An Thi Nguyen

# ALPACA: Asphalt Quality Prediction with Automated Computer Vision Assessment

Master's thesis in Computer Science

Supervisor: Pinar Øzturk

July 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

An Thi Nguyen

# ALPACA: Asphalt Quality Prediction with Automated Computer Vision Assessment

Master’s thesis in Computer Science
Supervisor: Pinar Øzturk
July 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

**An Thi Nguyen**

# ALPACA: Asphalt Quality Prediction with Automated Computer Vision Assessment

Master's Thesis, Spring 2020

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

TDT4900 - Computer Science, Master's Thesis
Main supervisor: Pinar Øzturk
Co-supervisor: Leendert Wienhofen

# Abstract

Roads are susceptible to pavement distress including cracks and potholes, and an uneven surface is a potential hazard to vehicles. Manual road inspection has the disadvantage of being time-consuming, hazardous due to traffic and weather, subjective and dependent on the experience of the personnel. This thesis aims to assist road maintainers in prioritising interventions and performing preventive maintenance.

We propose a pipeline for asphalt quality prediction with automated computer vision assessment, called the *ALPACA* pipeline, which uses time series forecasting to predict future states of pavement distress from smartphone videos of roads in Trondheim. The ALPACA pipeline is able to run on a conventional 8GB GPU, making this technology accessible on limited budgets. Furthermore, we build an annotated dataset for object detection of pavement distress in urban environments containing challenging seasonal and lightning variations, as well as an annotated dataset for semantic segmentation of pavement distress in smartphone images. Finally, we present experimental results evaluating each stage of our pipeline. Despite our pipeline being far from ready to be used in a real setting, our work offers some insight into how various computer vision, image processing and time series forecasting methods can be integrated to forecast pavement quality from smartphone videos or images.

# Preface

I would like to thank Maria Soleim for helping me install R, Ingvild Høgstøyl for HPC support, my mom and dad, and Yalda Akhgar, Levi Sørum, Bård Flugon and Erlend Ravlo for supporting me immensely throughout this semester. I would also like to thank Trondheim municipality and Trondheim Beta for providing me with the Jetson TX2 to play around with.

Last but not least, I would like to thank my wonderful supervisor and co-supervisor, Pinar Øzturk and Leendert Wienhofen, not only for their patience, helpful suggestions and encouraging me to work on this project even without prior experience in computer vision, but most of all for our road trip to the alpaca farm.

An Thi Nguyen
Trondheim, August 1, 2020

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Roads are susceptible to pavement distress including cracks and potholes, and an uneven surface is a potential hazard to vehicles. Manual road inspection has the disadvantage of being time-consuming, hazardous due to traffic and weather, subjective and dependent on the experience of the personnel. Using computer vision to automate the road inspection process may alleviate these issues. Although prior work has been done on pothole and crack detection from images [38] [3] [43] [26] [6] [28], few studies have utilised this information to forecast when a crack or pothole will reach an unacceptable condition. One reason for this might be a lack of models able to accurately distinguish between a variety of pavement distress types. In this master's thesis, we will evaluate a variety of possible solutions for predicting future severity of pavement distress using various computer vision and time series forecasting methods.

This thesis was an initiative from Trondheim municipality with the goal of making technology accessible for the public, even with limited budgets. For computer vision, this means enabling the models to run on an affordable GPU. Furthermore, we explore the possibility of using a standard smartphone for data collection. We have developed a pipeline which is able to run on a conventional 8GB GPU, and released two annotated datasets for public use [1].

[1] The datasets will be available on Github: `github.com/AlpakkAn/pavement-distress-datasets`

## 1.1 Goals and Research Questions

The overall goal of this thesis is:

**Goal** *Develop a pipeline to predict future states of pavement distress from videos of roads in Trondheim.*

Our goal is to predict future states of pavement distress in order to assist road maintainers in prioritising which roads to fix. The first stages of the pipeline should be able to accurately localise and classify pavement distress in videos, as well as quantify distresses. The last stage should use this quantification of pavement distress at various points in time to predict future states of pavement distress. We achieve our goal through answering the following set of research questions:

**Research question 1** *Which existing computer vision and image processing methods are suitable for detecting pavement distress in this application?*

**Research question 2** *How can pavement distress be quantified?*

By *pavement distress detection*, we mean the task of localising and classifying pavement distress in images [2]. By quantification of pavement distress, we mean determining its severity. As the pipeline should be able to be reused by municipalities with limited budgets, we require methods that can detect and quantify objects on an affordable GPU such as an NVIDIA Jetson TX2. Additionally, we require methods that are able to extract information from low-resolution smartphone images.

**Research question 3** *Which approach is suitable for predicting future states of pavement distress?*

**Research question 4** *Which features can be used for predicting future states of pavement distress in Trondheim?*

We require an approach that is able to utilise multiple time series. Furthermore, we explore the usage of various traffic and weather-related features.

[2]The literature is inconsistent in its usage of the term *pavement distress detection*, sometimes referring to the task of object detection of pavement distress, and other times to semantic segmentation of pavement distress. In this thesis, we adopt the term *pavement distress detection* to refer to the task of object detection of pavement distress, sometimes only referred to as *detection*, and the term *pavement distress segmentation* for the task of semantic segmentation of pavement distress.

## 1.2 Research Method

The research questions will be answered through three phases of this thesis. In the first phase, a structured literature review is conducted, where we review existing computer vision methods for detection and quantification of pavement distress. Furthermore, we review existing methods for pavement performance prediction. In the second phase, we select methods for developing our pipeline and test the selected detection and quantification methods on an NVIDIA Jetson TX2 to determine whether to go further with them in phase three. In the third phase, we conduct experiments around the remaining methods to see how they perform on each task of the pipeline: pavement distress detection, quantification and forecasting. The experiments are designed in such a way that each stage of the pipeline is evaluated separately.

## 1.3 Contributions

The main contributions of this thesis are as follows:

- We proposed a pipeline to predict future pavement distress severity from videos or images, which is able to run on a conventional 8GB GPU.
- We presented a quantitative evaluation of each stage of our pipeline. Furthermore, we performed an analysis of the features used for the last stage of our pipeline, pavement distress forecasting, to evaluate the importance of various features related to traffic and weather.
- We built an annotated dataset for object detection of pavement distress in urban environments containing challenging seasonal and lightning variations to be further used for training and evaluating new models. We also provided an annotated dataset for semantic segmentation of pavement distress in low-resolution smartphone images. We made these datasets openly available.

## 1.4 Thesis Structure

In the next chapter, Chapter 2, we cover the necessary background on different types of pavement distress and their causes of formation, as well as give a brief overview of the fields of object detection, semantic segmentation and time series forecasting. In Chapter 3, we review previous work related to our research questions. Chapter 4 presents our proposed pipeline, which is based on the findings from the prior chapter, and describes its rationale. In Chapter 5, we present the results of our experiments with the different components of our pipeline. Finally, we evaluate the experimental results with respect to our research questions, discuss the limitations of our work, and provide pointers for future work in Chapter 6.

# Chapter 2

# Background Theory

This chapter presents the background theory for this thesis. First, we cover various common types of pavement distress and their individual causes. Then, we will review different tasks in computer vision, eventually focusing on the You Only Look Once (YOLO) algorithm. Further, we briefly touch on segmentation, focusing on semantic segmentation. Finally, we describe time series forecasting.

## 2.1 Types of Pavement Distress and Causes of Formation

There exists a variety of different pavement distresses, each distress having different factors influencing its formation and life cycle. *Potholes* are depressions in the road surface, often caused by moisture weakening the underlying soil of a pavement section, thus facilitating breaking of the unsupported road surface by traffic [44]. Freeze-thaw cycles can further accelerate this process, causing frost heaving which damages pavement and allow further moisture to seep in [7]. *Alligator cracks* are a precursor to pothole formation, which develops under repeated traffic loading, causing the interconnected cracks resembling the back of an alligator seen in Figure 2.1. Alligator cracks usually start as a series of *longitudinal cracks*, cracks that occur parallel to the centerline of the pavement. Causes of longitudinal cracks also include low temperatures, daily temperature cycling, and poor lane joint construction [ast]. In the latter case, the resulting cracks often occur in the middle of two lanes, sometimes referred to as *lane longitudinal cracks* [28]. *Transverse cracks* occur perpendicular to the centerline of the pavement, and their causes are similar to those of longitudinal cracks. Both longitudinal and transverse cracks can also be caused by *reflective cracks*, which are again caused by various cracks underneath the surface layer. *Block cracks* are interconnected cracks that look similar to alligator cracks, but the cracks are shaped like rectangular pieces instead of many-sided pieces. Block cracks are primarily caused by asphalt concrete contraction and daily temperature cycling [ast].

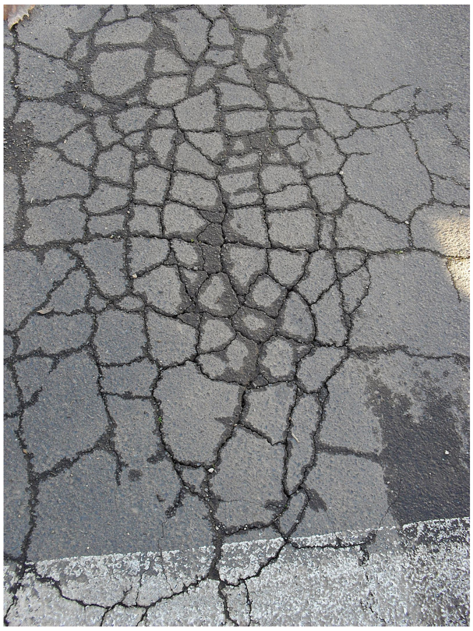


Figure 2.1: Alligator cracks containing moisture, which can be a symptom of a weakend soil underneath. "Asphalt deterioration" by Bidgee is licensed under CC BY 3.0.

## 2.2 Object Detection

Object detection is the task of predicting where objects are located and to what class each object belongs to in a given image. With the advances in hardware driving renewed interest in deep learning, Convolutional Neural Networks (CNNs) have been increasingly leveraged for object detection. CNN-based object detectors can primarily be divided into Region Proposal-Based Frameworks and Regression/Classification-Based Frameworks [49]. Region Proposal-Based Frameworks were first introduced in 2013 through Regions with CNN features

(R-CNN) [11], later followed by Fast R-CNN [10] and Faster R-CNN [37]. The main idea of Region Proposal-Based Frameworks is to produce a set of region proposals from the input image, selecting only a few windows as opposed to iterating over all windows. Although improving upon the then-state-of-the-art, they were not able to perform real-time detection. Regression/Classification-Based Frameworks, also known as one-stage detectors, You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD) and RetinaNet being some notable ones, allow for real-time detection by enabling the bounding boxes and class probabilities to be mapped directly from the image pixels [49]. Not long ago, region proposal-based methods in general performed better than one-stage detectors in terms of accuracy [49]. Now, one-stage detectors such as RetinaNet and YOLOv4 claim to be on par with region proposal-based methods while running at faster speeds [24] [4].

For the rest of this section, we will first describe various metrics for object detection. Afterwards, we describe the incremental improvements leading to YOLOv4.

### 2.2.1 Metrics for Object Detection

Intersection over Union (IoU) is used to evaluate the localisation performance of an object detection algorithm, and is defined as:

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{2.1}$$

where A is set of pixels in the predicted bounding box and B the set of pixels in the ground truth bounding box.

If the IoU between a predicted bounding box and ground truth bounding box is above a threshold (usually 0.5, 0,75 or 0.95), the prediction is regarded as a True Positive (TP). If the IoU is below the threshold, it is a False Positive (FP). A ground truth that is not detected is regarded as a False Negative (FN).

Precision is the percentage of correct positive predictions:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall is the percentage of true positive predictions among all ground truths:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Average Precision (AP) approximates the area under the Precision x Recall curve. Before 2010, this was done in the Pascal VOC challenges by applying

11-point interpolation:

$$AP = \frac{1}{11} \sum_{r \epsilon \{0,0.1,\ldots,1\}} p_{interp}(r) \tag{2.4}$$

where

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{2.5}$$

and $p(\tilde{r})$ is the precision at recall $\tilde{r}$.

From 2010, the Pascal VOC challenges measured the AP by interpolating through all the points:

$$AP = \sum_{r=0}^{1} (r_{n+1} - r_n) p_{interp}(r_{n+1}) \tag{2.6}$$

where

$$p_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{2.7}$$

The mean Average Precision (mAP) is the mean of all APs over all classes in the dataset:

$$mAP = \frac{1}{|classes|} \sum_{c \epsilon classes} AP_c \tag{2.8}$$

### 2.2.2 YOLO

YOLO is a one-stage object detection framework aimed for real-time detection. The first version, YOLOv1, was introduced in [34], with incremental improvements later being made in YOLOv2 [35], YOLOv3 [36] and YOLOv4 [4].

YOLO divides the image into an SxS grid, where each cell in the grid predicts one object whose center lies in that grid cell, as well as predicting a fixed number of bounding boxes. For each bounding box, a box confidence score is predicted, reflecting the likelihood of the box containing an object and how accurate the bounding box is. In addition, conditional class probabilities (the probability that a detected object belongs to a class given that the grid cell contains an object) are calculated for each grid cell. To remove duplicate detections for the same object, YOLO uses non-maximal suppression (NMS), removing all generated bounding boxes that have a class probability less than a given threshold and all bounding boxes where the IoU with the highest probability bounding box are above a certain threshold. Figure 2.2 shows the YOLOv1 architecture.

