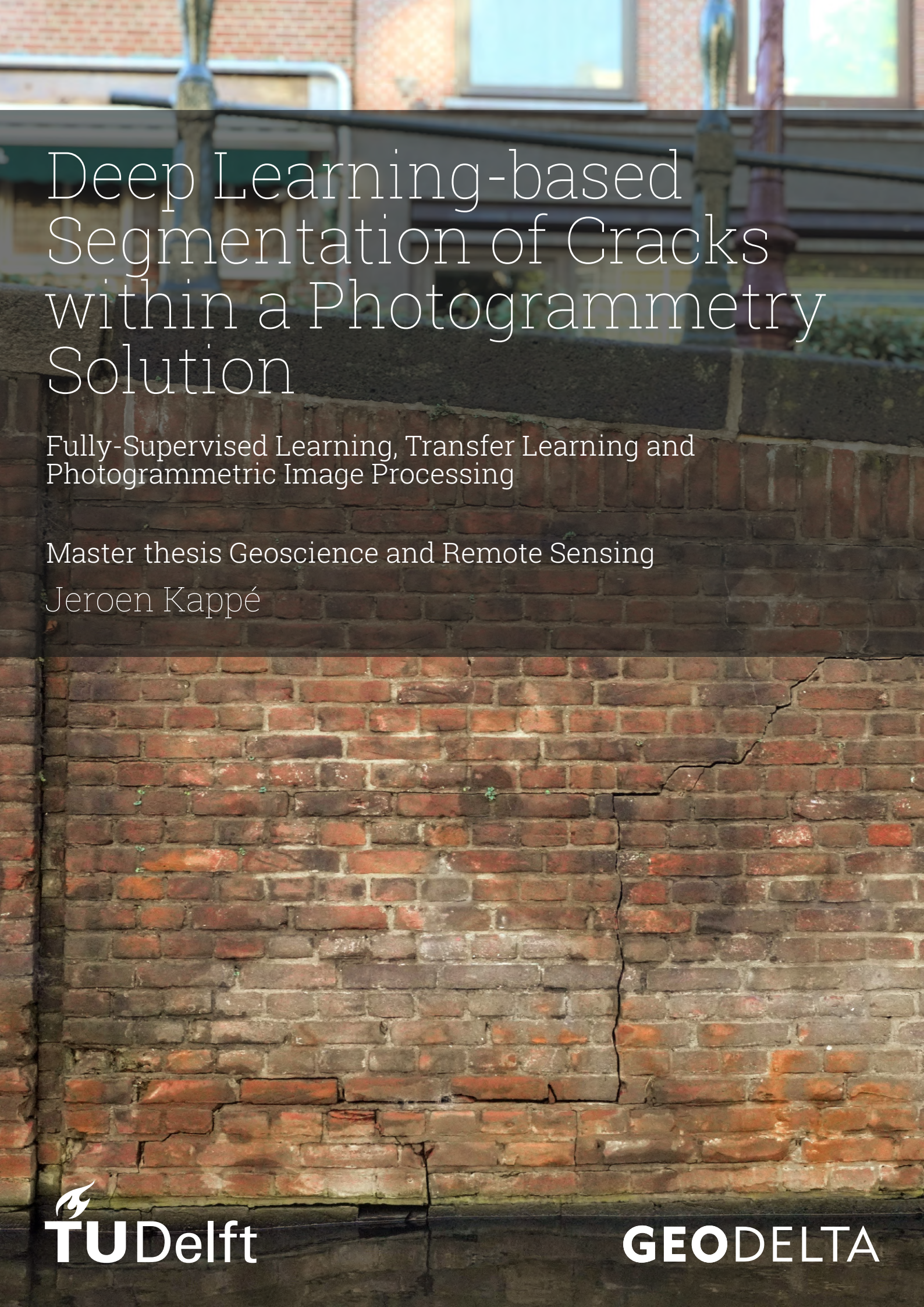


Deep Learning-based Segmentation of Cracks within a Photogrammetry Solution

Fully-Supervised Learning, Transfer Learning and Photogrammetric Image Processing

Master thesis Geoscience and Remote Sensing

Jeroen Kappé

TU Delft

GEO DELTA

Deep Learning-based Segmentation of Cracks within a Photogrammetry Solution

Fully-Supervised Learning, Transfer Learning and Photogrammetric Image Processing

by

Jeroen Kappé

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday April 5, 2024 at 15:00.

Student number: 4699378
Project Duration: September, 2023 - April, 2024
Thesis committee:

Dr.	R. C. Lindenbergh	TU Delft, chair
Dr.	M. A. Schleiss,	TU Delft
Ir.	P. A. Korswagen,	TU Delft
Ir.	M. P. Kodde,	Geodelta

TU Delft

Abstract

The city of Amsterdam faces the challenge of monitoring and assessing 200 kilometers of historic quay walls, of which much is deemed to be in poor condition. A key monitoring technique used is photogrammetry resulting in deformation testing. The fundamental data source forming the basis of this deformation analysis is a collection of overlapping images acquired of the masonry quay walls. Solely focusing on deformations overlooks a potential wealth of information which could be retrieved from this imagery, like the existence of cracks in the quay walls, a key sign of potential deformation of the structure. As manual visual inspection of this imagery is very time-consuming, this work proposes a methodology based on fully-supervised deep learning-based segmentation techniques with the goal of detecting and localizing cracks in the masonry quay walls. For this purpose, two neural networks are trained, one for the segmentation of quay walls in images, and one for the segmentation of cracks. The neural network architectures which are considered in this work are DeepLabV3+, FPN, MANet and LinkNet, together with different encoders and loss functions. For quay wall segmentation, we adopt transfer learning on a network trained on masonry walls and fine-tune it for quay walls specifically. Here, DeepLabV3+ with ResNeXt-50 was found to be most effective, achieving a F1-score of 96.3% on the test set. For crack segmentation, FPN with ResNeSt-50 performed best, resulting in a test set F1-score of 78.8%. The inference of the crack network is done with a multi-level scheme to detect cracks at different image scales and increase output confidence. The inherent photogrammetric properties of the imagery have proven to be vital for further post-processing steps, like aggregating overlapping predictions, resulting in more prediction confidence. Photogrammetry also enables converting pixel-wise predictions to crack length and crack width in the units of meters and millimeters respectively. The methodology additionally proposes photogrammetric image processing methods to transform neural network predictions to a 3D representation and a true-to-scale orthographic 2D image. Additionally a concise visual evaluation has been conducted to assess the prediction performance on an otherwise unlabelled dataset. This thesis presents an engineering effort for fully-supervised crack localization within the context of photogrammetric processed images, with generalization in mind for automatic assessment.

Acknowledgements

Working on this thesis has been a great journey and while reflecting back I am grateful to have delved into a topic which is so close to my interests. The last seven months have been an exciting experience where I learned a lot about deep learning, image processing and photogrammetry. It feels a bit surreal that the milestone of finishing this thesis consequently concludes my time as student at TU Delft.
A sincere thanks to Roderik, my daily supervisor, first of all for making this topic possible in collaboration with Geodelta. Besides, for the feedback and guidance throughout the period. After each of our bi-weekly meetings, I found myself energized to continue further. I also want to thank Martin from Geodelta and my thesis committee for making this topic possible as well. I highly valued the freedom I got to pursue research directions to my interests, and the enthusiasm with which you have received my work over the past months. The frequent discussions and short meet ups around the desk have been fun. Next, I want to thank Marc Schleiss and Paul Korswagen from my thesis committee for their involvement in assessing the project. Your feedback have been important during the shaping of this work. At last, I want to thank everyone at Geodelta for providing the inspiring working atmosphere. Most notably I want to thank Annemieke for the great help throughout the months, the value of your feedback has been immense.
Finally a word of thanks to my friends, girlfriend and family for many things. I truly value the good times, which come in many forms, but nonetheless have been much needed refreshments during the writing of this thesis. Thanks for being part of my life.

Jeroen Kappé
Delft, March 2024

Contents

1

Introduction

Recently, the city of Amsterdam has been assessing and monitoring potentially weak quay walls along the historic canals on a large scale [2]. Some of these quay walls date back to over 300 years ago and are an important part of the UNESCO-protected authentic image of the city. These historic quay walls are masonry walls founded on wooden piles and account for 200 kilometres out of the total of 600 kilometres of quay walls in Amsterdam. Due to aging and increasing traffic load in recent centuries, large sections of quay walls have started showing signs of damage and are prone to dangerous subsidence. At least 10 kilometres of quay wall is in such a bad state that there is a high risk of a collapse [27].

Although most subsidence is only within a small order of magnitude, it can potentially cause huge damage to adjacent houses and infrastructure. More dramatically, such deformation can even cause a sudden collapse, as has recently happened in September 2020 at the Grimburgwal quay. The main failure mechanism was found to be the horizontal bending of the piles, while an existing crack in the masonry wall reduced the redistribution of the forces in the quay and the stronger parts could no longer take over the load [37].

Any required renovation or installation of emergency measures are very costly especially taking into account the additional traffic restrictions required at busy and crowded sites. It is therefore important to only do so when strictly necessary, giving rise to the need for large-scale and precise monitoring. From recent years the municipality of Amsterdam has adopted the technique of photogrammetry to monitor potential deformations, by taking measurements every several months, of over 200 km of quay walls within the inner city centre. Photogrammetry is a technique to model 3D world points derived from 2D optical images by fundamentally solving a system of collinearity equations. This method of measuring yields a lot more measurement points compared to the traditional technique of tacheometry, while still allowing for sufficient precision. Another advantage is that photogrammetry is more cost and time efficient, as measuring is quicker and can be done with a relatively cheap digital camera. The fundamental data for photogrammetry is a collection of overlapping optical images, in this scope capturing segments of a quay wall.

In this photogrammetric approach, focusing exclusively on calculating deformation for evaluating the condition of quay walls overlooks a potential wealth of valuable information that could be retrieved from visual cues in the optical images captured at the sites. Predominately cracks in the masonry walls can be an early indicator of deformation potentially resulting in a future collapse, hence monitoring cracks is key to gain more qualitative and quantitative information about the quality of quay walls [14].

Manual inspection for detecting and localizing such cracks can be time-consuming and cumbersome, especially considering the many images during multiple epochs involved. Recent advancements in both computer vision and deep learning have been widely adopted in both industry and academia and achieved groundbreaking performance in a wide variety of tasks [49]. For this work we are interested in performing pixel-level semantic object segmentation [34] [39] of cracks in optical images. We furthermore explore how these results can be aggregated to a more semantically richer representation by means of photogrammetry, using the intrinsic overlap within the imagery.

This work adds novelty which can be expressed in two ways. First, starting from the existing photogrammetric measuring done for deformation analysis. We provide methodologies to extent this notion by researching a learned segmentation approach to classify and localize cracks. This increases the available information to assess the quality of quay walls in an automated manner, as can be derived from the images used for these photogrammetric calculations. In reverse reasoning, we demonstrate how the learned segmentations can be enhanced by leveraging the intrinsic property of overlap between images present in the photogrammetric imagery. Similarly, we will show how we can determine dimensions of cracks in real-world units (meters and millimeters), transcending the limitations of traditional pixel-based segmentation approaches. With these true-to-scale units, crack predictions can be analysed by their actual spatial dimensions of length and width. In these regards, this work will embody both the field of photogrammetry and deep learning to strengthen each other.

The main research question that this work aims to answer is as follows:

> **Main research question:** How can we localize and analyse cracks in masonry quay walls using fully-supervised deep learning methods together with photogrammetric image measurements.

To help formulate the objectives more concretely, the following three sub-research questions are formulated as well.

> **Sub-question 1:** What fully-supervised learning approach demonstrates to be effective for both quay wall and crack segmentation?

We demonstrate the training of the neural network through an experimental and iterative approach, focusing on the tuning of parameters and configurations. This is done for both tasks of quay wall segmentation and crack segmentation. For the learning of quay walls, a key aspect is the use of fine-tune learning. For crack segmentation, our efforts concentrate on optimizing the balance of predictive outputs.

> **Sub-question 2:** How can the neural networks, once trained, be integrated into an algorithmic workflow considering the photogrammetric context?

Upon developing the respective segmentation neural networks, we show that there is quite some ambiguity regarding the optimal way to deploy these models for inferring predictions. The inherent variance on both scale and translations of these networks are key motivators to come up with an algorithmic workflow detailing how these models are inferred in practice. Moreover, as we transition to an unlabelled dataset to apply this workflow on, we dedicate efforts towards quantifying the performance of this workflow as much as possible.

> **Sub-question 3:** How can the intrinsic properties of photogrammetry be leveraged to determine crack characteristics, such as length and width, in an algorithmic way?

We introduce methodologies that exploit the inherent photogrammetric properties present in the imagery to provide dimension estimations on both the length and the width of any detected crack. We will show how utilizing the overlap in between images enhances prediction confidence by aggregating overlapping predictions. Additionally, having access to data on the corresponding acquisition positions enables us to go from the units of image pixels to real-world units of meters or millimeters.

The remainder of this work is structured as follows. First we discuss the fundamental background knowledge in chapter 2, necessary to properly dive into the subsequent chapters. In chapter 3, the methodology is explained for the training of the neural networks and corresponding developed workflow. We will discuss how segmentation predictions are leveraged to extract meaningful crack characteristics by means of photogrammetry. Next, chapter 4 presents the acquired results from the experiments and the corresponding discussion is done in chapter 5. Lastly, we conclude the work in chapter 6 and provide suggestions for future work.

2

Background

Before we continue with the details in the proposed methodology, we first provide a primer of the core concepts used in this work. First we cover the basics of crack formations in masonry walls (section 2.1) from the fundamental perspective of construction mechanics and map that to our specific case study of quay walls in Amsterdam. Second, we discuss the principles of photogrammetry (section 2.2) and detail how this technique is used to map 2D optical images to a 3D real-world model. Lastly, we list out details about neural networks (section 2.3) specifically developed for segmentation tasks to explain how we utilize these techniques.

2.1. Cracks in masonry quay walls

Historic quay walls consist of a masonry wall placed on a timber floor founded with timber piles whereas more recently constructed quay walls make use of steel piles. One possible failure mechanism for quay walls using timber piles specifically is rotting, for which it is vulnerable when the piles are not fully submerged. In general, weakening of the piles or of the masonry wall result in a less uniform distribution of forces in the structure, meaning the total structure is more likely to deform under tension. Among others, this tension can be the variable load acting on top of the quay wall, for example due to heavy traffic on the adjacent street, and is therefore acting vertically on the quay wall. This failure mechanism and some other possible scenarios are depicted in figure 2.1, as retrieved from the recent work of Hemel [19] on quay walls in Amsterdam.

A direct consequence of such failure mechanisms resulting in change of force equilibrium could be the formation of cracks in the masonry structure, as described by De Vent [10]. This work additionally points out that in case of compression, cracks will likely occur parallel to the direction of the load as depicted in figure 2.2. However, the work is not concerned about masonry quay walls specifically, but still presents a plausible relation between load tension pressing upon the quay wall structures and the formation of vertical cracks. Although, while cracks can be a result of deformations acting on the structure, no absolute determination to the cause of masonry cracking can be made solely on the basis of visual observation [16].

Characteristics of cracks in masonry structures

Moreover, the work of De Vent [10] also discusses the material properties of the bricks and joints and although both can vary a lot, in most cases the joint material is more brittle and cracks more easily. In the context of masonry structures, ”joints” refer to the spaces between the bricks that are filled with mortar material. This results in a tendency for cracks to expand in a stairway pattern between the joint material and the masonry bricks.

Cracks will weaken the masonry structure even further making the quay wall more prone to more deformation and or cracking. In this way, crack formation and deformation are very much intertwined and cracks can both be a cause and consequence of deformation. This self-enforcing effect will only worsen the quality of the structure and hence, temporal growth along the length of cracks is a key sign of damage and deformation.

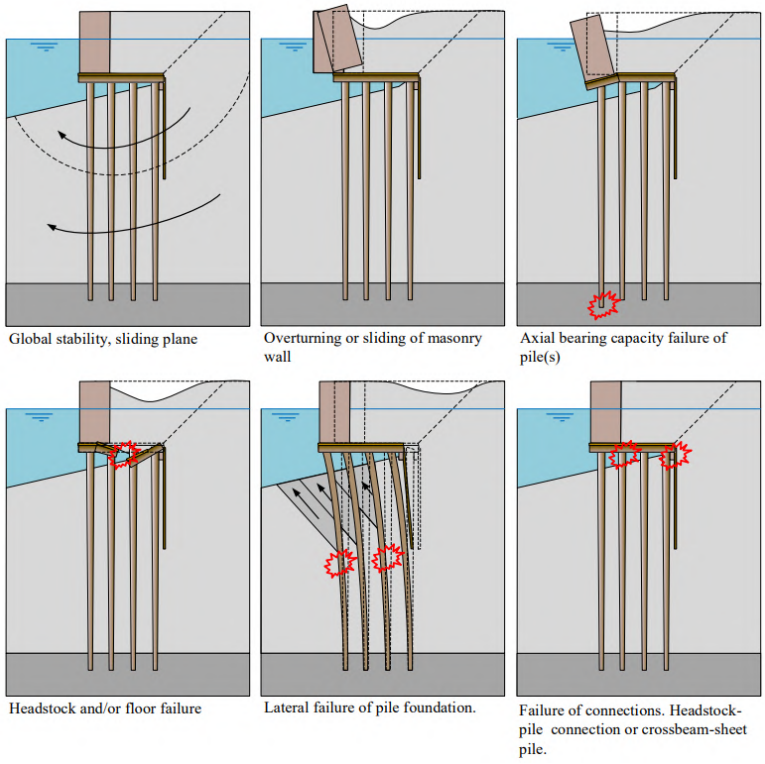


Figure 2.1: Several scenarios of failure mechanisms on historical quay walls (not all-encompassing). The scenario of lateral failure (middle down) is the most common one. Depiction is retrieved from the work of Hemel [19].

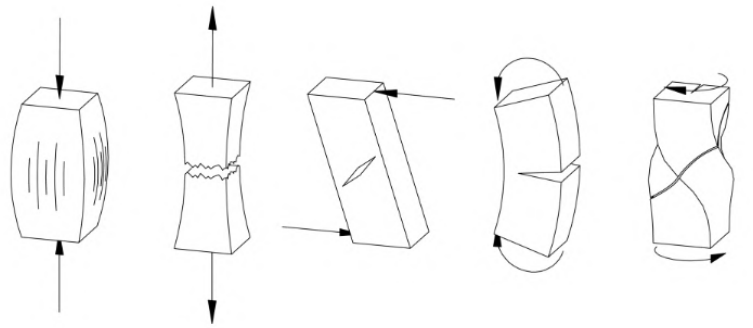


Figure 2.2: Several scenarios of crack development under different basic load situations, as retrieved from [10]. Notably, the scenario depicted left shows the situation of compression resulting in vertical cracks. This could be the case when severe load, by for example heavy traffic, is acting upon the quay wall.

To extent this, related literature commonly classifies crack severity by means of a six-point system, as introduced by Burland et al. [4] and later revised by Kastner et al. [24]. The system is based on damage level, expressed in terms of ease of repair and crack width, where damage level and crack width are positively correlated. This categorization is not specifically for masonry quay walls, but since to the best of our knowledge this is missing in literature, we adopt the same criteria. The six degrees of damage are defined as follows.

- Degree 0: Negligible. Hairline cracks less than about 0.1 mm.
- Degree 1: Very slight. Fine cracks with a width between about 0.1 mm and 1 mm.
- Degree 2: Slight. Typical crack widths between 1 mm and 5 mm.
- Degree 3: Moderate. Possibly some brickwork which needs to be replaced. Typical crack widths between 5 mm and 15 mm.

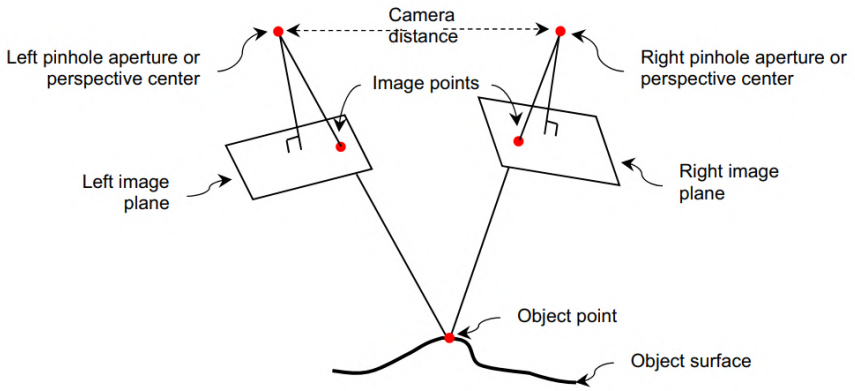


Figure 2.3: Basic principle of photogrammetry: illustration depicting the imaginary rays going through both the left and right image plane intersecting in the object point, retrieved from [3]. This depiction illustrates how a 3D object point can be estimated by intersecting the camera rays through corresponding pixel coordinates.

- Degree 4: Severe. Extensive repair work and replacing section of wall. Typical cracks width between 15 mm and 25 mm.
- Degree 5: Very severe. Crack widths exceeding 25 mm.

2.2. Photogrammetry

Photogrammetry is the science and engineering of establishing the geometric relation between 3D world-coordinates and the optical images capturing it. Photogrammetry can be utilized on varying spatial scales from aerial land surveying to close-range object reconstruction. Although photogrammetry is a complex field of study achieving success in many different fields, its power and flexibility can best be appreciated when considering a camera in its most basic form, a pinhole camera. This is illustrated in figure 2.3, showing how any point in the physical world captured by a camera is mapped to a 2D flattened representation on an imaginary camera plane. In the case where multiple cameras capture that same object, a mapping from 2D camera points to 3D object points can be made. In the next section we will briefly introduce this mapping in terms of its mathematical equations.

2.2.1. Mathematical fundamentals

There is a lot of interesting mathematics involved within the respective photogrammetric calculations. However, our work deals with imagery which has already undergone photogrammetric processing, resulting in derived positions for the corresponding cameras. These 3D world points describing the geometry scene are estimated by means of a nonlinear least-squares approach from the same imagery set, which minimizes the reprojection error. During this, the positions are furthermore related to terrain reference points yielding to true-to-scale coordinates. These derivations are outside the scope of this work, but it can be regarded as the respective backwards intersection estimation of the mapping we describe next. We solely write out how the mapping of 2D image points relate to 3D world points in its most simplified way and refer to existing literature for more extensive information [33].

The relation between 2D image points and the corresponding 3D world points is explained by once again referring to figure 2.3 and extending this depiction slightly by adding several mathematical variables. In this figure we can already see how an object point A, corresponding image point on the imaginary image plane a and perspective center of the camera PC all lie on the same straight line in three-dimensional real-world coordinates. Here the perspective center of the camera is the point from which the camera ”looks” through. In terms of camera coordinates, the image point and perspective center also lie on the same line. The relation between the lines in the different coordinate systems can be described by the so-called collinearity equations. These equations include parameters on the camera’s interior orientation parameters (focal length), as well as it’s exterior orientation parameters (position and rotation). To find all components of this equation we first indicate the perspective point and image point in the local camera coordinate system where c is the camera’s focal length. This local coordinate system is two dimensional so x_p, y_p and x_a, y_a denote the 2D coordinates of the perspective point and image point

respectively, as depicted in equation 2.1.

$$PC = \begin{bmatrix} x_p \\ y_p \\ c \end{bmatrix} \qquad A = \begin{bmatrix} x_a \\ y_a \\ 0 \end{bmatrix} \tag{2.1}$$

The real-world coordinate system counterparts of the perspective point and image point are (X_{PC}, Y_{PC}, Z_{PC}) and (X_A, Y_A, Z_A). The 3D vector from perspective center to image point is a scaled version of the one from perspective center to object point. This vector is defined as follows in both camera coordinate system ($\vec{v}$) and real-world system ($\vec{V}$).

$$\vec{v} = \begin{bmatrix} x_a - x_p \\ y_a - y_p \\ -c \end{bmatrix} \qquad \vec{V} = \begin{bmatrix} X_A - X_{PC} \\ Y_A - Y_{PC} \\ Z_A - Z_{PC} \end{bmatrix} \tag{2.2}$$

Transforming from the camera coordinate system to the real-world system requires the rotation of the camera described as a 3x3 rotation matrix$R$ with its three rotation angles ω, ϕ, κ and the ratio λ between the systems. Adding this transformation to the previously defined vectors gives the collinearity model with the mapping between the line in 2D and 3D coordinate systems.

$$\begin{bmatrix} x_a - x_p \\ y_a - y_p \\ -c \end{bmatrix} = \lambda \; R^T(\omega, \phi, \kappa) \; \begin{bmatrix} X_A - X_{PC} \\ Y_A - Y_{PC} \\ Z_A - Z_{PC} \end{bmatrix} \tag{2.3}$$

In this work we will utilize this mapping in three different ways. First, since the ratio λ and rotation R of the camera for every image is known (estimated by means of photogrammetry), we can utilize this to project any image point to a plane defined in the real-world coordinate system. This projection is done by intersecting the vector ($\vec{V}$) with the plane. The reverse calculation of finding an image point from a 3D world coordinate is also possible. Here the vector ($\vec{V}$) is constructed from a 3D object point to the perspective center and intersected with the image plane. This reverse approach is utilized in this work on an uniform grid of points on a plane in 3D distanced 1 mm by 1 mm to each other to finally yield a 2D image where each pixel has a real-world size of 1x1 mm.

