Improving Crack Detection in Concrete Structures using Augmented Data in Deep Learning Models

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

***Abstract*— While Convolutional Neural Networks are generally considered efficient structures for image processing, that task has been shown to be non-trivial in cases of datasets of imbalanced and low-quality images. Here, it is not only significant to train an algorithm towards a high accuracy enough, but also to do that in a limited amount of time. This can be understood from that low-quality datasets may require several experiments on choices of hyperparameters for an efficient network. Such experiments may be costly, not only generally in terms of time, but also in terms of payments for cloud services, as a development platform. This contribution builds on previous investigations that showed promising techniques for attacking a problem with images of cracks in bridge concrete. Results from that work is here further improved through data augmentation techniques where simply inverting images is proven to improve the performance of the network.**

Keywords—Concrete Crack Detection; Convolutional Neural Networks; Autmented Data; Performance Optimization

INTRODUCTION

The Öresund Bridge is a part of a link, together with a man-made island and a tunnel, between the very south of Sweden and Denmark. The bridge itself is about 8 kilometers (about 5 miles) and is operated by the Øresundsbro Konsortiet[1] . Here, operation means, among other things, checking out for cracks in the concrete of the bridge. Previously this was done by workers hanging in ropes, which certainly was both risky and time consuming. That method was later replaced by taking pictures of parts of the bridge by long distance cameras. Such pictures have contributed to a dataset of *6,639* images (*4,633* crack free and *2,006* having cracks), with the size of *256x256* images, however with a rather low quality. The study of [1], investigates those images through a Convolution Neural Network (CNN), where a purpose was to prepare for a next step of new modern methods more aligned with concepts of smart and sustainable cities. Here, an automated process, using drones, should detect the pieces of the bridge concrete holding cracks.

This contribution shall be seen as an experimental study that builds upon previous studies presented through [1] and aims to contribute to building blocks of knowledge to the problem context. The previous study pointed out a two-folded problem about training a CNN for analyzing images from the Öresund Bridge dataset, concerning both accuracy, and time. The low-quality images rendered CNNs with low accuracy, forcing several experiments with re-setting hyperparameters for possible improvements. Moreover, the size and the amount of the images implied several time-consuming training processes. Time is certainly generally important, but especially may lead to high expenses in the use of commercial cloud services.

The studies behind [1] addressed another approach to investigate further methods to achieve high accuracy levels more quickly. Here a third-party dataset[2] of higher quality images was used (a Mendeley dataset), and it was, for instance, shown that reducing the sizes of the images did not affect the quality of the images, thus leading to high accuracy but to a significantly lower time cost. Still, while improvements were shown also regarding the original Öresund Bridge images, it was also shown that even more investigations were needed for even further improvements, where a level of at least 95% accuracy was desired. This contribution intends to meet such requests using a rather simple technique of augmented data, here with images that are inverted to suit the CNN better.

The rest of the paper will start with a more detailed background behind the work of [1] and motivations behind the current approach. Thereafter, especially discussions on the approach to this study is provided. Discussions on aspects of time costs will be presented, as well as the main results of the implementations based on the inverted images, before a summary concludes the paper.

I. BACKGROUND

The MINST problem ([2]), that is, the classical problem of identifying handwritten digits in images of size 28x28 pixels, where the digits are in black, and the background is in white, has been solvable at an accuracy level of at least 98%. This was shown by [2] through a rather simple fully connected neural network with only one hidden layer. However, while being an effective example on image processing, such fully connected networks do not meet the challenges of more sophisticated problems, such as recognizing a specific face in a picture, or

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

[2] Mendeley Data set for Classification of Concrete Crack Images, Özgenel, Çağlar Fırat (2019), "Concrete Crack Images for Classification", Mendeley Data, V2, doi: 10.17632/5y9wdsg2zt.2

generally analyzing images of bigger sizes, such as of *256x256* pixels.

The structure of the CNN is basically inspired by the structure of the biological brain (as generally is the case for artificial neural networks), where it is recognized that e.g., the human brain uses receptive fields to identify smaller pieces of images, and where only some parts of the neurons, that is, not all connected neurons, are involved in this identification ([3])[3]. The CNN has been developed in many forms, such as the initial linear LeNet (that is, one sequence of layers, such as Convolutions, Pooling, Activation Functions, and Dense Layers [3]), and more complex variations, including several parallel layers ([4]). Such developments have proved that the CNN is capable to meet even more complex image processing challenges.