Despite being suitable for real-time processing, YOLOv1 lacked in accuracy compared to SSD and region-based methods and did not detect small objects close to each other well, as each grid cell can only have one class and predict a

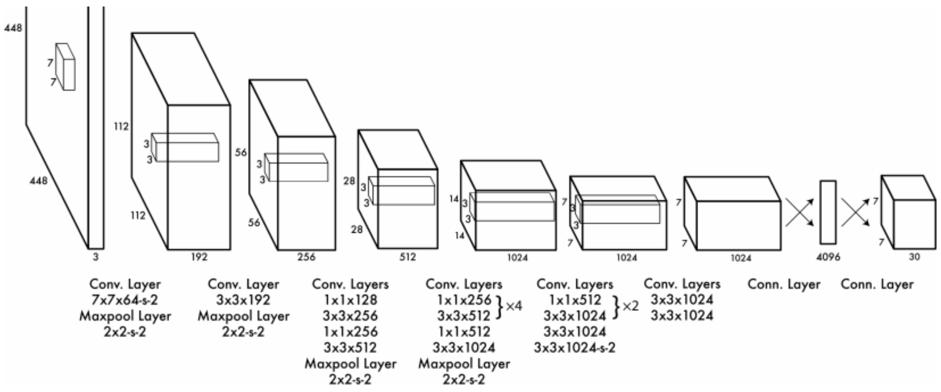
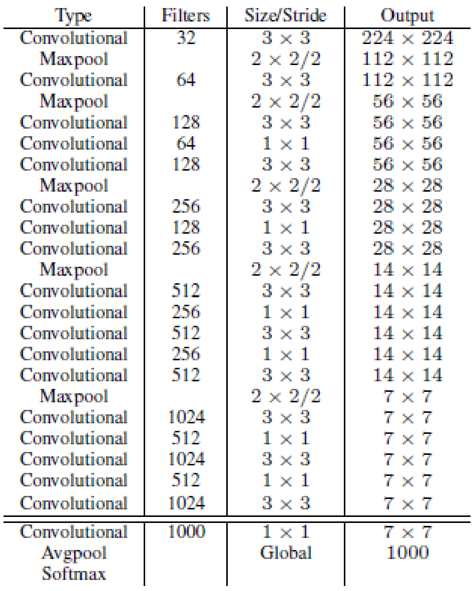


Figure 2.2: The YOLOv1 network design, consisting of 24 convolutional layers, which are followed by two fully connected layers. To reduce the feature maps from preceding layers, 1x1 reduction layers are used in some convolutional layers. The network outputs a (7, 7, 30) tensor. Figure source: Redmon et al. [34].

fixed number of bounding boxes. YOLOv2 was an improvement both in terms of accuracy and speed. There were many improvements added, one of them incorporating *anchor boxes*. By starting from a set of anchor boxes shaped like everyday objects and predicting offsets for these, initial training is more stable compared to arbitrarily initialising the bounding boxes. In addition, batch normalization and a higher-resolution 448x448 classifier was added, as well as multiple network design modifications. For instance, the last two fully connected layers were removed, and Darknet-19 was used as the backbone, the architecture of which can be seen in Figure 2.3. Additionally, class prediction was moved to the bounding box level.

Improvements in YOLOv3 increased performance significantly, making YOLOv3's performance on MS COCO on par with SSD in terms of accuracy, but three times faster. Furthermore, YOLOv3 has improved in detecting small objects. In YOLOv3, the Darknet-19 backbone is replaced by Darknet-53, shown in Figure 2.4, consisting of 53 convolutional layers. In addition, multi-label classification has been added to allow for object classes that are not mutually exclusive, i.e. "alpaca" and "animal".

YOLOv4 showed even greater improvements, claiming an accuracy of 43.5% AP for the MS COCO dataset with around 65 FPS inference speed on Tesla V100. In addition, it can be trained and used on a GPU with 8-16 GB-VRAM [4]. Some of the things that increase YOLOv4's accuracy while keeping inference speed high is its *Bag of freebies* and *Bag of specials*.

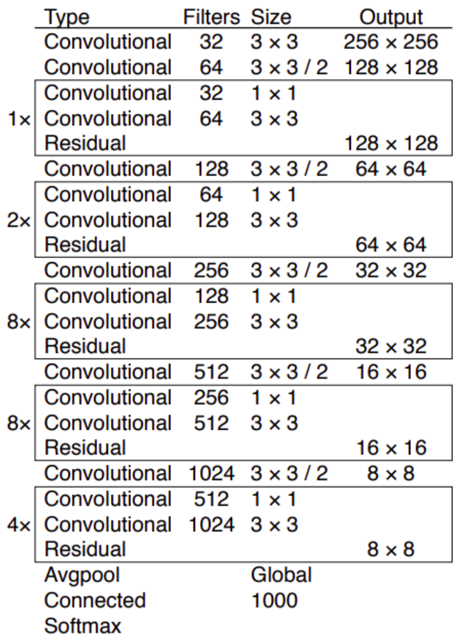


| Type | Filters | Size/Stride | Output |
|---|---|---|---|
| Convolutional | 32 | $3 \times 3$ | $224 \times 224$ |
| Maxpool | | $2 \times 2/2$ | $112 \times 112$ |
| Convolutional | 64 | $3 \times 3$ | $112 \times 112$ |
| Maxpool | | $2 \times 2/2$ | $56 \times 56$ |
| Convolutional | 128 | $3 \times 3$ | $56 \times 56$ |
| Convolutional | 64 | $1 \times 1$ | $56 \times 56$ |
| Convolutional | 128 | $3 \times 3$ | $56 \times 56$ |
| Maxpool | | $2 \times 2/2$ | $28 \times 28$ |
| Convolutional | 256 | $3 \times 3$ | $28 \times 28$ |
| Convolutional | 128 | $1 \times 1$ | $28 \times 28$ |
| Convolutional | 256 | $3 \times 3$ | $28 \times 28$ |
| Maxpool | | $2 \times 2/2$ | $14 \times 14$ |
| Convolutional | 512 | $3 \times 3$ | $14 \times 14$ |
| Convolutional | 256 | $1 \times 1$ | $14 \times 14$ |
| Convolutional | 512 | $3 \times 3$ | $14 \times 14$ |
| Convolutional | 256 | $1 \times 1$ | $14 \times 14$ |
| Convolutional | 512 | $3 \times 3$ | $14 \times 14$ |
| Maxpool | | $2 \times 2/2$ | $7 \times 7$ |
| Convolutional | 1024 | $3 \times 3$ | $7 \times 7$ |
| Convolutional | 512 | $1 \times 1$ | $7 \times 7$ |
| Convolutional | 1024 | $3 \times 3$ | $7 \times 7$ |
| Convolutional | 512 | $1 \times 1$ | $7 \times 7$ |
| Convolutional | 1024 | $3 \times 3$ | $7 \times 7$ |
| Convolutional | 1000 | $1 \times 1$ | $7 \times 7$ |
| Avgpool | | Global | 1000 |
| Softmax | | | |

Figure 2.3: Darknet-19 architecture. For the object detection task, the last convolutional layer is replaced by three 3x3 convolutional layers with 1024 output channels, followed by a 1x1 convolutional layer with the number of outputs needed. Figure source: Redmon and Farhadi [35].

Bag of freebies are additions during training that do not impact inference speed, of which YOLOv4 among others incorporate CutMix and Mosaic data augmentation and class label smoothing. CutMix is a data augmentation method which cuts out a region of an image, places it on top of another image and adjusts the labels accordingly, as illustrated in Figure 2.5. This prevents the model from being overconfident on some particular features while keeping the same amount of information in the image, as opposed to Cutout data augmentation where a part of the image is cut out. Mosaic data augmentation encourages objects to be detected outside their typical context by combining four images into one during training, as shown in Figure 2.6. Class label smoothing helps prevent overconfidence and overfitting by replacing the hard 1.0 upper bound classification targets with a lower value (0.9 for instance) [13].

Bag of specials lower inference speed slightly, but increase performance considerably in return, of which YOLOv4 among others utilise mish activation and DIoU-NMS. Mish is a smooth and non-monotonic activation function, and is de-



| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| | Convolutional | 32 | 1 × 1 | |
| 1× | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| | Convolutional | 64 | 1 × 1 | |
| 2× | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| | Convolutional | 128 | 1 × 1 | |
| 8× | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| | Convolutional | 256 | 1 × 1 | |
| 8× | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| | Convolutional | 512 | 1 × 1 | |
| 4× | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Figure 2.4: Darknet-53 architecture, consisting of 53 convolutional layers. Figure source: Redmon and Farhadi [36].

fined as $f(x) = x \cdot tanh(ln(1 + e^x))$. Mish is similar to the widely used activation function Swish, but outperforms both Swish and other activation functions in many deep networks across various datasets [30]. DIoU-NMS uses Distance-IoU (DIoU) as a factor when deciding to remove bounding boxes during NMS [50]. The DIoU of two bounding boxes B and $B^{gt}$ is defined as

$$R_{DIoU} = \frac{\rho^2(b, b^{gt})}{c^2}, \tag{2.9}$$

where b and $b^{gt}$ are the central points of B and $B^{gt}$, $\rho(\cdot)$ is the Euclidean distance, and c is the diagonal length of the smallest box covering the two boxes. By using the DIoU with the highest probability bounding box as a factor in bounding box suppression, DIoU-NMS helps when objects are occluded, as it also takes the central point distance between two boxes into account.

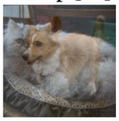
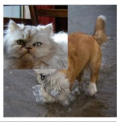
| | ResNet-50 | Mixup [48] | Cutout [3] | CutMix |
|---|---|---|---|---|
| Image | |  | |  |
| Label | Dog 1.0 | Dog 0.5 Cat 0.5 | Dog 1.0 | Dog 0.6 Cat 0.4 |
| ImageNet Cls (%) | 76.3 (+0.0) | 77.4 (+1.1) | 77.1 (+0.8) | **78.6 (+2.3)** |
| ImageNet Loc (%) | 46.3 (+0.0) | 45.8 (-0.5) | 46.7 (+0.4) | **47.3 (+1.0)** |
| Pascal VOC Det (mAP) | 75.6 (+0.0) | 73.9 (-1.7) | 75.1 (-0.5) | **76.7 (+1.1)** |

Figure 2.5: CutMix data augmentation. Figure source: Yun et al. [46].

## 2.3 Segmentation

Object detection methods cannot precisely trace the shape of an object, as they only aim to fit a bounding box around the object. Instead of predicting bounding boxes, semantic segmentation classifies each pixel into one of a fixed number of classes, creating segmentation masks of the classes. However, semantic segmentation does not distinguish between different instances of an object category. Instance segmentation, however, classifies each pixel into one of a fixed number of classes, but also determines if it makes up a different object instance.

For the rest of this section, we will first describe common metrics for semantic segmentation. Afterwards, we describe guided filtering, a method for post-processing the semantic segmentation results.

### 2.3.1 Metrics for Semantic Segmentation

Pixel accuracy measures the percentage of pixels correctly classified in an image. Although this metric is simple to understand, it does not work well for segmentation tasks with class imbalances, as a model that classifies all image pixels as the majority class will still have a high pixel accuracy.

Intersection over Union (IoU) in segmentation is, similarly to object detection, defined as

$$IoU(A, B) = \frac{A \cap B}{A \cup B}, \tag{2.10}$$

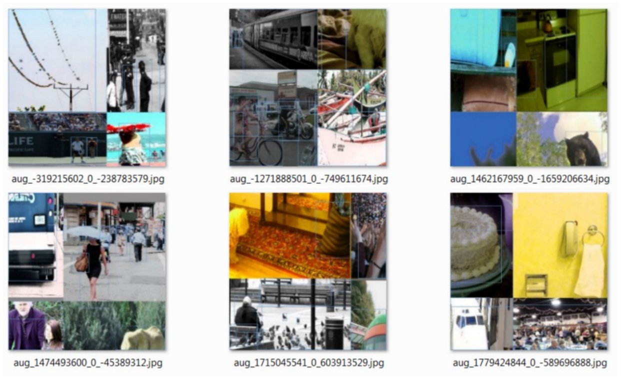

aug_-319215602_0_-238783579.jpg
aug_-1271888501_0_-749611674.jpg
aug_1462167959_0_-1659206634.jpg
aug_1474493600_0_-45389312.jpg
aug_1715045541_0_603913529.jpg
aug_1779424844_0_-589696888.jpg

Figure 2.6: Mosaic data augmentation. Figure source: Bochkovskiy et al. [4].

where A is the set of pixels in the predicted segmentation and B the set of pixels in the ground truth segmentation.

F1 score is defined as

$$F1 = \frac{2TP}{(TP + FP) + (TP + FN)} \tag{2.11}$$

### 2.3.2 Guided Filtering

Several post-processing methods can be used to refine the outputs from semantic segmentation, one of which includes guided feathering, based on guided image filtering [15]. Guided filtering is an image processing technique which performs edge-preserving smoothing on an input image by using a second image, either the input image, a different version of the input image, or an entirely different image, as a guide to perform filtering. Guided feathering performs guided filtering on a binary mask as the input image, using the original image as guidance to refine the mask edges, as illustrated in Figure 2.7.

