TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Roland Lõuk 163588IASM

PAVEMENT DISTRESS DETECTION FROM ORTHOPHOTOS WITH TWO-STREAM CONVOLUTIONAL NEURAL NETWORKS

Master's Thesis

Supervisor: Aleksei Tepljakov
Ph.D.

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Roland Lõuk 163588IASM

SÕIDUTEE DEFEKTIDE TUVASTAMINE ORTOFOTODELT KASUTADES KAHE VOOGA KONVOLUTSIOONILISI NÄRVIVÕRKE

Magistritöö

Juhendaja: Aleksei Tepljakov
Ph.D.

Tallinn 2020

Declaration of Originality

Declaration: I hereby declare that this thesis, my original investigation and achievement, submitted for the Master's degree at Tallinn University of Technology, has not been submitted for any degree or examination.

Deklareerin, et käesolev diplomitöö, mis on minu iseseisva töö tulemus, on esitatud Tallinna Tehnikaülikooli magistrikraadi taotlemiseks ja selle alusel ei ole varem taotletud akadeemilist kraadi.

Roland Lõuk

Date: January 2, 2020

Signature: .....

Abstract

Automated pavement distress detection is an important but challenging task towards the goal of timely road maintenance. Given the vastness of road networks across the world, there is a lot of labor involved in manual defect detection for roads. In the recent years, however, convolutional neural networks have been shown to achieve groundbreaking results in the field of image classification. This thesis seeks to research and develop methods for applying convolutional neural networks to pavement distress detection for sections of orthophotos (orthoframes) with a large resolution. To address GPU memory limitations and increase detection localization, a sliding-window approach is used to partition the orthoframe into 224×224-pixel segments, which are subject to binary classification. However, the sliding-window approach does not allow for the model to account for the context surrounding the segment and results may suffer due to the small window size. This thesis proposes a ResNet architecture based convolutional neural network which accounts for two inputs streams, one of which is the 224×224-pixel content segment, which is subject to classification, and the other is the downscaled context view around the content segment. Experiments on two different datasets show an increased classification accuracy for the two-stream approach compared to the single stream approach.

The thesis is written in English and it contains 58 pages of text, 7 chapters, 5 tables and 19 figures.

Annotatsioon

Sõiduteede automaatiseeritud defektide tuvastamine on oluline, kuid keeruline ülesanne, et tagada teede õigeaegne hooldus. Arvestades teedevõrgustike ulatuslikkust globaalsel tasandil, kulub teede defektide käsitsi tuvastamisele suur hulk tööjõudu. Teisalt on viimastel aastatel konvolutsioonilised närvivõrgud näidanud piltide klassifitseerimise valdkonnas murrangulisi tulemusi. Käesoleva väitekirja eesmärk on uurida võimalusi ja lahendusi, et rakendada konvolutsioonilisi närvivõrke sõiduteede defektide tuvastamiseks, kus lähteandmeteks on suure resolutsiooniga ortokaadrid. Suurendamaks defektide tuvastuse lokaliseeritust ja mahutamaks sisendandmed GPU mällu, tükeldatakse ortokaadrid 224×224-pikslisteks segmentideks, mis saavad olema binaarse klassifikatsiooni lähteandmeteks. Antud lähenemise nõrkuseks on asjaolu, et arvesse ei võeta segmenti ümbritsevat konteksti, mis võimaldaks klassifikatsiooni protsessile kaasa aidata. Käesolevas töös pakutakse välja ResNet arhitektuuri baasil töötav konvolutsiooniline närvivõrk, millel on kaks sisendvoogu. Üheks sisendiks on 224×224-piksline segment, mille põhjal toimub klassifitseerimine ning teiseks sisendiks on vastavat segmenti ümbritsev kontekstivaade. Kahel andmestikul läbiviidud eksperimendid näitavad, et antud kahe vooga konvolutsiooniline närvivõrk on suurema klassifitseerimistäpsusega, kui ühe vooga närvivõrk.

Antud lõputöö on inglise keeles ning sisaldab teksti 58 leheküljel, 7 peatükki, 5 tabelit ja 19 joonist.

Nomenclature

ConvNet	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
FN	False Negative
FP	False Positive
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
LIDAR	Light Detection and Ranging
MCC	Matthews Correlation Coefficient
RGB	Red, Green, Blue
TN	True Negative
TP	True Positive

Contents

List of Figures

List of Tables

1. Introduction

Road cracking and other types of pavement defects are a hindrance to the vehicles using the road and need to be identified in order for the road to be maintained timely. In Estonia, the distress on roads is amplified by the extreme temperature shifts taking place throughout the year, which means roads need to be inspected with a high frequency.

Reach-U Ltd, a company offering solutions in industries related to GIS and cartography, created a method to acquire panoramic photos and orthophotos via cameras mounted on a car in motion [4]. From the captured panoramic photos and derived orthophotos, trained personnel are used to mark down the defects with their categories and bounds. With current methods, a human is able to cover from 3 to 20 kilometres of road per day. However, given that as of January 1st, 2019, Estonia had a total of 40 610 kilometres of public roads, manual annotation of all roads is a massively laborious task. Moreover, it is also prone to human error due to, e.g., fatigue or change blindness [5].

With a growing amount of research in computer vision algorithms over the past decades, researchers have also begun investigating ways to apply computer vision techniques to automate the task of pavement distress detection. Fueled by the increasing computational resources due to more efficient hardware, convolutional neural network based algorithms have become particularly popular for image based road analysis [6, 7, 8, 9].

1.1. Objectives and Contributions

Using the data provided by Reach-U Ltd. and the computational resources provided by the department of computer control of Tallinn University of Technology, the objective of this work is to research and experiment with deep learning based computer vision techniques in order to come up with a a model that is capable of identifying road defects from orthophotos. Specifically, the task is to create a binary classifier for partitioned segments of the orthophoto with convolutional neural networks, which outputs whether the given input segment contains a road crack or not.

This thesis seeks to contribute by improving upon the previously proposed sliding-window classifier [10] by incorporating an additional image input stream for the classifier. The newly proposed two-stream approach addresses some of the shortcomings of the single stream sliding-window approach, which does not take into account the contextual information surrounding the partitioned segment input.

1.2. Thesis Outline

Chapter 2 begins with a literature review of research related to the one presented in this thesis. Then, the reader is given introductory concepts of deep learning based modeling and a brief review of software frameworks for deep learning is reviewed, in which the software choices of this work are justified. Lastly, the process of acquiring orthophotos provided by Reach-U Ltd. will be explained.

Chapter 3 covers and justifies steps taken to prepare the acquired orthophotos for deep learning tasks. This includes orthophoto analysis, data preprocessing and generation of datasets.

In Chapter 4, the reader is introduced to convolutional neural network based modeling in detail. Accompanied with theoretical knowledge are also the choices made for the models trained in this thesis. At the end of the chapter, an architecture for a convolutional neural network is presented for pavement defect detection.

In Chapter 5, a two-stream convolutional neural network based architecture is presented to address some of the shortcomings of the simpler architecture seen in Chapter 4.

Finally, in Chapter 6, all of the models trained in the context of this work are evaluated with their obtained performance metrics. Advantages and drawbacks of the best performing approaches are analyzed on a case-by-case basis.

2. Background

2.1. Related work

2.1.1. Automated pavement distress detection

Automating the process of identifying road defects from images has been in the interest of academic literature for the past few decades. Earlier approaches in this subject involved techniques such as elaborate image-enhancement algorithms [11], neural networks in conjunction with thresholding and moment invariants [12], wavelet transforms [13] and numerous others. More recently, convolutional neural network based models have dominated the field of research, as they have become vastly better at arbitrary image classification tasks than other methods.

In 2018, Gopalakrishnan [14] analyzed 12 recent papers on deep learning based pavement image analysis, highlighting different strategies and common difficulties in this task. The author concluded that interest in deep learning applications for pavement distress was rising fast, as two papers were published in 2016, seven in 2017 and three in 2018 (up to March). Nearly all of the published works on pavement distress in this timeframe included the usage of convolutional neural networks. The main difficulties in this task were concluded to be presence of shadows, ambiguity of defects in photos and illumination issues, which makes automated pavement distress detection still an open problem.

Like this work, many of the papers published in crack detection use a sliding-window approach, where the raw image is first segmented into small windows and predictions are made per segment [6, 9]. Zhang *et al.* [15] proposed an additional high-resolution analysis for segment-based classification, by combining the cracking classification and detection networks.

Notably, our team showed that convolutional neural networks are a feasible solution for segment-based classification of road cracking by utilizing methods of transfer learning to accelerate the training process [10]. The dataset and models used in that work are partially overlapping with the ones presented in this thesis.

To the best of author's knowledge, context-aware methods for segment classification have not been researched for the task of pavement distress detection, as many of the public pavement orthophoto datasets seem to only include smaller segments of the road without extra context. The public datasets reviewed include CRACK500 [8], GAPs384 [16], CFD [17], AEL [18] and cracktree200 [19]. All of these datasets were accessed from the public repository provided by Yang *et al.* [20].

2.1.2. Multi-stream convolutional neural networks

In 2016, Liu *et al.* implemented a two-stream convolutional neural network for fine-grained image classification [21]. They extracted features from a given image with two VGG-16 networks [22] in order to capture both the object to be segmented and the background in separate streams. The two feature extractors were connected at the classification layer to produce the final output. They showed that this approach improves the results for tasks such as flower and bird species segmentation due to the separated streams for context and content information.

Pang *et al.* [23] noted that simple convolutional neural networks are not well suited for the task of small object detection from high-resolution images, thereby proposing an improvement to the segment-based classification by using a convolutional neural network with a global attention block, which reduced false positives in large-scale remote sensing images.

There are similarities between the task of pavement distress detection from orthoframes and healthcare related problems such as tissue cell classification based on high resolution microscopy images. Namely, in both tasks, high resolution of the image prevents them from being transferred into the GPU memory in its entirety and the important details for classification might be located in a very small portion of the image. To address these hurdles, Tomita *et al.* [24] proposed a method of classifying segments of the whole-slide microscopy images by incorporating grid-based attention from nearby segments. Their approach showed an improved result compared to the typical sliding window segment-based approach. Similarly, Shaban *et al.* [25] proposed a method to incorporate contextual information from the 1792×1792-pixel image for a 224×224-pixel segment classifier for colorectal cancer histology images.

2.2. Machine learning and deep learning

Machine learning is the study of algorithms which are able to deal with tasks without having been explicitly programmed [26]. A common machine learning task is supervised learning, in which the machine is "taught" to classify inputs by learning from previously annotated *training* data.

The goal of supervised learning is to produce a classifier that is able to generalize well enough to produce correct labels for new input data from a distribution similar to the training data. To accomplish this, the annotated training data needs to be large enough to enable the model to encode relevant dependencies between its internal variables [27].

Overfitting or inability to generalize beyond the training data is a common pitfall in machine learning algorithms, which can be caused by models that are overparameterized, lack regularization or are overtrained on an insufficient training data. On the other hand, underfitting can occur when the model is incapable of capturing the pattern we are trying to model due to insufficient or improper parametrization [2]. Therefore, a robust machine learning model is one which finds the optimal regime between overfitting and underfitting.

With machine learning, it is necessary that the most relevant features of the source data are extracted in order to avoid the curse of dimensionality [28]. However, for deep learning based models, such as artificial neural networks, feature extraction is performed by the model.

2.3. Artificial neural networks

Inspired by biological neural networks of animal brains, the artificial neural network is a black-box modeling technique that is increasingly commonly used in the field of deep learning. The basic building block of the neural network is the node called neuron, which produces an activation in response to inputs that can then be used as an input for another neuron or a function. The output of a neuron can be written as:

$$y = \sum_{i=1}^{n} x_i w_i + b, \tag{1}$$

where x_i is the i-th input, w_i is the i-th input weight and b is the bias. To allow for modeling non-linear relationships, the neuron's output is also fed through a nonlinear activation function such as the sigmoid function.