Today there are several examples where the CNN has been implemented to analyze possible cracks in structures of, for instance, buildings and roads ([5], [6], [7]) with satisfactory results. For instance, from [7] *32K* images were used, achieving an accuracy of about *98%* around the *50th* epoch, a process that completed in *90* minutes on two GPUs while it took *1-2* days on a standard CPU. Furthermore, [5] points out a case where a high quality Mendeley image dataset was processed on 28K images of size *227x227*. While the accuracy was at an impressive level of *99%*, the time per epoch was surprisingly high, *133* seconds for a simpler form of CNN (AlexNet), and *2,827* seconds for a more sophisticated CNN (VGG16) ([1]).

In contrast to the examples outlined above, the study of [1] aims not only to achieve a satisfactory accuracy level, but also with limited time expenses. As a starting point, a Mendeley dataset, as addressed above, was used, and it was shown that reducing the size of the images from *227x227* to *64x64* pixels, resulted in an accuracy of *99.2%*, and to a time cost of, all in all, *226* seconds[4]. The experiment used 10K images used *20* epochs for training and validation.

While showing impressive results regarding the high quality Mendeley image dataset, developing a robust CNN for the blurry images of the Öresund Bridge dataset was still shown to be challenging. A further approach of the current contribution is to regard the core values of the image pixels, that commonly are defined by the higher intense (brighter, e.g., white = *255*) have higher values, while the lower intense (darker, e.g., black = *0*) have lower values[5].

Still, for the MNIST images (typically represented through CSV-files), it is the opposite ([2]). An explanation to this[6] says (which is obvious) that "in Neural Network updates, the weights corresponding to a *0* in the input are not going to be updated". That is, the weight corresponding to the significant black digits would else risk to not being updated. This situation is certainly also of concern for CNNs where higher values have greater impact in updating the internal values of the CNN, than the lower values.

Yet another twist on this is to see that a typical layer of the CNN depends on the Pooling, and especially often the MaxPooling. MaxPooling reduces the amount of information and replaces that with a 'higher order' representation, where a higher priority is put on higher numerical values. Figure 1 shows an example of MaxPooling, where the highest number of a pool is chosen as a representative number. Typically, cracks are darker areas of images, and then risk disappearing when processed during the CNN. Experiments on inverting images are therefore motivated also for improving identifications of cracks in concrete.

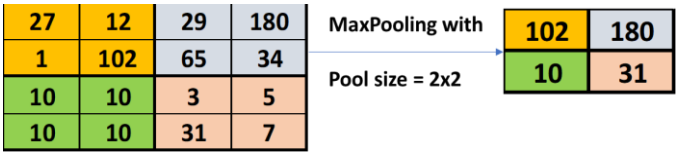

Fig. 1. A MaxPooling layer of the CNN Images from Mendeley dataset.

Furthermore, image preprocessing is a step of transforming data before training the neural network and is an important step to speed up training and improve classification [8]. Data augmentation is here referred to as techniques of image preprocessing, and according to [8] used to reduce problems of overfitting. The survey of [9] addresses data augmentation to extract more information from the dataset. Their study provides several examples on that approach, such as, color transformation, and random erasing ([9]). The investigation of the current contribution will use data augmentation, or image preprocessing, in terms of inverting the images, with respect to brighter and darker image areas. As a data augmentation technique that approach will be studied on several image datasets, to prove the potential of the technique also outside the scope of the Öresund Bridge dataset.

METHODOLOGY

The two-folded aim regarding accuracy and time was shown to be met by [1], with respect to the Mendeley dataset. Here, that aim will be further elaborated upon.

First, this contribution will study how to improve the value of accuracy, and especially for the Öresund Bridge images. Here, the following *assumption* will be a main approach: *Inverting the pixel values of the images representing cracks in concrete will improve the level of the accuracy*.

