Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Identification of Asset Integrity Issues using Image Based Data Analysis and Deep Learning Methods

Master's Final Degree Project

Andrius Ambrutis

Project author

Prof. dr. Mayur Pal

Supervisor

Kaunas, 2023

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Identification of Asset Integrity Issues using Image Based Data Analysis and Deep Learning Methods

Master's Final Degree Project

Applied Mathematics (6211AX006)

Andrius Ambrutis

Project author

Prof. dr. Mayur Pal

Supervisor

Doc. dr. Mantas Landauskas

Reviewer

Kaunas, 2023

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Andrius Ambrutis

Identification of Asset Integrity Issues using Image Based Data Analysis and Deep Learning Methods

Declaration of academic integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Andrius Ambrutis

Confirmed electronically

Ambrutis Andrius. Paviršių trūkių identifikavimas, paremtas vaizdų analize ir giliojo mokymosi metodais. Magistro baigiamasis projektas vadovas prof. dr. Mayur Pal; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: mašininis mokymasis, sąsūkų neuroniniai tinklai, perkeliamasis mokymasis, duomenų analizė, betono įtrūkimų aptikimas, įtrūkimų ant plytų sienų aptikimas.

Kaunas, 2023. 55 p.

Santrauka

Betono įtrūkimai kelia didelį iššūkį konstrukcijų stabilumui, o kai kuriais atvejais remontas gali kainuoti brangiai arba net prireikti visiško atstatymo, kad pastatas nesugriūtų. Tačiau laiku aptikus ir pašalinus žalą galima išvengti daugelio nelaimingų atsitikimų. Deja, aptikti įtrūkimus gali būti sudėtinga, ypač sunkiai pasiekiamose arba pavojingose vietose.
Atominės elektrinės, povandeniniai vamzdynai ir užtvankos, be kitų įrenginių, reikalauja reguliarių išorinių ir vidinių konstrukcijų patikrinimų, kurie gali būti brangūs ir sudėtingi. Neseniai buvo pasiūlyti mašininiu mokymusi pagrįsti metodai turto saugai tikrinti naudojant dronus. Šiame darbe nagrinėjamas mašininio mokymosi modelių naudojimas betono įtrūkimams aptikti, nes tai gali supaprastinti procesą ir sumažinti reguliarių patikrinimų išlaidas. Tačiau pastebėjome, kad kai kuriais atvejais mašininio mokymosi modeliai veikia prasčiau arba gali būti apgauti dėl prastos kokybės duomenų arba duomenų su dideliu poslinkiu.
Šiame tiriamajame darbe siūlome vaizdo gerinimo metodus, kad pagerintume modelio prognozes. Manome, kad dėl mūsų siūlomų matematinių ir mašininio mokymosi algoritmų įtrūkimų aptikimas bus lengvesnis, saugesnis ir ekonomiškesnis atliekant reguliarius patikrinimus.

Ambrutis Andrius. Identification of Asset Integrity Issues using Image Based Data Analysis and Deep Learning Methods. Master's final project supervisor prof. dr. Mayur Pal; Kaunas University of Technology, Faculty of Mathematics and Natural Sciences faculty.

Field and fields of study (group of study fields): Applied mathematics (Mathematics).

Keywords: machine learning, convolutional neural networks, transfer learning, data analysis, concrete cracks detection, cracks detection on brick walls.

Kaunas, 2023. 55 p.

Summary

Concrete cracks pose a significant challenge to the stability of structures, and in some cases, repairs may be costly or even necessitate complete rebuilding to prevent building collapse. However, detecting and repairing damage in a timely manner can prevent many accidents. Unfortunately, detecting cracks can be difficult, particularly in hard-to-reach areas or dangerous locations.
Nuclear power plants, subsea pipelines, and dams, among other facilities, require regular external and internal structural checks, which can be expensive and challenging. Recently, machine learning based approaches have been proposed for asset integrity inspections using UAV. This paper explores the use of machine learning models to detect concrete cracks, which could simplify the process and reduce the cost of regular check-ups. However, we observed that in some cases machine learning models underperforms or can be deceived by poor quality data or data with significant displacement. In this research work, we propose image enhancement techniques to improve model predictions. We believe that our proposed mathematical and machine learning algorithms will make crack detection easier, safer, and more cost-effective for regular check-ups.

Content

List of tables

List of figures

List of abbreviations and terms

Abbreviations:

AI - Artificial intelligence;

ANN – Artificial neural networks;

CNN – Convolutional neural networks;

IR – Infrared radiation;

ML – Machine learning;

RGB – Red green and blue;

TL – Transfer learning.

Terms:

Artificial intelligence – a broad field of computer science that focuses on creating intelligent machines capable of performing tasks that typically require human intelligence such as performing data classifications, segmentation, and other.

Convolutional neural networks – use specialized layers, such as convolutional layers, pooling layers, and fully connected layers, to process images, gather features and can be used for classification and other images related tasks.

Machine learning – a technique when mathematical model is created by making computer to create model from data.

Transfer learning – a method, which is based on information re-using, instead of training a model from scratch on a new task, transfer learning allows us to utilize the knowledge learned from previous tasks, which can significantly speed up training and improve performance, especially when data is limited.

Introduction

The identification and repair of concrete cracks in structures is a critical task as it can prevent costly repairs or even the collapse of the building. However, cracks can occur in hard-to-reach areas, making it challenging to detect and repair them. Regular check-ups for structural changes are required in facilities such as nuclear plants, oil or gas pipelines, and dams, which can be costly and difficult due to their location or other factors such as radiation. By utilizing drones or other mechanical creations with automated crack detection algorithms, the complexity and cost of regular check-ups can be reduced. This could potentially lead to significant benefits in terms of efficiency, cost savings, and increased safety, assuming the algorithm used has a high level of accuracy in identifying cracks. This study aims to investigate the feasibility of using machine learning models to detect concrete cracks, with the potential to simplify the process and reduce the cost of regular check-ups.

Historically cracks and similar structural damage caused mortal disasters. In 1976 the dam started cracking this allowed water to breakdown the structure and flood the area, report states that 14 people died and accident caused damage worth around 1 billion dollars [1]. Another accident occurred in 1998, although it is only partially influenced by the cracks (as main reason for the formation of cracks were shift in the earth), this led to closing mines near the dam and damage was estimated to be close to 42.5 million dollars [1]. Pipelines are also no exception, according to the media and researches the Keystone pipeline got 3 splits (potentially caused by 'fatigue cracks') in approximately 5 years [2,3,4,5]. Same sources states that system is harmful to environment, and it is only a matter of time till new accident will occur. All of this can be prevented from repeating if damage on structures will be spotted in time and actions to fix effected areas will be made.

Our model has demonstrated high accuracy in identifying cracks on various wall types, including brick walls, using only an image of the wall. To further automate the process, we propose attaching a camera to a drone and programming it to follow the crack line to gather more data. Our model is not affected by the scale of the image or different colours, as it was trained on a diverse range of images, including those with dark shadows that are difficult for the human eye to interpret. Dataset also contained data with damaged areas from various angle which allows method to be more trustworthy in real life situations. Therefore, this method can be used in various environments, such as underwater or high places like skyscrapers. In addition, our approach is cost-effective compared to current methods, such as hiring workers or using large networks.

Our model uses a faster convolution neural network, saving computational power and energy, which in general should reduce the requirement of manpower. Stuart Russell [6], a renowned computer scientist, has made a thought-provoking statement on the topic of building conscious machines, stating that "No one has a clue how to build a conscious machine, at all." Although technology has been created to facilitate human work, it should not replace the role of human beings. It is important to note that even the most advanced models can be susceptible to errors or manipulations. Therefore, it is highly recommended to have specialists conduct a re-check to ensure the accuracy of predictions made by the algorithm and update database so that model can be re-trained over time. However, studies point out that even fastest neural networks can give accurate predictions.

Although, our primary focus is on Space Invariant Artificial Neural Networks (SIANN) commonly referred to as convolutional neural networks (CNN) and transfer learning (TL). CNNs are a type of neural network that has proven to be highly effective for tasks involving image and video analysis due to their ability to learn hierarchical representations of input data. Compared to fully connected networks, called Multilayer perceptrons, which have neurons in one layer connected to all neurons in the next layer, CNNs have sparse connectivity which helps in reducing the risk of overfitting. To further prevent overfitting, we can use regularization techniques such as penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.). Since CNN is based on matrix multiplication (which in our case are huge matrices as each matrix has to keep some information about the image) it is commonly used practice to use CNN with TL and this way make training faster, which in a way also gives better results as model already contains some information about similar features. Transfer learning is a machine learning technique that can improve the accuracy and speed of training a model. This technique involves using a pre-trained model that has already been trained on a large dataset and has learned to recognize common patterns and features. This pre-trained model is then used as a starting point for a new task, instead of training a model from scratch. By leveraging the pre-existing knowledge in the pre-trained model, transfer learning can significantly reduce the time and resources required to train a new model. When using transfer learning, the pre-trained model is first "frozen," meaning that its weights are fixed, and it is not trained any further [7,8,9]. Then, the last layers of the model are replaced or modified to suit the new task, and the model is trained on the new dataset with these modified layers. This allows the model to learn new patterns and features specific to the new task, while still retaining the general knowledge learned from the pre-trained model. This approach is particularly useful when the new dataset is small, or the new task is similar to the original task the pre-trained model was trained on. However, we will explain this process better in further chapters of this work.

While significant progress has been made in crack detection, many of the current algorithms are limited to certain conditions. For example, some models may only perform well on images with specific colour backgrounds (such as white walls), or under static lighting conditions without random factors like rain or snow. Additionally, to achieve high accuracy in classifying cracks, many studies rely on heavy and complex networks, which can be computationally expensive and impractical for real-life scenarios. These limitations can hinder the practical application of these methods, highlighting the need for more robust and adaptable algorithms for crack detection.

This work aims to address these challenges by utilizing transfer learning and a larger dataset to develop a more accurate and versatile crack detection model that can be applied in various settings while trying to make fast but reliable predictions. Our research shows how to comprehensively analyse the problem of crack detection from multiple perspectives.

Initially, thesis starts by employing a popular approach using simple grayscale images with a crack database to train and test models. We attempt to improve the performance of existing models by adjusting parameters and modifying layers for neural networks. Next, we address the challenge of dealing with images that have a wide range of colours and shaders, which can potentially confuse pre-trained models as colourful images have different pixel values. To overcome this challenge, our study explores the possibility of retraining models to accurately identify cracks from coloured images, which significantly increases the complexity of the task.

To enhance the results of crack detection in more intricate scenarios, our study broadened the image database by generating additional images through various methods. One of the methods entailed utilizing cracks as masks to create customized crack shapes and train the model to detect them. This approach enabled the model to be prepared for the potential challenges that it might encounter during inspections in diverse locations. Another method involved generating random crack-like patterns through an algorithm that was specifically developed for this purpose. Moreover, this study did not confine itself to the aforementioned tasks. We also observed the impact of noise on crack detection and endeavoured to enhance our model predictions by addressing the issue of noise and blurriness in the image. To accomplish this, we evaluated various techniques, including image resolution modifications using diverse algorithms and mathematical solutions. The renowned scientist R. May [10] famously stated, in reference to chaos theory, that "simple mathematical models may exhibit complex behaviour." In fact, it has been observed that a simple mathematical expression can effectively distinguish potential crack areas from noise, which in turn can significantly improve the predictions of the artificial neural network (ANN) model meanwhile complicated methods such as resolution increasing or decreasing using Convolution Neural Networks (CNN) or other similar approaches do not give good results.

In conclusion, this study aimed to conduct a testing case on complex structures by creating a pipeline imitation. In the process exhaustive database of images was created and the experimental designing approach was used to test for the best parameters. This modelling exercise provided an opportunity to assess how the developed model would perform at different angles, while also enabling the use of various images that could be applied to pipeline model structures with different shapes, accurately imitating actual pipelines like the Nord Stream or other pipelines widely used for gas and oil transportation. Consequently, this study can lay the foundation for future works, wherein the model could be updated and refined with pipelines data to enhance its prediction capabilities even further. Therefore, for this project we raised such goals:

- To create large and well-balanced database of concrete and brick wall images with and without cracks.
- Implement Transfer Learning (TL) to achieve results comparable with other research works.
- Test model and see if it can be used for various complex environments as it might be required for future development and applicability of the model.

This research distinguishes itself from other papers in the field through its unique approach to data generation and the use of a transfer learning-based model to detect cracks on a variety of surfaces, including concrete and brick walls, chimneys, pipelines, and even painted walls. This approach is more comprehensive and versatile than previous methods, which have often been limited to detecting cracks in only one type of surface or material. Additionally, the use of transfer learning allows for the model to be trained on a smaller dataset, saving time and resources, while still achieving high accuracy. While it is true that the dataset used in this study is larger than that used in other research works, the real advantage of transfer learning becomes evident when the model needs to be updated with a limited amount of data. In such cases, the pre-trained model can be fine-tuned with the limited data to achieve good results, instead of having to train a new model from scratch with a larger dataset. This is especially useful in real-world scenarios where acquiring a large dataset can be difficult and costly.

1. Literature review

The topic of crack detection has been extensively studied in the literature, with various mathematical and machine learning techniques proposed to determine whether images depict damaged structures or not. However, some of these models are more complex than others, and similar data is often used in most cases. In this chapter, we will analyse and categorize suggested crack detection algorithms based on their usage and complexity. It should be noted that no single algorithm can be considered the best, as some models may perform better on specific data, while others may yield better results in general.

1.1. Reasons for crack forming

Cracks can occur on buildings and pipelines due to various reasons. Some common causes of cracks on buildings are settlement, thermal movement, moisture, and overloading. This subchapter will try to give deeper understanding of those reason with examples and suggestions from various researchers. Settlement occurs when the soil under the foundation of a building is not properly compacted, leading to the foundation shifting and causing cracks in the structure. Thermal movement happens when the temperature of a building fluctuates, causing it to expand or contract, resulting in cracks. Moisture, such as water damage or leaks, can weaken the structure of a building and make it more susceptible to cracking. Overloading can also cause cracks, as it puts stress on the building beyond its designed capacity. In 1989 Buck [11] pointed out that cracks can form due to fatigue as materials can change their physical properties over time. In 2002 Yakovlev [12] identified several main reasons for crack formation in concrete structures. These include low concrete strength due to a lack of water for cement hydration, the absence of shrinkage seams in the floor structure, incorrect reinforcement, and excessive thickness of the strengthening mineral coating of the concrete floor, which can initiate crack formation. Ensuring that all previously mentioned problems are solved might reduce the risk of cracks forming on concrete walls.

Similarly, pipelines can also experience cracks due to various factors. The most common reasons for pipeline cracking are corrosion, abrasion, and stress. Corrosion occurs when the material of the pipeline degrades due to environmental factors, such as exposure to chemicals or moisture. Abrasion can occur when pipelines are exposed to rough surfaces or objects, leading to damage, and eventually cracking. Stress on pipelines can also cause cracking, as it puts pressure on the material beyond its intended limit. Multiple studies state that stress corrosion [13,14,15,16] is one of the main factors leading to cracks formation and eventual splits in pipes, this way expressing that stress and corrosions often comes together. According to the studies [15,16], cracking is mostly caused by the stress and corrosion of in effected areas only accelerates the effect. Furthermore, research points out that in oil and gas industry Hydrogen Induced Cracking (HIC) is one of the main reasons for splits in steel pipelines [17,18]. This is mainly the case because hydrogen mixtures can easily react with materials in the pipe and this way damage them over time. However, since our main zone of expertise do not cover stell alloys and hydrogen reactions we will not go into deeper investigation of how to avoid this reaction.

In previous chapter we also mentioned that accidents of dams cracking are major problem which can lead to disasters, in this subchapter we will also analyse few reasons why dams can start to crack. Zhang (in 2010) suggested that crack in dams can form because of difference in temperature and that controlling temperature might play key role in preventing dams from being damaged [19]. This is especially true during the construction of the structure. Zhang offers to apply multiple cooling steps during the construction process and this way to avoid cracks which can form due to temperature changes. Another cause of crack formations would be ground shifting (due to earthquakes, tectonic movement, unstable ground foundations) erosion and overloading [1,20,21,22].

Overall, taking into consideration what forces affect the investigated object it can be estimated for what type of damage model should expect. Understanding the reasons why cracks could form on the object we can understand which model to chose in order to increase the chance in detecting the crack.

1.2. Classification of crack types

As it was already pointed out before, based on reasons why concrete surface started to crack the shape of crack can be different, this can directly influence the complexity of classification. While machine learning methods can overcome this problem, mathematical approaches might face some difficulties. The type of the crack can also affect the cost and methods to fix it, in this subchapter we will cover researchers' insights about crack types and what information it tells us.

In 2018 Sitara [23] stated that crack detection and classification techniques with quantitative analysis have a huge role in finding the severity of crack and that various quantitative metrics are length, width and area. Based on features of the crack different methods can be applied in order to detect it. Sitara splits cracks into three main groups such as the minor cracks, moderate cracks, and severe cracks.

He states that minor cracks are common in underwater dams and bridges. According to Sitara some on these cracks can be under very small and require methods which can capture breaks smaller than a pixel. In most cases it is tiny barely visible breaks on concrete surfaces which are rarely curvy but in majority of cases looks like a simple cut.

Another group of described crack types is moderate cracks, which mostly appear on concrete roads and dams. Sitara states in his article [23] (referencing Shi [24]) that it is difficult to identify such cracks unless solar image is used. However, Shi in his article [24] was talking about sonar images assuming that such cracks can be hard to detect without sonar.

Finally, the last group is severe cracks. These cracks are most dangerous as the can cause the collapse of structures [25]. Here are many factors which can cause these breaks, a good example can be earthquakes or overloads (for bridges and buildings).

In addition, Sitara [23] also points out that cracks can be classified by the shape or complexity of it. Overall, all those groups can be assigned to already described 3 main classes.

1.3. Crack detection on concrete surfaces

In previous subchapter we learned that cracks can be categorised into classes [23].These classes may have slightly different features but are still highly related. For example, minor cracks may be much less visible compared to major cracks, but as the scale of the image changes, the differences between these two groups may become less apparent. The same logic applies to complex and simple cracks, as every complex crack can be divided into multiple simple cracks on different scales. Crack detection on concrete walls was well analysed by Lins [26] proposed a crack detection and measurement algorithm based on a mathematical approach in 2016. The suggested algorithm uses particles to detect the shape of the crack and simplify the problem, then measures the distance between particles and estimates if the shape is a crack based on the density in the area. However, this technique has an error range of 7.51% - 8.59%, which makes it less reliable compared to newer machine learning methods. For example, in 2019, Dung [27] analysed three CNN models by comparing them on open source concrete cracks dataset of 40 000 images. His research showed that VGG16 model [28] performed slightly better compared to InceptionV3 [29] and around 3 percent better in comparison to ResNet [30]. All mentioned models had prediction accuracy of over 96%. Dorafshan tries to challenge neural networks in 2018 by comparing Deep Convolution Neural Networks (DCNN) with Edge Detector models (ED) [31]. Results show that in some cases for concrete images dataset ED models can match the results of DCNN and even obtain accurate map of crack area. However, it is worth noticing that he had to try multiple algorithms to deal with the noise on each image as even small noisy areas can be considered as an edge. Yet, in DCNN and ED cases best results with models give around 98% accuracy of cracks detection. Liu in his paper [32] combined this idea in a way creating a CNN which tries to extract the mask of the crack from the image. According to him, even the model trained on 57 images can reach 90% prediction accuracy for different complex situations. While in general it is not hard to increase database for training set of concrete walls, it should be noticed that some inspiration from his U-Net model can be used to train different type of models on various wall types considering that some unique images can be hard-to-get. However, research done by our colleges in KTU shows that while crack detection works well under laboratory conditions it can be heavily effected but environment in real life scenarios. For example paper by Pal [33] shows that adding shadows to testing set can make model predictions worse by up to 50%. Research by Pal shows that models trained on simple datasets are not reliable in real-life situations. Taking this to consideration it is required to include more complicated (with colours, patterns, shadows, etc.) data to make models more trustworthy.

