TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power Engineering and Mechatronics

DEEP LEARNING BASED PAVEMENT DISTRESS INSTANCE SEGMENTATION FROM ROAD ORTHOPHOTOS

TEEKATTEDEFEKTIDE INDIVIDUAALNE SEGMENTEERIMINE ORTOKAADRITELT SÜVAÕPPE MEETODIL

MASTER THESIS

Student: Kayode Hadilou ADJE

Student Code: 194360MAHM

Supervisor: Andri Riid Ph.D., Senior Research Scientist

Co-Supervisor Prof. Mart Tamre, Professor

Tallinn,2021

AUTHOR'S DECLARATION

I hereby declare that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

".....” ..... 20…..

Author: .....
/signature /

Thesis is in accordance with terms and requirements

".....” ..... 20….

Supervisor: ….....
/signature/

Accepted for defence

".....”.....20… .

Chairman of theses defence commission: .....
/name and signature/

Non-exclusive Licence for Publication and Reproduction of a Graduation thesis[1]

I Kayode Hadilou ADJE (date of birth 20/04/1995)

1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis Deep learning based pavement distress instance segmentation from road orthophotos,

supervised by Andri Riid Ph.D.,

co-supervised by Prof. Mart Tamre

1.1 reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2 published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

1.3 I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

2. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

---

[1] *The non-exclusive licence for Publication and Reproduction of Graduation Thesis is not valid during the validity of access, except the university's right to reproduce the thesis for preservation only purposes.*

_____ (signature)

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student: Kayode Hadilou ADJE,194360MAHM
Study programme MAHM02/18-Mechatronics
main speciality: Mechatronics
Supervisor: Senior Research Scientist, Ph.D. Andri Riid, Tallinn University of Technology
Co-Supervisor: Professor, Mart Tamre, Tallinn University of Technology

Thesis topic:

(In English) Deep learning-based pavement distress instance segmentation from road orthophotos
(in Estonian) Teekattedefektide individuaalne segmenteerimine ortokaadritelt süvaõppe meetodil

Thesis main objectives:

1. To research convolutional neural network-based methods for instance segmentation and road pavement detection
2. To implement a convolutional neural network method for road distress instance segmentation.
3. To implement the solution on existing defect database layers and compare results.

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Task understanding and initial literature review	October 2020
2.	Further literature review and dataset preparation	November 2020
3.	First network training with Mask RCNN as a baseline	December 2020
4.	Implementation and training of feature pyramid network on Mask R-CNN	January 2021
5.	Implementation and training of Cascade Mask R-CNN	February 2021
6.	Fine-tuning of the middle region proposal network	March 2021
7.	Integration of the network into existing defect detection databases, additional test	April, May 2021
8.	Thesis report	May 2021

Language: English **Deadline for submission of thesis:** "10" June 2021

Student: Kayode Hadilou ADJE "....."……..2021

/signature/

Supervisor: Andri Riid, Ph.D. "....."……..2021

/signature/

Co-Supervisor: Prof Mart Tamre "....."..... 2021

/signature/

Head of study programme: Prof Mart Tamre "....."..... 2021

/signature/

Table of contents

Table of contents 6
List of figures 8
List of tables 10
PREFACE 11
List of abbreviations 12
1 INTRODUCTION 13
2 LITERATURE REVIEW AND BACKGROUND 15
2.1 Road defect detection methods 15
2.1.1 Hand-crafted methods 15
2.1.2 Classical machine learning methods 17
2.1.3 Deep learning methods 18
2.1.4 Summary of road defect detection methods 21
2.2 Convolutional neural networks 22
2.3 Instance segmentation 24
2.3.1 Two-stage instance segmentation 25
2.3.2 Single-stage instance segmentation 25
2.4 Pavement data acquisition 26
2.5 Data annotation 27
3 DATA PREPARATION 29
3.1 Data analysis and partition 29
3.2 Data generation for instance segmentation 30
3.3 Data augmentation 32
4 MASK RCNN FOR PAVEMENT DEFECT INSTANCE SEGMENTATION 34
4.1 Backbone feature extractor 34
4.1.1 Convolution operation 35
4.1.2 Activation function 36
4.1.3 Pooling operation 37
4.1.4 Normalization 37
4.2 Region proposal network 37
4.2.1 Region proposal network’s architecture 37
4.2.2 Anchor box generation 38
4.2.3 Anchor box association with ground truth defects 38

4.2.4 Region proposal network's multitask loss 39
4.3 Region of interest align operation 39
4.4 Network heads 40
4.4.1 RoI Head architecture 40
4.4.2 RoI Head loss function 41
5 FEATURE PYRAMID NETWORK 42
5.1 Motivation 42
5.2 Feature pyramid network enabled multi-scale feature extraction 42
5.3 Region proposal network with feature pyramid network 43
5.4 RoI heads with feature pyramid network 44
6 CASCADE HEADS 45
6.1 Motivation 45
6.2 Cascade R-CNN 45
7 PERFORMANCE METRICS 47
7.1 Intersection over union, precision, recall 47
7.2 Mean average precision, mean average recall 47
7.3 Correlation and mean absolute error 48
8 EXPERIMENTS AND RESULTS 50
8.1 Experiment with pavement patches 50
8.2 Experiment with downscaled orthoframes 54
8.2.1 Experiment with feature pyramid network-enabled Mask R-CNN 54
8.2.2 Experiment with cascade heads 58
8.3 Additional Testing 63
9 SUMMARY 68
9.1 Conclusions 68
9.2 Future works 69
KOKKUVÕTE 71
LIST OF REFERENCES 73

List of figures

Figure 2.1 Image-based road defect detection methods.....16
Figure 2.2 Example of ConvNet for handwritten digit classification 23
Figure 2.3 U-NET: an example of ConvNet based network for image segmentation ...24
Figure 2.4 Data annotation. 27
Figure 2.5 Graphical user interface of the annotation tool 28
Figure 3.1 Focus area and rotation pre-processing steps.....30
Figure 3.2 Representation of different defect types in the whole dataset. 31
Figure 3.3 Dataset distribution per category in training, validation, and testing sets..32
Figure 3.4 Example of orthoframe augmentation. 33
Figure 4.1 High-level architecture of MaskRCNN 34
Figure 4.2 ResNet-based backbone feature extractor. 35
Figure 4.3 Region Proposal Network's (RPN) architecture.....38
Figure 4.4 Network Heads architecture. 40
Figure 5.1 Different scales of transverse cracking defect type 42
Figure 5.2 Feature Pyramid Network's architecture 43
Figure 5.3 FPN-enabled RoI Head's architecture 44
Figure 6.1 COCO Instance segmentation leaderboard [34] 45
Figure 6.2 Cascade RCNN architecture 46
Figure 8.1 Loss during training with sliding windows approach. 50
Figure 8.2 Sliding window approach: bounding box based average precision on the validation set. 51
Figure 8.3 Sliding window approach: bounding mask based average precision on the validation set. 52
Figure 8.4 Sliding window approach: intact ground truth(left) vs prediction (right) ...52
Figure 8.5 Sliding window approach: example of a good prediction(left) versus ground truth(right). 53
Figure 8.6 Sliding window approach: another example of good prediction (left) versus ground truth (right). 53
Figure 8.7 FPN-enabled Mask R-CNN: training loss.....55
Figure 8.8 FPN-enabled Mask R-CNN: box-based AP on the validation set 55
Figure 8.9 FPN-enabled Mask R-CNN: mask-based AP on the validation set. 56
Figure 8.10 FPN-enabled Mask R-CNN: example of network's output(left) correcting the manual annotation(right).....57
Figure 8.11 FPN-enabled Mask R-CNN: example of neighboring defect instances successfully predicted 57

Figure 8.12 FPN-enabled Mask R-CNN: a second example of neighboring defect instances successfully predicted 58
Figure 8.13 Foreground accuracies in cascade headers. 59
Figure 8.14 Categorical box based mean average precision for cascade headers 59
Figure 8.15 Categorical mask based mean average precision for cascade headers 60
Figure 8.16 Comparison (1) of results between standard head's (left) and cascade head's predictions (right) and ground truth orthoframe (bottom). 61
Figure 8.17 Comparison (2) of results between cascade head (left) and standard head's (right) predictions and ground truth data (bottom). 62
Figure 8.18 Example of the predicted defects integrated into QGIS software. 64
Figure 8.19 Metric correlations for defect types and overall defectiveness. 66
Figure 8.20 Example of weathering not detected by the network. 67
Figure 8.21 Example of predictions of network cracking compared to the annotations. 67

List of tables

Table 2.1 Comparison of road defect detection methods 22
Table 3.1 Defect types and characteristics 29
Table 3.2 Orthoframes augmentation techniques used for training 33
Table 4.1 Common activation functions used in ConvNets: sigmoid, tanh, and ReLU .36
Table 8.1 Comparison of results between cascade and standard heads 60
Table 8.2 Benchmarking of results with different network configurations 63
Table 8.3 Roads' descriptions for additional testing 63
Table 8.4 Correlation and mean absolute error. 64

PREFACE

I would like to express my sincere gratitude to Roland Lõuk, MSc, a data scientist at EyeVi who previously completed his master's degree within the same team and is currently working on pavement defect detection for his help mainly on getting started with the data and the task and performing a test under realistic scenarios.

List of abbreviations

AP	Average Precision
AR	Average Recall
BEMD	Bidimensional Empirical Mode Decomposition
CFD	Crack Forest Dataset
CNN	Convolutional Neural networks
ConvNet	Convolutional Neural networks
CTA	Conditional Texture Anisotropy
FCN	Fully Convolutional Network
GNSS	Global Navigation Satellite System
GPU	Graphical Processing Unit
IoU	Intersection over Union
KNN	K Nearest Neighbors
MAE	Mean Average Error
mAP	Mean Average Precision
mAR	Mean Average Recall
NMS	Non-Maximum Suppression
ReLU	Rectified Linear Unit
RoI	Region of Interest
RoIAlign	Region of Interest Alignment pooling operation
SVM	Support Vector Machine
VRT	A special type of file used to store geospatial information

1 INTRODUCTION

Defected roads are characterized by distress on the road material and a deformed shape different from the original. Road management and maintenance consist of implementing cost-efficient strategies for the measurement and assessment of road defects. Cost-efficient strategies imply an early assessment of pavement defects and maintenance of roads which traditional ways employing human labor fail to do. In regions with extreme temperatures such as Estonia, the effects of such temperature can intensify defects on roads, requiring automatic and timely inspection of road pavements.

The project named "Applied Research for creating a cost-effective interchangeable 3D spatial data infrastructure with survey-grade accuracy" [1] is an applied research project to create a next-level 3D spatial database layer and to integrate it into existing databases. One of the key tasks in the project is the use of artificial intelligence to automate road defect detection most efficiently. Previously, within the same project, a pipeline of three convolutional neural networks (ConvNet) has been proposed to identify road areas from orthophotos [2], classify road segments as defects or intact [3], and segment pavement defects in road segments [4].

However, individual networks of the proposed pipeline either classify road segments according to the class they belong to i.e., classification, or group all defective pixels of the same defect type together i.e., segmentation. Moreover, with a combination of classification and segmentation as a pipeline, it is still not possible to separate individual defects if they are bordering each other in pavement orthophotos.

The goal of this thesis is to research and implement a deep neural network-based technique for road defect instance segmentation. Instance segmentation is a computer vision task used to identify instances of objects (pavement defects such as potholes, cracking, repaired patches, or weathering) in images at a pixel level. Using orthophotos provided by the project partner EyeVi, a spin-off of Reach-U Ltd. [1] and labeled with defect types and masks by trained personnel plus computing resources provided by the Department of Software Science at Tallinn University of Technology, an instance segmentation network sharing classification and segmentation tasks was researched and implemented. By generating instance-level semantic segmentation masks for each defect in roads' orthophotos, the network successfully resolves the previous limitation of not being able to distinguish individual neighboring defects.

Chapter 2 begins with a comparative study of visual imaging-based road defect detection methods followed by an overview of convolutional neural networks for different vision tasks and specifically for instance segmentation. Finally, this chapter also describes the data acquisition and annotation processes.

Chapter 3 describes the dataset used and the pre-processing steps including data partitioning and data generation for training.

Chapter 4,5 and 6 describes modeling approaches and network configurations used for road defect instance segmentation. The baseline network Mask R-CNN is explained in chapter 4, in chapter 5 an improvement on the baseline using pyramidal features maps is explained. Chapter 6 gives a detailed description of how the network is further improved with a cascade of 3 detection heads.

In chapter 7, performance metrics such as mean average precision (mAP) used as standard in literature are described; in addition, the chapter also introduces correlation and mean absolute error used to access the performance of the solution under realistic conditions following the Estonian Road Administration's [5] definition of pavement defectiveness.

Chapter 8 presents different experiments and results with each network configuration. Illustrative examples and comparisons between approaches are given. The chapter also presents a benchmarking result of all modeling approaches.

In chapter 9, further testing under real-life scenarios is made on previously unused 16.6km of Estonian road. The results, the integration of results into existing pavement defect software modules, and performance using metrics following the Estonian road administration's recommendation are given. Case by case analysis is also performed to show the strengths and weaknesses of the proposed solution.

2 LITERATURE REVIEW AND BACKGROUND

2.1 Road defect detection methods

Methods for road defect detection can be classified based on the data used i.e., radar technology, 3D image, image, etc. In this thesis, methods using visual imaging are of particular interest since the data to be used is made of orthophotos as described in [6]. Road defect detection methods relying on visual imaging can be grouped in three categories based on the characteristics of methods used [7]: hand-crafted road defect detection systems, classical machine learning-based methods, and deep learning-based methods. Figure 2.1 gives an overview of each group; methods in each group may deal with one or many computer vision tasks (i.e., object detection, classification, segmentation), and can be applied to one or many road defect types (i.e., cracks, potholes, weathering).

2.1.1 Hand-crafted methods

This group of methods is based on experts' understanding and interpretation of specific defective roads. Rule-based methods rely on visually or statistically noticeable edges, topology, morphology features in road images to define the rules which are used to classify or segment defects in roads' images. It is also common for such methods to preprocess the images to reduce the noise captured during data acquisition.

For example, adaptive threshold-based segmentation is used in [8] to produce candidate defected pixels from roads' images. First, lane-markings in the image are removed using a binary threshold; the color of road marking is previously known to be white. Hough transforms are applied on top of binarized images to find lines with certain dimensions and orientations corresponding to actual characteristics of lane markings. Then, a textural measure called Conditional Texture Anisotropy CTA is used to dynamically separate defected pixels from intact ones [8]. This method is further improved with a machine learning method to successfully classify a total of 729 intact, crack, joint, and bridged road images with an accuracy above 90% [8].

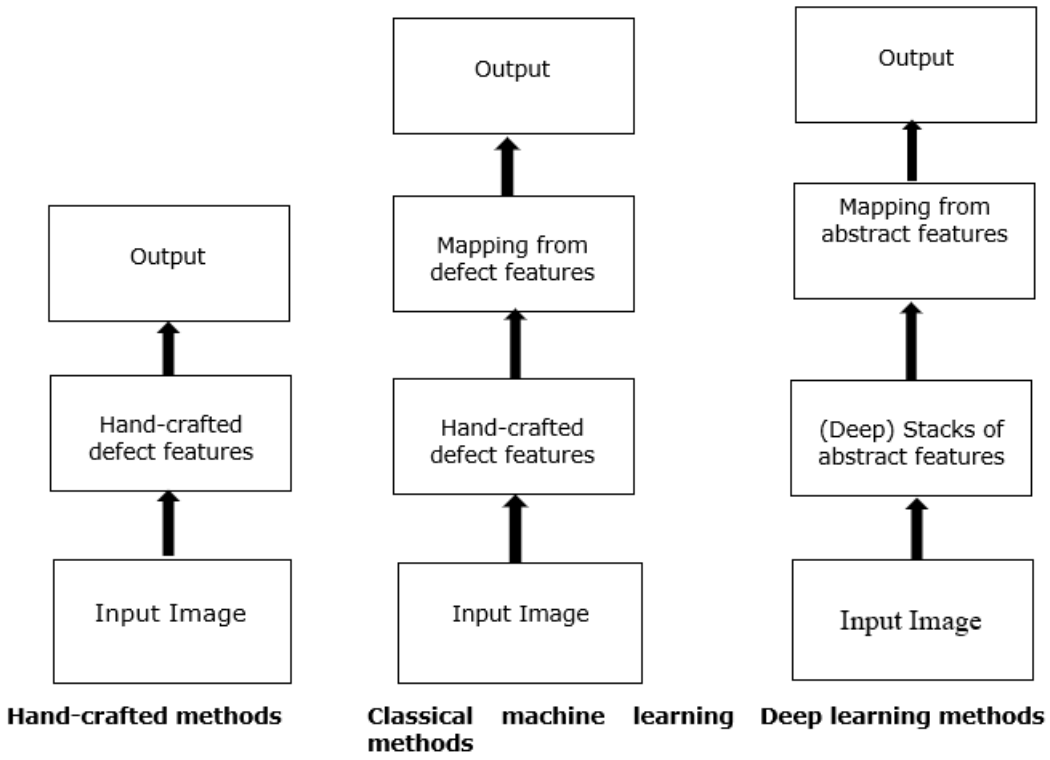


Figure 2.1 Image-based road defect detection methods

Another example of a hand-crafted method is in [9] where an ensemble of techniques is used to distinguish non-crack roads from superficial cracks, crocodile cracks, linear cracks, and transverse cracks. Road images are first converted to grayscale color space, blurred with a median filter, binarized, and transformed via morphological operations, erosion, and dilation in particular. Subsequently, features such as the total number of pixels in processed images, the ratio of the crack area over the entire area, the number of objects in processed images, the direction of objects in processed images were used to classify whether the defect type is intact, a superficial crack, a crocodile crack, or a horizontal crack, using a decision tree. Furthermore, the method provides possibilities to customize the rules on the run to adapt them to roads under different environmental conditions [9]. While the performance of the algorithm was not evaluated on an existing road defect dataset, for each detection, the classifier outputs a confidence score for all defect types, and the class with the highest score is selected.

The last example of a rule-based road defect detection is described in [10] where road surface images were first processed either by Gaussian filter or Bidimensional Empirical Mode Decomposition (BEMD) for noise filtering. Next, the Sobel edge detector was

applied to detect cracks in road images - a pixel is labeled as a non-edge defect or edge defect, based on the gradient magnitude and direction at a current location (pixel) and both sides of the pixel. The method was tested on a total of 15 images with different surface properties; overall the two methods successfully detected edge defects but BEMD was more efficient on plain cement concrete road images while the Gaussian filter yielded better results on asphalt roads images [10].

2.1.2 Classical machine learning methods

A known disadvantage of a rule-based system is the generalization problem: it is very common for such systems to fail to generalize to new unseen scenarios since the predefined rules do not apply anymore. Machine learning methods remedy that by allowing the systems to use the representation of defects and learn the mapping of the representation to desired outputs over a given dataset. As opposed to the rule-based methods described in 2.1.1, classical machine learning methods learn not only the defect representation on edge, topology, or morphological features but also the mapping from learned representations to desired outputs over a whole collection of road images. Usually, this yields better generalization to unseen images compared to hand-crafted methods [7].

In [11], CrackForest, a machine learning method based on structured random forests was developed to detect cracks in road images. CrackForest can be divided into three phases: integral channel features extraction, structural learning with random forest, and cracks description plus detection. Integral image patches were extracted with a sliding window of size $16 * 16$ from training data; patches with cracks edge at the center were labeled as positive and used in further steps. For each image patch, a mean intensity value m and a $16 * 16$ standard deviation matrix are used as features. Additionally, 13 integral channels (3 colors, 2 magnitudes, 8 orientations channels) were applied to the $16 * 16$ image patch to generate 16 $(13 * 16) = 3328$ candidate integral channel features; each feature contains a specific representation of cracks. The last features are textural features computed on a $m * m$ matrix where m is the mean pixel value and yields 300 more features for each channel. Random forests are then used to learn the mapping between the set of extracted features and patch annotation masks; after this step, image patches can be labeled with their semantic masks, but these masks were only considered as preliminary results as they contain textural noise. Predominant features from the previously extracted set are reused to compute statistical (occurrence of features) and neighborhood (co-occurrence of two features)

histograms. A mapping is learned between the normalized occurrence and co-occurrence information and the annotated defect type, using a set of KNN, SVM classifiers. The classifiers can distinguish cracks from textural noise. This, added to the semantic map produced by the random forest learners, made CrackForest surpass crack detection methods such as Canny, CrackTree, and CrackIT [11] on Crack Forest Dataset CFD [11] which contains 118 annotated images of Chinese urban roads acquired with a mobile phone. CrackForest shows a precision of 82%, a recall of 89%, and an F1 score of 85%.

The second example of a machine learning method in road defect assessment is [12] where SVM was used to classify features extracted from non-overlapping patches of road images. Each training image was divided into smaller patches; from each image, patch, top-hat filter, mean and top-hat filters, minimum filter, and adaptive histogram equalization were used to remove noise. Features such as intensity level, mean, variance, and different order of moments were used to train an SVM classifier. The system was then tested on two datasets containing respectively 56 road images of size $2048\,x\,1536$ and 165 road images of size $2048\,x\,4096$. On the first dataset, the method achieves a recall of 98% and 94% on the second dataset. However, if trained on a set of features coming from another data distribution, its performance suffers i.e., if trained on a set of features coming from the first dataset distribution, the method performs worse when tested on the second dataset and vice versa.