Second, matching image points corresponding to the same real-world object point over several (overlapping) images yield the ability to estimate the position of this object point. For this we find the intersection of the different lines through the image point, again in a least-squares approach called forward intersection. During this, the more images used the more redundancy and therefore more precision. The notion of finding matching image points within overlapping images is a common computer vision task and usually referred to as feature (point) detection and matching, as described in the next section.

Third, we can additionally make use of the matched image points between overlapping images to perform the transformation process of image rectification. In this process the images are transformed such that they lie on the same image plane and corresponding pixels between the images are overlaid onto each other. This is done by transforming overlapping images with an affine homography mapping. This affine mapping is estimated by means of a nonlinear least-squares way on the matched features, minimizing the projection error. This mapping can be used in two ways. First, registering images onto each other allows for stitching subsequent images together to create a panorama photo. Likewise, overlapping images onto each other allows for aggregating overlapping pixels within the same area of interest. In this work the first process is used for visualization purposes where a larger section of a quay wall captured by following images can be computed, while the second approach is used to combine the predictions over multiple overlapping prediction images.

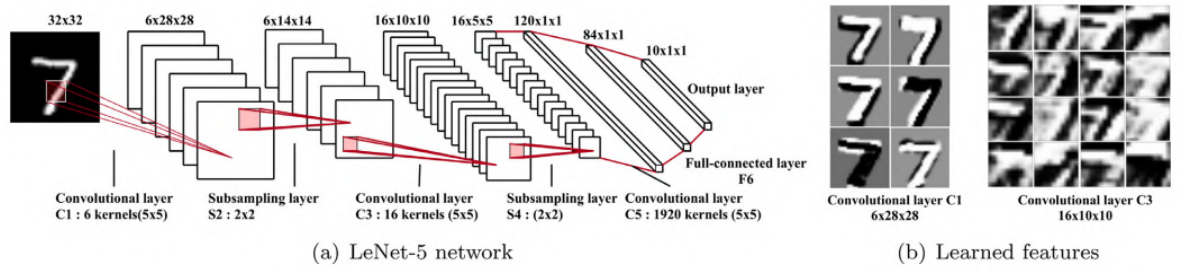


Figure 2.4: (a) Architecture overview of the first convolutional neural network LeNet as originally used to classify hand-written digits. (b) Visualization of the learned kernels of the first and third layers in the network. Illustration retrieved from [17].

2.2.2. Image feature detection and matching

Above we have explained how image points depicting the same real-world object point can be combined and leveraged to provide an estimate for this real-world position. This approach therefore deals with finding related image points on pixel-level. In literature this is often denoted by feature detection and feature matching between correlated images.

Finding and matching such image points on a large scale is only feasible in an algorithmic way and has been a widely studied topic in the field of image processing. While details of such algorithms are outside the scope of this work, it is still interesting to mention two of such approaches. The well-known algorithm SIFT, introduced by Lowe [36], is the traditional feature detector that exhibits scale, rotation, and translation invariance. The found image points are often matched between images based on the euclidean similarity of their descriptions.

More recently, deep learning based methods have exceeded the performance of such traditional methods. The self-supervised feature extractor SuperPoint [11] is shown to outperform SIFT in combination with SuperGlue [43], which is a learned feature matching graph neural network. For our work, the working of SIFT has found to work sufficiently well and therefore we focus on only utilizing this particular technique.

2.3. Image segmentation using deep learning

Image segmentation is a fundamental topic within computer vision with a substantial amount of algorithms been developed in literature. The rise of deep learning in recent years have surpassed traditional hand-crafted algorithms on popular benchmarks - resulting in a paradigm shift in the field. Supervised deep learning approaches are able to learn representations of data with multiple levels of abstractions in a highly nonlinear and complex way [29]. Such neural networks consist of multiple processing layers which iteratively learn from labelled training data by means of backpropagation. In the field of image understanding, deep learning is often combined with convolutional layers which can learn spatial features of images on different levels with convolutional kernels. For more fundamental details of the field of deep learning, we refer to the book ”Deep Learning” by Goodfellow, Bengio and Courville [15].

Deep learning based image segmentation can be formulated as the problem of classifying pixels with semantic labels (semantic segmentation), or partitioning of individual objects (instance segmentation), or both (panoptic segmentation) [38]. In the scope of this work distinguishing between multiple objects is not necessary so semantic segmentation suffices.

2.3.1. Convolutional neural networks

Convolutional neural networks (CNN) have become the standard approach for most image understanding tasks, ranging from image classification to object detection, after being introduced by LeCun [28]. The corresponding network, as depicted in figure 2.4, conveniently serves as a starting example to illustrate the core principles of CNNs, which are still used in later architectures.

A convolution is a mathematical operation of applying a filter on images to perform operations like blurring, sharpening or edge detection. CNNs are a specialized kind of neural network which learn sets of convolutions in multiple layers, still by means of backpropagation. The learned convolutions are

applied on (intermediate layer) input with a non-linear activation function and are shared among pixels as depicted in figure 2.4. Another key component of CNNs are pooling layers, where the (intermediate) convolved image input is subsampled, resulting in the effect of learning at different scales leading to more invariance to small local translations.

In contrast to the small network of LeNet, depicted in 2.4, with just five layers, the success of subsequent models have partly been to due to the deepening of similar networks. With increasing depth, the network can better approximate the target function with increased nonlinearity and get better feature representations. However, it also increases the complexity of the network, which makes the network suffer from vanishing gradients and makes it sensitive to overfitting on training data. An effective solution is the use of skip connections as proposed by Kaiming in the so-called Residual Network (ResNet) [18]. These networks are designed such that a layer will be skipped by means of regularization when its addition makes the performance of the network worse.

2.3.2. Encoder-decoder networks

Most of the segmentation models use some kind of encoder-decoder model architecture. As the name implies, such networks consist of two parts, first an encoder part followed by a decoder part. The encoder propagates the input image through successive convolutional, activation and pooling layers to a lower resolution, semantically richer, feature representation. The decoder then converts this feature representation and maps it, again in multiple levels, to a segmentation mask with the same spatial dimensions as the input image. Encoders are hence the part where the network learns to reason about what are important features and the decoder is responsible to map those features to a pixel-wise mask. Most decoder architectures can work with a variety of encoders and its efficiency will depend on the specific segmentation task. The LinkNet [5] architecture follows a straightforward encoder-decoder approach and is mainly praised for its relatively small number of parameters while still yielding good performance.

Two common encoder architectures which are in style of the previously discussed ResNet approach will be examined in this work. First, ResNeXt [50] which uses the same approach as ResNet, but unlike ResNet additionally introduces ”groups” containing parallel sets of convolutions. This usage of groups allows ResNeXt to comparatively learn more and better feature representations due to the widening of the network by means of parallel groups. Similarly, ResNeSt [52] also introduces groups but uses so-called split-attention mechanisms within each group. The details of the split-attention operation is outside the scope of this work, but we highlight the important aspect of retrieving a global context vector for each group. As the name indicated, this vector helps the network understand features within a larger (global) context of the image and is deemed to improve feature representation learning.

Furthermore, an encoder called MobileNetV2 [42] out of the family of MobileNet encoders [20] is considered in this work. MobileNets are lightweight networks specifically designed for mobile and embedded applications. The network architecture is comparatively a lot simpler, also using residual blocks, and for this work it is partially interesting to see the performance trade-offs for this much smaller network.

These encoders come in varying depths which is commonly added as suffix to the encoder name, ResNeSt-26 and ResNeSt-50 represent a 26 layered and 50 layered ResNeSt encoder respectively. Deeper encoders are generally better at learning more complex patterns but might be more prone to overfit on training data.

2.3.3. Multiscale pyramid networks

Multiscale analysis, a well established idea in image processing, has also been deployed in various neural network architectures. Making predictions at different resolutions of the input generally makes the network more robust and are therefore often integrated to at least some extend in many different approaches.

Feature pyramid network (FPN) [31] is one of such networks and conceptually consists of two parts, a bottom-up part and a top-down part as seen in figure 2.5. The bottom-up pathway computes features at different scales, whereas the top-down part upsamples to higher resolution features. Lower resolution scales lose the information of precise locations but are semantically stronger. For each spatial scale the segmentation prediction is made and aggregated to a final output, such that the semantically rich

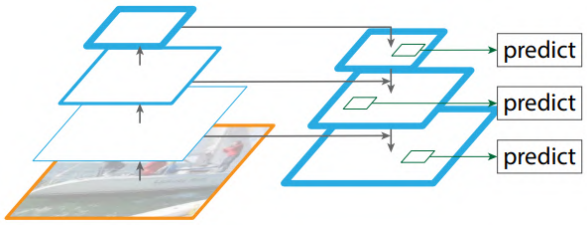
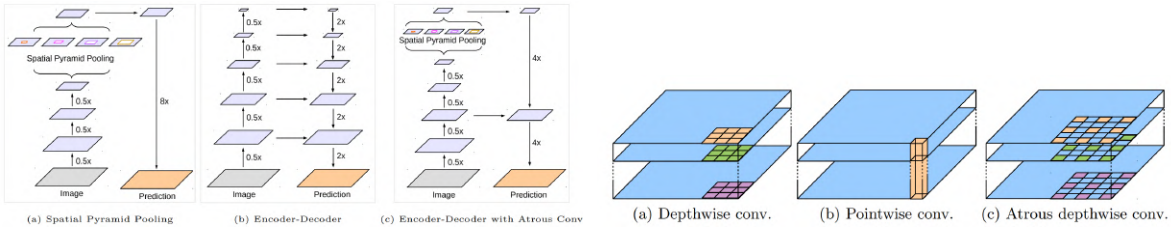


Figure 2.5: A depiction of the architecture overview of Feature Pyramid Network (FPN), as retrieved from the original paper [31]. It shows both the bottom-up and top-down pathways, where for each spatial scale a prediction is made and combined.



(a) Overview of DeepLabV3+ **(b)** Atrous convolutions

Figure 2.6: Two key depictions on the network design of DeepLabV3+, as retrieved from the original paper [6]. The encoder-decoder pathway is shown in (a) where the atrous spatial pyramid pooling is used during the encoding and notably the decoder is rather simple by design. A visualization on the workings of atrous convolutions is depicted in (b) on the rightmost sub-image, where the kernel is spaced with a stride of 1.

representation and precise locations are combined.

The architecture of DeepLabV3+ [6] also captures multi-scale information, with an encoder-decoder pathway similar to that seen in FPN. In contrast, it outputs predictions only on the original level of resolution and furthermore applies a slight variant of convolutions called atrous convolutions. These atrous convolutions are similar to standard convolutions, but instead the kernel has some spacing between its weights, yielding the ability to vary the spatial size of the total kernel. A depiction of atrous convolutions and on its application in the overall encoder-decoder design of DeepLabV3+ is shown in figure 2.6.

2.3.4. Attention-based networks

Next we discuss a different family of networks which commonly incorporate the dot-product attention mechanism, as originating from the field of natural language processing [48]. The details of the dot-product attention mechanism are outside the scope of this work, and instead we only highlight its key capability of capturing long-range dependencies in the data. In the context of natural language processing this ability to reason on a global scale improves understanding of sentences by ”remembering” important words early in the sentence, such as negation words. In the context of language, in this way the intuition of the attention mechanism is the ability to focus on certain segments of a sentence. In the context of images, words are replaced for sub-patches of the image and the attention mechanism emphasizes important patches and de-emphasizes less important ones. This ability to reason on a global context is rather different to how convolutions work, which inherently is a local operation proportional to the used kernel size.

Multi-attention network (MANet) [13] is an model architecture utilizing both a multi-scale approach and such a dot-product attention mechanism to be able to understand local details in the global context of the image. Comparatively to the local operation of convolution, the attention mechanism is more inline with the human visual perception. Furthermore, the authors point out that work has focused on decreasing the complexity of MANet compared to other attention-based networks. Notoriously, the attention mechanism is a computationally intensive operation so compared to the other solely convolutional networks discussed before the complexity is a lot higher.

2.3.5. Loss function and metrics

In order to evaluate and iteratively learn by means of back-propagation, a loss function formulating the error between the outcome of the model and the ground truth labels needs to be chosen. For relatively simple tasks such as classification (cross-entropy loss) or regression (mean-squared error) this is evident, while for segmentation tasks the loss function should be based on the degree of pixel overlap between prediction and label. Additionally, such metrics help to evaluate and compare the model performance.

A widely used loss function for segmentation tasks is the Sørensen–Dice coefficient, rooted in the field of statistics, and commonly named dice coefficient in the field of deep learning. The formula is calculated by dividing two times the area of intersection between predicted and ground-truth pixels by the total number of pixels of both areas, as seen in equation 2.4. Here A denotes the set of segmented pixels and B is the set of ground-truth pixels.

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|} \tag{2.4}$$

The more predicted and ground-truth pixels match, the more this coefficient will approach $Dice = 1$. No overlap results in $Dice = 0$. For the specific task of crack segmentation, the pixel occurrence of cracks (positive) compared to background (negative) pixels is very imbalanced, with the background taking up much more of the images. This will be quantified and visualized next in chapter 3. Because of this inherent class imbalance in the dataset, the learning model can obtain a very high Dice score by always outputting a completely negative image, as the ground truth background then overlaps with most of the image. A way to mitigate this class imbalance is to weight the loss per class differently depending on its occurrence. A slight variation of the dice coefficient named the generalized dice loss (GDice) [47] weights each class contribution inversely to its volume.

In a similar manner the (binary) weighted cross-entropy (WCE) loss is the weighted variation of the cross-entropy loss as shown in equation 2.5. Here p denotes the probability with which the pixel was segmented as positive (crack) or $(p-1)$ for negative (background) and w_p is the weight ratio between the two options. Furthermore, y is the binary ground-truth label, with 1 for positive and 0 for negative pixels. Averaging this loss for all pixels results in a number that again tends to 1 if the prediction matches the ground truth well and 0 if not.

$$WCE = -(y\ log(p)\ w_p + (1-y)\ log(1-p)) \tag{2.5}$$

Lastly we shine light on the focal loss function as proposed by Lin et al. [32]. It is a slight variation of the weighted cross-entropy loss, and similarly uses a class weight parameter, in this case named 'alpha'. Besides, focal loss introduces a new term named 'gamma', which is accountable for putting more emphases on easily misclassified pixels, compared to easy ones. The factor down-weights pixels which are predicted with high confidence (p), and likewise puts more emphasis on less confident predictions.

Segmentation evaluation metrics

In order to evaluate segmentation networks after they have been trained we use the confusion matrix items to compute scores for precision, accuracy and recall as depicted in 2.6. These items denote the difference between predicted output and ground-truth value, listed as: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). The values can be computed for a single pixel value and be further aggregated over the whole image or several images. While accuracy denotes the percentage of pixels correctly segmented, precision only looks at the pixels segmented as crack. Finally, recall indicates the percentage of the ground-truth crack that was correctly segmented as such.

$$Precision = \frac{TP}{TP+FP} \qquad Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad Recall = \frac{TP}{TP+FN} \tag{2.6}$$

Another metric used is the F1-score, which is the harmonic mean of precision and recall. From its formula, as listed below in equation 2.7, it becomes clear that this metric is equivalent to the dice loss

function. Both high precision and recall is desired but with limited learning abilities the two metrics are balancing each other out. Meaning that increasing precision often comes at the cost of decreasing recall and visa versa. In this way, optimizing the two metrics is a trade-off between prioritizing false negative errors versus false positive errors. Similarly the metric intersection over union (IoU) is a common segmentation evaluation metric. Again, the term A denotes the set of segmented pixels and B is the set of ground-truth pixels.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2\,TP + FP + FN} \qquad IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.7}$$

2.3.6. Pattern recognition and transfer learning

The universal approximation theorem discusses the capability of fully-connected neural networks to approximate any (continuous) function, given enough width of neurons in the hidden layer. There exists multiple proofs for different variations of neural networks. The work of G. Cybenko proves this universality for feed-forward fully-connected networks with at least one hidden layer using the Sigmoid activation function [7]. The universality property has also been proved for convolutional neural networks [53], proving the ability to approximate any function given enough depth in the network.

Although these proofs express the generalization power of neural networks, they do not tell how to design or train any network to optimize this ability. Nonetheless, we can in general expect more learning capabilities of larger CNN architectures. Besides network size, the effectiveness of learning also depend on the training hyperparameters such as the optimization algorithm or the loss function. On a more practical note, it is additionally constrained by computational power and training time (number of epochs).

This complexity of data predominately comes in two ways for our task of crack segmentation. First, the degree of diversity within the images, such as variations in lightning, scale, rotation or background can significantly increase the learning complexity. Furthermore, noise in either the images or corresponding labels can complicate learning if it adds sufficient inconsistencies across the dataset.

The limitations of learning capabilities has an important link with a common training technique within fully-supervised learning, called transfer learning [54]. The term transfer learning is a rather broad concept, and for this work we deal with fine-tune learning specially. Fine-tune learning deals with multiple datasets within the same or similar domain interest, where pre-training is done on one dataset and additional fine-tune training on another. The final dataset is the dataset of interest and the first dataset helps to already have weights in the network which are relevant.

Supervised learning with gradient descent

Fully-supervised learning of a neural network has the aim of minimizing a loss function over a certain data distribution. The corresponding loss landscape with all the parameters (weights and biases) of the network is highly multi-dimensional and optimizing accordingly is a key component of any deep learning task. This optimization can be seen as changing the parameters in the network with a small amount in the direction which reduces the loss. For every optimization method, the magnitude of this step can be changed with the 'learning rate' parameter. This mathematical process is called gradient descent and it is the fundamental method of learning during backpropagation. One of the oldest and most basic gradient descent optimization technique is called stochastic gradient descent (SGD). It uses a learning rate as described above with additionally allowing the learning rate to have some 'momentum'. The more sophisticated optimization method Adam [25] is more adaptive regarding learning rate during the training of a neural network.

During the training phase, it is a common procedure to split the dataset into three distinct subsets: a training, validation and test set. The training subset is used to train the network. The validation subset is used to evaluate the performance of the network during the training phase on the otherwise unseen dataset. A common pitfall to look out for is the effect of overfitting. Overfitting occurs when the network learns the training subset too well, affecting its performance on unseen data such as the validation or test set. Evaluating the performance on the validation set during the training phase helps to detect overfitting. One technique to prevent overfitting is called L2-regularization. This regularization adds an additional term to the loss function, the sum of squares of all the network's weights. As an effect,

large weights are penalized and smaller weights are preferred, leading to simpler networks. To note, the name 'L2' refers to the square of the weights, comparatively a similar method of 'L1'-regularization uses the magnitude of all the weights.

2.3.7. Learned segmentation of cracks in masonry structures

Fully-supervised segmentation techniques of cracks in images have been a widely studied topic in literature of structural engineering [9, 35, 8]. Such studies are likewise motivated to do so for automatic damage assessment using CNNs, often using the same or similar encoder-decoder CNN architectures as we will use for segmentation learning. These related works serve as an important inspiration during our development of the neural networks such as for choice of model architectures and training configuration. Although, direct comparison of the model performance is not possible due to the use of different datasets.

Both Loverdos et al. [35] and Dang et al. [9] stress the importance of geometric measurement of cracks (width and length), which is a critical measure of crack evaluation. Interestingly, the work of Dang et al. estimates this by relating image pixels to segmented bricks to result in true-size measurements in millimeters. For this task specifically, an additional brick segmentation network has been deployed. We achieve more accurate results with a photogrammetric mapping utilizing the photogrammetry measurements in the images. An additional benefit of working in a photogrammetric context is the possibility to combine segmentation results in overlapping images by means of applying the corresponding affine projection.

2.3.8. Semantic segmentation of point clouds

As mentioned above, image segmentation applied on overlapping images in the context of photogrammetry allows for further processing steps, such as geometric measuring, which would not be possible without this inherent overlap. Additionally, one of the deliveries of our work will be a photogrammetric reconstruction of cracks in 3D as well as a true-to-scale orthographic projection of the images.

These photogrammetry processing steps are part of an existing photogrammetric pipeline which derives point clouds from the imagery. The availability of such point clouds yields the trade-off whether or not it is desirable to segment cracks in the point cloud directly, instead of segmenting in the optical images and reconstructing to a 3D representation separately, as also discussed in the work by Stathopoulou and Remondino [46]. Learned segmentation on point clouds using neural networks is certainly a well-researched topic [51]. Point or graph convolutional neural networks are able to input point clouds directly into the network and can therefore be used for point cloud segmentation. Still, for our goal of segmenting cracks, doing so in 2D images is desired over 3D point clouds for two key reasons. First, the point clouds at our disposal are unlabelled whereas we have the availability of a large dataset of pixel-level labelled images. More fundamentally, another reason why this approach would not work is that cracks are not well represented in the point clouds with little to no uniquely identifiable features compared to non-crack points.

3

Methodology

In this chapter, the methodology regarding the development of the relevant segmentation neural networks together with their application within the encapsulating segmentation workflow is presented. The development of the (quay) wall segmentation networks in an iterative manner (section 3.1) discusses our proposed training strategies, from data preparation to fine-tuning configuration. In similar fashion, the development of the crack segmentation neural networks (section 3.2) is discussed, which comparatively proves to be a much more thoughtful task. Here the training design choices such as balancing predictive output and automated data pruning are listed. Both come together in the designed segmentation workflow (section 3.3). Subsequent post-processing exploits the strong information yielded from the inherent photogrammetric properties of the quay wall data set for the derivation of additional scaled crack characteristics: crack length and crack width (3.4). Lastly, implementation-specific details about the used software and hardware are listed (section 3.5).

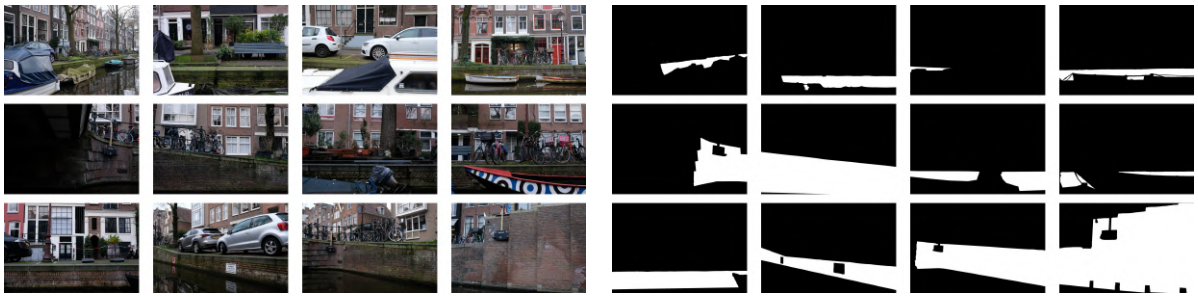
3.1. Semantic segmentation of (quay) walls

First, the development of the quay wall segmentation neural network is discussed, starting with the data setup and further detailing the training configurations. An impressive amount of 1023 labelled images of segmented masonry walls is available for this. Since the aim is to teach a model to be able to distinguish a masonry quay wall specifically, we created an additional dataset of 200 labelled images for this specific task. The learning on this new dataset of quay walls is done by means of fine-tuning to this specific dataset after having pre-trained on the general masonry wall dataset. Three example images of quay walls are shown in figure 3.1 to express the complexity and variety of the scenes in the quay wall images. This variety includes diversity in scene lightning, camera positions and quay wall textures.

Figure 3.1: Three example images of quay walls are depicted to illustrate the complexity and variety within the scene. This complexity includes variations in scene lightning, camera position or quay wall textures. Furthermore, the images often contain partial occlusion by for example boats, signs or ropes.

(a) Images **(b)** Masks

Figure 3.2: (a) A selection of images from the existing wall dataset (before augmentation processing) and (b) corresponding masks. Most of the images are from residential houses or other building facades.



(a) Images **(b)** Masks

Figure 3.3: (a) A selection of images from our quay wall dataset, containing a total of 200 data pairs, (before augmentation processing) and (b) corresponding masks.

3.1.1. Data preparation

Data labelling for quay wall dataset

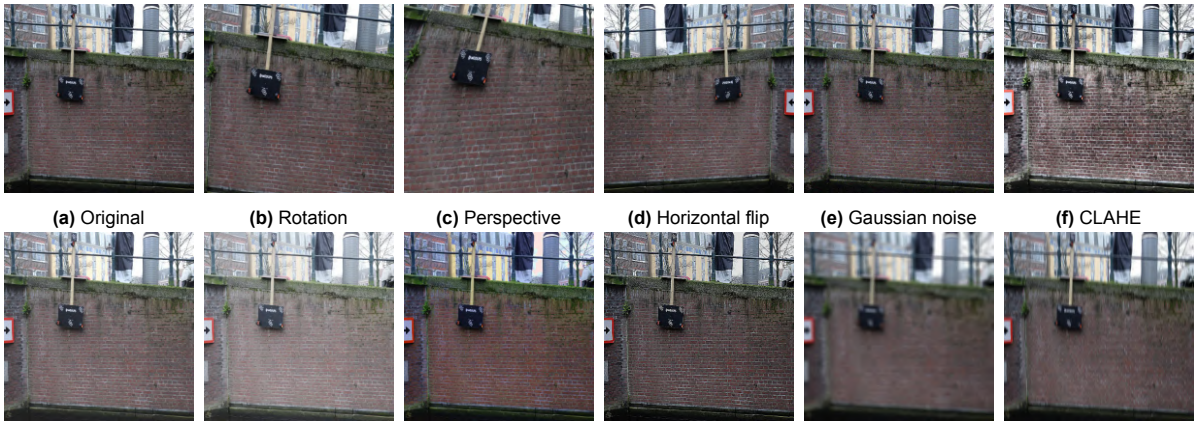
Our labelled data consists of images and their corresponding binary segmentation masks. To give an impression what this looks like, a small selection of both the existing dataset of masonry structures as well as our own labelled dataset of quay walls is depicted in figure 3.2 and figure 3.3 respectively.