By training the network, our goal is to optimize the weights and biases of the collections of neurons such that they would provide a better approximation for our desired function (this is expanded upon in Section 4.5.1).

According to the universal approximation theorem, the neural network is capable of approximating any continuous function with only a single layer of neurons and nonlinear activations, with the caveat that reaching the correct states for the neurons might not be learnable by known techniques [29]. Most commonly, though, the neurons are arranged in layers (refer to Figure 1), which results in an increased ability to model complicated functions.

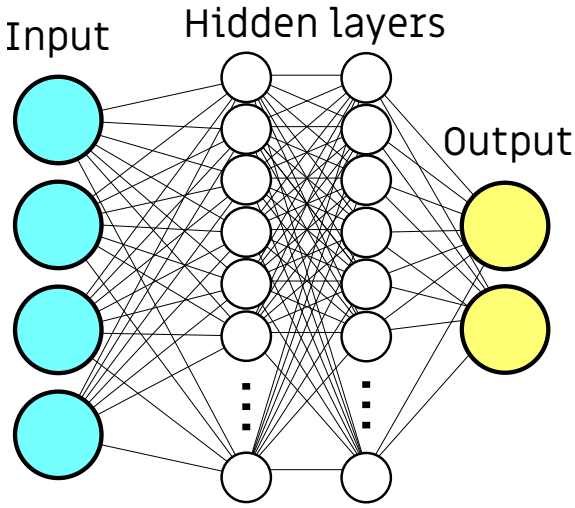


Figure 1. A fully connected neural network with 4 inputs, 2 hidden layers and 2 outputs.

2.4. Convolutional neural networks

Convolutional neural networks or ConvNets are a family of neural networks that were designed to process visual data by applying the convolution filter to its input. The core idea behind the ConvNet is to arrange these filters in such a way that meaningful features are possible to be extracted from any data that is arranged in a grid, such as an image. By processing the input through a filter, we obtain a new set of features from the initial image that can then be used as an input for another filter

to process, therefore extracting finer features with each passthrough. This process is inspired by how the visual cortex of the mammalian brain works [30].

The benefits of convolutional neural networks include:

- Computational efficiency. To process a 3-color 224×224-pixel image through a fully connected feed-forward neural network, we would begin with 150528 inputs. If we were to have a single layer of a 1000 neurons, we would be required to optimize around a 150 million parameters just for a single hidden layer. Meanwhile, a typical ConvNet architecture has around 1 to 100 million parameters in total that need to be optimized for the entire network, thus being more memory efficient than the fully connected feed-forward neural network.
- Shift invariance. As the convolutional filters parse the image with a small window size and a certain step, they are able to detect the patterns anywhere in the input without discrimination.

Commonly, the features obtained through the layers of convolutional filters are used as inputs for a fully connected neural network (as seen in Figure 1), which outputs the classification result.

2.4.1. Prominence of ConvNets in image classification

While ConvNets were used in practice as early as 1998 with the LeNet architecture for classifying 32x32 resolution hand-written digits [31], they remained unpopular for years to come due to their large computational costs. It was until around 2010, when Ciresan *et al.* won multiple image recognition competitions [32, 33, 34] by training ConvNets with GPU acceleration from Nvidia's CUDA, which accelerated the training process by 40 times.

Therefore, a big boost to ConvNet performance came from the advancement of GPUs to accelerate matrix computations, which is what Krizhevsky et al. also took advantage of in 2012. They demonstrated the capabilities of ConvNets with *ImageNet Classification with Deep Convolutional Neural Networks* [35], achieving a 37.5% error rate on a 1000-class classification problem, thereby winning the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012 [36]. Since then, computer vision

solutions of image classification problems have been dominated by convolutional neural networks trained on GPUs.

2.5. Software frameworks for deep learning

As deep learning based ideas have become more and more prominent both in academia and industry, there has also been an increase in the variety of different software frameworks that make it simpler to implement different deep learning solutions.

Most of the popular frameworks nowadays serve as libraries for the Python programming language [37], which is prefered due to its easy to understand code, big community and built-in libraries. As for other possible languages to consider, R [38] is a popular language among statisticians for statistical computing and data mining. MATLAB [39] with its machine learning toolbox is well-developed that could be used for prototyping purposes, as well as visualization of data. However, specifically for training deep neural networks, Python seems to be the most popular choice nowadays. For example, in the review of deep learning based pavement analysis papers by Gopalakrishnan [14], at least 11 of the 12 analyzed studies used a deep learning framework written for Python interface (one of the studies had not specified used libraries).

The most popular deep learning frameworks for Python include Tensorflow [40], Keras [41], PyTorch [42], Theano [43] and Caffe [44]. For the purposes of this work, the ease of ability to prototype with different convolutional neural network architectures was considered the key criterion in choosing the framework. The fastai library [45], which uses PyTorch as backend, was used for experimenting with transfer learning based ConvNet architectures, as it provides "out of the box" support for ConvNet training of various architectures. For more custom designs such as the two-stream ConvNet, it was decided to use PyTorch. In the past few years, PyTorch has been well-received in the academic community [1], as indicated by the amount of mentions in arXiv papers in Figure 2.

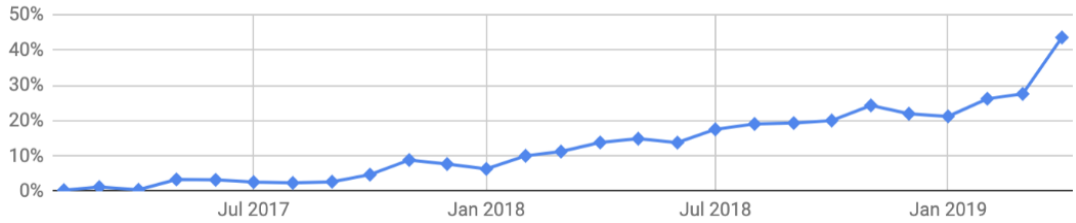


Figure 2. Percentage of arXiv deep learning papers that mention PyTorch [1].

2.6. Mobile mapping system for pavement analysis

In order to facilitate road analysis digitally, Reach-U Ltd. developed a mobile mapping solution for cars, which enables the possibility of capturing panoramic photos and geospatial data from a car that is travelling up to 80 km/h. The car is equipped with a LIDAR system (Light Detection and Ranging) for the purposes of generating a point cloud dataset and camera system to capture visual information and a GNSS (Global Navigation Satellite System) to match the geographic coordinates with the acquired images. In the context of this work, we will not be using data from the LIDAR system as we are focused on the techniques of image classification.

Therefore, the relevant components of acquiring high quality orthophotos with their geographic coordinates are:

- Ladybug 5+ spherical imaging system capturing panoramic photos at a resolution of 30 megapixels. A pixel 10 m from the camera would correspond to an area with a diameter of 0.78 cm, giving us sufficient detail to analyze the pavements for distress from the photos.
- GNSS (Global Navigation Satellite System) to gather geographical data for taken photos. With the real-time kinematic positioning technique, it is capable of achieving a positioning accuracy of 10 cm.
- Reach-U software for synchronizing GNSS and image collecting systems.

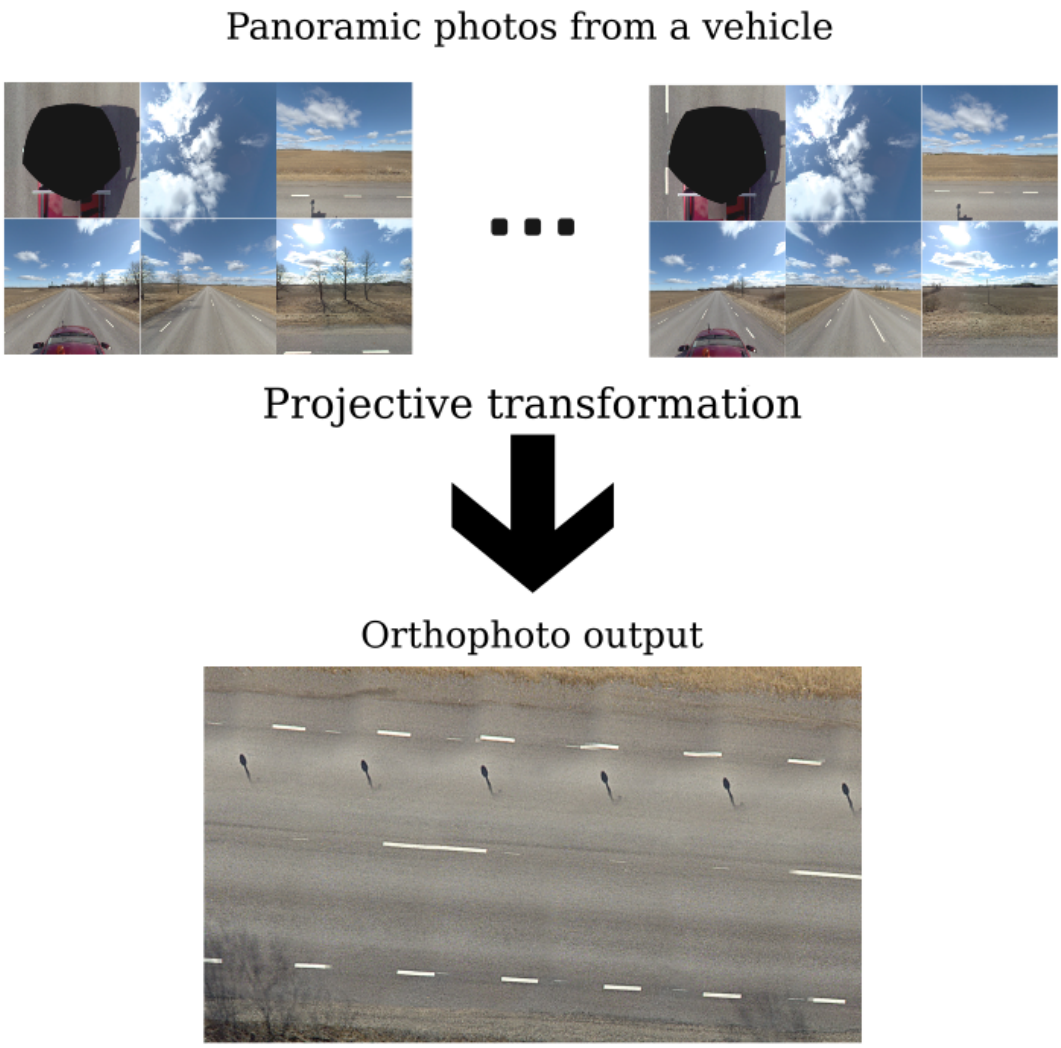


Figure 3. Acquiring orthophotos from panoramic photos.

2.7. Producing orthophotos

After the panoramic photos of the road are captured, they are able to be "stitched" together by applying a projective transformation technique, the result of which is depicted in Figure 3. Thus, it is possible to view long sections of the road in a single high-resolution image, while the image is also accompanied by its respective geospatial information, allowing us to match precise geographical coordinates of given roads. The exact algorithm of this process is part of the proprietary software developed by Reach-U and hence will not be discussed in this thesis.