The last example in this category classifies road images using graph-based features [13]. First, images are pre-processed to remove noise by applying Hough transform to eliminate circular and rectangular shapes, which may look like road defects while they are not. Then a local filter is applied, followed by a Gaussian filter. The resulting images are used to define a graph object for each defect. Features based on the graph's properties are extracted and used as input to an SVM classifier which outputs the type of the defect among 5 possible values: longitudinal crack, transversal crack, mixed cracking, alligator cracking, and healthy road. System performance was tested on a dataset consisting of 525 road images with a precision of 86%, a recall of 85%, and an F1 score of 85%.

2.1.3 Deep learning methods

Classical machine learning methods for road defect detection rely heavily on hand-crafted defect features introduced by experts to learn a representation and mapping of defects. While these features may be enough to characterize defects in particular restricted cases, in general, there are not: not all defects are visually perceptible. Some

features can even be abstractions difficult or impossible to define mathematically. For example, despite being actively researched within the computer vision community, the visual texture still has no precise definition; there are only attempts to describe texture [14]. Deep learning methods introduce a deep stack of features, from simpler to more general ones; by allowing more general features to be derived out of simpler ones, deep learning-based defect detection methods can learn complex defect features that cannot be necessarily defined by an expert or a human operator. Methods in this category are mostly empowered by convolutional neural networks (ConvNets) (section 2.2) and prone to a smaller generalization error.

The first example of a deep learning method for defect detection uses orthophotos collected with Ladybug 5+ by Reach U Ltd. In Estonia, to train a deep convolutional neural network to distinguish intact orthophotos from defected ones [6]. Road orthophotos of size $4096 * 4096$ were partitioned into small redundant patches which were augmented by random brightness, contrast, horizontal and vertical flips and rotation flips. In total, 3 different network architectures were tried and a ResNet with 101 layers was reported to have the best performance i.e., an accuracy of 97%, a precision of 90% and a recall of 87% on a test dataset consisting of 1007 defect-free and 185 defected road images patches [6].

An improvement of [6] was proposed in [3] where context-awareness is enhanced by an extra input stream to the network. This method was motivated by the fact that ConvNets work better with images of smaller size which are not often appropriate for detecting objects from high-resolution images. In addition to the segment containing the defect referred to as content, a larger area segment (referred to as context) surrounding the content is also cropped and resized to the content's size. Two ResNets are used to extract features from content and context image patches, the extracted features are then fed to a classifier consisting of three fully connected layers; in the last fully connected layer, features from context and content image patches are merged to produce the probability of a segment being a defect or not. The performance of the method was evaluated with different network architectures and compared to performances of single-stream networks; the highest MCC score of 91% on a test set consisting of 438 orthophotos of size $4096 * 4096$ was with a context-aware network while the best opposing[3].

An example of an engineered system for road defect detection is described in [4]; it introduces a pipeline of three different networks for defect detection in pavement orthophotos: the first network is used for generating road area masks from road

orthophotos, the second network is equivalent to the one described in [3] and performs road defect classification while the third network is for road defect segmentation. The output of the first network is used to generate image patches for the second network whose output is in turn used to select defective patches to segment by the third network. With this pipeline, road images' probability of being defective plus their semantic mask can be obtained. The system's performance has been tested on a set of 947 orthophotos of size $4096 * 4096$, the majority of which are from new unseen roads; an MCC score of 72% and a mIoU 51% have been reported respectively for defect classification and detection segmentation tasks.

In [15], a feature pyramid network boosted hierarchically has been proposed for crack detection. The network can be divided into four parts: a bottom-up network which consists of a succession of convolutions to generate features of different scales; a feature pyramid network based on a top-down architecture with upper layers providing contextual information to lower layers; side networks that use deep supervision-based learning to predict cracks at each layer of the network; and finally, a hierarchical boosting method based on reweighting adjacent samples from side networks. In the same work, the authors proposed a new method for evaluating defective roads in segmentation tasks: the Average Intersection Over Union AIU, or IoU calculated over different thresholds. The performance was compared to other methods including edge detection networks (such as HED) and fully convolutional methods on different defect datasets; their method outperformed the rest with an IAU of 49 % on the Crack500 dataset [16], [17].

In [18], two network models and a new dataset were proposed for the defect detection task. The dataset is made of 7237 images with 9 different classes used for training and testing. Two object detectors were used during training: YOLOv2 [19] and FasterRCNN [20]. Given an input image, the networks can output the bounding box and defect type. The performances of these networks were evaluated using precision, recall, and F1 score. With YOLOv2, the F1 score was 85% while FasterRCNN's F1 score was 65%.

Crack-Gan introduced in [21] is our last example of deep learning methods for pavement defect detection. It fixes the class imbalance and lack of ground truths data problems by using generative adversarial learning [22]. The method generates ground truth annotation with its adversarial module and performs pavement defect detection using an asymmetric version of U-NET. As result, the network can work with poorly annotated and imbalanced datasets to produce annotation masks for defective pavement images. The performance of Crack-Gan was compared to other methods such

as CrackIT, CrackForest; while the method does not always have the highest precision and recall values, it has the best computational cost and solves class imbalance and lack of ground truth training data.

2.1.4 Summary of road defect detection methods

Table 2.1 recapitulates different types of road defect detection methods, their characteristics, advantages, and disadvantages. Due to the latest advances with ConvNets, deep learning defect detection methods have the advantage of being able to generalize well to unseen roads images without the need for expert knowledge but need more annotated images and computing resources; however, machine learning defect detection methods can also be sufficient in restricted scenarios though they require more effort in feature extraction.

Table 2.1 Comparison of road defect detection methods

Category	Characteristics	Advantages	Disadvantages
Rule-based methods	Use defect features based on the edge, morphological, topological, and textural properties of defects. Ex: Morphological features + decision tree [9]; BEMD or Gaussian filter + gradient based thresholding [10]	No need for a dataset. Fast and light.	Generalization problem.
Classical machine learning methods	Use defect features based on the edge, morphological, textural, topological properties of defects. Learn to map defect features with desired outputs. Ex: CrackForest [11], Graph-based features + SVM [12]	Can perform well under restricted cases. Can catch well-known and well-described defect features.	Needs a training dataset. Generalization problem.
Deep learning methods	Learn deep stacks of abstract defect features independently. Learn to map learned features with the desired output. Ex: ConvNet based defect classification and segmentation [3], [4], [23],[6], [15], CrackGan [21]	Can generalize quite well to unseen cases. No need for domain knowledge to extract defects features.	Needs a relatively large training dataset. Needs more computing resources. The black box and non-interpretability issue.

2.2 Convolutional neural networks

Convolutional neural networks, also called CNNs and referred to in this work as ConvNets, are a special type of neural network that owns its name to the mathematical operation called convolution; this type of network is good for modeling data with grid-like topology referred to as tensors in machine learning terminology. The concept of ConvNets dates back to 1950 but one of the first real uses of ConvNet for image classification is from 1998 with the well-known LeNet [24], [25] by Yann LeCun for

handwritten digits image classification, using a dataset of 60k training images of size 28 x 28.

ConvNets were forgotten till 2012 when Alex Krizhevsky reduced the error rate from 26% to 15% in the well-known ImageNet [26] classification challenge with his ConvNet model called AlexNet [27]. AlexNet was trained on a set of approximately 1.2 million images of 1000 different classes for the first time on GPU machines. The error rate was reported on a test set of 150.000 images.

Based on the task performed, ConvNets can contain different layers. For example, it is common to have a succession of convolution, pooling, and activation layers followed by fully connected layers in the classification task. An example of a ConvNet for handwritten digit classification is shown in Figure 2.2.

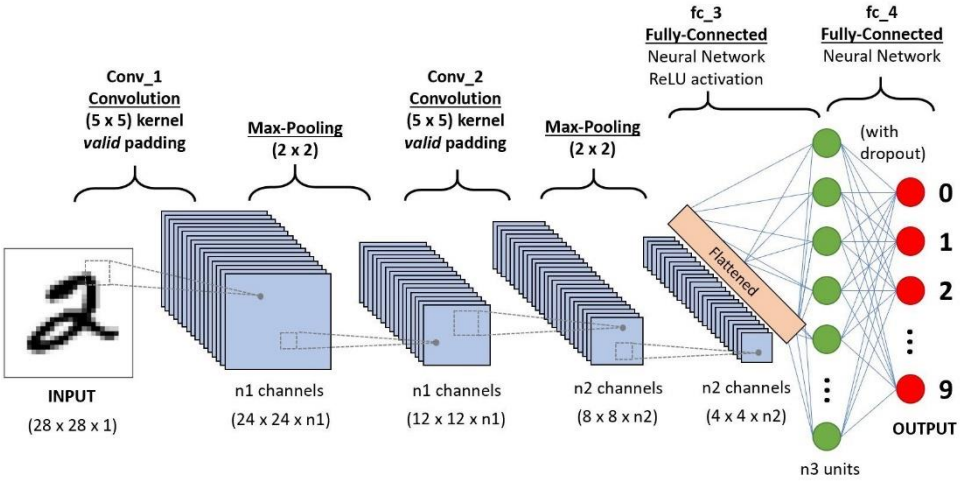


Figure 2.2 Example of ConvNet for handwritten digit classification [27]. *Conv* denotes the convolutional and *fc* for fully connected layers.

In Figure 2.2, an image of a handwritten digit of size $28 * 28$ is fed as input to a succession of convolution and maximum pooling layers; the output of the last pooling layer is activated with ReLU before being fed to fully connected layers to predict the type of the handwritten digit.

Many ConvNet based network design approaches have been proposed for the semantic segmentation task, a common and successful one uses fully convolutional layers. U-NET [28] is a fully convolutional network that was first proposed in 2015 for medical image segmentation. As shown in Figure 2.3, U-NET has two parts: an encoder or the contracting part consisting of a succession of convolution plus ReLU activation and

maximum pooling operations; the decoder or expansive part that is a succession of up-sampling, up-convolution, concatenation, and two convolution operations followed by ReLU activation at the end (See Figure 2.3).

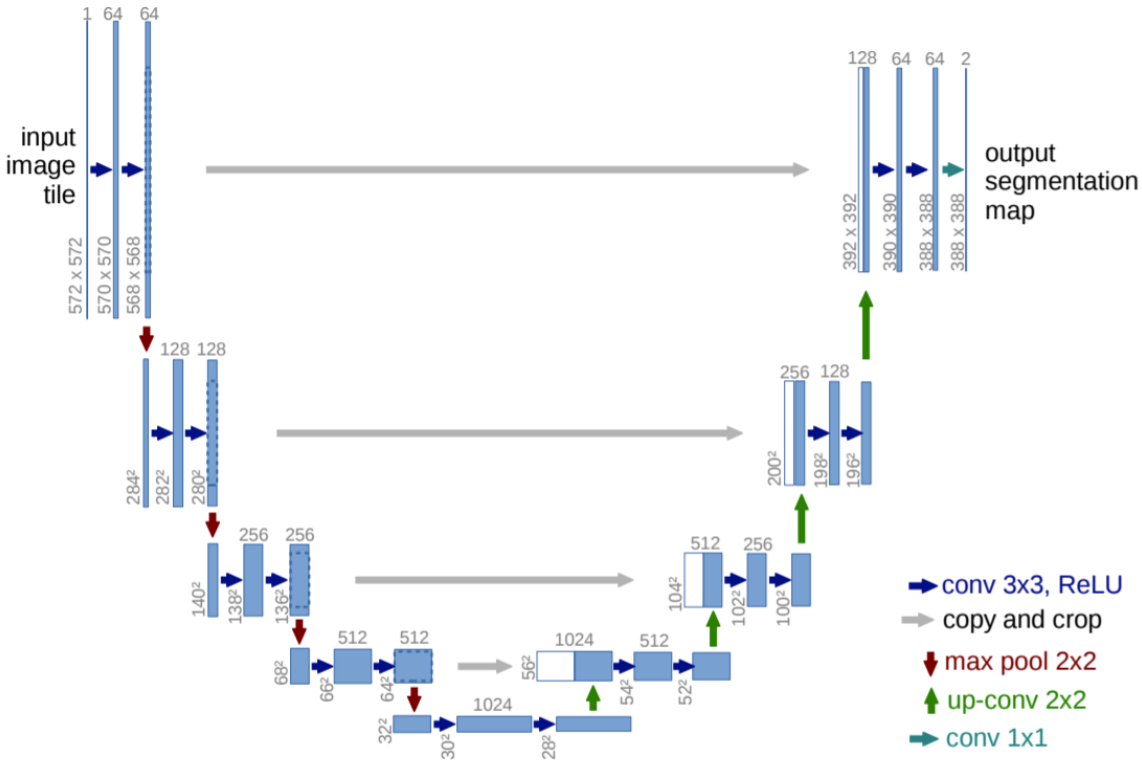


Figure 2.3 U-NET: an example of ConvNet based network for image segmentation [28]

However, the usage of ConvNet is not only limited to image classification and segmentation, but many other computer vision tasks also including image tagging, object detection, human pose estimation, instance segmentation, and panoptic segmentation are now being tackled with variants of ConvNets [24], [29].

2.3 Instance segmentation

In computer vision and image processing literature, object detection consists of detecting instances of objects of a certain class in images or videos; semantic segmentation consists of grouping pixels belonging to the same class. Instance segmentation is a challenging task that does both object detection and semantic segmentation by labeling each pixel with a class label and an instance label. ConvNets based solutions for instance segmentation can be grouped into two classes: single and two-stage instance segmentation.

2.3.1 Two-stage instance segmentation

This group of methods is characterized by networks with two stages, the first stage generates regions of interest using methods such as selective search or even more complex methods such as region proposal network. Generated regions of interest are evaluated to keep only the good ones. In the second stage of the network, the selected proposals are classified, regressed, and semantically segmented with methods that vary based on whether the detection and segmentation are run in parallel or succession.

In [30], a network for simultaneous detection and segmentation using Support Vector Machine (SVM) as a classifier is presented. The network is divided into four subsystems: a feature extraction with ConvNet plus region proposals generation followed by a proposal classification step using SVM and finally a region refinement with Non-Maximal Suppression (NMS).

One of the most successful two-stage instance segmentation methods is Mask RCNN [31], which extends RCNN [32], Fast RCNN [33], and Faster RCNN [19] by adding a mask predictor in addition to classification and regression heads present in Faster RCNN. In Mask RCNN, the three tasks share the same backbone layers, only the heads are task-specific and during training, the tasks are all learned together. This results in better performance.

Since its appearance, improved versions of MaskRCNN have been leading the COCO [34] object detection challenge. However, MaskRCNN and other double stage networks are slow during inference: on MMDetection's [35], MaskRCNN implementation runs with an inference speed between 8fps and 16.1 fps, depending on the depth of the feature extractor, making it not a preferable choice for real-time applications. Moreover, some two-stage networks utilize an additional anchor generation step, which needs to be manually tuned based on objects' geometric properties. Such models might not generalize well for unseen data with different object shapes and need further post-processing.

2.3.2 Single-stage instance segmentation

Single-stage networks for instance segmentation have been propelled by advances in proposals-free object detection methods. Methods in this approach run end-to-end instance segmentation without the necessity for intermediate object bounding boxes.

These methods are underexplored compared to two-stage algorithms which first detect the bounding boxes and then classify, crop and segment the regions [36].

Fully convolutional Instance Aware Segmentation [37] is a Fully Convolutional Network (FCN) for instance segmentation running end-to-end. Inspired by the results of FCN for semantic segmentation, the network first uses an FCN to generate score maps, which will be further processed by an assembling module and a set of spatial operations. On a K40 GPU machine, it can predict instance mask with a speed of 0.24s per image and its performance is still lower (on COCO dev-set detection challenge) than the performance of many two-stage networks such as Mask R-CNN.

TensorMask is another example of a single-stage instance segmentation network where the performance is closer to Mask RCNN on COCO benchmark [36], TensorMask comes with a new view of the instance segmentation task: dense instance segmentation is considered as a prediction task over a 4D tensor where instance-level masks are computed without the need for object detection. While the results are not as satisfactory as they are with Mask RCNN, TensorMask can lead to further research in dense sliding windows-based instance segmentation.

One of the main advantages of single-stage instance segmentation is its ability to run fast, this is especially useful for real-time applications such as autonomous driving. Recent works on single-stage instance segmentation such as YOLACT [38] and SOLO2 [39] can offer better speed-accuracy trade-offs, however, two-stage networks are still the leading solutions in object detection challenges such as COCO.

2.4 Pavement data acquisition

The project partner, EyeVi, an expert in geographical information systems has developed a mobile mapping system (MMS) for large-scale automatic collection of pavement images. The system is built on top of a car and uses 5 high-resolution LadyBug 5+ sensors that capture panoramic images, a Global Navigation Satellite System (GNSS) that provides accurate localization information to the panoramic images, and a Light Detection and Ranging (LIDAR) sensor. The cameras have a spherical field of view of 360°, providing a pixel average accuracy of 2mm from a distance of 10m and can work under temperatures ranging from -20°C to 50°C making it a good choice for Estonian weather. The LIDAR system is not of interest for this thesis project which solely focuses on visual imaging data acquired by the cameras.

The panoramic images captured by LadyBug 5+ are further processed by EyeVi to generate orthographic pavement images of size $4096x4096$ referred to as orthoframes in this work.

2.5 Data annotation

Orthoframes acquired using the mobile mapping system are manually annotated to produce data to be used to train AI algorithms. Previously, the team has developed a fast and easy-to-use annotation tool [4][40]. The tool provides a graphical user interface to allow trained personnel to annotate pavement defects in the area of the paved surface in orthoframes. The tool allows digitizers to select folders containing orthoframes to annotate; using the smart brush functionality, they can easily segment an area on the image and assign its defect type. The tool generates a grayscale mask file for each orthoframe where each color in the mask represents a particular predefined defect type such as a pothole, weathering, transverse cracking, etc. The same thing can be done for the paved area of the road to generate a road mask area where the paved area of the orthoframe is represented by white pixels. Examples of an input orthoframe and annotated masks are shown in Figure 2.4 whilst the annotation process is shown in Figure 2.5.

Optionally, if previous annotations of orthoframes exist as shapefiles or defect masks, digitizers can load them together with corresponding VRT files to speed up the annotation process (VRT files are a special type of file used to store geospatial information of each orthoframe, they are necessary for mapping each pixel in orthoframes to a real point on the road). This functionality is especially useful while using predictions made by the AI mentioned in [4] to accelerate the digitization of new unseen roads.

Figure 2.4 Data annotation. On the left, the digitized greyscale defect masks with defects of different types; in the middle, the orthoframe, and on the right, the paved road area mask.

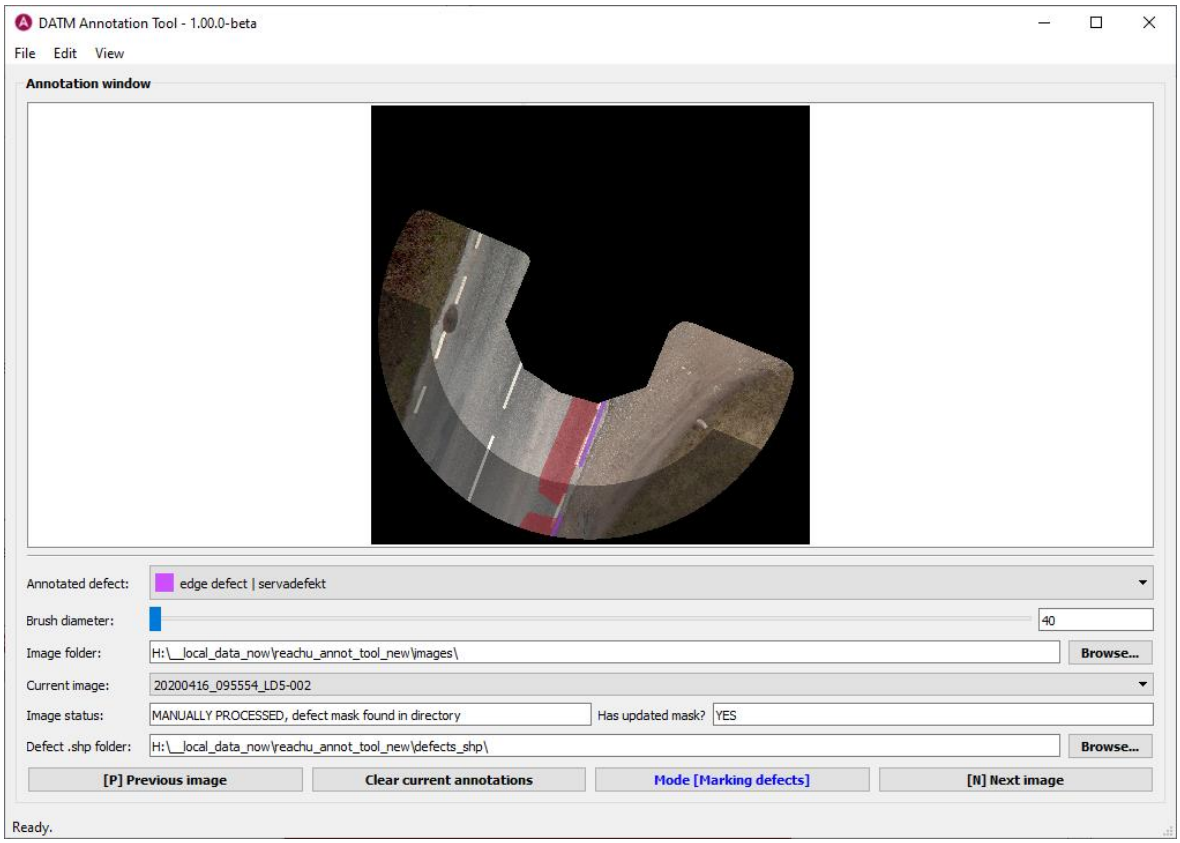


Figure 2.5 Graphical user interface of the annotation tool

3 DATA PREPARATION

3.1 Data analysis and partition

In total, 19046 orthoframes of size $4096 * 4096$ were annotated with defect types by trained personnel employed by the Department of Software Science at Tallinn University of Technology.The annotators have previous experience in geoinformatics. There exist seven defect types presented in Table 3.1.

