Deep Learning Approaches for Crack Detection in Bridge Concrete Structures

Daniel Einarson
Department of Computer Science
Kristianstad University
Kristianstad, Sweden
daniel.einarson@hkr.se

Dawit Mengistu
Department of Computer Science
Kristianstad University
Kristianstad, Sweden
dawit.mengistu@hkr.se

***Abstract*—Convolutional Neural Networks are among the most effective algorithms for image analysis applications. However, the accuracy of the algorithms depends on the availability of powerful computational resources and the quality of the images used to train the models. This paper investigates ways to build robust models to detect cracks in concrete structures using low resolution images and third-party datasets. Our experiments show that reducing image sizes by a factor of 4 does not significantly impact the accuracy. This is helpful to shorten execution time and hence lower cloud service costs. It is also observed that applying a model trained on one image dataset to detect cracks in images from a different source is not a trivial task.**

Keywords—Concrete Crack Detection; Convolutional Neural Networks; Deep Learning; Performance Optimization

I. INTRODUCTION

Convolutional Neural Networks (CNN) have been successfully used for image analysis in different application domains. The use of CNN models to detect cracks in concrete structures has been investigated and promising results have been achieved [1]. These models can be effectively employed to detect existing cracks and predict potential damages in roads railways, buildings, bridges, etc. [2], [3]. Machine learning models have a major advantage over the traditional manual approach used for crack detection. Manual inspection compromises safety as it exposes people to work in dangerous and inaccessible parts of the concrete structures. Automating the process by employing robots to capture images in hazardous and inaccessible areas and feeding these images to a machine learning pipeline is recommended to address this problem.

This paper presents a study that investigates ways to improve the accuracy of concrete crack detection on the Öresundsbron (the Öresund Bridge). The Bridge management is tasked with performing inspection of the bridge for preventive maintenance. The inspection involves early detection of cracks in the bridge's structures. It is intended to automate this task by using drones to take images of the concrete structure, and by applying machine learning techniques on these images for crack detection.

The objective of this study is to investigate how machine learning models can be implemented and incorporated into the Bridge's preventive maintenance system. From a research perspective, the main issues investigated in this work are classification accuracy of the model and, improvements of computational performance, mainly in the cloud. The rest of this paper is organized as follows. A brief background on the Öresund Bridge and review of CNN models is presented. This is followed by descriptions of the experimental methodology applied in this study. The achieved results are then presented along with discussions on their significance. The report concludes by summarizing the important findings and citing directions for future work.

II. BACKGROUND

A. *Brief Overview of the Öresund Bridge*

The Öresund Bridge is a combined railway and motorway bridge that links Sweden and Denmark. The bridge runs nearly 8 kilometers (5 miles) from the Swedish coast to the man-made island Peberholm. The island is connected to the Danish coast via the 4-kilometre (2.5 mi) Drogden Tunnel on its other end. Øresundsbro Konsortiet[1] is the management body responsible for maintaining the bridge, and related assets.

At present, the bridge management has a large repository of footages that were not systematically organized. Because the images were taken from afar, they lack clarity and sharpness desired for an accurate analysis. As a result, it has been difficult to draw meaningful conclusions using the repository in its present form.

A pilot study to investigate the possibilities of using CNN on this dataset was performed as a thesis work at Kristianstad University [5]. A dataset of 6,639 images (4,633 crack free and 2,006 having cracks) was available for this study. The original images in the dataset have a 6013x3376 pixels size. However, they were cropped down to 256x256 pixels for ease of analysis, inspired by the work of [1]. Fig. 1 shows few examples of images containing cracks vs crack-free.

The study showed that there are two major challenges to achieve the intended objective. First, because the study was performed on a laptop with moderate resources (CPU, memory), the model training phase took a long time. Second, it was not possible to achieve the desired level of accuracy owing to the limited size of the training dataset. The quality of the images and

[1] Øresundsbro Konsortiet - https://www.oresundsbron.com/en/info/company

the balance between image classes (number of with-crack vs. no-crack) also weighed on the accuracy of the models.