## 2.4 Time Series Forecasting

A time series is a sequence of observations ordered in time, often decomposed into individual patterns: trend, seasonality and cycle. Trends are characterised by a long-term rise or fall in the data, seasonality by the effect of seasonal factors (i.e. weekday or time of the year), and cycle by increases and decreases in the data without any fixed frequency. An additive decomposition is written as the sum of

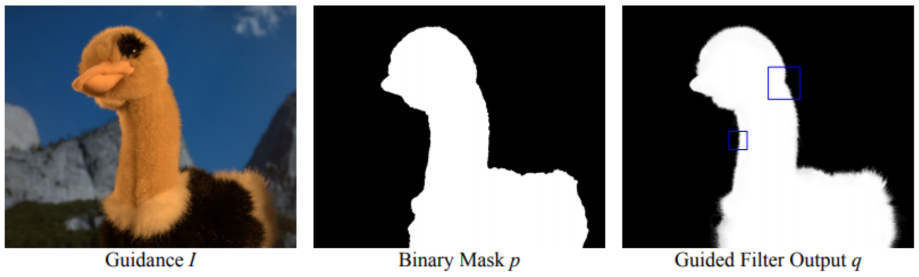


Figure 2.7: Guided feathering. A binary mask p is refined using a guidance image I. Figure source: He et al. [15].

these components, where the trend and cycle is merged into a single component:

$$y_t = S_t + T_t + R_t, \tag{2.12}$$

where $y_t$ is the data at period t, $S_t$ the seasonal component, $T_t$ the trend-cycle component, and $R_t$ the remainder component [17].

In this thesis, we consider multiple multivariate multi-step time series forecasting, meaning that we model multiple time series at the same time, each having multiple variables, as well as predicting multiple time steps into the future. In the rest of this section, we describe the Naïve method. Then we describe cross-validation. Lastly, we describe some metrics commonly used in time series forecasting.

### 2.4.1 Naïve Method

The Naïve method is one of the simplest time series forecasting methods, and is often used as a baseline. The Naïve method sets all predictions to the value of the last observation:

$$\hat{y}_{T+h|T} = y_T, \tag{2.13}$$

where $\hat{y}_{T+h|T}$ is the estimate of $y_{T+h}$ based on a series of observations $y_1, ..., y_T$.

### 2.4.2 Cross-validation

K-fold cross-validation is commonly used to evaluate machine learning models. In K-fold cross-validation, the dataset is split into K groups and training and testing is done K times, where each time a different group is used as the test set while the remaining are used for training. Standard K-fold cross-validation has been shown to fail for some time series models [33]. The approach to cross-validation often used in time series forecasting is *evaluation on a rolling forecast origin*. In this approach, the test sets consist of a single observation, while earlier observations are used as the corresponding training data, so that no future data are used in the prediction [17]. Figure 2.8 shows evaluation on a rolling forecast origin for multi-step time series.

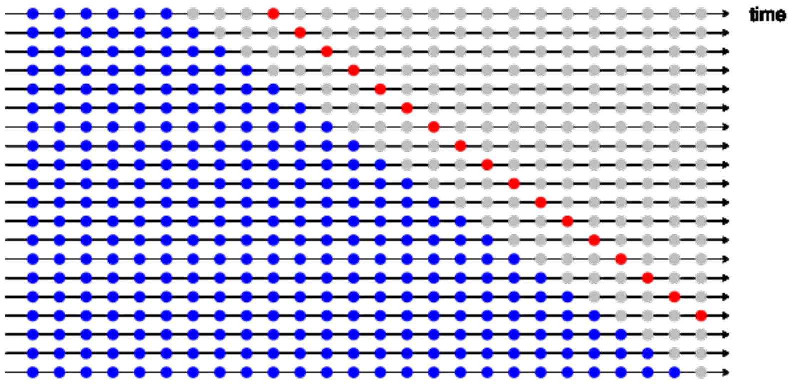


Figure 2.8: Cross-validation for multi-step time series. Figure source: Hyndman and Athanasopoulos [17].

Although K-fold cross-validation is not valid for some time series models, [2] argue that it is a valid approach for purely autoregressive models with uncorrelated errors, which includes many machine learning methods.

### 2.4.3 Metrics

The accuracy of a time series forecasting method can be measured through its forecast errors, which is the difference between a forecast and the actual value. Two commonly used scale-dependent error metrics are Mean absolute error (MAE) and Root mean squared error (RMSE):

$$MAE = mean(|e_t|), \tag{2.14}$$

$$RMSE = \sqrt{mean(e_t^2)}. \tag{2.15}$$

Although MAE is easier to interpret, RMSE is useful when it is important to penalise large errors more.

# Chapter 3

# Related work

In this chapter, we describe closely related work to our research questions. First, we describe our structured literature review protocol. Then, we present the findings from our literature review.

## 3.1 Structured Literature Review Protocol

This section describes our structured literature review protocol. First, we present the research questions guiding our literature review. Then, we describe the search terms, selection of primary studies and inclusion criteria.

### 3.1.1 Research Questions

The following research questions (RQ) were guiding for our literature review:

**RQ1** Which machine learning methods have been applied to pavement performance prediction?

**RQ2** Which features have been used in pavement performance prediction?

**RQ3** Which computer vision methods have been used to detect pavement distress?

**RQ4** Which computer vision methods have been used to quantify pavement distress?

**RQ5** Has semantic segmentation been used on pavement distress?

**RQ6** Has instance segmentation been used to detect and quantify pavement distress?

**RQ7** Have image based analysis techniques been used to quantify pavement distress for input to a pavement performance prediction model?

### 3.1.2 Identification of Research

The following two searches were conducted to answer RQ1-7. Google Scholar was searched, and at least the first five pages of each search were reviewed.

#### Search 1

RQ1, RQ2 and RQ7 was explored through the search terms in Table 3.1.

| | Group 1 | Group 2 | Group 3 | Group 4 |
|---|---|---|---|---|
| Term 1 | preventive maintenance | predict | asphalt | computer vision |
| Term 2 | crack growth | forecast | pavement | visual based |
| Term 3 | crack propagation | | road | image based |
| Term 4 | pavement performance | | | machine learning |
| Term 5 | pavement distress | | | |

Table 3.1: Search terms for RQ1 and RQ6.

#### Search 2

RQ3-6 was explored through the search terms in Table 3.2.

| | Group 1 | Group 2 | Group 3 |
|---|---|---|---|
| Term 1 | pavement | distress | quantify |
| Term 2 | asphalt | defect | semantic segmentation |
| Term 3 | road | crack | instance segmentation |
| Term 4 | | pothole | detect |
| Term 5 | | | object detection |

Table 3.2: Search terms for RQ3-6.

### 3.1.3 Selection of Primary Studies

The set of returned primary studies were limited by removing:

1. Duplicates (kept the highest ranking source),

2. The same study published in different sources (kept the highest ranking source),

3. For Search 3, studies published before 2015,

4. Studies not focusing on pavement distress in asphalt,

5. Studies using models, datasets or frameworks which are not open source.

In the following sections, we present our findings from the literature review, grouped by the tasks of distress detection and quantification, distress segmentation, and distress forecasting.

## 3.2 Distress Detection and Quantification

Most studies attempting to detect pavement distress have only focused on a small number of distress types, some detecting potholes only [38] [3], while others have focused on sealed cracks [20] [48]. [43] utilised Local Binary Pattern (LBP) cascade classifiers from the OpenCV library to classify potholes and fatigue cracks, as well as longitudinal and transverse cracks as one class. [26] detected pavement distress from smartphone images using SSD, defining six classes related to asphalt defects and two classes related to crosswalk and white line blur. However, this study defined different types of pavement distress as one class (e.g. rutting, bump, pothole and separation as one class). Although this makes it easier to achieve a high object detection accuracy, it does not take into account that different distress types have different causes and life cycles, making this approach unsuitable for our pipeline which predicts future states of pavement distress. In [6], YOLOv3 was used to detect seven distress types, and in [28], YOLOv2 and Faster R-CNN models were trained for detection of the nine distress types the authors claim affect pavement quality the most: reflective; transverse; block; longitudinal; alligator; lane longitudinal; sealed reflective; and sealed longitudinal cracks; and potholes.

We did not find any studies employing instance segmentation on pavement distress. Most previous studies using computer vision for distress detection or quantification have treated them as separate problems. [27] attempted to develop a hybrid model consisting of YOLOv2 and U-Net pretrained on retina vessel images, which detects asphalt defects and quantifies the defects' severity in parallel before merging the two results. Although this was the only study we found that attempted to detect and quantify defects simultaneously, the U-Net model was evaluated only on mean squared error (MSE), which is not commonly used as a metric for semantic segmentation, making it difficult to compare with other segmentation models. Furthermore, the study lacked a discussion of how the

U-Net model performed on pothole segmentation, as it is unclear whether there is a pothole equivalent in retina vessel segmentation.

## 3.3 Distress Segmentation

Although extensive research has been carried out on crack segmentation, we did not find any studies which account for other types of pavement distress such as potholes. Earlier attempts at crack segmentation used hand-engineered features, such as CrackForest [40], a crack segmentation framework based on structured random forest which outperformed the then-state-of-the-art crack segmentation methods. In recent times, CNNs have been utilised for crack segmentation, achieving good results. DeepCrack [25], a deep hierarchical CNN, achieved state-of-the-art performance. However, the models were only evaluated on the authors' own proposed benchmark dataset. Feature Pyramid and Hierarchical Boosting Network (FPHBN) [45] outperformed the state-of-the-art methods, including CrackForest, in terms of generalisability and accuracy on a variety of crack datasets.

## 3.4 Distress Forecasting

Although research has been carried out on computer vision for pavement distress detection and quantification, we found no studies using image based analysis techniques to quantify pavement distress for input to a model predicting future states of pavement quality. There is, however, extensive research on pavement performance prediction, prediction of various pavement performance indicators, the earlier attempts including utilising Markov chains [23]. In addition, neural networks have been used extensively [47] [21] [9] [19], including in combination with other methods such as Long Short-Term Memory [5], Support vector machine (SVM) [42] and Random forest [8]. Other pavement performance prediction methods include Gradient boosted trees and K-nearest neighbour [18], SVM [51] [39] [21] [9] and Random forest [29] [12]. However, despite the extensive research on pavement performance prediction, most of these studies only make long-term forecasts, ranging from 2 to 10-year horizons. From a road maintenance perspective, shorter forecast horizons might be more useful.

A variety of different features have been used for pavement performance prediction models, some common ones being previous pavement performance values, pavement age, pavement thickness, traffic, pavement structure and various weather-related features [12] [39] [29] [18] [8].

# Chapter 4

# Architecture

This chapter first presents a high-level overview of the proposed architecture and describes its rationale. Then, we describe in detail its individual components for detection, quantification and forecasting.

## 4.1 Overview and Rationale

Figure 4.1 provides an overview of our proposed pipeline, its three stages and their contents. We name our pipeline the *Asphalt Quality Prediction with Automated Computer Vision Assessment* (ALPACA) pipeline. The first stage of the pipeline performs object detection, predicting where pavement distress is located in a given video frame and to which of nine different distress types each distress belongs to. The second stage performs semantic segmentation of the detected distresses to quantify severity, classifying each pixel in a video frame as "crack" or "non-crack", thus generating a mask of the pavement distress. The third stage applies time series forecasting to predict future severity with a time horizon of 3, 6 and 9 weeks.

Predicting masks within bounding boxes, as is the task performed in the first and second stage of our pipeline, is usually done with an instance segmentation model such as Mask R-CNN [14]. However, as we did not find any previous work employing instance segmentation on pavement distress, detecting and quantifying pavement distress in one instance segmentation stage would require rigorous mask annotations of a variety of pavement distress types. As there are relatively large datasets available on the separate tasks of object detection and semantic segmentation of pavement distress, where the object detection dataset is expert-annotated, we choose to instead perform these tasks in two separate stages, partly inspired by [27]. Our architecture differs from [27] in respect to the semantic segmentation model used, as we use DeepCrack [25] instead of a U-Net model implemented for retina blood vessel segmentation, and YOLOv4 instead of YOLOv2 for object detection.

## 4.2 Detection

The pavement distress detection stage of our pipeline classifies pavement distress from video frames using an object detection model. We require a model that can operate on a Jetson TX2. With this in mind, we reviewed current open-source models for object detection. We choose YOLOv4 based on state-of-the-art performance on the MS COCO dataset, ease of implementation and ability to perform real-time detection on a conventional 8GB GPU [4]. The model is trained on a Tesla V100.

As the data the model will be working with when deployed consists of low-resolution smartphone images, as well as being different to the training data with respect to camera pitch angle and number of urban objects, we perform some image pre-processing steps prior to object detection, including contrast enhancement, cropping and padding. After object detection, the detected pavement distresses are cropped for input to the quantification stage.