For our own quay wall dataset, imagery which is as diverse as possible has been selected and annotated accordingly. It is deemed that a dataset as generic as possible will yield a learning task which is most realistic. Besides, a small number of more difficult images have intentionally been selected as well. These harder to correctly segment images are for example images where the quay wall is completely covered by a boat or images with a masonry building facade close to the quay wall, which could easily be confused as part of the quay wall, due to highly similar pixel texture.

Furthermore, the labelled masks exclude the upper capstone of the quay wall, since it has been shown that the joint between the capstone and masonry wall is often confused as being a crack. Any other non-masonry parts of the quay wall, like signs, are also excluded in the labels. Small objects, like ropes or chains, which occlude small parts of the quay wall have been labelled as part of the quay wall, assuming that the crack segmentation model can handle this variety appropriately.

Data augmentation

The first data preparation step for all input images of the network is to resize them to a fixed size of 512 x 512 pixels. During early experimenting, this size was found to be a reasonable upper-bound with regards to memory constraints, both during training and local inference. Next, to artificially increase the diversity of the database, random (but realistic) transformations are applied to the image data to create synthetic new data. These transformations change the color space or geometry and are only done for the training subset during the training phase. Any data pair from the validation or test subset will not be altered by data augmentation. The addition of this new data, although synthetically made, will reduce the tendency of the model to overfit on training data [44].



(a) Original **(b)** Rotation **(c)** Perspective **(d)** Horizontal flip **(e)** Gaussian noise **(f)** CLAHE

(g) Brightness **(h)** Gamma distortion **(i)** HSV distortion **(j)** Sharpening **(k)** Blur **(l)** Motion blur

Figure 3.4: A set of all the possible augmentations in geometry, color space or blur for an example image of the wall dataset.

Type	Transformation	Occurrence
Geometry	Rotation	*p=0.25*
	Horizontal flip	*p=0.5*
	Four point perspective	*p=0.25*
Color space	Gaussian noise	
	(Contrast Limited Adaptive Histogram Equalization) CLAHE [41]	*Apply one of*
	Change brightness and contrast	*the color space*
	Change gamma	*augmentations*
	Change hue, saturation, value	
Kernel-based	Sharpen	*Apply one of*
	Blur	*the kernel-based*
	Motion blur	*augmentations*

Table 3.1: Composition of augmentations as applied to training images for (quay) wall segmentation. For each of the geometry augmentations, the chance of that augmentation being applied on a data entry is listed in the right column. For both the color space and the kernel-based augmentations, one single augmentation is selected out of each set.
Additional details about the used parameters for these augmentation operations are depicted in figure 3.5.

The set of possible augmentations is visualized in figure 3.4 and is furthermore listed in the augmentation composition table 3.1. The corresponding details of these augmentations and the values for the most important parameters used are listed in figure 3.5. For each augmentation step, the parameters regarding the degree of this augmentation is chosen to be mild. This is to prevent augmentations to be unrealistic and therefore likely confusing for the neural network. During the training on both datasets, each training image is sampled twice and augmented according to the randomly chosen set of augmentations. The frequency of randomly applying any of the augmentations is also listed in the previously mentioned table 3.1. For the transformations on geometry, all three have an independent chance of being applied on a data pair each time. For the transformations in color space and the kernel-based ones, one out of the list of possibilities is randomly selected each time. In this way, a composition of multiple augmentations will be applied for a training image.

3.1.2. Model selection and training configuration

In this study, the segmentation models considered for learning (quay) wall segmentation are FPN, DeepLabV3+, LinkNet and MANet, as all of these are well established architectures found in many different studies. All of the models are pre-trained on ImageNet and the following backbone encoders - with varying depth - are used: ResNet, ResNeXt, ResNeSt. Two different loss functions, including binary-cross-entropy (BCE) and the dice loss, are considered in order to find the most appropriate loss function for each model.

Details of the (quay) wall augmentation operations

- **Rotation**: A rotation is performed between with a degree random sampled between (-7°, 7°).
- **Four point perspective**: A four point perspective transformation is done with four randomly selected points around the image corners. The points are sampled from a Gaussian distribution with a random standard deviation between (0.05, 0.15) relative to image size. This means that 1 equals to full image width or height.
- **Gaussian noise**: On each color channel, Gaussian noise filter is performed with a variance randomly sampled between (25, 50) with a mean of zero. This is done on colors with values ranging from zero to 255.
- **CLAHE** [41]: During this operation, tiles of 8x8 pixels are created on the images and an upper threshold is randomly chosen between (1, 3) for the histogram equalization for each tile.
- **Brightness and contrast**: The brightness and contrast will both be adjusted by multiplication with randomly sampled values between (0.75, 1.25) and (0.85, 1.15) accordingly.
- **Gamma**: The gamma value of the image will be set to a value randomly sampled between (30, 120).
- **Hue, Saturation, Value**: The values for hue, saturation and value (HSV) will independently be shifted by a randomly sampled factor between (-20, 20).
- **Sharpen**: The operation of image sharpening is performed and merged with the original image with a alpha value ranging between (0.2, 0.5).
- **Blur**: Blur operation with a kernel size randomly sampled between (3, 7).
- **Motion blur**: Motion blur operation with a kernel size randomly sampled between (3, 9).

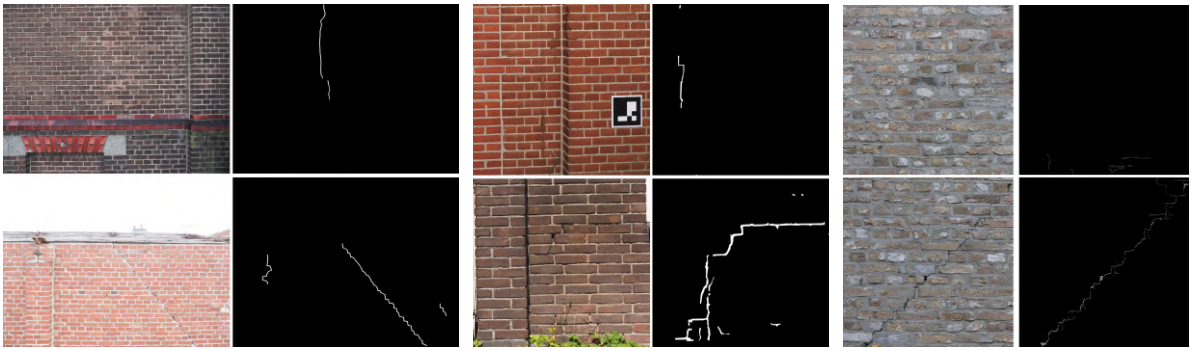
Figure 3.5: Subsequent details of the augmentation operations as listed in table 3.1. For each augmentation, besides the evident horizontal flipping, the corresponding parameters are presented.

All the experiments are done with the Adam optimizer [25] as early experimenting has found superior performance over the stochastic gradient descent (SGD) optimizer. Most notably the Adam optimizer adopts an adaptive learning rate during training compared to being fixed in SGD. Furthermore, a L2-regularization factor of 0.0001 is used and the learning rate is reduced by half when the validation loss has not been improved over 4 successive epochs, starting with a learning rate of 0.0001. To prevent training for unnecessary long, as well as to prevent overfitting, the technique of early stopping is used with a threshold of 10 non-improving epochs. This results in the training loop being stopped after 10 successive non-improving epochs based on the loss. During training, the weights present at the epoch of lowest loss are yielded as final configuration and hence also used for the test set evaluation. These weights are saved as checkpoint during training and later on retrieved after the training loop is finished. At last, the training for every configuration is done for a minimum of 75 epochs.

3.1.3. Transfer learning on quay walls

The best model discovered when trained on the big dataset of masonry walls and its weights will be the starting point for the training of the quay wall segmentation. The intuition behind this transfer learning is that the model will already have a good understanding of what masonry structures are and is next fine-tuned to learn the structure of quay walls specially. This is finally the desired learning goal. This allows for faster training convergence and the possibility to utilize the large dataset which is already available.

When fine-tuning on a new dataset, either all weights can be retrained or only the ones in some layers, typically the lower ones, or the encoder and decoder can be frozen while letting the other weights be mutable. Whether to freeze some layers or let the complete model be freely retrainable is a consideration of how related the target dataset is to the source dataset [23] and how big the target dataset is [45]. Freezing weights serves as a measure of remembering some of the past learning and not to overfit on the new (small) dataset. Since our target dataset is of moderate size with 200 images and the images are furthermore highly related, freezing some weights is deemed to limit performance. Because of this, we will instead perform fine-tune learning with a slightly lower learning rate. This reduces the rate of change on the weights during fine-tuning in order to not overfit.



(a) Dataset 1 **(b)** Dataset 2 **(c)** Dataset 3

Figure 3.6: An arbitrary selection of images and corresponding labels from each of the three datasets, without any data augmentation like tiling performed. Notably (c) shows the pairs of the dataset comprising of generated tiles from high resolution orthographic images.

3.2. Semantic segmentation of cracks

In this section the segmentation model development is discussed starting with the necessary data setup and continuing with the model selection and training configuration. Here we acknowledge the exorbitant number of parameters and possibilities to pursue to find the most optimal model and discuss the heuristic local strategy we used to narrow this search down in order to make it feasible in this scope. During the model development we will first be guided by metrics of the labelled data whereas next visual inspection of the overall pipeline is the primary assessment as discussed in section 3.3.

3.2.1. Data preparation

For this work we have access to an extensive amount of labelled data (1835 labelled images) of cracks in masonry walls. These images are not specifically acquired from masonry quay walls but of other masonry structures like residential houses. In total, it comprises of three datasets with binary pixel-level labelled cracks. Two of these datasets contain unprocessed images of masonry walls while the third contains image tiles taken from a small set of 27 super-resolution images, which are the result of frontal view photos of houses stitched to an orthographic projection. Out of each of these high-resolution images, tiles of 1536 by 1536 pixels are made where tiles not containing any crack label are discarded. This operation of sampling tiles from the larger orthographic images is done once and the following processing steps are the same for each of the three datasets. Two example pairs of images and labelled masks from each of the three datasets are depicted in figure 3.6. In total the combination of all datasets yield 1835 labelled images.

Tiling data to uniform sub-patches

Before data augmentation steps are performed, first the images of varying size are mapped to multiple tiles with an uniform size of 512x512 pixels. The tiles are created on parts of the image with a segment of the mask present, if any exists at all in the data pair. If there is no crack label present in the data pair, a tile is cropped at a random position in the image. An example of the creation of three tiles from a raw input image can be seen in figure 3.7. For each raw image, three tiles are made during training. This number has been chosen such that each time a balanced representation of tiles over the whole images is yielded, while still feasible regarding batch size limitations.

Notably, there is no enforcement of constraining tiles to be partly exceeding the image dimensions. Therefore, it is possible for a tile to be cropped near one of the borders of the image and partly overflow on the image and corresponding label. The neural networks still require an uniform input size of 512x512 pixels so in this case, the image should be padded to preserve this constraint. The most straightforward way of doing so is to just fill the remaining areas with black pixels. However, Huang et al. interestingly point out possible degradation of model performance due to these artificial edges [21]. Removing the typical background of masonry bricks in these areas will add noise to the dataset. Instead, mirroring bordering pixels preserves much of the pixel characteristics of the background. Both strategies are depicted in figure 3.8. Since literature hints to comparative performance improvements when using pixel mirroring, this strategy has been adopted for our experiments.

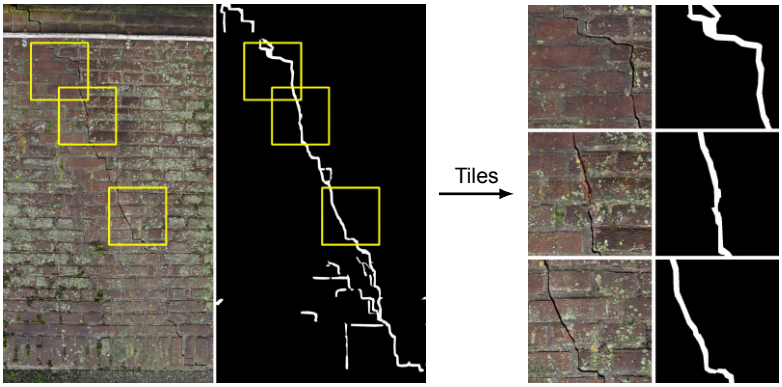


Figure 3.7: An example of the selection of three tiles from the large images yielding three pairs of tiled images and masks, which will be the input of the neural network.

To further stimulate scale-invariance during model training, the tiles are taking at slightly varying scale, but always resized to the uniform size of 512x512 pixels. By slightly zooming in and out, we improve the ability of the networks to reason on different spatial scales. This is especially important since inference of the crack segmentation network is done on multiple scales with unknown real-world spatial size of a pixel.

Data augmentation

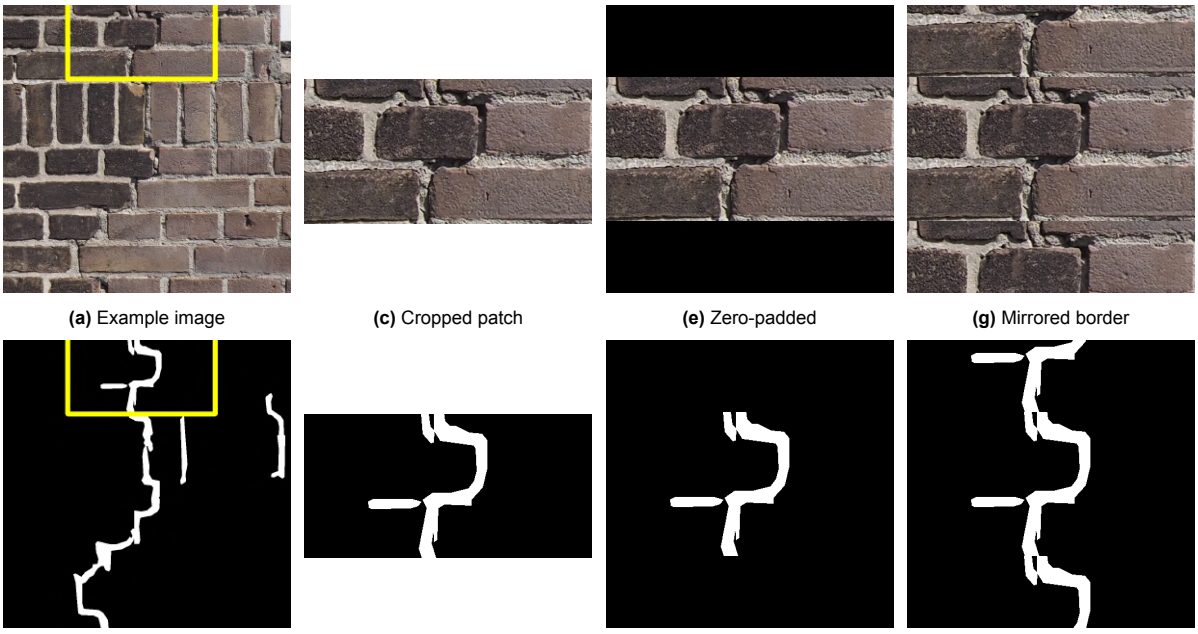
Similar data augmentation as applied on the wall dataset is also applied on the crack dataset for each tile, as yielded from the steps described above. The set of possible augmentations is the same besides additionally including vertical flipping of the image. The frequency of occurrence for this extra data augmentation step as well as the other augmentations is listed in table 3.2. In this table is listed how comparatively to (quay) wall learning the augmentation steps for the crack dataset are performed less often. Similarly, the parameters of the color distortion augmentation such as for contrast limited adaptive histogram equalization (CLAHE) and HSV color shifts are chosen to be more mild compared to the wall data augmentations. Likewise, the blurring kernels are smaller to reduce the amount of blur. These parameters are summarized in figure 3.10 with further details on each augmentation step. The reason for lowering the frequency as well as the intensity of most of the data augmentations is that the segmentation of cracks is involved on a more detailed spatial scale and too much image manipulation might confuse the learning. For an example crack tile from the dataset the set of image augmentations is depicted in figure 3.9. It excludes the effect of horizontal flipping in order to be able to concisely fit it to the page.

Data pruning for noisy labels

Unfortunately, the crack dataset used in this study was labelled with a different intention and interest, where many small sections of missing or damaged joints between bricks are also included in the labels. For our purpose, we are only interested in significant cracks running over several bricks and want to omit such small segments. This label inconsistency due to different domain interest is something we want to mitigate as much as possible in an automated way. We pre-process the dataset before the training loop in order to steer the model into focusing on learning significant cracks only. In an opposite approach, during post-processing small segments could simply be filtered out from the predictions of an already learned model. In this work we choose the pre-processing method to be able to incorporate the variable degree of omitting small-scale segments into the learning process.

Luckily these small blobs of missing or damaged joints are well distinguishable by simple image filtering on the label masks with a certain threshold on their spatial size. Working on the binary annotation masks, we first apply a morphological closing operation, allowing close-by pixels to be merged together, followed by filtering any blob of pixels below a certain threshold. The parameters for both operations are relative to the image size and fixed for every image in each of the three datasets. The proper values for these parameters are estimated by means of visual inspection on a significant portion of the dataset.

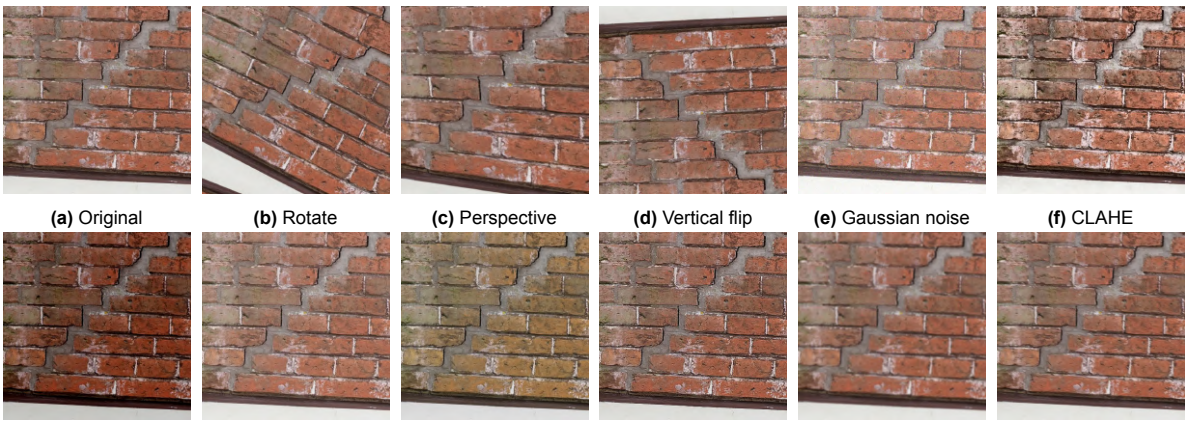
In contrast, although a lot less common, some areas of labels in the dataset with a clear crack are



(a) Example image **(c)** Cropped patch **(e)** Zero-padded **(g)** Mirrored border

(b) Corresponding label **(d)** Cropped label **(f)** Zero-padded **(h)** Mirrored border

Figure 3.8: A visualization of two approaches for image padding with the goal of preserving an uniform dimension: zero-padding and mirroring of border pixels. The need for it arises when tiles are created at the edges of the original image and label and partly exceed the image dimension. Plots (e) and (f) show the result of padding by adding zero pixels around the image or label patch, (g) and (h) alternatively fill the pixels by means of mirroring the pixels around the borders.



(a) Original **(b)** Rotate **(c)** Perspective **(d)** Vertical flip **(e)** Gaussian noise **(f)** CLAHE

(g) Brightness **(h)** Gamma distortion **(i)** HSV distortion **(j)** Sharpening **(k)** Blur **(l)** Motion blur

Figure 3.9: The set of all the possible augmentations in geometry, color space or blur for an example image of the crack dataset. Notably, horizontal flipping is omitted from this illustration to accommodate page layout constraints.

Type	Transformation	Occurrence
Geometry	Rotation	*p=0.2*
	Horizontal flip	*p=0.4*
	Vertical flip	*p=0.4*
	Four point perspective	*p=0.2*
Color space	Gaussian noise	
	(Contrast Limited Adaptive Histogram Equalization) CLAHE [41]	*With p=0.4, apply one of the color space augmentations*
	Change brightness and contrast	
	Change gamma	
	Change hue, saturation, value	
Kernel-based	Sharpen	*With p=0.4, apply one of the kernel-based augmentations*
	Blur	
	Motion blur	

Table 3.2: Composition of augmentations as applied to training images for crack segmentation. For each of the geometry augmentations, the chance of that augmentation being applied on a data entry is listed in the right column. For both the color space and the kernel-based augmentations, one single augmentation is selected out of each set.
Additional details about the used parameters for these augmentation operations are depicted in figure 3.10.

Details of the crack augmentation operations

- **Rotation**: A rotation is performed between with a degree random sampled between (-25°, 25°).
- **Four point perspective**: A four point perspective transformation is done with four randomly selected points around the image corners. The points are sampled from a Gaussian distribution with a random standard deviation between (0.05, 0.10) relative to image size. This means that 1 equals to full image width or height.
- **Gaussian noise**: On each color channel, Gaussian noise filter is performed with a variance randomly sampled between (10, 40) with a mean of zero. This is done on colors with values ranging from zero to 255.
- **CLAHE** [41]: During this operation, tiles of 4x4 pixels are created on the images and an upper threshold is random chosen between (1, 2).
- **Brightness and contrast**: The brightness and contrast will both be adjusted by multiplication with randomly sampled values between (0.80, 1.20) and (0.90, 1.10) accordingly.
- **Gamma**: The gamma value of the image will be set to a value randomly sampled between (40, 110).
- **Hue, Saturation, Value**: The values for hue, saturation and value (HSV) will independently be shifted by a randomly sampled factor between (-10, 10).
- **Sharpen**: The operation of image sharpening is performed and merged with the original image with a alpha value ranging between (0.1, 0.3).
- **Blur**: Blur operation with a kernel size randomly sampled between (3, 5).
- **Motion blur**: Motion blur operation with a kernel size randomly sampled between (3, 5).

Figure 3.10: Subsequent details of the augmentation operations as listed in table 3.2. For each augmentation, besides the evident horizontal and vertical flipping, the corresponding parameters are presented.

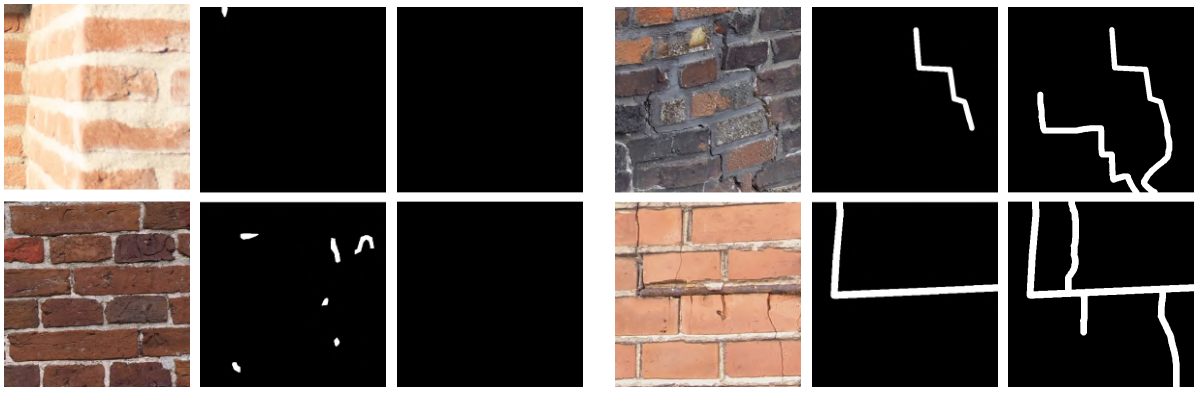


Figure 3.11: Set of examples of data pairs with undesired annotated labels, separated in two types of inconsistencies: Type I) Small damaged or missing parts of the joint between the brick is annotated as a crack. Type II) A part of the image with a clear crack has not been annotated as a crack. For both types the actual ground-truth label is also depicted in **(c)**.

excluded in the annotated label, likely due to human errors during annotating. In figure 3.11 both types of errors, the damaged or missing joints blobs (type I) and the faulty exclusion of actual crack annotation (type II) are depicted. In the same figure the desired ground truth label is also shown, which is manually depicted for these examples. The occurrence of type II inconsistencies are rare while type I errors are present in most of the data pairs. Knowing this and acknowledging that mitigating type II errors is difficult to do in an automated way, we focused on the automated approach to filter out type I errors.

3.2.2. Model selection and training configuration

For crack segmentation we initially consider the same models as for quay-wall segmentation, but given the size of the MANet model being several times higher than the FPN, LinkNet and DeepLabV3+ models, the MANet model is deemed to be unsuitable for segmenting cracks. Using the ResNet-50 encoder within those architectures, the number of learnable parameters for FPN, LinkNet and DeepLabV3+ are 26.1 million, 31.2 million and 26.7 million respectively. In contrast, the MANet model using ResNet-50 adds up to an enormous 147 million parameters. The main reason why this is undesired for segmenting cracks, while satisfactory for segmenting quay walls is that the crack segmentation model will be processed over multiple tiles from a single image, yielding an undesired amount of inference time. More details on how the model is applied in inference is discussed later on in section 3.3. Additionally this larger model complexity increases training time as well.