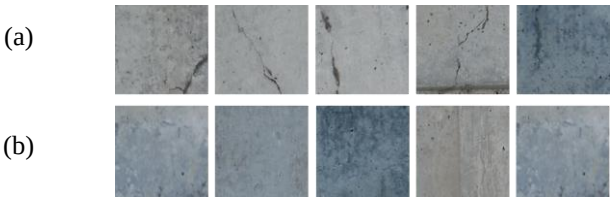

Fig. 1. Images from Öresund dataset. (a) with Cracks. (b). cracks-free

As is the case in image classification projects, tackling these limitations is of significant interest. Deploying the model on the Cloud has become the preferred approach as the Cloud proved to be an efficient high performance computing environment. To address the accuracy aspect, we build the CNN model using a third-party dataset and test the model's performance with the images from our Bridge dataset. Accordingly, we shall use the Mendeley dataset[2], which is available in a public repository and used for similar studies [4]. Fig. 2 illustrates representative images from the Mendeley dataset to show the difference between with-crack and no-crack image classes.

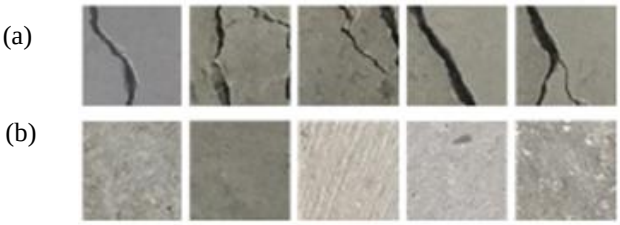

Fig. 2. Images from Mendeley dataset. (a) with Cracks. (b). cracks-free

B. Review of Convolutional Networks

While CNNs can be applied in several application domains, they have shown superior performance in image recognition, due to their built-in structures [6]. A CNN basically consists of Convolutions, Pooling, Activation Functions, and Dense Layers. These operations are performed in a number of layers through which the original input data is matched against possible output, as illustrated by Fig. 3. Such layers are often supplemented with a final *Softmax* operation before mapping to Output. For more information about this, see, [1] and [7].

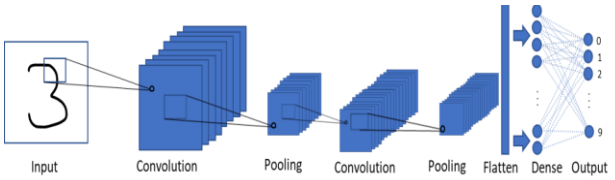

Fig. 3. A CNN to classify handwritten digits

Fig. 3 relates to the classic MNIST-example [8], which examines neural networks to recognize hand-written digits of size 28x28 pixels. A study in [8] tackles the problem on the basis of variations of the LeNet CNN-structure (Fig. 3). Although a LeNet-based CNN may reach an accuracy above 99%, the MNIST-problem may actually be solved by a significantly simpler and faster, fully connected neural network. A simple feed forward network with only one hidden layer achieves an accuracy level of about 98% for this problem [9].

Techniques relying on simple networks to detect cracks were proposed by [11]. However, this is not the case for our crack detection dataset. This is because the images from the Öresund Bridge are significantly larger and too blurry to be handled by simple networks. Therefore, the need for advanced algorithms such as convolutional neural networks is evident.

Convolutional networks are becoming increasingly complex and evolved with the growing challenges in image classification problems [12]. Earlier versions of CNN such as LeNet-5 (1998), AlexNet 2012) and VGG16 (2014) consist of a sequence of a lower number of layers. Later variants of CNNs such as Inception-v3 (2015), and ResNeXt-50 (2017) have complex structures consisting of combinations of layers organized in parallel.

Previous studies on crack detection show that CNNs of the type VGG-16 can produce accurate models [1]. It can be observed in the results in [13] that VGG-16 achieved a good accuracy on a dataset of 2500 images with 256x256 resolution. The CNN in this study has 13 convolutional, and 3 fully connected layers. An accuracy of 92.27% is achieved in 50 epochs. An AlexNet network consisting of 5 convolutional, and 3 fully connected layers proposed in [14] achieves an accuracy level of about 98%. Surprisingly, the more advanced VGG16 models actually show lower accuracy than the less advanced AlexNet.