## 4.3 Quantification

The pavement distress quantification stage of our pipeline assigns a severity level to each detected pavement distress based on distress type and pixel count of the binary mask, which is obtained using semantic segmentation. We reviewed open-source models for semantic segmentation of pavement distress, initially choosing FPHBN [45] for its state-of-the-art performance in terms of generalisability and accuracy on a variety of crack datasets. However, FPHBN ran out of memory on the Jetson TX2 even when reducing the batch size to 1. We therefore settled on DeepCrack [25], a deep hierarchical CNN achieving state-of-the-art performance on the authors' own proposed benchmark dataset. Figure 4.2 shows the DeepCrack architecture.

## 4.4 Forecasting

The pavement distress forecasting stage of our pipeline uses the assigned severity levels of detected distresses and their associated distress types, combined with traffic and weather data retrieved from the Norwegian Meteorological Institute's Frost API and the Norwegian Public Roads Administration's NVDB API, to forecast future severity levels of the detected pavement distresses. We implement this stage for two models: Random forest for its ability to perform relatively well with default hyperparameters, and XGBoost for its automatic handling of missing values, in addition to both models providing feature importances.

## 4.5 Summary

Figure 4.3 provides a summary of which models were evaluated for the three stages of our pipeline, and which of the models were selected. In the next chapters, we describe our experiments with the components of the pipeline and evaluate whether the proposed architecture can provide satisfactory answers to our research questions presented in Section 1.1.

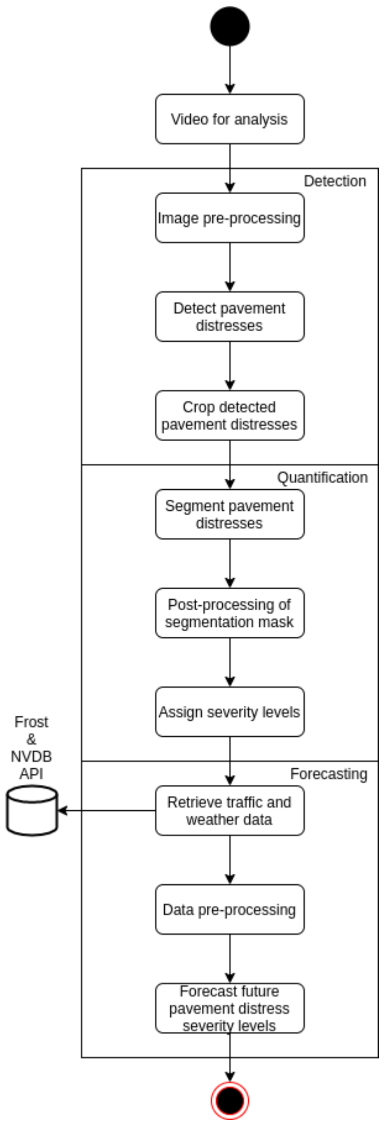

Video for analysis
Detection
Image pre-processing
Detect pavement distresses
Crop detected pavement distresses
Quantification
Segment pavement distresses
Post-processing of segmentation mask
Assign severity levels
Forecasting
Frost & NVDB API
Retrieve traffic and weather data
Data pre-processing
Forecast future pavement distress severity levels

Figure 4.1: Our proposed ALPACA pipeline and its three stages: detection, quantification and forecasting.

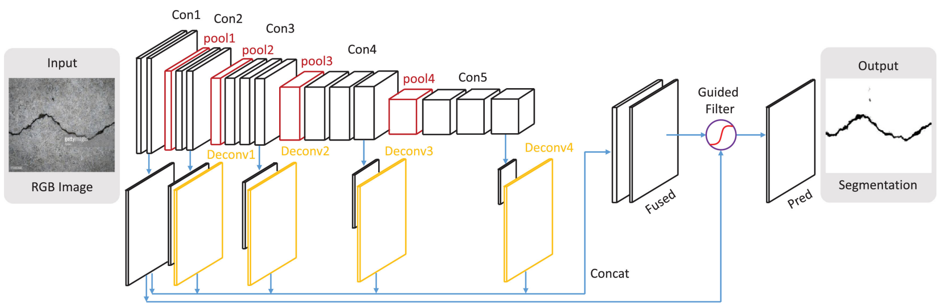


Figure 4.2: DeepCrack architecture. Figure source: Liu et al. [25]

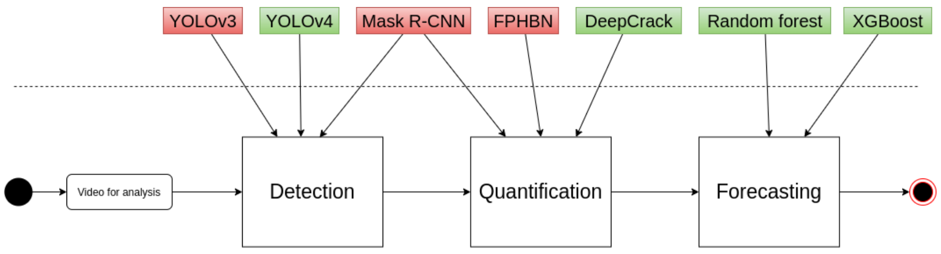


Figure 4.3: Our proposed ALPACA pipeline with the models evaluated for the three stages above the dashed line. Models marked in green were selected for the final pipeline.

# Chapter 5

# Experiments and Results

This chapter presents the experiments and results. First, we describe our experimental setup. Then, we present the results of our experiments.

## 5.1 Experimental Setup

In this section, we first describe the different datasets we use to train and evaluate the various components of our pipeline. Further, we present the experimental plan and describe the relationships between our planned experiments, pipeline components and research questions. Finally, we describe our data pre-processing steps.

### 5.1.1 Datasets

To test the detection, segmentation and pavement distress forecasting models' performance on Trondheim roads, we collect video footage from seven road trips using a smartphone. The road trips are spaced approximately 3 weeks apart, taking place between January 15 and April 16. The video files vary in angle, light and weather conditions, and shakiness. The observed route is marked in Figure 5.1. We focus on this route as it contains a variety of pavement distress types in large quantities, as well as variations in amount of traffic between the different road segments.

We extract 1 frame per second of video and manually label them with bounding boxes and binary segmentation masks. If a pavement distress is present in multiple consecutive frames, we add one of the frames to the test set. As neither the bounding boxes nor segmentation masks are labeled by an expert, the test

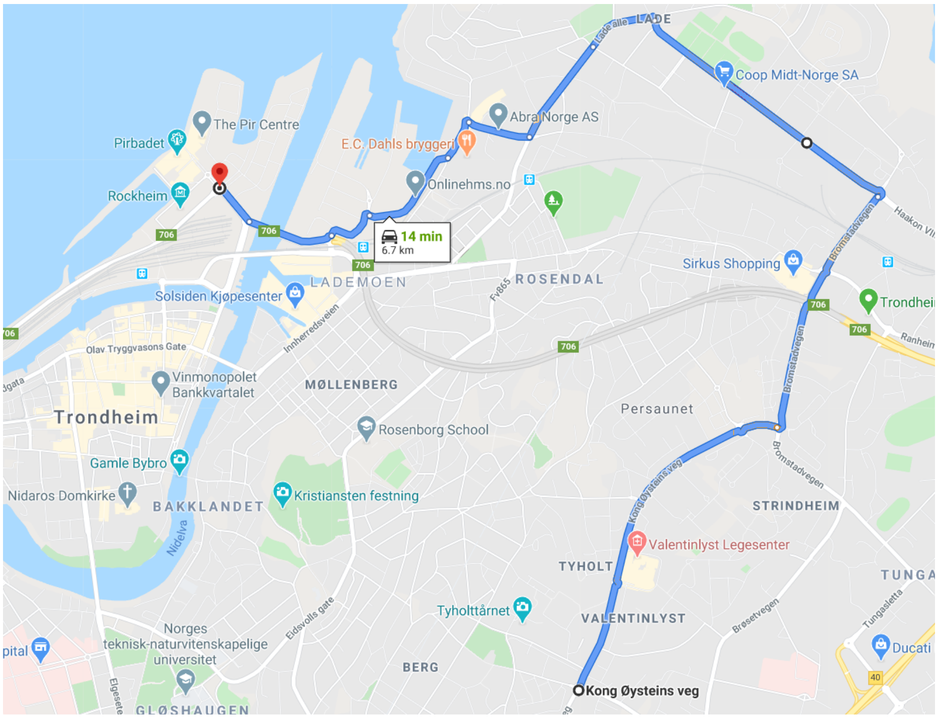


Figure 5.1: The observed route through all seven trips.

sets may contain errors. In addition, it is sometimes difficult to determine pavement distress type from the low-resolution images, in particular distinguishing between sealed and non-sealed cracks. We adopt the following standard: Dark, smooth cracks and cracks with a lighter area inside are labeled as sealed.

## Detection

We use a combination of the PID dataset [28], a pothole dataset [31] [32] and half of the Trondheim dataset for training and validating our object detection model. The PID dataset consists of 7,237 images taken with a pitch angle of -70 degrees and expert-annotated into nine pavement performance-critical distresses. However, as can be seen from Figure 5.2, this dataset is imbalanced, as there are few potholes relative to reflective, lane longitudinal and sealed longitudinal cracks. To account for this, we undersample the majority classes slightly and increase the minority class samples by using a variety of data augmentation methods, which

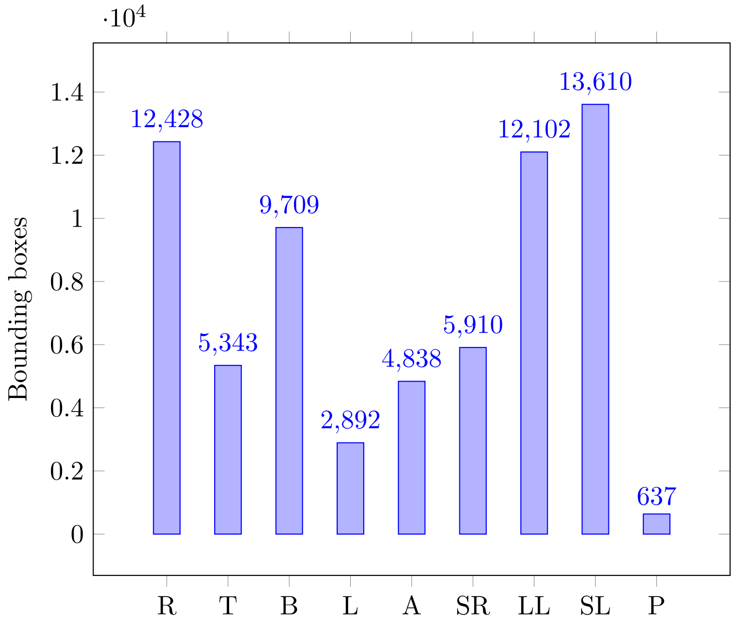


Figure 5.2: The class distribution of the original PID dataset. R=Reflective, T=Transverse, B=Block, L=Longitudinal, A=Alligator, SR=Sealed reflective, LL=Lane longitudinal, SL=Sealed longitudinal, P=Pothole.

are further described in Section 5.2.1. As each image can contain a variety of different pavement distresses, we found it difficult to balance the dataset further while ensuring the augmentation of a variety of different images so that overfitting does not occur.

As potholes are especially critical for pavement performance, we add additional potholes from another dataset (from [31] and [32]). In the final undersampled and combined dataset, we use a dataset split of 80:20 for training and validation, selected with a fixed random seed. The final class distribution can be seen in Figure 5.3.

To evaluate additional approaches such as data augmentation on data that is representative to the data the model will be working on when deployed, containing motion blur, shadows, different weather, etc., we add 50% of the Trondheim dataset to the validation set. The remaining 50% will be used for evaluating the final model. Table 5.1 shows the number of images used for training, validation and testing, distributed between the different datasets used.

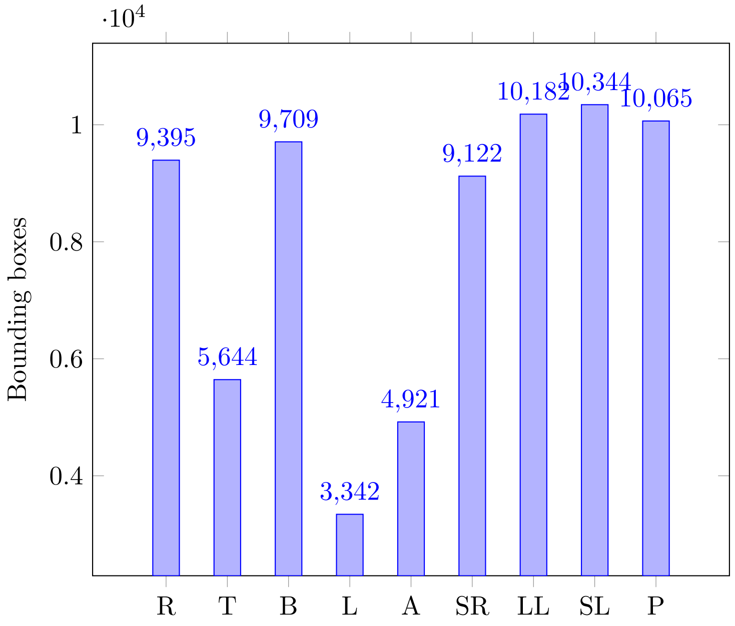


Figure 5.3: The final class distribution after undersampling, data augmentation and pothole addition. R=Reflective, T=Transverse, B=Block, L=Longitudinal, A=Alligator, SR=Sealed reflective, LL=Lane longitudinal, SL=Sealed longitudinal, P=Pothole.

| Dataset | PID | Potholes | Trondheim |
|---|---|---|---|
| Training | 4,472 | 3,281 | |
| Validation | 1,118 | 820 | 95 |
| Testing | | | 96 |

Table 5.1: Number of images in the training, validation and test set, distributed between the different datasets used.