1.4. Crack detection on complex surfaces

In previous subchapter we already covered most well-known methods for crack detection. However, in many cases simple algorithms are not enough to perform detection especially on complicated surfaces such as brick walls, graffities, wall corners. Even changes in lightning can make detection task much more complicated and trick model into thinking that difference in colours is created by the crack. Yet, scientists have suggested several methods on how to search for damaged areas on the surfaces. Work by Hallee [34] suggests that for this task Random Forest would be most fitted algorithm (expect for neural networks) with an accuracy of over 86% while over methods give around 84% correct predictions on brick walls dataset. However, on dataset made in laboratory CNN was able to reach around 92% accuracy making it without a question the best performing algorithm in the paper. Yet, results on real life data were poor with huge drop in accuracy which makes model predictions questionable. On the other hand, Loverdos and Sarhosis [35] counters this research by stating that all machine learning architectures are giving similar results (between 95.98% to 96.87% validation-accuracy) with DLV3+ [36] model statistically showing best predictions. Size of training dataset could have huge impact to validation accuracy, since in Hallee’s research 598 images were used form training versus over 2000 pictures used by Loverdos.

1.5. Cracks underwater

Water is perhaps one of the most extensively investigated environments for crack detection, particularly in the context of oil and gas underwater pipelines. These pipelines require regular inspections to ensure that fuel is not leaking into the ocean, which can be a costly and risky process.

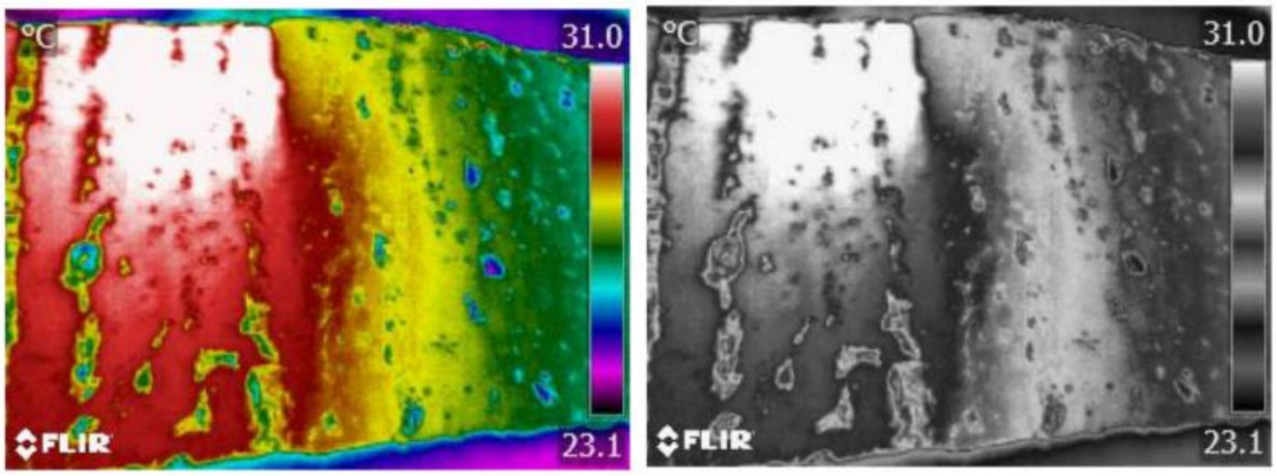
The cost can be high due to the need to hire experienced inspectors, who can earn over $150 000 per year according to some sources. In many cases, the inspection process can be simplified using modern technologies and techniques.

Scientists offered multiple methods for detecting damaged areas underwater. Starting from heat detection [37,38,39,40], followed by sonar and other sound-based methods [24,41,42], and finishing with simple optical image based algorithms [43,44,45,46]. However, all other mentioned research works are saving data as pictures (either it would be heat signatures, sound or light trace they all stored as images) this means that all methods are related to each other and only quality of information is main aspect to talk about as most researches use similar solutions (convolution neural networks (CNN), edge detectors and other machine learning (ML) algorithms) to solve this problem. On the other hand, gathering good quality data can be hard and might require extra modifications.

1.5.1. Heat based detection

In dark and noisy environments (in which case sound can be dispersed) images made using thermal imaging can be the only option in order to obtain any information about the cracks. In addition, in some cases heat signature can appear before the crack becomes visible (for example pipes breaking from the inside). Yang and colleagues [47] explained how heat based crack detection can be applied in search of cracks on ground and steel surfaces. Their suggested convolutional neural networks give over 95% accuracy for predicting damaged areas on the ground or steel surface. Amjad [48] suggested to search for heat signatures of damages areas by using an IR microbolometer which according to the study costs only around 1% of cost using the other popular thermal detection techniques. Moreover, they state that this method is capable of detecting cracks which are wider than 1 millimetre. However, research does not mention how accurate this method is. Zhang [37], in 2021, offered thermal detection algorithm which according to him can find cracks underwater with high accuracy. The results in his research show that the monitoring scheme with casing tube can detect the crack position in still water, moreover, this monitoring scheme can detect both crack position and width in flowing water. This approach enables to perform crack detection not only in lakes but rivers and seas as well. Zhu suggested another method [38] in 2020, which is based on numerical simulations. In his work Zhu takes such parameters as density, specific heat and thermal conductivity into consideration which allows him and colleagues to track temperature changes on surfaces. Work shows that concrete surfaces have 2.5 times higher density compared to water. In addition, thermal conductivity is over 2 times higher for concrete surfaces in comparison to water meaning that heat transferring is much better in dames or pipes compared to environment. This gives tells us that increase in temperature will be faster noticeable in undamaged areas and that alone carriers a lot of information about cracks. Moreover, cracks will be filled with water which due to lower thermal conductivity and over 4 times higher specific heat will have much lower temperature allowing us to notice affected areas in thermal images. 2021 Cheng [39] continued this work by analysing numerical simulations even farther. Cheng and colleagues suggested the use of porous casing in a way that casing tubes are displayed all over the concrete structure and heated. This way cleaner thermal image of the damaged area can be obtained. He backs this work up by investigating the cooling effects in this system. As we already mentioned, it was noted that concrete surfaces have much better thermal conductivity making heat changes in it much faster than in water. Even more, Cheng estimated cooling process with R-squared value approximately equal to 0.99, which explains almost all estimated data. Any mismatch in cooling can be indicating possible cracks in the observed objects.

For better understanding on how damage on surface looks like in thermal image, this can be seen in figure 1 which shows thermal image of steel plate. As it can be noticed, the affected areas have huge gradient changes in colour values (RGB values). However, while it can be seen in grayscale image as well, but view is not so clear, if we would compare temperature bars, we could see that red, green and blue in grayscale have same shaders (colours matches) this makes it harder for any model to detect changes.



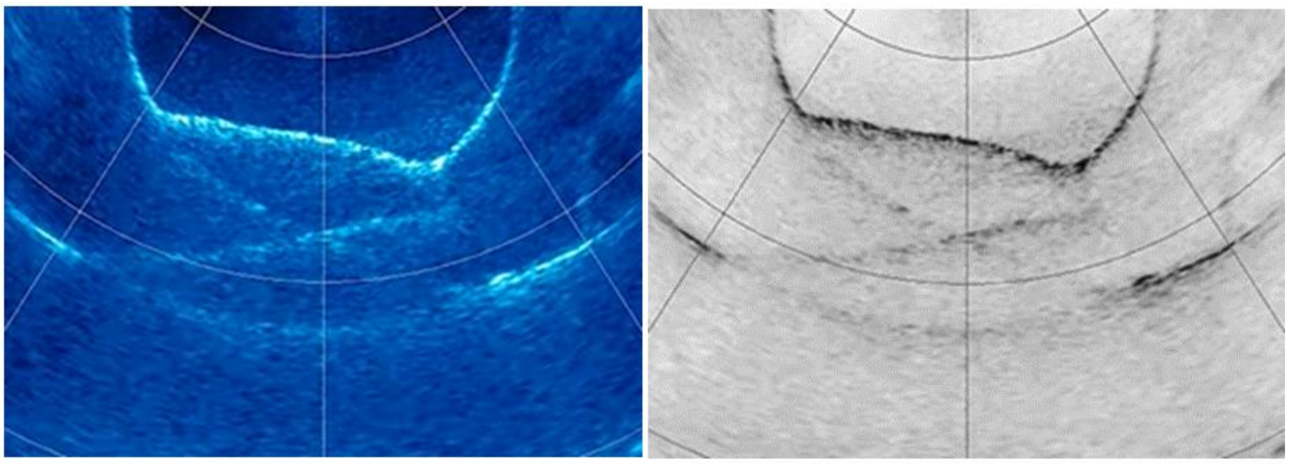
1 fig. An example of steel plate with damaged surface, where left side shows thermal image (taken from [47]) and right side is same picture converted to the grayscale by our study

Overall, although heat based detection is one of the more expensive crack search techniques, it does have its advantages considering that in some cases other methods might not be applicable, a good example of that can be environments with high density which can cover the crack in a way that it is not possible to obtain clear picture and sound wouldn't be able to give much information because of density as well. Taking that to consideration, heat-based detection is one of the more reliable ways to find cracks on various surfaces and especially underwater or over complicated environments.

1.5.2. Sonar based methods

Sonar based algorithms can avoid some limitations which other methods are facing. For example, unlike image processing techniques, sonar images analyses do not require light source to obtain data and results are not influenced by the changes in lightning. Also, this method is cheaper compared to heat detection methods as it does not require extra preparations and images can be obtained faster. All images can be done using autonomous underwater vehicle (AUV) as mentioned by Shi [24]. Shi in his article proposed the crack block tree (BT) algorithm. Idea of this approach is based on minimum spamming trees. Shi suggest that image can be subdivided into blocks where each block holds some information about the crack. Using clustering analysis similar features between damaged areas were obtained and boundary separating non-cracks from cracks were found. Following obtained similarities between damaged areas the minimum spamming tree was created which tells us how the crack looks like (estimated). Shi pointed out that using algorithm damaged areas can be found even if environments are complex and the cracks are tiny. Deep Trekker, company which specializes in creation of underwater robots, released an article [41] in which they mark that in some cases fuel or air can flow out of affected areas this way indicating damage. According to them and remotely operated vehicles (ROVs) can clearly capture these events. They give a demonstration of it providing images with cracks covered in bubbles filled with air. In 2017 Shi [42] expanded his previous work

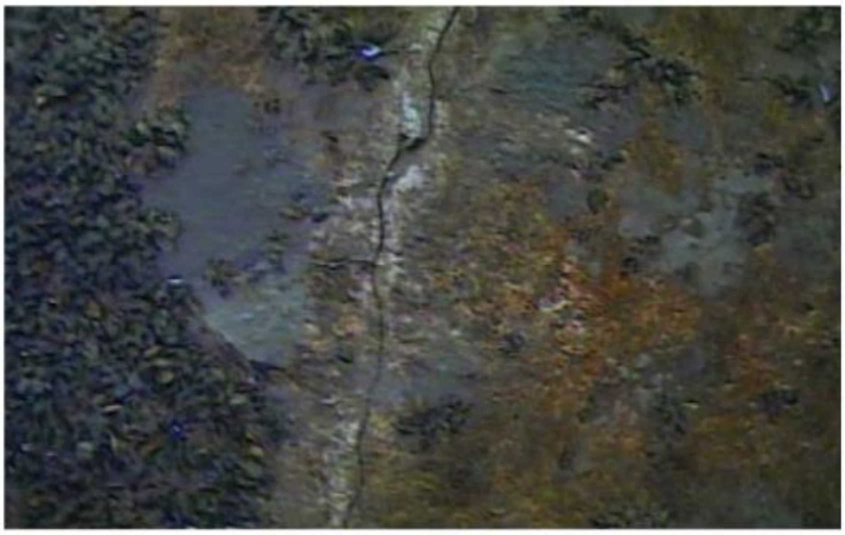
[24] by using the dual-frequency sonar and proposing expansion of his previously suggested method which is able not just recognise cracks but also to classify them based on type. Shi and colleagues take image noisiness into consideration which relates it to the earlier analysed article [41] in a way that noise can be created by crack itself (leaking fuel, air bubbles, etc.) and reduce visibility of the damaged area. He tries to avoid that by estimating the continuation of the crack and tires to relate it with other already observed case. A sonar image used on Shi's work is in figure 2. From figure we can see that cracks appear in the form of light curves, yet most algorithms consider cracks not to be light but rather dark. We converted this image to grayscale and after that performed colour inversion, final output gives an image almost no different from optical images. This allows us to do an assumption that sonar and optical images can be highly related. However, as we already pointed out, it can be affected by environmental noise and clear data can be hard to get even if the sonar or image adjustments are made. This means that methods based on sonar imaging should be trained on images which do imitate some noise in order to get more realistic predictions.



2 fig. An example of cracks captured by sonar (data from [41]) in the right side and same image but converted to grayscale with colour inversion after that on the left side.

1.5.3. Image processing approach

The cheapest crack detection approach would be to use simple images made with camera. This is because anyone can gather data using their phone or other devices which have camera. Therefore, this is the most widely used and well documented approach. However, in order to obtain good quality image target has to be well lit as static and good lighting plays key role in making high quality pictures. A good example of it can be work by Cao who in his paper [43] points out the impact of light refraction which can give unclear imagines as light travel in air faster than in water and as it enters different environments it can bent at certain angles directly influenced by the ratio of densities. Because of earlier mentioned process light can scatter and images can by unclear or light can bend and slightly affect the scale of cracks. Taking all that to consideration Cao managed to create Fully convolutional networks (FCN) model which gave 99.4% accuracy for underwater dams and tunnels. His other suggested model (graph convolutional neural network (GCN)) performed slightly worse with 94.3% on same dataset.



3 fig. An example of an optical image of underwater crack on the dam surface (taken from [40])

In order to achieve better understanding of what problems researchers have to face in order to detect cracks correctly we are giving you an example (see figure 3). As we can notice from the figure 3, crack itself is surrounded by huge environmental noise (corrosion, biological materials, etc.) which can make detection much harder. Aliff analysed the impact of noise in his work [44]. He performed crack detection and segmentation task using photos of pipe from different angles. According to the research, changing angle of the pipe makes some parts of it closer to the camera, naturally, this has impact on details and quality of the data. Aliff notes that near half of all cases were segmentate with some noise. Considering that methods are based on image filters (Raspberry Pi and Canny Edge) we can assume that quality of the image played a big role in overall results, meaning that filters considered some high-quality noise in the data as cracks (edges detection). In 2022 Qi released a paper [45] which is highly focused on image processing. Qi stated (by rephrasing Berman [49]) that underwater images suffer from colour shifts and that most of the shifts appear as a bluish or greenish tone. To solve that problem, he offered a colour correction pre-processing which removes various tones and makes data cleaner. Pre-processed images were used for crack detection and segmentation. For this task Qi created CNN which splits photos to patches and every patch gets label (either it's a crack or not). Next patches with same labels get tested if they are located near each other. Qi states that such model is able to achieve over 93% accuracy for crack detection (classification) on underwater images dataset. Similar research was done by KTU scientists, paper by Orinaitė [50] shows that CNN model based on AlexNet, which was trained on dataset which included underwater concrete cracks with optical effects, can achieve over 99% accuracy. Orinaitė together with colleagues created unique dataset. She used Blender [51] modelling tool to imitate water effects and applied them on concrete cracks images. This approach allows to get generate more data cheaply and can be used to create complicated patterns which can be hard to obtain, as it can be pricey. However, it must be noticed that if possible, testing should be done on real data instead of generated. Yet, that is not always the case as in some cases we want to imitate some patterns and test if model can still keep accurate predictions. In the end, researches by Cao [43] and Orinaitė [50] shows that CNN models can almost perfectly perform crack detection in underwater environment.

1.6. Comparing research

In previous subchapters the increase in complexity of classification task was covered, in this subchapter previously mentioned methods in works from literature will be compared. Comparison will be done taking origin of data to consideration. It is important to analyse the effect of image size, data type (complexity, origin, side factors), when model was created, how many images were used to test it (reliability of results) and most importantly, what was the accuracy of the model or method.

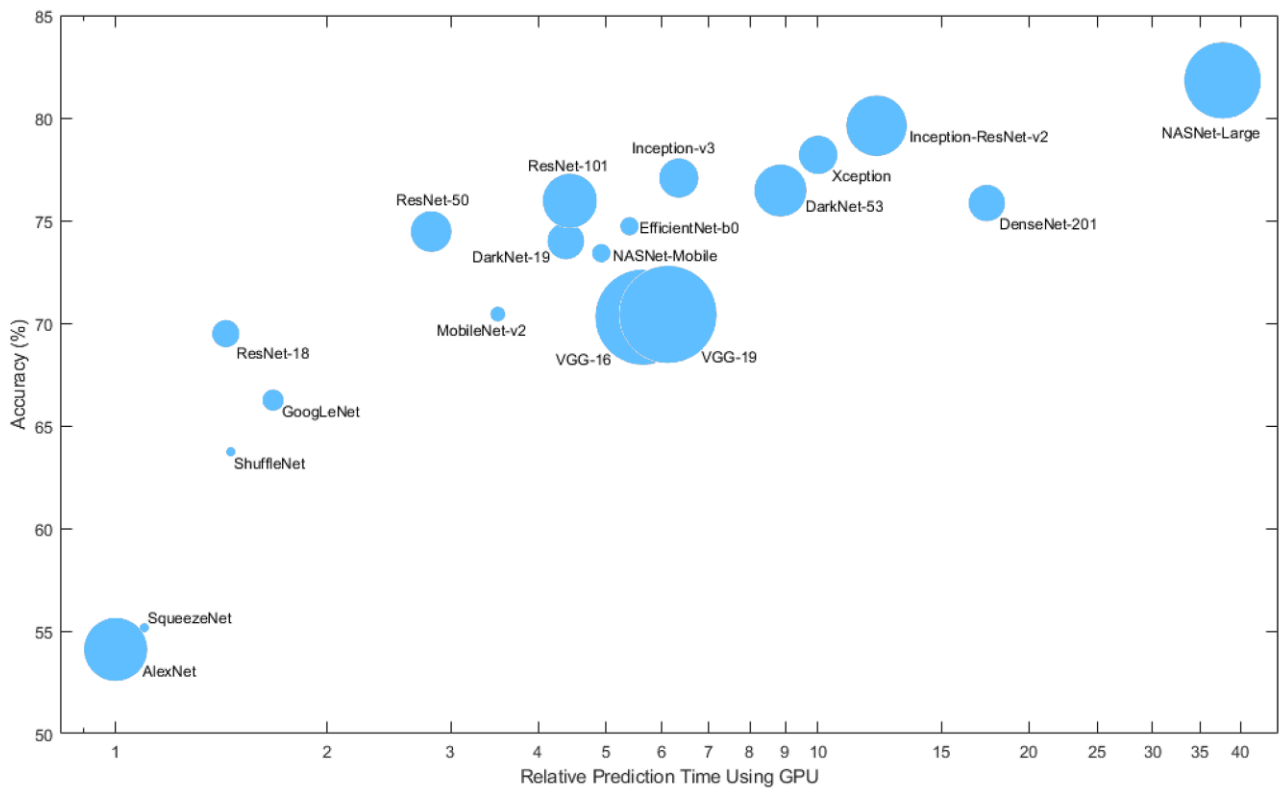
Information about models gathered from various papers and summarized in table 1. Table shows that complex networks can almost perfectly classify datasets to ‘crack’ and ‘non-crack’ classes for simple datasets which only contains grayscale concrete images. However, results are much worse for complicated patterns such as brick walls. As pointed out in paper by Hallee [34] testing on real life data gave poor classification assuming that data had more variety and CNN model was able to learn only a small portion of the background information. However, Loverdos [35] shows that increase of image size and increase of database can highly influence the quality of neural networks raising classification accuracy up to over 96% meaning that models are able to learn patterns well. These results support the popular assumption that neural networks with infinite number of neurons can approximate and recreate any function or behaviour [52,53], however we will talk more about how and why this is possible in methodology chapter. Considering underwater case, it must be noticed that models can very well ignore sea colour effects and distinguish cracks from background, assuming that enough data is provided for model training. This is good because it means that we do not need to completely remake models for such environments as water and it can be enough just to update model with more training data from that environment. Such solution might lead to more versatile algorithm creation as single method might be applied for various cases.