The study in [1] gives interesting insights about the accuracy and execution performance of CNNs. The model developed in this study has 8 layers: 4 convolutional, 2 pooling, 1 ReLU, and 1 Softmax layers. The concrete images were taken at a close distance (1 – 1.5 meters from the concrete). In total, 32K images were used, achieving an accuracy of about 98% around the 50th epoch. The experiment completed in 90 minutes on two GPUs while it took 1-2 days on a standard CPU. This shows the need for accelerated computing environment, to achieve speed and performance.

A study in [4] compares several types of CNNs, such as AlexNet, VGG16, and ResNet50, based on the Mendeley image dataset, and shows impressive results, with an accuracy of above 99%. The dataset contains 40K high quality images, with 227x227 pixel resolution. The time required to train 28K dataset per epoch is, however, unacceptably high. For instance, the time required for AlexNet is shown to be 133 seconds, and for VGG16, it is 2,827 seconds.

From the above studies, it can be seen that a simpler CNN such as AlexNet can achieve an acceptable accuracy and performance. However, early experiments with the Öresund Bridge images of size 256x256 pixels showed unsatisfactory results. Worse to note that the time to train the system to get such dismal results was about 24 hours (on one machine), pointing to the clear need for alternative approaches to address the problem.

[2] Mendeley Data set for Classification of Concrete Crack Images, https://data.mendeley.com/datasets/5y9wdsg2zt/2

III. METHODOLOGY

We conducted experiments in two different computational environments, to investigate execution performance issues. The first experiment is performed on a laptop computer (we call it the local version) having 11th Generation Intel core i5-11400H, 2.30GHz processor, 12MB cache and 16 GB RAM.

The second experiment is performed in the Amazon cloud on custom-bult, accelerated computing instance of *ml.g4dn.4xlarge* type. This powerful machine has 16 virtual CPUs, 64GB RAM and a T4 GPU (2560 cores). This cloud instance is provisioned with AWS SageMaker Notebook preconfigured for deep learning in the Python language. In both cases, our training models are based on an AlexNet-type CNN consisting of six layers as follows:

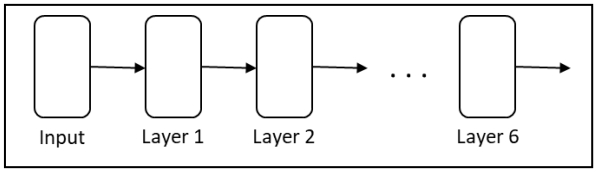

Fig. 4. A six-layer AlexNet-type of CNN

1. A Convolution layer of size 16 (C-16), an Activation function (A) of type Relu, a MaxPooling (P) of size 5x5, and a DropOut (D) of 20%
2. C-32, A-Relu, P-3x3, D-20%
3. C-64, A-Relu, P-2x2, D-20%
4. A Flatten Layer
5. A Dense Layer of size 32, and A-Relu
6. Dense-1, and a Sigmoid Activation Function finally providing output.

The experiments were run for 20 epochs, with learning rate set to 0.001 for the first 10 epochs and decreased to 0.0005 for the rest[3]. The batch size chosen is 64.

The original Mendeley dataset[4] contains 40 000 images split into 20 000 Positive (with cracks), and 20 000 Negative (without cracks). We selected some 10K images from this dataset to conduct our experiment. The following two-step approach was further employed.

1. To investigate the effect of image size on classification accuracy and execution performance, the original images were resized to different sizes and used to train the CNN (procedure A).
2. For each category of resized images, the experiment was repeated by varying the proportion of positive images (containing crack) used to train the model. This step was used to investigate the effect of unbalanced data on classification accuracy (procedure B).

Both procedures were implemented and evaluated on both the local and cloud-based experiments.

A. Resizing the images

As explained earlier, we chose 5100 positive (having crack) images as well as 5100 negative (crack-free) images from the Mendeley dataset. Three test-cases are prepared with 100 images removed from each group and saved aside for use as test images. Therefore, we have three different sets of 5000 (positive plus negative) images for training, and an additional 100 images (positive plus negative) for testing. From the training set, 20% of the images were used for validation, thus, the other 80% are used for building the CNN model.