The Adam optimizer is again selected, together with L2-regularization using the same penalty of $1e^{-4}$, to mitigate the effects of overfitting. Early experimenting has shown that using a learning rate of $1e^{-4}$ is a good fit for each training configuration. As discussed earlier in section 3.2.1, for each data image three sub-tiles of 512x512 pixels are made, effectively extending the dataset. Artificially prolonging the dataset like this acts similar to increasing the training duration. To balance this, the minimum number of training epochs is reduced from 75 to 50, compared to (quay) wall training. Likewise the early stopping threshold is reduced from 10 to 5 epochs, although in practice the runs have shown to not improve after 50 epochs. In the same fashion as for training the wall segmentation models, the learning rate is scheduled to be halved after 4 consecutive epochs of no reduction in loss.

On a pixel-level, there is severe class-imbalance with only $2.18\% \pm 0.04$* of the pixels annotated as a crack. To combat this class imbalance and remove this bias as much as possible, the following weighted loss functions are considered: generalized dice loss (GDice), weighted binary cross-entropy (WBCE) and focal loss. The latter two have the adjustable weight and alpha hyperparameters to trade off the predictive output of the model into making either false positive or false negative errors. Dedicated runs varying these parameters are used to heuristically find the most optimal value for both.

*Note: this class coverage metric is a random variable since it is calculated after the previously discussed data augmentation with non-deterministic operations like random zooming and random tile creation. Therefore the standard deviation is also reported.

(a) Original resolution **(b)** Down-scaled resolution (512x512)

Figure 3.12: A depiction of the problematic effect of losing important detail to distinguish the fine-grain crack when resizing the whole image from 9504x6336 to 512x512 pixels. (a) depicts the image at original size (left) and a close-up view (right) of the crack located near the bottom left of the 'no parking' sign. (b) shows the same image and close-up view after downsizing the image to 512x512 where the crack is almost vanished due to this down-scaling.

3.3. Segmentation workflow

After building the neural networks for segmenting both quay walls and cracks in masonry walls, we design an automated workflow detailing how these models are applied in inference. We first present this workflow and motivate the reasoning behind this design (section 3.3.1). Afterwards we present a method to evaluate prediction consistency over multiple images in section 3.3.2.

3.3.1. Prediction with overlapping sliding window

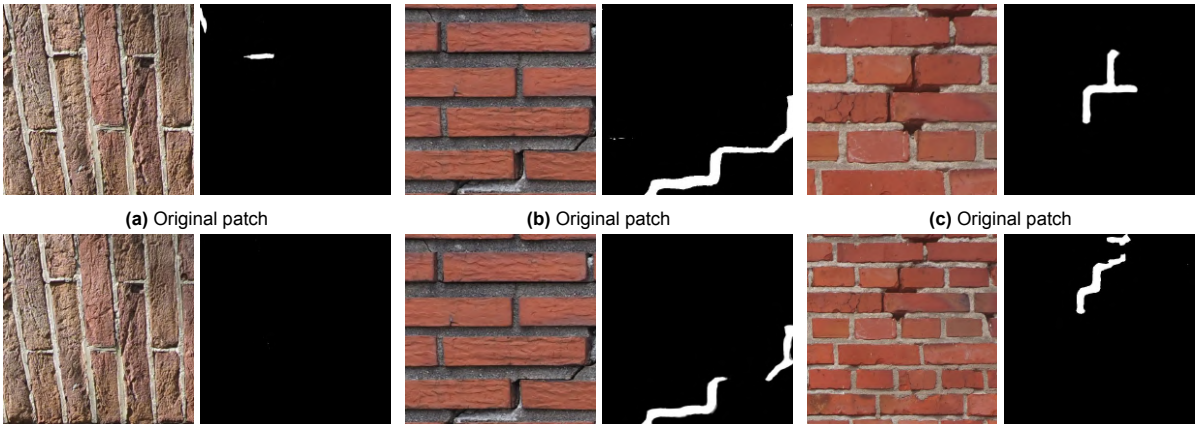
Motivation of sliding window approach

The design of a structured workflow for crack segmentation is largely inspired by the work of Huang et al. [21]. This work deals with the inference of segmentation networks on areal remote sensing imagery and addresses the issue of such images being too large to process as a whole. Instead, an approach where the image is divided and processed into smaller overlapping sub-tiles from a sliding window grid over the whole image and the resulting segmentation results are stitched together is proposed. We adopt a similar strategy since we also deal with large images of roughly 9000x6000 pixels, likewise creating the tiles with overlap between them. In contrast, simply resizing input images to 512x512 pixels, like our approach for segmenting quay walls, will likely remove the details we are interested in since cracks are often only a few pixels wide. This undesired effect is shown in figure 3.12 for an example image. Furthermore as pointed out by Huang et al [21], the benefit of the utilized overlap is motivated by the expected translational variance of segmentation models as well as its tendency to predict zeros at the edges of the tiles.

The segmentation models are sensitive to both scaling and translations when processing input images. In practice this means that a tile and a slightly scaled or translated version can all three have different outputs when ran through the segmentation neural network. This variance on prediction output is depicted in figure 3.13 for three example images. Much of these sensitivities are mitigated by the multi-scale pyramid approaches adopted in the network architectures as well as in the data augmentation strategy by performing random zooming during training. To furthermore mitigate this effect during inference, we extend the sliding window grid approach as explained up until this point by performing this on three different scales. Inferring on three different levels of scale allows the model to infer at different levels of resolution as well as to sequentially create tiles at slightly shifted positions. The most effective scale of inference is in general unknown, therefore adopting this multi-scale approach is beneficial by considering several scales and translations. The intermediate results obtained on each of these three scales are then merged together and an appropriate threshold is chosen to average to a final result.

Details of sliding window approach

This approach of inferring in a grid on different scales is schematically summarized in figure 3.14. The workflow starts with processing the input image to distinguish the quay wall against the background, using the developed neural network. The whole image is first pre-processed by resizing to the uniform size of 512 x 512 pixels when inferred for the quay wall segmentation. Next, a slightly larger area of the resulting output is retrieved, by taking the minimum encapsulating bounding box, to allow for some



(a) Original patch **(b)** Original patch **(c)** Original patch

(d) Re-scaled **(e)** Translated **(f)** Re-scaled and translated

Figure 3.13: Three examples which show the translation and scale variance of the neural network predictions. On top the image tiles and corresponding inferred predictions of those tiles are shown. Below a slightly translated or scaled variant of the same images and their corresponding predictions are shown. Notably, these predictions are slightly different than the original ones.

extra surrounding pixels before processing further. The predicted quay wall mask is later on used by removing any crack predictions positioned outside the mask area.

Experiments have been done to heuristically find the most optimal values for degree of overlap, scale resolutions and averaging threshold. Early experimenting has shown effective performance when using an overlap of 128 pixels, for tiles with a size of 512 x 512 pixels. A degree of overlap larger than that increases the number of tiles significantly while resulting in marginal prediction improvements. This finally yields an automated workflow which is able to segment cracks on a wide variety of images of quay walls. These results as well as supporting visualizations are presented in the corresponding section in chapter 4.

Segmentation workflow evaluation

After designing the crack segmentation workflow that incorporates the developed neural networks, we face the challenge of evaluating its performance despite the absence of any ground-truth labels in this context. Relying on performance metrics from the labelled (test) dataset as used during fully-supervised training of the crack segmentation networks will not depict a truthful picture on the performance of the overall pipeline on the unlabelled quay wall imagery.

First, a manually selected small batch of quay wall images is yielded and further manual evaluation is performed by means of categorizing the predictions to pre-defined types. These categories can be regarded as sub-categories on top of the confusion matrix items, to be able to highlight the performance in more detail. The definition of each category is listed in the next chapter at section 4.3.1, more specifically in table 4.5. The images are manually selected across several acquisitions without looking at any predictions at first. Since the occurrence of cracks is rather rare (luckily), a roughly equal split between images without cracks and images with at least one present is enforced to ensure the dataset is balanced Expressing the performance on image-level instead of doing so pixel-wise with pixel-level labels makes it feasible for our work. This post-inference labelling on image-level is however more sensitive for human bias.

3.3.2. Determination of output consistency by overlap

For the next processing step, we consider that the dataset consists of images taken with the purpose of performing photogrammetry, resulting in sequential images having a lot of overlap between them. We propose a method that makes use of this overlap to quantitatively assess the consistency of the output, which is a much desired property of the segmentation workflow. When a crack is found in an image, it can be assumed that images capturing the same section of the quay wall also contain and detect this same crack. Furthermore, as the images are in most cases also sequential in time, the lighting and other image properties are mostly consistent. This means that if the workflow works well,

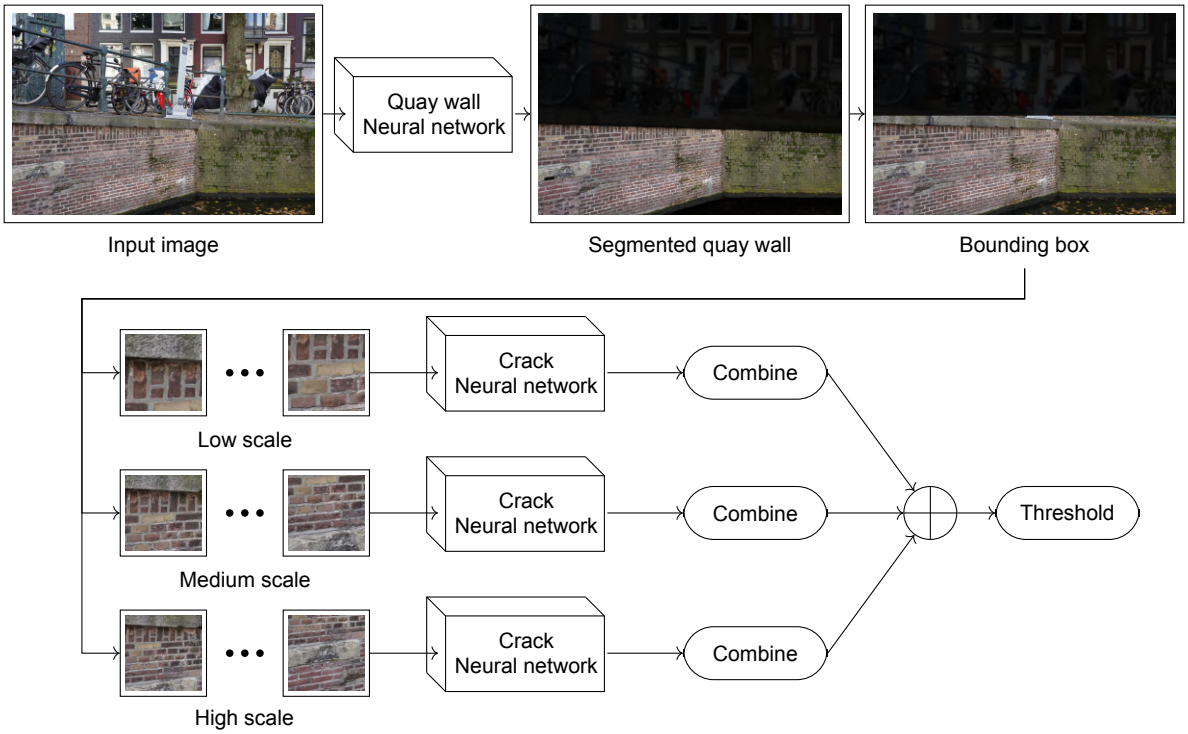


Figure 3.14: The workflow for segmenting cracks in quay wall images. First, the quay wall itself is segmented from the background. This is done by resizing the arbitrary-sized input image to the fixed size of 512 by 512 pixels and infer it on the quay wall segmentation neural network. This pre-processing step is left out in the scheme to allow for a more concise visualization. Afterwards, the surrounding minimum bounding box is retrieved and next split into a grid of overlapping tiles on three different scales: low, medium and high scale. These tiles are again resized to 512x512 pixels and processed by the crack segmentation neural network. At last, the output for each tile is stitched together to output an prediction mask at original resolution at every scale and combined together.

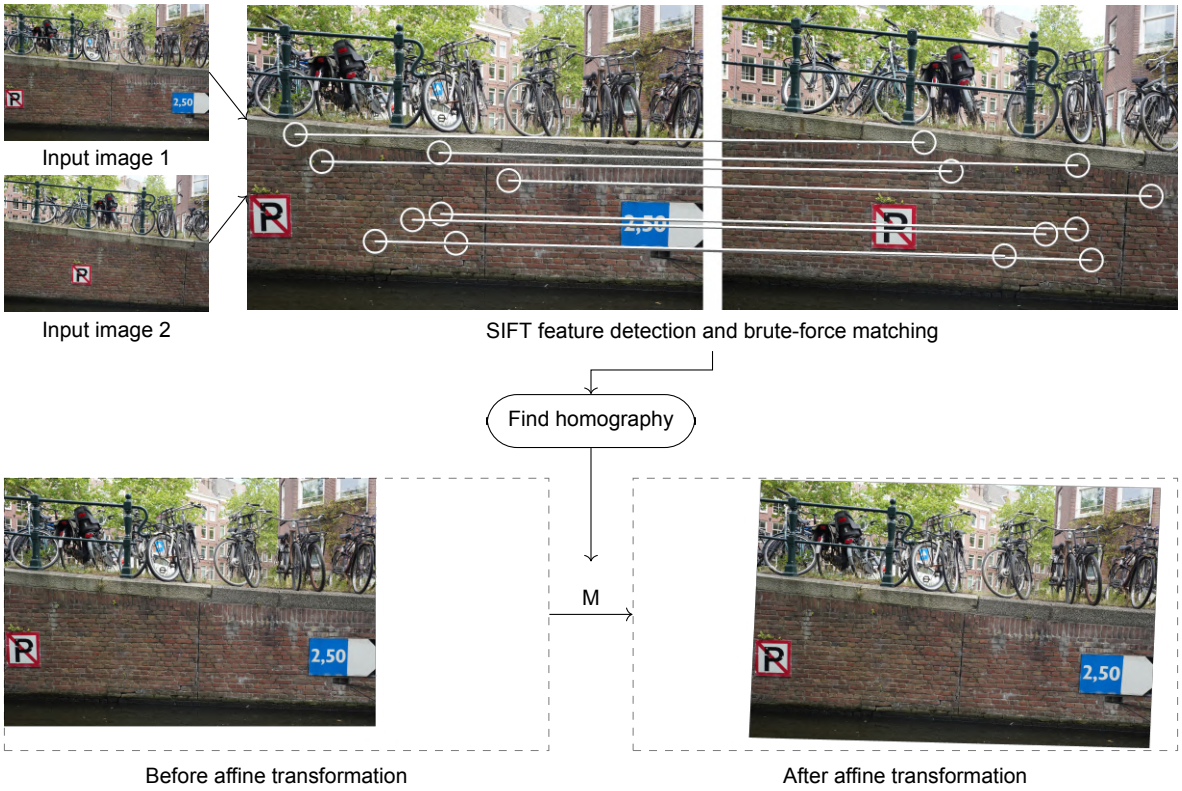


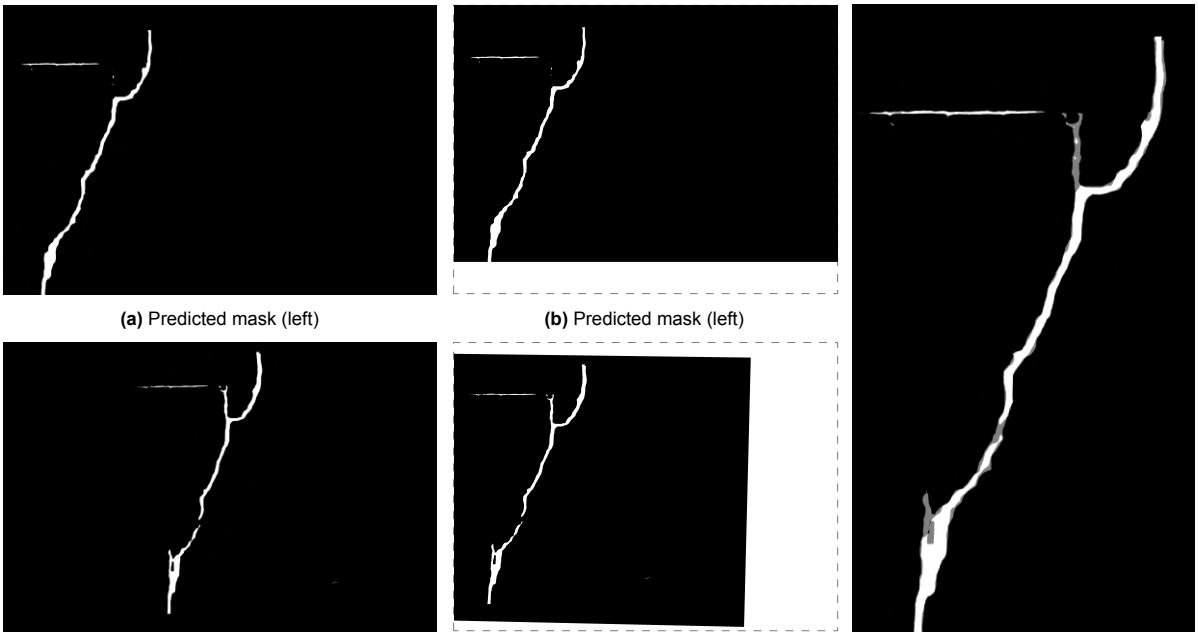
Figure 3.15: A schematic visualization of the processing steps performed on two subsequent images to estimate an affine transformation mapping from one image to the other. The first step is to find SIFT feature points in both images and afterwards find matches between those by means of brute-force matching on their descriptors. For visualization purposes only a few matches are shown in this depiction, although normally several thousands are found. By using RANSAC, the best fitted transformation is then computed on the found matching points. This method finally allows for overlaying overlapping images onto each other. This same mapping can also be performed on their corresponding predicted segmentation masks.

we can reasonably expect its segmentation outputs to also be similar. In this way we can assess the consistency regarding slightly differing input, where images similar in input space are deemed to result in outputs being close to each other as well. Before detailing how this approach is performed using photogrammetry, it should be noted that the assumption of high similarity among subsequent images may not hold in every instance. Variations in image acquisition can introduce discrepancies like motion blur, that do not necessarily reflect on the segmentation workflow's consistency but rather on the impact of such anomalies. In such cases we will not assess the consistency of the segmentation workflow but merely the impact of such effects on it.

The method works by taking two overlapping images and performing feature detecting and matching onto this pair. The classic method of doing so with SIFT detection and brute-forcing matching has found to work well. On the resulting matching points, RANSAC is used to find the best mapping from one image to the other in a least-squares way. This finally results in a 3x3 affine projection matrix which will be applied on one of the corresponding prediction masks to end up with the two predictions being registered and overlaid correctly. From this, the consistency metric can naturally be derived by calculating the intersection over union (IoU). These steps have been schematically visualized for a concrete summary in figure 3.15. The result of such an affine warping on an example pair of overlapping prediction masks is shown in figure 3.16.

3.4. Crack characteristics post-processing

In the section above we have shown the segmentation workflow and how we can exploit the photogrammetric property of image overlap to quantitatively assess its output consistency. In this section we continue utilizing the additional information available from the photogrammetric adjustment, to design methods assessing crack characteristics. As pointed out before, both crack length and crack width are



(a) Predicted mask (left) **(b)** Predicted mask (left)

(c) Predicted mask (right) **(d)** Warped predicted mask (right) **(e)** Both predictions on top of each other

Figure 3.16: An example result of an affine projection applied on a pair of overlapping crack prediction masks in order to estimate the intersection over union (IoU) between the two. The affine projection matrix is derived from the matching on the corresponding quay wall images and estimated in a least-squares manner. (a) and (c) depicts the predictions mask from the left and right image respectively. The result of applying the affine projection of the second mask is depicted in (d), while (b) shows the original mask of the left image with additional padding to match the dimensions of the other warped mask. (e) finally shows the sum of the left mask and warped right mask from which the IoU metric is derived.

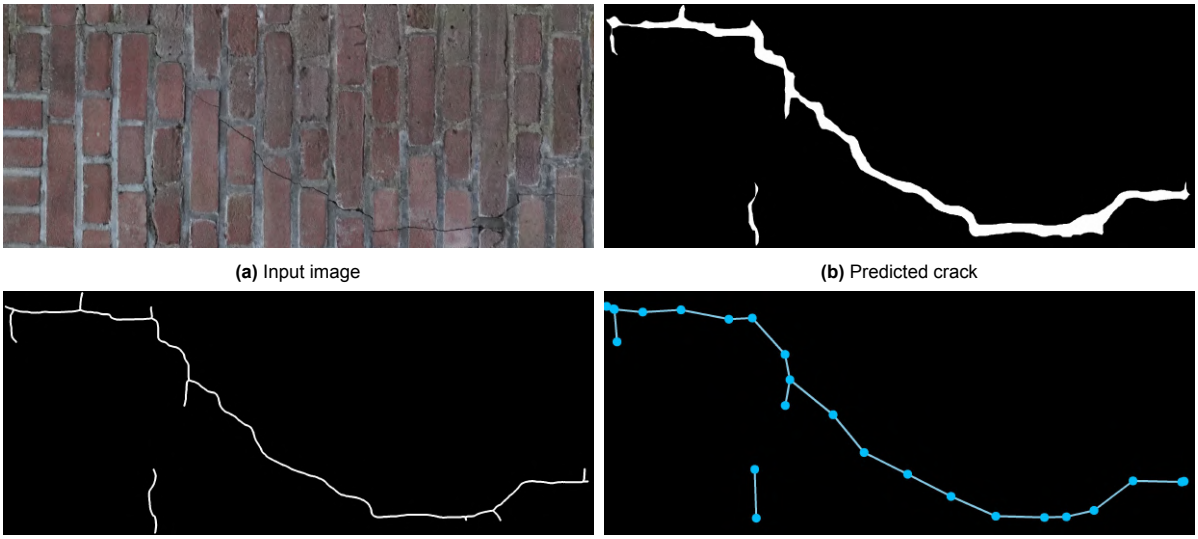
key indicators of crack severeness and hence this part of our methodology focuses on their evaluation in an algorithmic way. For the original aim of deformation assessment, the imagery has been adjusted by means of backwards intersection where 3D positions are accurately derived from 2D image points. The acquired 3D coordinates are additionally linked with terrain reference points to yield true-to-scale camera positions. For the goal of deriving crack characteristics this is especially useful since the output of such characteristics can be elevated to be in the unit of meters or millimeters, instead of solely on pixel-level.

Graph crack representation and length estimation

Next, an algorithmic arrangement of image processing steps is presented as a method for estimating the crack length on the prediction masks from the segmentation workflow. The crack length is first estimated in the prediction mask on pixel-level, after which a forward intersection is done using the known positions and interior camera properties to project the 2D crack points onto a plane. This plane is positioned through the quay wall such that the projected points all lie on the quay wall.

The predicted crack segmentation mask represents a crack or multiple segments of cracks as a binary path of certain thickness along the length of the crack. This path can be reduced to a 1 pixel wide representation without losing its original connectivity. In image processing literature this is often called the skeleton of a binary mask. For our work we adopt the method of making a skeleton as proposed by Lee [30], since it minimizes any undesired small side branches being made.

From this 1 pixel wide representation we sample pixels laying on this skeleton path to construct a 2D graph representation of the crack. Starting from any end of the skeleton, all of the skeleton pixels are visited in a breath-first run through the whole mask. During this run, when the euclidean distance to the closest sampled pixel is greater than a certain threshold, a new pixel point is sampled and added to the graph with an edge connection to the previous pixel. Doing this for every end point of the skeleton mask ensures that every pixel has been visited at least once. These image processing steps have been summarized in algorithm 1. A depiction of the skeleton and graph representation of an example prediction mask can be seen in figure 3.17.



(a) Input image

(b) Predicted crack

(c) Crack skeleton (dilated)

(d) Constructed multi-line path

Figure 3.17: Four visualizations of intermediate processing outputs during the construction of a graph representation from a crack segmentation prediction. (a) and (b) depict the image patch containing a crack and its corresponding binary crack segmentation predicted by the workflow. (c) shows the 1 pixel wide skeleton of this mask, slightly enlarged by morphological dilation for visualization enhancement. (d) shows the final graph representation with its points and connecting edges, resulted from the breath-first run through the skeleton pixels.

Algorithm 1 Construction of graph representation on segmentation mask

Input: Binary prediction mask of a crack
Output: Multi-line path (graph) defined by set of image points and connecting edges

Step 1) Reduce binary mask to 1 pixel wide representation, its skeleton [30]
Step 2) Compute all the endpoints by performing morphological hit-or-miss transformation with 3x3 kernels for each of the four possible pixel directions.
for each endpoint **do**
  if *endpoint* has been visited by any BFS run before: skip it and process next *endpoint*
  Starting from *endpoint*: perform breath-first Search (BFS) along skeleton pixels and store of visited pixel-points every *variable* amount of pixels as path points
  During BFS: keep track of previously placed path point to link edge when next one is placed
end for
Step 3) Result of the multiple BFS runs is a set of path points and connecting edges
Step 4) Optionally filter path points which are too close to a neighbouring one

Result: Graph of the crack mask: a set of image points and connecting edges

These points and edges derived on the images are finally mapped to a 3D representation. This mapping is done by forward intersecting the graph points to a plane positioned through the quay wall in 3D world space. The steps of estimating this 3D plane are twofold. First, at least one overlapping image and corresponding prediction mask is retrieved as well. On these overlapping images, SIFT points are found and matched between images. This set of matching SIFT points is furthermore reduced to the subset surrounding the crack pixels in the prediction masks. Second, the matched SIFT points are transformed to 3D world points using forward intersection. Subsequently, by means of minimizing the projection error with principal component analysis (PCA), a plane is fit through this set of points. Having estimated this plane, the graph points and edge connectivity is transformed to 3D by projecting these image points onto this plane. This is done by means of intersecting the image point vectors with the plane. Since the forward intersection used in this step is based on true-to-scale coordinates, the 3D graph representation of the crack will be in the unit of meters. The total length is then the sum of the distances of all the edges.