Table 3.1 Defect types and characteristics

Name in English	**Estonian shortcode**	**Characteristics**
Longitudinal Crack	KPIKIPR	Defects parallel to the centerline of the paved road area.
Transverse Cracking	POIKPR	Defects perpendicular to the centerline of the paved road area.
Joint Reflection Crack	KVUUK	Defects in-between two surfaces of different levels.
Weathering	MUREN	Defects characterized by surface erosion which become visible with time.
Pothole	AUK	Defects creating a hole in pavements, usually have a circular shape.
Patched Road	PAIK	Represents pavements area repaired by patching.
Network Cracking	VORK	Defects caused by fatigue of the road material, usually have the shape of a spider web.
Edge Defect	SERVA	Defects lying on the edges of the paved road area.

Since consecutive orthoframes can capture the same part of the road, special attention was paid to how the data is split into training, validation, and testing sets. Instead of random sampling, we opted for a step-based sampling where out of each N consecutive orthoframes, n_1 consecutive orthoframes are selected for validation, n_2 for testing and the rest goes to the training set. A value of $N = 100$, $n_1 = 15$ and $n_2 = 10$ were used. This

approach allows having fewer overlapping orthoframes in training, validation, and testing sets compared to random sampling.

However, edge defects are heuristically assigned to defects close to edges of the road and never predicted by a deep learning model. The assignment algorithm is out of the scope of this thesis.

3.2 Data generation for instance segmentation

As observed in [6], pixels far from the shooting camera lose details and are not useful in distinguishing defects of different types. Therefore, the same pre-processing operation explained in [6], referred to as a focus operation, is applied to orthoframes and masks to cut off the area of the image far away from the camera. The focus operation shown in Figure 3.1 does cause any loss of data since consecutive orthoframes are partially overlapping.

Another step found useful in the previous work was rotating the mask so that its centerline is parallel to the y-axis. The rotation step is applied to images before training and inference, predictions are de-rotated back after inference.

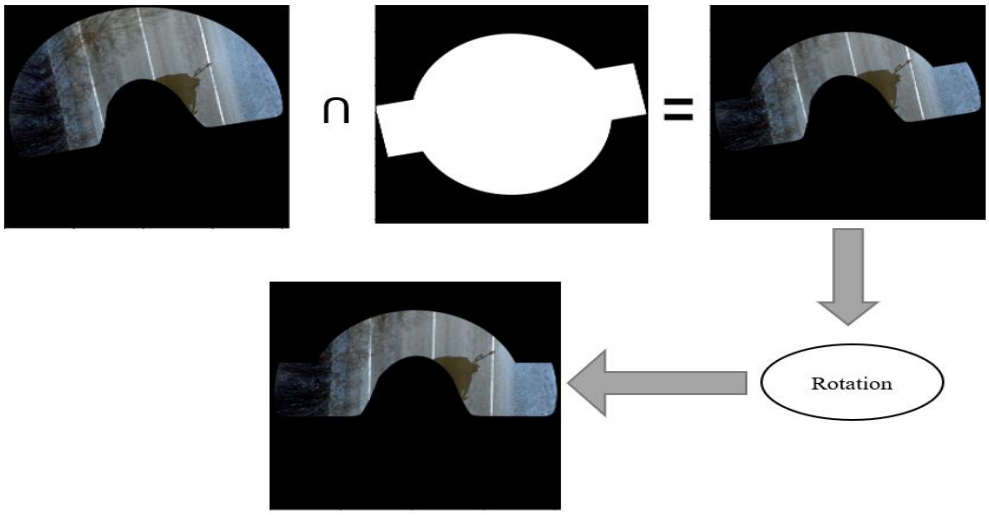


Figure 3.1 Focus area and rotation pre-processing steps.

The orthoframe on the left is multiplied by its focus mask (in the middle), the resulting image (right image) is then rotated (last image).

Due to memory limitations on the GPUs, we were either obliged to downscale orthoframes by a factor of k or use a sliding window of size $S * S$ to generate patches of images referred to as ortho-patches. A value of $k = 4$ was chosen for downscaling and $S = 512$ for sliding window size. The same resizing and downscaling operations were performed on both orthoframes and ground truth defect masks. With the sliding windows approach, masks patches with empty pixels (pixels with a value of 0) are ignored as they do not contain any information that might be used for learning.

At the time of writing this thesis, the benchmark for instance segmentation is the COCO challenge [34]. We adopted the same input format used in the benchmark where each instance of a defect is represented by its enclosing up-right rectangle bounding box, its mask represented as a list of points and its type is chosen between the ones listed in Table 3.1. By default, annotated defect masks are grayscale masks with each grey pixel corresponding to a defect type hence the need to process the raw data to extract each defect instance referred to as annotations. Annotations are saved following the COCO format and used as input to instance segmentation models.

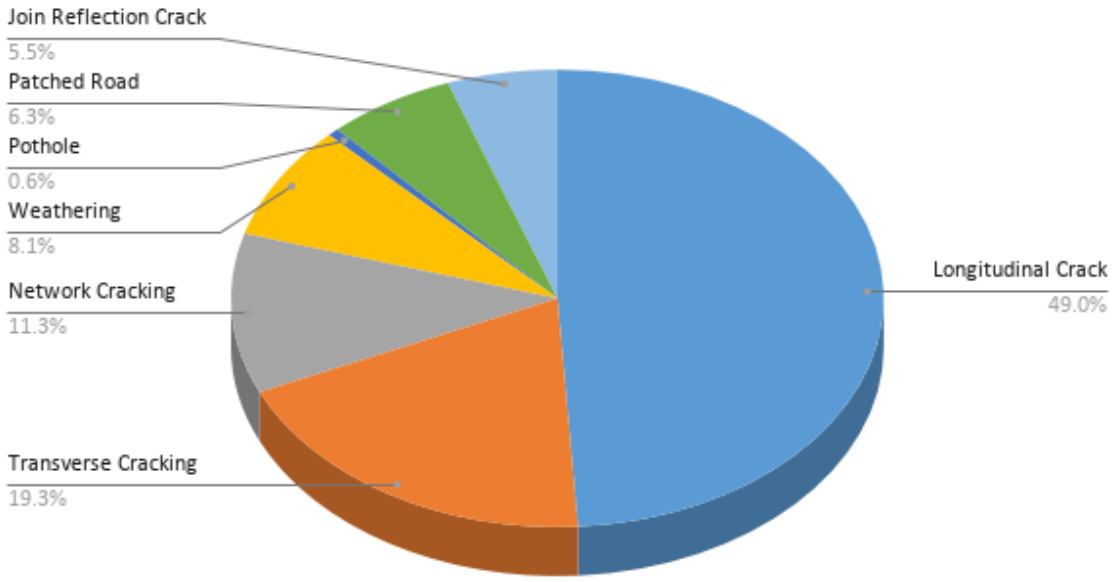


Figure 3.2 Representation of different defect types in the whole dataset.

Figure 3.2 shows that potholes are the least represented class with a probability of less than 1%, almost 1 out of 2 defects is of longitudinal type, transverse cracking is the second dominant class, followed by network cracking. Joint reflection crack, patched road, and weathering types represent less than 10% of the whole dataset. This highlights that data is highly imbalanced.

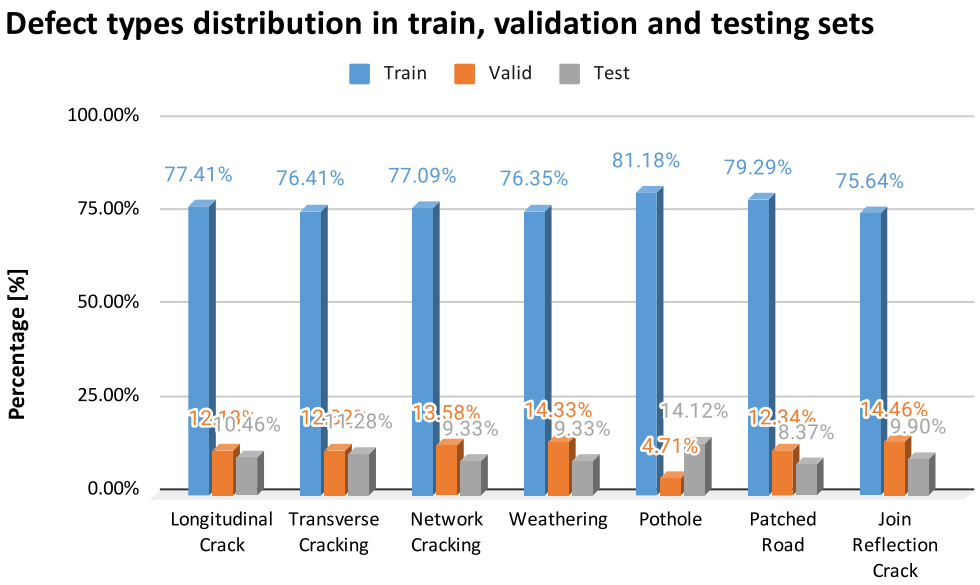


Figure 3.3 Dataset distribution per category in training, validation, and testing sets

The data split strategy adopted in 3.1 ensures that every defect type is equally represented in training, validation, and testing sets in terms of percentage, this can also be observed in Figure 3.3. Overall, the training set represents 76.5%, the validation set 13.5% and the testing set 10.1% of the whole dataset.

3.3 Data augmentation

The performance of deep learning models highly depends on the amount and quality of data used for training. With complex models made up of different stages, the role of data is even more crucial because as the number of neurons increases so does the number of trainable parameters [41]. Data augmentation is a well-known technique to avoid overfitting in deep neural networks: by generating new artificial data points from existing training data, the model is forced to learn new representations far away from the training points, thus minimizing the generalization error on new unseen samples.

For computer vision tasks, such as image classification, semantic segmentation, and object detection where the main input to models are images, data augmentation can be grouped into three groups: simple augmentations techniques, deep learning-based approaches, and meta-learning-based approaches. In this thesis, we focused on simple augmentations techniques, readers are referred to [41] for a detailed taxonomy of data augmentation techniques.

We used Albumentation [42], a fast cross-platform data augmentation library with support for different computer vision tasks and deep learning frameworks to add some comprehensive augmented images to the training set.

Table 3.2 Orthoframes augmentation techniques used for training

Data augmentation	**Usage Probability**	**Limit / Range**
Horizontal flipping	50%	-
Vertical flipping	50%	-
Shortest edge resizing	100%	(640,960)
Longest edge resizing	100%	1024
Center crop	10%	(512,512)
Random brightness	20%	(-20%, 20%)
Random contrast	20%	(-20%, 20%)

For random contrast and brightness, the limit values are the factor range for applying the transformation, for crop the limit sets the size for the width and height of the crop, and for edge resize, the size of the new image value corresponds to the given range.

We paid attention not to use transformations that might confuse the model such as rotation, since some defects, especially longitudinal and transverse cracking, have well-known directions on orthoframes plus the team already adopted a rotation policy for orthoframes as described in Figure 3.1. Examples of the above-mentioned augmentations randomly applied to orthoframes are shown in Figure 3.1together with the original image.

Figure 3.4 Example of orthoframe augmentation. The first image on the left is the original image, followed by random brightness, center crop, vertical flipping, and random contrast transformations, respectively.

4 MASK RCNN FOR PAVEMENT DEFECT INSTANCE SEGMENTATION

The baseline network used in this thesis is Mask R-CNN, this is justified by the fact that leading solutions in COCO object detection challenges are based on it [31]. The high-level architecture of Mask R-CNN is shown in Figure 4.1; it can be divided into three components: a backbone for feature extraction, a region proposal network that generates defect candidates with their objectness scores, and the headers which perform object detection (classification + localization) and semantic segmentation in parallel. By fusing outputs from the final classification, regression, and mask layers, we obtain instance-level masks and confidence scores for each defect (see Figure 4.1).

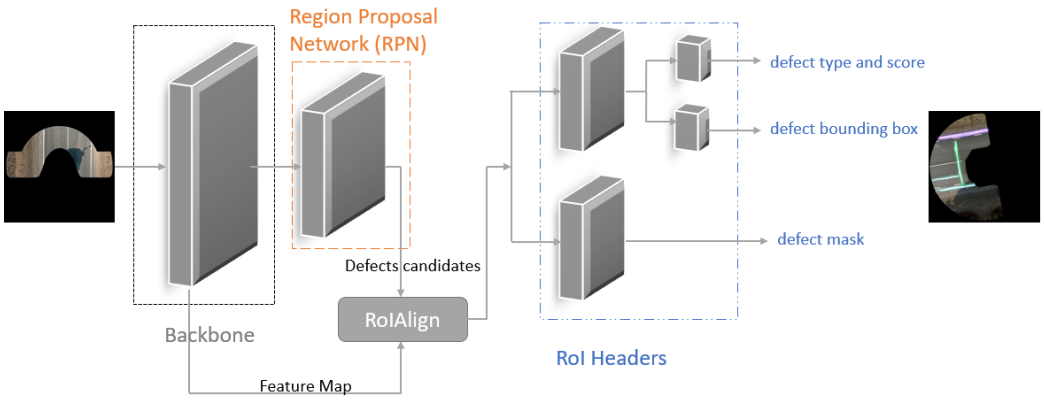


Figure 4.1 High-level architecture of MaskRCNN

4.1 Backbone feature extractor

The backbone feature extractor is a deep convolutional neural network that takes an input image and generates a feature map. Preference is given to deeper neural networks with high representational capacity such as ResNet which are known to perform better with computer vision tasks [43]. Deep residual backbone networks are made of a succession of residual blocks, the total number of layers (50,101) in the blocks defines the version of the network (ResNet50, ResNet101); residuals blocks contain a succession of ReLu activated convolutional layers with shortcut connections to introduce identity mapping. The last pooling and fully connected layers of original ResNets are removed in the backbone networks used in the thesis.

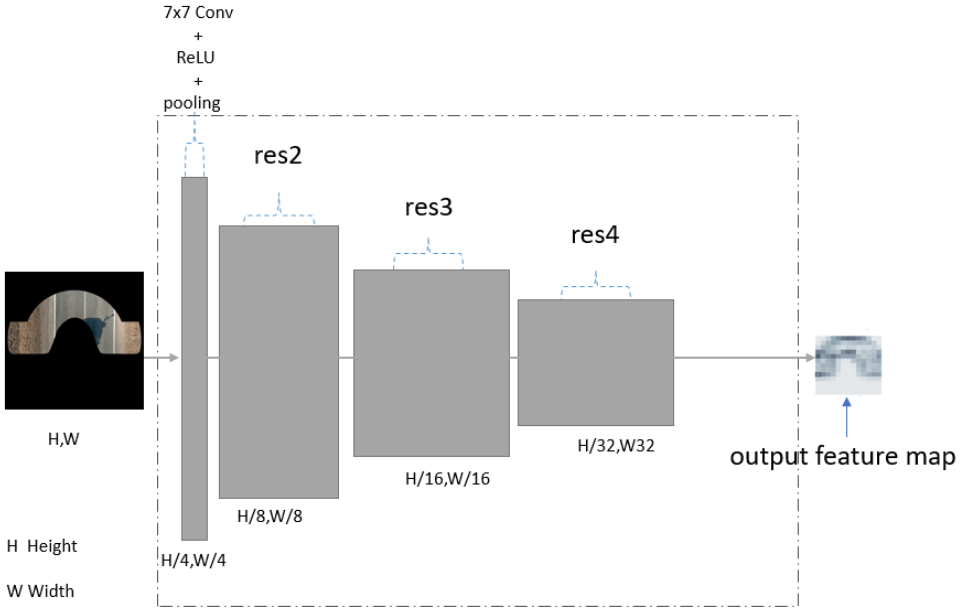


Figure 4.2 ResNet-based backbone feature extractor. *Res2*, *res3,* and *res4* are the names of residual blocks. Each block is made up of convolutions, activation, and normalization layers. Layers in the same block have the same spatial dimensions.

4.1.1 Convolution operation

A convolution is a mathematical operation denoted by an asterisk $*$ and performed on two matrices: I or the input data in form of a tensor and a kernel denoted as K. The convolution operation slides the kernel on top of the input data by performing element-wise multiplication between the kernel K and the part of the input I the kernel is currently on; the multiplication results at each sliding step are summed up into a single pixel. The sliding process is repeated over the entire input I to obtain a multidimensional matrix representing the weighted sums of input features; the output of a convolution is often referred to as a feature map.

The mathematical expression of a convolution is given below [24]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (4.1)$$

Where I - a two-dimensional tensor,

K the filter or kernel.

In ConvNets, filters are used to extract meaningful features from input images and the output of convolution can also be used with another filter for another convolution

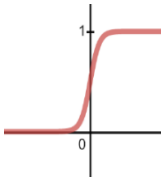
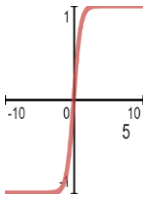
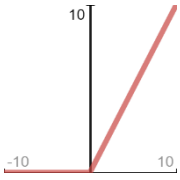
operation; considering the deep architectures of ConvNets, this results in extracting more perceptible or visually recognizable features from the original input.

In addition, the filter is moved across the image left to right, top to bottom, with the amount of movement controlled by the stride. In each block, the stride is set to 1 except for the first layer where the stride of 2 is used to reduce the feature map size by 2. Zero padding, on the other hand, adds zero pixels to the frame to counterbalance the size reduction caused by stride.

4.1.2 Activation function

The activation function is the second stage of a typical ConvNet layer and is referred to as the detector stage. At this stage, the output of a convolution operation is fed to a nonlinear activation function to allow the network to capture complex features and to scale features between certain range to reduce computational costs. Common activations functions found in ConvNets are resumed in Table 4.1.

Table 4.1 Common activation functions used in ConvNets: sigmoid, tanh, and ReLU [24]

Function	Graph	Formula
sigmoid		$\sigma(x) = \frac{1}{1+e^{-x}}$ (4.2)
tanh		$\tanh(x)$ (4.3)
ReLU		$ReLU = \max(0, x)$ (4.4)

Unless otherwise specified, ReLU is the activation function used in this thesis because of its similarity to linear units and ability to make the gradient direction in active units useful during the learning process [24].

4.1.3 Pooling operation

It is common to see pooling operations as the third stage immediately after activation function in ConvNets. Pooling functions substitute outputs of activation function with some summary statistics within a predefined neighborhood [24]. One example of such function is maximum pooling which replaces outputs at certain locations with the maximum possible output in the neighbor rectangle or average pooling which replaces the outputs at a given location with the mean pixel value of the neighborhood.

Pooling functions make modeling using ConvNet invariant to translations and improve computational efficiency by reducing the number of inputs to the next layer [24], [29]. The single pooling operation used before the first residual block is a maximum pooling operation. The regular max pooling found at the end of a ResNet layer is removed and not part of the backbone network for feature extraction.

4.1.4 Normalization

In deep ConvNets, normalization is used to standardize inputs of convolutional layers to a common scale using a predefined function. Normalizations are known to facilitate optimization and convergence of ConvNets; batch normalization is commonly used in computer vision tasks by subtracting the mean from input features and dividing by the standard deviation at the mini-batch level. However, with small batch sizes, batch normalization becomes inefficient and group normalization can be used [44]. It works independently of the batch size by performing group-wise normalization instead of batch normalization. We used group normalization in this thesis due to memory limitations not allowing us to work with larger batch sizes.

4.2 Region proposal network

The role of the region proposal network (RPN) is to use the feature map generated by the backbone feature extractor to generate a set of pavement defect candidates. The region proposal network was first proposed in Faster-RCNN object detection framework and then reused in Mask RCNN [31].

4.2.1 Region proposal network's architecture

The region proposal network is a simple fully convolutional network of a $3x3$ convolutional layer shown in Figure 4.3 as an intermediate layer followed by two parallel layers for objectness classification and box regression.

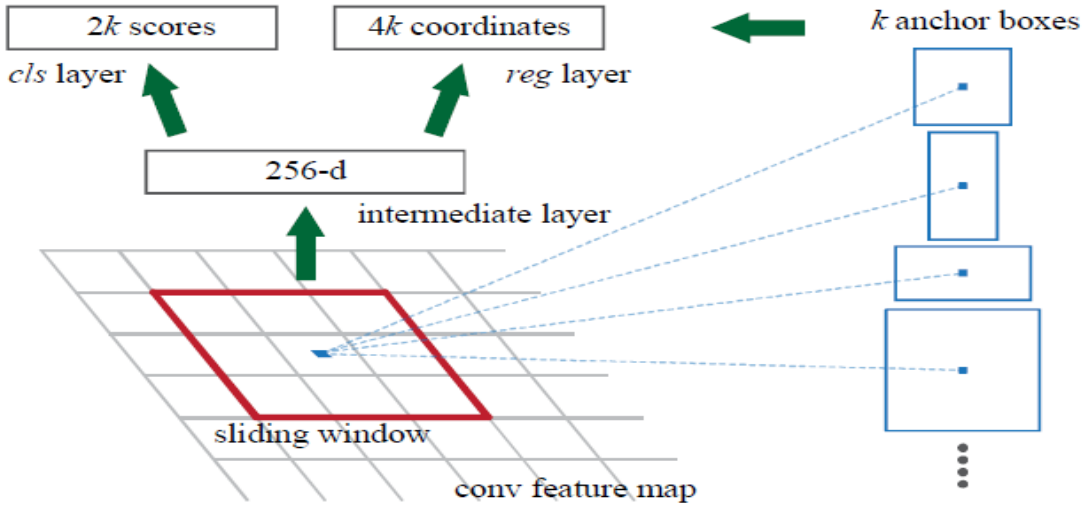


Figure 4.3 Region Proposal Network's (RPN) architecture [19]

The RPN is slid over a 3x3 input sampled from the feature map, and for each point on the feature map, it generates $2k$ objectness scores and $4k$ coordinates representing bounding boxes of k probable defects.