3. Data acquisition methodology

3.1. Analysis of the orthophotos

While the level of detail captured in the generated images by the Reach-U mobile mapping method is vastly superior compared to satellite orthophotos, we are still left with various problems that may hinder the evaluation process of the roads:

- Cars, pedestrians or cyclists can be captured in the photos, covering patches of roads. In addition, these unwanted sections of the image might be enlarged because of the projective transformation applied to the panoramic photos.
- Inconsistent quality across sections of the images. Parts of the photos that were further from the camera are inevitably less detailed. Past a certain distance, the road cracks might even become indistinguishable from the road texture.
- The images suffer from various shadows cast on the road. Depending on the relative position of the sun, the shadows could also be cast by the vehicle that is used for taking the photos, leaving trails of shadows (this can be noticed in the output of Figure 3).
- Misalignment of the road due to stitching. This was especially noticeable upon close inspection of road markings (see Figure 4).

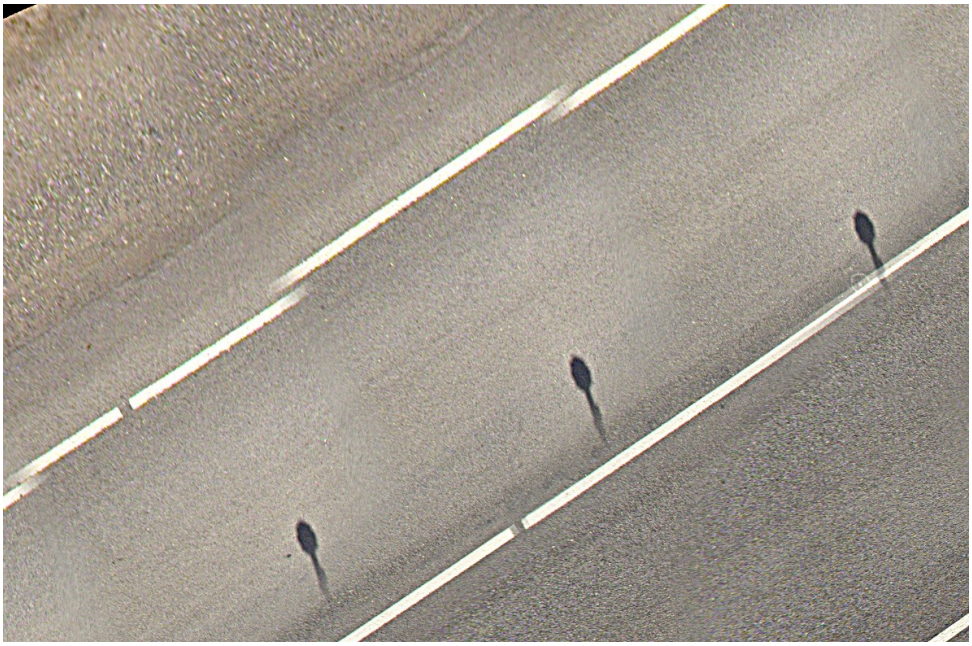


Figure 4. Persistent camera shadows and misaligned stitching from a generated orthophoto.

3.2. Orthophoto preprocessing and annotation

As described in our previous research [10], we decided to focus on smaller 4096 × 4096-pixel portions of the generated orthophotos, which shall be refered to as the orthoframes. This choice was made to eliminate the misalignment issues due to stitching. In addition, the orthoframe is further reduced in size by applying a mask with the radius of 1500 pixels to the center of the frame. This is done because the pixels closer to the center were closer to the camera and thus more detailed. Fortunately, this reduction did not result in "blindspots" between consecutive orthoframes, as they were already partly overlapping.

The provided dataset of Estonian roads had each of its orthophotos supplied with manually marked lines or bounding boxes of the defects, in the format of a *shapefile*. However, a noticeable portion of these annotations were seemingly misplaced or even missing altogether, which led our team to manually redigitize the orthoframes of interest. For this purpose, a mask-based annotation tool was developed by A. Tepljakov [46], which was used to draw masks of the defects as well as the boundaries of the road (see Figure 5).

Annotation was carried out by four members of the team and from all of the potential

road defects, only cracks were marked down, as cracks are easiest to notice without expertise and they are the most common form of road defect.

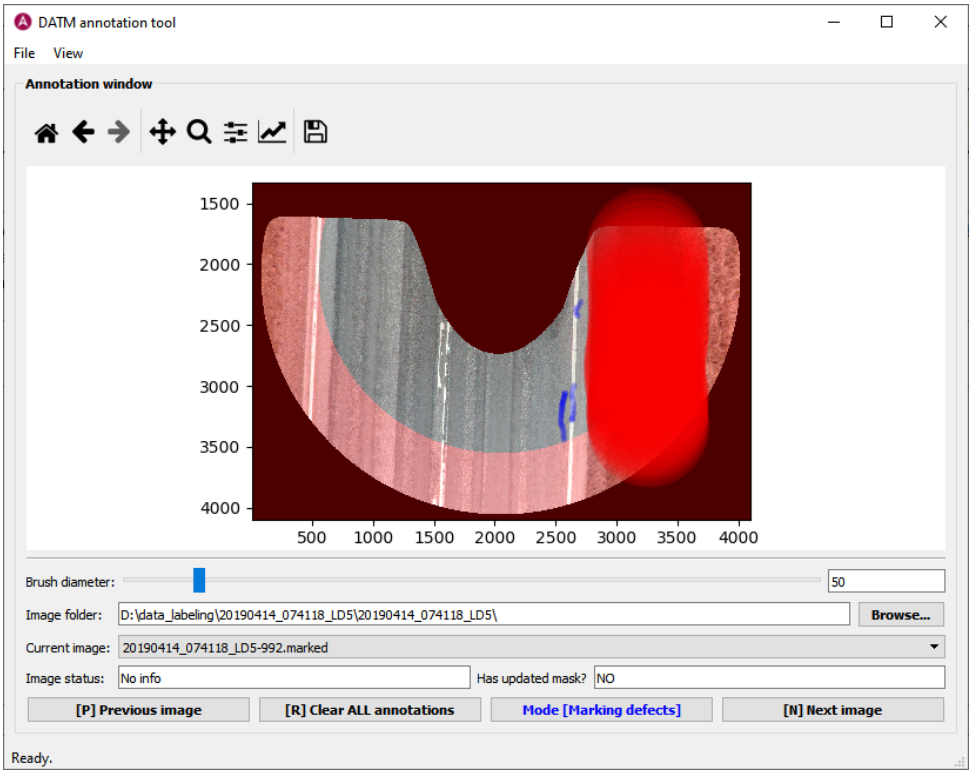


Figure 5. Annotation of an orthoframe with the DATM annotation tool. Blue masks drawn correspond to defects, red masks drawn correspond to the road edge that should not be used in the dataset.

3.3. Dataset selection

All of the orthoframes were preselected to include at least one defect, as data was abundant and most of the orthoframes were free of defects. In total, 1572 orthoframes were manually digitized for the purposes of our previous work [10]. Due to some circumstances, only 1168 of these orthoframes were able to be used in this work. These orthoframes came from 3 different roads and were used for training the models.

For monitoring the performances of the model, two different datasets were considered:

1. Validation set with 55 orthoframes, for which the orthoframes were selected from four different sessions preselected to have crack defects. Additionally, orthoframes with shadows, overly ambigious looking cracks or with other

difficult circumstances were not sampled. This was also the test set used to evaluate models in [10]. The purpose of this validation set was to calibrate the hyperparameters of the trained models and monitor how well the trained models generalized.

2. Test set with 90 orthoframes, for which orthoframes were randomly selected from three different trips, without discarding any of them. This means many of the defects are ambigious and that the orthoframes may contain shadows or other potentially confusing circumstances that made even digitizing them a difficult task (as seen in Figure 6). The purpose of this dataset is to evaluate how well the models would work in realistic conditions, where none of the orthoframes were preselected.

Figure 6. Difficult to assess part of the orthoframe belonging to test the set. The difficulties include shadows cast by trees, obfuscated alligator cracking on the left side of the road, inconsistent lighting, a dark patch and unclear road margins.

3.4. Segment generation for ConvNet training data

Typical ConvNets are able to work with fixed-size inputs and increasing the resolution of the input results in higher GPU memory requirements, especially given that during training it is beneficial to process batches of 8 to 64 images in parallel [47]. Therefore, in order to use deeper ConvNet architectures, we will need to partition the orthoframe into smaller windows that will be refered to as segments. An example is given in Figure 7. The segments were chosen to be of size 224×224-pixel, as bigger sizes will have troubles fitting into the 8GB of GPU RAM that the used Geforce GTX1080 graphics card provides. The orthophoto partinioning approach for our dataset was first proposed by Tepljakov *et al.* [48].

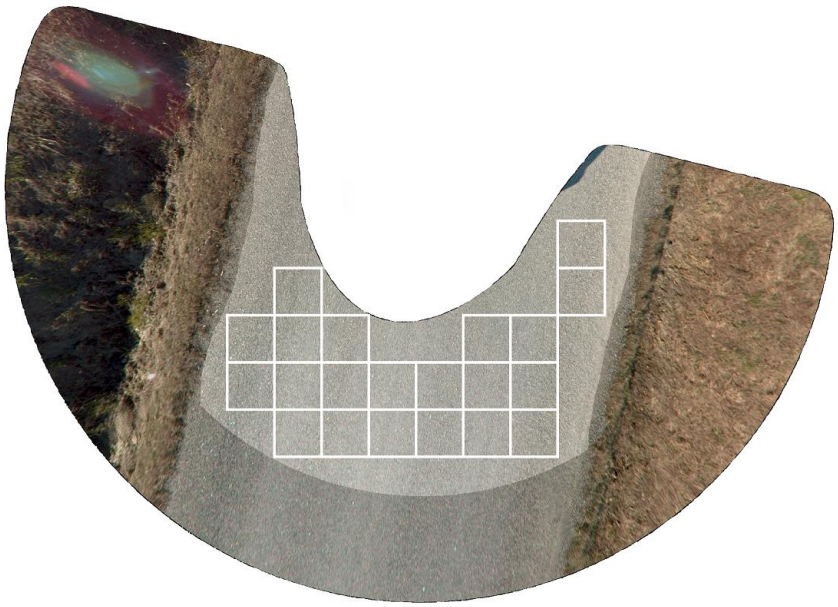


Figure 7. Orthoframe partitioned into 21 segments of size 224 by 224 pixels. The highlighted section refers to the area of the orthoframe from which segments are extracted.

For our previous work [10], we generated N non-overlapping segments from each of the orthoframes, such that $N_{defects} = N_{nondefects}$, as deep learning networks perform best when trained on a balanced dataset [49]. Given that the vast majority of segments did not include a crack defect, we chose only a select few of the more represented class (usually the "nondefected" class) randomly in order to match segment counts per orthoframe. This sampling strategy is called undersampling.

For this work, the oversampling strategy was tried, so every possible generated segment was included in the dataset. In order to enforce that $N_{defects} = N_{nondefects}$,

copies of the less represented class need to be made per orthoframe. The benefit of this approach is a much larger training set as no segment is discarded from the dataset. However, the natural concern is the lack of diverse data for the defected segments (due to copies) and the possibility of overfitting. Interestingly, even with imbalance ratios of over 20, the oversampling technique can work without causing overfitting [49]. In addition, data augmentation techniques (described in Section 3.5) are able to be used to alleviate the data diversity concern caused by oversampling.

The two-stream ConvNets developed in this work takes two 224×224-pixel RGB inputs. One of those inputs is the "content" segment, which is subject to classification. The other input is the "context" segment, which gives an overview of the area around the content segment. This is visualized as the input in Figure 8. The raw input of the context segment has the shape $[672 \times 672 \times 3]$ and it is resized to $[224 \times 224 \times 3]$. The content-context segment pairs are generated such that the content segment lies exactly in the center of the context segment. The label for a given pair is "defected" if over 5% of the pixels in the content segment are masked as such and "not defected" otherwise.