1 table. Comparison of crack detection methods in literature

Method (settings)	Accuracy (%)	Testing dataset size	Type	Size (pixels)	Year	Paper
VGG16	99.88	4000	Simple gray walls	227 × 227	2019	[27]
InceptionV3	99.78					
ResNet	96.95					
DCNN (TL)	98	3420	Simple gray walls	256 x 256	2018	[31]
DCNN (FT)	97					
DCNN (CL)	97					
ED (Roberts)	95					
ED (Prewitt)	97					
ED (Sobel)	97					
ED (LoG)	98					
ED (Gaussian)	98					
ED (Butterworth)	95					
SVM	83.6	213	Bricks dataset (created in laboratory)	512 × 512 scaled down to 100 x 100	2021	[34]
RF	86.4					
GP	84.1					
MLP	84.1					
NB	82.2					
QDA	82.2					

CNN (A)	90.2					
CNN (B)	92.5					
CNN (C)	88.7					
CNN (A)	61.5	90, but unknown if study uses entire image	Brick walls			
CNN (B)	75.5					
CNN (C)	81					
U-Net (#6)	96.02	~700	Brick walls	224 × 224	2022	[35]
U-Net (#10)	95.98					
LinkNet (#1)	96.13					
FPN (#3)	96.07					
DLV3+ (#3)	96.27					
DLV3+ (RMSP, F1L)	96.72					
FCN	99.4	Mentions 66 (out of 522) keyframes with cracks	Underwater surfaces	32 x 32	2022	[43]
GCN	94.3					
AlexNet	>99	~6000	Underwater surfaces	227 × 227	2022	[50]

Original idea of this work was to make algorithm which would be able to work and give predictions in real-time. Problem with that is in computation speed of each algorithm. Assuming that we wish to investigate and find cracks using drone without stopping it, it is required that model would give predictions fast. Performance and prediction speed evaluation was well documented by Pal [33] who together with colleges analysed models created with MATLAB [54]. Results of their research are visualized in figure 4, which shows that in general case AlexNet works faster than other mentioned machine learning models with SqueezeNet being in second place. Study also shows that AlexNet gives lower prediction accuracy than other models. However, this can be countered by using paper written by Orinaitė [50] as an example which proofs that with enough data provided for training set this network can give almost perfect predictions for identification of cracks on concrete surfaces in underwater environment. Taking all of this to consideration it is fair to assume that AlexNet is one of the most suited algorithms for crack detection in real-time.



4 fig. Speed versus accuracy comparison of various neural networks (data taken from [33])

1.7. Summary of key findings

Complex neural networks are capable of classifying simple concrete cracks almost perfectly. However, increase in complexity of the task by adding different background on which crack appears reduces accuracy by a lot, this is especially noticeable from the work done by Hallee [34]. Moreover, it must be noted that cracks on different environments can have different shapes and origin. Therefore, model which gave good predictions underwater might not work as well as model which works well on concrete buildings and vice versa.

Another aspect of crack detection is speed, it must be weighted if benefit of classifying crack correctly is worth the cost it requires to do the prediction. In machine learning case this cost can be evaluated as classification speed. For example research by Pal [33] shows that AlexNet is fastest model of all investigated models. However same research shows that AlexNet is also the least accurate with up to 30% difference in accuracies compared to other networks but faster by up to 40 times. However, research by Orinaitė [50] shows that this CNN can actually give almost perfect predictions (over 99%). This result raises a question if using another model which would cover the remaining <1% (if possible) is needed if it would take up to 40 times longer to get the results.

1.8. Overview

Considering all in earlier subchapters covered information it is clear that methods for automatic crack detection can be improved and applied in commercial use in order to reduce the risk and cost of human labour. For this to happen the fast but accurate algorithm is required. As some researchers pointed out, fast working algorithms can be improved by training them with more data. For this reason, it is needed to create a complex database with huge variety of cracks in various environments and try to train fastest models with created data. It is highly possible that with huge enough material provided to networks (such as AlexNet) they can almost perfectly classify images of concrete, brick walls or even be applied to underwater crack detection. All of this should make it possible for automatic methods to support humans by reducing the risk and load of their work. Doing so should also reduce the cost of regular check-ups which in a process can be more regularly and lead to safer workspace.

2. Data

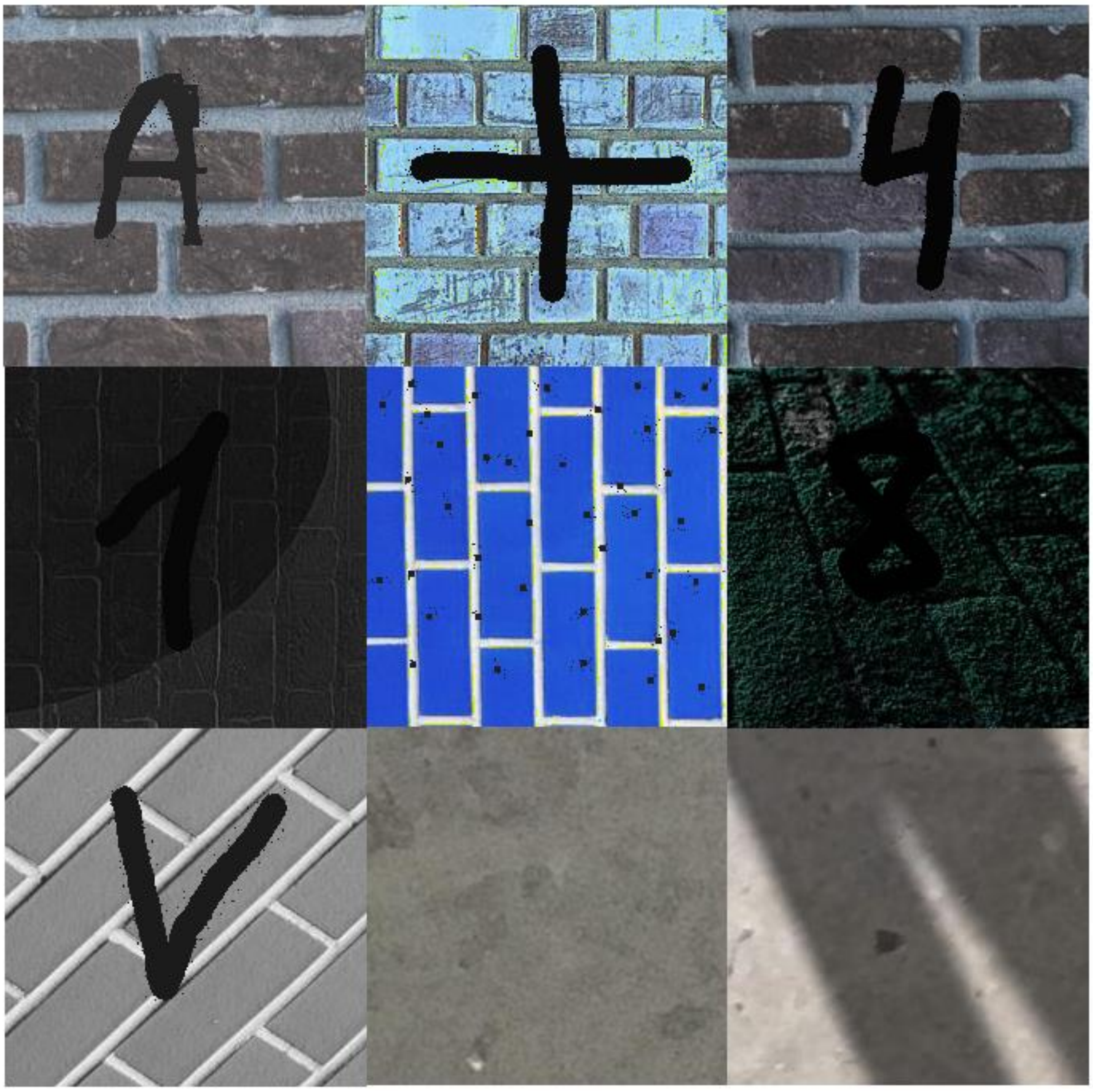
This chapter will cover data gathered and used for model training and testing. Naturally, since study tried to achieved database with more variety the generation of cracks was performed using multiple methods. Also, in the end of this chapter we will cover 3d model example for visualizing underwater pipeline system which can be applied in order to simulate crack detection in various environments.

2.1. Database analyses

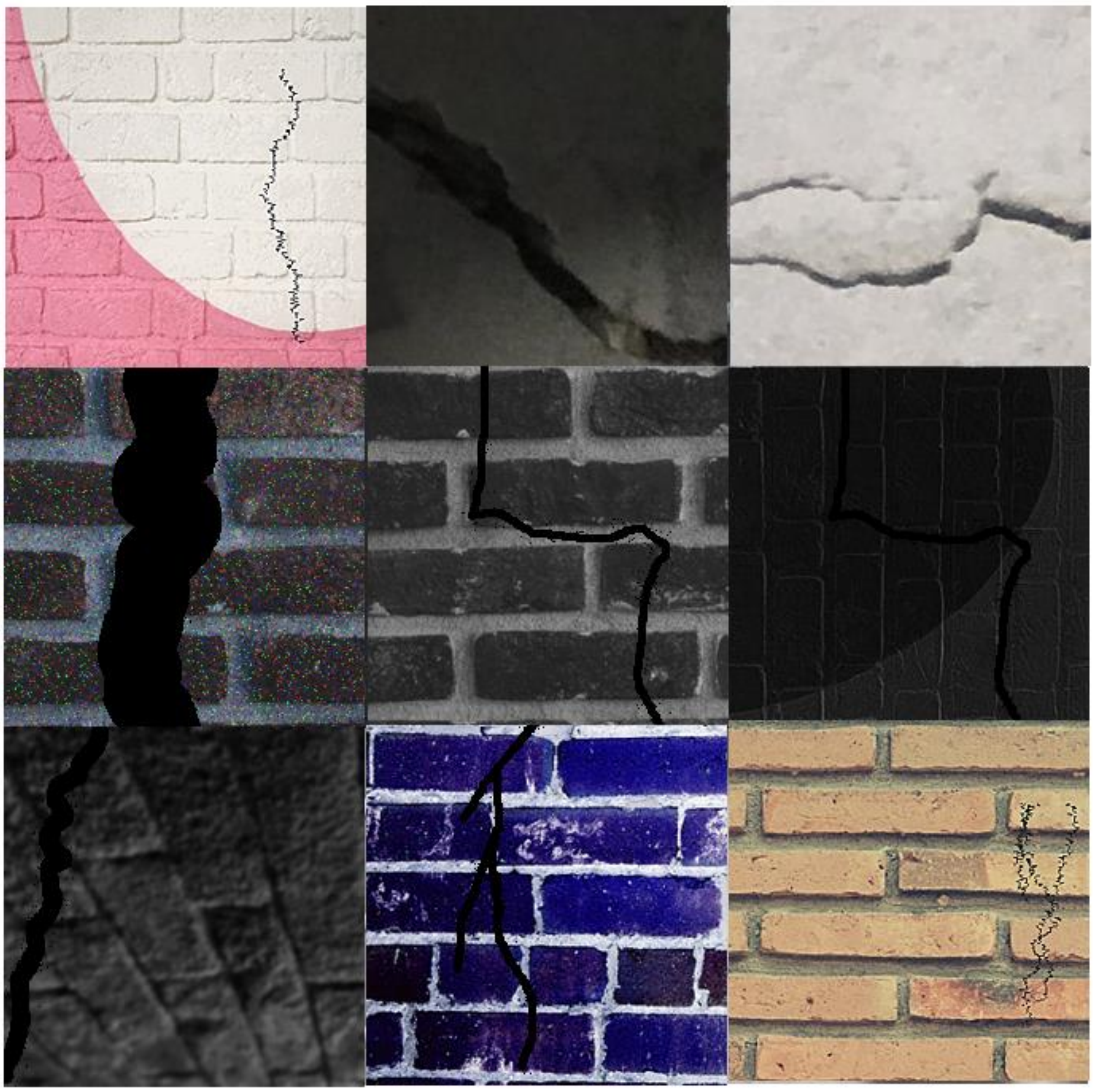
Starting about dataset it must be noted that entire database can be splited into a couple of groups. We have data of greyscale images of concrete cracks and walls without cracks. This set of images was taken from online library widely used for similar researches [55], also we added some shadow images provided by Pal [33]. Another half of the data was gathered from various pictures of brick walls, these images were taken mostly from online websites [56] or captured by ourselves. For all pictures we applied some modifications; some have noise, others shadows, grafiti, filters and so on. Finally to get larger dataset cracks were generated. Generation was done in two ways:

- By applying created masks;
- By running algorithm which generates curve on top of the image.

Examples of positive and negative images datasets are in figures 5 and 6. As can be seen in figures, images are complicated, this should ensure that model would learn various patterns leaving small edge for errors.



5 fig. Examples of non-crack images in our database



6 fig. Examples of crack images in our database

Both sets are similar in size (36476 crack and 38428 non-crack images). To prevent model from learning just to predict dark areas as cracks various symbols were added to negative images dataset this way imitating graffiti and various writings, study believes that it will give more fair data as some so called ‘cracks’ are painted. At the same time this allows study to use same images only changing masks which improves model understanding of how crack should change the image as it can train on various variations of that picture. Using this technique allows model to learn how cracks look like instead of only predicting if crack can be found in the given image. Theoretically, this method should be capable of recreating image before crack and in future works might be used for architecture restoration. However, the task of this study is to predict cracks, therefore this assumption was not tested.

It must me pointed out that study performed multiple image adjustments in which case some images were affected by filters (Gaussian blur) and noises (salt and pepper, Gaussian, Poisson, speckle) all in order to created data with more variety and complexity.

Formula 1 describes gaussian distribution of two-dimensional space in such way:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

We can normalize Gaussian (formula 2):

$$G(u,v) = \frac{1}{S}\exp\left(-\frac{u^2}{2\sigma^2} - \frac{v^2}{2\sigma^2}\right) \tag{2}$$

Here *v* denotes column, $u, v \in \{-\omega, -\omega+1, \dots, \omega-1, \omega\}$
S is the normalization constant (formula 3):

$$S = \sum_{u=-\omega}^{\omega} \sum_{v=-\omega}^{\omega} \exp\left(-\frac{u^2}{2\sigma^2} - \frac{v^2}{2\sigma^2}\right) \tag{3}$$

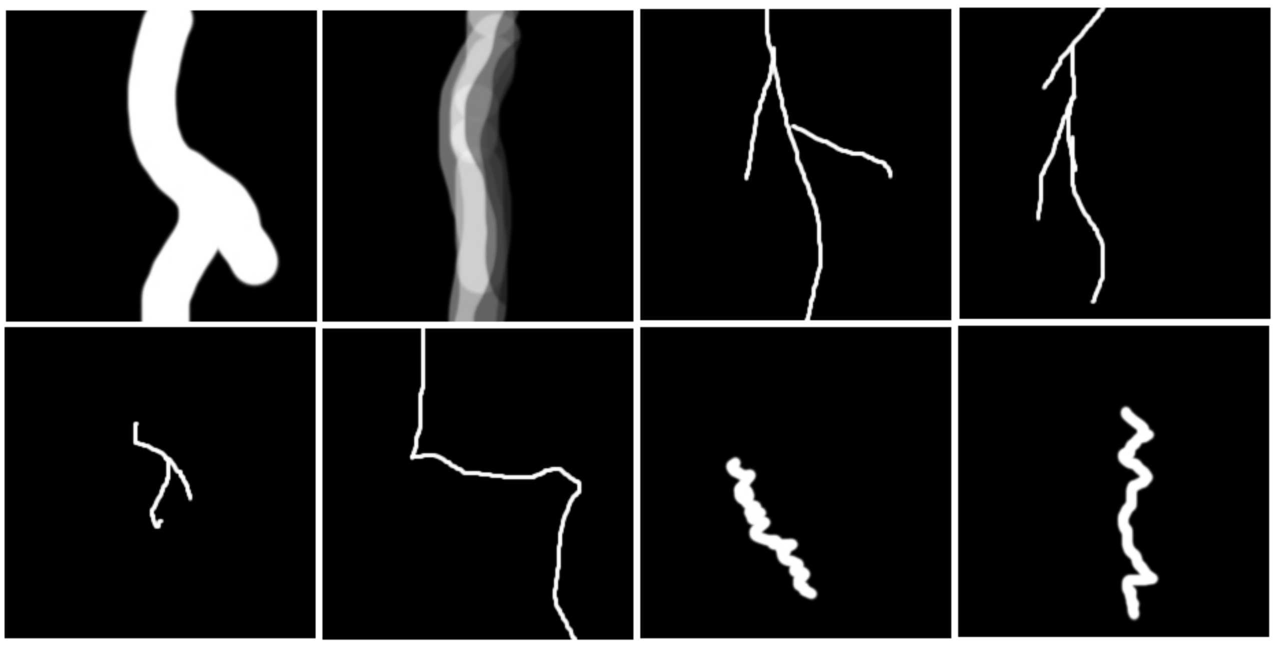
All pixels of the image are recalculated with a Gaussian convolution kernel. The resulting image looks like the image is blurred; therefore, it is called Gaussian blur.