4.2.2 Anchor box generation

As shown in Figure 4.3, the RPN outputs k defect proposals for a given location in the feature map based on k anchor boxes used as reference. For each sliding window, the k anchor boxes are centered in the window and defined by predefined aspect ratios and scales; we first tried 3 aspect ratios (1:2,1:1, 2:1) and 5 scales (32x32, 64x64, 128x128, 256x256 and 512x512) which results in $k = 15$ anchor boxes for each 3x3 window.

If we consider the feature map at ResNet's layer *res3* mentioned in 4.1, the feature map will have an output of 64x64. In total, 64x64x15 = 61,440 anchors can be generated.

4.2.3 Anchor box association with ground truth defects

We need to select good anchor boxes to use as reference boxes during the learning phase in the region proposal network. For that, we calculate the intersection over union (IoU) of each anchor with the ground truth defects' boxes; anchors with an IoU higher than 0.7 are labeled as positive or foreground anchors, anchors with an IoU less 0.3 are labeled as negative or background and anchors with an IoU between 0.3 and 0.7 are

ignored and not used for training. With this association rule, most of the anchors are background and a ground truth defect box can be associated with multiple anchor boxes.

4.2.4 Region proposal network's multitask loss

The RPN's loss function [19] is a combined function of objectives for classification and regression.

$$L = \frac{1}{N_{cls}} \sum L_{\log}(p_i, p_i^*) + \frac{\lambda}{N_{reg}} \sum p_i^* L_{smooth\ l1}(t_i, t_i^*) \tag{4.5}$$

Where p_i is the score of an anchor i being a defect or not,

p_{i^*} is 1 for positive anchors and 0 otherwise,

t_i is a set of 4 values derived from anchors' bounding boxes,

t_{i^*} is the predicted transformation representative of anchor boxes,

N_{cls} is the number of regions in an input orthoframe, typically 256,

N_{reg} is the number of anchors' location in the feature map,

λ is an empirical parameter used the balance the weight of the regression loss.

Log and smooth L1 losses are used respectively in the classification and regression layers, the regression loss is computed only on positive anchors. The network is trained on randomly selected 256 anchors from an orthoframe using a backpropagation and gradient descent algorithm. The output of the regression layer is transformed back to bounding box coordinates and, the top-100 defect proposals are selected if more than 100 defect proposals exist after applying non-maximum suppression.

4.3 Region of interest align operation

Defect candidates from the RPN can have non-integer bounding boxes making it difficult to pool out their corresponding ROI from the feature map. Region of Interest Align or simply RoIAlign is a quantization-free pooling operation that uses bilinear interpolation to pool the RoI corresponding to the defect candidates from the feature map [31]. A pooling frame of size K^2 is defined; defect candidates bounding boxes are divided into K^2 bins overlaid on the feature map. Bilinear interpolation is used to estimate the pixels points of the feature map close to the sampling points in each bin.

In Detectron2's implementation of RoIAlign, an offset of 0.5 is subtracted from the candidate boxes to make the interpolation results more accurate [44]. We used a default pooler size 7x7 which results in 256x7x7 ROI for one training batch. During the training and before the pooling operation, defect candidates are augmented by ground truth boxes of the batch to speed up the training process [31].

4.4 Network heads

4.4.1 RoI Head architecture

The RoIs pooled by RoIAlign are fed to network headers to produce the defect class, bounding box, and semantic mask. The network head is made of two components: the object detection (classification plus localization) head and the semantic segmentation head.

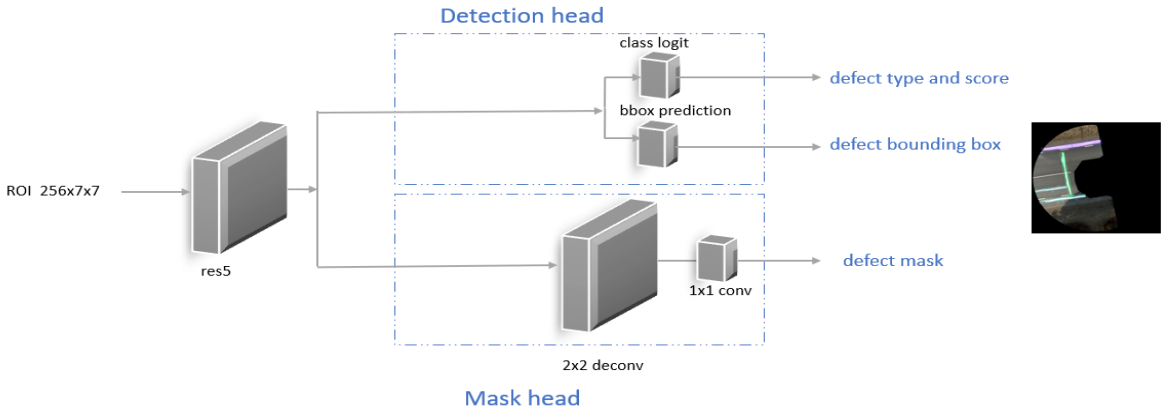


Figure 4.4 Network Heads architecture. *conv* stands for convolution and *deconv* for deconvolution.

The common part is the fifth stage of the backbone ResNet network mentioned in 4.1

The detection head contains defect type and box prediction layers while the mask head is a fully convolutional network with a 2x2 deconvolution layer and the final mask prediction layer which is a 1x1 convolution layer.

4.4.2 RoI Head loss function

The ROI head constitutes the second stage of the network, it is a combined loss function made up of classification, regression, and segmentation losses. Given a labeled ROI and *K* classes, the total loss function is defined as following [31].

$$L = L_{cls} + L_{box} + L_{mask} \tag{4.6}$$

$$L = -\log pu + \lambda[u \geq 1] smooth_{L_1}(t^u, v) - \beta\,[k]\,\frac{1}{K}\sum y_u \log(y_{pred}) \tag{4.7}$$

Where $p = (p_0, \ldots, p_k)$ for *K+1* class including background computed by a softmax,

pu is the probability for the true class u,

$\lambda; \beta$ are parameters used to balance box and mask loss scales respectively,

$p = (p_0, \ldots, p_k)$ for *K+1* class including background,

$[u \geq 1]$ indicates the box loss is defined for at least one true class,

$[u]$ indicates the mask loss is only defined for true classes,

y_{pred} is the predicted binary mask for a ground truth class u computed by a per-pixel sigmoid.

t^u is a linear transformation representative of the predicted bounding box [33],

v is a linear transformation representative of the ground truth bounding box [33].

For each experiment, we chose λ and β to adjust the losses to a common scale. By defining a mask for each class, Mask R-CNN guarantees no competition among different classes so that the network can output different masks for each region of interest.

5 FEATURE PYRAMID NETWORK

5.1 Motivation

Pavement defects of the same or different classes have different scales across orthophotos, this is also an eminent problem in object detection where we want to learn multiscale representations of objects in the training data [46]. In

Figure 5.1, we highlight a typical scenario of transverse cracking defect types with different scales in two orthophotos; this is common for other defect types as well.

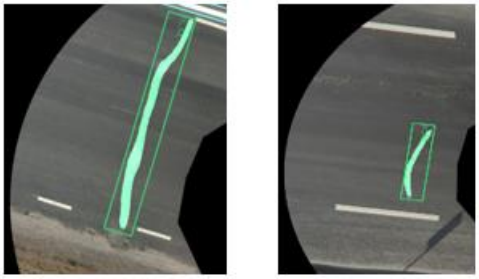


Figure 5.1 Different scales of transverse cracking defect type

Feature pyramid network (FPN) is a method of learning object representations at multiple scales, it can be easily integrated into any backbone feature extractor such as ResNet in any two-stage network [46].

5.2 Feature pyramid network enabled multi-scale feature extraction

The architecture of the feature pyramid network (FPN) enabled multiscale feature extractor is shown in Figure 5.2 with ResNet used as a backbone, *P2-P6* are the names given to multiscale output feature maps. FPN down-samples the last output of the ResNet (*res5)* with a max-pooling of kernel size 1 and stride of 2, this downscales *res5*'s output by a factor of 2. Output feature maps from all stages of the backbone feature extractor (a stage is a set of layers of the same size) are fed to a 1x1 convolutional layer to reduce the channel size to 256 channels through a lateral connection, up-sampled by a factor of 2 and added to the next laterally convolved feature map from the bottom-up backbone. The merged feature maps are fed to the last convolution layer of 3x3 to generate the final multi-scale feature maps to be used in RoI Heads and RPN.

The advantage of FPN lies in the fact that ROI can be extracted from features with different scales instead of a unique feature map.

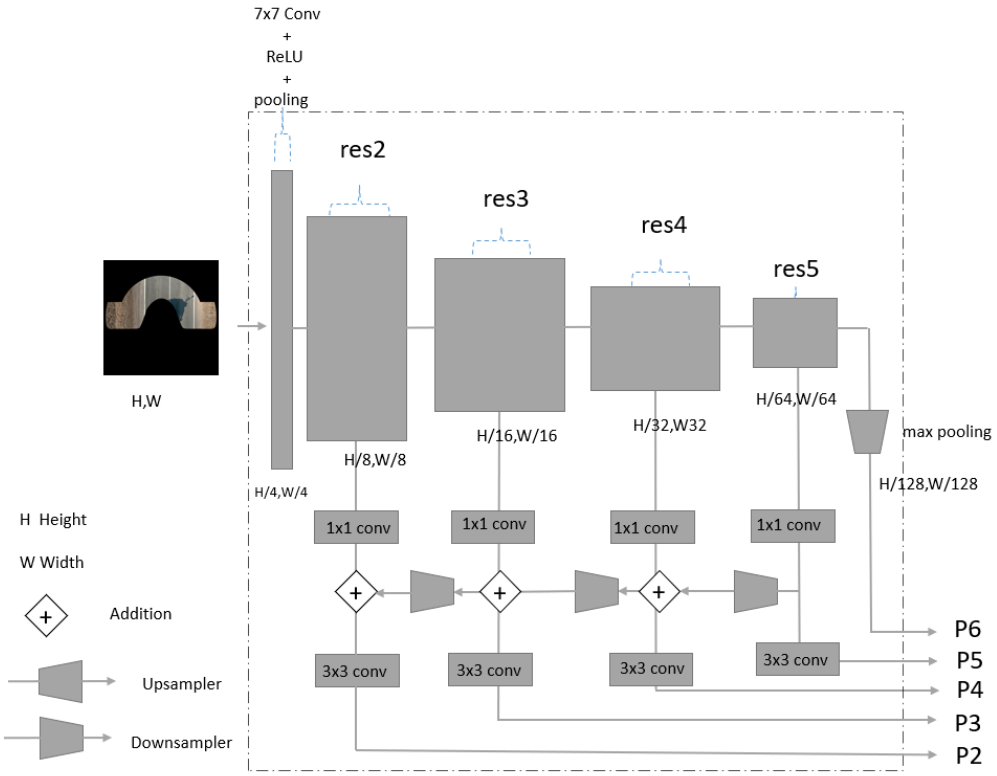


Figure 5.2 Feature Pyramid Network's architecture

5.3 Region proposal network with feature pyramid network

The region proposal network is the same as in 4.2, with the exception that defect candidates are generated not only for one feature map but for the 5 pyramids of features. This results in more defective candidates and probably with higher quality (bounding box scales) but after non-maximum suppression, we can choose to keep the top candidates.

Now that we have 5 multi-scale feature maps, we must decide which one to use in RoIAlign i.e., which pyramid feature map corresponds to a given defect candidate. This was solved in [46] with an assignment rule relating the RoI scale to a single level of pyramid feature. Proposals with smaller scales are assigned to a higher-level feature pyramid since they have finer resolution while proposals of higher scales are assigned to a low-level feature pyramid.

5.4 RoI heads with feature pyramid network

Regions of interests all share the same RoI head independently of the feature maps they are pooled from.

As shown in Figure 5.3, the new RoI head is different from the one introduced in 4.4.1. The detection head contains two fully connected layers, the 256x7x7 ROI is flattened to a tensor of size 12,544 and fed to the first fully connected layer while the second layer is of size 1024x1024 [31]. The mask branch is fully convolutional with four 3x3 convolutional layers, followed by a 2x2 deconvolution and 1x1 convolution layers to output the predicted mask class-agnostic binary mask [31].

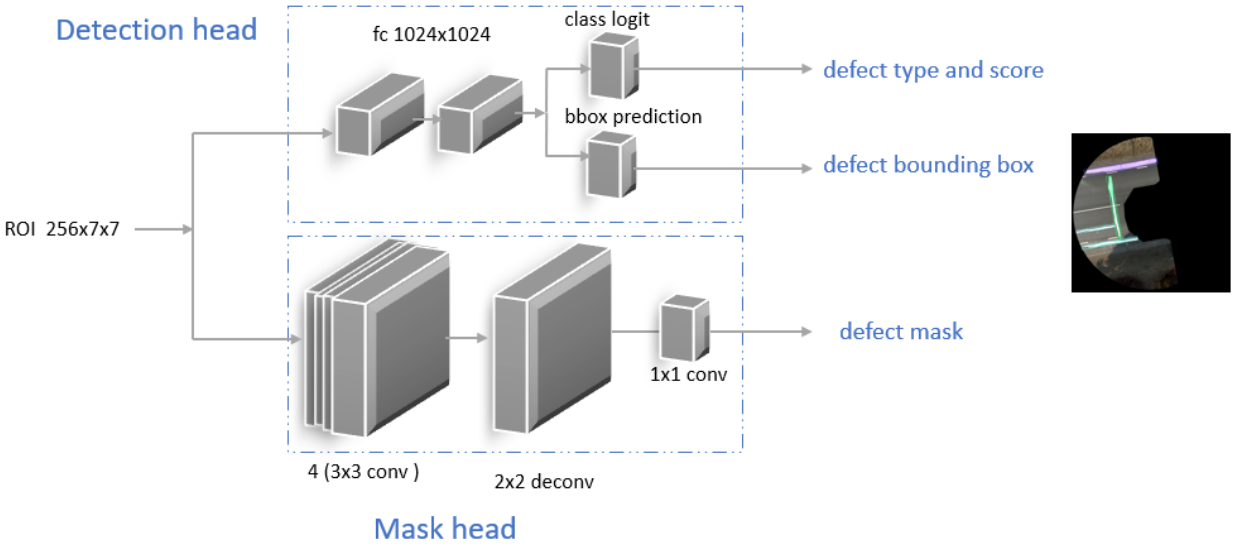


Figure 5.3 FPN-enabled RoI Head's architecture

6 CASCADE HEADS

6.1 Motivation

A known limitation of object detectors is the remarkable decrease of the network performance with increasing IoU thresholds. Figure 6.1 shows mask-based mean average precision (mAP) of the leading solutions on COCO instance segmentation task: As the IoU threshold increases, the mAP decreases considerably.

	AP	AP^{50}	AP^{75}
Megvii	0.531	0.768	0.586
ppdet	0.527	0.766	0.578
mmdet	0.513	0.747	0.565
Alibaba-Vision	0.512	0.743	0.566
XForwardAI	0.510	0.739	0.565
DeepAR (ETRIxKAIST_AIM)	0.496	0.727	0.547
MMDet	0.490	0.730	0.539
Megvii (Face++)	0.488	0.737	0.536
DetectoRS	0.474	0.714	0.520

Figure 6.1 COCO Instance segmentation leaderboard [34]

Cascade headers mitigate this problem by introducing increasing IoU stages of detection to the head of any two stage-detector. Cascade R-CNN introduces stages consisting of only classifiers and box refinement modules [47] while Cascade Mask R-CNN follows both Cascade R-CNN and Mask R-CNN methods to include the mask branch in each cascade stage.

6.2 Cascade R-CNN

Cascade RCNN reuses the first part of a two-stage detector for output object proposals and adds extra stages of classification and regression in the network head [47]. We empirically found 3 stages of increasing IoU thresholds (0.4, 0.5, 0.6) useful in this work. The output at a stage is used as input to train the next higher quality stage.

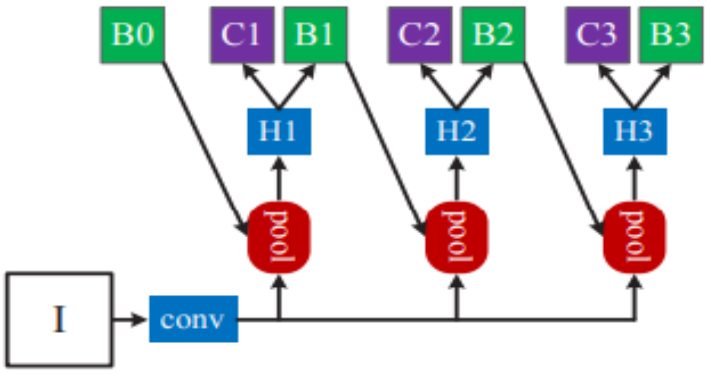


Figure 6.2 Cascade RCNN architecture [47].*Conv* is used for backbone convolutional network, *pool* for the RoI Align operation, *B0* for the proposal' box from the RPN, *H* for the common part of detection head, *C* for classification layer, and *B* for bounding box layer. At the first stage, the proposals are used as inputs, an IoU threshold of 0.4 is used to separate foreground classes from background classes; at the second stage the threshold is increased to 0.5 and the outputs of the first steps are used as inputs. Same thing for the last step where the IoU threshold is 0.6.

The mask branch responsible for defect segmentation is kept intact as it is in 4.4.1

Cascade RCNN was first proposed for Faster R-CNN; while it does increase the performance on object detection, its impact on the segmentation task is limited. Cascade Mask R-CNN includes the mask branch in parallel to classification and regression branches at each stage of the cascade headers. In Figure 6.2, for the first stage, for example, this consists of adding a mask branch after the detection branch and using its outputs for the mask branch in the second stage. We used the same IoU thresholds as in 6.2 to distinguish between foreground and background at each stage.

7 PERFORMANCE METRICS

7.1 Intersection over union, precision, recall

Intersection over union or simply IoU or Jaccard index is a popular metric used in computer vision tasks such as object detection and semantic segmentation to measure the area of overlap between the ground truth object and its prediction.

$$IoU\ (gt, pred) = \frac{|gt\ \cap pred|}{|gt \cup pred|} \tag{7.1}$$

Where gt denotes the ground truth mask or bounding box and $pred$ denotes the predicted mask or bounding box.

In the context of object detection, an IoU threshold is used to distinguish among predictions. A true positive or TP is a detection where the ground truth and the prediction have the same class, the confidence score is higher than a predefined score (0.05 % in this work) plus the IoU between the ground truth and the prediction is higher than or equal to the IoU threshold. When the IoU falls below the threshold, the prediction is a false positive (FP). False-negative (FN) corresponds to non-predicted ground truth objects or cases where the IoU is higher than the threshold, but the class is wrongly predicted.

Precision measures the number of correctly predicted objects out of all predictions and recall is used to find the rate of correctly predicting a ground truth object.

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.3}$$

7.2 Mean average precision, mean average recall

The expression average precision (AP) is used interchangeably with mean average precision (mAP). For a given defect class, mAP is the average of areas under curve of the precision-recall curves for predefined IoU thresholds. In object detection, we use

the definitions of precision and recall in 7.2 to draw the precision-recall curve. However, instead of trying to find the exact area under curve, we sample the recall curve in 101 recall points (from 0 to 1 with a step of 0.01) to interpolate the area under curve.

We follow COCO's evaluation rules and define the mAP at 10 IoU thresholds from 0.50 to 0.95 with a step of 0.05 (0.50:0.05:0.95). The mAP for a class is then given as in the following equation.

$$AP_{class} = \frac{1}{10} \sum_{iou \in IoU\ Thresholds} AP_{class}(iou) \tag{7.4}$$

Finding the mAP of a whole dataset consists of averaging the mAP values for all *K* classes present in the data.

$$AP = \frac{1}{K} \sum_{class \in classes} AP_{class} \tag{7.5}$$

To compute the average recall AR or mAR, we use the recall-IoU curve, and follow the same process described above to first compute the AR for each class and then the general mAR for a dataset.

With instance segmentation models where we predict both bounding box and semantic masks, we distinguish box-based mAP from mask-based mAP simply by how the precision and recall are calculated (either using mask IoU or box IoU).

7.3 Correlation and mean absolute error

During testing, for every 100m long road strip, metric and object correlations are calculated to find the relationship between the predicted and ground truth defects. The metric correlation measures the relationship between the lengths or areas or numbers of predicted and ground truth defects while the mean absolute error measures the agreement between the predictions and the ground truth. The mathematical formulas for metric correlation and mean absolute error are given in Equations 7.6 and 7.7.

$$Corr\ [Class] = \frac{\sum(\delta_{pred} - \overline{\delta}_{pred})\ (\delta_{gt} - \overline{\delta}_{gt})}{\sqrt{(\sum(\delta_{pred} - \delta)^2 \sum(\delta - \overline{\delta}_{gt})^2}} \quad (7.6)$$

Where $Corr$ is the metric correlation and

$\delta_{pred}\ and\ \delta_{gt}$ are predictions and ground truth defects' lengths (in m) if the defect type is longitudinal, joint reflection or transverse cracking; areas (in m^2) if the defect type is network cracking, patched road, or weathering and counts if the defect type is a pothole.

$$MAE = \frac{\sum\ |\delta_{pred} - \delta_{gt}\ |}{N} \quad (7.7)$$

Where MAE is the mean absolute error;

$\delta_{pred}\ and\ \delta_{gt}$ have the same definitions as in the equation (7.7)

N is the number of data points.