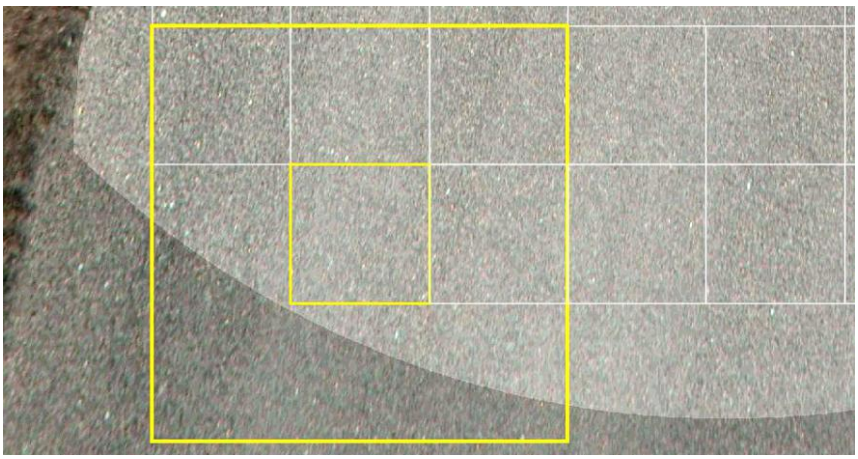


Figure 8. A close up on the segmented orthoframe. For illustrative purposes, one of the content segments (inner yellow square) is accompanied with its surrounding context area (outer yellow square).

3.5. Data augmentation

In order to increase the robustness of deep learning based computer vision models, it is common to apply various image operations to the input samples, which is a way to effectively synthesize new data [2]. For example, to prevent the network from

learning possible correlations between the road defects and the road orientation (due to sparse data), we could apply a random rotation to the training samples. Similarly, if many of the defects were captured in orthoframes with bright lighting conditions, the network could create a correlation between brightness and defects, which we do not want to happen.

Fortunately, for the case of pavement analysis from orthophotos, there are many possible augmentation techniques that can be applied as the data is still realistic if flipped, rotated or slightly shifted in color channel values. For the models trained in this work, we randomly apply each training sample per training epoch with random transformations as seen in Table 1. Potential outputs of these transformations can be seen in Figure 9. For the two-stream ConvNet, the same transformation is applied both to the context and the content segment per sample pair.

Table 1. Data transformations applied to each training sample (different for each training epoch).

Data transformation	**Probability to apply**	**Range**
Rotation	0.5	$(-180, 180)$ degrees
Horizontal flipping	0.5	–
Vertical flipping	0.5	–
Brightness and contrast adjustment	0.5	$+/-35\%$
RGB value shift	0.5	$+/-25\%$

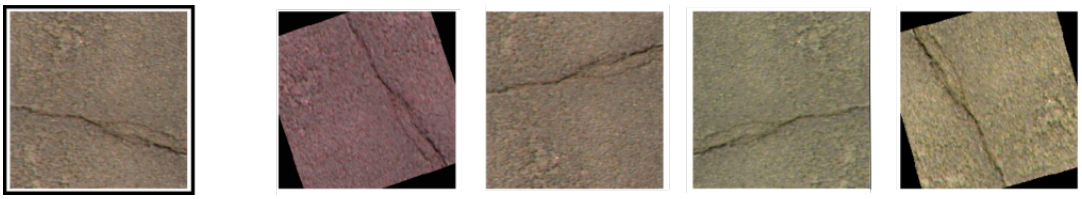


Figure 9. The original training sample (left, framed) and its possible transformations to be used in training.

4. ConvNet based modeling

This chapter will go over the building blocks of the ConvNets used in this work and also the steps taken to optimize them for the task of pavement defect classification. As a reference, an illustration of a typical convolutional neural architecture with its main building blocks can be seen in Figure 10.

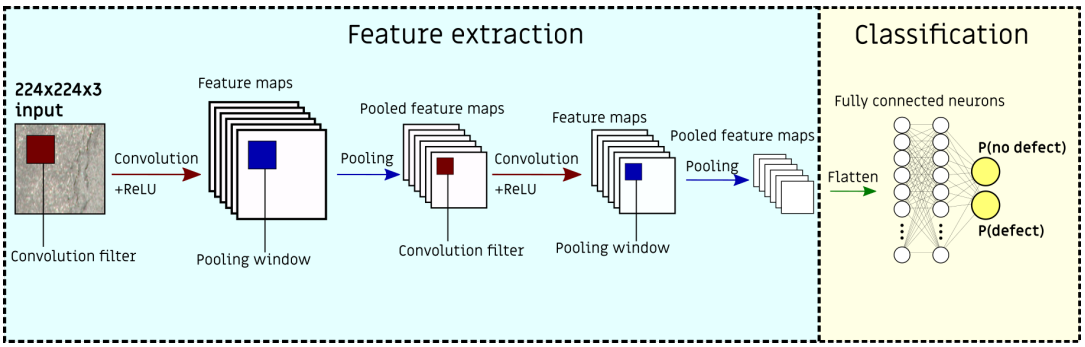


Figure 10. A scheme of a typical convolutional neural network.

4.1. Convolution filter

The convolution filter (also known as the kernel) is a matrix of weights arranged in $N \times N$, where N is usually between 2 and 7. This filter acts as a window which slides (convolves) through the width and height of the input, where in each step the element-wise multiplication will be applied and the sum of these products will be the value of the output (feature map). Typically, the window slides with a stride of 1, meaning no pixels are skipped over. Larger strides can be used as a way to capture information of the input more sparsely.

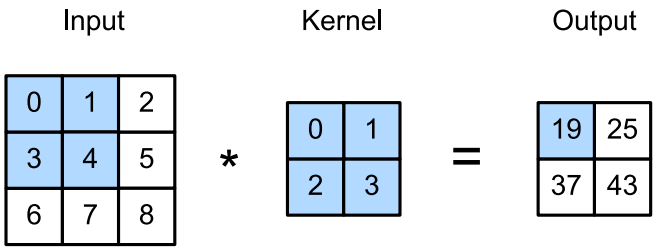


Figure 11. A 2×2 convolution filter applied to a 3×3 input. [2]

The filter is applied across all the channels or depths of the input. For an RGB image

input, the filter needs to be of shape $N \times N \times 3$. For the layer after that, the filter needs to be of shape $N_1 \times N_1 \times F$, where F is the number of filters in the previous layer and N_1 is the width and height of the feature maps in the previous layer.

Padding is introduced to ensure all of the pixels of the input get covered by the convolution filter and no information gets lost. Padding is usually done by augmenting the borders of the input with either zeros or by replicating the values at the borders.

4.2. Activation function

Connecting the outputs of filters or neurons to another filter or neuron allows us to only make linear combinations with the combined weights, therefore preventing us from reaping the benefits of utilizing multiple hidden layers [2]. For this reason, a non-linear activation function is crucial to have for each connection of the neural network architecture.

Every connection except for the final classification layer in this work uses the rectified linear unit (ReLU) as the activation function, which is a simple and fast non-linear function. ReLU is defined as:

$$\mathrm{ReLU}(z) = \max(z, 0). \tag{2}$$

Therefore, ReLU only retains positive inputs and discards negative inputs.

For the final layer, it is common to use the softmax activation function to attain the probability distribution as the output of the network. The two-class softmax function used in this work can be written down as:

$$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{e^{z_0} + e^{z_1}}, \tag{3}$$

where z_0 and z_1 represent the neurons for the "nondefect" class and "defected" class respectively.

4.3. Pooling

The maximum pooling operation takes as an input a window (typically 2×2) of a given tensor and outputs the maximum singular value of the window, thus producing a downscaled output. Like with the convolutional filter, the pooling window slides across the width and height of the image.

The reason for downscaling the convolution maps is to learn a more global representation of the features for the later layers [2]. The ResNet architecture (which this thesis heavily relies upon) uses only a single max-pooling layer after the first convolution layer [3].

Average pooling for the ResNet is used for the last convolution layer, which calculates the average values of the elements in the pooling window.

4.4. Batch normalization and dropout

It has been demonstrated that normalizing the inputs of every layer per input batch results in faster convergence towards the local minimum and increased generalization [50]. For each batch, the input at a given layer is normalized by subtracting the mean of the batch and dividing by the standard deviation of the batch. In this work, batch normalization is applied to each convolutional and fully connected layer of the networks.

Another commonly applied regularization technique is the dropout method [51]. The dropout layer simply removes a portion of randomly chosen connections between layers per training batch (usually between 25% and 75%), thus forcing the network to learn a more generalized representation of the features during training.

It must be noted that applying both batch normalization and dropout in conjunction may in some cases be disadvantageous [52], thus a 50% dropout is only used for the first fully connected layer for models in this work.

4.5. Training

Neural network training is a process which involves optimizing the weights and biases of the network such that the network would produce the desired output with respect to the given input. In our case, the input is a 224 × 224-pixel RGB image and the desired output is either "nondefect" or "defect".

The models covered in this thesis were trained on a Geforce GTX1080 graphics card, provided by the department of computer control of Tallinn University of Technology.

4.5.1. Gradient descent, backpropagation and the loss function

One of the key algorithms that has been the backbone of deep learning is the gradient descent optimization algorithm with backpropagation. The idea of gradient descent is to calculate the derivatives of every weight with respect to the loss function of the output via backpropagation and then subtracting a small portion of the respective derivative from each respective weight.

The cross-entropy loss function was used in training the models in this work:

$$L = -(y \log(p) + (1 - y) \log(1 - p)), \tag{4}$$

where L is the cross-entropy loss, $y \in \{0, 1\}$ is the label for the given input and $p \in [0, 1]$ is the model's given probability for the correct label.

As is standard practice for ConvNet training, models trained in this work were trained with the mini-batch gradient descent algorithm [53]. Training samples were shuffled into batches of 16 and the weights were optimized according to the mean gradients of each sample in the batch. In PyTorch, gradient computation is done for all the parameters of the network with the *backward()* function.

For the optimization algorithm, *Adam* was used, which is a gradient-based optimization algorithm has been shown to often work better than stochastic gradient descent [54].

4.5.2. Learning rate scheduling

The learning rate is the proportion of the respective derivative that is to be subtracted from the respective weight after each processed batch. The learning rate scheduler handles how the learning rate changes during the process of training.

For models trained in this work, the training began with a learning rate of 10^{-4} and decreased linearly during the training process until it reached 4×10^{-5}, which happened after 5 epochs.

4.6. Transfer learning

Transfer learning refers to using models with pre-trained weights as a starting point for the new task. By using transfer learning we are taking advantage of the fact that filter weights trained on photographic images, even from completely different scenarios, are likely to be a better starting point than completely randomly initialized weights. The result is an accelerated training process and an increased ability for generalization, due to the vast datasets they were previously trained on [55].

A common approach for ConvNets with transfer learning is to freeze the feature extraction part so only the classification layers are optimized. Another approach is to optimize the feature extraction layers with a different learning rate than the classification layers. These approaches were used in our previous work [10]. However, it became noticeable that similar results were attainable with a simpler training scheme. Therefore, all models used in this work were fully unfrozen and trained in one loop.

4.6.1. ResNet

All of the models presented in this work were using pretrained ResNets optimized for the task of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [36], which provides 1.2 million training images with 1000 classes (such as dog breeds, food, vehicles). For this daunting task, the ResNet-50 architecture was reported to achieve a 20.74% error rate, which is on par with human performance [3].