Next, we created new datasets of lower resolution from the original 227X227 images. The new image sets have sizes of 128x128, 64x64, and 32x32 pixels. Accordingly, a test-suite of twelve different cases is created, that is, four image-sizes times three test-cases for each. The reference scale on which the images are resized is the original size of the Öresund Bridge images, which is 256x256 pixels. Halving the sizes of the images is considered significant enough to observe clear effects. The resizing is done through an *INTER_AREA* method, as described in [16].

The outcome of image resizing is illustrated in Fig. 5. A crack is seen on the original and the resized images. The assumption here is that, if cracks are clearly visible to the eye, they should certainly be detected by a CNN model as well.

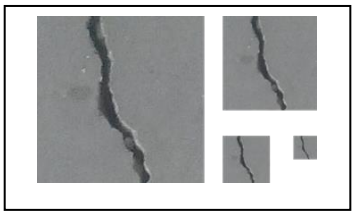

Fig. 5. Different size images with cracks

B. Unbalancing the datasets

In the real world, cracks in concrete are a seldom occurrence. This means, the number of images with cracks is certainly much less than that without cracks if the dataset is assembled naturally. This imbalance poses clear difficulties in training the CNN model in the usual way. The use of unbalanced classes of data (positive vs. negative) has obvious impact on accuracy.

In this part of the experiment, we start with completely balanced sets of images 5000 positive plus 5000 negative and proceed to lower the number of positive images in the next runs. This is achieved by successively removing positive images so that the number of positive images is 40%, 20%, 10%, 5%, 1% and 0.1% of the negative images and then rerun the test. For instance, an imbalance factor of 40% corresponds to 5000 negative, and 2000 positive images.

The accuracy figures should be regarded with special care. For instance, in a case where the number of positive images in the dataset is less than 10% of the negative images, an accuracy of 90% may not be satisfactory. Such an accuracy result could

[3] A learning rate of 0.001 is a standard setting, tuning the value is a common way to further improve accuracy.

[4] The Bridge dataset has 4,633 images considered without cracks, resp. 2,006 images considered with cracks.

lead to reporting many false negative or false positive cases. Therefore, the result must be complemented with the confusion matrix[5].

IV. RESULTS

Our results are reported below according to the structure outlined in the Methodology in Section III. The tables show size of images, time to train and validate a CNN in seconds, percentage of training accuracy (*Accuracy)*, and *Validation accuracy*, for all test cases 1-3. The details from the confusion matrix tables (*TP*, *TN*, *FP*, *FN)* are presented for both procedures (experiments with resized images and unbalanced datasets).

A. Local (laptop-based) experiments

TABLE I, shows the experiments run for the three testcases, and for the different image sizes. It can be observed that the values of the training as well as the validation accuracy are surprisingly high. That is, due to the sharpness of the images of the Mendeley dataset, even downscaled images give high accuracy.

As can be seen, the execution time and image size are not linearly related. Doubling image size by a factor of about 2, implies quadrupling the number of input nodes and hence the total number of edges will be accordingly higher. This seems to result in a difference in execution time by a factor of about 3. This is clearly demonstrated in TABLE I.

TABLE I. RESIZING IMAGES THROUGH DOWNSCALING

Case	*Size*	*Exec. Time (sec)*	*Accuracy (%)*	*Val. Accuracy (%)*
1	32	72.3	98.4	98.7
1	64	225.9	99.2	98.8
1	128	819.3	99.7	99.0
1	227	2556.2	99.8	99.3
2	32	77.9	98.5	99.0
2	64	257.7	99.3	98.9
2	128	844.0	99.4	98.3
2	227	2551.9	99.6	99.0
3	32	79.7	98.6	99.1
3	64	261.9	99.2	98.9
3	128	844.0	99.5	98.9
3	227	2567.7	99.7	98.9

The accuracy results need to be interpreted to evaluate their significance. Therefore, we generated the data in TABLE II so that conclusions can be drawn as to whether the obtained results are fair enough for all image sizes.