In addition to correlations, the overall defectiveness for each bin of 100m was accessed using the following equation:

$$D[\%] = \frac{(L_{POIKR} * 2.5 + L_{KPIKIPR} * 0.5 + L_{KVUUK} * 0.1 + N_{AUK} + A_{VORK} + A_{MUREN} + A_{PAIK} + A_{SERVA)}}{L_{section} * W_{section}} \quad (7.8)$$

Where *L* is the length of the corresponding predicted defect type represented by its code as defined, *N* is the number of such defect type and *A,* is the area of such type in m^2. $L_{section}\ and\ W_{section}$ are the length and width of the road strip under consideration. The coefficients are defined by the Estonian Road Administration [5]. D is the overall defect sum in percentage.

8 EXPERIMENTS AND RESULTS

Unless otherwise specified, Detectron2's implementation of Mask R-CNN [45] was used as baseline code and further modified for the experiments.

8.1 Experiment with pavement patches

As mentioned in 3.2, one way to generate data for instance segmentation task is by using sliding windows of size 512^2 to create patches of pavement images; patches containing at least 500 defective pixels are considered for training. The training, validation, and test sets contain respectively 25517, 4274, and 3335 defect instances. The distribution of defects among classes is identical to the one described in Figure 3.3.

A standard ResNet50 based Mask R-CNN with feature pyramid network pre-trained on COCO dataset is used as the model. A mini-batch gradient descent algorithm is used as a solver with each batch containing 8 training images [48]. The training was performed on a single Nvidia GeForce 2080 Titan GPU for a total of 100k iterations; the learning rate was initialized to 0.001 and decayed by 10 at 50k and 75k iterations, respectively. A momentum of 0.9 and a weight decay of 0.001 were used. Figure 8.1 shows how the total loss changes during the training in training and validation sets.

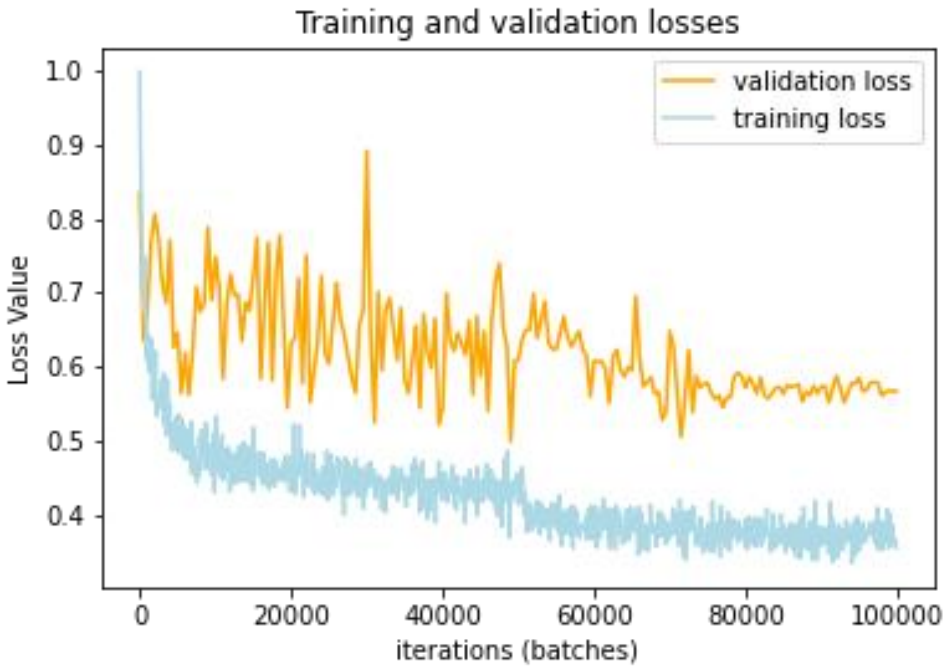


Figure 8.1 Loss during training with sliding windows approach.

In Figure 8.1, the training loss is smoother than the validation loss; this is because the training loss is logged more often than the validation loss.

During the training, average precision values computed on the validation set are also monitored as shown in Figure 8.2 and Figure 8.3.

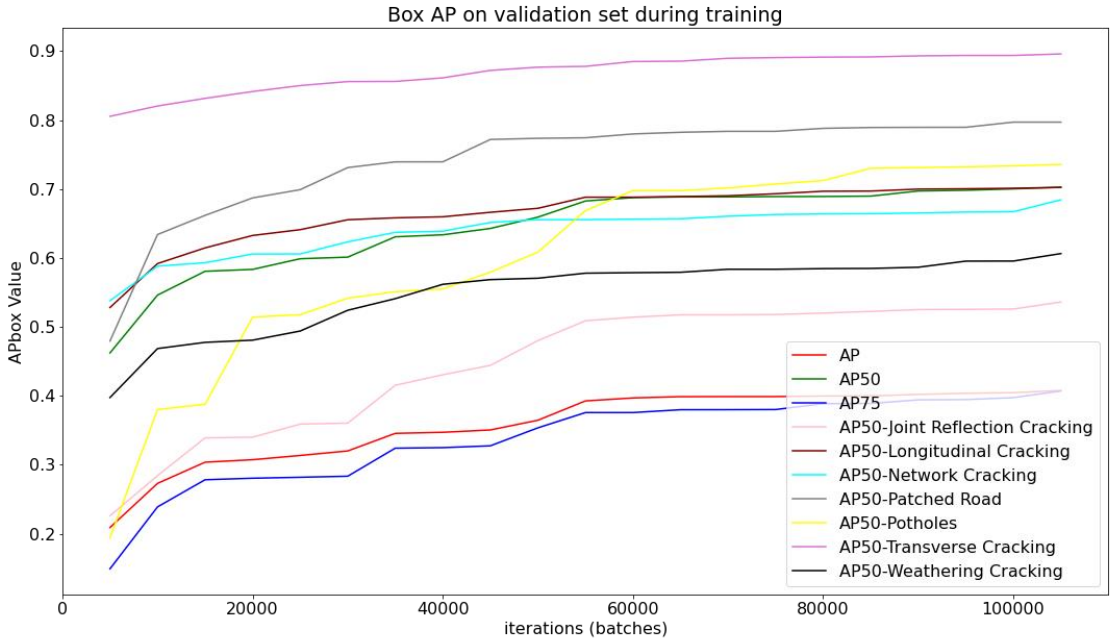


Figure 8.2 Sliding window approach: bounding box based average precision on the validation set.

The box-based APs are higher than mask-based APs; it is because localization is an easier task compared to dense pixel-level classification i.e., segmentation. The highest value of AP_{50} is 69% for the bounding box and 66% for the mask. However, it is important to remember that both the training and validation data were made of only defective patches. So, while those values seem acceptable compared to instance segmentation results found in the literature [31], it assumes inputs to the model are all defective.

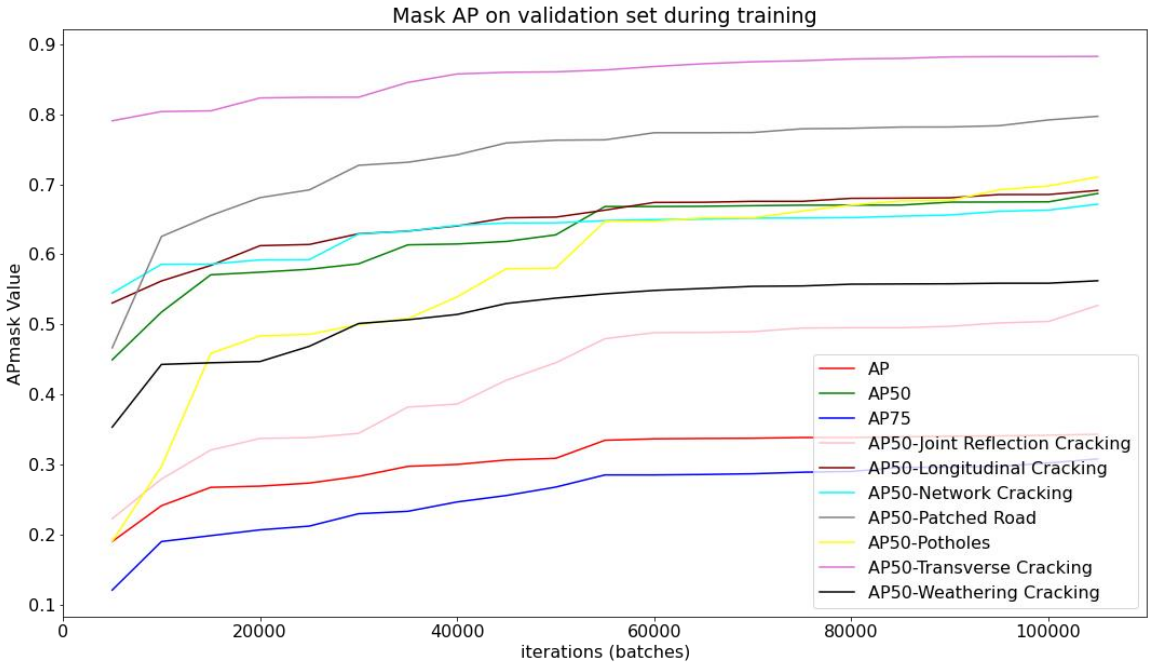


Figure 8.3 Sliding window approach: bounding mask based average precision on the validation set.

During the testing, we ideally want to run the network on full pavement orthoframes; for that, the same sliding window is used to slice the orthoframe into patches of size 512^2 which are all fed to the network for predictions (no matter if they are defective or not).

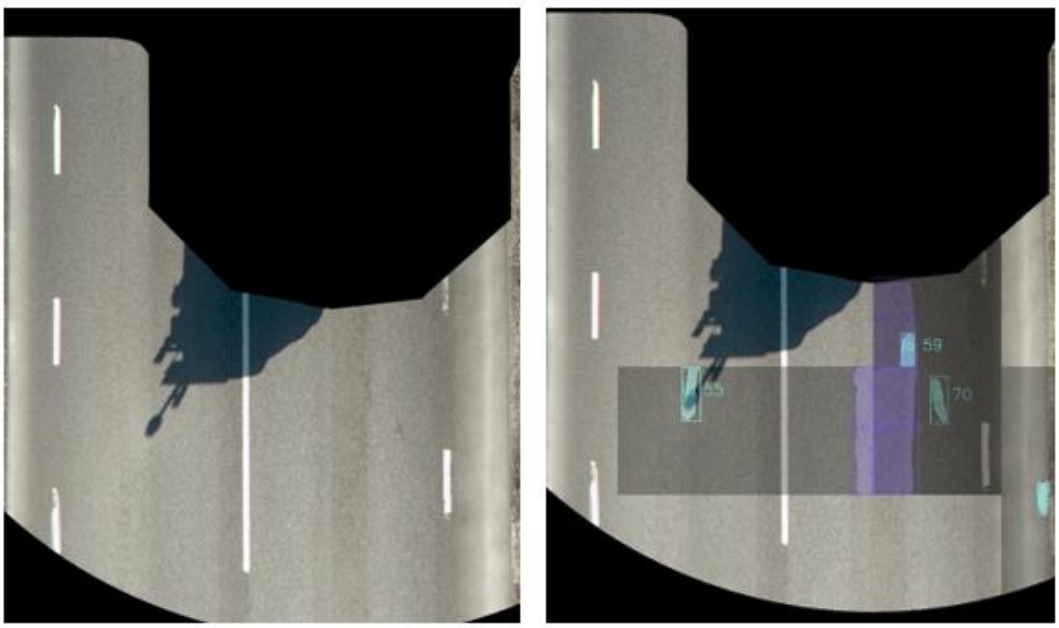


Figure 8.4 Sliding window approach: intact ground truth(left) vs prediction (right) , light blue is for longitudinal cracking and violet for network cracking. The grey color indicates defective patches.

Figure 8.4 shows an example of the predictions of a full orthoframe. Some non-defective patches are also predicted as defective as it can be seen in Figure 8.4 thus making the network not the optimal solution to use for full orthoframes containing defective and non-defective patches. However, this can be used if prior knowledge of defective patches is known through a middle classification network for example.

More examples of good detections by the network can be found in Figure 8.5 and Figure 8.6.

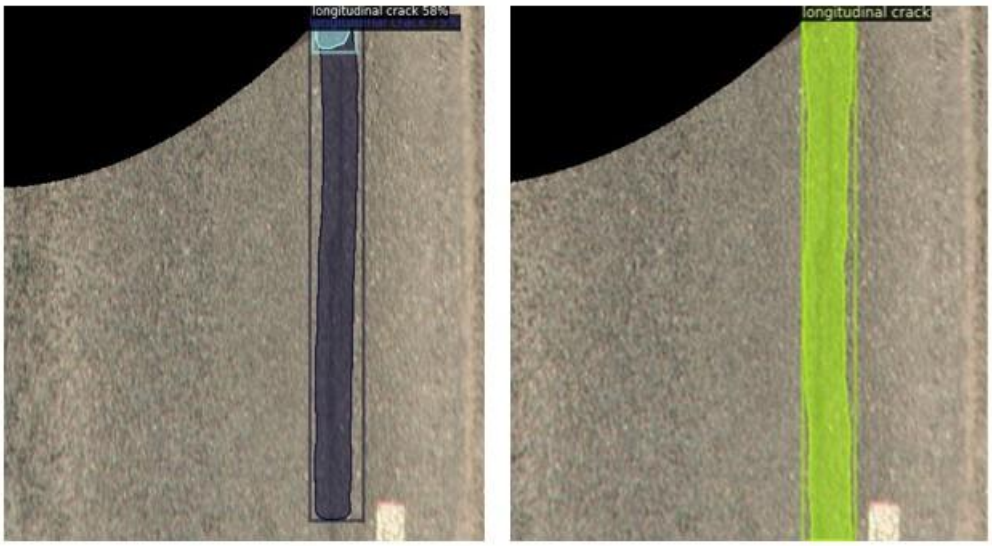


Figure 8.5 Sliding window approach: example of a good prediction(left) versus ground truth(right). The ground truth defect instance is successfully predicted by the model as two instances of the same class with 75% and 58% confidence scores.

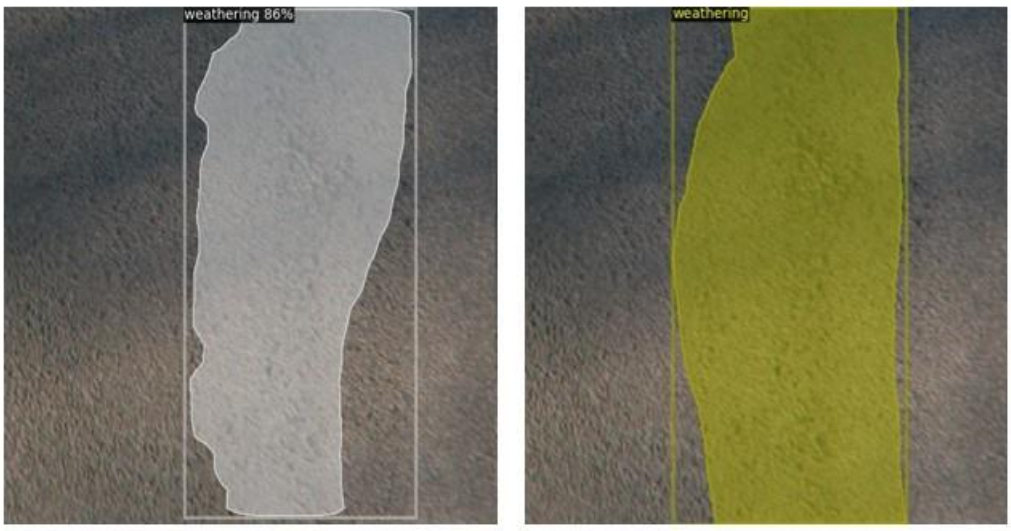


Figure 8.6 Sliding window approach: another example of good prediction (left) versus ground truth (right). In this example, weathering is successfully detected by the network with a score of 86%.

8.2 Experiment with downscaled orthoframes

With the sliding window approach, the network is only able to learn from defective patches; to counterbalance this, orthoframes were downscaled by a factor $k = 4$ so they can be fed to the GPU machine. While this approach results in loss of resolution, it allows the network to learn from a full orthoframe instead of only defective patches. With downscaled orthoframes, experiments were conducted with standard FPN-enabled Mask R-CNN and FPN-enabled Mask R-CNN with cascade headers.

8.2.1 Experiment with feature pyramid network-enabled Mask R-CNN

In this experiment, orthoframes were downscaled by a factor of 4 resulting in images of size 1024x1024. Nearest neighbor interpolation was used for greyscale masks to avoid having interpolated pixels values that do not correspond to any defect type while bicubic interpolation over a 4x4 pixel neighborhood (is slower and known to produce limited artifacts) was used for orthoframes. Small annotations with an area less than 500 pixels were filtered out. Augmentations defined in Table 3.2 including edge resizing are applied to the training set. In total, there exist 32460, 5718, and 4278 defect instances in training, validation, and test sets, respectively. The distribution of instances among classes is the same as in Figure 2.2.

Mask R-CNN with pyramidal features is used as a model; the bottom-up module of the backbone feature extractor is the conventional ResNet50 without its last pooling layer. The network is trained for 100k batch iterations (1 batch = 8 images) with a learning rate of 0.001 decayed by 10 at 25K,50K, and 75K using mini-batch gradient descent and backpropagation. A linear warmup scheduler introduced in [49] was adopted to linearly scale up the learning rate for the first 1k iterations. A single training epoch corresponds to ca. 1821 batches since there are 14570 orthoframes in the training set. The learning curve is given in Figure 8.7.

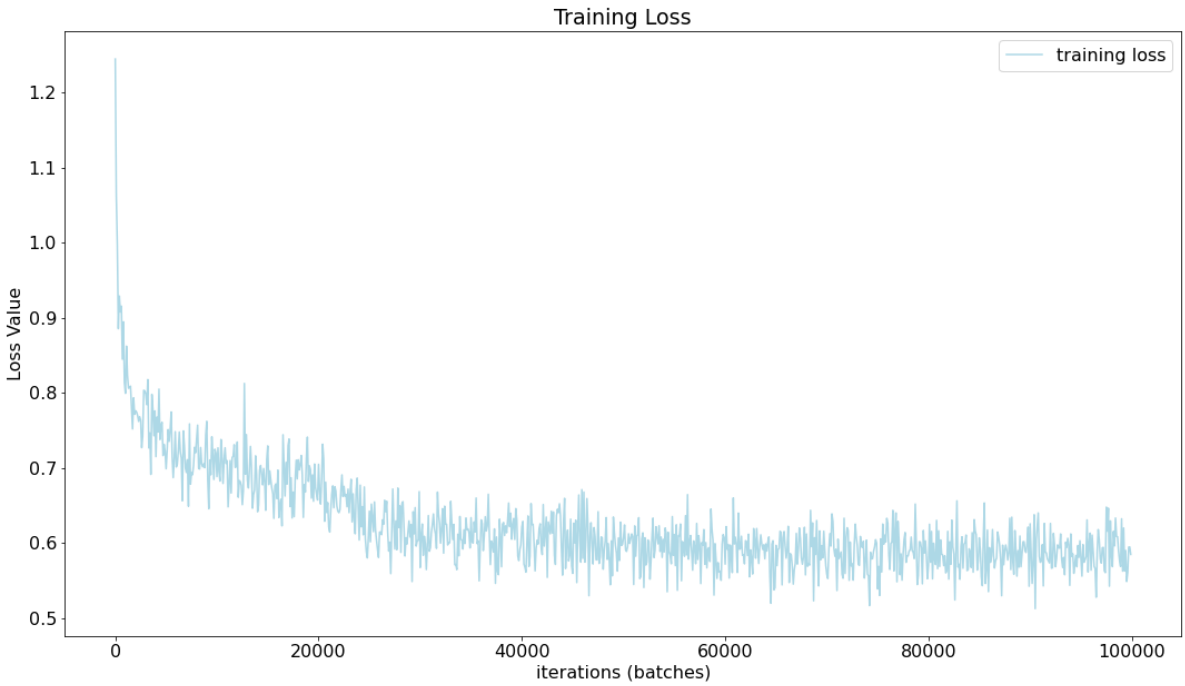


Figure 8.7 FPN-enabled Mask R-CNN: training loss.

The weights were saved every 5k iterations. After 55k iterations, there were no further improvements neither in the training loss (see Figure 8.7) nor in the box and mask-based AP (see Figure 8.8 and Figure 8.9), so the weights at 55k (10k iterations after the first learning rate decay) were selected to be the best.