### Quantification

The test set consists of 10 annotated images, comprised of a variety of pavement distress types: longitudinal cracks, sealed longitudinal cracks, alligator cracks and potholes. Some images are challenging, containing pavement marks or unclear potholes.

### Forecasting

Table 5.2 presents the dataset variables used for time series forecasting of pavement distress. We use severity levels of 0-2, corresponding to a low-medium-high assessment, as the outcome variable to be predicted. The severity of pavement distress is assessed manually. As two of the trips were subject to weather conditions where it was hard to assess severity (i.e. large amounts of water covering the distresses), we only consider the remaining five trips. We pick out 12 pavement distresses that are not obscured in any of the remaining video footage and manually assess their severity over the five trips, using the PCI method of assessing individual pavement distresses [ast]. Figure 5.4 displays the resulting severity levels of the 12 pavement distresses over the period of observation [1]. Each individual distress is assigned a distress id, which identifies that specific time series.

| **Variable** | **Type** |
|---|---|
| severity | Outcome, lagged |
| distress id | Group |
| distress type | Static |
| yearly daily traffic (YDT) | Static |
| YDT, percentage of long vehicles | Static |
| total precipitation | Lagged |
| maximum temperature | Lagged |
| minimum temperature | Lagged |
| freeze-thaw cycles | Lagged |
| month | Dynamic |

Table 5.2: Dataset variables.

The Norwegian Public Roads Administration's NVDB API is used to retrieve properties of road segments containing the observed pavement distresses, such as

[1] Images of the observed pavement distresses over the five trips can be found on our Github: `github.com/AlpakkAn/pavement-distress-datasets`

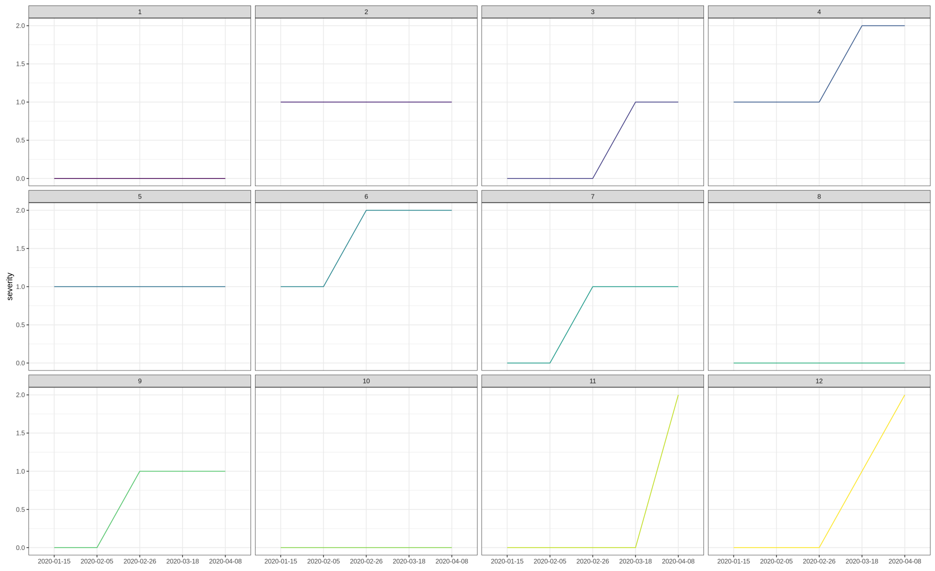


Figure 5.4: Severity levels of the 12 observed pavement distresses over the period of observation.

yearly daily traffic and percentage of long vehicles. In addition, total precipitation, maximum and minimum temperature and number of freeze-thaw cycles in the period between observations is retrieved using the Norwegian Meteorological Institute's Frost API. The number of freeze-thaw cycles are calculated based on the method in [16]. As the method uses pavement temperature data which are not available for the observed areas, we use the temperature conversion equations in [22] to convert from air temperatures to pavement temperatures [2].

### 5.1.2 Experimental Plan

Figure 5.5 describes the relationships between the planned experiments and our pipeline components, as well as which research questions (RQ) they aim to answer. The research questions are repeated below for clarity:

**Research question 1** *Which existing computer vision and image processing methods are suitable for detecting pavement distress in this application?*

[2]The resulting data table containing 60 rows can be downloaded from our Github: `github.com/AlpakkAn/pavement-distress-datasets`

**Research question 2** *Which existing computer vision and image processing methods are suitable for quantifying pavement distress?*

**Research question 3** *Which approach is suitable for predicting future states of pavement distress?*

**Research question 4** *Which features can be used for predicting future states of pavement distress in Trondheim?*

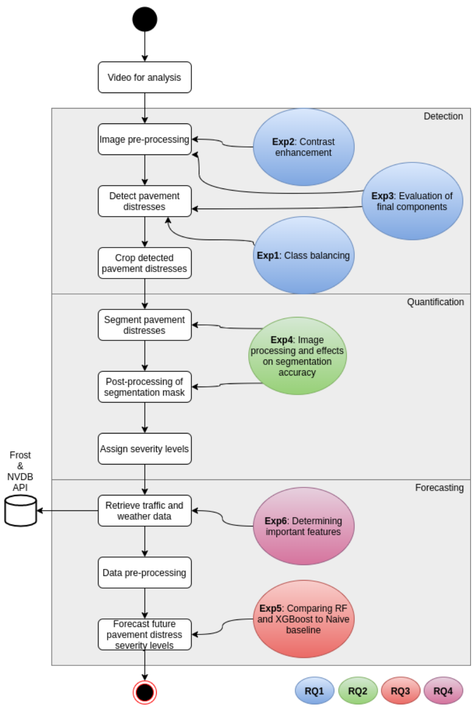

Video for analysis
Detection
Image pre-processing
Exp2: Contrast enhancement
Detect pavement distresses
Exp3: Evaluation of final components
Crop detected pavement distresses
Exp1: Class balancing
Quantification
Segment pavement distresses
Exp4: Image processing and effects on segmentation accuracy
Post-processing of segmentation mask
Assign severity levels
Frost & NVDB API
Forecasting
Retrieve traffic and weather data
Exp6: Determining important features
Data pre-processing
Forecast future pavement distress severity levels
Exp5: Comparing RF and XGBoost to Naive baseline
RQ1
RQ2
RQ3
RQ4

Figure 5.5: Relationships between experiments, pipeline components and research questions.

## Experiment 1: Class Balancing

Potholes are the most influential pavement distress in respect to pavement performance, being hazardous and highly contributing to the degradation of pavements [ast]. However, they also naturally constitute only a small proportion of most pavement distress datasets. To alleviate this class imbalance problem, we undersample the majority classes slightly and combine the PID dataset [28], with another dataset [31] [32] containing only South African potholes. Furthermore, we increase the other minority class samples by using a variety of data augmentation methods, comparing the methods' resulting accuracy on the Trondheim validation set.

### Experiment 2: Contrast Enhancement

In order to make the fine patterns of pavement distress more distinguishable from the pavement in low-resolution images, we perform different levels of global contrast enhancement on the validation images, comparing the validation accuracy of each level.

### Experiment 3: Evaluation of Final Components

We evaluate the final components from Experiment 1 and 2 on the Trondheim test set.

### Experiment 4: Image Processing and Effects on Segmentation Accuracy

As the cropped distresses can benefit from local contrast improvement, we think Contrast Limited Adaptive Histogram Equalization (CLAHE) can improve the contrast significantly, thereby improving DeepCrack's segmentation accuracy on the low-resolution images. Furthermore, as DeepCrack achieved good results when applying guided filtering as a post-processing step to refine the final output [25], we compare the segmentation accuracies of different combinations of CLAHE and guided filtering.

### Experiment 5: Comparing Random Forest and XGBoost to a Naïve Baseline

In order to perform road maintenance, one-step forecasts may not be enough, as road maintainers may need forecasts for multiple steps ahead. Additionally, we would like to explore whether different traffic and weather features can be utilised in making forecasts, as well as model each observed pavement distress as individual time series. Therefore, we model our problem as a multiple multivariate multi-step time series forecasting problem, making the simplifying assumption of no human intervention. We train XGBoost and Random forest for direct time series forecasting with forecasting horizons of 3, 6 and 9 weeks and compare them to a baseline Naïve forecast.

To utilise as much of the data we have as possible, we use cross-validation for the machine learning methods, following [2], splitting our dataset into three validation windows of size 1, 2 and 2. As Naïve forecast is not a purely autoregressive model with uncorrelated errors, making cross-validation an invalid approach, we evaluate it on a rolling forecast origin for each forecast horizon. Since being off by two severity levels is more than twice as bad as being off by one level, the forecasting models are evaluated on Root mean squared error (RMSE), as it penalises large errors more.

#### Experiment 6: Determining Important Features

To determine important features for predicting future states of pavement distress severity, we analyse the feature importances of the Random forest and XGBoost models from Experiment 5. We remove features with the lowest importance and evaluate the final models.

### 5.1.3 Data Preprocessing

#### Detection

As only the part of the image containing asphalt is important for pavement distress detection, the top half of the images in the detection test set are cropped out. This has the additional advantage of reducing inference time. As our YOLOv4 model is trained on resized 416x416 images, we pad the top with zeros to produce a square image. We do not resize the image further, as we need higher resolutions of the detected pavement distresses for quantification and as we need to detect relatively small objects. The resulting size for the images in the test set is 1920x1920.

As the pothole dataset has pixel resolutions of 3680x2760, we hypothesise that our object detection model might have a hard time detecting smaller potholes if trained on this dataset. In order to better detect potholes of varying sizes as well as reduce the number of unlabeled cracks in the pothole dataset, we crop out the image edges so that only the parts of the image containing potholes are included. We found no unlabeled cracks in the resulting dataset. The resulting size for the images in the pothole dataset varies between 214 and 3680 pixels in width, and 202 and 748 pixels in height.

#### Forecasting

As Random forest does not handle missing values, we impute the missing lagged values using MissForest [41].

## 5.2 Results

### 5.2.1 Detection

In the following pavement distress detection experiments, YOLOv4 pretrained on the MS COCO dataset is trained for 18,000 iterations with a batch size of 64 and learning rate of 0.001, on a Tesla V100. Furthermore, all reported AP values are for $IoU > 0.5$.

**Experiment 1: Class Balancing**

Table 5.3 shows the Trondheim validation results from the initial training of YOLOv4 on the PID dataset alone. Although the AP is low for most classes, it is relatively high for block and longitudinal cracks. The 0 % AP for reflective and sealed reflective cracks can be explained due to there not being any instances of these distresses in the Trondheim validation set, as seen in Figure 5.6. However, despite the relatively high occurrence of potholes in the validation set, the AP for potholes is at 0 %. We hypothesise that this is due to the imbalanced training dataset.

| **Distress type** | **AP@0.5** |
|---|---|
| Reflective crack | 0.00 % |
| Transverse crack | 4.76 % |
| Block crack | 33.33 % |
| Longitudinal crack | 17.94 % |
| Alligator crack | 0.75 % |
| Sealed reflective crack | 0.00 % |
| Lane longitudinal crack | 9.72 % |
| Sealed longitudinal crack | 1.25 % |
| Pothole | 0.00 % |
| mAP@0.5: 7.53 % | |

Table 5.3: Validation AP for each pavement distress type. The last row shows the validation mAP over all distress types.

During the initial training, we found the highest number of false positives for sealed longitudinal cracks. Furthermore, the model failed to detect any potholes despite constituting 26 samples of the validation set. To alleviate this, we incorporate additional potholes (from [31] and [32]) to the training set, as well as

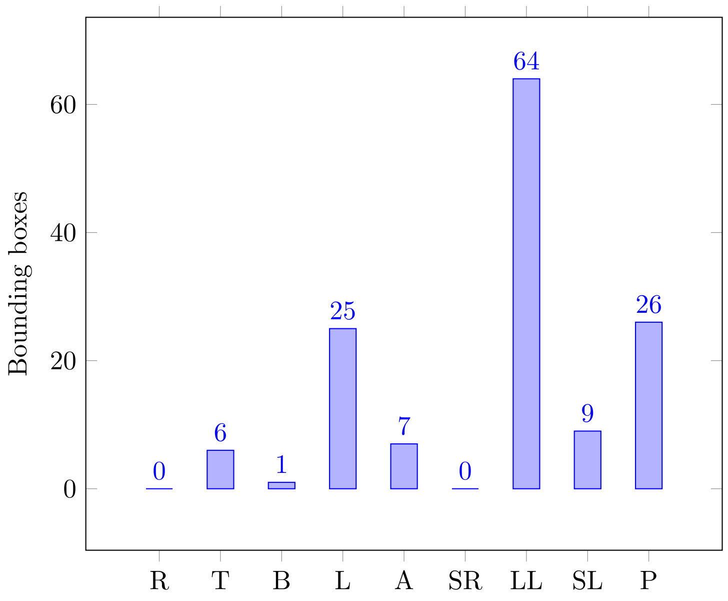


Figure 5.6: The class distribution of the Trondheim validation set. R=Reflective, T=Transverse, B=Block, L=Longitudinal, A=Alligator, SR=Sealed reflective, LL=Lane longitudinal, SL=Sealed longitudinal, P=Pothole.

undersample the majority classes slightly. Furthermore, we increase the minority class samples by using a variety of data augmentation methods.