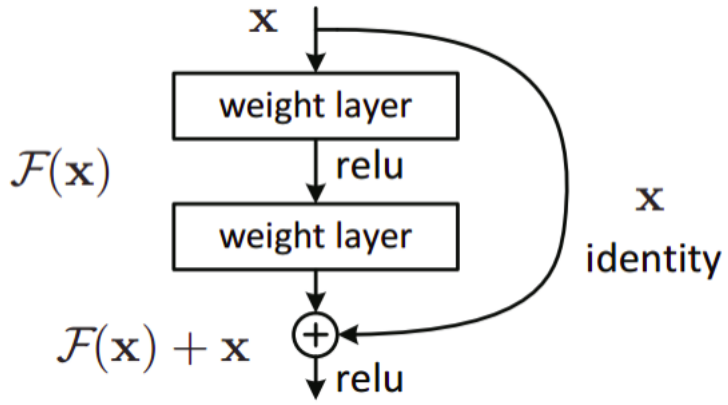


Figure 12. The residual block of the ResNet architecture. [3]

The ResNet architecture is mostly similar to the conventional ConvNet architecture (Figure 10) but with the introduction of the residual block (Figure 12).

By using skip connections from preceding layers, the model is able to retain information that might get lost in a network with many more layers. This results in the ability to build very deep networks without causing them to overfit, due to the properties of nested functions [2].

4.7. ResNet based pavement segment classifiers

Using the orthoframe partitioning approach as described in Section 3.4 and pretrained ResNet models provided in the fastai library, it is possible to create a model to classify segments of the orthoframe based on the presence of defects.

In our previous work [10], the fastai library methods were used to design and train the models for classifying segments. In this thesis, the PyTorch library approach was prefered as it seemed more straight forward to define custom training loops and dataloaders to experiment with.

An overview of the ResNet18 architecture for the segment classifier can be seen in Figure 13. In addition, ResNet50 and ResNet101 models were trained to verify if adding layers results in a better performance.

The model takes a normalized 224×224-pixel RGB image as an input and extracts

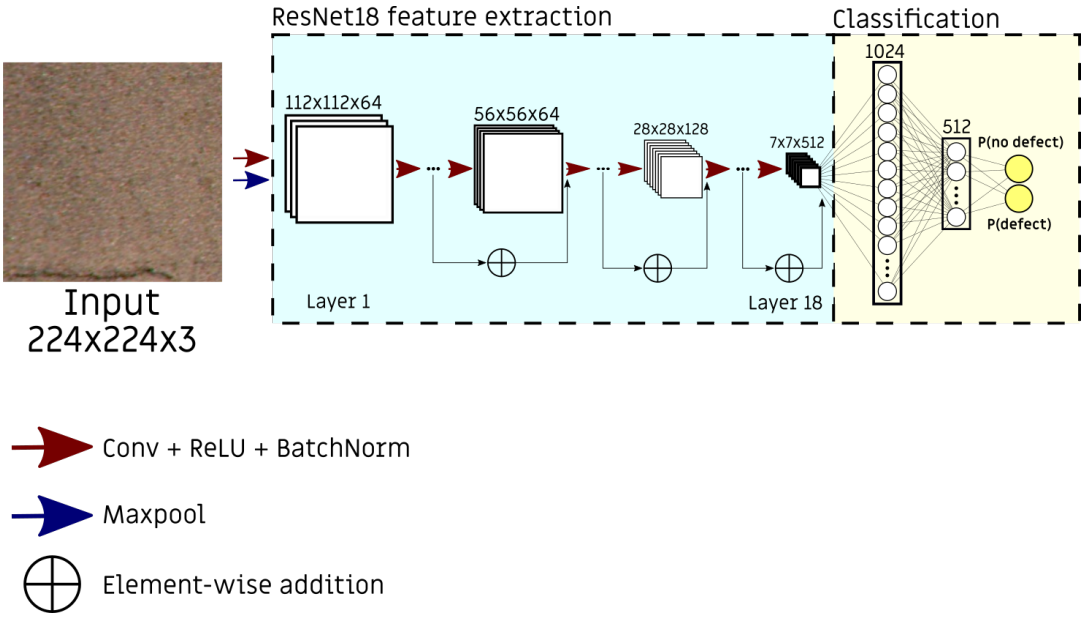


Figure 13. ResNet18 based pavement segment classifier.

features from it via the ResNet architecture. The normalization of the inputs is done with respect to the means and standard deviations of the ImageNet [36] dataset. After obtaining the final 512 7x7 feature maps, the average pooling and maximum pooling operation are applied which results in 1024 neurons. Batch normalization and dropout with $p = 0.5$ are also applied in the classification layers. For the final layer, the softmax activation is used to get class probabilities.

The number of parameters for different ResNet models used in this work can be seen in Table 3. Information about performances of the models can be seen in Table 2.

5. Two-stream ConvNet

This chapter describes the implementation of the proposed two-stream ConvNet approach for pavement segment classification.

5.1. Motivation

While the single ConvNet based classification for pavement cracking detection already had satisfactory results (Table 2), it did not make use of the rest of the orthoframe in its classification process. By judging the segment of the orthoframe in isolation, it can sometimes be difficult to assess whether the edge of the segment is partly a crack or some miscellaneous formation (Figure 14), resulting in a false positive.

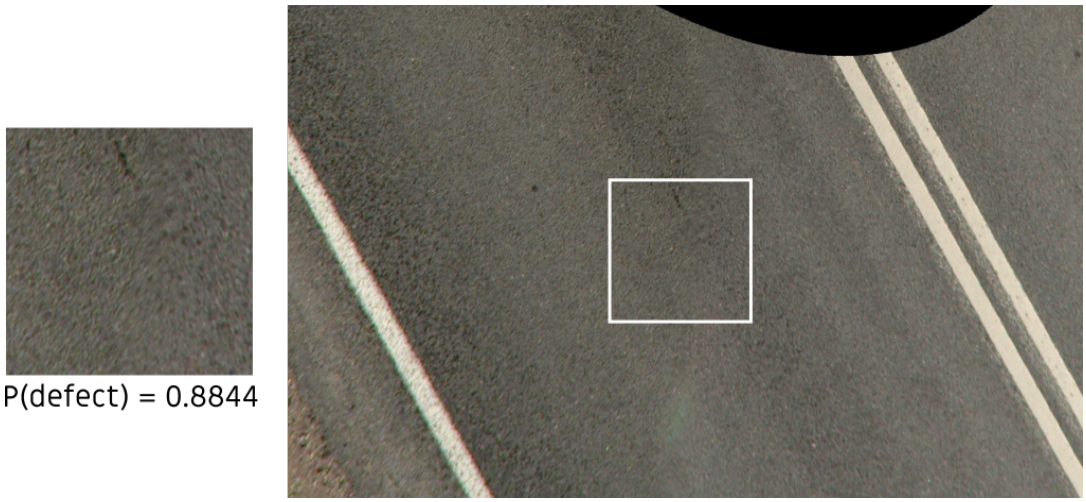


Figure 14. Case of the ResNet101 model predicting a false positive due to lack of context.

Similarly, false negatives may arise due to lack of context in the classification process, as the cracking will be less distinct in some parts of the photo (Figure 15).

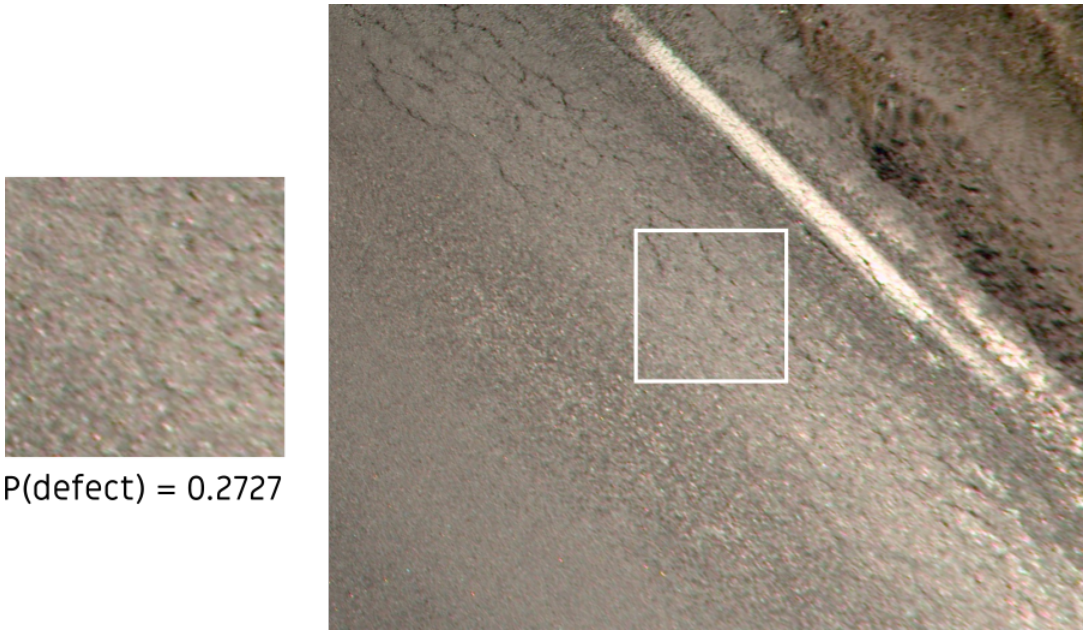


Figure 15. Case of the ResNet101 model predicting a false negative due to lack of context.

Additionally, stemming from the fact that defects may lie at the further end of the orthoframes, they might become very hard to distinguish due to the blurriness of the photo, in which case it would be helpful to see the surrounding road.

Finally, during manual data annotation, it was evident that annotating with the whole orthoframe as reference was a lot simpler, from a human perspective.

5.2. Implementation

Following the reasoning in the previous Section, it is clear that we are not using all of the useful information given to us by simply looking at one segment in isolation. Therefore, methods to incorporate contextual information for the ConvNet's decision making were considered.

Inspired by the research discussed in Section 2.1.2, it seemed like combining multiple ConvNet feature extractors could be a feasible approach to provide the model with contextual features. As ConvNets excel in finding relevant features and the fully connected neural networks excel in combining features in the most relevant way, the idea was to create a two-stream ConvNet to combine the content and the context features.

The proposed network architecture for the context-aware segment-based classifier can be seen in Figure 16.

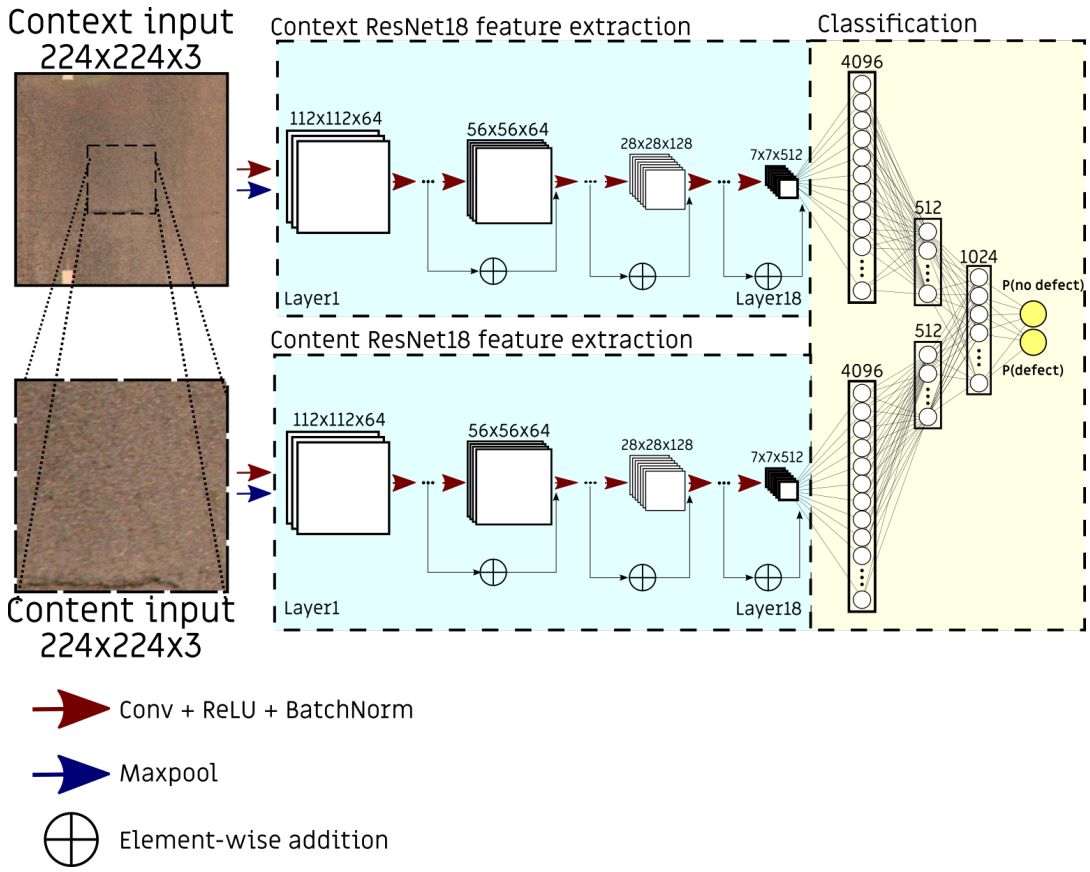


Figure 16. Proposed architecture of the ResNet18 based context-aware pavement segment classifier.