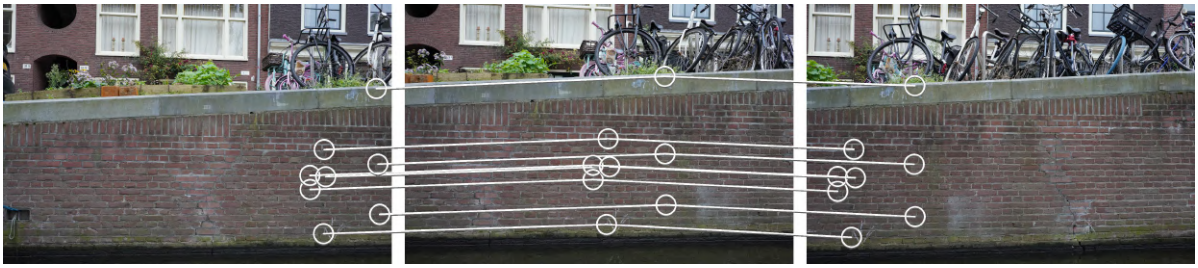


Figure 3.18: An example feature detection and matching done on three subsequent images, forcing matches to exist between all three images. This extra enforcement reduces the amount of matches roughly by one fourth. In this specific example, there are 4846 matches between the left and center image and 4738 between the center and right image, and 1167 matches found the exist between all three images. This extra enforcement adds more redundancy during the forward intersection and its enhances precision.

The steps for forward intersection are summarized in algorithm 2, which concludes all the steps for calculating crack length starting from the estimated 2D graph. It mentions the SIFT matching and plane estimation from three overlapping images since this is beneficiary compared to only two images for added redundancy and therefore increased precision for the intersection estimation. An example of three overlapping images being matched with each other is depicted in figure 3.18. To further strengthen the prediction results of the three images, the predictions are combined to a single prediction using affine transformation mapping as discussed before and was depicted in figure 3.15.

Algorithm 2 Determination of crack length from segmentation predictions

Input: Three overlapping images and corresponding crack prediction masks
Output: A sparse 3D graph representation and its derived length in meters

Step 1) SIFT feature detection and matching on all three images
Step 2) Project all three prediction onto each by other using affine projection estimated with RANSAC
Step 3) Construct 2D graph on combined mask (see algorithm 1)
Step 4) For each of the three images: compute minimum bounding box around crack and filter SIFT points positioned inside
Step 5) Forward intersect these filtered points to 3D points and fit plane by performing principal component analysis (PCA)
Step 6) Forward intersect all points on 2D graph on detected plane

Result: 3D graph representation of the crack; length is sum of all edges

Estimating crack width by orthographic projection

Utilizing the fitted projection plane through the quay wall as constructed before, we provide a method for computing a rough estimate on the crack width. As was the case for estimating crack length, doing so on pixel-level is rather straightforward but also of poor value. Instead we construct an image where pixels relate to true scale in millimeters. Most of the photogrammetric and image processing details needed for this are already discussed above for the method of assessing crack length, allowing us to be rather concise here.

Applied on the fitted projection plane, a grid of points with equal distance of 1mm by 1mm between each is made at the positions surrounding the predicted mask. From this, a new image is constructed from our original image and prediction mask, having a pixel point for each 3D point from this grid. This reconstruction from 3D world points to 2D image points yields an image with an orthographic projection where every pixel has a approximate size of 1 mm by 1mm. In doing so, camera or view distortions are removed as well. The result of this true-to-scale orthographic photo for an example image is displayed in figure 3.19.

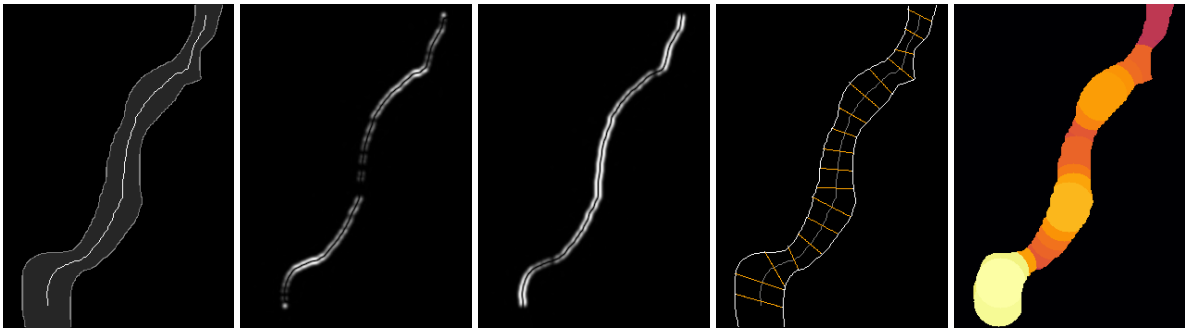
First, we define the width of a crack at any position on the crack to be the distance, from edge to edge, perpendicular to its local direction. Crack width is therefore a metric defined at a specific location on the crack which can be further aggregated along the crack's length. From the true-to-scale orthographic

(a) Input image **(b)** Scaled ortho photo (zoomed to crack)

Figure 3.19: A depiction of an example input image of a quay wall containing a crack (a), together with its estimated ortho photo scaled to a uniform pixel size of 1mm by 1mm. This projection removes any perspective, rotation and camera effects. This is done by constructing a grid and sample points every 1mm by 1mm on the best fitting plane along the quay wall.

image, the method of calculating the width starts by uniformly sampling points along the prediction mask and determining a vector perpendicular to the local direction. The points are sampled on the earlier defined skeleton and the spatial direction of this skeleton is determined by the magnitudes of the corresponding gradient images in both x and y direction. In order to spatially smooth those values, a Gaussian blur filter is performed on the skeleton image before calculating the gradient images. The gradient images in both directions are combined and normalized to result in a vector placed perpendicular to the direction of the skeleton. As a final step, the pixel-wise width at that position is calculated as the length of the constructed line between both edges along the perpendicular vector. Figure 3.20 depicts both gradient images from a small patch of a prediction mask and the determined perpendicular widths along this mask. The final visualization result is constructed by means of morphological dilation to cover the area of the prediction mask.

Since we are interested in estimating on a very fine detail of only a few millimeters, it should be noted that this method of estimating the crack width works within the value range of the noise which can be expected from the photogrammetric forward intersection. The predicted output from the segmentation workflow is also not precise enough to exactly match the pixels of the crack. From a mere philosophical point of view you can even argue that this definition of what pixels are part of a crack and which are not is not well defined. It becomes clear that the combined inaccuracy of both the photogrammetric properties as well as the segmentation workflow output does not allow us to determine any crack width on the critical thresholds of only a few millimeter as found in literature. Instead we focus our experiments on evaluating crack width regarding categorizing those on different orders of magnitudes. This is done by constructing a histogram representing the occurrence of width values along the skeleton of the crack.



(a) Mask, skeleton, edges **(b)** X-axis gradient **(c)** Y-axis gradient **(d)** Perpendicular lines **(e)** Width interpolation

Figure 3.20: Four images depicting key processing steps for the calculation of crack width at sampled points within the true-to-scale orthographic photo. At image (a) is shown a small patch of a prediction mask as well as the derived skeleton and edges. Images (b) and (c) are the gradient images in direction of the X-axis and Y-axis respectively after Gaussian blurring on the skeleton image. From these two, (d) is the depiction of the result showing uniformly sampled point on the skeleton and lines through those perpendicular to the skeleton's direction. The width is calculated by means of retrieving the length of the line between the edges of the mask. At last, (d) shows the derived width values and interpolated over the whole prediction mask by means of morphological dilation.

3.5. Implementation

The implementation of the code for training the neural segmentation models is done using PyTorch Lighting [12]. The networks together with pre-trained weights on ImageNet are retrieved from the package Segmentation Models [22].

The training of these neural networks is done on the DelftBlue [1] supercomputer on its NVIDIA Tesla V100 32GB GPU.

The inference of the neural networks and the photogrammetric mapping is done on a laptop with a NVIDIA RTX 3050 Ti 4GB GPU. This less powerful GPU acts as a bottleneck on how big the neural network input can practically be, where 512 by 512 pixels is found to be a reasonable upper bound.

Experimenting is done on a wide variety of quay wall images from photogrammetry acquisitions with derived camera positions in real-world coordinate system, which has been made available by Geodelta. Their supporting methods for forward intersection as well as for the construction of true orthographic images is utilized for our work.

3.5.1. Reproducibility and data availability

The datasets considered for this work are not publicly available, which is therefore limiting the degree of reproducibility. If desired, the datasets can be replicated with a segmentation annotation tool, of which some are freely accessible. Because this is heavily labor intensive, it is alternatively possible to rely on any publicly available dataset of cracks in masonry walls. Notably, the work of Dais et al. [8], which is referred to earlier in this thesis, has an open source imagery dataset of masonry wall cracks. In any case, when data is limited, we suggest to omit the use of the masonry wall dataset, as its contribution to the overall performance is probably small. Instead, directly training on a masonry quay wall dataset is deemed to produce sufficiently good results.

Furthermore, we heavily relied on the computation power of DelftBlue for the training of the resulting neural networks, which already consumed several hours. Acknowledging that similar resources might not be readily available to everyone, we suggest that effective learning outcomes are still achievable albeit to a lesser degree. By adjusting the training approach to accommodate less powerful computing environments, one can manage to reduce both training time and memory usage. Accordingly, it can be decided on to consider smaller neural networks.

4

Results

This chapter contains the results related to the methodology presented up until now. First, the evaluation of the developed neural networks for both (quay) wall learning (section 4.1), as well as crack segmentation learning (section 4.2) is presented. For both tasks the effectiveness of the different architectures and training configurations is listed in a table. Notably, the fine-tune learning for quay walls and the predictive output balancing for crack segmentation are discussed. Next, the results of the segmentation workflow, combining the developed neural networks, are also shown (section 4.3). The results on the analysis of crack length and width utilizing the photogrammetry principles in shown in section 4.4. At last two interesting case studies showing significant cracks in quay walls are discussed (section 4.5).

4.1. Semantic quay wall segmentation

4.1.1. Segmentation of walls

For the development of the (quay) wall segmentation networks, four different segmentation model architectures are considered. Additional training configuration details have been earlier discussed in section 3.1.2. We opted for a batch size of 8 as this is empirically found to be a reasonable upper-bound for every configuration regarding memory constraints. The training on the large dataset of general walls is done for at least 75 epochs. The results of the different configurations are summarised in table 4.1, reporting the performance metrics as evaluated on the test set.

As discussed earlier, the most meaningful metrics are the F1-score and IoU metrics but the others have been reported as well. The configurations are compared to each other by their IoU score and for each model, the ones scoring highest on this metric are highlighted. In case of a tie, the configurations with the least amount of parameters are chosen. Below is explained why such a smaller neural network is preferred. Additionally the epoch when the lowest loss was obtained is also listed. The weights of this best performing model are saved and these are considered as the final model for each run, which is also used for the test set evaluation.

Due to the stochastic nature of the Adam optimizer and the data augmentation, the metrics from a training run can best be regarded as a random variable, and hence the mean and standard deviation of every metric over 5 independent runs is reported. Although it can be noted that the variance between runs is rather small.

For each combination of model and backbone encoder, the amount of trainable parameters (weights) is also reported. It can be noted that the MANet architecture is significantly larger compared to the other three model architectures. In general a smaller model is preferred over a bigger model without significant reduction in performance for two main reasons. First, smaller models are less likely to overfit on relatively small amount of data, which is important to keep in mind when finetuning on the small dataset of masonry quay walls specially. Second, smaller models are generally faster in inference. In practice the inference time for the much larger MANet models was still found to be workable and

Model	Backbone	Loss	Parameters* [millions]	Epoch**	Accuracy [%]	Precision [%]	Recall [%]	F1-score [%]	IoU [%]
FPN	ResNet-50	BCE	26.1	64 ± 9	96.4 ± 0.1	97.9 ± 0.1	96.7 ± 0.2	97.3 ± 0.1	94.7 ± 0.1
		Dice	-	60 ± 7	96.5 ± 0.1	97.2 ± 0.2	97.6 ± 0.2	97.4 ± 0.1	94.9 ± 0.2
	ResNeXt-50	BCE	25.6	42 ± 17	96.4 ± 0.1	97.9 ± 0.2	96.7 ± 0.2	97.3 ± 0.1	94.7 ± 0.2
		Dice	-	57 ± 12	96.5 ± 0.1	97.3 ± 0.2	97.5 ± 0.2	97.4 ± 0.1	94.9 ± 0.1
	ResNeSt-50	BCE	28.0	52 ± 14	96.6 ± 0.1	97.9 ± 0.2	97.0 ± 0.1	97.4 ± 0.1	95.0 ± 0.2
		Dice	-	63 ± 11	96.7 ± 0.1	97.4 ± 0.2	97.7 ± 0.1	97.5 ± 0.1	95.2 ± 0.2
	ResNeSt-26	BCE	17.6	53 ± 7	96.5 ± 0.1	97.9 ± 0.2	96.9 ± 0.1	97.4 ± 0.1	94.9 ± 0.1
		Dice	**-**	$\mathbf{69 \pm 6}$	$\mathbf{96.7 \pm 0.1}$	$\mathbf{97.4 \pm 0.2}$	$\mathbf{97.6 \pm 0.2}$	$\mathbf{97.5 \pm 0.1}$	$\mathbf{95.2 \pm 0.1}$
	ResNeSt-14	BCE	11.2	46 ± 14	96.2 ± 0.1	97.7 ± 0.2	96.7 ± 0.1	97.2 ± 0.1	94.5 ± 0.2
		Dice	-	60 ± 9	96.3 ± 0.2	97.1 ± 0.1	97.5 ± 0.1	97.3 ± 0.1	94.7 ± 0.2
DeepLabV3+	ResNet-50	Dice	26.7	47 ± 16	96.5 ± 0.2	97.1 ± 0.3	97.7 ± 0.1	97.4 ± 0.1	95.0 ± 0.3
	ResNeXt-50	**Dice**	**26.1**	$\mathbf{60 \pm 11}$	$\mathbf{96.6 \pm 0.1}$	$\mathbf{97.3 \pm 0.3}$	$\mathbf{97.7 \pm 0.2}$	$\mathbf{97.5 \pm 0.1}$	$\mathbf{95.1 \pm 0.2}$
LinkNet	ResNet-50	Dice	31.2	60 ± 10	96.5 ± 0.1	97.3 ± 0.1	97.5 ± 0.2	97.4 ± 0.1	94.9 ± 0.2
	ResNeXt-50	Dice	30.6	56 ± 11	96.7 ± 0.1	97.4 ± 0.3	97.7 ± 0.1	97.6 ± 0.1	95.3 ± 0.2
	ResNeSt-50	Dice	33.1	70 ± 14	96.7 ± 0.1	97.5 ± 0.1	97.7 ± 0.2	97.6 ± 0.1	95.3 ± 0.2
	ResNeSt-26	**Dice**	**22.7**	$\mathbf{80 \pm 14}$	$\mathbf{96.8 \pm 0.1}$	$\mathbf{97.5 \pm 0.2}$	$\mathbf{97.7 \pm 0.1}$	$\mathbf{97.6 \pm 0.1}$	$\mathbf{95.3 \pm 0.2}$
	ResNeSt-14	Dice	16.2	67 ± 11	96.5 ± 0.1	97.3 ± 0.1	97.6 ± 0.1	97.4 ± 0.1	95.0 ± 0.2
MANet	ResNet-50	Dice	147	70 ± 14	96.5 ± 0.1	97.3 ± 0.2	97.5 ± 0.1	97.4 ± 0.1	95.0 ± 0.2
	ResNeXt-50	**Dice**	**146**	$\mathbf{59 \pm 12}$	$\mathbf{96.8 \pm 0.1}$	$\mathbf{97.6 \pm 0.1}$	$\mathbf{97.7 \pm 0.2}$	$\mathbf{97.6 \pm 0.1}$	$\mathbf{95.3 \pm 0.2}$
	ResNeSt-50	Dice	149	68 ± 8	96.6 ± 0.3	97.4 ± 0.4	97.6 ± 0.1	97.5 ± 0.2	95.1 ± 0.4
	ResNeSt-26	Dice	138	63 ± 6	96.6 ± 0.2	97.6 ± 0.1	97.4 ± 0.2	97.5 ± 0.1	95.1 ± 0.2
	ResNeSt-14	Dice	132	64 ± 7	96.5 ± 0.1	97.5 ± 0.2	97.4 ± 0.1	97.4 ± 0.1	94.9 ± 0.1

Table 4.1: Various architecture, encoder backbone and loss functions considered and the corresponding metrics evaluated on the test set averaged over 5 independent runs. For each model architecture, the ones yielding highest IoU score have been highlighted. For the FPN architecture, both BCE and Dice loss functions are considered, but since dice loss shows consistently better performance over BCE for all configurations, only dice loss is used for subsequent runs, to help reduce the amount of experiments. It can be noted that all configurations yield very high performance, with larger encoders performing slightly better.
* The total number of trainable parameters of a network.
** Epoch where the lowest loss was obtained for the validation set.

the issue of overfitting is additionally mitigated by using L2-regularization when training, with a higher penalty parameter set when finetuning on the smaller quay wall dataset.

First the FPN model was evaluated where the dice loss function showed superior performance over the BCE loss function for every configuration. In order to further reduce the viable search space of possible training configurations, only the dice loss function is used for subsequent model configurations, assuming it continues to outperform the BCE loss.

4.1.2. Fine-tune learning on quay walls

The four best performing models, as highlighted in table 4.1, on the large wall dataset and corresponding weights are used as starting point for fine-tuning to the small dataset of 200 quay wall images. Additionally to using these weights as initial weights, the minimum amount of epochs trained for is reduced to 50, the learning rate is halved to $5e{-}5$ and the L2-regularization parameter is increased tenfold to 0.001. The reason for these changes is to prevent the network from overfitting due to training too much. The rest of the training configuration is unchanged compared to the previous training runs. The same metrics together with the epoch of lowest loss have been summarized in table 4.2. The metrics of each of the four configurations are again similar and all rather high, although the difference between them and the standard deviations are higher than found previously in table 4.1. The model configuration using the DeepLabV3+ architecture achieves the highest IoU score on the test set. On top of that, it also achieves the most desired results when evaluated visually compared to the others, as further discussed next, and hence is highlighted as best performing model.

The loss over epochs during training is depicted in figure 4.1 for both training on the large dataset and

Model	Backbone	Loss	Parameters* [millions]	Epoch**	Accuracy [%]	Precision [%]	Recall [%]	F1-score [%]	IoU [%]
FPN	ResNeSt-26	Dice	17.6	46 ± 12	98.1 ± 0.3	94.8 ± 1.0	96.9 ± 0.8	95.8 ± 0.6	92.0 ± 1.0
DeepLabV3+	**ResNeXt-50**	**Dice**	**26.1**	**43 ± 4**	**98.3 ± 0.0**	**94.8 ± 0.4**	**98.0 ± 0.3**	**96.3 ± 0.1**	**93.0 ± 0.2**
LinkNet	ResNeSt-26	Dice	22.7	54 ± 10	98.3 ± 0.1	95.4 ± 0.8	97.0 ± 0.9	96.1 ± 0.4	92.6 ± 0.7
MANet	ResNeXt-50	Dice	146	44 ± 9	98.1 ± 0.4	95.1 ± 0.9	96.2 ± 2.7	95.6 ± 1.2	91.8 ± 1.9

Table 4.2: Various architecture, encoder backbone and loss functions considered and the corresponding metrics evaluated on the test set of the best found model when fine-tuned on the smaller quay wall dataset. The initial weights are set to the best found model for each configuration trained on the big masonry wall dataset. For the FPN architecture both BCE and Dice loss functions are considered, but since Dice shows consistent superior performance over BCE for all configurations, only Dice loss is used for subsequent runs.
* The total number of trainable parameters of a network.
** Epoch where the lowest loss was obtained for the validation set.

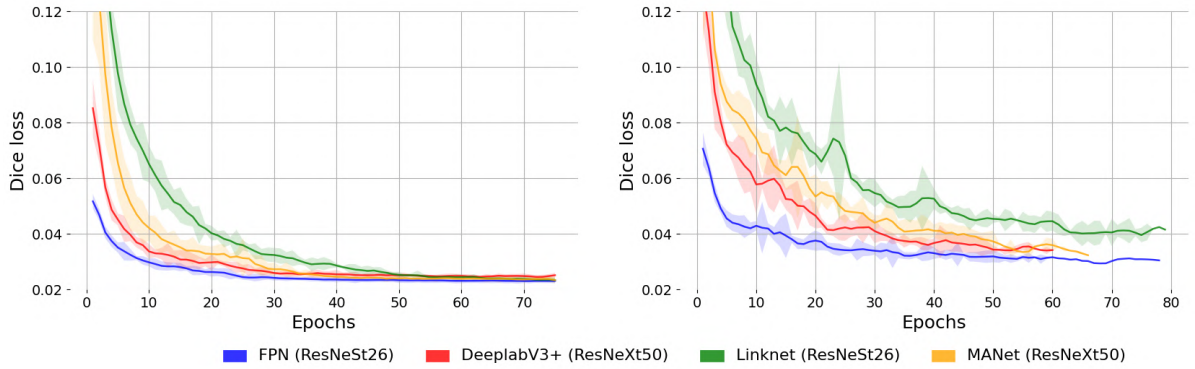


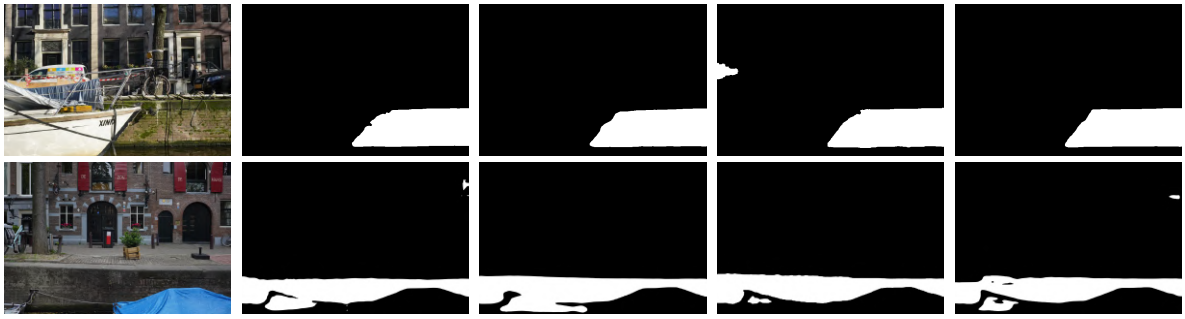
Figure 4.1: The mean dice loss of five independent runs on the validation set during training iterations for each model architecture considered in combination with the best working backbone encoder. (Left) shows the loss during training on the big general walls dataset (1023 images) whereas (right) depicts the loss during training when fine-tuned on the smaller dataset of 200 quay wall images with initial weights being the weights of the previously trained models.

fine-tuning on the quay wall dataset for the four different configurations. The loss reduces significantly during the first few epochs and further reduces with a lower rate afterwards until it reaches a plateau, as can in general be expected when training neural networks. The convergence to this plateau during fine-tuning is at a slightly higher loss value, telling us that learning quay wall segmentation is a more difficult task for the networks compared to training on the larger dataset. Furthermore the loss over epochs shows a bit more bumps for fine-tuning, which might be an indicator that the learning rate was set too high, which especially seems to be the case for the LinkNet model.

Visual evaluation on unlabelled images

Up until this point, every training configuration and corresponding neural network is evaluated and its performance is compared to others by means of its test set metrics, predominately looking at the IoU metric. Every neural network performs really well and all are able to achieve high metrics with only little difference compared to each other and hence frankly, most will likely be sufficient for our use-case. Additionally to acquiring these metrics, the output of the four fine-tuned neural networks have been visually assessed on many different MODUPS images as well, both on labelled images from the test set and on unlabelled images.

When visually assessed, they also perform really similar to each other with some key differences. In general the DeepLabV3+ network has the most difficulties segmenting fine details while the other three are better at doing so. Four example images, corresponding ground truth and the predictions from the DeepLabV3+ model are depicted in figure 4.3, showing this coarsely detailed output. For example ropes or small parts of boats in front of the quay wall are often included in the predicted mask for this model. Still, a coarse mask of the quay wall with such areas included is deemed to be sufficient for our use-case, since we assume that the crack segmentation network can handle these non-masonry wall segments appropriately. Therefore, this inability to exclude fine details within the segmentations is



(a) Image **(b)** FPN **(c)** DeepLabV3+ **(d)** LinkNet **(e)** MANet

Figure 4.2: Two images from the test set of the quay wall dataset (a) and the four segmentation predictions from the best model configurations of FPN (b), DeepLabV3+ (c), LinkNet (d) and MANet (e). The two images show the rare occurrence of false positive predictions where small parts of the streets are also predicted as quay wall for the LinkNet model as well as for the FPN and MANet models.