Although YOLOv4 itself performs a variety of data augmentation methods to reduce overconfidence and overfitting, we choose to experiment with additional data augmentation methods in order to balance our dataset. As the deployed model will work on data potentially containing motion blur, winding roads and different light and weather conditions, we experiment with horizontal flipping, rotation between -30 and 30 deg., gaussian and motion blur, flipping and rotating, flipping and blurring, and weather augmentation. A summary of the results of each of these data augmentation strategies, combined with undersampling and the addition of potholes, can be seen in Table 5.4, and detailed results for the augmentation strategy yielding the highest mAP are in Table 5.5.

As can be seen from the lower AP for most distress types in Table 5.5, it may seem like the undersampling does more harm than good by leaving out necessary data, while not contributing much to pothole accuracy as potholes and the majority classes, reflective, lane longitudinal and sealed longitudinal cracks, have relatively different features. Furthermore, the addition of new potholes does not affect the AP for potholes, perhaps due to the potholes in the South African dataset containing different features from those in the Trondheim validation set.

| Data augmentation | mAP@0.5 |
|---|---|
| No augmentation | 7.53 |
| Flip | 3.78 |
| Rotation | 2.66 |
| Gaussian and motion blur | 4.62 |
| Flip and rotation | 1.23 |
| Flip and blur | 4.35 |
| Weather augmentation | 4.43 |

Table 5.4: Data augmentation techniques, combined with undersampling of majority classes and the addition of potholes, and their respective validation mAP.

### Experiment 2: Contrast Enhancement

We hypothesise that the low accuracy may be due to the model not being able to distinguish the fine patterns of pavement distress from pavement in low-resolution images. We therefore experiment with different levels of global contrast enhancement on the images in the Trondheim validation set. As seen in Table 5.6, there seems to be an optimal level between 100 and 125.

As can be seen from Table 5.7, which compares the validation results of level

| **Distress type** | **AP, blur** | **AP, initial** |
|---|---|---|
| Reflective crack | 0.00 % | 0.00 % |
| Transverse crack | 0.62 % | 4.76 % |
| Block crack | 16.67 % | 33.33 % |
| Longitudinal crack | 20.24 % | 17.94 % |
| Alligator crack | 0.00 % | 0.75 % |
| Sealed reflective crack | 0.00 % | 0.00 % |
| Lane longitudinal crack | 1.03 % | 9.72 % |
| Sealed longitudinal crack | 3.02 % | 1.25 % |
| Pothole | 0.00 % | 0.00 % |
| mAP@0.5 | 4.62 % | 7.53 % |

Table 5.5: Comparison of validation AP for blur augmentation and initial training for each pavement distress type.

100 contrast enhancement and initial training for each distress type, the high increase in mAP is primarily due to the one block crack in the validation set being detected correctly. Nevertheless, we see an increased AP for most distress types, most notably a 15 and 16 percentage point increase for sealed longitudinal and alligator cracks, respectively.

| Level | 0 | 50 | 75 | 100 | 125 | 150 |
|---|---|---|---|---|---|---|
| mAP | 7.53 % | 8.95 % | 9.42 % | 18.32 % | 17.25 % | 5.29 % |

Table 5.6: Influence of image contrast enhancement levels on Trondheim validation mAP, ranging from level 0 (no contrast enhancement) to 150.

### Experiment 3: Evaluation of Final Components

The final components, the YOLOv4 model trained only on the PID dataset, combined with level 100 contrast enhancement, is evaluated on the Trondheim test set. Table 5.8 compares the results on the validation and test sets. The significantly lower mAP on the test set can be partly attributed to the one block crack in the test set (Figure 5.7), but overall the results indicate that the components are unable to generalise to new pavement distresses.

| Distress type | AP, level 100 contrast | AP, initial |
|---|---|---|
| Reflective crack | 0.00 % | 0.00 % |
| Transverse crack | 1.39 % | 4.76 % |
| Block crack | 100 % | 33.33 % |
| Longitudinal crack | 22.43 % | 17.94 % |
| Alligator crack | 16.48 % | 0.75 % |
| Sealed reflective crack | 0.00 % | 0.00 % |
| Lane longitudinal crack | 8.68 % | 9.72 % |
| Sealed longitudinal crack | 15.89 % | 1.25 % |
| Pothole | 0.00 % | 0.00 % |
| mAP@0.5 | 18.32 % | 7.53 % |

Table 5.7: Comparison of validation AP for level 100 contrast and initial training for each pavement distress type.

| Distress type | AP, test | AP, validation |
|---|---|---|
| Reflective crack | 0.00 | 0.00 % |
| Transverse crack | 0.00 | 1.39 % |
| Block crack | 20.00 | 100 % |
| Longitudinal crack | 4.75 | 22.43 % |
| Alligator crack | 2.06 | 16.48 % |
| Sealed reflective crack | 0.00 | 0.00 % |
| Lane longitudinal crack | 6.67 | 8.68 % |
| Sealed longitudinal crack | 2.20 | 15.89 % |
| Pothole | 0.00 % | 0.00 % |
| mAP@0.5 | 3.96 % | 18.32 % |

Table 5.8: Comparison of test and validation AP for level 100 contrast for each pavement distress type.

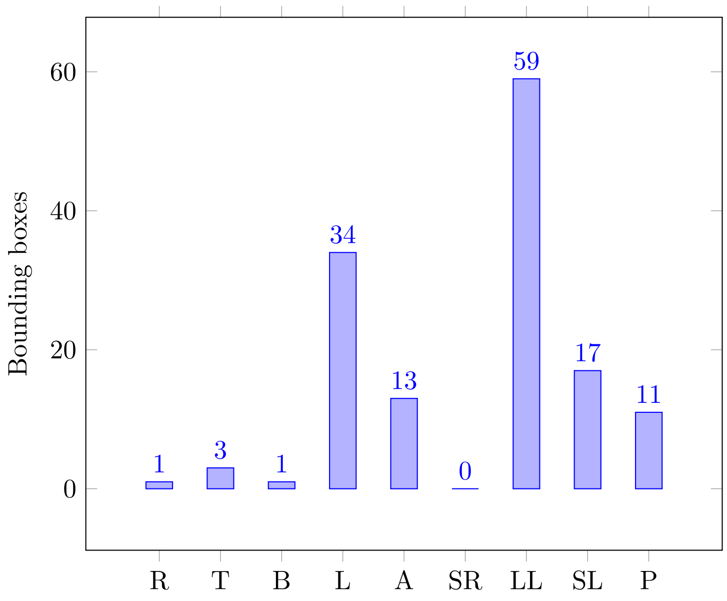


Figure 5.7: The class distribution of the Trondheim test set. R=Reflective, T=Transverse, B=Block, L=Longitudinal, A=Alligator, SR=Sealed reflective, LL=Lane longitudinal, SL=Sealed longitudinal, P=Pothole.

### 5.2.2 Quantification

#### Experiment 4: Image Processing and Effects on Segmentation Accuracy

Table 5.9 compares the results of applying DeepCrack to the original Trondheim dataset and the dataset with improved contrast using CLAHE, as well as comparing the results of refining both segmentation outputs using guided filtering. For guided filtering, we use the parameters $radius = 10$ and $\epsilon = 50$. CLAHE improves the mean IoU and F1 score by 6.7 and 21.5 % from the original images, respectively. Likewise, performing guided filtering on the CLAHE segmentation output improves the mean IoU and F1 score by 3.3 and 6.3 %, respectively.

| | Best threshold | Mean IoU | F1 score |
|---|---|---|---|
| Original | 0.02 | 0.5568 | 0.3576 |
| Original + GF | 0.47 | 0.5830 | 0.3956 |
| CLAHE | 0.04 | 0.5941 | 0.4346 |
| CLAHE + GF | 0.49 | 0.6136 | 0.4618 |

Table 5.9: A comparison of applying or not applying CLAHE and guided filtering and their corresponding mean IoU, F1 score and best threshold for F1 score.

Figure 5.8 shows some examples of applying DeepCrack with CLAHE and guided filtering to the Trondheim test set. As illustrated by the last row of images, the initial results from applying DeepCrack to potholes are not very good.

### 5.2.3 Forecasting

#### Experiment 5: Comparing Random Forest and XGBoost to a Naïve Baseline

We train XGBoost and Random forest models for direct time series forecasting with forecasting horizons of 3, 6 and 9 weeks and compare them to a baseline Naïve forecast. The evaluation of the Random forest and XGBoost models are done using 3-fold cross-validation, while the Naïve forecast is evaluated on a rolling forecast origin. We use feature lags of 1 for Random forest as this produces the best total RMSE, as can be seen in Table 5.10. Although the best total RMSE for Xgboost is achieved with no lagged features, we choose to incorporate feature lags of 3 in order to study the feature importance of different lags later in Section 5.2.3. Figure 5.9 shows a comparison of the RMSE of the three models collapsed across all validation windows and direct forecast horizons, for each of the 12 observed pavement distresses. Table 5.11 summarises the total RMSE for the three models. Random forest and XGBoost outperform the Naïve forecast slightly with a 32.79 and 32.34 % decrease in forecast error, respectively.

| Number of lags | Random forest | XGBoost |
|---|---|---|
| 0 | 0.540 | 0.408 |
| 1 | 0.453 | 0.540 |
| 2 | 0.677 | 0.456 |
| 3 | 0.842 | 0.456 |

Table 5.10: Influence of number of lags on total RMSE for Random forest and Xgboost.

Figure 5.10 shows the forecast error of Random forest and XGBoost for each of the three direct forecast horizons collapsed across validation windows.

| Model | RMSE |
|---|---|
| Naïve | 0.674 |
| Random forest | 0.453 |
| XGBoost | 0.456 |

Table 5.11: Summary of total RMSE for Naïve forecast, Random forest and XGBoost.

### Experiment 6: Determining Important Features

As seen from Figure 5.11 and 5.12, the severity lags seem to be important features for the models. Severity lags from 3 weeks prior (*severity_lag_1*) has the highest importance for the 3-week forecast horizon in both models. Furthermore, severity lags from 6 weeks prior (*severity_lag_2*) has the highest importance for the 6-week forecast horizon in XGBoost, and 9-week severity lags (*severity_lag_3*) the second highest importance for the 9-week horizon. The distress id feature seems to be important, which might be attributed to that we did not perform target encoding on this feature. Distress type seems to be moderately important for both models. Yearly daily traffic (*ydt*) and month are moderately important for the Random forest models.

The percentage of long vehicles (*ydt_long*) and weather-related features seem to have little importance for both models. By removing these features, we get the total RMSE as seen in Figure 5.13, corresponding to a 43.6 % decrease in forecast error for Random forest compared to Naïve.

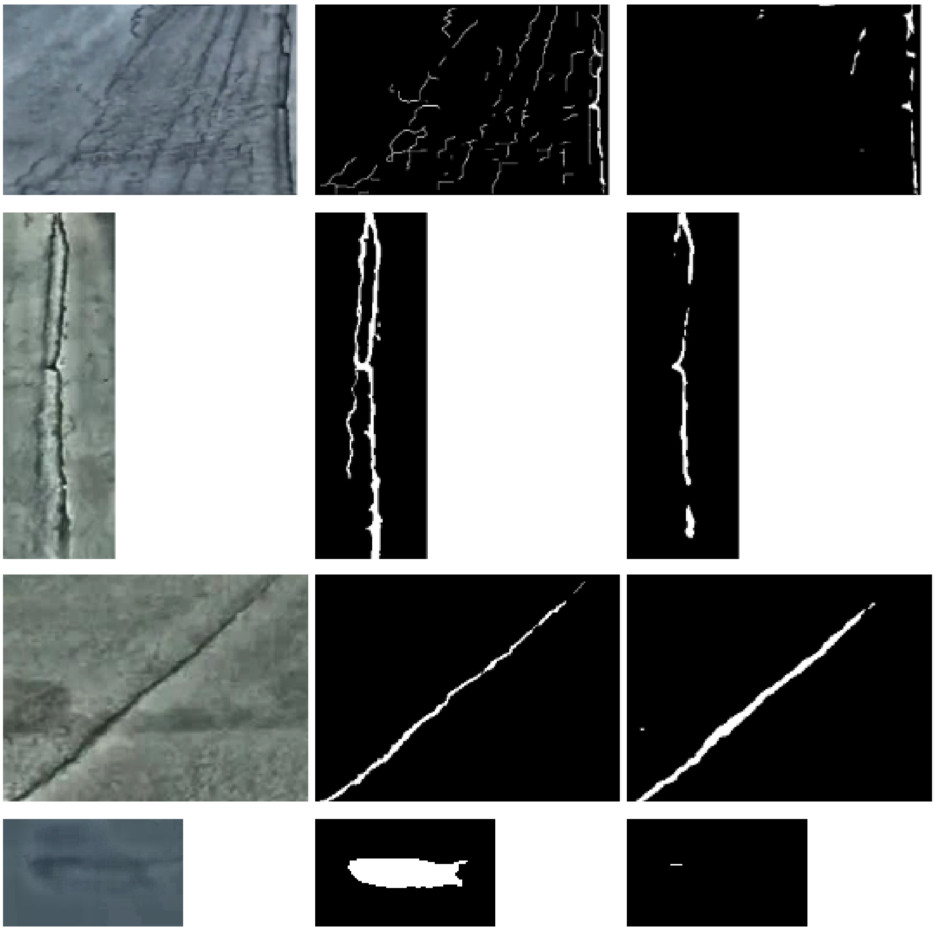


Figure 5.8: Some results of applying DeepCrack to the Trondheim dataset. The first column contains the contrast-enhanced images, the second column the ground truth binary mask, and the third column the prediction.