Let the output image be *M*, the input image be *N*, the data in the *i*-th row and the *j*-th column are represented as *N(i,j)* and *M(i,j)*, then the size is $(2\omega + 1) \times (2\omega + 1)$. This means that we only adjust pixel value which is in i-th row and j-th column in the image matrix. The calculated result of the Gaussian kernel with standard deviation σ is (formula 4):

$$M(i,j) = \sum_{u=-\omega}^{\omega} \sum_{v=-\omega}^{\omega} N(i+u, j+v) G(u,v) \tag{4}$$

However, such filtering is very time consuming, and it is not advisable to use it on large images.

2.2. Data generation using masks



7 fig. Examples of masks used in the research.

For data generation using masks we created 30 unique crack images like those mentioned in the figure 7. To simplify extraction, crack masks have white colour and its variations to imitate shadows while non-crack areas are black. Due to black background combining images can be simply done with summing crack mask (*ImgC*) and image on which we want to add crack (*Img*) as show in formula 5.

$$combImg = 0.3 * ImgC + 0.7 * Img \tag{5}$$

This is simply possible because black background has values of 0 while white areas are closer to maximum allowed values (based on used encoding it can be close to 1 or close to 255 and so on). Other cases were tested as well, multiplication gave interesting results in which case, for method to work colours (black and white) have to be switched making white areas black and vice versa. However, it was noticed that multiplication had problem with being too sensitive and ignoring shadows most of the time by making areas with colour-variations still look the same. O the other hand, sum operation performed much better, this is why it was used instead of alternatives. Weighted average in formula 5 gave most realistic results (based on authors point of view) that is why 0.3 and 0.7 values were used.

2.3. Data generation via code

Cracks creation with code is much more challenging process which needs to be checked so that images would have visual crack-like features. In general, the code is created imitation 4 aspects of the crack:

- Location
- Size
- Thickness
- Variation.

It is no secret that cracks are random patterns-like looking structures. In this work we wanted to make an algorithm which would give that impression but at the same time would make crack look natural. This can be achieved with modified random walking. For simplicity we will guide readers over the process how cracks can be generated on the xy-plane. In image case matrix can be rotated and since all images which we use in the project are 227 x 227, this method can be applied to any direction without any problem. Originally, random walk can be described by formula 6, where ε_t is a random value (usually 1 and -1 are used to imitate change):

$$X_t = X_{t-1} + \varepsilon_t \tag{6}$$

For this study we will express formula 6 as matrix which changes over time (iteration), that we can note in formula 7:

$$\begin{cases} X_t = X_{t-1} + \mathrm{E}_t = \begin{pmatrix} i_t \\ j_t \end{pmatrix} \\ j_t = j_{t-1} + 1, \quad 0 \le j_t \ll 227,\ j_t \in \mathrm{N} \\ i_t = i_{t-1} + \varepsilon_t, \quad 0 \le i_t \le 227, i_t \in \mathrm{N} \end{cases} \tag{7}$$

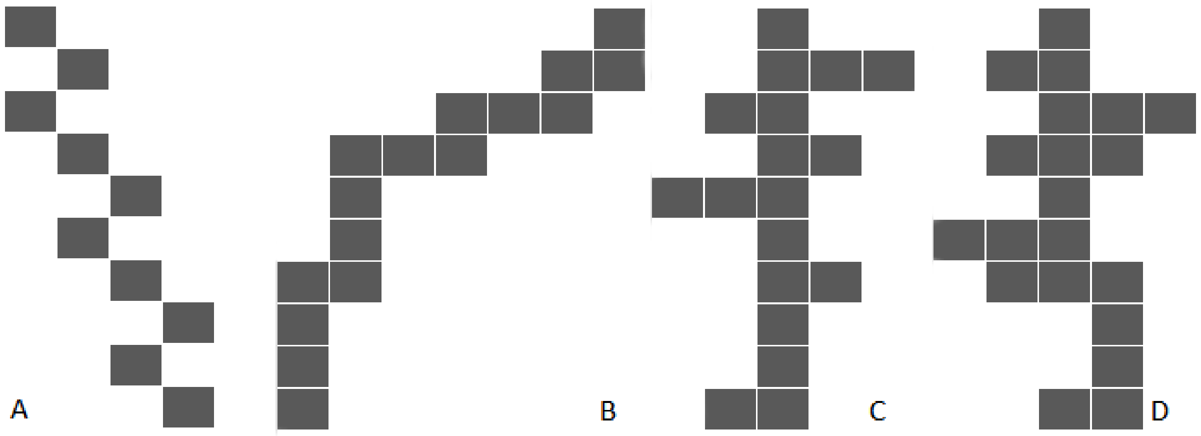
For simplicity let us denote X_t as *X(i,j,t)* in which case *X* value will represent location at iteration *t*, where *i* and *j* mark position in xy-plane. As a result, formula 7 would be truth and it would generate crack similar to depicted in figure 8 A.

In this work an adjustment was made as crack has to have a thickness which would change over time or iteration (*t*). This was done by adding extra parameter as shown in formula 8, which will express how thick crack will be the position *t*:

$$X_t = X_{t-1} + \mathrm{E}_t + \Theta_t \tag{8}$$

Parameter Θ_t only gives information of how many pixels crack should cover in x axis and does not modify y axis at all, the pattern created by following formula 8 is visualized in figure 8 B. Since all movement in image can only be done pixelwise this means that thickness will represent the number of pixels in the row or column in the matrix. This allows us to create crack imitations. However, study noticed that generated patterns lacked variety and complexity of the natural crack. Taking that to consideration a suggestion was made to make parameter $\varepsilon_t$with values 1 and -1 to work as parameter which would flip direction to which cracking is happening instead of showing position. Changes were made and model was generating complicated cracks, but all cracking started from same position. This rule expressed in formula 9 using the Hadamard product as multiplication result and it is visualized in figure 8 C.

$$X_t = X_0 + \mathrm{E}_t \circ \Theta_t \tag{9}$$



8 fig. Explanation of crack generation process; A – random walking, B – random walking with crack thickness, C – with direction modification and D – with cracking position adjustment.

In order to make crack to start at other positions in the row we added adjustment parameter Φ_t, which adjusts crack origin in the new row by making random value from previous already covered interval as noted in formula 10:

$$\begin{cases} X_t = \mathrm{E}_t \circ \Theta_t + \Phi_t \circ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \quad \Phi_t \in [X_{t-2}, X_{t-1}] \end{cases} \tag{10}$$

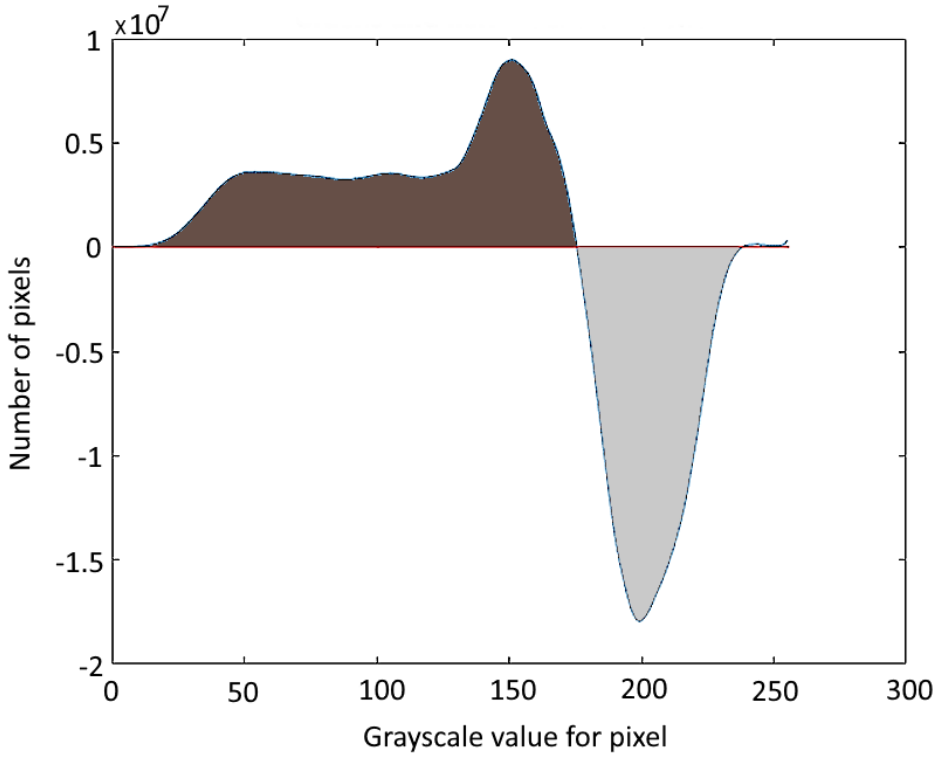
Formula 10 can be simple by explain stating that Φ_t, with component for y-axis equal to 0, shows the location at which cracking starts in xy-plane. This is true because using Hadamard multiplication we can make 2nd coordinate of Φ_t equal to 0 (like it is done in formula 10), doing so removes Θ_t impact of y-axis and only allows it affect x-axis. In addition, E_t shows the direction of the cracking and Θ_t marks how many pixels in that selected direction crack should cover. Following formula 10 and by covering all pixels from the denoted interval in every iteration t, we obtain mapping shown in figure 8 D.

Finally, every covered (by mapping) pixel got low RGB value assigned. Parameter values were changed for different simulations but in majority of cases RGB values for each crack was under 60 (out of 255) making cracks almost black as every colour channel obtained same value.

It is an open question for discussion if colour assignment in such way is the best option, however, study wanted to avoid adding colour variation with randomizing and obtained soft shading by applying soft blurring filters (such as Gaussian) in the end result. In addition, in some cases, crack generation was repeated multiple times as in real life situations cracks can split into multiple branches.

2.4. Database analysis

Features of dataset can have huge impact of results. Since half of our data is concrete images widely used by other researchers [55] we decided to investigate the differences between positive (cracks) and negative (non-cracks) classes. Considering that selected dataset does not include images with huge variety of colours we can assume that the conversion to grayscale will not affect dataset a lot. However, it will simplify analyses process for us. Study computed distribution of grayscale values for both classes, each group contained 20 000 images, this allowed us to ensure that data is well balanced. Finally, in order to find the boundary separating positive and negative classes we subtracted obtained histograms. Result of the positive class distribution and negative class distribution subtraction is shown in the figure 9.



9 fig. The difference in grayscale values between cracks dataset and non-cracks dataset (analyses was done on concrete images from kaggle.com [55])

For figure 9 we can see that images belonging to positive class have more low grayscale values (more pixels have values <170) and vice versa. This allows us to raise a hypothesis that if pixel is darker than 170 (low grayscale value) it is more likely to be a crack.

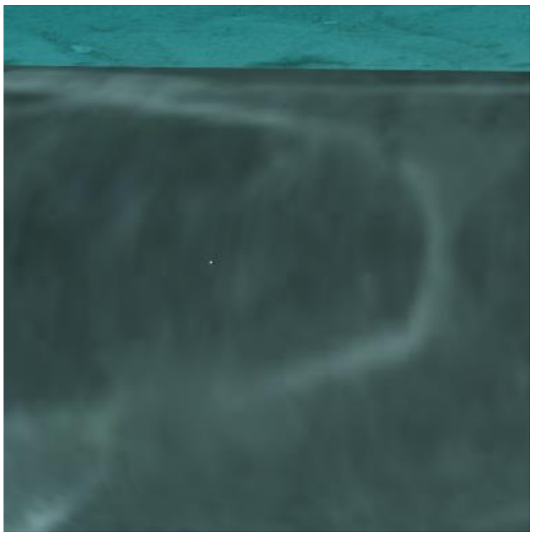
2.5. Pipeline 3D model for data simulation

In order to be able to test model, as well as to make more complicated data with the intend to update CNN in future works, study created 3D pipeline system model (see figure 10). This allows us to gather huge variety of complicated images which might be costly to make. However, for this work we only generated few images with the idea to test how CNN predictions would look like.

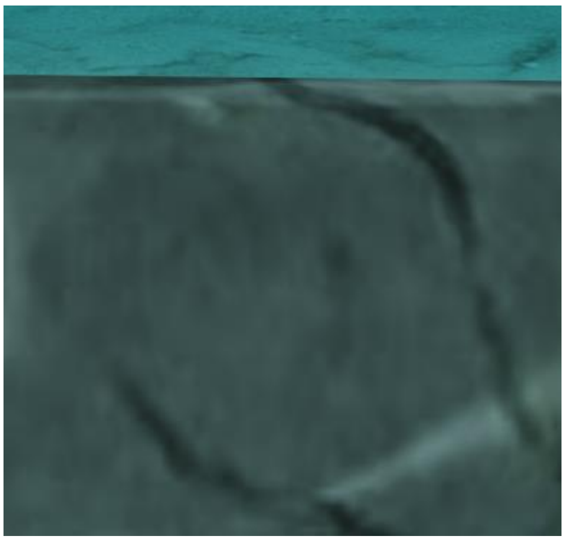
10 fig. 3D pipeline model created in Blender

As can be seen in figure 10, the pipeline can have any shape and length. In addition, user is allowed to select different environments and backgrounds, this makes it possible for generating any type if crack in any environment (underwater, underground, on land, etc.) with the intend to simulate situations in which data might be hard to access. Moreover, Blender is based on rendering images using camera nodes (its simply in software item which shows which area and with what options engine should render), this allows us to select what we want to generate in every frame. Since engine is supporting python programming language it is possible to imitate drone movement which can be useful in future work.

As it is not hard to get pipeline images on land, we decided to make task a bit harder and generate data with underwater environment. All image rendering was done in Blender cycle rendering engine, which is free 3D modelling and texturing software. Generated images using this method are visualized in figures 11 and 12. In this case figure 11 depicts underwater pipe without crack and vice versa for figure 12.

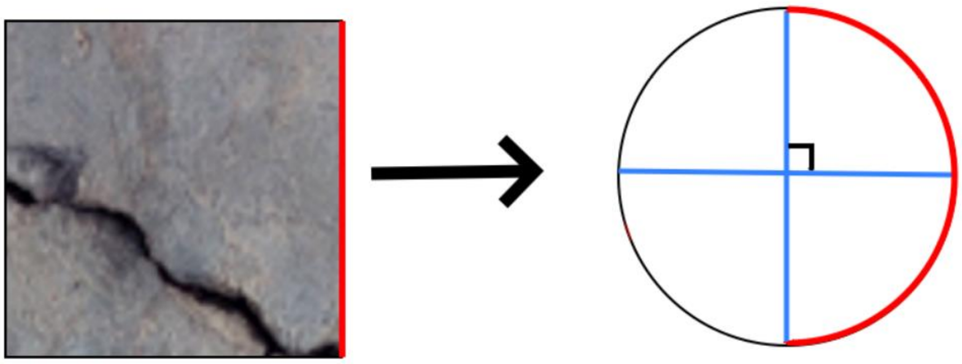


11 fig. Generated underwater non-crack pipe image



12 fig. Generated underwater pipe image with a crack on it

Mapping of image is visualized in figure 13. While x coordinates are not affected by mapping, y coordinates have to be transformed. Transformation can be done by considering the expression of unit circle (see formula 11).



13 fig. Image mapping on 3D cylinder rule where red line shows the y-axis modification on circle (cylinder view from top) on zy-plane

$$\begin{cases} x^2 + y^2 = r^2 \\ r = 1 \end{cases} \tag{11}$$

From this we can express the transformation function by denoting transformation result of y coordinate as y^* in which case transformation result can be written by formula 12.

$$y^* = \sqrt{r^2 - y^2} \tag{12}$$

However, we are talking about image transformation in 3-dimensional space. In our case transformation does not modify xy-axis itself but describes a rule how y axis can be visualized in z axis ($z = y^*$). Therefore, the image transformation function $F(x, y)$ wraps picture on cylinder and converts 2D image to 3D view based on rule (see formula 13):

$$F(x, y) \to \left(x, y, \sqrt{r^2 - y^2}\right) \tag{13}$$

The mapping rule which we explained right now works for cylinder shapes in case cylinder is rotated mapping would change based on the rotation function used in engine. Same logic applies for pipe bending, if bending is performed mapping can be adjusted based on curve which describes the shave of the pipeline. Since too many combinations are possible, we will not derive mapping rule for modified system. In addition, since Blender handles it automatically, it is enough for us to know formula 13.

3. Methodology

Selection of correct method is the main part of good model creation. In this chapter we will go over the process why specific methods were selected, how they work and how exactly they can be applied in order to create model which is capable to perform crack detection with high accuracy.

3.1. Convolutional Neural Networks

Convolution Neural Networks (CNN) are a type of neural network that is commonly used in image and video recognition, analyses and encoding tasks. They are created in such a way that CNN can automatically learn and extract features from images by analysing the patterns. Usually, main parameter describing the pixel is its colour value.

Convolution Neural Networks is a structure of Convolution, Pooling and Fully connected hidden layers combined together in order to learn features of images or matrixes in general and repeat those features or predict classes based on information which network was able to extract. In a convolutional layer, the network applies a set of learnable filters to the input image to extract features such as edges, corners, and shapes. The output of the convolutional layer is passed through a non-linear activation function, such as the ReLU (Rectified Linear Unit), to introduce non-linearity in the model. The structure of fully connected layers can be expressed with formulas 14, 15 and 16:

$$Y_{k-1}W_k = w_{k,0}y_{k-1,0} + w_{k,1}y_{k-1,1} + \cdots + w_{k,n}y_{k-1,n} \tag{14}$$

$$Y_k = actF(Y_{k-1}W_k + B_k) \tag{15}$$

$$B_k = (b_0, .., b_{k-1}) \tag{16}$$

As stated by Khedgaonkar, the convolutional layer is considered an essential block of the CNN. It is necessary to understand that the layers’ parameters and channel are comprised of a set of learnable neurons. These neurons have a small receptive field. In the feed forward process, every individual channel goes over the dimensions of the input, thus calculating the dot product from the filter (kernel) pixels and the input pixels. The result of this calculation is a two-dimensional feature map (matrix) [57].

The idea of pooling layer is that we use various filters to reduce dimension of previous layer, this way trying to create smaller size feature map or its compressed variant. It is needed in order to train faster model as well as to create model for more general case as some feature might only be found in large scale images and not give much information. With pooling layer, we take only key information with each image is bringing this way model is less effected by noise and is almost sure that most of the images in the class will share same features. Pooling layers reduce the dimensionality of the feature maps by down sampling the image, which helps in reducing the computation and making the model less prone to overfitting. The most common pooling method is the MaxPooling, which selects the maximum value from a pool of adjacent values in the feature map.

Finally, the fully connected layers take the output from the convolutional and pooling layers and use them to classify the input image into the desired categories. The fully connected layers use standard neural network techniques, such as backpropagation and gradient descent, to learn the weights of the model and minimize the loss function during training.

Overall, CNNs are very powerful and widely used for various computer vision tasks such as image classification, object detection, and segmentation. They have achieved state-of-the-art performance in many benchmarks and competitions and are widely used in industry and academia. This being said, CNNs are one of the most often used methods for crack detection (at least based on literature sources which we analysed).