The network receives inputs in content-context pairs. The context part of the input depicts the area around the content input. In an effort to balance the amount of relevant context, the context input was chosen to have a diameter 3 times that of the content input, essentially providing a zoomed out view of the content. To address GPU memory limitations and have feature maps of the same size, the context input was rescaled to the same dimensions as the content input.

As with the single stream model, the task of classification is to answer whether the segment contains a crack or not. The label for a given pair is "defected" if over 5% of the pixels in the content segment are masked as defected by the manual annotation and "not defected" otherwise (as described in Section 3.4). Therefore, any defects that lie outside the center of the context model do not affect the label for the input pair.

Relevant features of the inputs are then extracted in two different streams and the features of the content and context streams are combined in a final fully connected layer, to produce the prediction of the model.

Various configurations for the fully connected layers were experimented with. The configuration presented in Figure 16 was found to give the best results as seen in Table 2. Additionally, the results for a slightly simpler classification architecture in which the final 1024 layer was omitted is presented in 2.

6. Experiments and evaluation

6.1. Evaluation metrics

In order to choose the most informative evaluation metrics, we must consider the distribution of the dataset. The accuracy of a classifier can be considered as the sum of correctly classified segments divided by all of the segments in the dataset. However, classification tasks in the case of severely imbalanced datasets (as in our case) would obtain an unfairly high accuracy from a classifier that simply votes for the majority class. Therefore, accuracy would not be a very good indicator of performance, as the average road orthoframe has vastly more defect-free segments than defected segments. Following this reasoning, let us consider the four possible outcomes of any prediction:

- True positive (TP) – the segment is correctly identified as defected.
- True negative (TN) – the segment is correctly identified as non-defected.
- False positive (FP) – the segment is incorrectly identified as a defect when it was labeled as a non-defect.
- False negative (FN) – the segment is incorrectly identified as a non-defect when it was labeled as a defect.

Thus, it is possible to calculate the precision (5) and recall (6) metrics. In words, the precision metric of a classifier is informative of the classifier's ability to avoid false positives and the recall metric tells us of the classifier's ability to avoid false negatives.

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{6}$$

These metrics are useful to evaluate whether our classifier has a bias towards one of the categories, so we could adjust the defect detection threshold P_{det}. In order for an input to be classified as defected the following must hold true:

$$P(\text{defect}) \geqslant P_{det}, \tag{7}$$

where $P(\text{defect})$ is the ConvNet's probabilistic output. The threshold P_{det} was chosen based on the validation set such that Precision≈Recall, which happened to be 0.6 for all models.

A metric that works well even across datasets with different class distributions is the MCC. It is considered to be the most informative metric in the case of a binary classification task with data imbalance [56]. MCC is able to calculated from the confusion matrix and it is defined to be as

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{8}$$

In deciding which classifier is superior, we shall refer to the MCC, as it best captures the classifier performance with a single coefficient.

6.2. Training phase evaluation

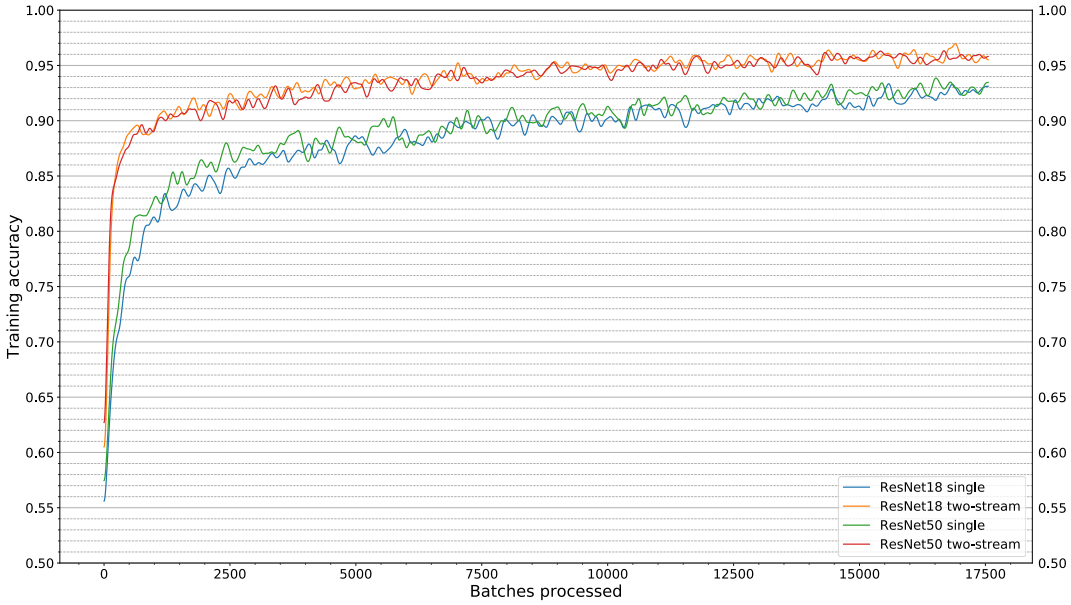
For the training data, a total of 56 178 samples were provided, 28 089 without cracks and 28 089 with cracks. It should be noted that only 32 140 of the samples/sample pairs were from a unique source segment, due to the oversampling strategy as described in Section 3.4.

The training figures 17 and 18 provided include training accuracy and validation loss. The reason validation accuracy was not considered a good metric is due to the class imbalance in the validation set.

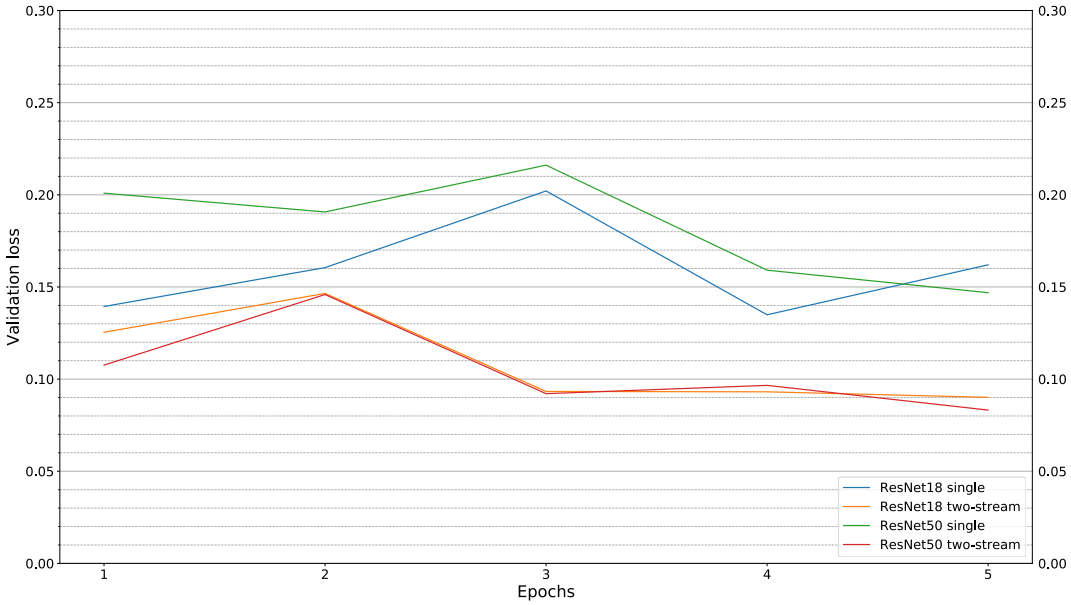
In most cases training accuracy is not very informative, as there are strategies that achieve good results with a lower training accuracy [10] or strategies that overfit

to training data that also achieve good results [57]. In our case, however, training accuracies of the two-stream and single stream approaches were monitored to compare learning abilities of the different approaches as they were subject to identical training schemes.

A training epoch in Figure 17 and Figure 18 corresponds to 3511 batches of 16 samples (or sample pairs for two-stream models).



(a) Training accuracies (higher is better).



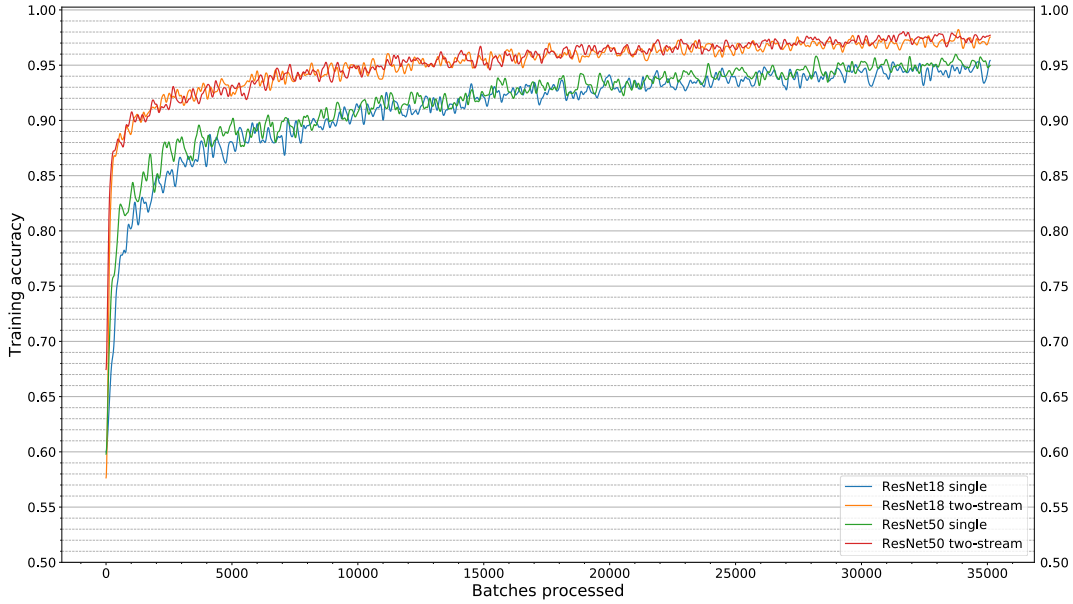
(b) Validation losses (lower is better).

Figure 17. Training accuracies (a) and validation losses (b) over 5 epochs for 4 different models.