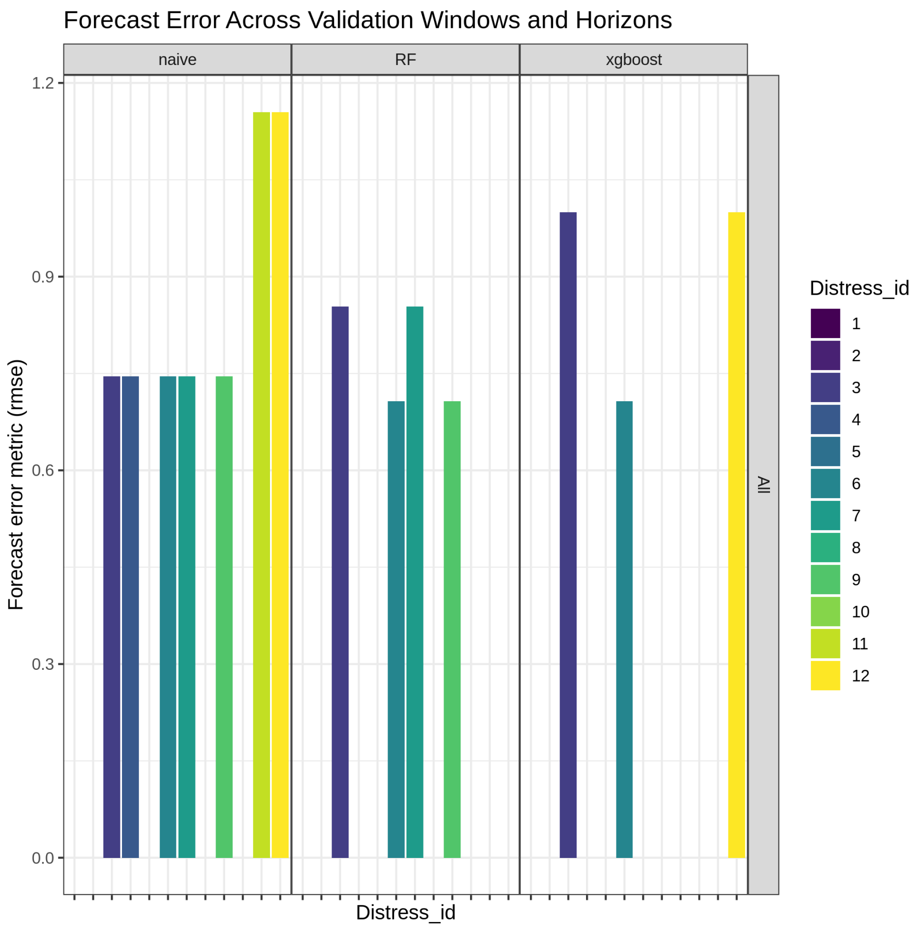


Figure 5.9: Comparison of RMSE of Naïve, Random forest and Xgboost across validation windows and forecast horizons.

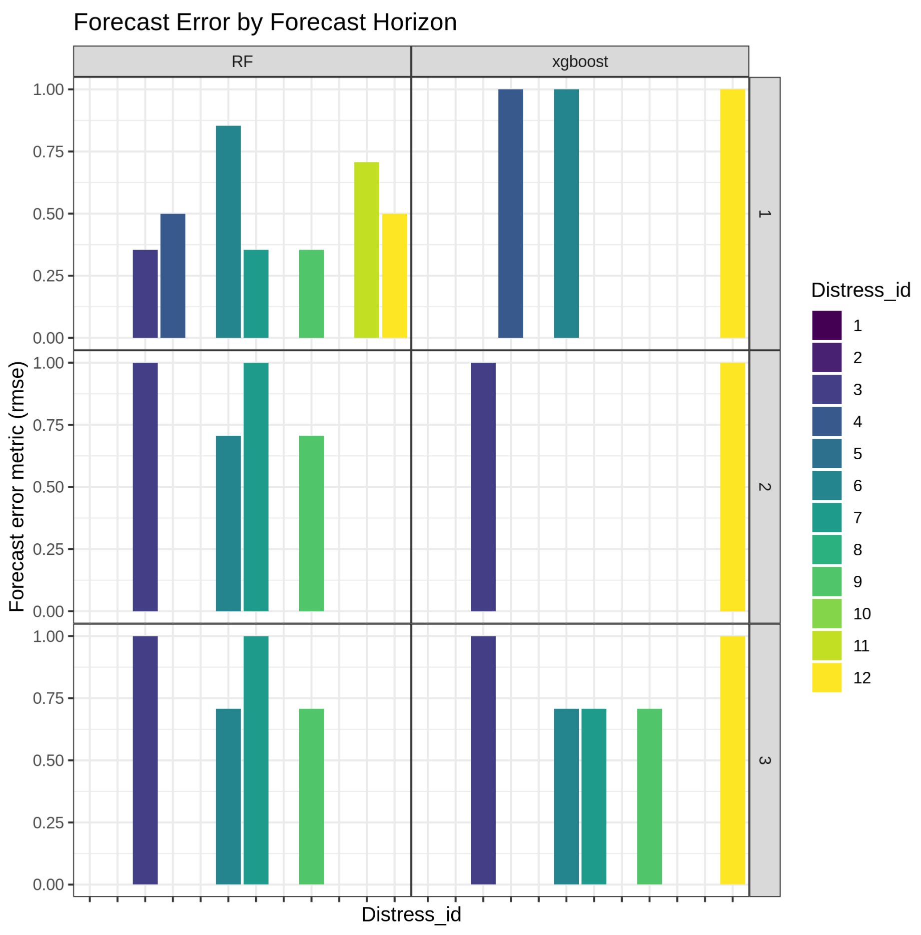


Figure 5.10: Comparison of RMSE of Random forest and Xgboost by forecast horizon. Horizon 1 = 3 weeks, Horizon 2 = 6 weeks, Horizon 3 = 9 weeks.

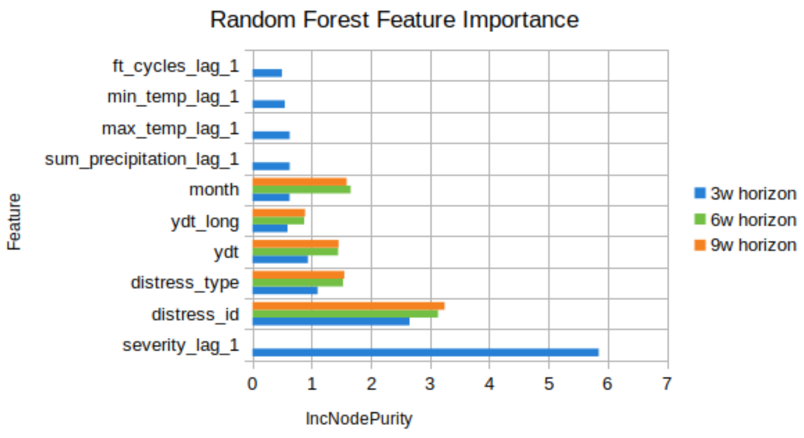


Figure 5.11: Comparison of increase in node purity for different Random forest features, using feature lags of 1.

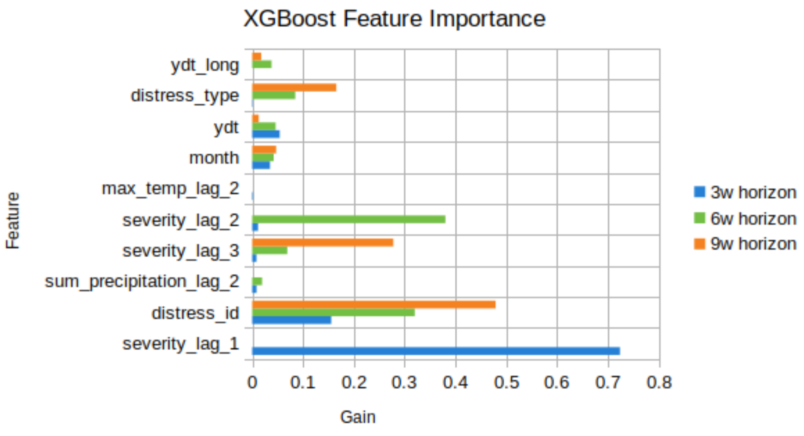


Figure 5.12: Comparison of gain for different XGBoost features, using feature lags of 3. Features with negligible gain, e.g. some or all lags of minimum temperature, maximum temperature, total precipitation and number of freeze-thaw cycles, are not shown.

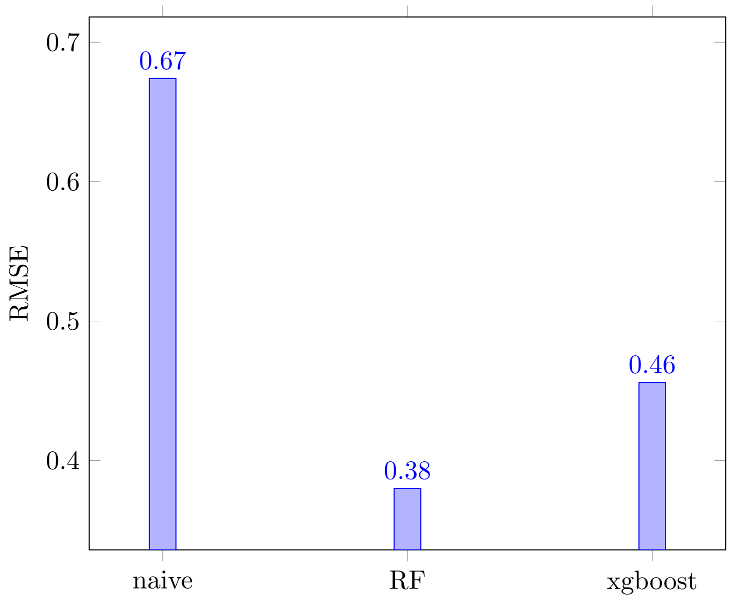


Figure 5.13: Comparison of total RMSE after feature selection.

# Chapter 6

# Discussion and Conclusion

## 6.1 Evaluation

In this section, we discuss our results with respect to the research questions introduced in Section 1.1. Then, we discuss the limitations of our work. Finally, we summarise our main contributions and describe potential extensions for future work.

**Research question 1** *Which existing computer vision and image processing methods are suitable for detecting pavement distress in this application?*

Table 5.6 and 5.7 suggest that contrast enhancement could be a viable approach to improve detection accuracy in low-resolution images. However, with our small validation set of 95 images, caution must be applied. Furthermore, as Table 5.8 shows, there is a large discrepancy between the validation and test accuracy, indicating that our trained YOLOv4 model is unable to generalise to new pavement distresses in the Trondheim test set. A possible explanation for these results may be the lack of adequate training samples for Trondheim roads, as the training set we used is different from the test data in respect to occurrences of urban objects and seasonal conditions. Additionally, different distress types can visually be very similar, only minor features differentiating them, further amplifying the need for larger, more diverse datasets.

**Research question 2** *How can pavement distress be quantified?*

The results in Table 5.9 suggest that CLAHE and guided filtering are viable approaches to improve semantic segmentation accuracy of cropped-out distresses in low-resolution images. However, these findings may be somewhat limited by the small test set size. As Figure 5.8 illustrates, DeepCrack performs relatively well for certain distresses, but not quite as well for potholes and thinner cracks.

Furthermore, our pipeline assigns each pavement distress a severity level based on the distress type and segmentation mask pixel count. However, existing standards for manual inspection are based on crack width or, in the case of alligator cracks, the width of the whole damaged area. Additionally, potholes are assessed based on diameter and depth [ast]. It is not clear what the best approach to distress quantification is. As existing standards are developed for manual inspection, they might not transfer well to automated assessment. Further experiments in collaboration with domain experts are needed to determine a suitable approach to severity assignments based on the obtained segmentation masks.

**Research question 3** *Which approach is suitable for predicting future states of pavement distress?*

Figure 5.13 shows an RMSE reduction of 43.6 % and 32.3 % from the baseline for the final Random forest and XGBoost models, respectively, with default hyperparameters. We have only compared machine learning methods, which allows us to take advantage of k-fold cross-validation on our limited dataset. However, further experiments are needed with a more extensive set of methods, comprised of both a variety of statistical and machine learning models, to be able to establish the comparative suitability of our chosen methods to others.