3.2. Transfer learning and how it works

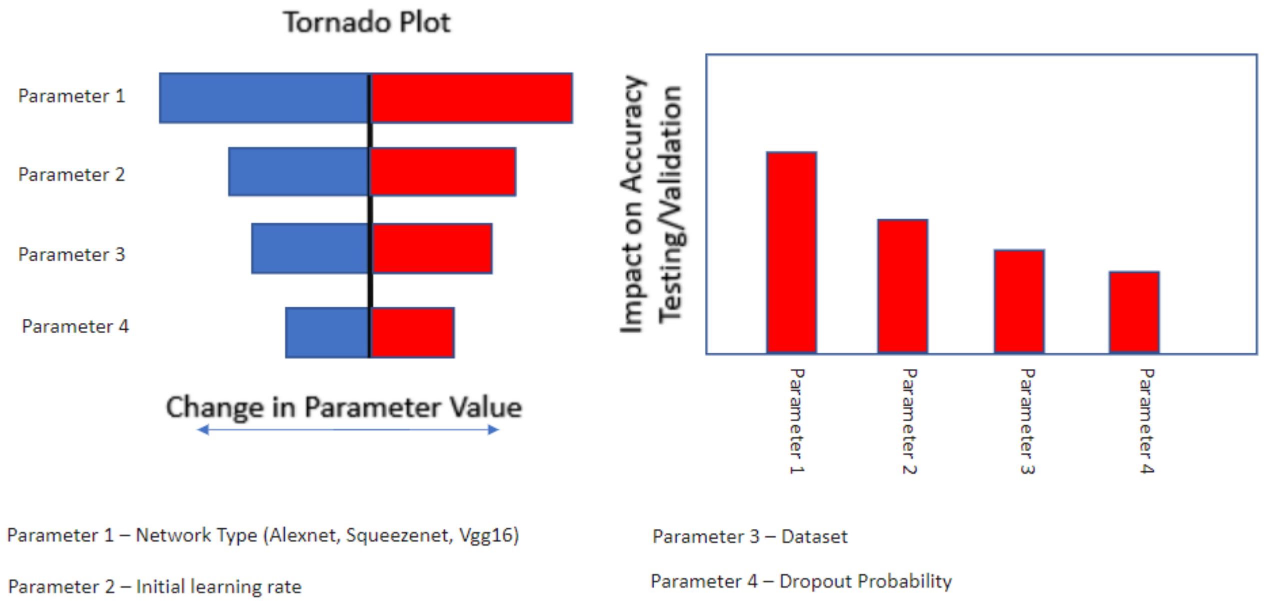
Transfer learning (TL) on the other hand, is a technic first mentioned in 1976 by Bozinovski and Fulgos [58]. TL methodology states that patterns from one model can be applied to train another similar model. This training is more effective as it requires less time and training data to learn new patterns while it is also showing accurate results if feature maps are similar. Transfer learning, as we slightly mentioned in the beginning, is a machine learning technique where knowledge learned from one network is applied to simplify the training of another network with data which have similar features. It involves taking a pre-trained model that has been trained on a large dataset and using it as the starting point for a new model trained on a smaller dataset for a different task.

In TL, the pre-trained model's learned features are transferred to the new model, and only the last few layers are fine-tuned on the new dataset. This is done because the initial layers of a pre-trained model capture low-level features, such as edges and textures, which are likely to be relevant to the new task as well. By using Transfer learning, the new model can be trained with a smaller dataset, leading to faster training times and better generalization performance. Additionally, the pre-trained model can act as a form of regularization, preventing overfitting on the new dataset. TL has been successfully applied to a wide range of machine learning tasks, including image classification, object detection, natural language processing, and speech recognition. It is particularly useful in cases where the new dataset is small or lacks diversity, or when computational resources are limited.

Pan and Weis [59,60] explained TL with a wonderful real-life example of musicians. According to them we can imagine TL thinking about two people who want to learn to play the piano. One person has no previous experience playing instruments, and the other person has extensive music knowledge through playing the guitar. The person who already knows how to play the guitar will be able to learn the piano in a more efficient manner by transferring previously learned music knowledge to the task of learning how to play the piano.

3.3. Hyperparameter optimization

We finished last subchapter with an example of musician. Yet, no matter how good pianist person is, he will never be able to play well if song itself is not good. In machine learning hyperparameters can most likely be compared with notes. We can select any parameter and model will surely going to work. Person can play any notes and he will make a song, but will it song good? In order to create good model, one has to select optimal parameters. One of the simplest way to find optimal parameters is by using MATLAB’s Experimental Manager [61]. This toolbox allows user to set his entire experimental design and try various case by running them and showing all main statistics about each case. A good example of this can be work by Pal [33] who used it to find best options for concrete cracks detection (part of which we took into account in our work as well). His research shows (see figure 14) that network and learning rate have the largest effect on the accuracy of models.



14 fig. Effect of parameters for concrete cracks detection (data based on Pal [33])

In our work we also use Experimental Manager toolbox to find best parameters out of all tested cases. It must be noticed that this does not ensure global best values as we have limited resources and cannot test all cases. However, the effect of each parameter was tested and described detail, which should give good understanding which combinations might give most optimal options.

3.4. Optional pre-processing

Following the analysis of the data this study made a conclusion that predictions can be improved by adjusting data furthermore. However, we suggest that this method should be optional as it might not work for all data. As it was already mentioned, our data analysis shows that images with crack often have more darker pixels while vice versa applies for non-crack images. The difference in histograms is shown in figure 9 (back in data analysis chapter). Following this we ran a test and results shows that our networks predict single colour images as cracks if the value of colour is under 120, this is simply because majority of images with cracks had average close to that value.

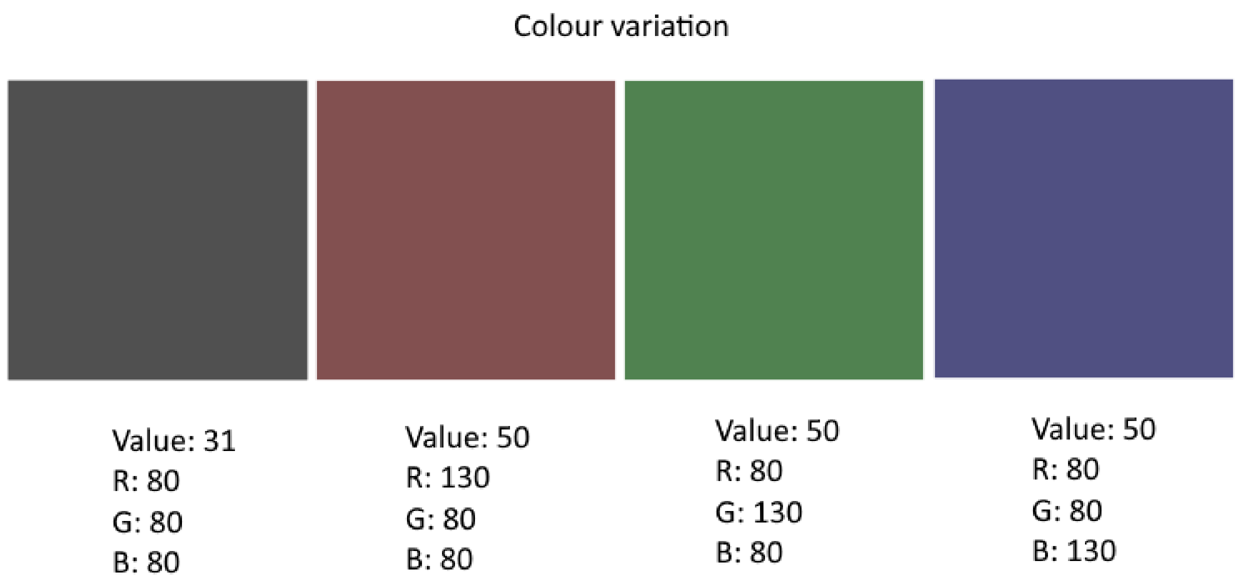
We also believed that improving quality of images can fix the issue with misclassification. This can be done in multiple ways. Yet our study tried to do it using Super Resolution (SR) method based on CNN as suggested by Kim [62]. However, our research into this shows that improving quality of image using convolution neural networks which would add more details to pictures is not a good approach as extra details makes model to classify images as cracks in majority of cases. We believed that this is the case because of noise which method is creating on image as well, however, removing noise with networks using similar technics as autoencoding didn't help in solving the problem. For this reason, we decided to take different approach and apply mathematical solution in order to improve predictions of our main neural network.

Assuming that naturally cracks are always variations of dark colours (black, gray etc.) and never white, red, green, or blue (unless it is under some lighting) we wrote a program which would change image colour in comparison with average colour of that image. In general, wall and crack should have huge difference in colours and while cracks are dark, normal wall most likely will be lighter, meaning anything above average value will not be crack. As we already mentioned, this might not be always the case as we can be looking at black wall or crack can be under unique colour lighting (sunrise, neon lights and similar). Yet, in most cases this rule would work. By allowing person to manually select this option (as users most likely will know the environment in which data was taken) we can avoid this problem and highly improve predictions. Therefore, to be able to use this method data should fit few criteria:

- Cracks must have some darkish shader, which means that cracks cannot be white, red etc.
- Environment cannot be darker than crack.

Method is based on few phases, first, we measure RGB values of all 3 colours. After that we check if difference between maximum and minimum values are greater than 50 (this value was taken experimentally by trying multiple values and making sure that cracks would not get into that range). We also tested that in most cases shaders have 10 to 30 in terms of difference between values of colour channels. As we earlier mentioned, the crack is almost always will be a shade of black or gray and will almost never have sub-colour (reddish, greenish, blueish, etc.). Figure 15 shows the combinations of 50 value for RGB colours. By this idea we state that if pixel has difference in colour channels larger than 50 it is not a crack (it can be noise, maybe even can be part of the wall, it doesn't really matter as long as this is not a crack) so what we do is that we take same image just in grayscale format and give that image pixels high colour values (for this research we decided to use 199 as value to give as it is higher than 120 (edge at which models split cracks and non-cracks as well as biggest difference between cracks and non-cracks histograms based on figure 9. For simplicity the rule of how colour adjustment should be made can be written by formula 17.

$$\text{If} \begin{cases} \max(R,G,B) - \min(R,G,B) > 50, & value = 199 \\ \max(R,G,B) - \min(R,G,B) \leq 50, & value = value \end{cases} \tag{17}$$



15 fig. Images show how adding 50 to each RGB channel can change image.

This colour removing method is needed because in some cases when neural network converts RGB images to grayscale images some pixels will have dark shaders and model will consider them to be cracks even if in the reality it used to be just a noise the shader of red, blue, or green.

Next step is a bit more complicated, as we want to remove noise, we try to imitate segmentation. We already talked that in most cases cracks are darker than non-crack areas and average can be good way to determine the that. Yet, if we analyse image without cracks average will not work that well as half of pixels will be slightly under average value, for this in this work we tested that multiplying mean value by 0.8 and using it as border to split non-crack area from unclear area is best solution. Also, since method consider values under 120 as cracks, we want to give this area a solid colour which would be equal or over 120. In order not to separate image in terms of contrast too much we suggest using mean value of histogram (if it is higher than 120) or 120 itself (if mean is under 120) and make all pixels' values based on formula 18.

$$\text{If} \begin{cases} value \geq mean \times 0.8, & value = \max(mean, 120) \\ value < mean \times 0.8, & value = value \end{cases} \tag{18}$$

Following this rule will make most of the image to have single value colour and neural network will only needs to evaluate area which is not single value. However, to avoid creating fake gradient of colour values we also used weak Gaussian filter on output image which would merge zones together but would not affect image quality a lot.

4. Results

We ran our modification of the AlexNet model on different size datasets (for code example see appendix 1). First tries with around 1130 images brick walls dataset gave over 60% accuracy rate. However, all images used in set were complicated. We noticed that testing those images with model trained just on concrete walls gives around 60% accuracy as well, but most of the images were classified as cracks meaning that model does not separate colours from cracks well. However, we observed that combining datasets improves predictions drastically. This can be explained by assuming the fact that overall, in majority of cases cracks on brick walls still have same features as concrete cracks. On the other hand, in order to ensure that model wouldn't overlearn or wouldn't predict single class we had to keep balance between cracks and non-cracks data. Also, same rule would apply to brick walls as due to complexity of images bricks are more reminding cracks, so we need good amount of data to avoid that.

4.1.1. Results of CNN

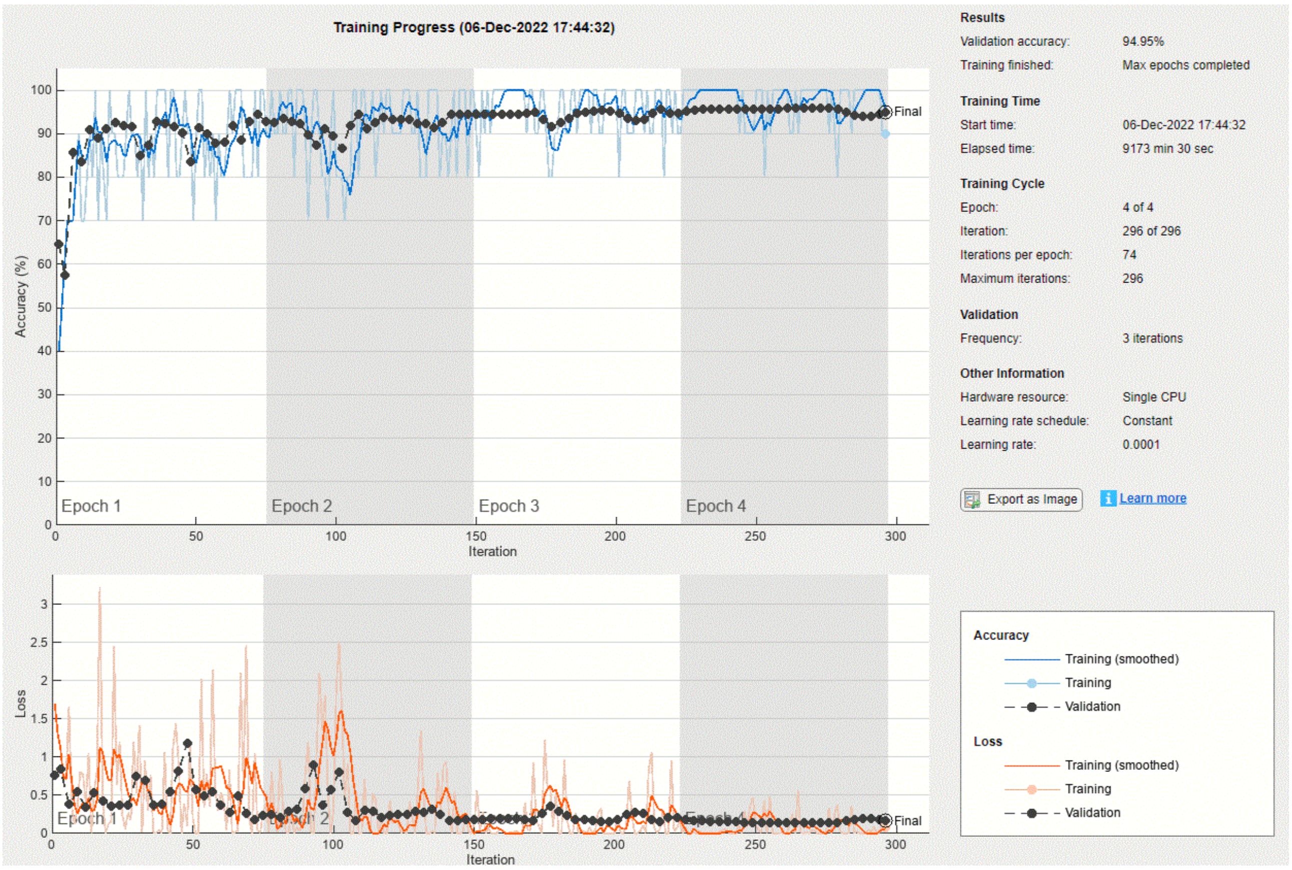
To investigate the effect of parameters hyperparameters optimization was performed using experimental designer toolbox in MATLAB. For each trial 10% of database was used for model training (around 7500 images). This gave very good results even for low amount of data reaching over 98% accuracy in three cases. Investigation shows that using low learning rate highly improves prediction accuracy. Our research noticed that best performance was obtained with 'sgdm' optimizer. It can be noticed as well that turning off verbose can slightly improve classification. However, the effect of verbose parameter is not very high. We observed that validation frequency of [3, 7] iterations give best results. Yet, this can be very dependable on what data was taken for training. Notable results are written down in table 2. From this table we should point out that although model accuracy is high in some cases, but it can have higher validation loss compared to model which has lower accuracy. Validation loss shows how well model can separate two classes; therefore, high loss value means that CNN is mis-classifying part of the images with huge error. Large errors can be a problem if we try to predict cracks on unseen database which have some unique features.

2 table. Hyperparameter optimization using experimental designer on 10% of all data

Trial	Learning rate	Optimizer	Validation frequency	Verbose	Validation accuracy (%)	Validation loss
20	0.0001	sgdm	5	False	98.1703	0.0542
8	0.0001	sgdm	3	False	98.0101	0.0614
56	0.0001	sgdm	3	True	98.0073	0.0504
32	0.0001	sgdm	7	False	97.9328	0.0598
68	0.0001	sgdm	5	True	97.8316	0.0647
16	0.0001	adam	5	False	97.5196	0.0853
24	0.0001	rmsprop	5	False	97.4170	0.1191
12	0.0001	rmsprop	3	False	96.9926	0.1453
36	0.0001	rmsprop	7	False	96.7706	0.4430
52	0.0001	adam	3	True	96.5303	0.0985
44	0.0001	sgdm	9	False	96.4052	0.0864
28	0.0001	adam	7	False	96.3082	0.1459

72	0.0001	rmsprop	5	True	96.2956	0.1235
48	0.0001	rmsprop	9	False	96.1382	0.2149
40	0.0001	adam	9	False	95.9189	0.1056
64	0.0001	adam	5	True	95.6590	0.1803
4	0.0001	adam	3	False	95.5395	0.1603
60	0.0001	rmsprop	3	True	94.4574	0.4630
73	0.1	adam	7	True	51.5051	248850.3125
1	0.1	adam	3	False	51.3027	1816.7488

It was observed that learning rate gave biggest effect to predictions accuracy. Yet, this was only tested on 10% of entire database. As it was mentioned before, to reach our goal, large database with around 75000 images (similar sizes for each class) was created. Next the effect of database size was investigated. The training was performed on 1% of this dataset and results reached over 94% while predicting mixed data (brick walls and concrete walls). The training process is visualized in the figure below (figure 16). Since 2 of 3 best performed trials had validation frequency equal to 3, we decided to use it in this investigation as well same as turning off verbose option.