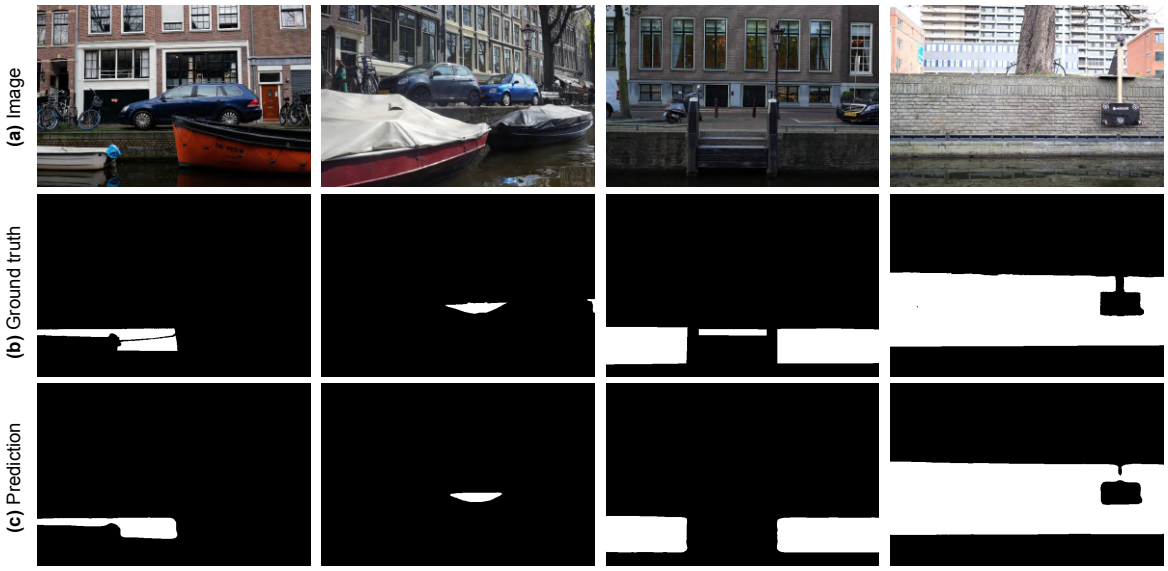


Figure 4.3: Four images from the test set (a) and corresponding ground truth masks (b) together with the predictions from the DeepLabV3+ model (c). The model is able to properly segment the quay wall on a coarse level excluding fine details.

satisfactory.

There are a few cases, although the occurrence is quite rare, where the the FPN, LinkNet and MANet models incorrectly include small parts of the facade of a masonry house in the prediction, while the DeepLabV3+ is correctly segmenting these areas as background, as depicted in figure 4.2. We reason that this is more problematic as compared to the inability of the DeepLabV3+ model to exclude fine details on the quay walls, and consequently utilize the DeepLabV3+ configuration as network for the segmentation workflow.

4.2. Semantic crack segmentation

Most training configurations for segmenting quay walls have shown satisfactory results. As we will see next, segmenting cracks in masonry brick walls is an inherently more complex problem requiring more thought on the selection of loss function, data pruning and data augmentation strategies. Here we leave out the MANet model because of its large size, being several times bigger than the other model architectures, as we have seen in the section above. Although we have argued that this bigger size is not an issue for the quay wall segmentation network, the expected drop in inference speed is deemed to be problematic when segmenting cracks. This is because the crack segmentation model is inferred on numerous smaller sub-tiles of the original image as part of the final segmentation workflow, whereas

for quay wall segmentation the image is processed only once. FPN, DeepLabV3+ and LinkNet model architectures are considered with several training configurations and backbone encoders to find the best-suited one.

Due to the significantly larger dataset, training of the crack segmentation networks has found to be notably slower compared to the training of the wall segmentation networks. The training time highly depends on the model architecture and backbone encoder used as well, with bigger networks resulting in higher training time. To reduce the training time for training the crack segmentation networks, the minimum number of training epochs is reduced from 75 to 50, the early-stopping threshold is reduced from 10 to 5. Additionally, 3 runs per training configuration are performed instead of 5. This reduction in training time makes it feasible to perform more experiments with varying training configurations. For similar reasoning, we switch out the ResNet-50 encoder for the much smaller MobileNet encoder, since ResNet-50 has shown weaker performance compared to its counterparts. The MobileNet encoder is particularly interesting because of its much smaller size.

4.2.1. Selection of loss function

For the task of segmenting quay walls, the binary cross-entropy (BCE) loss and Dice loss have shown very similar results. The selection of loss function for segmenting cracks requires more careful consideration, since it can have a huge impact on mitigating the inherent class imbalance present in the dataset as well as for balancing the precision and recall. The latter is important in deciding between what errors are more desirable, false positive or false negative errors. For these reasons we evaluate the following loss functions: generalized dice loss (GDice), weighted binary cross-entropy (WBCE) loss, and focal loss. For the latter two, the weight and alpha parameters respectively are used to address the class imbalance and predictive trade-off and several values are considered to determine the most optimal one. For the GDice loss this weight is fixed to be inversely proportional to the labelled pixel volume across the dataset at the start of training.

Both the focal and WBCE loss give flexibility to balance false positive and false negative errors by means of their weighting parameter. This ability to balance the model is especially interesting in relation with the noisiness in the labels in the dataset, as pointed out earlier. This noise is a combination of incomplete or missing cracks in the labels or parts of the image misclassified as a crack. Would we have known which type of error is predominantly present, this noise could be counteracted by using appropriate weight and alpha values. In practice, both types of errors are considered equally undesired and hence we base the selection of the optimal alpha and weight values on the F1-score, the harmonic mean of precision and recall.

The multiple values for the weight and alpha parameters are considered for an otherwise fixed training configuration, using the FPN architecture with MobileNet backbone encoder. To reduce the search space of hyperparameters further down the line, these values are assumed to also be optimal for other training configurations. This assumption is made since these parameters are only related to dataset properties. The results from the test set are depicted in figure 4.4 for both loss functions with a precision-recall curve and a depiction of achieved F1-scores, presenting the metrics on varying alpha and weight values. The balancing act between precision and recall is clearly visible in the precision-recall curve where values for both parameters lay in order on the curve, representing the segmentation ability of the models. In this space the perfect model would lie at the point $(1, 1)$. An alpha value of 0.85 and a weight value of 4 yields the highest F1-scores in this configuration and is therefore used for subsequent runs.

4.2.2. Evaluation of training configurations

After having optimized the parameters for both loss functions, the training runs for all the other considered configurations are performed. These results are listed in table 4.3. The best model in terms of achieved IoU and F1-score is the FPN model using a ResNeSt-50 backbone encoder and GDice loss function. Consequently, this model and trained weights are integrated as part of the segmentation inference workflow. There are some close runner-ups like its variant using the focal loss or the LinkNet ResNeXt-50 GDice configuration. Interestingly, configurations using the less deep encoder variants ResNeSt-26 and ResNeSt-14 show similar high performance as well. Instead of directly choosing the one yielding absolute highest test set metrics, it would also be a proper decision to go for any of these

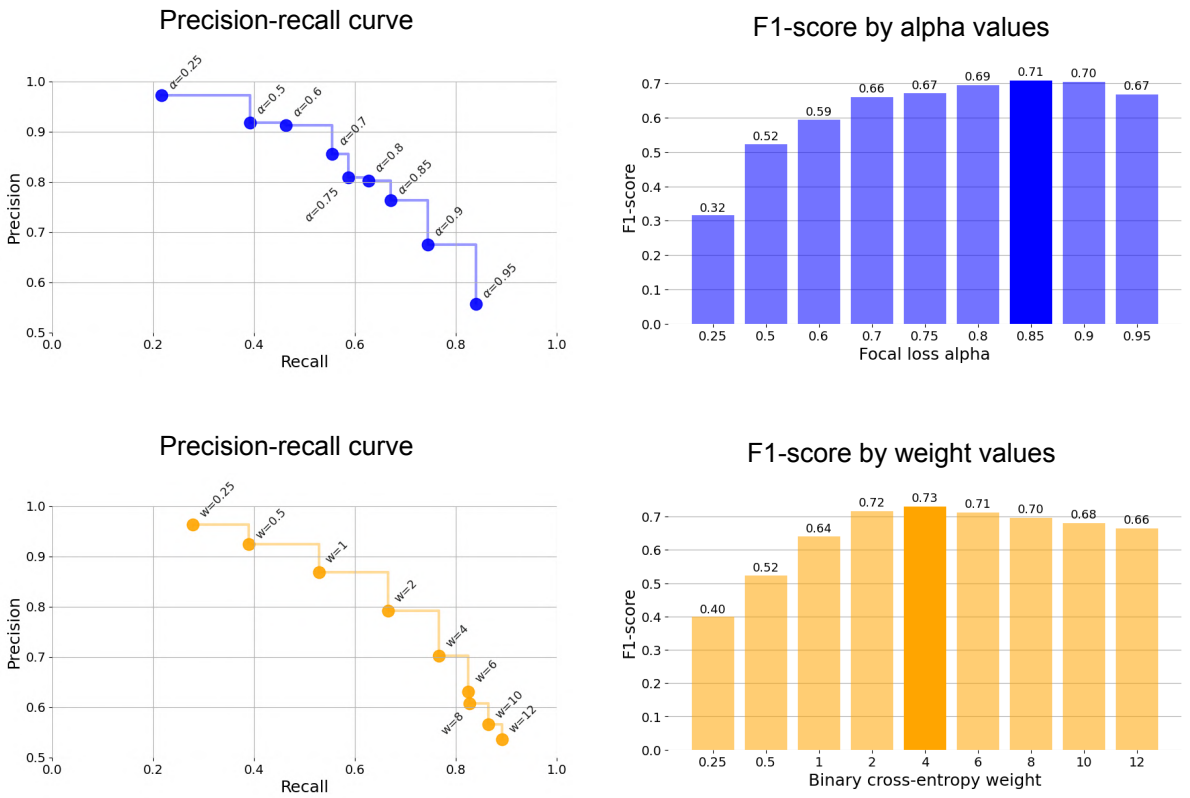


Figure 4.4: For both the focal (up) and weighted binary cross-entropy (WBCE) loss (down) loss, two plots depicting the predictive output by varying values of the weight and alpha parameters respectively are shown. For every configuration the FPN model architecture with MobileNet backbone encoder is used. (Left) shows the precision-recall curve for different values of alpha and weight while (right) shows the F1-scores, which are the harmonic mean of precision and recall, for each alpha and weight value. Both plots are based on metrics from the test set.

lighter variants. It is assumed however that the FPN ResNeSt-50 size is sufficiently small.

Furthermore there are a numerous of other interesting patterns and information which could be derived from this table. To start, note how the accuracy column shows very little variety between runs and every achieved accuracy is very high, close to its maximum value. This can be easily understood by looking at the accuracy formula again:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Because of the severe class-imbalance present in the dataset, most pixels will be rightly segmented as background pixels (TN) and hence be the major contributor of the final accuracy metric. Therefore this metric will always be very high and therefore yield little information.

Next, it is rather surprising how the configurations using the focal loss are outperforming the corresponding ones using WBCE loss, even though both loss functions use optimized weighting parameters. This could be explained by the fact that the focal loss additionally reduces the loss contribution of more confident data pairs and therefore focuses more on the more difficult ones during training, as controlled by the parameter gamma.

Moreover, it can be seen how the predictive balance between precision or recall of several models differ. To highlight one such example, note how LinkNet with ResNeSt-14 using WBCE yields an F1-score of 74.9 while LinkNet with ResNeSt-50 and GDice results in a F1-score of 74.6. With both F1-scores rather similar, the respective precision and recall scores respectively yield 80.7 and 70.4 for the one using ResNeSt-50 and 72.7 and 77.8 for the ResNeSt-14 configuration. The metrics from both runs are therefore reversely correlated to each other. Besides seeing this trade-off, it is also impressive that

a much smaller model with half of the size of the other is able to achieve very similar output.

At last it is clear that there is more variety of performance between different configurations than seen before during the task of wall segmentation learning. On top of that, it is ambiguous which encoder yields the best results, since this actually differs between model architectures, where some seem to work better for a certain architecture than as part of another. The complexity of the dataset also becomes clear, where during wall segmentation learning the metrics were close to the maximum value, this is not the case for the task of learning cracks. Apparently it is more difficult to learn patterns for this dataset.

Model	**Backbone**	**Loss**	**Parameters* [millions]**	**Epoch** [%]**	**Accuracy [%]**	**Precision [%]**	**Recall [%]**	**F1-score [%]**	**IoU [%]**
FPN	MobileNet	WBCE	3.6	45 ± 4	98.7 ± 0.1	70.2 ± 1.2	76.7 ± 2.3	73.0 ± 0.9	58.3 ± 1.0
		GDice	-	49 ± 2	99.0 ± 0.0	79.3 ± 1.0	72.8 ± 1.8	75.5 ± 1.3	61.9 ± 1.7
		Focal	-	42 ± 3	98.7 ± 0.1	76.3 ± 1.8	67.1 ± 1.2	70.8 ± 0.5	56.1 ± 0.4
	ResNeXt-50	WBCE	25.6	44 ± 1	98.8 ± 0.1	67.9 ± 1.4	75.3 ± 2.1	69.5 ± 2.3	58.6 ± 1.4
		GDice	-	33 ± 0	98.6 ± 0.2	77.9 ± 1.8	67.8 ± 2.6	72.0 ± 4.4	57.6 ± 3.5
		Focal	-	43 ± 6	98.9 ± 0.0	76.0 ± 2.1	77.3 ± 2.2	76.4 ± 1.1	63.0 ± 1.4
	ResNeSt-50	WBCE	28.0	47 ± 2	98.4 ± 0.1	62.9 ± 1.2	77.9 ± 0.6	69.1 ± 0.7	53.5 ± 0.4
		GDice	-	$\mathbf{49 \pm 4}$	$\mathbf{99.0 \pm 0.0}$	$\mathbf{82.9 \pm 0.8}$	$\mathbf{76.1 \pm 2.4}$	$\mathbf{78.8 \pm 1.0}$	$\mathbf{66.4 \pm 1.2}$
		Focal	-	38 ± 3	98.9 ± 0.1	76.7 ± 0.7	78.9 ± 1.2	77.6 ± 0.3	64.4 ± 0.6
	ResNeSt-26	WBCE	17.6	44 ± 3	98.5 ± 0.4	66.1 ± 0.8	73.7 ± 1.2	69.5 ± 0.2	54.0 ± 0.4
		GDice	-	47 ± 2	98.9 ± 0.0	80.5 ± 0.5	72.9 ± 0.3	75.9 ± 0.2	62.8 ± 0.2
		Focal	-	44 ± 4	98.9 ± 0.1	77.9 ± 3.7	78.5 ± 3.2	77.8 ± 0.4	64.8 ± 0.6
	ResNeSt-14	WBCE	11.2	47 ± 4	98.7 ± 0.2	69.2 ± 0.9	75.3 ± 0.8	71.8 ± 1.0	56.8 ± 1.1
		GDice	-	44 ± 2	98.9 ± 0.0	79.3 ± 0.8	73.8 ± 0.4	75.8 ± 1.0	62.7 ± 0.9
		Focal	-	45 ± 3	98.9 ± 0.1	76.1 ± 2.5	76.5 ± 3.4	76.0 ± 0.6	62.4 ± 0.7
DeepLabV3+	MobileNet	WBCE	3.2	38 ± 3	98.7 ± 0.1	72.8 ± 3.2	70.7 ± 1.6	71.3 ± 0.9	56.2 ± 1.1
		GDice	-	43 ± 1	98.9 ± 0.0	79.1 ± 1.8	67.3 ± 3.8	72.2 ± 1.8	57.5 ± 2.1
		Focal	-	44 ± 6	98.8 ± 0.0	75.9 ± 1.0	71.8 ± 0.6	73.5 ± 0.4	59.3 ± 0.6
	ResNeXt-50	WBCE	26.1	45 ± 3	98.8 ± 0.0	74.1 ± 1.8	77.0 ± 1.3	75.2 ± 0.7	61.2 ± 0.8
		GDice	-	53 ± 6	99.0 ± 0.1	80.6 ± 0.6	75.9 ± 1.3	77.9 ± 1.0	65.4 ± 1.4
		Focal	-	40 ± 4	99.0 ± 0.1	78.9 ± 2.2	77.1 ± 1.2	77.6 ± 1.4	64.9 ± 1.7
LinkNet	MobileNet	WBCE	2.5	46 ± 2	98.6 ± 0.1	70.2 ± 1.7	67.3 ± 2.2	68.1 ± 1.9	52.6 ± 2.3
		GDice	-	43 ± 1	98.8 ± 0.1	78.0 ± 2.1	67.8 ± 1.9	71.9 ± 0.4	57.4 ± 0.2
		Focal	-	41 ± 4	98.7 ± 0.1	74.9 ± 2.6	64.7 ± 2.6	68.7 ± 1.3	53.5 ± 1.4
	ResNeXt-50	WBCE	30.6	42 ± 5	98.7 ± 0.0	71.9 ± 0.9	75.5 ± 2.5	73.2 ± 1.7	58.5 ± 2.3
		GDice	-	47 ± 2	99.0 ± 0.0	83.5 ± 0.9	74.3 ± 0.6	78.2 ± 0.4	65.5 ± 0.4
		Focal	-	44 ± 2	98.9 ± 0.0	78.3 ± 2.1	76.9 ± 1.9	77.3 ± 0.8	64.9 ± 0.6
	ResNeSt-50	WBCE	33.1	47 ± 2	98.6 ± 0.2	67.5 ± 4.2	73.0 ± 3.3	69.2 ± 1.8	54.0 ± 1.9
		GDice	-	44 ± 3	99.0 ± 0.0	80.7 ± 1.8	70.4 ± 2.0	74.6 ± 1.1	60.6 ± 1.3
		Focal	-	48 ± 1	99.0 ± 0.0	77.7 ± 1.7	74.1 ± 1.2	75.4 ± 0.8	61.5 ± 1.0
	ResNeSt-26	WBCE	22.7	44 ± 3	98.8 ± 0.1	72.3 ± 2.0	75.9 ± 1.2	73.9 ± 1.2	59.5 ± 1.5
		GDice	-	47 ± 3	98.9 ± 0.0	79.7 ± 1.1	74.0 ± 1.1	76.0 ± 0.7	62.4 ± 0.9
		Focal	-	44 ± 4	98.9 ± 0.0	79.2 ± 0.8	76.1 ± 2.8	77.4 ± 1.3	64.3 ± 1.5
	ResNeSt-14	WBCE	16.2	49 ± 0	98.8 ± 0.1	72.7 ± 0.8	77.8 ± 0.7	74.9 ± 0.6	60.6 ± 0.8
		GDice	-	46 ± 4	98.9 ± 0.0	80.8 ± 0.4	71.8 ± 1.4	75.6 ± 0.8	61.7 ± 1.1
		Focal	-	47 ± 0	98.9 ± 0.1	78.2 ± 1.2	75.7 ± 1.3	76.7 ± 0.3	63.5 ± 0.1

Table 4.3: Various architecture, encoder backbone and loss functions considered and the corresponding metrics evaluated on the test set averaged over 3 independent runs. The Focal and WBCE loss functions encode a weight and alpha parameter of 4 and 0.85 respectively. The configuration FPN - ResNeSt-50 using GDice, as highlighted, yields best F1-score and IoU score.
* The total number of trainable parameters of a network.
** Epoch where the lowest loss was obtained for the validation set.

On top of assessing the performance of the model based on the metrics applied on the test set, the most effective model has also been visually evaluated. The FPN model with ResNeSt-50 backbone

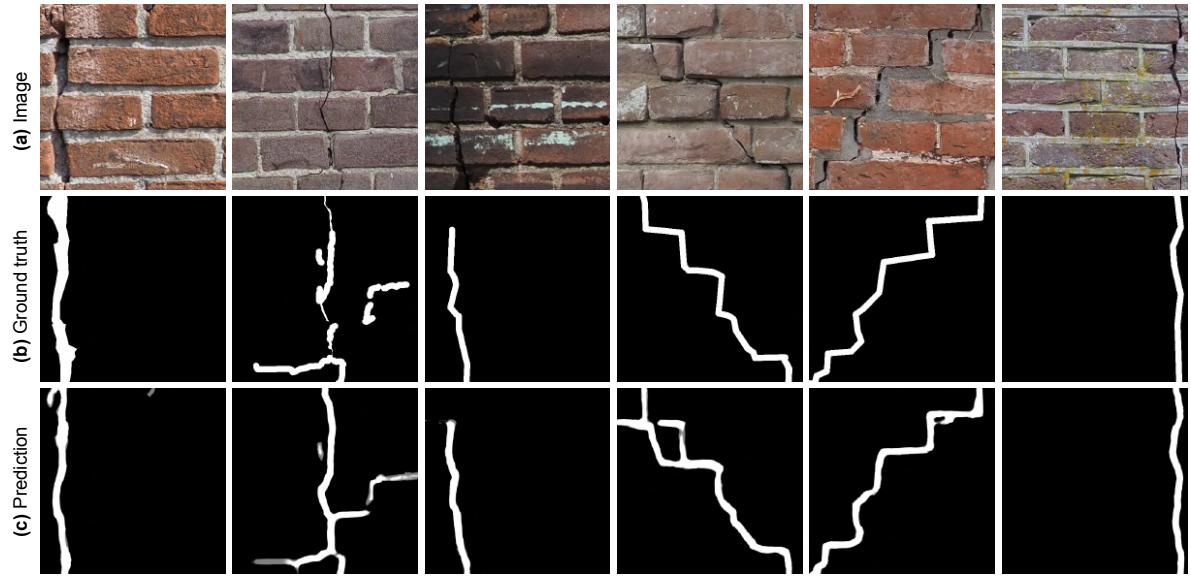


Figure 4.5: A depiction of six images (a) and corresponding labels (b) from the test set. The prediction masks as outputted from the FPN model using ResNeSt50 encoder with GDice loss is furthermore shown (c). Visually it can be seen that the labels and prediction masks have a large overlap between each other.

using the GDice loss function as applied on the test set for some cases is depicted in figure 4.5. From this visualization it becomes clear that the labels and the predictions agree for a large degree.

4.2.3. Data pruning and data augmentation

Having explored the workings of different network and training configuration, we next experiment with additional data augmentation strategies. First, the dataset is assumed to contain too many small labels of cracks or damage on a very fine level of detail. This is undesired for our work since we are interested in segmenting cracks on a larger spatial scale, excluding too-fine segments. To steer the model into leaving out these predictions, these small labels are filtered out from the labelled masks during data fetching (data pruning). Second, the yielded data tiles are created at different levels of scale practically resulting in random zooming data augmentation. This is assumed to be beneficial because the model will be inferred on multiple different spatial scales within the segmentation workflow. These two variations as applied on the data during training are performed on the best working configuration found earlier: FPN - ResNeSt-50 with GDice loss. The metrics as yielded on the test set are listed in table 4.4. For each of the three subsequent training configurations, slightly worse test set metrics are the result. This degradation in performance, although of small order of magnitude, is an indicator that these additional data augmentations make the data set more difficult to learn.

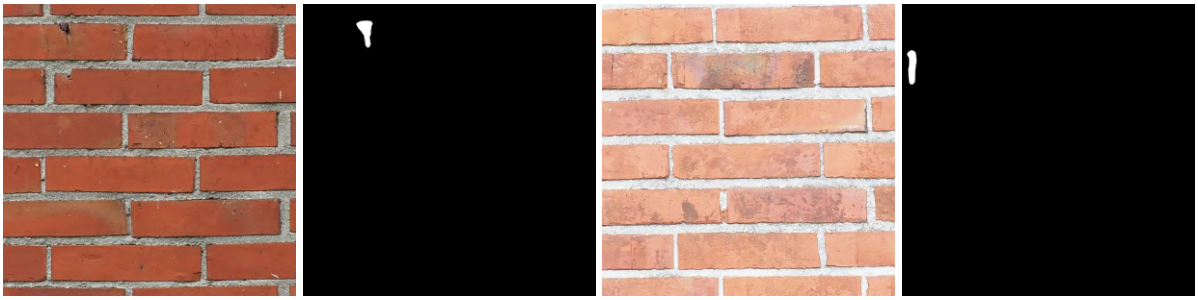
Visual evaluation to compare each of these variants of models has also been done on the test set. Visually it has been found that the base model and the model with zooming augmentation comparatively show little difference in prediction output. More interestingly, the model trained with the data pruning strategy predicts notably less small blobs of pixels. The exclusion of otherwise predicted small segments is illustrated for two examples in figure 4.6. This effect is as expected and results in more desired prediction outputs. Since the tile zooming augmentation does not show visual significance, the corresponding model is omitted. The data pruning has visually found to be effective on some images, so the yielded model is considered as the final crack segmentation model.