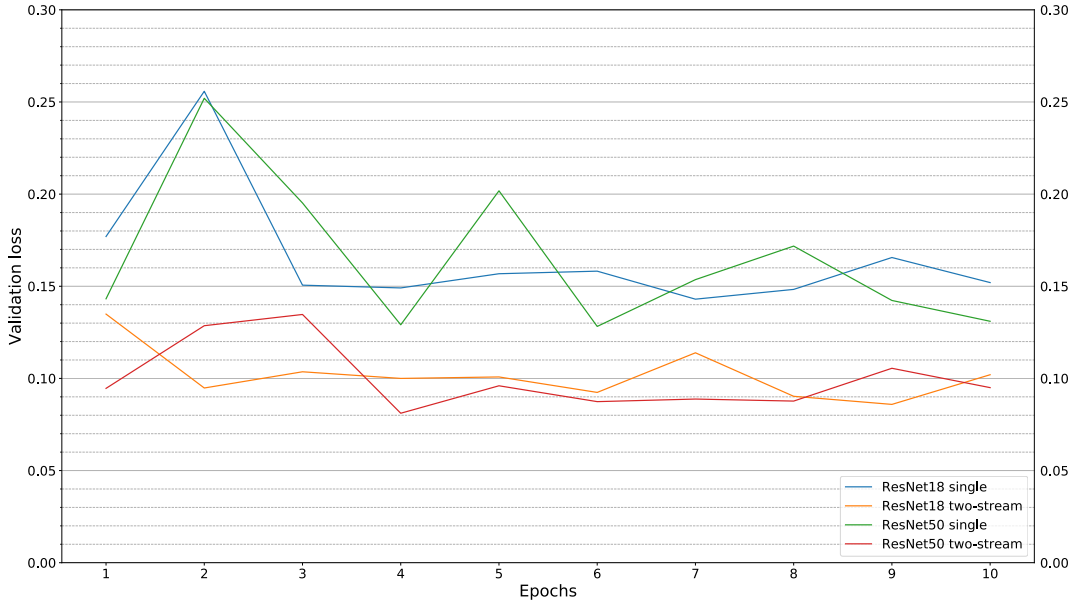
It is interesting to note that a training accuracy of over 90% was achieved in about 1500 processed batches for the two-stream models, while achieving the same training accuracy took about 7500 batches for the single stream models. This could be interpreted as the two-stream models learning more from the same amount of data. Additionally, the deeper single ResNet50 model seems to achieve slightly higher accuracies than the shallower single ResNet18 model. This phenomenon is not noticed for the two-stream models.

Another observation can be made with regard to the validation loss. As some of the models had their best validation losses in the earlier epochs, it seems that the models are able to learn well enough within the first few epochs. Additionally, it seems as if the validation loss across is volatile, which might be an indication of the validation set being too small or not diverse enough (90 orthoframes with 1192 generated samples/sample pairs in total). Therefore, statements about validation performance cannot be made with a high degree of certainty.

Something that is also noticeable is that the training accuracy of the solo models is still increasing after 5 epochs. In light of this observation and the volatility of the validation loss, it was decided to verify if the models were to benefit from further training, so the four models were trained again for 10 epochs with a linearly decreasing learning rate. The results of the extra training are displayed in Figure 18.



(a) Training accuracies (higher is better).



(b) Validation losses (lower is better).

Figure 18. Training accuracies (a) and validation losses (b) over 10 epochs for 4 different models.

While training accuracies kept rising for all models from epoch 5 to 10, the validation losses did not get significantly better or worse. Therefore it is probably the case that an extensive amount of training is not needed to achieve the best results for our task.

6.3. Obtained metrics and performance with different models

In total, 7 models were trained with the same random seed in order to make sure weight initialization does not affect the results. That said, PyTorch does not guarantee completely reproducible results even when using identical seeds, so comparison of models should be approached with some caution (Table 2), as attempts to reproduce the experiments gave slightly different results.

Table 2. Performance metrics of the models trained over 5 epochs with a fixed random seed.

Model	**Validation set (ideal conditions)**			**Test set (non-ideal conditions)**	
	MCC	**Precision**	**Recall**	**Precision**	**Recall**
Single ResNet18	0.85	0.87	0.88	0.69	0.62
Single ResNet50	0.85	0.88	0.88	0.69	0.62
Single ResNet101	0.86	0.88	0.89	0.70	0.62
Two-stream ResNet18s	**0.89**	**0.89**	**0.94**	**0.68**	**0.71**
Two-stream ResNet50s	0.88	0.93	0.88	0.71	0.67
Two-stream ResNet50s, simpler classification architecture	0.86	0.90	0.88	0.68	0.69
Two-stream ResNet101s, simpler classification architecture	0.89	0.90	0.92	0.61	0.64

*Best results from the previous attempts [10], using undersampling and 25% more data, were 0.87 precision and 0.90 recall for the validation set. Accuracies for the two-stream ResNet18s are 97.1% for validation set and 96.2% for test set.

The best results for the validation and test set were produced by the two-stream ResNet18s. Quite similar results were attained with deeper models and with slightly different classification architectures. It seems to be the case that adding layers to the two-stream feature extractors does not improve performance. However, for the

single-stream models, adding layers seems to result in slight improvements. This phenomenon was also noticed in slightly different conditions in our previous work [10].

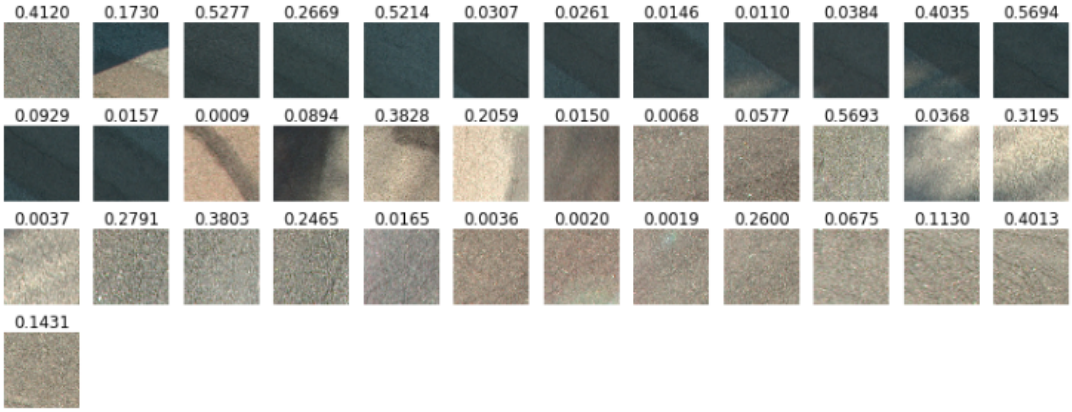
In addition to the performance over the validation and test sets, it is important to consider the inference speed of the model, as predicting over large datasets could potentially take a long time. As seen in Table 3, the better-performing two-stream ResNet18 operates faster and has less parameters than the single ResNet101, due to the reduced number of layers in the feature extraction stream, despite having two input streams.

Table 3. Size and inference speed of the models.

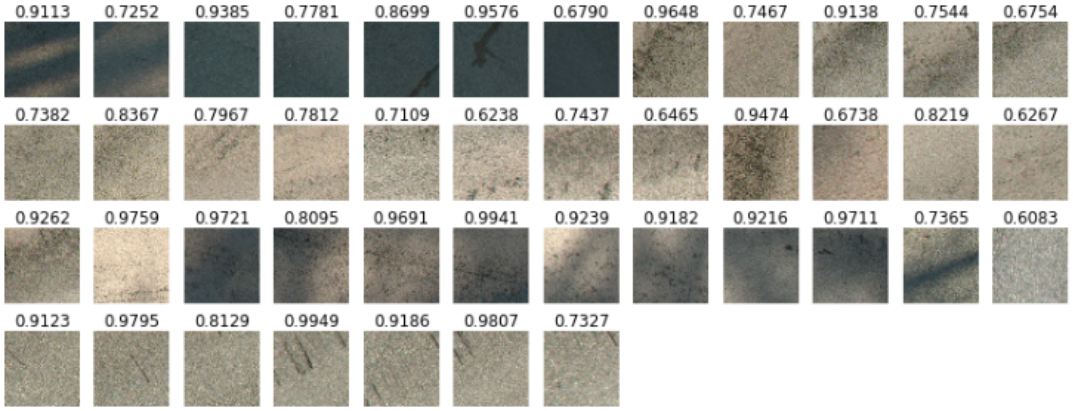
Model	**Parameters**	**Time to predict a segment (or segment pair)**	**Time to train**
Solo ResNet18	11,705,410	9.2ms	41m
Solo ResNet50	25,615,938	20.1ms	1h 22m
Solo ResNet101	44,608,066	37.5ms	2h 02m
Two-stream ResNet18s	24,067,462	17.9ms	2h 15m
Two-stream ResNet50s	51,888,518	41.2ms	3h 46m
Two-stream ResNet50s, simpler classification architecture	51,233,926	39.8ms	3h 32m
Two-stream ResNet101s, simpler classification architecture	89,218,182	76.6ms	6h 25m

6.4. Visual analysis of the results over the test set

In order to better understand the strengths and weaknesses of the best-performing ResNet18 two-stream model, all of the false positives and false negatives over the test set were plotted out as shown in Figure 19.



(a) False negatives.



(b) False positives.

Figure 19. All of the false negatives (a) and false positives (b) produced by the two-stream ResNet18s across the test set of 2011 sample pairs. The numbers above each segment indicate the model's $P(defect)$. Only the content inputs are displayed.

It can be seen that many false negatives are produced due to the cracks that are blurry or hard to distinguish. Many false negatives were also produced for segments with less illumination. For false positives, the two-stream model confused a dark looking patch on the road for a crack. Also, dirt or non-crack road degradation that produced high contrast areas were mistakenly classified as defects.

All in all, the results over the test set can be considered satisfactory, as many of the produced errors were of the segments that were also hard to annotate, indicating that the model struggles with the same data as humans do. However, there could be some room for improvement with regard to the model's robustness to different lighting conditions, as a big chunk of the errors were produced on a small section of the dataset with darker conditions.

It is important to point out that the task of annotating cracks manually is not an easy one, especially with no clear guidelines. In hindsight, by looking at the outputs of the models, many of the labels given for the samples from manual annotations were inspected to have been wrong. To prevent the creation of a possibly biased dataset towards the model, these segments were not re-labeled or removed but kept as such. It follows from this that even an ideal classifier over the test set cannot be expected to have a performance without false negatives or false positives and the results need to be taken with a grain of salt.

6.5. Case-by-case comparisons

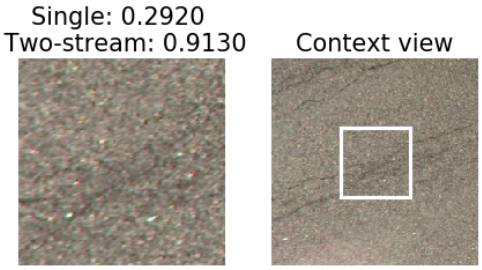
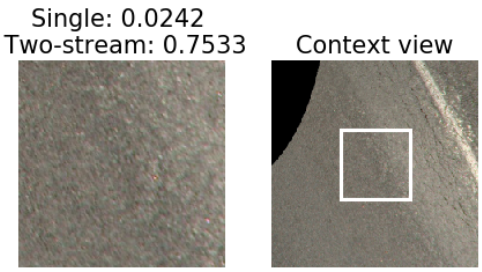
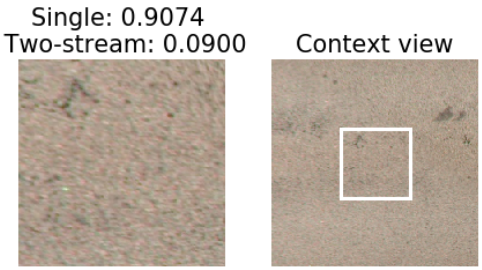
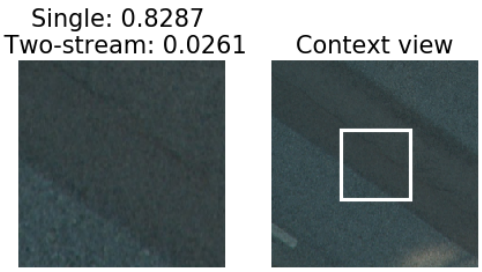
To have a more comparative analysis between the best performing single and two-stream models, total proportional amount of unique false positives and false negatives were calculated (Table 4). Interestingly, a large amount of the errors produced by both models were on different samples, indicating that each model struggled with different types of data.