Moreover, [1] concerned how time required to train the CNN with respect to resizing the images changed. In this contribution, studies on time will be performed dependent both on the number of images in a dataset, and the sizes of those images.

3 Actually, studies on receptive fields, or the visual cortex, of brain structures were initially performed on cats and monkeys, and later led to a Nobel Prize in 1981, see, e.g., https://homl.info/71, 72, and 73, and https://www.nobelprize.org/prizes/medicine/1981/summary/

4 The experiment was run on a laptop computer, having 11th Generation Intel core i5-11400H, 2.30GHz processor, 12MB cache and 16 GB RAM.

5 On image pixel representations, see for instance, What is Pixel in Image Processing ? | myMusing or How are Images Stored on a Computer | Grayscale & RGB Image Formats (analyticsvidhya.com)

6 On inverted pixel values, https://stats.stackexchange.com/questions/220164/impact-of-inverting-grayscale-values-on-mnist-dataset

Furthermore, the previously mentioned *assumption* will be tested out on four different datasets:

a) The Mendeley dataset, experimented on by [1], with distinct and clear differences between cracked, and non-cracked images. The dataset originally[7] contains a balanced set of *40,000* images of sizes *227x227*. Here, as in [1] that dataset has been reduced to *6,600* to more be aligned with the size of the Öresund Bridge images.

b) A Mendeley dataset[8], with images with different levels of cracks, and where this contribution will focus on levels that are far less distinct in differences than those of a). The dataset of use, include *10,000* images (equally split between low crack value and no cracks), of size *227x227* pixels.

c) Images of asphalt[9], where it is harder to see distinctions between cracked vs. non-cracked images. The dataset contains *400* images (*200* with cracks, *200* without crack) of size *448x448* pixels (while not being images of concrete, it is still seen as interesting to study those since they have structures similar to those of concrete).

d) The Öresund Bridge images are originally of size *256x256* pixels. The dataset contains *6,639* images, *2,006* with cracks, and 4 633 without cracks.

Figure 2 shows examples of images from a) – d) above, ordered similarly, and without cracks above, and with cracks below. Furthermore, Figure 3, shows the same images as those of Figure 2, however, inverted.

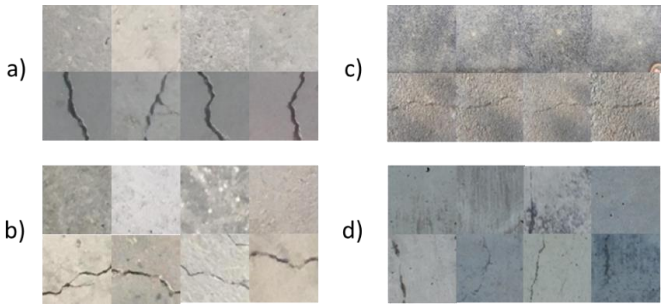

Fig. 2. Datasets samples a) - d), without, and with cracks.

The CNN that will be used for the experiments is the same the one of [1] for the sake of consistency in the discussions. Still, *25* epochs (instead of *20*) will be used in these experiments for the purpose of clearer effect. In short, the CNN of use is a six-layer AlexNet with layers for Convolutions, MaxPooling, etc. Please, see [1] for more information on the structure.

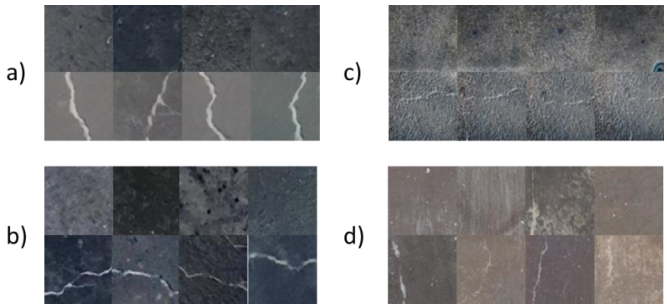

Fig. 3. Datasets samples inverted a) - d), without, and with cracks.

IMPLEMENTATION AND RESULTS

This section will start with studies concerning the cost in time. Thereafter the results based on inverted images will be presented.