4.3. Segmentation workflow

The major motivation for the design of the segmentation workflow is the property of variance in both translations and scales of the crack segmentation network. Additionally, the workflow is designed to infer at an appropriate level of scale, preferably only segmenting significant cracks and excluding any small-scale features. The level of scale which is utilized during the creation of the tiles is determined by visual inspection on its output on a limited set of quay images. Another driving factor on determining

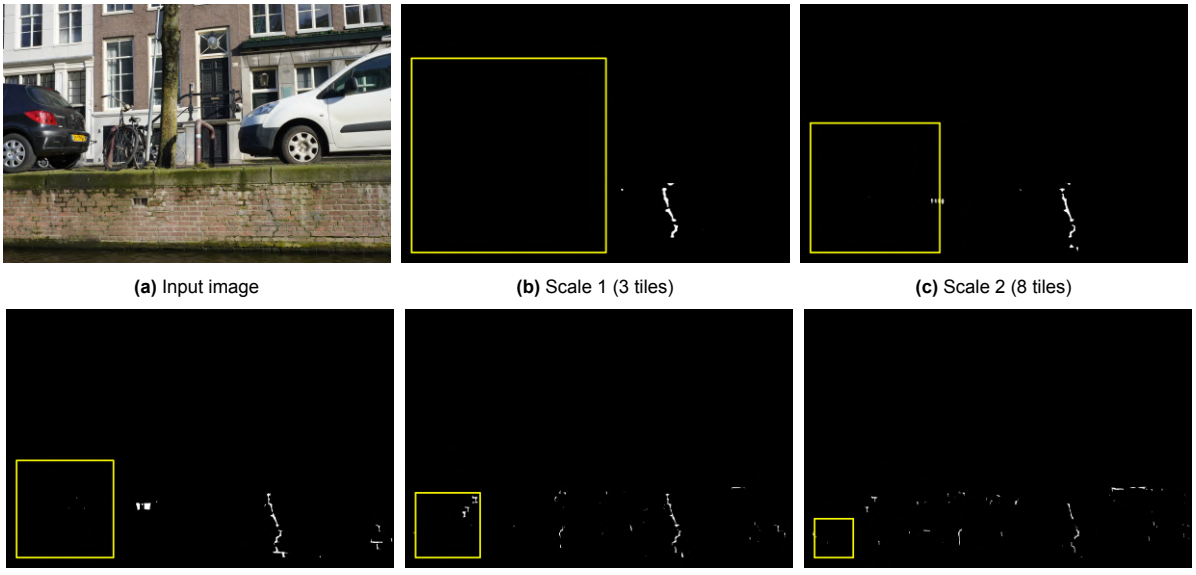
Model	Description	Epoch*	Precision [%]	Recall [%]	F1-score [%]	IoU [%]
base	FPN - ResNeSt-50 + GDice loss	49 $\pm$ 8	82.9 $\pm$ 0.8	76.1 $\pm$ 2.4	78.8 $\pm$ 1.0	66.4 $\pm$ 1.2
Tile zoom	Data augmentation step: random zoom-out of tile (at max 300%)	46 $\pm$ 2	78.9 $\pm$ 1.7	73.0 $\pm$ 2.0	75.4 $\pm$ 1.0	61.6 $\pm$ 1.3
Data pruning	Data augmentation step: Filtering of small label blobs of pixels	19 $\pm$ 4	75.7 $\pm$ 2.4	76.4 $\pm$ 1.3	75.1 $\pm$ 0.7	61.1 $\pm$ 0.9
Combined	Tile zooming + data pruning	26 $\pm$ 3	80.5 $\pm$ 0.3	72.1 $\pm$ 1.1	75.2 $\pm$ 0.8	61.7 $\pm$ 1.5

Table 4.4: Test set metrics of the FPN - ResNeSt-50 architecture trained with the GDice loss function together with test metrics of this same model with additional tile zooming and label pruning performed. The tile zoom is done on random scales each time and at most 300% zoomed out compared to the original scale. The data pruning filters out blobs of labels smaller than a certain threshold of amount of pixels relative to total image size.



(a) Example test image 1 **(b)** Base prediction 1 **(c)** Example test image 2 **(d)** Base prediction 2

Figure 4.6: Two example tiles from the test set and corresponding predictions made by the base neural network. These two examples are especially interesting because for these images the predicted segments are excluded on the data-pruned neural network. This explains how the data-pruned neural network is not taught to segment small segments.



(a) Input image **(b)** Scale 1 (3 tiles) **(c)** Scale 2 (8 tiles)

(d) Scale 3 (12 tiles) **(e)** Scale 4 (24 tiles) **(f)** Scale 5 (56 tiles)

Figure 4.7: A depiction of several intermediate segmentation workflow predictions on different level of scale regarding tile size for input on the crack neural network. The chosen tile size during construction of this inference grid is relative to image size and roughly at equal size increments between them. The yellow box indicates the scale of the input tiles. A higher scale number relates to a more zoomed-in tile selection, yielding to more tiles being created.

a proper value for these three scales is the performance overhead for creating the additional tiles.

In general, tiles created at a smaller spatial scale yield more fine-detailed features, whereas similarly larger tiles are likely to exclude these. The reason for this is that the larger tiles are down-scaled more to a 512x512 pixel tile, which consequently removes some level of detail. This effect is illustrated in figure 4.7 for several different scales of inference. Since we want to omit most of these fine-details, three levels of scale have been chosen which down-scale the original image considerably. These three scales correspond to scales 1, 2 and 3 in figure 4.7, where the size of the tile as input to the neural network after scaling is indicated with a yellow box. The intermediate predictions yielded on each of the three different scales are aggregated and then thresholded, such that pixels positively predicted in at least 2 out of 3 images are returned as the final output.

The influence of the chosen overlap between tiles is found to be less compared to that of the level of scale. Some experimenting has found an overlap of 128 pixels, for the tiles of 512x512 pixels, to work well. Higher levels of overlap rapidly blows up the amount of tiles created with negligible improvement in prediction.

4.3.1. Evaluation

The evaluation of the segmentation workflow is predominately done by manual visual inspection, because of the limitation of the dataset being unlabelled. To still provide structure for this assignment, a table to classify any segmentation output on is designed at first. These classification categories are listed in table 4.5, and can best be regarded as the confusion matrix with two additional subcategories for true positive. These two subcategories indicate the degree of success in segmenting the crack. This degree ranges from the crack being mostly segmented as a whole (category 1), about half of the crack pixels being segmented (category 2) and less than half of the crack pixels being segmented (category 3). The remaining three categories are the default confusion matrix items. The classification is done per image, however, one image can of course contain false positives (category 5) independent of how well any cracks present in the image are segmented, resulting in some images being classified into two categories.

This evaluation table is applied on a manually selected test dataset of 60 quay wall images, with 30 images showing a crack and the remaining 30 images without any crack. These images are the direct prediction of the crack detection model on the images, so no aggregation of multiple overlapping images

Label	Description	Sub-categories
True positive	Correctly finds one or more crack(s)	1. The crack(s) are mostly correctly segmented
		2. The crack(s) are found but about half of the crack pixels is segmented as background
		3. The crack(s) are found but more than half of the crack pixels is segmented as background
True negative	Correctly segments background	4. Whole image is correctly segmented as background
False positive	Incorrectly segments background as crack	5. At least a small amount of pixels that should be background are segmented as crack
False negative	The crack is (mostly) missed	6. Almost all crack pixels are mostly or completely segmented as background

Table 4.5: Evaluation categories as used to visually classify prediction output from the segmentation workflow. These categories serve as a manual evaluation protocol to assess segmentation workflow predictions on an image level and can be regarded as an extension of the confusion matrix.

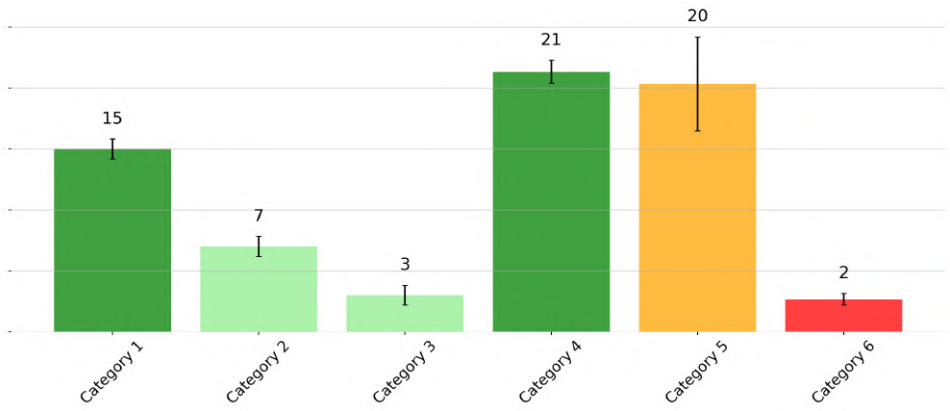


Figure 4.8: A summary of the categorized segmentation workflow predictions ran on the test-set of 60 quay wall images and manually classified in the predefined categories. The corresponding categories are listed in table 4.5 and act as a small extension to the confusion matrix categories. The classification evaluation is done by three users independently and the following mean and standard deviation is depicted per category.

to strengthen the prediction is done for this specific experiment. To remove some of the human bias, a user study is done with three users who independently assess the prediction results based on the evaluation table. The predictions resulting from the designed segmentation workflow for each of these images are subsequently categorized, and this result is summarized in figure 4.8. The frequency of occurrence for each category is shown by their mean and standard deviation. Although this cannot be directly concluded from the bar plot itself, it can be analysed from the user study that in many cases any of the true positives classifications is often combined with category 5. The categories are colored by severity and it can be seen that most predictions are satisfactory (category 1 and 4).

Most interestingly, we also show four wrong predictions from category 6 and 5 respectively in figure 4.9. This figure depicts two cracks which, unusually, are not dark but instead rather bright, which is likely to be the reason of it being left out. Additionally is shown how a chain and a shadow from a pole are misinterpreted as crack, which has been found to be an issue in a few other images as well. Furthermore, the frequency of false positive errors (category 5) is notably high as can be seen from figure 4.8. Often some small blobs of background pixels are predicted as crack, even in cases where the crack is found (true positive).

(a) Bright-colored crack missed by workflow **(b)** Bright-colored crack missed by workflow **(c)** Shadow wrongly predicted as crack **(d)** Chain wrongly predicted as crack

Figure 4.9: Four different patches of quay wall after inference of the segmentation workflow showing faulty results. Both (a) and (b) show a crack being brightly colored, contrasting with its typically dark appearance, which is likely to confuse the prediction. The opposite effect is noticeable for patches (c) and (d) showing a shadow of a pole and a chain respectively being predicted as a crack.

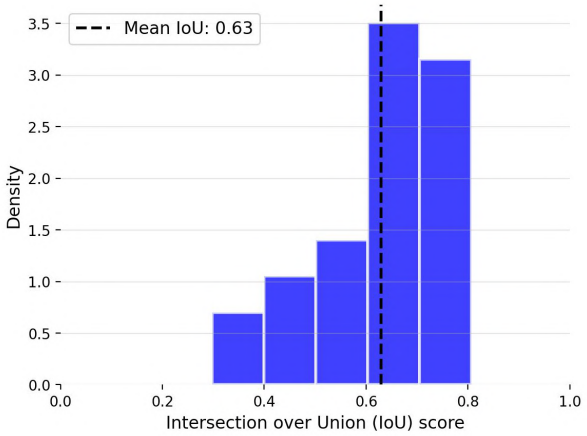


Figure 4.10: A histogram showing the distribution of Intersection over Union (IoU) values as derived from the segmentation predictions on 30 pairs of overlapping quay wall images.

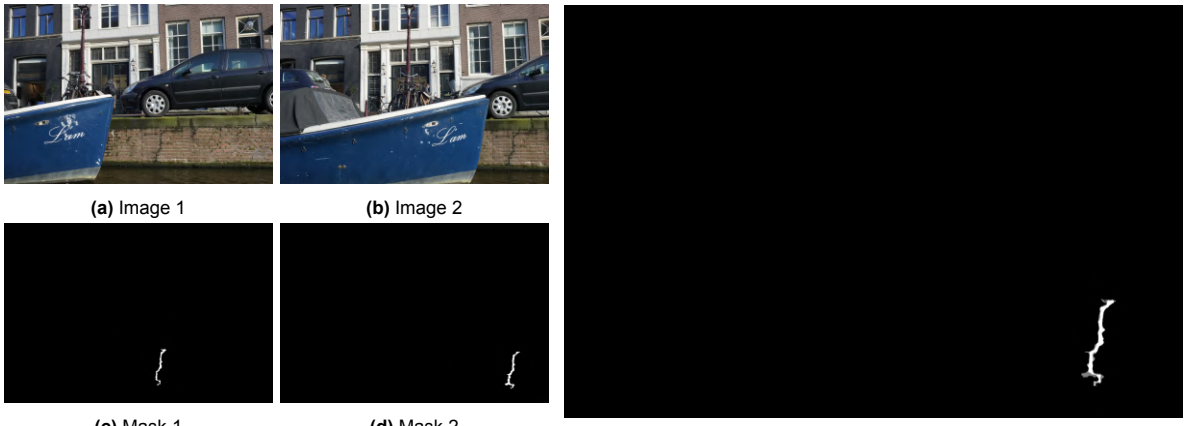
Prediction consistency in overlapping images

From each of the 30 images with the presence of a crack in the evaluation test-set discussed above, the subsequent image from the acquisition is utilized. Together this combination results in 30 image pairs of overlapping quay wall images containing a crack. For each of such pair, the corresponding predictions are matched and registered together to result in an aggregated combination. An example result of this affine transformation on the prediction masks for a pair of images is depicted in figure 4.11. The registration is done with an affine transformation on the corresponding images to project the images on top of each other. Next, the overlap between these two predictions is calculated. A depiction of the distribution of these intersection over union (IoU) metrics over all 30 image pairs is shown in figure 4.10. During this method, the calculation is only done on the part of the images which is visible on both images. From the evaluated images, an IoU score of 80% seems to be the upper-bound, whereas the mean is at a value of 63%.

This high degree of overlap within the prediction masks of overlapping images gives us confidence that the segmentation workflow has good consistency in output. Because of this property, overlapping prediction masks can be combined to a weighted final result. This method will strengthen the confidence of the prediction. Notably, aggregated results have not been used for the evaluation user study as discussed earlier.

4.4. Crack characteristics post-processing

Before discussing the results of the proposed crack length and crack width estimation methods, we shed light on the underlying assumption during the fitting of a plane by means of RANSAC in these



(a) Image 1 **(b)** Image 2 **(c)** Mask 1 **(d)** Mask 2 **(e)** Combined masks

Figure 4.11: An example pair of images (a, b) and corresponding masks (c, d) which has been calculated for their combined intersection over union (IoU) score (e). The calculation is done by means of an affine transformation based on the detected SIFT points to minimize re-projection error. In this case the IoU score is 78.5%.

methods. This plane is fitted through the detected SIFT points lying around the predicted crack mask, and therefore it is assumed that these points all lie on one single plane. While this is the case for most sections of quay wall, it is not for cases with the presence of extra objects or a bent quay wall section, as illustrated in figure 4.12. In this work we exclude steps to mitigate this limitation.

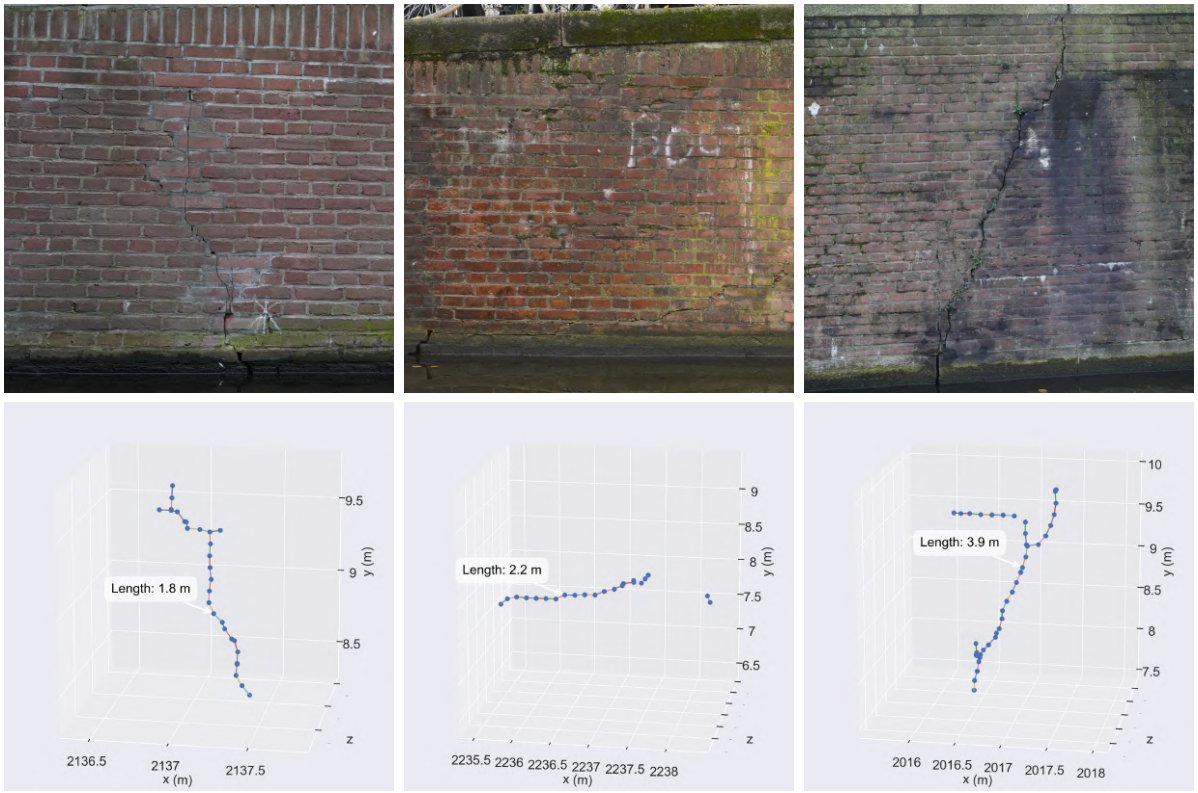
Figure 4.12: Two example images of quay walls demonstrate scenarios where the RANSAC plane fitting for forward intersection of the present crack will not work properly. This plane fitting assumes a completely flat quay wall; however, the left image depicts a pillar on the quay wall, while the right image has the crack exactly on the border where the wall bends. To aid visual inspection, yellow boxes around the cracks have been drawn. As a result, the forward intersection of found SIFT points will be mapped to faulty 3D positions.

Crack length estimation

The alternative representation of connected 3D points resulted from the skeleton of the predicted mask is found to be a good fit for most evaluated cracks. This 3D representation is shown for three different predicted cracks in figure 4.13. This reduction to representing a crack into a 3D graph gives extra need for good connectivity in the crack prediction masks. Disjoint segments of a crack logically result in disjoint segments into this 3D representation as well, which is undesired if the segments do belong to the same crack. Since we only report the final length over all the segments combined, for the estimation of crack length, this will not propagate to the final value of the length.

In this process, there is some freedom of choice regarding the spatial frequency of points being placed on the mask's skeleton. Opting for less points results in a more simplified 3D representation, which might be preferred for some use-cases. Conversely, a denser distribution might be desired for a more accurate length estimation and a more detailed view. It is important to note that there is neglectable performance overhead for selecting more graph points.

In absence of any ground-truth measurements, we rely on visual assessment on the accuracy of the crack length estimations as done on a limited selection of images. For one, this is done on a rough scale by counting the bricks in the images and furthermore knowing that a brick has dimensions of around 20 by 10 centimeters. Additionally this is done on the uniformly sized orthographic projections made on these images, which is further discussed in the next section on crack width estimation. In these images, the pixel dimension is 1 by 1 mm and the crack length can therefore be counted on pixel-level and related in units of meters. We evaluate that the length estimation is sufficiently accurate.



(a) An input image and 3D representation **(b)** An input image and 3D representation **(c)** An input image and 3D representation

Figure 4.13: An illustration of three different cracks processed by the segmentation workflow and mapped to its 3D representation by means of forward intersection.

Crack width estimation

The same three cracks as depicted for the determination of crack length (figure 4.13), have also been studied on their width estimation. The results are presented in figure 4.14, where the pixels on the predicted mask is colored relative to its width value. With respect to their accuracy, the left and right cracks depict two large clusters of pixels which are subsequently colored red. In both cases it actually does not contain a wide crack segment, but instead both parts show a crack split into two diverging sub-parts. The segmentation model is not predicting this in fine detail but instead wrongly shows one single cluster. In this regard, the prediction mask does not always correspond to the true appearance of the crack. Furthermore, it becomes clear that the predicted mask is often a few pixels wider than the actual crack segment. Because of this level of noise, we conclude that the estimated width cannot be determined at high precision. Instead we focus our efforts on distinguishing crack width by categorizing segmentation predictions on different orders of magnitudes.

This categorization on crack width for two example cracks with notable difference in width is shown in figure 4.15. Both images of cracks are similarly processed by the segmentation workflow and depicted with the same color-scheme. At last, width values for each pixel on its skeleton are aggregated to a final histogram depiction, showing the frequency of occurrence per width value. It can be noted that the fine crack clusters around a width value of 15 millimeters, whereas the larger crack has most width values at roughly 40 mm millimeters. For such an extreme case, comparing a fine crack and a very severe

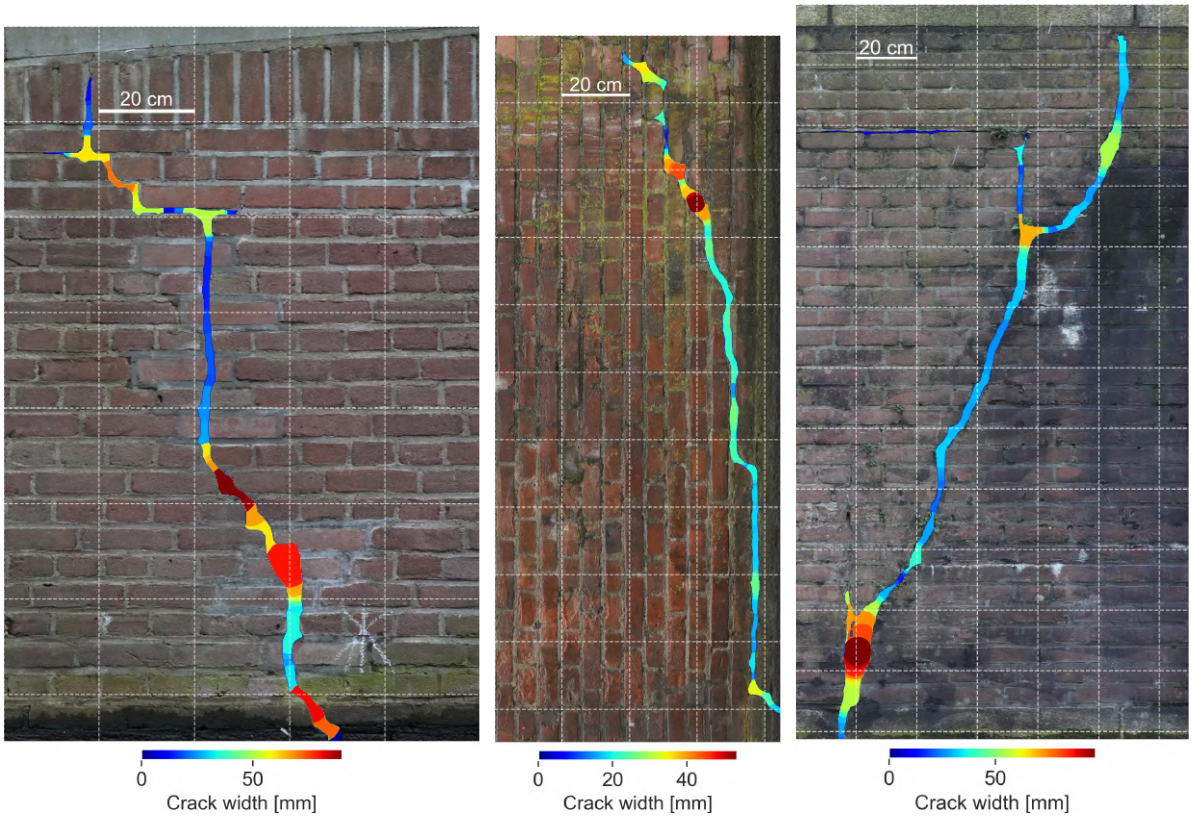


Figure 4.14: Depictions on the width estimations for each of the three cracks, as also been shown in figure 4.13 for their length estimation. The crack depicted in the middle has been rotated 90 degrees for better visualisation. The width is derived from the distances for each pixel on its skeleton to the edge of the prediction mask as colored by this metric.

one, we have therefore shown separability in classified crack widths from the reduced representation of a histogram. Additional experimenting on different cracks will have to show the ability to possibly extent this with more categories.

Compared to the task of crack length estimation, the method of width estimation is a much more precise task where the order of magnitude is millimeters instead of meters. The obtained level of precision mainly depends on the pixel-wise precision of the segmentation workflow predictions as well as the precision from the photogrammetric post-processing steps. It is outside the scope of this work to determine any statistics of both inaccuracies combined but since the photogrammetric estimation is precise enough for deformation measurements, we can expect sufficient precision regarding this aspect for the determination of crack width as well. The prediction masks however are quite coarsely detailed spatially and often several pixels wider than the actual mask, as we have seen above.

Next we quantify the precision of the photogrammetric post-processing steps, which includes SIFT matching, plane fitting, and forward intersection, in the existing photogrammetric imagery set. The estimation of crack width has been performed on seven independent crack predictions as resulted from the segmentation workflow. Over all runs, the detected and matched SIFT points have found to have an average distance to the RANSAC-inferred plane of 0.56 millimeters. This low distance gives us confidence that the corresponding photogrammetric measurements are of sufficiently high precision.