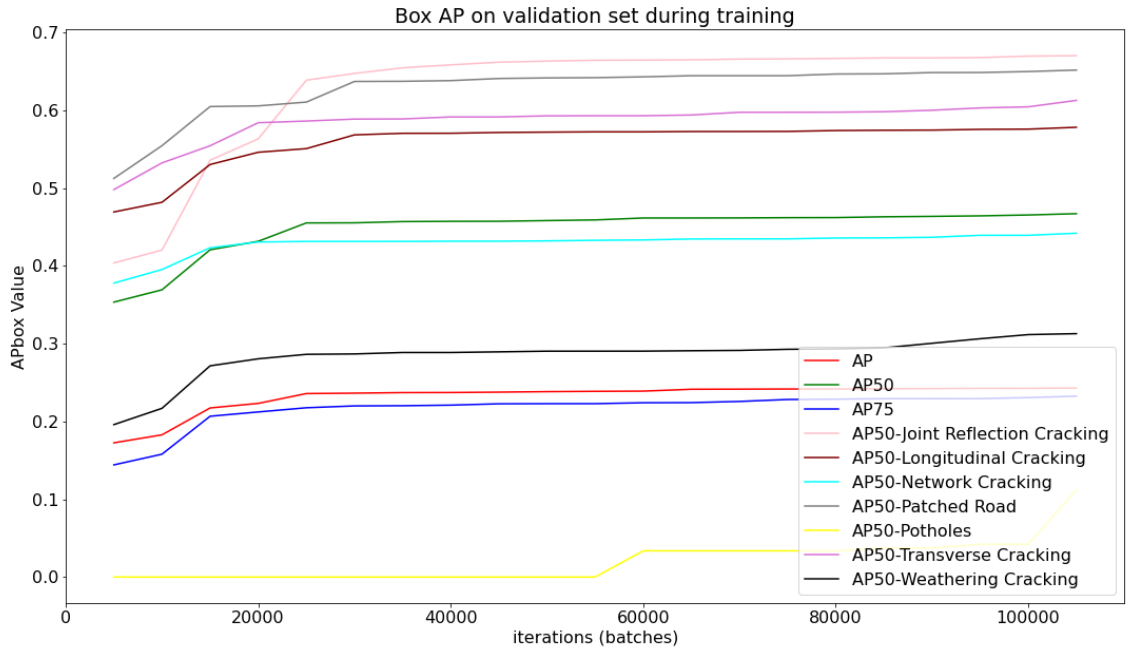


Figure 8.8 FPN-enabled Mask R-CNN: box-based AP on the validation set

The overall average precision was 18.5% and 23% for respectively segmentation and localization tasks while the AP50 was 41% and 45%. This is because the increase of IoU thresholds leads to more false positives which results in lower AP values. The model performs well with certain defect types such as patched road and transverse cracking where the AP_{50} for both localization and segmentation tasks is at least 60% but suffers for example in predicting potholes which account only for ca. 0.5% of the total validation set and weathering where the AP_{50} is less than 30%.

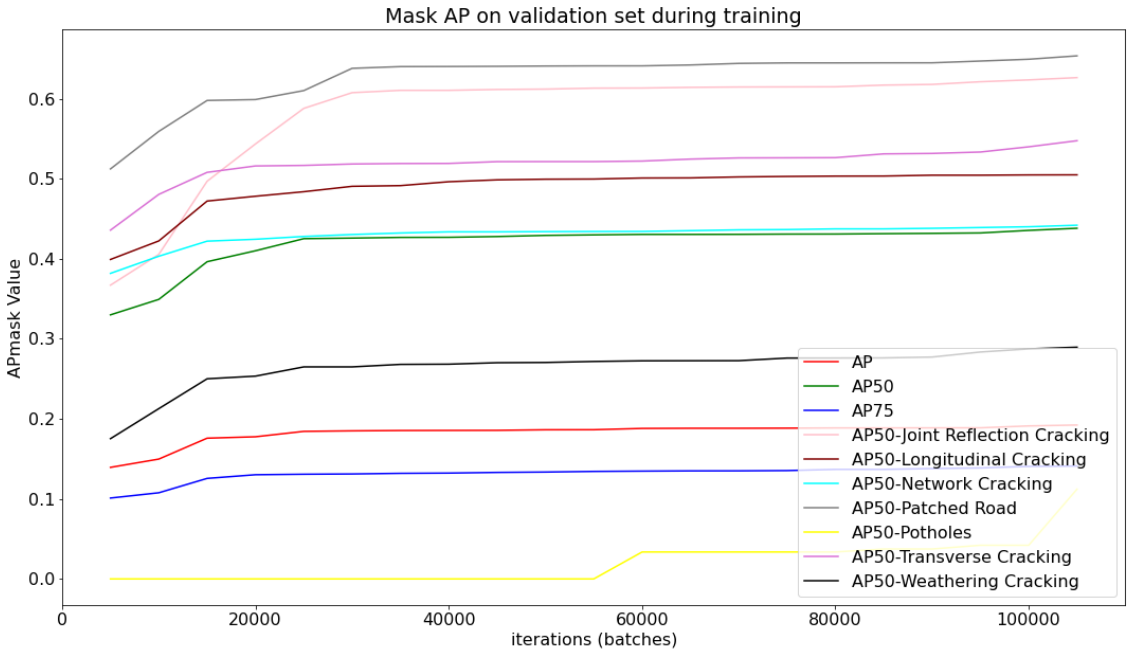


Figure 8.9 FPN-enabled Mask R-CNN: mask-based AP on the validation set.

Compared to the experiment in 8.1, the network learns a better representation of non-defective pixels since full resized orthoframes are used as inputs. In some cases, the network can also outperform the manual annotation as shown in Figure 8.10.

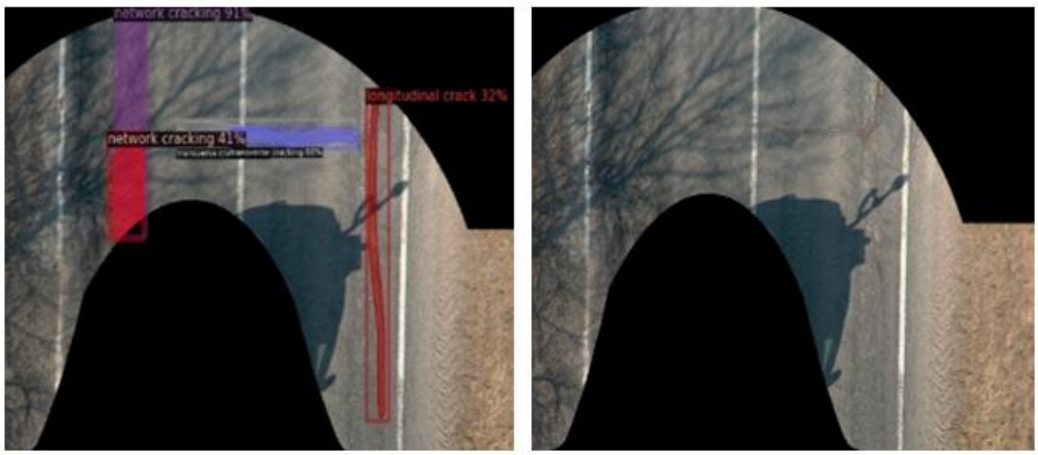


Figure 8.10 FPN-enabled Mask R-CNN: example of network's output(left) correcting the manual annotation(right). The orthoframe is defective, with a closer look, it is possible to identify the defects which were missed out by the annotators but detected by the model with acceptable confidence scores. The overlapping instances of the same type are merged during the postprocessing step.

One of the main advantages of an instance segmentation network over a succession of segmentation and classification networks is its ability to successfully distinguish neighboring instances of the same or different defect types. This is in part due to the region proposal network that can use regions from multiscale feature maps to generate multiple defect candidates for a single image. The candidates with an IoU greater or equal to 0.5 (in this experiment) are considered as foreground and used in the network heads for further box refinement, plus class and mask prediction.

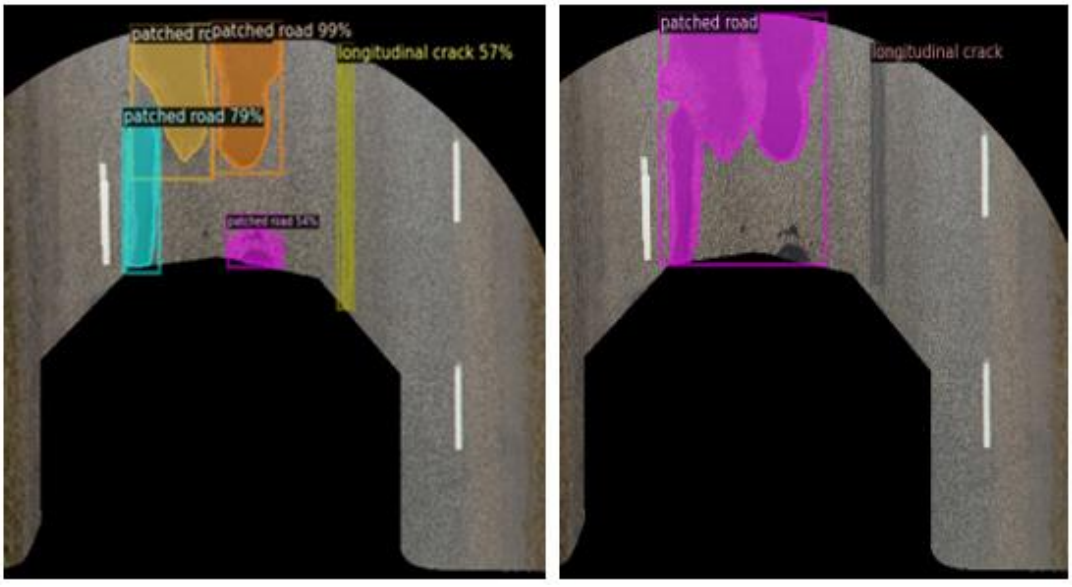


Figure 8.11 FPN-enabled Mask R-CNN: example of neighboring defect instances successfully predicted as different instances (left) versus ground truth (right).

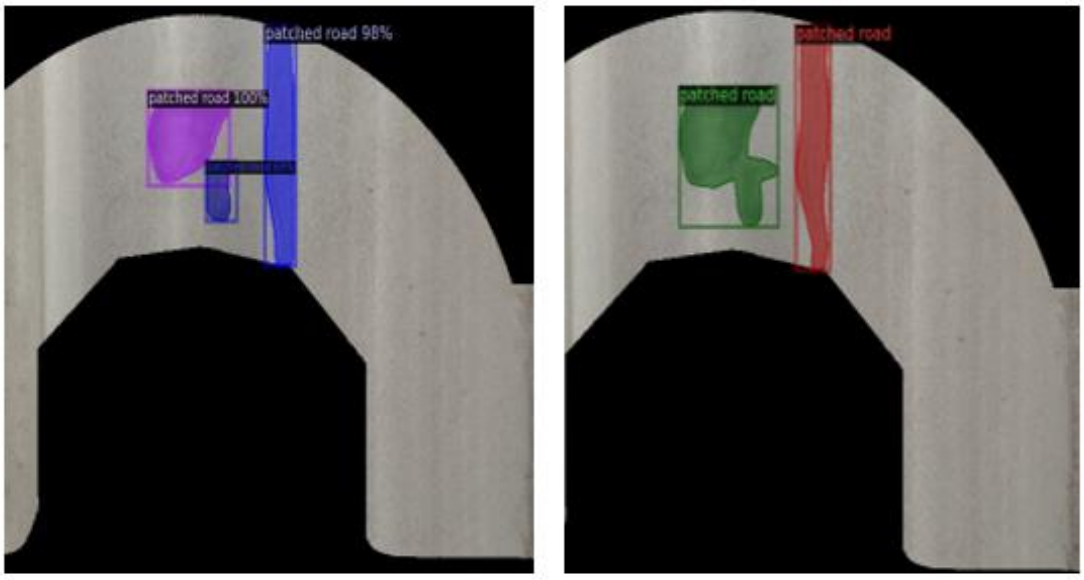


Figure 8.12 FPN-enabled Mask R-CNN: a second example of neighboring defect instances successfully predicted as different instances (left) versus annotated ground truth (right).

Figure 8.11 and Figure 8.12 show examples of the model confidently distinguishing close patched road defects instances even though those were grouped by annotators. This is particularly useful as the Estonian Road Administration for example relies on correlations metrics calculated using the number of defect instances in their decision-making process.

8.2.2 Experiment with cascade heads

The experiment with cascade headers replicates the previous experiment in 8.2.1 except that the RoI head responsible for final detection is replaced by a cascade of heads. The cascade architecture described in 6.2 contains three successive localization heads with 0.4,0.5,0.6 IoU thresholds, respectively. We use pyramidal backbone features to generate defect candidates of which only the ones with at least an IoU of 0.4 with a ground truth defect are considered as positives and fed to the first localization head of the cascade architecture. The output from the first localization head with an IoU of at least 0.5 is used as input to the last head whose outputs are also filtered based on an IoU of 0.6.

In each stage, the foreground accuracy defined as the number of correct foreground predictions divided by the total number of ground truth objects is monitored and shown in Figure 8.13.

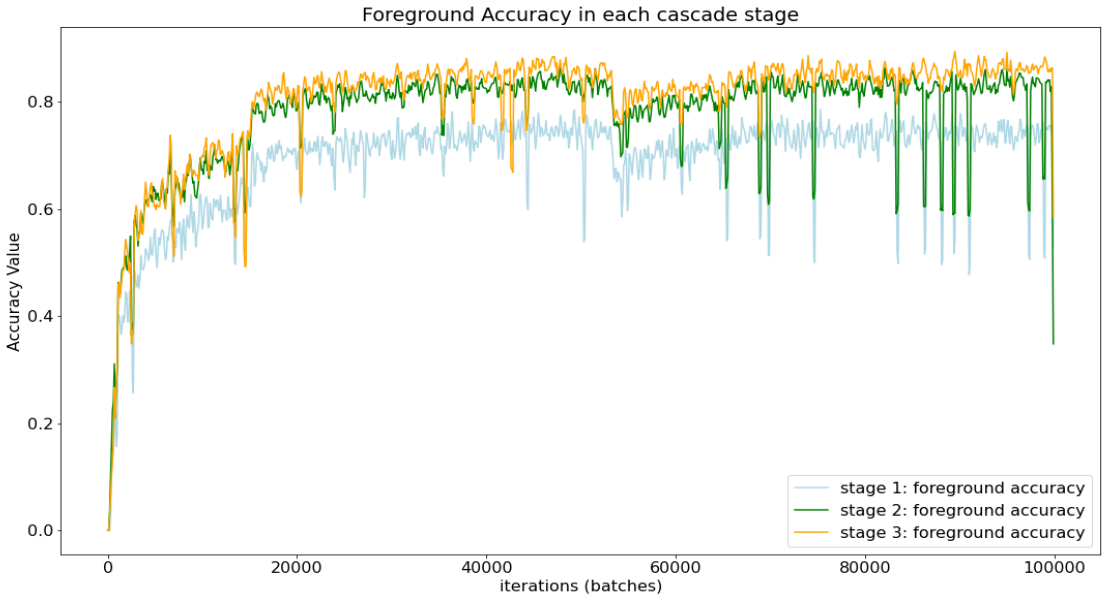


Figure 8.13 Foreground accuracies in cascade headers.

Figure 8.13 shows that the foreground accuracy increases in successive stages. In other words, by successively increasing the IoU threshold used to differentiate between foreground and background proposals during training, the probability of a defect proposal matching a ground truth object is increased. Using the same stages with the same IoU thresholds during inference leads to better defect proposals and hence detectors of higher quality. The comparison between cascade and standard headers (in 8.2.1) for all defect types confused is given in Table 8.1.

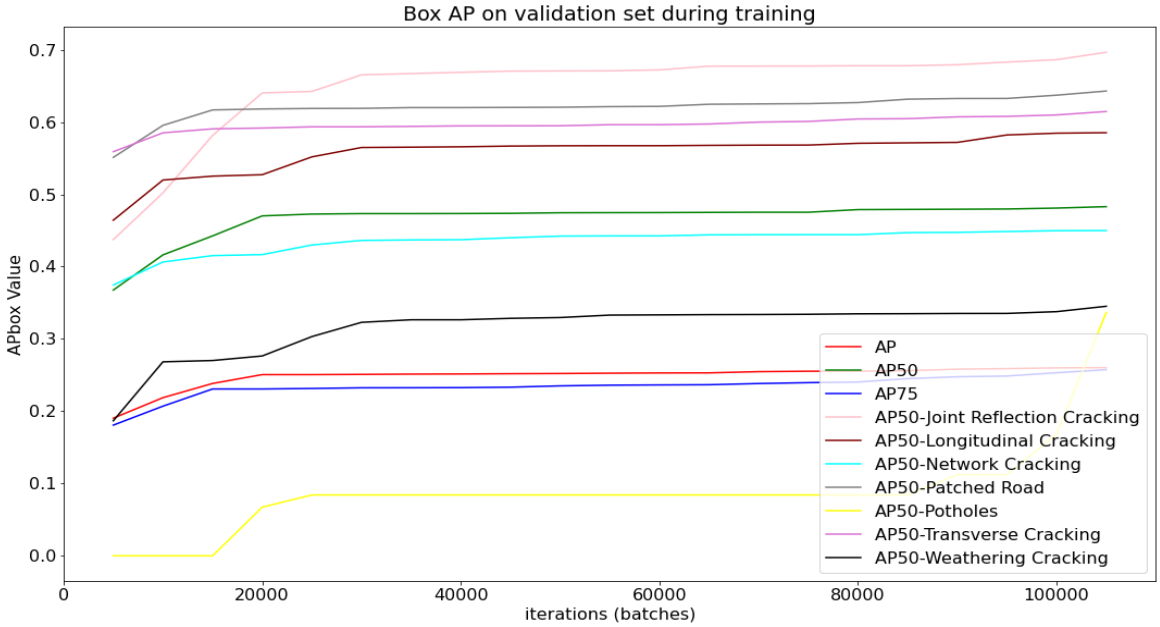


Figure 8.14 Categorical box based mean average precision for cascade headers

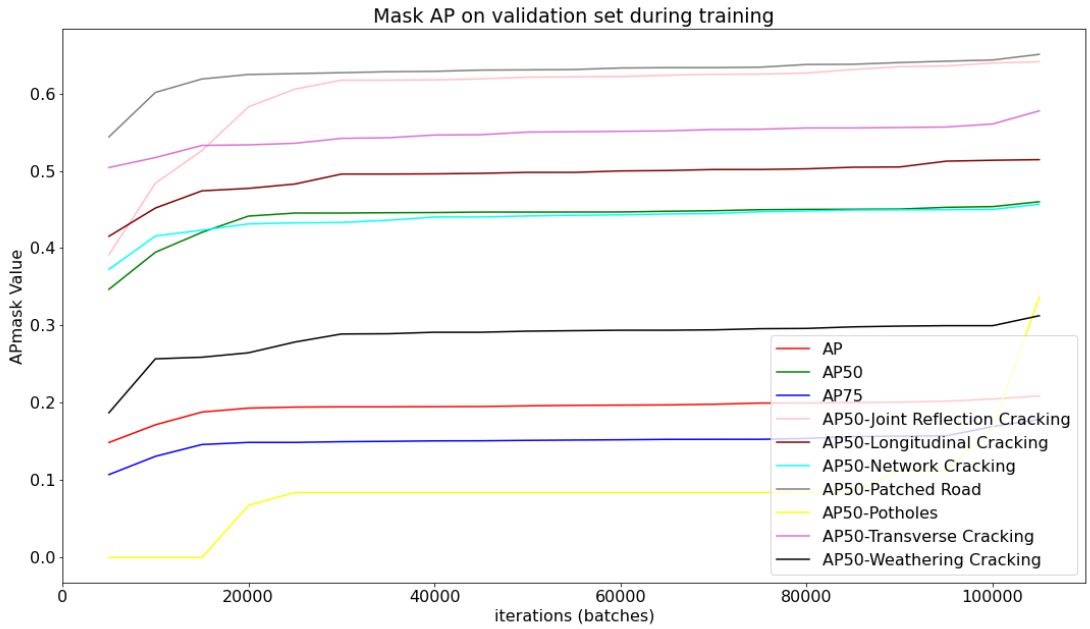


Figure 8.15 Categorical mask based mean average precision for cascade headers

The categorical mean average precision for localization and segmentation tasks (see Figure 8.14 and Figure 8.15) are both slightly higher than in the experiment with only feature pyramid and single network heads. One of the most visible improvement is with pothole defect type where the AP_{50} reaches 30% after 100k iterations but that fast jump of the AP_{50} should not be relied on since there are very few instances of potholes in the validation set and might be that the network luckily predicts some potholes instances; so, the weights at 65k were chosen to be the best.

Table 8.1 Comparison of results between cascade and standard heads, cascade performs better

	Box			**Mask**		
	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
Standard Heads	24.22	46.54	22.56	19.23	43.83	14.10
Cascade Heads	26.03	48.30	24.76	19.97	45.02	15.64

The importance of cascade headers is their ability to get rid of possible false positives and improve both the confidence score and the box shape.

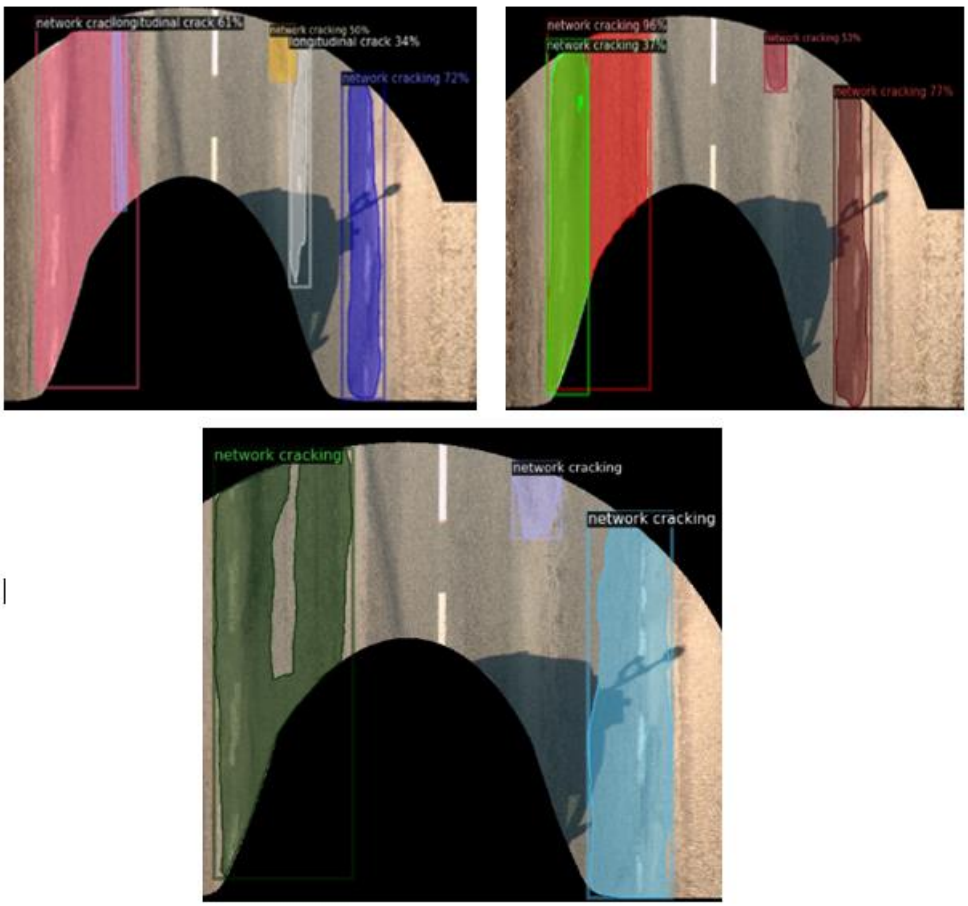


Figure 8.16 Comparison (1) of results between standard head's (left) and cascade head's predictions (right) and ground truth orthoframe (bottom).