Results	
Validation accuracy:	94.95%
Training finished:	Max epochs completed
Training Time	
Start time:	06-Dec-2022 17:44:32
Elapsed time:	9173 min 30 sec
Training Cycle	
Epoch:	4 of 4
Iteration:	296 of 296
Iterations per epoch:	74
Maximum iterations:	296
Validation	
Frequency:	3 iterations
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

16 fig. Training process on 1 percent of dataset

It can be noticed (see figure 16) that starting from around 150th iteration prediction loss and accuracy measures almost stabilized. This shows that most of the information about model training can be covered within first 150 iterations and remaining iterations do not give significant information to model. However, they do increase classification accuracy, therefore, since study did not notice anything what would show possible overlearning these options were not changed.

It must be pointed that increasing training data from 1 percent up to 5 percent (out of ~75 000 images) increased predictions accuracy from 94.9% up to over 98% but difference in accuracy between 5% and 10% of database size is only minimal. It shows that most of the data share same features and only small fraction of database are complicated cases. We tried to confirm our assumption by analysing confusion matrices (see table 3) and indeed, it can be noticed that models trained on 5% and 10% of entire data have highly noticeable change in mis-predicting cracks as 5% model is more likely to classify crack as not crack and vice versa for 10% model. Increasing training set size to 15% gives predictions in between. This shows that CNN is searching for optimal boundary conditions of how to separate two classes and increasing data seems to be part of the solution.

3 table. Confusion matrices of 5, 10 and 15 percent of entire database size cases

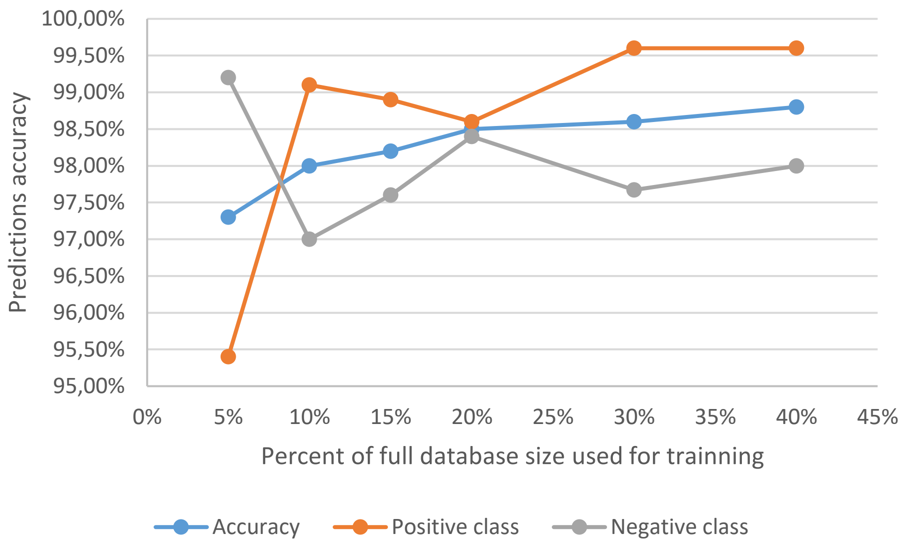
% of database	**Predictions**	
5%	**Negative**	**Positive**
Negative	36209	297
Positive	1590	33062
10%	**Negative**	**Positive**
Negative	33550	1034
Positive	286	32542
15%	**Negative**	**Positive**
Negative	31892	771
Positive	345	30660

From table 4 we can see that increasing training database size from 10 to 40 percent only gives 0.82% accuracy. Yet, while this value looks low, but we wish to note that it was validated on thousands of images meaning that even part of percent can show that dozens of cracks were detected correctly. This is very important seeking to ensure that model can be used for crack detection in high-risk areas such as nuclear plants, dams etc. Moreover, accuracy equal to 98.82% shows that this research can match top 5 models analysed in this work (from literature).

4 table. Model accuracy for different training set sizes

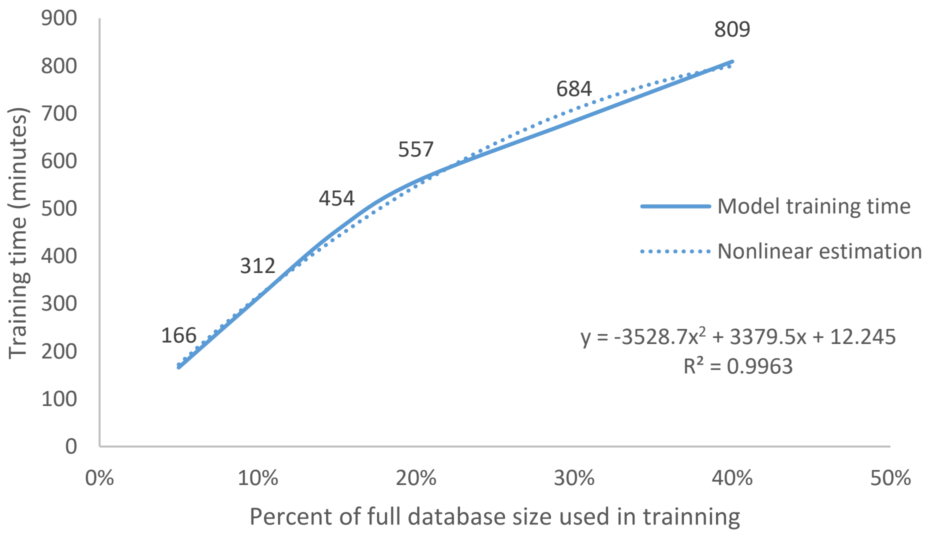
Training database percent	**Model accuracy**
1%	94.9%
5%	97.3%
10%	98.0%
15%	98.2%
20%	98.5%
30%	98.63%
40%	98.82%

Taking growth of accuracy compared to increase in size of training dataset into account (see figure 17), it is very likely that increasing training dataset even more would improve predictions. However, as we can notice this increase is not high and other factors such as time it takes to re-train model or overlearning should be considered.



17 fig. Plot of observed accuracies by our model

Since we already checked possible accuracy, it is only natural to estimate how long the training would take with increase of training dataset (just reminding that full database size is ~75 000 images). All expected changes in model training time versus database size expressed as percent (from ~75 000) is plotted in figure 18. It can be noticed that change in training time is not linear. It was observed that 2nd order polynomial function is able to explain training time based on dataset size with R-squared equal to 0.9963. On the other hand, these results are only approximate as on another computer we observed that training speed was from 3.36 to 3.38 times slower than the values visualized in figure 18.



18 fig. Model training time based on amount of data

Next, testing on completely unseen data gathered from internet was done (for code example see appendix 2). Structure of testing set:

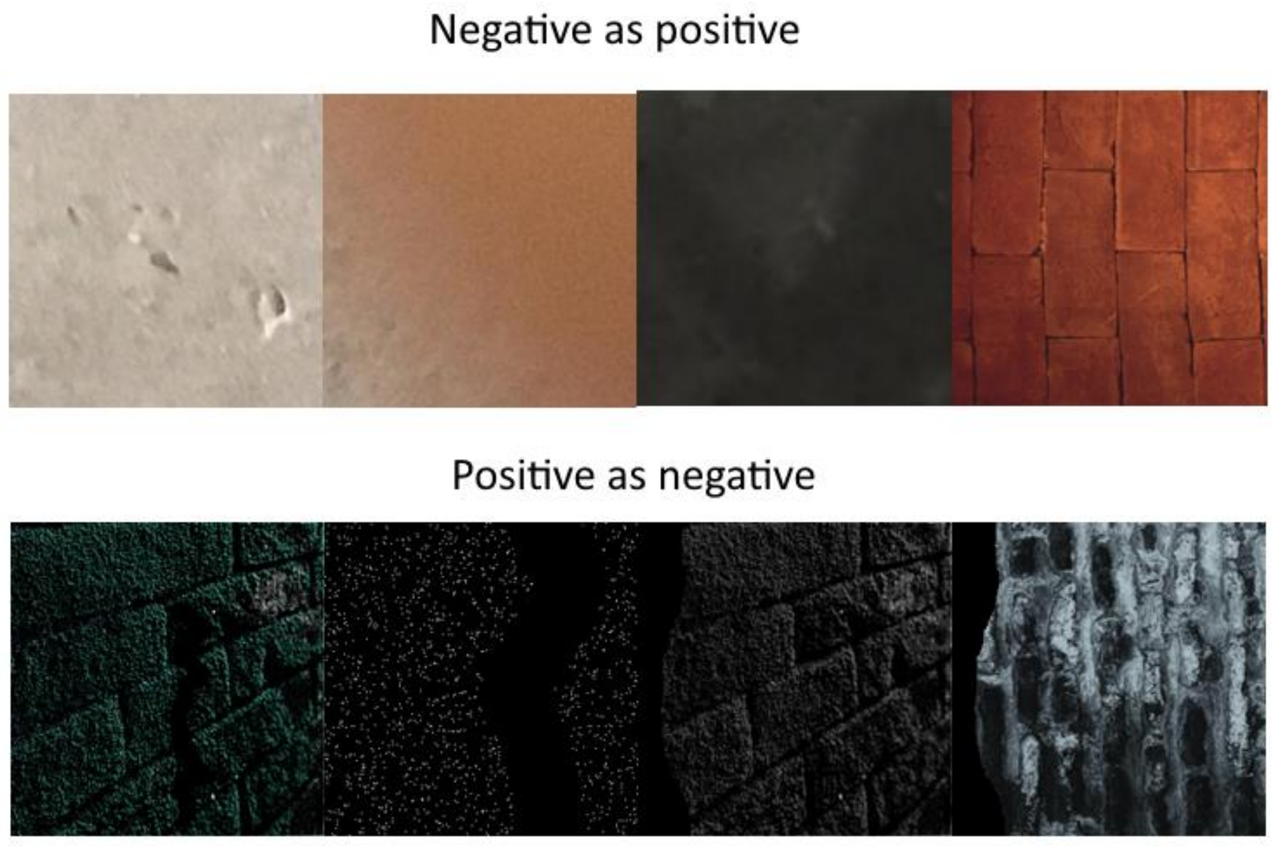
- 25 concrete crack images
- 25 concrete non-crack images
- 25 brick wall images without cracks
- 25 brick wall images with cracks

We observed that our model gave 82% accuracy on this dataset by making some errors with brick walls. However, adding gaussian blur with sigma equal to 0.3 raised predictions up to 99%. This shows that our model is still can be highly influenced by noise and some kind of pre-processing approach is needed to obtain more reliable predictions.

In addition, model was tested in real-time by connecting it with camera. Walking across the room and filming walls with and without cracks proves to give satisfying results. This shows that model can be used for real-time detection. Yet, it must be pointed out that fast camera movement over areas covered by shadows can give unstable predictions but as soon as camera stops to move model gives clear results.

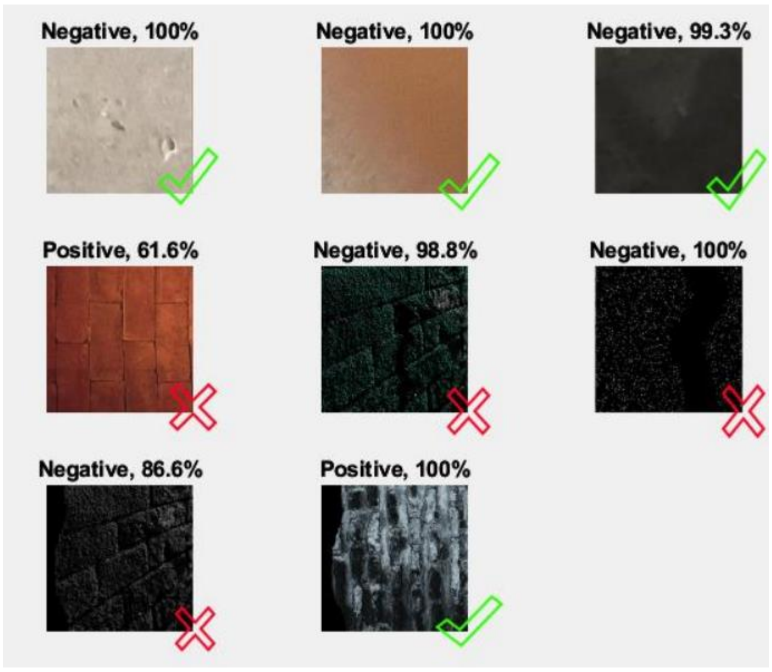
4.1.2. Challenges

As mentioned earlier in this work model faced some challenges. Figure 19 shows the misclassification cases which our model made. Although expanding training set improved predictions and solved some issues, but main challenges remain. One on the most important problem is that model predictions are highly affected by noise. From figure 19 we can see that images with noise can easily be predicted as cracks.



19 fig. Misclassified images by model

Another problem which we which to point out is that dark images can be labelled as not cracks. Although, we must mention that this issue is mostly noticed for generated cracks as we did not have a not of dark surfaces with real cracks. Also, this is mostly true if image contains a lot of noise.
Since our database had a lot of low-quality pictures, we tried to test the effect of resolution increase and decrease techniques. It was noticed that resolution increasing using CNN can make model to predict more images as cracks. Again, we believe that it is related with creation of extra details (noise).

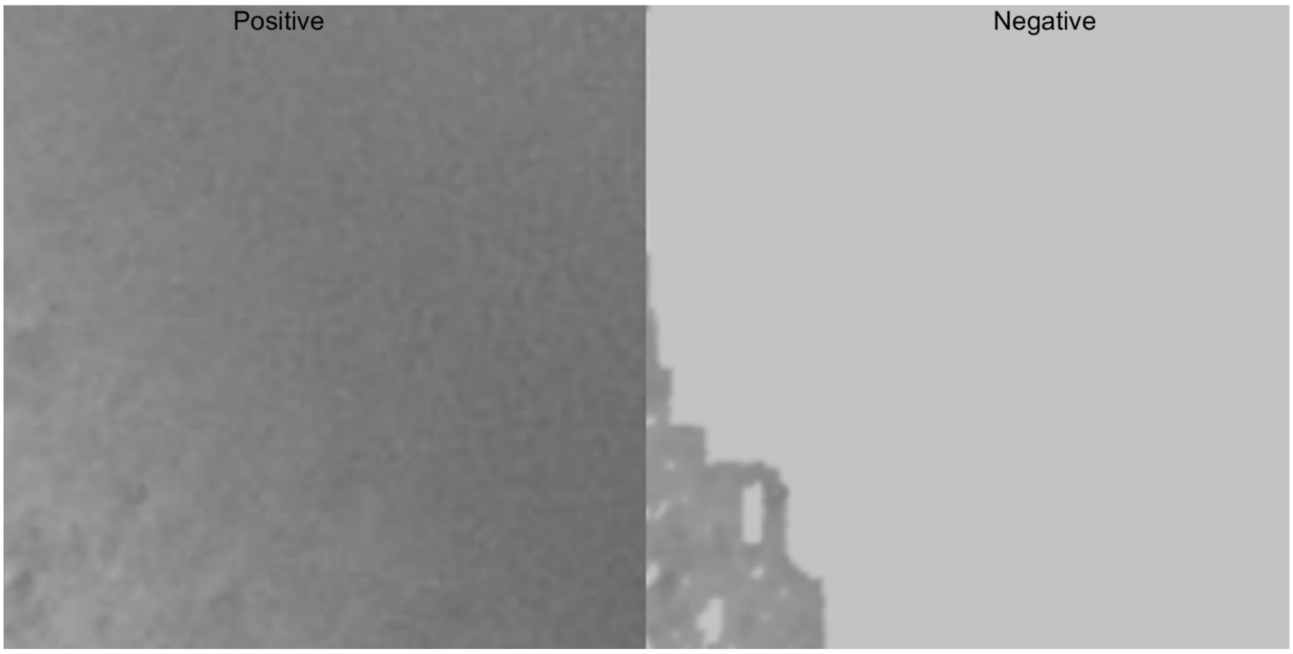


20 fig. The effect of lower resolution

Alternative was tested as well, and resolution was reduced (see figure 20). We observed that reducing resolution can make predictions better in some cases as it is reducing the impact of noise in CNN computations. However, this does not give much better results as in some cases it becomes harder to recognize a crack if decrease is very high. Yet, using weak blur filters or just zooming out picture can lead to balanced predicting. Wherefore, we point out that that models should always be tested on similar environment and devices before using them on actual detection.

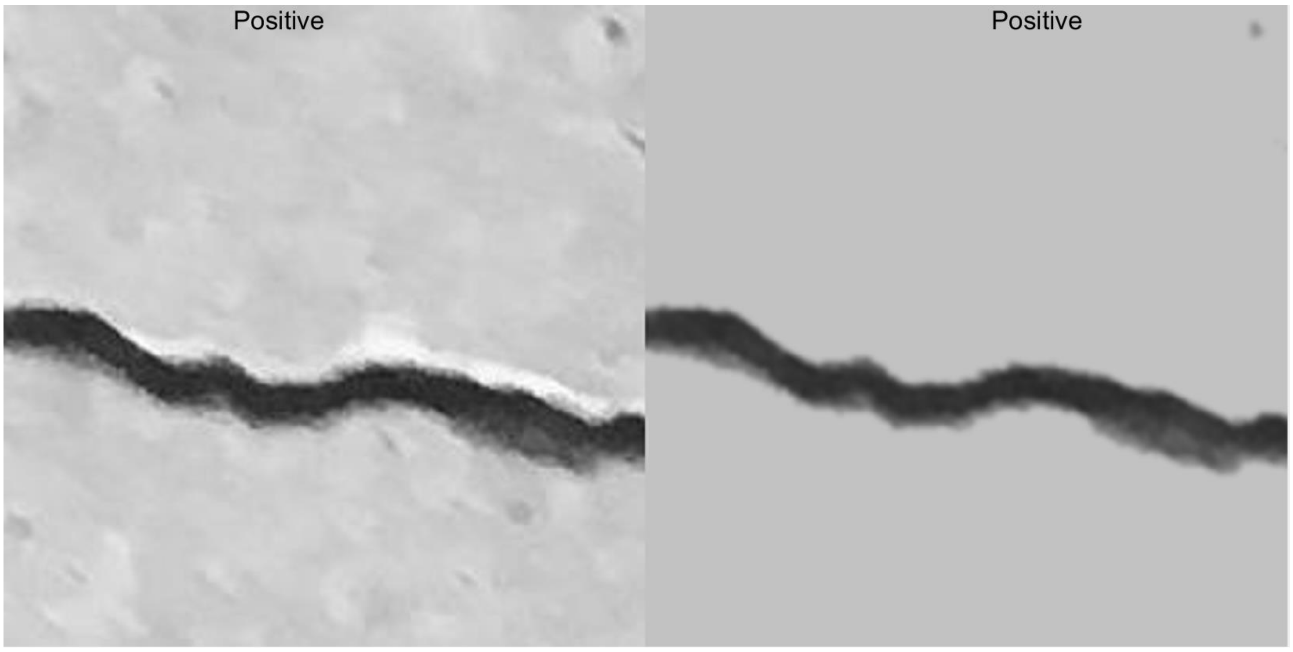
4.1.3. Results of optional pre-processing

Prediction results applying our created pre-processing\masking technique (code example is in appendix 3) are given in figures bellow.