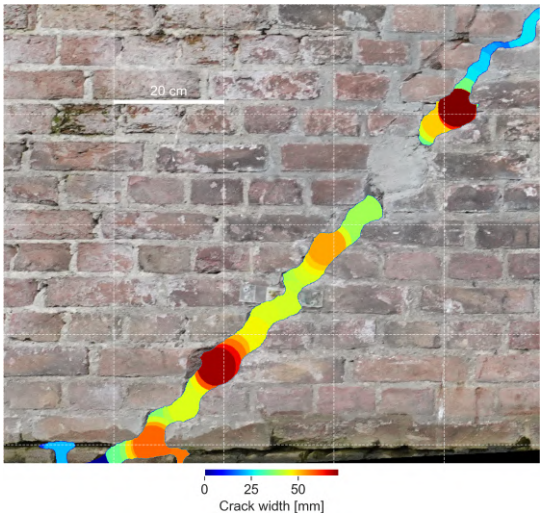
4.5. Case studies

We end the chapter by finally depicting two interesting case studies of predicted cracks derived from the segmentation workflow. Both case studies are particularly worthy of discussion due to their (potential) relation to measured deformation. Both case studies are depicted as an elongated orthographic view on a section of a quay wall and the cracks as predicted from the workflow.

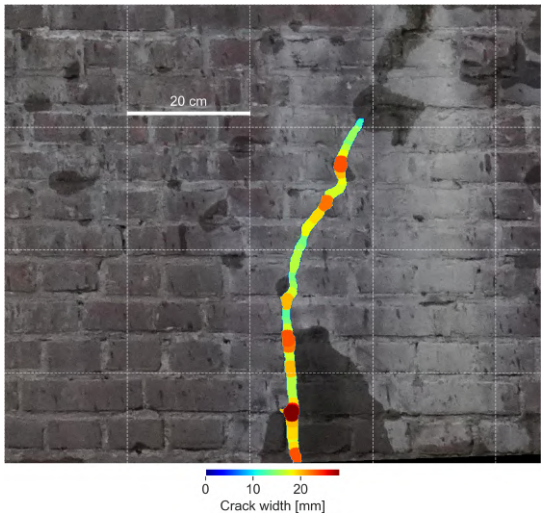
The first study case is shown in figure 4.16, showing several significant cracks, as rightfully predicted

(a) Scaled ortho photo of severe crack

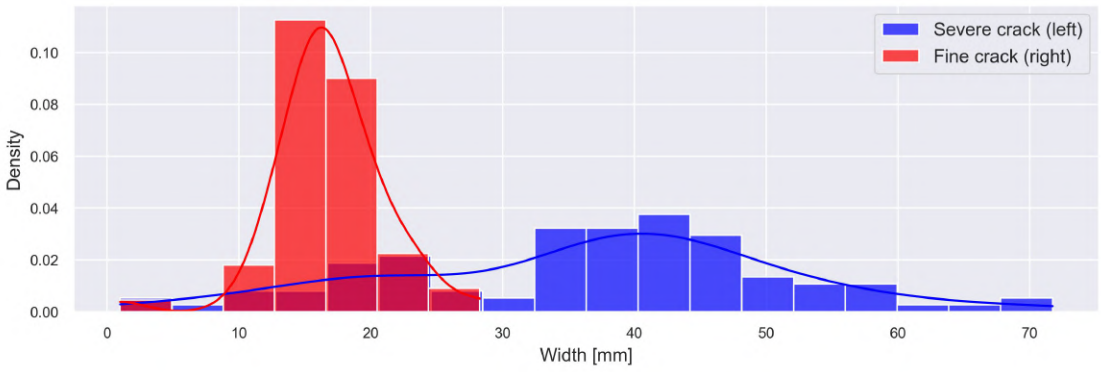
(b) Scaled ortho photo of fine crack



(c) Width estimation along crack



(d) Width estimation along crack



(e) Histogram depiction on the density of widths as calculated along the length of the skeleton of both cracks. The histogram colored blue corresponds to the severe crack depicted left, while the red histogram related to the one shown at the right.

Figure 4.15: A comparison of two cracks with comparatively significant differences in width, as can also be derived from the corresponding density plot shown as a histogram (e). The orthographic projections, scaled to true scale, of both cracks is depicted in (a) and (b) respectively. In these projections, each pixel has an estimated size of 1x1 mm. The corresponding predictions, mapped to this same projection, is shown in (c) and (d). This visualisation colors each pixel relative to the closest distance to the edge of the binary mask. At last, the corresponding width values for both predicted masks along its skeleton is shown in the histogram (e). The most significant peaks for each crack, located between 5-10 mm and 15-20 mm respectively, is at separate and distinguishable width values.

by the segmentation workflow. This section of the quay wall is especially interesting because there is also deformation measured at this site, making the presence of such a significant crack a key indicator of damage.

The second study case is shown in figure 4.17, likewise showing notable cracks. These two vertical cracks being close to each other might be a possible indicator of underlying deformation. It could be that forces in between them acting upon the wall cause the wall to bend slightly, which could explain the crack formation.

Figure 4.16: Case study 1: subsequent images stitched together to an orthographic projection combined with predicted masks resulting from the segmentation workflow. Along the length of this section of the quay wall, deformation has been measured, making it the presence of these significant cracks especially interesting.

Figure 4.17: Case study 2: subsequent images stitched together to an orthographic projection combined with predicted masks resulting from the segmentation workflow. The presence of these two significant vertical cracks might hint to some deformation or other forces acting upon the quay segment in between.

5 Discussion

In this chapter, we first answer the research questions (section 5.1) based on the results presented in chapter 4. We will go through each sub-question and relate to the corresponding results to finally conclude by discussing the main research question. Next, the limitations and shortcomings of this thesis are critically discussed in section 5.2.

5.1. Answering the research questions

In this thesis, we formulated the following research questions:

> **Sub-question 1:** What fully-supervised learning approach demonstrates to be effective for both quay wall and crack segmentation?

In this work, we have explored various training methodologies which yield effective learning capabilities for both tasks of (quay) wall and crack segmentation learning. Regarding (quay) wall learning, transfer learning has found to be effective by leveraging an existing resourceful data source. This allowed the models to first learn the representation of masonry structures in general and from there fine tune on masonry quay walls specifically. To put it in a different way, utilizing this rich and extensive data source allowed us to yield satisfactory learning results by only annotating a relatively small dataset of quay walls ourselves. In addition, we have found very little performance difference between different neural network architectures, each achieving high levels of effectiveness.

We have opted for the DeepLabV3+ architecture as it has proved to provide the most consistent results over a wide variety of imagery. We also demonstrated how this network design comparatively results in a more coarse prediction, where more details are excluded. Initially we assumed that this is satisfactory in practice since we thought that the subsequent crack segmentation network can handle this appropriately. In the results of the segmentation workflow however, we have evaluated that the predictions are sometimes confused by non-masonry structures like signs, chains and ropes. Filtering out such image segments by the quay wall network, serving as additional pre-processing constraint, could remove these effects in the results. A possible solution to achieve this is to deploy any of the other trained quay wall segmentation models as these have found to produce more detailed masks. To exclude these parts even better, the quay wall data annotations should be revisited to exclude these parts in the annotation labels. In this case the neural networks should be re-trained on this new data as well.

The task of designing the crack segmentation networks was found to be more thoughtful. Interestingly, the same architectures yielded lower learning abilities as evaluated on the validation and test set compared to quay wall learning, telling us that the dataset is inherently more complex. We put special effort into balancing the predictive output of such networks by tuning the loss functions parameters. We have provided methodologies to steer neural networks during training into favoring either false positive or false negative errors, to be used for trade-off considerations.

Furthermore a method has been adopted to filter out small data annotations of cracks as pre-processing

step during the training phase. This is done with the aim of teaching the crack neural network into focusing on significant larger cracks only. The data pruning approach has its limitations due to the subsequent difficulties of choosing a proper threshold value on pixel blob size. This is estimated by means of visual inspection and fixed for the complete dataset. It can be expected that a more flexible threshold will perform more effectively.

> **Sub-question 2:** How can the neural networks, once trained, be integrated into an algorithmic workflow considering the photogrammetric context?

We have proposed a methodology regarding the inference of crack segmentation within an structured workflow, where quay wall segmentation is a key pre-processing step. Driving factors on the design of this workflow were the requirement to infer cracks on a high resolution and to mitigate some of the effects of scale and translation variance which can be expected by convolutional neural networks. For these reasons, we opted for a sliding window approach where in a grid structure several overlapping tiles are created and afterwards served as intermediate input for the subsequent segmentation network. The slight overlap helps by considering different translations and minimises possible noise around at the borders. This sliding window approach is extended to multiple scale and can be a key consideration on the desired level of detail. The workflow's modular design ensures that networks and hyperparameters can be easily configured and switched.

The intrinsic photogrammetric property of overlap in the imagery set has found to be very powerful to elevate prediction results. We have found to be able to register such images and corresponding prediction output by means of an affine projection in a least-squares manner. This property adds more confidence to predicted output when predicted on multiple overlapping images. On a selected evaluation test set, we have found a mean overlap of 63% between the predictions of subsequent images. In this regard, we have opted to perform majority voting on the three overlapping images and aggregate the corresponding predictions thresholded to be visible in two out of three in total. From the same evaluation test set we have seen problematic cases of faulty segmentation predictions. Most notably, brightly coloured cracks are often missed and chains and ropes are wrongly predicted as crack. These findings should be taken into consideration for any future development.

> **Sub-question 3:** How can the intrinsic properties of photogrammetry be leveraged to determine crack characteristics, such as length and width, in an algorithmic way?

Lastly, we have shown how we can utilize the known positions of acquired images, as estimated by photogrammetry, present in the the imagery to further analyse prediction results. We have provided methodologies to convert crack predictions from pixel-level to true-scale units of meters and millimeters. This is especially important for the derivation of both crack length and width, for which pixel is a rather poor unit. Instead, we were able to map the predicted binary mask to a 3D graph representation, from which the length in meters can naturally be derived. Here the assumption is made that the crack image points all lie on the same plane, which is estimated in a least squares approach. A 3D graph representation, having nodes and edges, has found to be a proper representation for cracks.

For the derivation of crack width, an image where one pixel has a dimension of 1 by 1 millimeter is constructed as a first step. On this true-to-scale orthographic image, the width is calculated for sampled points along the crack by calculating the length of the perpendicular vector to the crack direction. A comparison between a severe crack and a hairline crack has found good separability between the two based on the distribution of width values from this method of calculation. Relating this result with literature, which details a common approach by categorizing severity of a crack by means of six classifications on their widths [24, 4]. These values range from sub millimeter for negligible cracks to 15 - 25 millimeters for severe cracks and beyond 25 millimeters for very severe ones. Although these studies are done on masonry structures in general, we assume roughly the same boundaries also apply for masonry quay walls specially. Notably, our method of calculation does not have sufficient precision to be able to classify according to these six classifications. The first category of cracks having a width smaller than one millimeters is not possible to distinguish because we create an image having a pixel size of 1 by 1 millimeter. Further classifications are difficult because of the spatial property of the prediction masks being rather large and coarsely detailed.

Further work will have to indicate to what degree classifications can be made with a certain level of confidence. Although our proposed method of width calculation is a lot less precise compared to man-

ual inspection, it does have the benefit of calculating the width on a lot more points along the crack. Comparatively, manual inspection is only feasible at at most a few points along the crack. Manual inspection can also benefit from the true-to-scale orthographic images which are constructed for the process. From these images, the crack and surrounding area can be visually assessed. This is especially interesting to do over several epochs.

Having covered the three sub-questions, let us finally revisit the main research question that this work aims to answer:

> **Main research question:** How can we localize and analyse cracks in masonry quay walls using fully-supervised deep learning methods together with photogrammetric image measurements.

This work has provided methodologies which together form an initial engineering effort for the goal of localizing cracks in masonry quay walls. Since no earlier work has addressed this goal within the same context and data sources, direct comparison with previous research is not applicable. The fundamental neural networks have shown to be very flexible when handling complex imagery data. During the design of these models a lot of considerations came into play in order to make them perform to the best desire. Their subsequent deployment in the segmentation workflow further introduces engineering choices when designing the most effective approach. Additional research on the incorporation of the photogrammetric properties and information on the otherwise neural network based methods results in an important bridge between the two principles. Photogrammetry based techniques are discussed to increase prediction confidence as well as to upgrade the pixel-wise predictions to true-to-scale units of meters and millimeters. This is an important research and engineering finding compared to related techniques solely using deep learning based segmentation.

5.2. Limitations of our approach

To conclude the discussion on a critical note, we highlight the most important limitations of our proposed methodologies and research. Supplementary on this, future research directions are discussed in the next chapter as well (section 6.2), although these have been formulated with a broader scope of consideration in mind.

5.2.1. Revision of segmentation workflow

Although the segmentation workflow fulfills the initial aim of processing large images at a fine level of detail, two main shortcomings can be detected. First, deciding on the most appropriate level of scales for generalization across a large set of imagery has been challenging. Limiting this to a fixed set of scales will highly constrain the generalization since effectiveness varies significantly from one image to another. The reason for this limitation is that the number of pixels and the real-world spatial scale of pixels are uncorrelated. In contrast, basing these scale values on some kind of information retrieved from the image itself will enable better estimations of spatial scale. For example, detecting and counting the number of bricks in the image could estimate spatial scale.

The second limitation of the segmentation workflow results from the inherent sliding window approach. Because the large input image is divided into smaller sub-patches, the receptive field of the overall inference will always be local to the extent of the tile size used. A prediction made on a certain tile will be independent of the other predictions surrounding this tile. In practice the presence of a crack is very much dependent on the surrounding area, due to the continuity along the crack. However, in the current segmentation workflow design, this additional knowledge is not incorporated.

Precision of crack width processing

In many cases, the prediction masks generated tend to be larger than the actual dimensions of the cracks. While this inconsistency has a negligible effect on crack length calculations, it significantly impacts crack width estimations. Consequently, this limitation restricts us to distinguish cracks only on order of magnitude instead of actual crack width. More precise predictions with more level of detail will improve the analytical capabilities regarding crack width estimation. Exploring additional image post-processing steps, such as those based on contrast enhancement or edge detection, could present an interesting direction for further research.

Evaluation with limited domain knowledge

The assessment of the segmentation model prediction effectiveness is challenging, given its application within a highly specialized domain. The conducted user study has its limitations, firstly due to the small number of participants involved. Additionally, accurately determining what counts as a crack, and to what extent, proves to be difficult. This complexity introduces ambiguity into the evaluation process.

6

Conclusion

In this chapter we finally wrap up this thesis with a closing conclusion (section 6.1) as well as by suggesting recommendations for future work in (section 6.2).

6.1. Conclusions

Damage assessment on the masonry quay walls has proved to be vital for the city of Amsterdam. Many quays are deemed to be in poor condition and overdue maintenance has resulted in the challenge of maintaining over 200 kilometers of historic quay walls. This significantly impacts the city in many different ways and the recent adaptation of photogrammetry for the goal of deformation analysis has proved to be valuable. By means of photogrammetric processing, insightful deformation information can be extracted on a data source of imagery.

The goal of this thesis was to investigate potential leveraging of this existing photogrammetry imagery data source by means of automatically extracting visual cues of damage in these masonry quay walls. Predominately, crack formation is often related to deformation and therefore essential to monitor for a more resourceful quality assessment. Because of the inherent large scale of this problem, localizing cracks in an automated way is the only viable option and our learned segmentation methodologies have proven to achieve effective results. Not only do we enrich the photogrammetric pipeline by this additional image understanding, the same photogrammetric principles are subsequently able to leverage our learned crack predictions.

In an incremental and experimental approach, we researched several training configurations to achieve the highest effectiveness in both tasks of quay wall and crack segmentation (research question 1). Several existing neural network architectures are considered: FPN, MANet, DeepLabV3+ and LinkNet. Variations of training strategies are explored with several loss functions. Notably, transfer learning and predictive balancing have found to be key considerations for quay wall learning and crack learning respectively. We deployed the trained neural networks by proposing a segmentation workflow to infer crack predictions on the photogrammetric imagery (research question 2). The workflow embeds a sliding window approach on three different level of scales to mitigate the effects of scale and translational variance. On top on that, the photogrammetric measurements on the images have found to be useful to exploit the crack predictions further to a scaled projection. In this true-to-scale system, we can determine crack length in meters and coarsely categorise crack width in millimeters (research question 3). Additionally, the inherent overlap between images are utilized in order to put more confidence on the resulting segmentation outcomes.

6.2. Future work

We conclude this thesis with some recommendations for future work. We identify four different directions for future work, as listed in order of what we believe to be their relevance.

6.2.1. Acquisition of quay wall crack labels

Although powerful in its learning abilities, fully-supervised approaches as we have researched in this work are notoriously data-hungry techniques. It is therefore obvious that labelled annotations on the quay wall imagery will improve segmentation abilities as compared to this work. Not only will it be able to serve as training data on the actual data source of interesting, something which was missing in our work, it will also allow for more accurate (pixel-wise) evaluations. The availability of these data annotations on the dataset of interest will be much more depictive into classifying types of errors. This will improve model evaluation and more correctly represent its prediction capability.

Admittedly, the acquisition of such labels are resource and time intensive, especially taking into the account to necessary domain knowledge required. Therefore for the next three suggestions of future work we outline innovative engineering techniques, aiming to also enhance the predictive capabilities as compared to the findings of this work.

6.2.2. Photogrammetric pre-processing for standardization

While the ability of a neural network to generalize over a widely varied dataset is typically a desired property, we now sketch a possibility where we reject this thinking. We opt for the incorporation of an additional pre-processing step to significantly reduce the variety within the dataset. This pre-processing step is then applied before inference of the segmentation workflow. The projection mapping to a true-to-size orthographic representation, as we have seen for the crack characteristics post-processing will ensure scale uniformity and eliminate perspective distortion. The resulting orthographic image will have a pixel size of 1 by 1 millimeter. This yields a simplified data representation for any learned model to recognize patterns on. It is deemed that applying this transformation before inference will result in more consistent and predictable outputs. Additionally this uniformity likely eliminates the requirement to infer on multiple levels of scale. With similar learning efforts, this will therefore potentially result in a higher achieved performance. This can be especially significant in combination of additional labelled data, as we have discussed above.

6.2.3. Continuity-preserved crack segmentation

To build upon the loss functions we have considered for this work, it would be interesting to consider the workings of loss functions specially designed for the task of crack segmentation. More concretely, a loss function which discourages non-connected segments as proposed by Pantoja-Rosero et al. [40] would be worthwhile. Connectivity of the crack segmentation is a well desired property, especially taking into account the additional photogrammetric post-processing steps which are build upon the assumption of this being present.

6.2.4. Weakly-supervised learning of crack segmentation

In relation to the discussed future directions of data standardization, we next shed light on an alternative learning approach where this data uniformity might be an important requirement. We want to highlight the possibility to learn crack segmentation in a weakly-supervised way, as proposed by Konig et al. [26]. The data source of this weakly-supervised approach is a collection of classification tiles, compared to pixel-wise crack annotations. The classification denote the presence or absence of a crack in the image tile. In this way a learned crack classification network is used to generate pseudo-segmentations by means of exploiting a heatmap of interest on the image. This heatmap is derived by highlighting any pixels of interest during this crack classification, which consequently highlights the crack pixels on a coarse level.

Weakly-supervised segmentation is especially interesting in the context of time or resource constraints. The creation of further data annotations is a much more simplified task. In the case of weak supervision, tiles of images need to be annotated according to classification, stating the presence of a crack in the tile, compared to pixel-wise segmentation masks. Early experimenting as part of our work has shown undesired results with this approach, but we think that this method can work sufficiently well when combined with our proposed data standardization pre-processing step.

Bibliography

[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 1)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1. 2022.

[2] Gemeente Amsterdam. *Actieplan bruggen en kademuren 2023-2026*. 2022. URL: https://assets.amsterdam.nl/publish/pages/973509/actieplan_programma_bruggen_en_kademuren_2023-2026.pdf (visited on 11/17/2023).

[3] Javad Baqersad et al. "Photogrammetry and optical methods in structural dynamics – A review". In: *Mechanical Systems and Signal Processing* 86 (2017). Full-field, non-contact vibration measurement methods: comparisons and applications, pp. 17–34. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2016.02.011.

[4] John Burland, B. Broms, and V. Mello. "Behaviour of foundations and structures". In: *9th International Conference on Soil Mechanics and Foundation Engineering (Tokyo)* (1978).

[5] Abhishek Chaurasia and Eugenio Culurciello. "Linknet: Exploiting encoder representations for efficient semantic segmentation". In: *2017 IEEE visual communications and image processing (VCIP)*. IEEE. 2017, pp. 1–4.

[6] Liang-Chieh Chen et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[7] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[8] Dimitris Dais et al. "Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning". In: *Automation in Construction* 125 (2021), p. 103606.

[9] L Minh Dang et al. "Deep learning-based masonry crack segmentation and real-life crack length measurement". In: *Construction and Building Materials* 359 (2022), p. 129438.

[10] Ilse A. E. De Vent. "Structural damage in masonry: Developing diagnostic decision support". PhD thesis. 2011.

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236.

[12] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning.

[13] Tongle Fan et al. "MA-Net: A multi-scale attention network for liver and tumor segmentation". In: *IEEE Access* 8 (2020), pp. 179656–179665.

[14] Rahim Ghorbani, Fabio Matta, and Michael A Sutton. "Full-field deformation measurement and crack mapping on confined masonry walls using digital image correlation". In: *Experimental Mechanics* 55 (2015), pp. 227–243.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[16] Clayford T Grimm. "Masonry cracks: a review of the literature". In: *Masonry: Materials, design, construction, and maintenance* (1988), pp. 257–280.

[17] Jiuxiang Gu et al. "Recent advances in convolutional neural networks". In: *Pattern recognition* 77 (2018), pp. 354–377.

[18] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[19] Mart-Jan Hemel. “Amsterdam quays under pressure: Modelling and testing of historic canal walls”. PhD thesis. 2023. DOI: 10.4233/uuid:102edff8-8960-4633-830c-369aef8e279f.

[20] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[21] Bohao Huang et al. “Tiling and stitching segmentation output for remote sensing: Basic challenges and recommendations”. In: *arXiv preprint arXiv:1805.12219* (2018).

[22] Pavel Iakubovskii. *Segmentation Models Pytorch*. https://github.com/qubvel/segmentation_models.pytorch. 2019.

[23] Mohammadreza Iman, Hamid Reza Arabnia, and Khaled Rasheed. “A review of deep transfer learning and recent advancements”. In: *Technologies* 11.2 (2023), p. 40.

[24] Richard Kastner, Jamie Standing, and Oddvar Kjekstad. *Avoiding damage caused by soil-structure interaction: Lessons learnt from case histories*. Thomas Telford, 2003.

[25] Diederik P. Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Jacob König et al. “Weakly-supervised surface crack segmentation by generating pseudo-labels using localization with a classifier and thresholding”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.12 (2022), pp. 24083–24094.

[27] Marc Kruyswijk. *Staat van kademuren is nog slechter dan gedacht*. In Dutch. 2019. URL: https://www.parool.nl/nieuws/staat-van-kademuren-is-nog-slechter-dan-gedacht~b299b057 (visited on 11/20/2023).

[28] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[30] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. “Building skeleton models via 3-D medial surface axis thinning algorithms”. In: *CVGIP: Graphical Models and Image Processing* 56.6 (1994), pp. 462–478.

[31] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[32] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[33] Wilfried Linder. *Digital photogrammetry*. Vol. 1. Springer, 2009.

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[35] Dimitrios Loverdos and Vasilis Sarhosis. “Automatic image-based brick segmentation and crack detection of masonry walls using machine learning”. In: *Automation in Construction* 140 (2022), p. 104389.

[36] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[37] Rita Esposito Mandy Korff Mart-Jan Hemel. *Bezwijken Grimburgwal: Leerpunten voor het Amsterdamse areaal*. In Dutch. Delft University of Technology, 2021. URL: http://www.repository.tudelft.nl.

[38] Shervin Minaee et al. “Image Segmentation Using Deep Learning: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (2022), pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.

[39] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning Deconvolution Network for Semantic Segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[40] Bryan G Pantoja-Rosero et al. “TOPO-Loss for continuity-preserving crack detection using deep learning”. In: *Construction and Building Materials* 344 (2022), p. 128264.

[41] Stephen M. Pizer et al. “Contrast-limited adaptive histogram equalization: speed and effectiveness”. In: *[1990] Proceedings of the First Conference on Visualization in Biomedical Computing*. 1990, pp. 337–345. DOI: 10.1109/VBC.1990.109340.

[42] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

[43] Paul-Edouard Sarlin et al. “Superglue: Learning feature matching with graph neural networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4938–4947.

[44] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48.

[45] Deepak Soekhoe, Peter Van Der Putten, and Aske Plaat. “On the impact of data set size in transfer learning using deep neural networks”. In: *Advances in Intelligent Data Analysis XV: 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings 15*. Springer. 2016, pp. 50–60.

[46] E-K Stathopoulou and F Remondino. “Semantic photogrammetry–boosting image-based 3D reconstruction with semantic labeling”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2019), pp. 685–690.

[47] Carole H Sudre et al. “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations”. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*. Springer. 2017, pp. 240–248.

[48] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[49] Athanasios Voulodimos et al. “Deep Learning for Computer Vision: A Brief Review”. In: *Computational Intelligence and Neuroscience* 2018 (Feb. 2018). Ed. by Diego Andina. Publisher: Hindawi, p. 7068349. ISSN: 1687-5265. DOI: 10.1155/2018/7068349. URL: https://doi.org/10.1155/2018/7068349.

[50] Saining Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV].

[51] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. “Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation”. In: *IEEE Geoscience and Remote Sensing Magazine* (2020). DOI: 10.1109/MGRS.2019.2937630.

[52] Hang Zhang et al. “ResNeSt: Split-Attention Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2022, pp. 2736–2746.

[53] Ding-Xuan Zhou. “Universality of deep convolutional neural networks”. In: *Applied and computational harmonic analysis* 48.2 (2020), pp. 787–794.

[54] Fuzhen Zhuang et al. “A comprehensive survey on transfer learning”. In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76.