The network with a single detection stage predicted a false positive with a confidence score of 34% in addition to the defects present in the ground truth; this is corrected by a cascade of 3 detection stages (see Figure 8.16). Moreover, there is a noticeable improvement in the confidence scores with a cascade of 3 detection stages. The same behavior can be observed in Figure 8.17 where the model with a cascade head successfully drops a false positive defect of network cracking type.

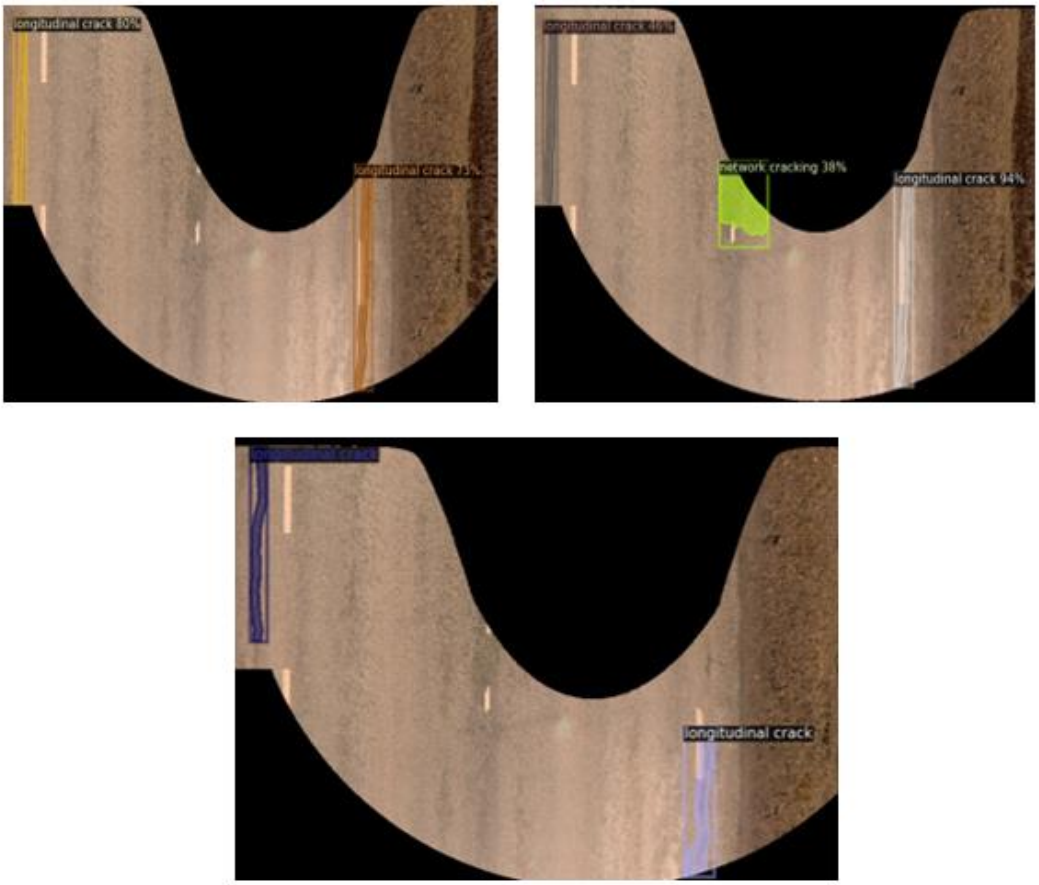


Figure 8.17 Comparison (2) of results between cascade head (left) and standard head's (right) predictions and ground truth data (bottom).

However, it should be noted that the increase in AP also comes with more complex network architectures, hence a slower inference speed. A benchmark comparing the approximated number of floating-point operations per second (FLOPS), the number of parameters, and the inference speed of network configurations described in 8.1 and 8.2 is shown in Table 8.2.

The benchmark test was performed on a single NVIDIA GeForce RTX 2080Titan GPU machine with 11 GB of memory. The configuration with the slowest inference speed (seconds/image) and a smaller number of FLOPS and parameters is the standard Mask RCNN with small patches where a single scale RestNet50 is used as a feature map. But even the slowest network configuration has an inference speed of 0.0442 second/image which means 5000 orthoframes can be predicted in less than 4 minutes on the GPU machine.

Table 8.2 Benchmarking of results with different network configurations

Configuration	Total FLOPS [Giga]	Total number of parameters [Million]				Inference Speed [second/image]
		Backbone	RPN	Head	Total	
Cascade heads as in 8.2.2	141.2097±2.4419	26.8	0.6	44.3	71.7	0.0442
Mask R-CNN with FPN as in 8.2.1	197.9363±1.1368e-13	26.8	0.6	16.6	44	0.0427
Small patches as in 8.1	68.3246±3.0250	8.5	9.5	17.1	35.1	0.1627

8.3 Additional Testing

The additional test was performed to evaluate the performance of the network in real-life conditions using not only performance metrics found in the literature but also the ones defined by the Estonian road administration. The testing data contained 5511 annotated orthoframes that were previously never used. The orthoframes come from three different road sections as described in Table 8.3.

Table 8.3 Roads' descriptions for additional testing

	Coordinates (longitude, latitude) [Decimal Degrees]		Description	Length [Km]
	Start	End		
Road A	23.82326735,59.03420139	23.71051293,58.99469076	from Palivere-Keedika to Keila-Haapsalu	8.2
Road B	25.20221481,58.08738897	25.19883943,58.08765739	from 69306 Viljandi County to Soo 16	0.2
Road C	25.15270102,59.33924343	25.0269943,59.3690348	from Raasiku Tee (Kulli) to Lagedi-Arukula-Peningi	8.2

Data preparation and augmentation described in 3.2 and 3.3 were applied to the orthoframes. Rescaled orthoframes of size 1024x1024 were fed to the network with pyramidal features and cascade headers described in 8.2.2 for inference. The network configuration described in 8.2.2 was used. The semantic greyscale masks produced by the network are digitized using an in-house software module developed by EyeVi. The

digitized defects can be integrated into public geographical information system software such as QGIS [50].

An example of a road strip from road A is shown in Figure 8.18. Each defect type is represented by a single color.

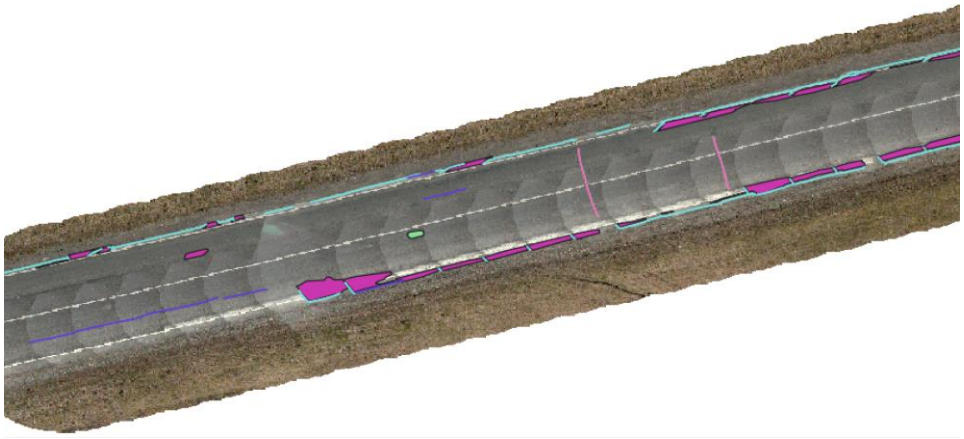


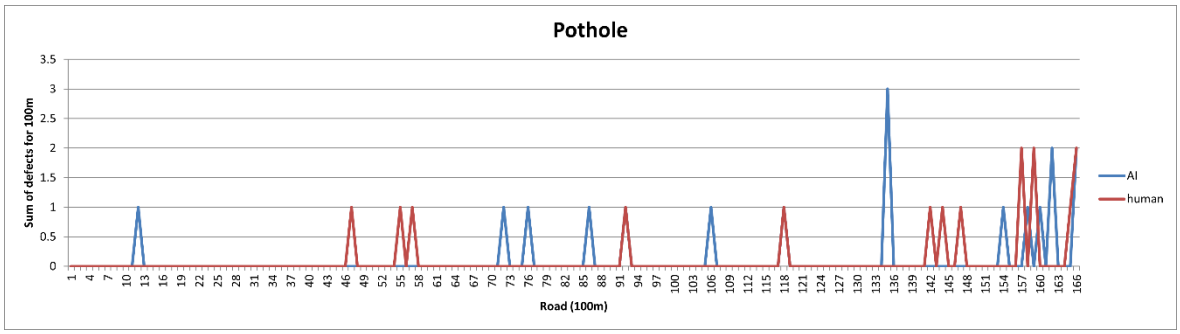
Figure 8.18 Example of the predicted defects integrated into QGIS software.

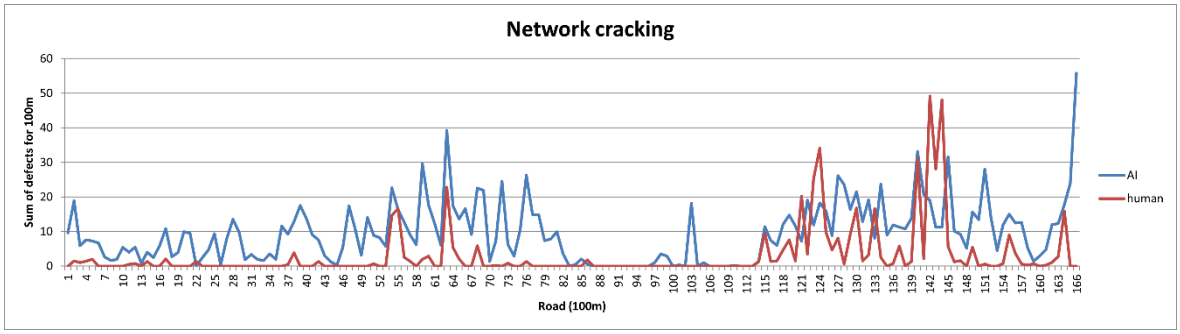
Moreover, correlation and mean absolute error defined in 7.3 were computed as shown below to compare the predictions to the annotations.

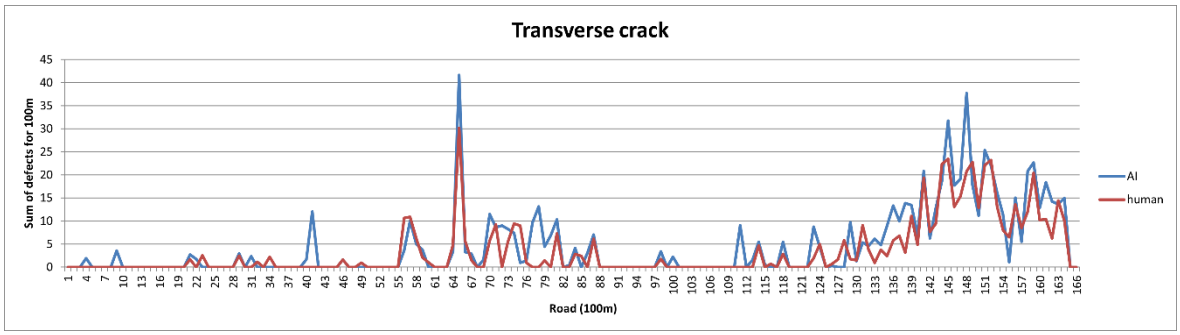
Table 8.4 Correlation and mean absolute error.

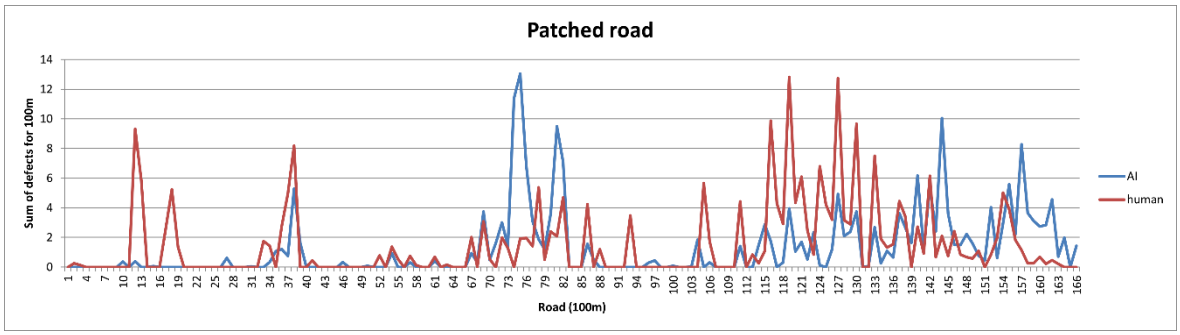
Defect type	**Correlation**	**Mean Absolute Error**
Pothole $[count]$	0.2	0.14
Longitudinal cracking $[m]$	0.67	14.04
Joint Reflection cracking $[m]$	0.67	4.24
Transverse cracking $[m]$	0.89	1.97
Weathering $[m^2]$	0.56	2.56
Patched road $[m^2]$	0.32	1.49
Network cracking $[m^2]$	0.35	8.04
Defect Sum $[\%]$	**0.86**	**3.38**

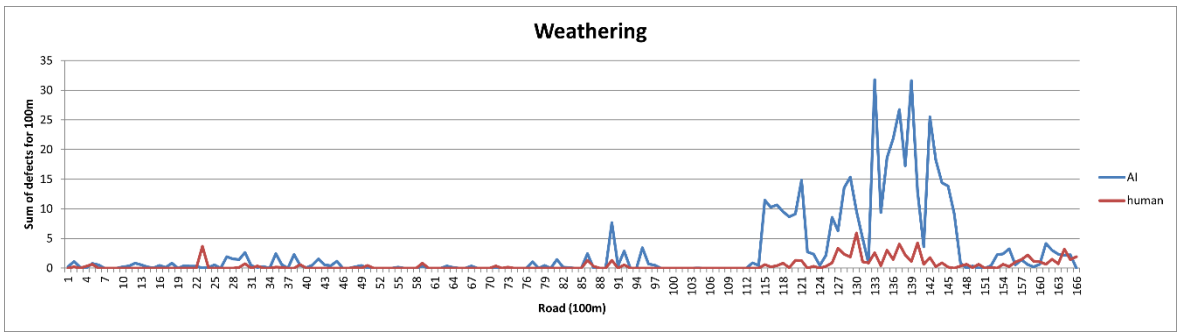
A threshold of 40% was used as a minimum confidence score. On average, the metrics for potholes were the lowest as they are underrepresented. Patched road and network cracking detection suffer in performance as well.


Pothole
Sum of defects for 100m
Road (100m)
AI
human


Network cracking
Sum of defects for 100m
Road (100m)
AI
human


Transverse crack
Sum of defects for 100m
Road (100m)
AI
human


Patched road
Sum of defects for 100m
Road (100m)
AI
human


Weathering
Sum of defects for 100m
Road (100m)
AI
human

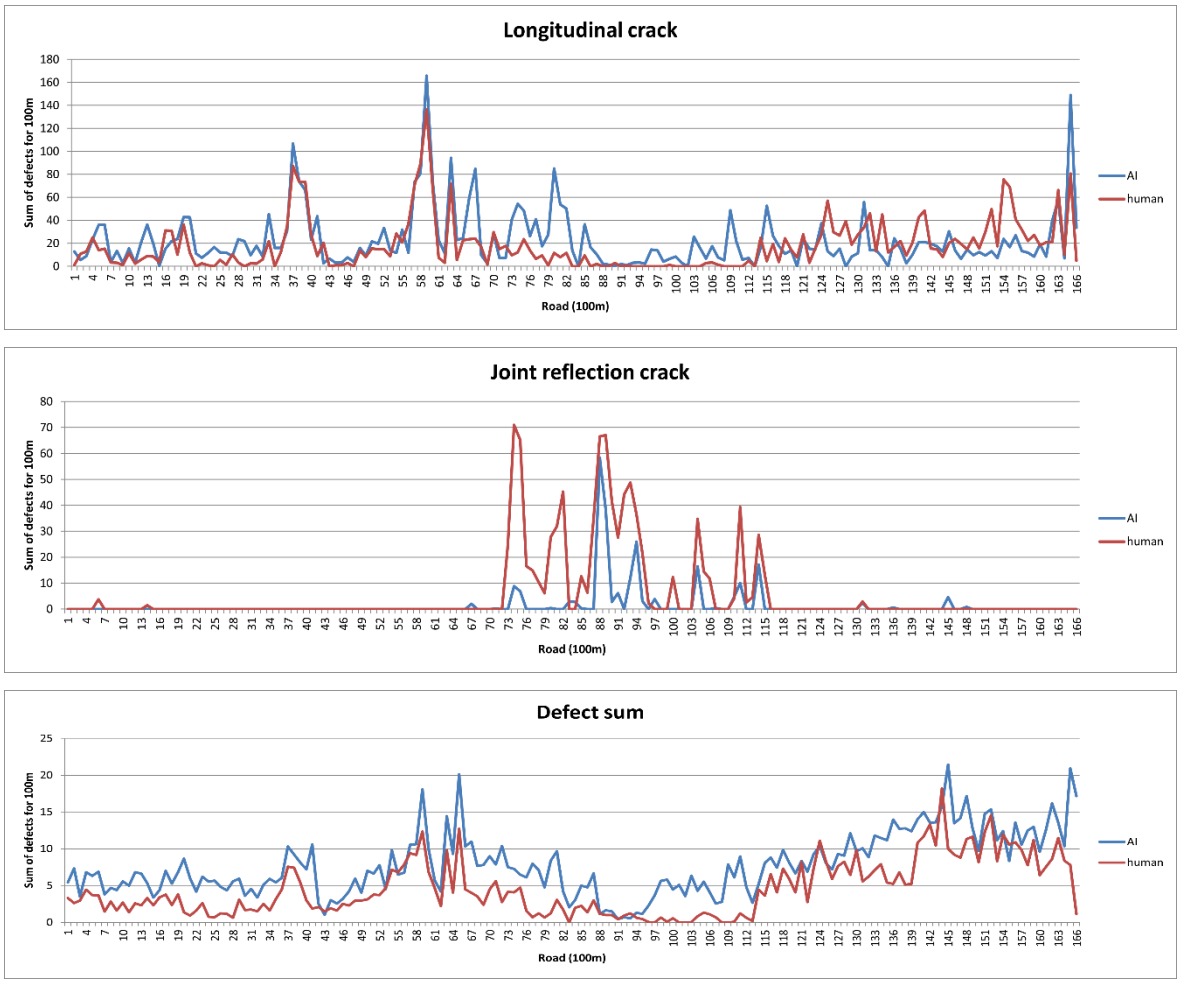


Figure 8.19 Metric correlations for defect types and overall defectiveness. Blue is for predictions and red for annotations. The abscissa corresponds to road strips of 100m and the ordinate to the sum of defects of such type for a certain road section.

The highest correlation was achieved with transverse cracking, probably because of their unique characteristics in orthoframes. Transverse crackings are perpendicular to the centreline of the paved road area. No other defect type has that same characteristic. The overall defectiveness correlation of 86% and MAE of 3.38% show that despite the issues with some defect types, the model still performs well and can be relied on. For longitudinal cracking and network cracking, the model tends to predict more defects than there are in the ground truth data.

Weathering is not easily recognizable even for humans, for example, in Figure 8.20 the image on the left is the annotation with 2 instances of weathering colored in light-blue while the image on the right (prediction of the network) does not contain any defect. The network misses all the instances of weathering present in the image. Those regions with weathering are characterized by erosion and coarser aggregates.