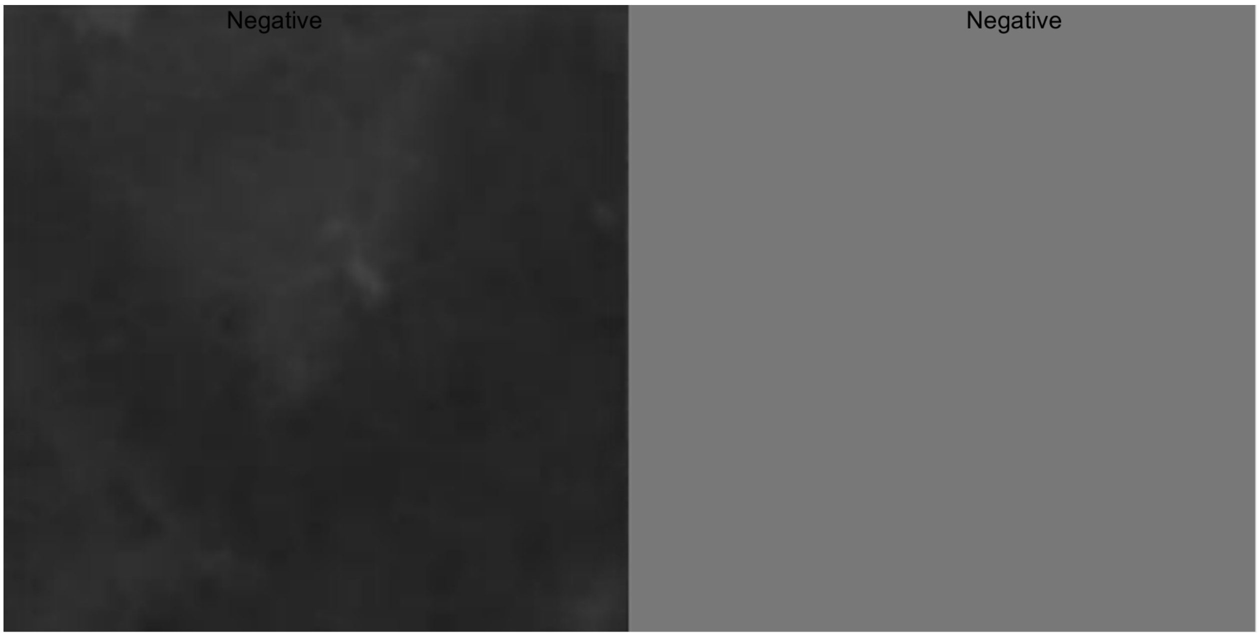
21 fig. Image with noise and same image with most of the noise removed from it

Figure 21 shows that our created algorythm removed all noise only keeping a small part of concrete wall left. It must be pointed out that this fixes prediction problems which our created model had in the first place.



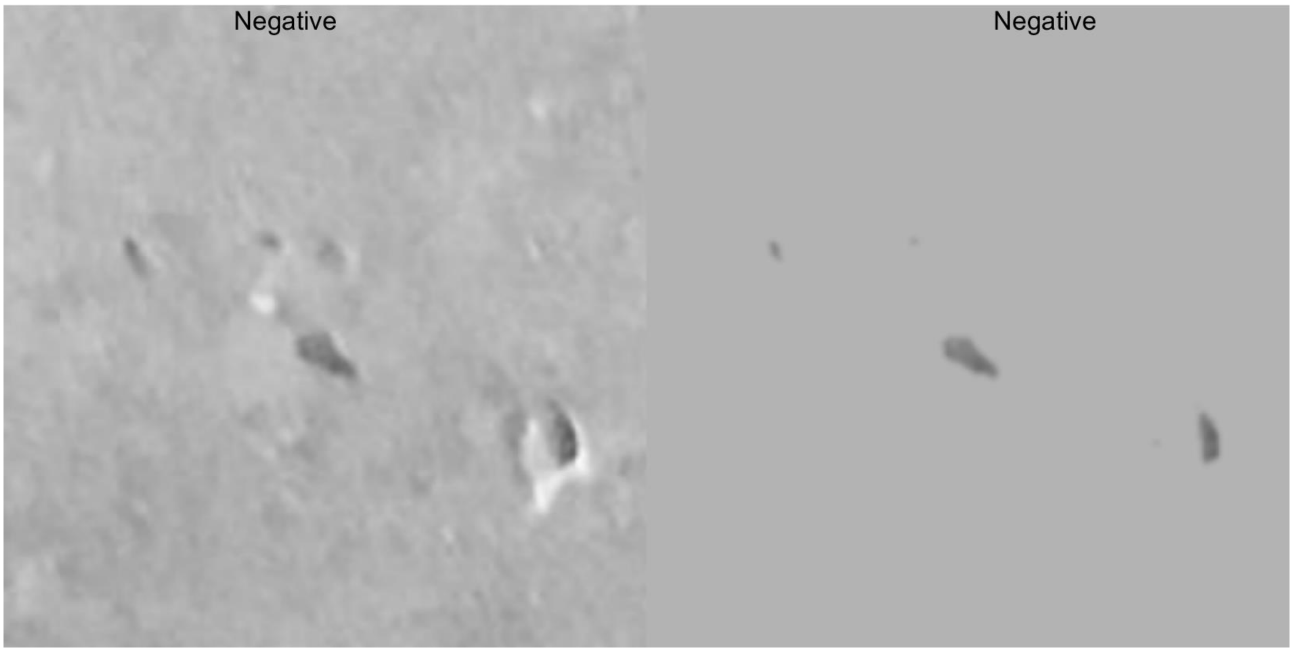
22 fig. Pre-processing algorithm effect on crack images

Main problem of data pre-processing is how to make algorithm not to remove any important information. In our case most important information which we have is the area of the crack (or suspected crack). From figure 22 we can see that only damaged area is left uncovered by the mask. This is good because model give prediction just from suspected area and in future works might be retrained to work faster by ignoring other parts of the image.



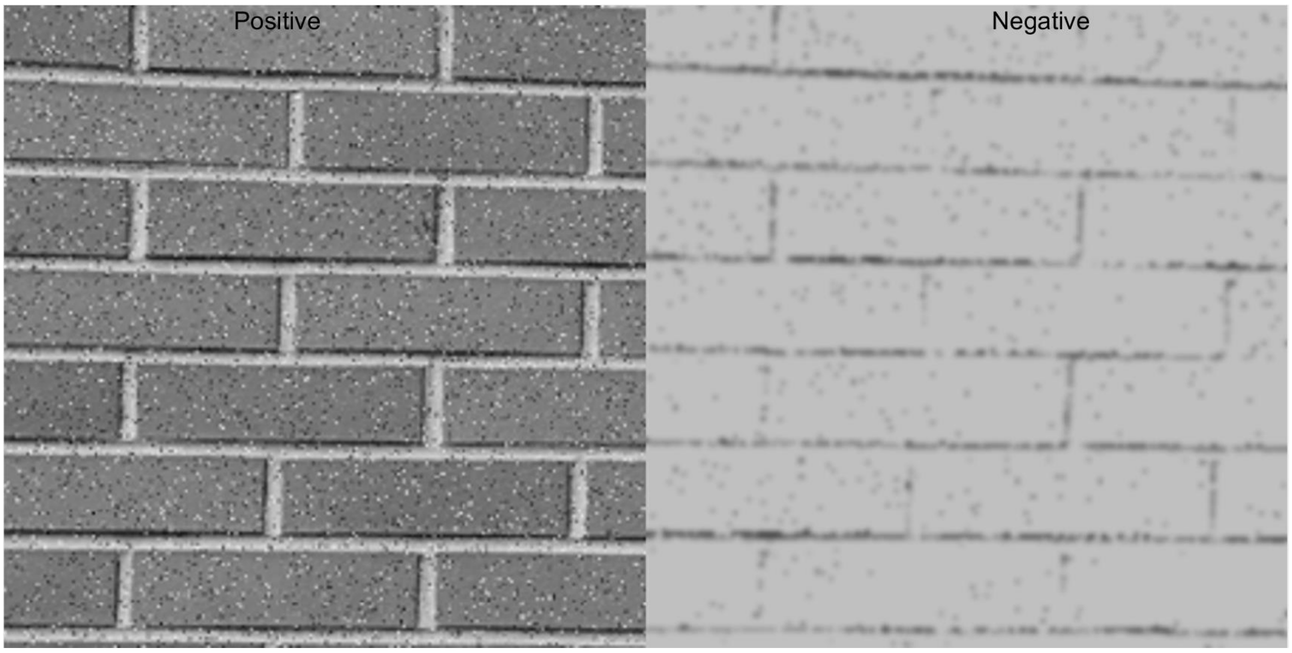
23 fig. Removing noise from dark pictures

Dark images were another problem which our network was facing. Since model had trouble performing detectiong on dark noisy walls a solution was to those wall much lighter (see figure 23) if they are not suspected to contain a crack. In addition, areas which are suspected would would still keep the original colours. This is posible as cracks and walls still keep different colour variations. However, although this technique worked well for our tested cases, it cannot be stated for sure if it should be used on dark surfaces as change in colours might be weak in some situations and this study did not have enough data of dark images without generated cracks to fully test it. This shows that more reasearch for it is needed.



24 fig. Segmentation of lightly damaged concrete walls

Problem with every wall which we observed is that at some point any area will have some kind of damage. In many cases this damaged areas do not indicate cracks but can be the features of materials from which those walls are made. A good example of the can be rock walls which would have gaps. Yet, while performing segmentation or noise removal, we do not want to cover such areas as model should decide if it is cracked area or not. One of such cases is visualized in figure 24. It shows some lightly damaged wall but we do not think that it can be classified as crack just yet.



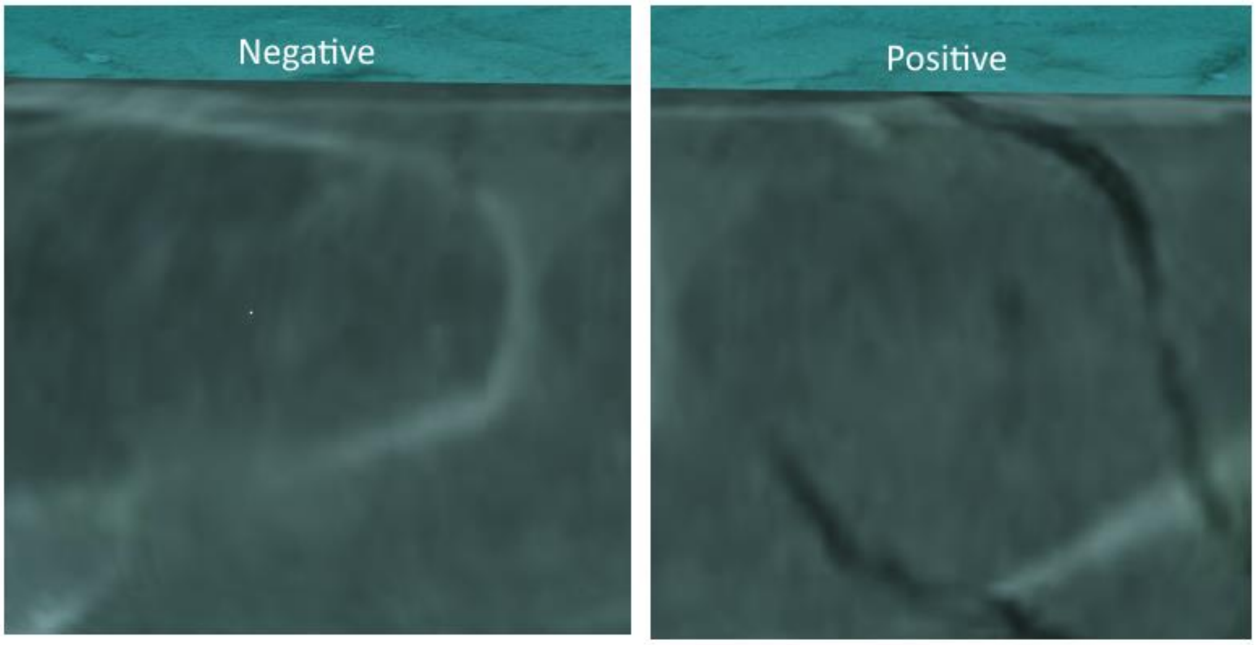
25 fig. Pre-processing tested on brick walls

On the other hand, this project contained brick walls as well and algorithm had to be tested on the also (see figure 25). However, although it can improve production of our model in some cases, but we do not recommend it to be used on brick walls as more detail research for that is needed. This is simply because noise removal method is based on picture separation into two main areas (potential crack and non-crack). Using it on surfaces which have clear repeating texture might not be appropriate as it can remove part of patterns which can make predictions more complicated in some cases (especially if surface is highly affected by shadows).

To conclude this subchapter, for almost all tested images this technic gave good results except very dark images. However, we did not have non-generated dark pictures contaning cracks, this is why we cannot without a doubt say that this method can or cannot be used on black walls. Yet, results with generated crack images shows that this method is not recomended to be applied on dark (black) walls as well as we believe that it should not be used on brick walls. On the other hand results on simple concrete walls shows that this technique improves pretictions by a lot as all tested concrete walls were classified well (except dark).

4.1.4. Testing on generated pipeline images

Pipeline system was a way to check if model would be able to identify cracks in slightly different environment. At the same time, we wanted to make system which would be able to generate cracks in situations for which gathering data would be too expensive. Underwater crack image (use in article [50]) was applied to our pipeline system and two pictures were generated. In addition, the model for both pictures give correct predictions (see figure 26). Yet, this does not mean that model can do it without mistakes as for that we would need to generate more data, but it is enough to state that our created model is capable of crack identification underwater as well. Wherefore further development can be made by training it with more generated data in future works.



26 fig. Model predictions for generated pipeline data (positive stands for crack and negative if crack was not detected)

More information about the results and how to access model for personal testing can be found online on Mendeley.com (see appendix 4). To motivate further development of the model, our study made this data publicly available and fully reproduceable.

5. Discussion

In this work we were able to create model which shows high accuracy for predicting cracks on concrete and brick walls. Yet, some problems remain which require more detail investigations. For example, our model can be tricked by noise or dark images. This means that while model shows good results in general, it can still be unstable in specific cases. We already suggested a noise removal method which might fix some of the issues, but this method requires more testing on dark surfaces as well as brick walls. Yet, we believe that expanding dataset with more similar cases (we mean images like misclassified pictures) and retraining model might solve it in the future works.

To further improve the accuracy of this model and to ensure that model would work well in the future we created Discord sever and connected it with our created model which is written in MATLAB programming language. Anyone who wishes to contribute to the project is free to visit discord server and test model by asking bot on server to check if image shows crack (a correct answer should be added after command as well), after that server bot will ask model to evaluate image and will show its prediction. In addition, this way we automatically gathering more information and obtain labelled pictures on which model can be updated periodically. This way study tried to achieve model applicability as well as move toward lifetime learning approach. This method can be useful as it removes the need to constantly search for better algorithms as it allows researchers to work together in creating universal algorithm for crack detection. This was inspired by nowadays AI approaches as chatbots become more and more popular in everyday life and while chatbots let users to send feedback about quality of the result we, in our work, allow users to send the true value which should be predicted in a way obtaining feedback before prediction. This feedback is not influencing result of model but helps to evaluate model accuracy and create dataset for periodical updates.

Conclusions

After finishing this project, we can make such conclusions:

1. Large and complicated database containing cracks on concrete and brick walls was created. It is the largest dataset compared to what was found in literature.
2. Model for the automated crack detection on our created database was developed, this was done using convolution neural networks or more specifically, it is based on AlexNet with transfer learning approach.
3. Model shows high accuracy (over 98%) which is as good as top 5 models in the literature.
4. Experimental design approach was used to find the most suitable parameters for high accuracy. The approach could also be used for proxy model generation.
5. Some challenges remain, related to false positive and false negative which required pre-processing of images. The methodology is presented as an option approach that can adjust model mispredictions which were caused by the noise in the image.
6. Finally, a simple 3D pipeline system for data generation was created and model has been tested on few generated underwater images, all of which it predicted correctly. Additional work is needed to make the pipeline crack detection system more robust.

Although we did create large database with variety of cracks and we obtain high accuracy, it cannot be taken for granted and stated that our created method it better or worse, we only say that our modification of AlexNet is able to predict cracks on images similar to those which we used in this project with over 98% accuracy. For other datasets and other challenging onshore and offshore environments further testing is needed.

Bibliography

1. SimScience. Cracking Dams: Scenarios, a. Online. [viewed 2023-04-28]. Available from: <https://sethna.lassp.cornell.edu/SimScience/cracks/advanced/failures.html>.
2. FOX, Matthew R. and Adrienne V. LAMM. Keystone Pipeline Rupture Investigation. In *Journal of Failure Analysis and Prevention*. 2021. Vol. 21, no. 3, p. 738–746.
3. HARRIGAN, R. TransCanada's Keystone XL Pipeline: Politics, Environmental Harm & Eminent Domain Abuse. Available from: <https://scholarworks.law.ubalt.edu/cgi/viewcontent.cgi?article=1010&context=ubjld>.
4. SAINATO, M. Keystone pipeline raises concerns after third major spill in five years. In *The Guardian* [online]. 2022. [viewed 2023-04-28]. Available from: <https://www.theguardian.com/environment/2022/dec/21/oil-spills-keystone-pipeline-seem-worse-kansas>.
5. BRECHER, J. and SMITH, B. Online. 2011. Available from: <https://www.labor4sustainability.org/articles/pipeline-climate-disaster-the-keystone-xl-pipeline-and-labor/>. [viewed 2023-04-28].
6. PRADO, G.M.D. Intelligent robots don't need to be conscious to turn against us. In *Business Insider* [online]. [viewed 2023-04-27]. Available from: <https://www.businessinsider.com/artificial-intelligence-machine-consciousness-expert-stuart-russell-future-ai-2015-7>.
7. CHEN, Y.; TONG, Z.; ZHENG, Y.; SAMUELSON, H. and NORFORD, L. Transfer learning with deep neural networks for model predictive control of HVAC and natural ventilation in smart buildings. In *Journal of Cleaner Production*. 2020. Vol. 254, p. 119866.
8. HOULSBY, Neil; Andrei GIURGIU; Stanislaw JASTRZEBSKI; Bruna MORRONE; Quentin De LAROUSSILHE et al. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning* [online]. [s.l.]: PMLR, 2019. p. 2790–2799. [viewed 2023-04-30]. Available from: <https://proceedings.mlr.press/v97/houlsby19a.html>.
9. NOWAK, Sebastian; Narine MESROPYAN; Anton FARON; Wolfgang BLOCK; Martin REUTER et al. Detection of liver cirrhosis in standard T2-weighted MRI using deep transfer learning. In *European Radiology*. 2021. Vol. 31, no. 11, p. 8807–8815.
10. MAY, R.M. Simple mathematical models with very complicated dynamics. In *Nature*. 1976. Vol. 261, no. 5560, p. 459–467.
11. BUCK, O. Crack Formation and Propagation. In *MRS Bulletin*. 1989. Vol. 14, no. 8, p. 16–17.
12. YAKOVLEV, Grigory and Nikolai KHOKHRIAKOV. CRACK FORMATION IN VACUUM CONCRETE. Online. Available from: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwigt9Ht0frAhUTrYsKHVBYChAQFnoECBcQAQ&url=https%3A%2F%2Fjournals.vilniustech.lt%2Findex.php%2FJCEM%2Farticle%2Fdownload%2F9109%2F7950&usg=AOvVaw298LKzlS_WSZL24Liv93m4 >.
13. CHARLES, E. A. and R. N. PARKINS. Generation of Stress Corrosion Cracking Environments at Pipeline Surfaces. In *Corrosion*. 1995. Vol. 51, no. 7, p. 518–527.
14. CHEN, W. 30 - Modeling and prediction of stress corrosion cracking of pipeline steels. In EL-SHERIK, A.M.Sud. *Trends in Oil and Gas Corrosion Research and Technologies* [online]. Boston: Woodhead Publishing, 2017. p. 707–748. ISBN 978-0-08-101105-8. [viewed 2023-04-29]. Available from: <https://www.sciencedirect.com/science/article/pii/B9780081011058000309>.
15. FANG, B. Y.; A. ATRENS; J. Q. WANG; E. H. HAN; Z. Y. ZHU et al. Review of stress corrosion cracking of pipeline steels in "low" and "high" pH solutions. In *Journal of Materials Science*. 2003. Vol. 38, no. 1, p. 127–132.