TABLE II. CONFUSION MATRIX FOR RESIZED IMAGES

Case	*Size*	*TP*	*TN*	*FP*	*FN*
1	32	99	96	4	1
1	64	100	98	2	0
1	128	100	99	1	0
1	227	100	98	2	0
2	32	91	100	0	9
2	64	97	100	0	3
2	128	100	99	1	0
2	227	99	99	1	1
3	32	100	97	3	0
3	64	100	97	3	0
3	128	100	98	2	0
3	227	99	100	0	1

TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative

For procedure B, image sizes of 64x64 pixels are used to run the experiments with unbalanced images sets. The results of these experiments for different levels of imbalance are shown in TABLE III. It is interesting to note the achieved high accuracy values and, low false positive and false negative rates in these experiments.

From TABLE III, it can be observed that the images of the Mendeley dataset are sharp enough to give high accuracy with unbalanced datasets. The false negative rate is low if the imbalance factor remains above 10%. It is important to note that the accuracy figure alone can be deceptive. As can be seen in the table, a test with an imbalance factor of 0.1% gives 99.9% accuracy, while its false negative rate is 100 (or predicted all crack images wrong).

TABLE III. UNBALANCING SETS OF IMAGES OF SIZE 64X64 PIXELS

Imbalance factor (%)	*Accuracy (%)*	*TP*	*TN*	*FP*	*FN*	*Exec. Time (sec.)*
40	99.5	100	96	4	1	154.5
20	99.5	99	99	1	1	134.2
10	99.4	98	100	0	2	115.5
5	99.7	96	99	1	5	114.1
1	99.6	31	100	0	69	114.2
0.1	99.9	0	100	0	100	110.5

Furthermore, the values TP, TN, FP, and FN, may be used to identify further measurements, such as *Sensitivity (Sv)*, and *Specificity (Sp)*. Those are defined as follows ([15]):

$$Sv = TP / (TP + FN) \qquad (1)$$

[5] Please, see, e.g., [15] for more information on the concepts of the confusion matrix.

$$Sp = TN / (FP + TN) \qquad (2)$$

In the context of the results shown in TABLE III, *Sv* explains how accurately images showing cracks can be identified by our model. Conversely, *Sp* refers to the model's ability to correctly identify crack free images. Table IV shows these values computed from the results shown in TABLE III.

TABLE IV. EXTENSION TO TABLE III

Imbalance factor (%)	*Sensitivity*	*Specificity*
40	99%	*96%*
20	99%	*99%*
10	98%	*100%*
5	95%	*99%*
1	31%	*100%*
0.1	0%	*100%*

It can be seen in TABLE IV that the sensitivity of the model declines drastically as the proportion of positive images (imbalance factor) in the training set is reduced. This is expected because the network is exposed mostly to negative images during the training phase. On the other hand, the model is robust for crack-free images.

B. Cloud based experiments

In this part of our work, the laptop-based experiment is repeated on the cloud albeit with different sizes of data. The aim of this task is to obtain a more accurate model by using a much larger dataset. Additionally, it is interesting to observe if the model trained on images from one concrete structure can be used for prediction in a different structure. This possibility would simplify the job of practitioners who have a limited dataset to train a new model of their own. It is also beneficial for those users whose image dataset is not yet labelled as manual labelling of a large dataset requires opening and visually inspecting each image file. Accordingly, we used images from two different sources for the test set. The first test case used the Mendeley dataset, similar to the laptop- based experiment. The second test case used images from the Bridge dataset itself. The experiment was repeated for different image sizes. The results of both test cases are shown in figures 6 and 7.