Table 4. Comparison between the unique false positives (FP) and false negatives (FN) produced by the best performing single model ResNet101 and the best performing two-stream model ResNet18s.

Model	**Validation set**		**Test set**	
	Unique FPs	**Unique FNs**	**Unique FPs**	**Unique FNs**
Two-stream ResNet18s	41%	25%	44%	57%
Single ResNet101	43%	57%	82%	21%

To verify if the hypotheses posed in Section 5.1 held true, a closer inspection was given to some of the segments which produced conflicting outcomes between the single ResNet101 model and two-stream ResNet18 models. Four selected inputs with author's commentary can be seen in Table 5.

Table 5. Analysis of selected inputs where single and two-stream models produce different predictions.

Outputs	Analysis
	The two-stream model correctly identifies the fuzzy lines as defects, while the single stream model is uncertain, producing a false negative.
	The single model correctly identifies the segment as not defected but the two-stream model produces a false positive. This could be due to the close proximity with other cracks nearby in the context view, indicating that having the context stream might sometimes have adverse affects.
	The two-stream model correctly identifies the segment as not defected but the single stream model claims there to be a defect with high certainty (possibly due to the dark formations). The context view provided shows that there are no cracks nearby.
	The single model correctly identifies the crack in the shadow-ridden image, while the two-stream model is certain there is no defect. One interpretation could be that due to the dark setting of the road and relatively thin single crack, the context stream of the input guides it to classify as not defected.

7. Conclusions

It is evident that convolutional neural networks excel in the task of image classification and the work presented in this thesis makes it clear that this also applies to pavement distress detection. Using transfer learning, models trained in hours on a single modern GPU achieved near human-level performance for roads without adverse conditions such as the presence of shadows in the orthophoto. However, in situations where the input data had non-ideal conditions, the performance suffered. Accuracy wise, the best performances were 97.1% for the ideal conditions and 96.2% for the non-ideal and more realistic conditions. The more realistic conditions had fewer defects per orthoframe, so the MCC dropped more drastically than the accuracy, from 0.89 to 0.70.

The aim of this thesis was to build a classification model for pavement distress detection from orthophotos. To account for the large size of the orthoframe input which ConvNet models are not well-equipped to deal with, the approach to partition the orthoframe into segments was used. A benefit of this approach was the increased localization of the prediction outputs. However, partitioning of the orthoframe removed some of the potentially relevant input features for the model. Therefore, a model was proposed which read inputs for content stream and the context stream separately. Experiments across different datasets proved that the context-aware two-stream approach indeed slightly outperformed the single stream approach. It was expected that the context-aware two-stream approach would benefit mostly in more adverse conditions but it was found to be the case that the results also improved under ideal conditions, perhaps even more so than under adverse conditions. However, due to the shortage of diverse data, these results need to be further verified.

7.1. Future work

Many long and short term avenues are open for future work with the attained results, some of which are described in this Section.

Semantic segmentation has become a widely researched topic for ConvNets with promising results [58]. This means that the ConvNet classifier could output a pixel level prediction for a given input, vastly increasing the road crack localization of our

current 224×224-pixel classification approach. This approach has been tested in numerous pavement distress related studies [16, 18, 13]. However, the downside of this approach is a vastly more time-consuming annotation process.

With respect to the feasibility of the proposed two-stream ConvNet, it would be interesting to validate the results on different datasets. Unfortunately, none of the public datasets for pavement distress seemed to include a similar setting in which large-resolution orthophotos are given. That said, this approach could be verified for problems in different fields. For example, similar ConvNet architectures for high-resolution medical image analysis had been proposed and verified to be superior to the single stream sliding-window approach [24].

The datasets annotated in this work came from only about 10 different sessions in total. Therefore, it should be verified if adding more diverse data would benefit in training a more robust classifier. As manual annotation is an arduous and error-prone process, methods to minimize the effort put into annotation should be used. Seichter *et al.* [7] proposed incremental learning approaches to address the data diversity problem. Their proposed approach is to gather the data that would be most beneficial for the training data, in order to minimize human effort. This could be done by feeding unlabeled sections of the road to the model and returning those roads where the model has the highest uncertainty (in our case, this would mean $P(defect)$ is close to 0.5).

The main goal of this work was to experiment with modeling. Thus, implementing a software solution for practical purposes was not a priority. Previously, Tepljakov *et al.* developed a GUI for applying models trained in Keras [48], so the software implementation for models developed in this work could be using the previously developed GUI as a basis.

References

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[2] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*, 2019, http://www.d2l.ai.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[4] "EyeVi - Reach-U," https://www.reach-u.com/eyevi.html, 2019.

[5] L. Nowell, E. Hetzler, and T. Tanasse, "Change blindness in information visualization: a case study," in *Proc. INFOVIS 2001. IEEE Symp. Information Visualization*, Oct. 2001, pp. 15–22.

[6] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.

[7] D. Seichter, M. Eisenbach, R. Stricker, and H.-M. Gross, "How to improve deep learning based pavement distress detection while minimizing human effort," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 63–70.

[8] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Sep. 2016, pp. 3708–3712.

[9] K. Zhang, H. Cheng, and B. Zhang, "Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning," *Journal of Computing in Civil Engineering*, vol. 32, no. 2, p. 04018001, 2018.

[10] 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," *Applied Sciences*, vol. 9, no. 22, p. 4829, 11 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/22/4829

[11] H. D. Cheng and M. Miyojim, "Novel system for automatic pavement distress detection." *Journal of Computing in Civil Engineering*, vol. 12, no. 3, p. 145, 1998. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=830977&site=ehost-live&scope=site

[12] JaChing Chou, W. A. O'Neill, and H. D. Cheng, "Pavement distress classification using neural networks," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Oct 1994, pp. 397–401 vol.1.

[13] A. Cuhadar, K. Shalaby, and S. Tasdoken, "Automatic segmentation of pavement condition data using wavelet transform," in *IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No.02CH37373)*, vol. 2, May 2002, pp. 1009–1014 vol.2.

[14] K. Gopalakrishnan, "Deep learning in data-driven pavement image analysis and automated distress detection: A review," *Data*, vol. 3, no. 3, p. 28, 2018.

[15] K. Zhang, H.-D. Cheng, and S. Gai, "Efficient dense-dilation network for pavement cracks detection with large input image size," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 884–889.

[16] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H. Gross, "How to get pavement distress detection ready for deep learning? a systematic approach," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, May 2017, pp. 2039–2047.

[17] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, Dec. 2016.

[18] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2718–2729, Oct. 2016.

[19] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Crack tree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, pp. 227–238, 02 2012.

[20] F. Yang, L. Zhang, S. Yu, D. V. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," *ArXiv*, vol. abs/1901.06340, 2019.

[21] J. Liu, C. Gao, D. Meng, and W. Zuo, "Two-stream contextualized cnn for fine-grained image classification," in *AAAI*, 2016, pp. 4232–4233. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11772

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[23] J. Pang, C. Li, J. Shi, Z. Xu, and H. Feng, "$\mathcal{R}^2$ -CNN: Fast tiny object detection in large-scale remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5512–5524, Aug. 2019.

[24] N. Tomita, B. Abdollahi, J. Wei, B. Ren, A. Suriawinata, and S. Hassanpour, "Attention-based deep neural networks for detection of cancerous and precancerous esophagus tissue on histopathological slides."

[25] M. Shaban, R. Awan, M. M. Fraz, A. Azam, D. R. J. Snead, and N. M. Rajpoot, "Context-aware convolutional neural network for grading of colorectal cancer histology images," *CoRR*, vol. abs/1907.09478, 2019. [Online]. Available: http://arxiv.org/abs/1907.09478

[26] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, Jul. 1959.

[27] Y. LeCun, S. Chopra, R. Hadsell, F. J. Huang, and et al., "A tutorial on energy-based learning," in *Predicting Structured Data*. MIT Press, 2006.

[28] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966. [Online]. Available: https://science.sciencemag.org/content/153/3731/34

[29] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[30] S. Deutsch, "Conjectures on mammalian neuron networks for visual pattern recognition," *IEEE Transactions on Systems Science and Cybernetics*, vol. 2, no. 2, pp. 81–85, Dec. 1966.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[32] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, vol. abs/1003.0358, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1003.html#abs-1003-0358

[33] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Proc. Int. Joint Conf. Neural Networks*, Jul. 2011, pp. 1918–1921.

[34] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2013, pp. 411–418.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[37] G. Van Rossum and F. L. Drake Jr, *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[38] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik, *cluster: Cluster Analysis Basics and Extensions*, 2013.

[39] *MATLAB version 9.3.0.713579 (R2017b)*, The Mathworks, Inc., Natick, Massachusetts, 2017.

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning

on heterogeneous systems, 2015," *Software available from tensorflow. org*, vol. 1, no. 2, 2015.

[41] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[43] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[44] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[45] J. Howard *et al.*, "fastai," https://github.com/fastai/fastai, 2018.

[46] A. Tepljakov, "Annotation tool for datm project," https://github.com/extall/datm-annotation-tool, 2019.

[47] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[48] A. Tepljakov, A. Riid, R. Pihlak, K. Vassiljeva, and E. Petlenkov, "Deep learning for detection of pavement distress using nonideal photographic images," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2019, pp. 195–200.

[49] M. Buda, A. Maki, and M. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, 10 2017.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[52] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2682–2690.

[53] S. Ruder, "An overview of gradient descent optimization algorithms."

[54] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[55] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?"

[56] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric." *PloS one*, vol. 12, p. e0177678, 2017.

[57] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019.

[58] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

List of Publications

P1. 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," Applied Sciences, vol. 9, no. 22, p. 4829, 11 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/22/4829

Abstract

The subject matter of this research article is automatic detection of pavement distress on highway roads using computer vision algorithms. Specifically, deep learning convolutional neural network models are employed towards the implementation of the detector. Source data for training the detector come in the form of orthoframes acquired by a mobile mapping system. Compared to our previous work, the orthoframes are generally of better quality, but more importantly, in this work, we introduce a manual preprocessing step: sets of orthoframes are carefully selected for training and manually digitized to ensure adequate performance of the detector. Pretrained convolutional neural networks are then fine-tuned for the problem of pavement distress detection. Corresponding experimental results are provided and analyzed and indicate a successful implementation of the detector.

Author's contributions to the paper

The author of this thesis was responsible for proposing a convolutional neural network based model for the partitioned orthoframe segments. The author developed and implemented the transfer learning based models in fastai. Along with the coauthors of the paper, the author of this thesis was also responsible for manually annotating 400 of the orthoframes for defects. The author proposed the methods to test the models and did so by annotating an additional 55 orthoframes from different sessions than the training and validation data. The author was responsible for the experiments performed with the models and modified Python scripts previously developed by Dr. Aleksei Tepljakov to visualize the defect localization on an orthoframe. The author co-wrote a part of the paper relating to modeling methodology and co-wrote a significant part of the experimental results.