16. VAN BOVEN, G.; W. CHEN and R. ROGGE. The role of residual stress in neutral pH stress corrosion cracking of pipeline steels. Part I: Pitting and cracking occurrence. In *Acta Materialia* . 2007. Vol. 55, no. 1, p. 29–42.
17. GHOSH, Goutam; Paul ROSTRON; Rajnish GARG and Ashoutosh PANDAY. Hydrogen induced cracking of pipeline and pressure vessel steels: A review. In *Engineering Fracture Mechanics*. 2018. Vol. 199, p. 609–618.
18. XUE, H.B. and CHENG, Y.F. Characterization of inclusions of X80 pipeline steel and its correlation with hydrogen-induced cracking. In *Corrosion Science*. 2011. Vol. 53, no. 4, p. 1201–1208. .
19. ZHANG, G; Y. LIU. P. YANG; Y. BAI and X. MA. Analysis on the causes of crack formation and the methods of temperature control and crack prevention during construction of super-high arch dams. In *Shuili Fadian Xuebao/Journal of Hydroelectric Engineering*. 2010. Vol. 29, p. 45–51.
20. BHATTACHARJEE, S. S. and P. LÉGER. Seismic cracking and energy dissipation in concrete gravity dams. In *Earthquake Engineering & Structural Dynamics*. 1993. Vol. 22, no. 11, p. 991–1007.
21. PAN, Jianwen; Chuhan ZHANG; Yanjie XU and Feng JIN. A comparative study of the different procedures for seismic cracking analysis of concrete dams. In *Soil Dynamics and Earthquake Engineering*. 2011. Vol. 31, no. 11, p. 1594–1606.
22. SHERARD, J.L. EMBANKMENT DAM CRACKING. In *Publication of: Wiley (John) and Sons, Incorporated* [Online]. no. 0. Available from: <https://trid.trb.org/view/40458>. [viewed 2023-04-29].
23. SHEERIN SITARA, N. M; KAVITHA, S. and G. RAGHURAMAN. Review and Analysis of Crack Detection and Classification Techniques based on Crack Types. In *International Journal of Applied Engineering Research* . 2021. Vol. 13, no. 8, p. 6056.
24. SHI, Pengfei; Xinnan FAN and Gengren WANG. A novel underwater dam crack detection algorithm based on sonar images. In *2015 5th International Conference on Computer Sciences and Automation Engineering (ICCSAE 2015)* [online]. [s.l.]: Atlantis Press, 2016. p. 452–456. [viewed 2023-04-25]. Available from: <https://www.atlantis-press.com/proceedings/iccsae-15/25848197>.
25. WANG, T.T. Characterizing crack patterns on tunnel linings associated with shear deformation induced by instability of neighboring slopes. In *Engineering Geology*. 2010. Vol. 115, no. 1, p. 80–95.
26. LINS, Romulo Gonçalves and Sidney N. Automatic Crack Detection and Measurement Based on Image Analysis. In *IEEE Transactions on Instrumentation and Measurement*. 2016. Vol. 65, no. 3, p. 583–590.
27. DUNG, Cao Vu. and Le Duc ANH. Autonomous concrete crack detection using deep fully convolutional neural network. In *Automation in Construction*. 2019. Vol. 99, p. 52–58.
28. SIMONYAN, Karen. and Andrew ZISSERMAN. Very Deep Convolutional Networks for Large-Scale Image Recognition. Online. [s.l.]: arXiv, 2015. arXiv:1409.1556 [cs]. [viewed 2023-01-15]. Available from: <http://arxiv.org/abs/1409.1556>.
29. SZEGEDY, Christian; Vincent VANHOUCKE; Sergey IOFFE; Jonathon SHLENS and Zbigniew WOJNA. Rethinking the Inception Architecture for Computer Vision. Online.[s.l.]: arXiv, 2015. arXiv:1512.00567 [cs]. [viewed 2023-04-15]. Available from: <http://arxiv.org/abs/1512.00567>.

30. HE, Kaiming; Xiangyu ZHANG; Shaoqing REN and Jian SUN. Deep Residual Learning for Image Recognition. Online.[s.l.]: arXiv, 2015. arXiv:1512.03385 [cs]. [viewed 2023-01-15]. Available from: <http://arxiv.org/abs/1512.03385>.
31. DORAFSHAN, Sattar; Robert THOMAS and Marc MAGUIRE. Comparison of Deep Convolutional Neural Networks and Edge Detectors for Image-Based Crack Detection in Concrete. In *Construction and Building Materials*. 2018. Vol. 186, p. 1–56.
32. LIU, Jingwei; Xu YANG; Stephen LAU; Xin WANG; Sang LUO et al. Automated pavement crack detection and segmentation based on two-step convolutional neural network. In *Computer-Aided Civil and Infrastructure Engineering*. 2020. Vol. 35, no. 11, p. 1291–1305.
33. PAL, Mayur; Paulius PALEVIČIUS; Mantas LANDAUSKAS; Ugnė ORINAITĖ; Inga TIMOFEJEVA et al. An Overview of Challenges Associated with Automatic Detection of Concrete Cracks in the Presence of Shadows. In *Applied Sciences*. 2021. Vol. 11, no. 23, p. 11396.
34. HALLEE, Mitchell J.; Rebecca K. NAPOLITANO; Wesley F. REINHART and Branko GLISIC. Crack Detection in Images of Masonry Using CNNs. In *Sensors*. 2021. Vol. 21, no. 14, p. 4929.
35. LOVERDOS, D. and SARHOSIS, V. Automatic image-based brick segmentation and crack detection of masonry walls using machine learning. In *Automation in Construction*. 2022. Vol. 140, p. 104389.
36. DU, Shouji; Shihong DU; Bo LIU and Xiuyuan ZHANG. Incorporating DeepLabv3+ and object-based image analysis for semantic segmentation of very high resolution remote sensing images. In *International Journal of Digital Earth* . 2021. Vol. 14, no. 3, p. 357–378.
37. ZHANG, Chanqing; Jiang CHEN; Ying LUO; Feng XIONG and Anming XU. Crack width identification for underwater concrete structures using temperature tracer method. In *Measurement Science and Technology*. 2021. Vol. 32, no. 12, p. 125107.
38. ZHU, Yuxuan; Jiang CHEN; Yuanyuan ZHANG; Feng XIONG; Fengfei HE et al. Temperature tracer method for crack detection in underwater concrete structures. In *Structural Control and Health Monitoring*. 2020. Vol. 27, no. 9, p. e2595.
39. CHEN, Jiang; Feng XIONG; Yuxuan ZHU and Huiqun YAN. A crack detection method for underwater concrete structures using sensing-heating system with porous casing. In *Measurement*. 2021. Vol. 168, p. 108332.
40. CHEN, Dong; Ben HUANG and Fei KANG. A Review of Detection Technologies for Underwater Cracks on Concrete Dam Surfaces. In *Applied Sciences*. 2023. Vol. 13, no. 6, p. 3564.
41. Deep Trekker. Online. [viewed 2023-04-25]. Available from: <https://www.unmannedsystemstechnology.com/feature/case-study-underwater-tunnel-inspection-using-sonar-technology/>.
42. SHI, P; FAN, X; NI, J.; Z. KHAN and M. LI. Underwater dam crack classification based on the fusion of images obtained from dual-frequency sonar. Online. [viewed 2023-04-25]. Available from: <https://bio-protocol.org/exchange/minidetail?id=2089935&type=30>.
43. CAO, W. and LI, J. Detecting large-scale underwater cracks based on remote operated vehicle and graph convolutional neural network. In *Frontiers of Structural and Civil Engineering*. 2022. Vol. 16, no. 11, p. 1378–1396. .
44. ALIFF, Mohd; Nur Farah HANISAH. Development of Underwater Pipe Crack Detection System for Low-Cost Underwater Vehicle using Raspberry Pi and Canny Edge Detection Method. In *International Journal of Advanced Computer Science and Applications* [online]. 2022. Vol. 13, no. 11. Available from: <http://thesai.org/Publications/ViewPaper?Volume=13&Issue=11&Code=IJACSA&SerialNo=52>.[viewed 2023-04-26].

45. QI, ZhiLong; Donghai LIU; Jinyue ZHANG and Junjie CHEN. Micro-concrete crack detection of underwater structures based on convolutional neural network. In *Machine Vision and Applications* [online]. 2022. Vol. 33, no. 5. [viewed 2023-04-26]. Available from:<https://doi.org/10.1007/s00138 -022-01327-5>.
46. SHI, Jiajun; Wenjie YIN; Yipai DU and John FOLKESSON. *Automated Underwater Pipeline Damage Detection using Neural Nets*. 2019.
47. YANG, Jun; Wei WANG; Guang LIN; Qing LI; Yeqing SUN et al. Infrared Thermal Imaging-Based Crack Detection Using Deep Learning. In *IEEE Access*. 2019. Vol. 7, p. 182060–182077.
48. AMJAD, K.; LAMBERT, P.; MIDDLETON, C. A.; GREENE, R. J. and E. A. PATTERSON A thermal emissions-based real-time monitoring system for in situ detection of fatigue cracks. Online. Available from: <https://royalsocietypublishing.org/doi/epdf/10.1098/rspa.2021.0796>. [viewed 2023-05-02].
49. BERMAN, D.; TREIBITZ, T. and S. AVIDAN. Diving into Haze-Lines: Color Restoration of Underwater Images. Online. Available from: < http://www.bmva.org/bmvc/2017/papers/paper044/ paper044.pdf >.
50. ORINAITĖ, Ugnė; Paulius PALEVIČIUS; Mayur PAL and Minvydas RAGULSKIS. A deep learning-based approach for automatic detection of concrete cracks below the waterline. In *Vibroengineering PROCEDIA*. 2022. Vol. 44, p. 142–148.
51. BRITO, A. Blender 3D. Online. [viewed 2023-05-16]. Accessed from: <https://s3.novatec.com.br/ capitulos/capitulo-9788575222805.pdf>.
52. LESHNO, Moshe; Vladimir Ya. LIN; Allan PINKUS and Shimon SCHOCKEN. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. In *Neural Networks*. 1993. Vol. 6, no. 6, p. 861–867.
53. HAGAN, M.T. and DEMUTH, H.B. Neural networks for control. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*. 1999. p. 1642–1656 t. 3.
54. MathWorks - Makers of MATLAB and Simulink. Online. [viewed 2023-04-30]. Accessed from: <https://www.mathworks.com/>.
55. ÖZGENEL, Ç.F. and GÖNENÇ SORGUÇ, A. Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings. Surface Crack Detection. ISARC 2018, Berlin. In [online]. 2018. [viewed 2023-04-30]. Available from: <https://www.kaggle.com/datasets/arunrk7/ surface-crack-detection>.
56. Pexels. Brick Wall Cracks Photos, Download The BEST Free Brick Wall Cracks Stock Photos & HD Images. Online. Available from: <https://www.pexels.com/search/brick%20wall%20cracks/>. [viewed 2023-04-30].
57. KHEDGAONKAR, Roshni. Kavita SINGH. and Mukesh RAGHUWANSHI. Chapter 10 - Local plastic surgery-based face recognition using convolutional neural networks. In N, P. and other. Sud. *Demystifying Big Data, Machine Learning, and Deep Learning for Healthcare Analytics* [online]. [s.l.]: Academic Press, 2021. p. 215–246. ISBN 978-0-12-821633-0. [viewed 2023-01-15]. Available from: <https://www.sciencedirect.com/science/article/pii/B9780128216330000015>.
58. BOZINOVSKI, S. and FULGOSI, A. The influence of pattern similarity and transfer learning upon the training of a base perceptron B2. In *Symposium Informatica*. Croatia, 1976.
59. PAN, S.J. and YANG, Q. A Survey on Transfer Learning. In *IEEE Transactions on Knowledge and Data Engineering*. 2010. Vol. 22, no. 10, p. 1345–1359.

60. WEISS, Karl; Taghi M. KHOSHGOFTAAR and DingDing WANG. A survey of transfer learning. In *Journal of Big Data*. 2016. Vol. 3, no. 1, p. 9.

61. Matlab. Design and run experiments to train and compare deep learning networks - MATLAB. Online. Available from: <https://www.mathworks.com/help/deeplearning/ref/experimentmanager-app.html>. [viewed 2023-05-15].

62. KIM, Jiwon; Jung Kwon LEE and Kyoung Mu LEE. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* . 2016. p. 1646–1654.

Appendix(es)

1 appendix. Code for model training

```
%Script for main program
clear;clc;close all
load('netTransfer_alexnet_shadow_images.mat')
%imds                                                                    =
imageDatastore('C:\Users\Mayur\Documents\Apps\Andrius\Images','IncludeSubfolders',true,
'LabelSource','foldernames');
imds                                                                     =
imageDatastore('C:\Users\Mayur\Documents\Apps\Andrius\JPGGen','IncludeSubfolders',true,
'LabelSource','foldernames');
numExample=9;
idx = randperm(numel(imds.Files),numExample);
for i=1:numExample
   I=readimage(imds,idx(i));
   %I_tile{i}=insertText(I,[1,1],string(imds.Labels(idx(i))),'FontSize',20);
end
% use imtile function to tile out the example images
% I_tile = imtile(I_tile);
% figure;imshow(I_tile);title('examples of the dataset')

[imdsTrain,imdsValidation] = splitEachLabel(imds,0.70,'randomized');
numTrainImages = numel(imdsTrain.Labels);

%net = squeezenet;
net = alexnet;
netName = 'alexnet';
%netName = 'squeezenet';
%CANNOT USE squeezenet in my matlab 2022 version.

%net = googlenet();
%netName = 'googlenet';
analyzeNetwork(net);
% The first layer, the image input layer,
% requires input images of size 227-by-227-by-3, where 3 is the number
% of color channels.
net.Layers(1)
inputSize = net.Layers(1).InputSize;
layersTransfer = net.Layers(1:end-3);
```

```
numClasses = numel(categories(imdsTrain.Labels))
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
% layers(1:110) = freezeWeights(layers(1:110));
% lgraph = createLgraphUsingConnections(layers,connections);

pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
    'DataAugmentation',imageAugmenter,'ColorPreprocessing','gray2rgb');
augimdsValidation                                                                =
augmentedImageDatastore(inputSize(1:2),imdsValidation,'ColorPreprocessing','gray2rgb');
% augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);

% Train Network
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',4, ...
    'InitialLearnRate',1e-4, ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'ValidationPatience',Inf, ...
    'Verbose',false ,...
    'Plots','training-progress');

% The network is trained on GPU if available. It is specified by 'ExecutionEnvironment',"auto" as
above.
netTransfer = trainNetwork(augimdsTrain,layers,options);

% Classify test Images to calculate the classification accuracy
[YPred,scores] = classify(netTransfer,augimdsValidation);

YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
M = confusionmat(YValidation,YPred);
```

```
confusionchart(M,["Negative","Positive"])
acc = sum(diag(M)) / sum(M,'all');
title(sprintf("Accuracy: %g",acc));

%filename = 'MastersNetwork6.mat';
filename = 'BrickwallsCheck.mat';
save(filename)
```

2 appendix. Code for model testing

```
%clear
%load("MastersNetwork6.mat", "netTransfer");
%load('netTransfer_alexnet_shadow_images.mat', 'netTransfer');
s = 0;
imagefiles = dir('C:/Users/98and/OneDrive/Desktop/Studijos/JPGGen/Positive/*');
nfiles = length(imagefiles);
for i = 3:nfiles
   %Im = imread(strcat('C:/Users/98and/OneDrive/Desktop/Studijos/Magistras/modified/positive/',int2str(i),'.png'));
   Im = imread(strcat('C:/Users/98and/OneDrive/Desktop/Studijos/JPGGen/Positive/',imagefiles(i).name));
   pic = imresize(Im, [227, 227]);
   %pic = imresize(Im, [227, 227]);

   %New = imgaussfilt(pic,0.3);
   %pic = New;
   pic(:,:,2) = pic(:,:,1);
   pic(:,:,3) = pic(:,:,1);
   pred = classify(net, pic);
   if pred == 'Negative'
      i
      s = s +1;
   end
end
s/(nfiles-2)
```

3 appendix. Code for model testing with pre-processing

```
function [result] = KTUnet(I)
   load("C:\Users\98and\OneDrive\Desktop\Darbas\KTU\DiscordBot\MastersNetwork6.mat",
"netTransfer");
   Im = imread(strcat('Images\',int2str(I),'.png'));
   pic = imresize(Im, [227, 227]);
   [si, sj, ~] = size(pic);
   GI = rgb2gray(pic);
   EI = GI;
   for i = 1:si
      for j = 1:sj
         RGBmax = max(pic(i, j, :));
         RGBmin = min(pic(i, j, :));

         if (RGBmax - RGBmin) > 50
           % It is noise or color
           EI(i, j, 1) = 199;
         end
      end
   end

   vid = mean2(EI);
   minGI = min(min(GI));
   maxGI = max(max(GI));
   Vmin = vid/minGI;
   Vmax = maxGI/vid;

   if double(Vmin) - double(Vmax) >= 1
     %it will have a crack
     for i = 1:si
        for j = 1:sj
           if EI(i, j, 1) > vid*0.8
              % It is noise or color
              EI(i, j, 1) = max(vid,120);
           end
        end
      end
    else
      for i = 1:si
         for j = 1:sj
             EI(i, j, 1) = max(vid,120);
```

```
            end
          end
       end

      New = imgaussfilt(EI,1);
      pic = New;
      pic(:,:,2) = New(:,:,1);
      pic(:,:,3) = New(:,:,1);

      pred = classify(netTransfer, pic);

      if pred == 'Positive'
         result = 1;
      else
         result = 0;
      end
   end
```

4 appendix. All with project related data

All data such as database, models, testing cases and other information can be found on Mendeley.com. Please note that datasets are part of this thesis, for more information search for:

Ambrutis, A.; Pal, M. (2023). Gathered and generated database for crack detection on concrete and brick walls. Mendeley Data. V1. Available from: <doi: 10.17632/vr37ss8xm6.1>.