Main Roads WA dataset. To help further mitigate the effects of class imbalance, it is suggested to also incorporate oversampling and data augmentation techniques. Since roads have curvatures in their paths, augmentation based on slight rotations in the image might be a suitable fit.

References

[1] W. Paterson, "Road deterioration and maintenance effects : models for planning and management," 10083, 1987. Accessed: Apr. 14, 2021. [Online]. Available: http://documents.worldbank.org/curated/en/222951468765265396/Road-deterioration-and-maintenance-effects-models-for-planning-and-management.

[2] R. Wix, I. Espada, A. Rooke, and T. Martin, "Guide to asset management - technical information part 15: technical supplements," Sydney, Jul. 2018. Accessed: Apr. 14, 2021. [Online]. Available: https://austroads.com.au/publications/asset-management/agam15.

[3] J. D. Roberts and T. C. Martin, "Recommendations for monitoring pavement performance : national strategic research project," ARRB Transport Research Ltd, Vermont South, Victoria, 1996.

[4] T. Henning, R. Dunn, S. Costello, and C. Parkman, "A new approach for modelling rutting on the New Zealand State Highways," *Road Transp. Res.*, vol. 18, Mar. 2009.

[5] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. 13-es, Dec. 2006, doi: 10.1145/1177352.1177355.

[6] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11. Institute of Electrical and Electronics Engineers Inc., pp. 3212–3232, Nov. 01, 2019, doi: 10.1109/TNNLS.2018.2876865.

[7] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. 2019.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Sep. 2014, Accessed: Oct. 13, 2021. [Online]. Available: https://arxiv.org/abs/1409.1556v6.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 770–778, Dec. 2015, Accessed: Oct. 14, 2021. [Online]. Available: https://arxiv.org/abs/1512.03385v1.

[11] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise Separable Convolutions for Neural Machine Translation," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, Jun. 2017, Accessed: Oct. 13, 2021. [Online]. Available: https://arxiv.org/abs/1706.03059v2.

[12] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, Accessed: Oct. 13, 2021. [Online]. Available: https://arxiv.org/abs/1704.04861v1.

[13] D. Arya *et al.*, "Global road damage detection: State-of-the-art solutions," in *2020 IEEE International Conference on Big Data (Big Data)*, Nov. 2020, pp. 5533–5539, doi: 10.1109/bigdata50022.2020.9377790.

[14] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 12, pp. 1127–1141, Dec. 2018, doi: 10.1111/mice.12387.

[15] A. A. Angulo, J. A. Vega-Fernández, L. M. Aguilar-Lobo, S. Natraj, and G. Ochoa-Ruiz, "Road Damage Detection Acquisition System based on Deep Neural Networks for Physical Asset Management," in *Mexican International Conference on Artificial Intelligence*, Sep.

2019, pp. 3–14, Accessed: Apr. 16, 2021. [Online]. Available: http://arxiv.org/abs/1909.08991.

[16] R. Roberts, G. Giancontieri, L. Inzerillo, and G. Di Mino, “Towards Low-Cost Pavement Condition Health Monitoring and Analysis Using Deep Learning,” *Appl. Sci.*, vol. 10, no. 1, p. 319, Jan. 2020, doi: 10.3390/app10010319.

[17] A. Tedeschi and F. Benedetto, “A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices,” *Adv. Eng. Informatics*, vol. 32, pp. 11–25, Apr. 2017, doi: 10.1016/j.aei.2016.12.004.

[18] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, “Road crack detection using deep convolutional neural network,” in *Proceedings - International Conference on Image Processing, ICIP*, Aug. 2016, vol. 2016-August, pp. 3708–3712, doi: 10.1109/ICIP.2016.7533052.

[19] Z. Fan, S. Member, Y. Wu, J. Lu, and W. Li, “Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network.” Accessed: Apr. 16, 2021. [Online]. Available: https://arxiv.org/abs/1802.02208.

[20] H. Majidifard, Y. Adu-Gyamfi, and W. G. Buttlar, “Deep machine learning approach to develop a new asphalt pavement condition index,” *Constr. Build. Mater.*, vol. 247, p. 118513, Jun. 2020, doi: 10.1016/j.conbuildmat.2020.118513.

[21] M. I. Rajab, “Profiling of external deformation in asphalt pavement rutting using image analysis method,” in *Proceedings - 2013 IEEE 9th International Colloquium on Signal Processing and its Applications, CSPA 2013*, 2013, pp. 99–102, doi: 10.1109/CSPA.2013.6530022.