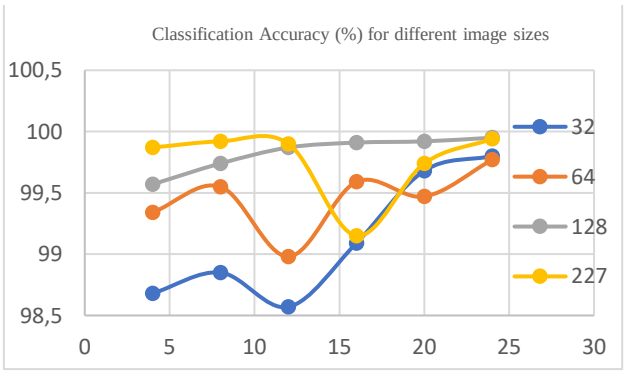

Fig 6. Experiment with Mendeley images

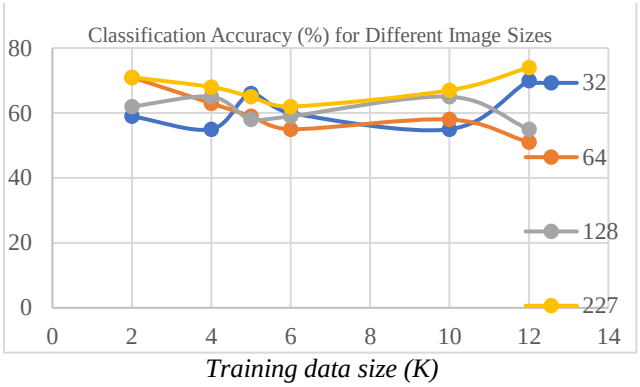

Fig 7. Experiment with Bridge images

The trained model gives a good prediction accuracy (close to 99%) when tested with Mendeley images. However, the accuracy of the model on the actual Bridge images is not satisfactory. A closer examination of the images shows that the Bridge images were blurry and therefore, the training dataset needs further processing to accommodate this fact.

Another important observation we made is that the model has an accuracy above 90% when tested with the crack-free (negative) images from the Bridge dataset itself. However, the accuracy is very low for crack containing (positive) images of the Bridge. This is an indication of a high rate of false negatives and thus our model cannot reliably detect cracks. We explored other options such as increasing the proportion of positive images, to train the model with more images showing cracks. However, the accuracy improvement is not significant.

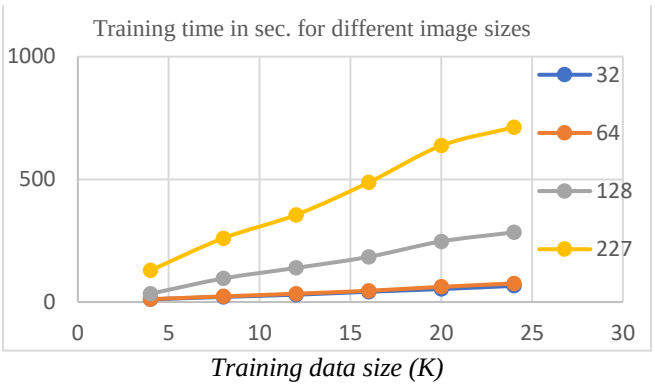

Fig 8. Execution performance on cloud

An important observation we made on the cloud execution environment relates to performance. As expected, the cloud-based implementation is much faster than the lap-top version. It could train 24K images in less than 12 minutes on the 227X227 set. This is in fact many folds improvement over the local version. It is also possible to see that the experiment with 32X32 images executes 10 times faster than the 227X227 set while there is no significant difference in their accuracies. Achieving comparable accuracies with low resolution images has a significant impact as there will be substantial reduction in cloud services cost.

Another important observation we made is that transferring large datasets from the repository to the cloud execution node could take longer time than the actual training time. This incurs significant costs specially on repeated and long-running experiments. Furthermore, due to the inherent inefficiency of Python's memory management, vectorizing the images to prepare them for training consumes a substantial part of the memory. We were able to tackle these challenges and managed to train large datasets by implementing improved data caching and memory optimization techniques.

V. CONCLUSIONS AND FUTURE WORK

This study addressed two major issues in machine learning: accuracy, and performance of CNN models in concrete crack detection use case. Because the dataset provided by the user was not in a readily usable form, the Mendeley crack dataset was used in the study. The experiment was tested on a laptop for proof-of-concept and a larger version was deployed and evaluated on the cloud. Resizing the images did not show significant loss of accuracy. The 64X64 images give an accuracy level of 99.2%, while their execution time is less than 10% of the 227X227 images.