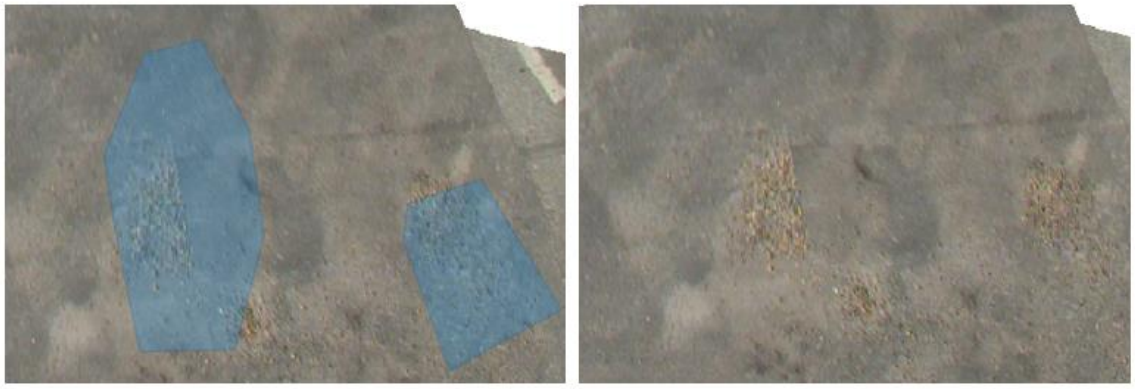


Figure 8.20 Example of weathering not detected by the network. The annotation is on the left and the prediction on the right.

In Figure 8.21, the blue color outlines the annotated defects while the red outlines the prediction of the same type (network cracking). The areas of the predicted defects are larger. The model also predicts multiple consecutive instances of network cracking while the annotation contains only a few. In such cases, the network is more reliable as the orthoframe truly contains multiple instances of network cracking.

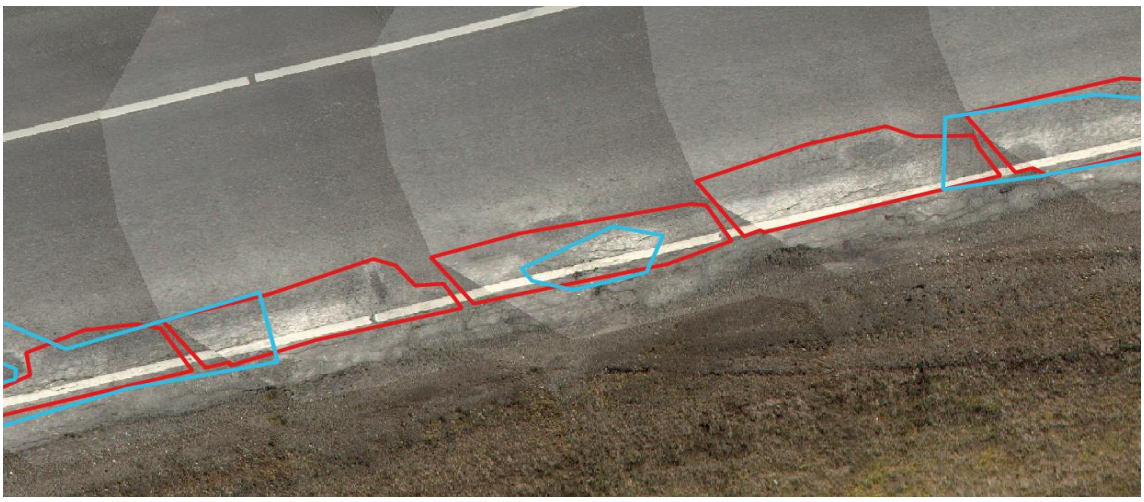


Figure 8.21 Example of predictions of network cracking compared to the annotations. The blue denotes the annotated defects and the red the predictions.

9 SUMMARY

9.1 Conclusions

Automatic and timely road inspection is of high importance for cost-efficient road infrastructure management. Solutions powered by deep convolutional neural networks (ConvNet) offer expert-level performance while reducing the processing time in automating defect detection on roads. ConvNet based models for classification, and semantic segmentation, can detect defective areas of road pavement at a pixel-level accuracy but will fail in separating adjacent defects which ConvNet models for instance segmentation can successfully do in addition to eliminating the need for a pipeline of classification and segmentation networks.

In this thesis, a two-stage instance segmentation model based on ConvNet (Mask R-CNN [31]) was researched and implemented as a baseline for defect instance segmentation. The baseline model achieved an AP_{50} of 69% and 66% for the localization and segmentation tasks, respectively, on a validation dataset made of only defective roads orthophotos' patches.

The baseline network was improved with pyramidal features maps [46] to enhance the ability to learn multiscale defect representations. The new network was then trained on downscaled orthophotos to achieve an AP_{50} of 46.5% and 43.8% on localization and segmentation tasks. While downscaling the orthoframes did result in a decrease in performance, it did enable the network to be able to make more reliable and correct defect predictions on full orthophotos not just defective orthophotos' patches. At this stage, the network can successfully separate adjacent defects even when grouped as one by annotators.

To mitigate the degradation of performance with increasing IoU threshold (used to label object candidates as positive or negative in two-stage object detectors [47]) and to move toward a higher quality detector, a cascade architecture made of 3 stages of detections with increasing IoU thresholds (0.4,0.5,0.6) was adopted following its original implementation for object detection in [47]. This increases the AP_{50} from 46.5% to 48.3% and from 43.8% to 45% for localization and semantic segmentation tasks respectively, while running with an inference speed of 0.045 seconds/image on GeForce RTX 2080Titan GPU machine of 11GB. The increase of performance was possible because the network could differentiate positive defect candidates from negative ones in multiple detections stages better than in single stage.

Finally, a test was performed under realistic scenarios on previously unseen 16.6km of Estonian roads coming from 3 different road sections. The overall correlation of 86% calculated following the definition of pavement defectiveness provided by Estonian Road Administration and the mean absolute error of 3.38% show that even though the network struggles with defect types such as weathering and potholes, it can still be relied on for automatic pavement distress detection.

9.2 Future works

Currently, the dataset distribution is very unbalanced: the main reason for the low performance with potholes is due to their low presence (<0.5%) in the training set. It would be more interesting to validate the results found in this thesis on a more balanced dataset because the overall mean average precision is the mean of average precisions of individual defect types, no matter how well represented they are in the dataset.

A known limitation of two-stage object detection models such as Mask R-CNN used as the baseline in this work is the usage of fixed-sized anchor boxes as references for proposal generation. It has been proven that fixed-sized predefined anchors can cause a decrease in the overall performance of the network [51]. Recently, anchor-free instance segmentation models which allow targets to map any region in the feature map without any restrictions have been studied and researched in the literature [51]–[55]. Even though research on that direction is still in its early stage and suffers in performance, it would be important to consider them as future works; if successful anchor-free models will not just get rid of manually defined anchors but also lead to a probable faster inference and training speeds since most of those models are single stage.

After potholes which have the lowest performance because of their low representation in the training data, the second defect type with a low mean average precision is weathering. As shown in Figure 8.20, weathering defects are characterized by erosion and coarse aggregates. Such characteristics of defects could be better modeled with textural information of roads' pavement. The question of whether ConvNets learn better complex texture or shape information is arguable [56], [57]. To design the next ConvNets models, it would be important to estimate and understand how much shape or textural information is learned in current ConvNet based solutions for pavement detections.

Finally, two-stage instance segmentation models are multitasking models but do not fully benefit from the multitask learning paradigm [58],[59], [60]. In this thesis, for

example, the target objective optimized in the network's detection head is the weighted sum of localization, classification, and segmentation losses and even though the three tasks share common backbone parameters, there are still many parameters specific to each task. For the sake of research, aspects of multitask learning paradigm such as parameters sharing between different tasks and multitask objective optimization can be researched.

KOKKUVÕTE

Tee olukorra jälgimine ja defektide õigeaegne tuvastamine (vältimaks tee olukorra edasist halvenemist) on maanteeinfrastruktuuri haldamisel olulise tähtsusega. Süvaõppe konvolutsiooniliste närvivõrkude põhised automatiseeritud lahendused võistlevad täpsuselt inimannoteerijatega ning on suurema jõudlusega. Sellised defektide klassifitseerimise ja semantilise segmenteerimise mudelid suudavad tuvastada teekatte defektsed alasid piksli täpsusega, kuid ei suuda eristada kõrvuti asuvaid defekte. Instantside segmenteerimiseks mõeldud mudelitel on selline võimekus olemas, samuti on tegu kõik-ühes lahendusega, mis tähendab, et ei ole vaja treenida eraldi klassifitseerimis- ja segmenteerimisvõrke.

Käesolevas väitekirjas uuriti ja rakendati defektide segmenteerimiseks kaheetapilist instantside segmenteerimise mudelit (Mask R-CNN [31]). See mudel saavutas vastavalt 69% ja 66% keskmise täpsuse (AP_{50}) lokaliseerimis- ja segmenteerimisülesannetes valideerimisandmestikul, mis koosnes ainult defektsetest teede ortofotode segmentidest.

Baasvõrku täiendati püramiidsete tunnuskaartidega [46], et parandada mudeli võimet ära tunda eri suurusega defekte. Seejärel treeniti uut võrku vähendatud resolutsiooniga ortofotodel ja saavutati vastavalt 46,5% ja 43,8% AP_{50} lokaliseerimis- ja segmenteerimisülesandes. Kuigi ortofotode resolutsiooni vähendamine halvendas tulemusi, võimaldas see võrgustikul teha usaldusväärsemaid ja täpsemaid defektiprognoose täissuuruses ortofotodel, mitte ainult defektsete ortofoto segmentide kohta. Praeguses etapis suudab võrk edukalt eristada kõrvuti asuvaid defekte isegi siis, kui annoteerijad eelistaksid need üheks defektiks grupeerida.

Selleks, et kompenseerida tulemuste halvenemist objektikandidaatide selekteerimiseks kaheastmelises objektituvastuses [47] kasutatava IoU künnise suurendamisel, võeti kasutusele kolmeastmeline kaskaadarhitektuur, milles kasutatakse kolmeastmelist suurenevat IoU künnist (0,4,0,5,0,6). Tänu sellele tõusis AP_{50} 46,5% pealt 48,3% peale lokaliseerimisülesandes ja 43,8% pealt 45% peale segmenteerimisülesandes. Süsteem kulutab 0,045 sekundit ühe ortofoto analüüsi peale (11 GB GeForce RTX 2080Titan GPU). Jõudluse suurenemine tuleneb asjaolust, et mitme tuvastamisetapiga võrk suudab rohkem defektikandidaate elimineerida kui üheetapiline võrk.

Süsteemi testimiseks kasutati ortofotosid, mille kogumaht teepikkuses on 16.6 kilomeetrit ja mis pärinevad kolmelt erinevalt teelõigult, mida polnud eelnevalt kasutatud treenimisel ega valideerimisel. Korrelatsioon 0,86 ja 3,38%-line keskmine

absoluutne viga võrreldes Eesti Maanteeameti metoodika järgi arvutatud tee defektsusega näitavad, et kuigi võrk on osaliselt raskustes mõnede defektitüüpidega nagu murenemine või (maanteedel harvaesinevad) teeaugud, on seda võimalik kasutada automatiseeritud teekatte defektituvastuses.

LIST OF REFERENCES

[1] "ETIS - Kuluefektiivse ühildatava geodeetilise täpsusega 3D ruumiandmete taristu loomise rakendusuuring." https://www.etis.ee/Portal/Projects/Display/e344ff8c-ce7d-4b03-8db0-ed85cb4a8170?lang=ENG (accessed Nov. 24, 2020).

[2] A. Riid, R. Pihlak, and R. Liinev, "Identification of Drivable Road Area from Orthophotos Using a Convolutional Neural Network," in *2020 17th Biennial Baltic Electronics Conference (BEC)*, 2020, pp. 1–5. doi: 10.1109/BEC49624.2020.9277392.

[3] R. Lõuk, A. Tepljakov, and A. Riid, "A Two-Stream Context-Aware ConvNet for Pavement Distress Detection," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020, pp. 270–273. doi: 10.1109/TSP49548.2020.9163538.

[4] R. Lõuk, A. Riid, R. Pihlak, and A. Tepljakov, "Pavement Defect Segmentation in Orthoframes with a Pipeline of Three Convolutional Neural Networks," *Algorithms*, vol. 13, no. 8, p. 198, Aug. 2020, doi: 10.3390/a13080198.

[5] "Maanteeamet," *Maanteeamet*. https://www.mnt.ee/eng (accessed May 21, 2021).

[6] A. Riid, R. Lõuk, R. Pihlak, A. Tepljakov, and K. Vassiljeva, "Pavement Distress Detection with Deep Learning Using the Orthoframes Acquired by a Mobile Mapping System," *Appl. Sci.*, vol. 9, no. 22, p. 4829, Nov. 2019, doi: 10.3390/app9224829.

[7] W. Cao, Q. Liu, and Z. He, "Review of Pavement Defect Detection Methods," *IEEE Access*, vol. 8, pp. 14531–14544, 2020, doi: 10.1109/ACCESS.2020.2966881.

[8] T. S. Nguyen, M. Avila, and S. Begot, "Automatic Detection and Classification of Defect on road Pavement using Anisotropy Measure," p. 6.

[9] B. Akarsu, M. Karaköse, K. Parlak, E. Akin, and A. Sarimaden, "A Fast and Adaptive Road Defect Detection Approach Using Computer Vision with Real Time Implementation," *Int. J. Appl. Math. Electron. Comput.*, pp. 290–290, Dec. 2016, doi: 10.18100/ijamec.270546.

[10] A. Ayenu-Prah and N. Attoh-Okine, "Evaluating Pavement Cracks with Bidimensional Empirical Mode Decomposition.," *EURASIP J Adv Sig Proc*, vol. 2008, Jan. 2008.

[11] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic Road Crack Detection Using Random Structured Forests," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3434–3445, Dec. 2016, doi: 10.1109/TITS.2016.2552248.

[12] A. S. Marques and P. L. Correia, "Automatic Road Pavement Crack Detection Using SVM," p. 4.

[13] K. Fernandes and L. Ciobanu, *Pavement Pathologies Classification Using Graph-Based Features*. 2014. doi: 10.1109/ICIP.2014.7025159.

[14] D. Zhou, *Texture Analysis and Synthesis Using a Generic Markov-Gibbs Image Model*. University of Auckland, 2006. [Online]. Available: https://books.google.de/books?id=shR8NwAACAAJ

[15] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection," *ArXiv190106340 Cs*, Jan. 2019, Accessed: Jan. 09, 2021. [Online]. Available: http://arxiv.org/abs/1901.06340

[16] S. Xie and Z. Tu, "Holistically-Nested Edge Detection," *ArXiv150406375 Cs*, Oct. 2015, Accessed: Jan. 09, 2021. [Online]. Available: http://arxiv.org/abs/1504.06375

[17] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer Convolutional Features for Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1939–1946, Aug. 2019, doi: 10.1109/TPAMI.2018.2878849.

[18] H. Majidifard, P. Jin, Y. Adu-Gyamfi, and W. G. Buttlar, "Pavement Image Datasets: A New Benchmark Dataset to Classify and Densify Pavement Distresses," *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2674, no. 2, pp. 328–339, Feb. 2020, doi: 10.1177/0361198120907283.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *ArXiv150601497 Cs*, Jan. 2016, Accessed: Nov. 24, 2020. [Online]. Available: http://arxiv.org/abs/1506.01497

[20] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *ArXiv161208242 Cs*, Dec. 2016, Accessed: Jan. 09, 2021. [Online]. Available: http://arxiv.org/abs/1612.08242

[21] K. Zhang, Y. Zhang, and H.-D. Cheng, "CrackGAN: Pavement Crack Detection Using Partially Accurate Ground Truths Based on Generative Adversarial Learning," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–14, 2020, doi: 10.1109/TITS.2020.2990703.

[22] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," *ArXiv14062661 Cs Stat*, Jun. 2014, Accessed: Jan. 09, 2021. [Online]. Available: http://arxiv.org/abs/1406.2661

[23] A. Tepljakov, A. Riid, R. Pihlak, K. Vassiljeva, and E. Petlenkov, "Deep Learning for Detection of Pavement Distress using Nonideal Photographic Images," p. 6.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, ch. 9, pp. 330-371.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[26] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 25, pp. 1097–1105. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *ArXiv150504597 Cs*, May 2015, Accessed: Jan. 07, 2021. [Online]. Available: http://arxiv.org/abs/1505.04597

[29] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2021,ch. 14, pp.427-462. [Online]. Available: http://mlbayes.ai

[30] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous Detection and Segmentation," *ArXiv14071808 Cs*, Jul. 2014, Accessed: Nov. 24, 2020. [Online]. Available: http://arxiv.org/abs/1407.1808

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *ArXiv170306870 Cs*, Jan. 2018, Accessed: Dec. 04, 2020. [Online]. Available: http://arxiv.org/abs/1703.06870

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *ArXiv13112524 Cs*, Oct. 2014, Accessed: Nov. 24, 2020. [Online]. Available: http://arxiv.org/abs/1311.2524

[33] R. Girshick, "Fast R-CNN," *ArXiv150408083 Cs*, Sep. 2015, Accessed: Nov. 24, 2020. [Online]. Available: http://arxiv.org/abs/1504.08083

[34] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *ArXiv14050312 Cs*, Feb. 2015, Accessed: Nov. 24, 2020. [Online]. Available: http://arxiv.org/abs/1405.0312

[35] K. Chen *et al.*, "MMDetection: Open MMLab Detection Toolbox and Benchmark," *ArXiv Prepr. ArXiv190607155*, 2019.

[36] X. Chen, R. Girshick, K. He, and P. Dollar, "TensorMask: A Foundation for Dense Object Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 2061–2069. doi: 10.1109/ICCV.2019.00215.

[37] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully Convolutional Instance-aware Semantic Segmentation," *ArXiv161107709 Cs*, Apr. 2017, Accessed: Apr. 19, 2021. [Online]. Available: http://arxiv.org/abs/1611.07709

[38] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time Instance Segmentation," *ArXiv190402689 Cs*, Oct. 2019, Accessed: May 12, 2021. [Online]. Available: http://arxiv.org/abs/1904.02689

[39] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting Objects by Locations," *ArXiv191204488 Cs*, Jul. 2020, Accessed: May 12, 2021. [Online]. Available: http://arxiv.org/abs/1912.04488

[40] *is-centre/datm-annotation-tool*. Centre for Intelligent Systems, 2021. Accessed: Apr. 22, 2021. [Online]. Available: https://github.com/is-centre/datm-annotation-tool

[41] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, p. 60, Jul. 2019, doi: 10.1186/s40537-019-0197-0.

[42] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations," *Information*, vol. 11, no. 2, 2020, doi: 10.3390/info11020125.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *ArXiv151203385 Cs*, Dec. 2015, Accessed: Jan. 03, 2021. [Online]. Available: http://arxiv.org/abs/1512.03385

[44] Y. Wu and K. He, "Group Normalization," *ArXiv180308494 Cs*, Jun. 2018, Accessed: May 06, 2021. [Online]. Available: http://arxiv.org/abs/1803.08494

[45] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*. 2019. [Online]. Available: https://github.com/facebookresearch/detectron2

[46] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *ArXiv161203144 Cs*, Apr. 2017, Accessed: May 09, 2021. [Online]. Available: http://arxiv.org/abs/1612.03144

[47] Z. Cai and N. Vasconcelos, "Cascade R-CNN: High Quality Object Detection and Instance Segmentation," *ArXiv190609756 Cs*, Jun. 2019, Accessed: Feb. 03, 2021. [Online]. Available: http://arxiv.org/abs/1906.09756

[48] S. Ruder, "An overview of gradient descent optimization algorithms," *ArXiv160904747 Cs*, Jun. 2017, Accessed: May 21, 2021. [Online]. Available: http://arxiv.org/abs/1609.04747

[49] P. Goyal *et al.*, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *ArXiv170602677 Cs*, Apr. 2018, Accessed: May 21, 2021. [Online]. Available: http://arxiv.org/abs/1706.02677

[50] QGIS Development Team, *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2009. [Online]. Available: http://qgis.osgeo.org

[51] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation," *ArXiv200100309 Cs*, Apr. 2020, Accessed: Apr. 08, 2021. [Online]. Available: http://arxiv.org/abs/2001.00309

[52] Y. Lee and J. Park, "CenterMask : Real-Time Anchor-Free Instance Segmentation," *ArXiv191106667 Cs*, Apr. 2020, Accessed: Nov. 23, 2020. [Online]. Available: http://arxiv.org/abs/1911.06667

[53] E. Xie *et al.*, "PolarMask: Single Shot Instance Segmentation with Polar Representation," *ArXiv190913226 Cs*, Feb. 2020, Accessed: Jun. 03, 2021. [Online]. Available: http://arxiv.org/abs/1909.13226

[54] H. ul M. Riaz, N. Benbarka, and A. Zell, "FourierNet: Compact mask representation for instance segmentation using differentiable shape decoders," *ArXiv200202709 Cs Eess*, Oct. 2020, Accessed: Jun. 03, 2021. [Online]. Available: http://arxiv.org/abs/2002.02709

[55] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and Fast Instance Segmentation," *ArXiv200310152 Cs*, Oct. 2020, Accessed: Jun. 03, 2021. [Online]. Available: http://arxiv.org/abs/2003.10152

[56] R. Geirhos, C. Michaelis, F. A. Wichmann, P. Rubisch, M. Bethge, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," p. 22, 2019.

[57] M. A. Islam *et al.*, "Shape or Texture: Understanding Discriminative Features in CNNs," *ArXiv210111604 Cs*, Jan. 2021, Accessed: Jun. 04, 2021. [Online]. Available: http://arxiv.org/abs/2101.11604

[58] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *ArXiv170605098 Cs Stat*, Jun. 2017, Accessed: Nov. 24, 2020. [Online]. Available: http://arxiv.org/abs/1706.05098

[59] R. Caruana, "Multitask Learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997, doi: 10.1023/A:1007379606734.

[60] Y. Zhang and Q. Yang, "A Survey on Multi-Task Learning," *ArXiv170708114 Cs*, Jul. 2018, Accessed: Nov. 04, 2020. [Online]. Available: http://arxiv.org/abs/1707.08114