**Research question 4** *Which features can be used for predicting future states of pavement distress in Trondheim?*

Figure 5.11 and 5.12 indicate that severity lags are the most important features for each forecast horizon, while some traffic- and weather-related features are less important. In particular, the percentage of long vehicles, minimum and maximum temperature, total precipitation and number of freeze-thaw cycles have low importance. When removing these features, we see a decrease in RMSE for the Random forest model, while the XGBoost RMSE stays the same. These findings are contrary to [18] and [12], but consistent with that of [29], which found that their pavement performance prediction model was most sensitive to previous values of pavement condition, while not being sensitive to the removal of other features such as annual average precipitation, temperature, freeze index, humidity or daily truck traffic. However, contrary to [29], our XGBoost model achieved the lowest RMSE using no feature lags. The reason for this is not clear, but a possible explanation could be overfitting due to the short time series or little hyperparameter tuning. Furthermore, although our results show a relatively moderate importance for the distress type feature, this finding may be somewhat limited by our dataset comprising only 12 distresses.

## 6.2 Discussion

Our work is limited by the absence of a domain expert not only to verify our approach, but also to verify our test set annotations. In particular, it is difficult to distinguish between sealed and non-sealed cracks from the images. Our standard for labeling sealed cracks, defined in Section 5.1.1, might cause non-sealed cracks to be mislabeled as sealed when there is a combination of moisture filling the cracks and certain lightning. Furthermore, the manually labeled segmentation masks may contain inaccuracies due to the difficulty in distinguishing between crack and non-crack pixels from the low-resolution images. In addition, variations in car height and weather conditions might have influenced severity assessments. A more top-down angle might have resulted in a higher severity assessment as more of the pavement distress fills the image. Furthermore, the combination of sunlight and moisture filling the cracks makes the cracks more noticeable, possibly leading to a higher severity assessment, requiring more data points to reduce this noise.

Another limitation of our work is that a significant amount of pavement distresses in the validation and test set for object detection are the same, but in various weather conditions and different points in time, possibly making the final evaluation results slightly more optimistic. To alleviate this issue, multiple different driving routes should be included for data collection for distress detection. The increase in accuracy for guided filtering of the segmentation mask may also be too optimistic. As we had very few annotated images, we did not use a validation/test set split, and used the best test set threshold to perform the filtering.

A weakness of the quantification stage of our ALPACA pipeline is that it crops around the bounding boxes of detected distresses prior to performing semantic segmentation. Although this approach reduces computation and noise from other objects, it is also dependent on accurate bounding boxes in order to ensure accurate quantification. Another issue that was not addressed in our work was whether our pipeline component that assigns severity levels can produce similar assessments as domain experts.

Furhermore, the five remaining data points for observation after the removal of corrupted data, taking place between January 15 and April 8, is not enough to capture seasonality in the forecasting models. This makes our models less generalisable across seasons. In addition, some of the traffic-related features retrieved from the NVDB API, e.g. yearly daily traffic and percentage of long vehicles, are partly based on discretion.

## 6.3 Contributions

Despite its exploratory nature, this thesis offers some insight into how various components can be integrated to predict future pavement distress severity from smartphone videos or images. In summary, the main contributions of this thesis are as follows:

- We proposed a pipeline to predict future pavement distress severity from videos or images, which is able to run on a conventional 8GB GPU.
- We presented a quantitative evaluation of each stage of our pipeline. Furthermore, we performed an analysis of the features used for the last stage of our pipeline, pavement distress forecasting, to evaluate the importance of various features related to traffic and weather.
- We built an annotated dataset for object detection of pavement distress in urban environments containing challenging seasonal and lightning variations to be further used for training and evaluating new models. We also provided an annotated dataset for semantic segmentation of pavement distress in low-resolution smartphone images. We made these datasets openly available.

## 6.4 Future Work

This section presents suggestions for improvement and possible extensions to the ALPACA pipeline, as well as other directions for further research.

### 6.4.1 Suggestions for Improvement

As indicated by our evaluation and discussion in Section 6.1 and 6.2, further experiments with an extensive set of methods, using more collected data and additional features, should be conducted in order to assess the methods' comparative performance, as well as explore other sets of features. Although the roads we observed did not have any structural data on pavements available, some of the larger roads provide such data, i.e. pavement thickness, mass consumption and pavement age.

Furthermore, as our evaluation and discussion indicated, further research in collaboration with domain experts is needed to determine a suitable approach to severity assessments based on segmentation masks. To improve object detection of pavement distress, calculating new anchors based on common pavement distress shapes could be done. Other improvements include hyperparameter tuning and feature selection for the forecasting models, target encoding the distress id feature, as well as using top-down images for more accurate quantification.

### 6.4.2 Forecasting Pavement Section Quality

For road maintainers, forecasts for particular pavement distresses might not be as useful as forecasts for whole pavement sections, which may make it easier to plan maintenance interventions. Further research might explore this direction, computing an index based on the forecasted severity levels of each distress type for a particular pavement section, which can be used to prioritise interventions. Development within this direction could be beneficial to road maintenance and potentially other infrastructures, as such distress prediction could lead to a focus on damage prevention instead of repair.

### 6.4.3 Building an Instance Segmentation Dataset

As indicated by our literature review in Chapter 3, we found no prior studies using instance segmentation to detect and segment multiple types of pavement distress. A reason for this could be that building such a dataset for instance segmentation would be time-consuming, requiring manual labeling of thousands of fine masks for each distress type. One solution for this could be to use our pipeline, or a similar multi-stage approach, to partly generate this dataset. Performing distress detection and segmentation with instance segmentation would not only be more computationally efficient than our approach, but also remove the pipeline's dependence on accurate bounding box predictions from the prior stage.

# Bibliography

[ast] ASTM D6433 - 18 Standard Practice for Roads and Parking Lots Pavement Condition Index Surveys.

[2] Bergmeir, C., Hyndman, R. J., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics and Data Analysis*, 120:70–83.

[3] Bhatt, U., Mani, S., Xi, E., and Kolter, J. Z. (2017). Intelligent pothole detection and road condition assessment. *arXiv preprint arXiv:1710.02595*.

[4] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

[5] Dong, Y., Shao, Y., Li, X., Li, S., Quan, L., Zhang, W., and Du, J. (2019). Forecasting pavement performance with a feature fusion lstm-bpnn model. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1953–1962.

[6] Du, Y., Pan, N., Xu, Z., Deng, F., Shen, Y., and Kang, H. (2020). Pavement distress detection and classification based on yolo network. *International Journal of Pavement Engineering*, pages 1–14.

[7] Eaton, R. A., Joubert, R. H., and Wright, E. A. (1981). Pothole primer; a public administrator's guide to understanding and managing the pothole problem.

[8] Fathi, A., Mazari, M., Saghafi, M., Hosseini, A., and Kumar, S. (2019). Parametric study of pavement deterioration using machine learning algorithms. In *Airfield and Highway Pavements 2019: Innovation and Sustainability in Highway and Airfield Pavement Technology*, pages 31–41. American Society of Civil Engineers Reston, VA.

[9] Georgiou, P., Plati, C., and Loizos, A. (2018). Soft computing models to predict pavement roughness: a comparative study. *Advances in Civil Engineering*, 2018.

[10] Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1440–1448.

[11] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587.

[12] Gong, H., Sun, Y., Shu, X., and Huang, B. (2018). Use of random forests regression for predicting iri of asphalt pavements. *Construction and Building Materials*, 189:890–897.

[13] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[14] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2020). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397.

[15] He, K., Sun, J., and Tang, X. (2013). Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409.

[16] Ho, E. and Gough, W. A. (2006). Freeze thaw cycles in Toronto, Canada in a changing climate. *Theoretical and Applied Climatology*, 83(1-4):203–210.

[17] Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

[18] Inkoom, S., Sobanjo, J., Barbu, A., and Niu, X. (2019a). Pavement crack rating using machine learning frameworks: Partitioning, bootstrap forest, boosted trees, naïve bayes, and k-nearest neighbors. *Journal of Transportation Engineering, Part B: Pavements*, 145(3):04019031.

[19] Inkoom, S., Sobanjo, J., Barbu, A., and Niu, X. (2019b). Prediction of the crack condition of highway pavements using machine learning models. *Structure and Infrastructure Engineering*, 15(7):940–953.

[20] Kamaliardakani, M., Sun, L., and Ardakani, M. K. (2016). Sealed-crack detection algorithm using heuristic thresholding approach. *Journal of Computing in Civil Engineering*, 30(1):04014110.

[21] Kargah-Ostadi, N. (2014). Comparison of machine learning techniques for developing performance prediction models. In *Computing in Civil and Building Engineering (2014)*, pages 1222–1229.

[22] Kwiatkowski, K. P., Stipanovic Oslakovic, I., Ter Maat, H., Hartmann, A., Chinowsky, P., and Dewulf, G. P. (2020). Modeling Cost Impacts and Adaptation of Freeze-Thaw Climate Change on a Porous Asphalt Road Network. *Journal of Infrastructure Systems*, 26(3):04020022.

[23] Li, N., Haas, R., and Xie, W.-C. (1997). Development of a new asphalt pavement performance prediction model. *Canadian Journal of Civil Engineering*, 24(4):547–559.

[24] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and DollÃ¡r, P. (2017). Focal loss for dense object detection.

[25] Liu, Y., Yao, J., Lu, X., Xie, R., and Li, L. (2019). DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, 338:139–153.

[26] Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., and Omata, H. (2018). Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1127–1141.

[27] Majidifard, H., Adu-Gyamfi, Y., and Buttlar, W. G. (2020a). Deep machine learning approach to develop a new asphalt pavement condition index. *Construction and Building Materials*, 247:118513.

[28] Majidifard, H., Jin, P., Adu-Gyamfi, Y., and Buttlar, W. G. (2020b). PID: A New Benchmark Dataset to Classify and Densify Pavement Distresses. (arXiv:1910.11123v1 [cs.CV]). *arXiv Computer Science*, page 036119812090728.

[29] Marcelino, P., de Lurdes Antunes, M., Fortunato, E., and Castilho Gomes, M. (2019). Machine learning approach for pavement performance prediction. *International Journal of Pavement Engineering*, pages 1–14.

[30] Misra, D. (2019). Mish: A Self Regularized Non-Monotonic Neural Activation Function.

[31] Nienaber, S., Booysen, M., and Kroon, R. (2015a). Detecting Potholes Using Simple Image Processing Techniques and Real-World Footage. *Proceedings of the 34th Southern African Transport Conference (SATC*, (Satc):153–164.

[32] Nienaber, S., Kroon, R. S., and Booysen, M. J. (2015b). A comparison of low-cost monocular vision techniques for pothole distance estimation. In *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, pages 419–426.

[33] Opsomer, J., Wang, Y., and Yang, Y. (2001). Nonparametric regression with correlated errors. *Statistical Science*, pages 134–153.

[34] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 779–788.

[35] Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 6517–6525.

[36] Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement.

[37] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

[38] Ryu, S.-K., Kim, T., and Kim, Y.-R. (2015). Image-based pothole detection system for its service and road management system. *Mathematical Problems in Engineering*, 2015.

[39] Schlotjes, M. R., Burrow, M. P., Evdorides, H. T., and Henning, T. F. (2015). Using support vector machines to predict the probability of pavement failure. In *Proceedings of the Institution of Civil Engineers-Transport*, volume 168, pages 212–222. Thomas Telford Ltd.

[40] Shi, Y., Cui, L., Qi, Z., Meng, F., and Chen, Z. (2016). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3434–3445.

[41] Stekhoven, D. J. and Buhlmann, P. (2011). Missforest–non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112â118.

[42] Tabatabaee, N., Ziyadi, M., and Shafahi, Y. (2013). Two-stage support vector classifier and recurrent neural network predictor for pavement performance modeling. *Journal of Infrastructure Systems*, 19(3):266–274.

[43] Tedeschi, A. and Benedetto, F. (2017). A real-time automatic pavement crack and pothole recognition system for mobile android-based devices. *Advanced Engineering Informatics*, 32:11–25.

[44] Wilson, T. P. and Romine, A. (2001). Materials and procedures for repair of potholes in asphalt-surfaced pavements–manual of practice. Technical report.

[45] Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., and Ling, H. (2020). Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1525–1535.

[46] Yun, S., Han, D., Chun, S., Oh, S. J., Choe, J., and Yoo, Y. (2019). CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 6022–6031.

[47] Zeiada, W., Dabous, S. A., Hamad, K., Al-Ruzouq, R., and Khalil, M. A. (2020). Machine learning for pavement performance modelling in warm climate regions. *Arabian Journal for Science and Engineering*, pages 1–19.

[48] Zhang, K., Cheng, H., and Zhang, B. (2018). Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning. *Journal of Computing in Civil Engineering*, 32(2):04018001.

[49] Zhao, Z. Q., Zheng, P., Xu, S. T., and Wu, X. (2019). Object Detection with Deep Learning: A Review.

[50] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., and Ren, D. (2019). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression.

[51] Ziari, H., Maghrebi, M., Ayoubinejad, J., and Waller, S. T. (2016). Prediction of pavement performance: application of support vector regression with different kernels. *Transportation Research Record*, 2589(1):135–145.