It has been shown that the original images of 227x227 pixels can be scaled down to 64x64 pixels without any significant loss in accuracy, and with extraordinary results (model accuracy about 99.2%), compared with previously published studies. Furthermore, the effect of unbalanced datasets has been studied by progressively decreasing the proportion of positive images in the training set. It was found that the prediction accuracy is robust even when the model is trained on a dataset with only 10% positive images. Preliminary studies on the Öresund Bridge dataset also show promising results, with about 98% accuracy, on images scaled down to 64x64 pixels, and an imbalance factor in the training dataset of about 40%.

While this work can be considered a preliminary study, it yielded important results and the experimental findings opened additional research inquiries for future work. Relevant questions to ask in this regard include: *How far can one simplify the CNN model without compromising accuracy*? *How can the relation between image-sizes and CNN-structure on one side, and execution time on the other be balanced*? *How can we build an accurate prediction model on one dataset and use it on images from other sources than it trained on? How can we reduce the effect of dataset imbalance for this particular use case*?

ACKNOWLEDGMENT

The authors would like to acknowledge students involved in the initial experiments of this project. Special thanks also to the managers of the Öresund Bridge for interesting and valuable discussions and support.

REFERENCES

[1] Cha Y, Choi W, Büyüköztürk O. "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks". Computer-Aided Civil and Infrastructure Engineering. 2017;32(5):361-378.

[2] Lei Zhang , Fan Yang , Yimin Daniel Zhang, and Y. J. Z., Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016). "Road crack detection using deep convolutional neural network," 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3708-3712, http://doi.org/10.1109/ICIP.2016.7533052

[3] Soukup, D. & Huber-Mork, R. (2014), Convolutional neural ¨ networks for steel surface defect detection from photometric stereo images, in Proceedings of 10th International Symposium on Visual Computing, Las Vegas, NV, 668–77.

[4] Özgenel, Ç.F., Gönenç Sorguç, A. "Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings", ISARC 2018, Berlin.

[5] Martijn de Redelijkheid Kristian Kokoneshi, A Machine Learning Analysis of Photographs of the Öresund Bridge, Bachelor Thesis at Dept. of Compuiter Science, Kristianstad University, 2020, available at http://hkr.diva-portal.org/smash/record.jsf?pid=diva2%3A1451429&dswid=-6276

[6] LeCun, Yann & Bengio, Yoshua, Convolutional Networks for Images, Speech, and Time Series, The Handbook Brain Theory Neural Networks, vol. 3361, 1995.

[7] Saha Sumit, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, towards data science, Dec 15, 2018, available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[8] LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278–2324.

[9] Rashid T., Make Your Own Neural Network, Createspace Independent Publishing Platform, ISBN 9781530826605, 2016.

[10] Neural Network, Databricks, No date, available at https://databricks.com/glossary/neural-network

[11] Hoang N-D., Detection of Surface Crack in Building Structures Using Image Processing Technique with an Improved Otsu Method for Image Thresholding, Advances in Civil Engineering, Volume 2018 |Article ID 3924120 | https://doi.org/10.1155/2018/3924120, 2018.

[12] Karim Raimi, Illustrated: 10 CNN Architectures - A compiled visualization of the common convolutional neural networks, towards data science, Jul 29, 2019, available at: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

[13] Silva W, Lucena D. "Concrete Cracks Detection Based on Deep Learning Image Classification." Proceedings. 2018;2(8):489

[14] Kim H, Ahn E, Shin M & Sim S. "Crack and Noncrack Classification from Concrete Surface Images Using Machine Learning." Structural Health Monitoring. 2018;18(3):725-738.

[15] Géron A., Hands-On Machine Leraning with Scikit-Learn, Keras & TensorFlow. O'Reilly, ISBN 978-1-492-03264-9, 2019.

[16] Dong W., What is OpenCV's INTER_AREA Actually Doing?, Medium, 2018, available at: https://medium.com/@wenrudong/what-is-opencvs-inter-area-actually-doing-282a626a09b3