A. Results based on Inverted Images

Experiments to see the effect of training the CNN on inverted images have been performed on each of the datasets of *a) – d)* pointed out inf Figures 2, and 3. The result of the experiment is outlined in TABLE I. For each dataset, experiments have been done according to:

- The original images, with the original sizes (ORIG)
- The original images inverted (ORIG_INV)
- Original images downscaled to *64x64* pixels (64)
- The 64x64 pixel images inverted (64_INV)

TABLE I captures information on the experiments including values of accuracy (ACC), validation accuracy (VAL), time in seconds (TIME), size of images (SIZE), and number of images of dataset (AMOUNT).

TABLE I shows improvements for all experiments when the images are inverted, thus motivating the previously stated assumption. For instance, dataset d), the Öresund Bridge images, do imply clear improvements when inverted. A conclusion from those experiments therefore encourages to new experiments on the Öresund Bridge images, where the process of scaling up the epochs for training the CNN may be done with a higher confidence. Several experiments show that the value of the accuracy of the Öresund Bridge images improves to a level of about *96 - 97*% which is a significant improvement from previous experiments.

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

8 Levels of cracks, Qi, Zhanfeng (2020), "Concrete cracking level", Mendeley Data, V1, doi: 10.17632/bs7rjwywfm.1

[9] Asphalt images, A, Jayanth Balaji; G, Thiru Balaji; M S, Dinesh; Nair, Binoy; D. S, Harish Ram (2019), "Asphalt Crack Dataset", Mendeley Data, V2, doi: 10.17632/xnzhj3x8v4.2

TABLE I. EXAMPLES ON DATASETS A) - D), ORIGINAL SIZES, 64X64, AND THOSE INVERTED.

	DATASET	ACC	VAL	TIME	SIZE	AMOUNT
a)	ORIGINAL	0,99567	0,98417	1338,19	227	6600
	ORIG_INV	0,998117	0,99548	1293,156	227	6600
	64	0,992846	0,98568	190,1024	64	6600
	64_INV	0,994541	0,99548	178,0259	64	6600
b)	ORIGINAL	0,996125	0,9915	2069,76	227	10 000
	ORIG_INV	0,99875	0,998	2301,628	227	10 000
	64	0,9915	0,9925	301,1926	64	10 000
	64_INV	0,995625	0,995	287,7996	64	10 000
c)	ORIGINAL	0,696875	0,6125	352,4197	448	400
	ORIG_INV	0,859375	0,525	342,29	448	400
	64	0,796875	0,7	8,709253	64	400
	64_INV	0,85625	0,8875	8,132479	64	400
d)	ORIGINAL	0,8718	0,7046	4884,816	256	6639
	ORIG_INV	0,9142	0,8651	4681,186	256	6639
	64	0,856	0,8455	203,187	64	6639
	64_INV	0,8756	0,8831	194,991	64	6639

TABLE II points out *3* experiments running on *100* epochs with a dataset of *6,639* images as previously addressed. As can be seen, a robust and repeatable, rather high accuracy results from this. For reasons of clarifications a mean value is calculated for the accuracy (*ACC*, *VAL*, as for TABLE I), as well as the cost in time (*TIME*).

TABLE II. THE ÖRESUND BRIDGE IMAGES, 64X64 PIXELS, INVERTED, WITH ACCURACY AND TRAIN TIME.

	ACC	VAL	TIME
CASE 1	0,96687	0,9254	852,7985
CASE 2	0,96216	0,91409	807,0407
CASE 3	0,96913	0,92012	761,9552
MEAN	0,966053	0,91987	807,2648

The experiments did execute for *100* epochs, within a timeframe of about *800* seconds. The matter of time aspects and the cost in time will be further elaborated on in the next sub-section, and where those values will be discussed within that context.

B. *On Time Cost*

With the structure of the CNN set (concerning number of layers, sizes of those, batch size, etc.), the time required to train the CNN mainly depends on the size of the datasets, and the size of the images. Certainly, the calculation on time cost only counts for that specific CNN, and not to CNNs generally.

While keeping the image size fixed, e.g., to *64x64* pixels, an experiment on dataset b) with the images downsized (from *227x227* to *64x64*), show time performance on subsets of *1000*, *2000, ... 10,000* equally distributed in cracks, and no-cracks images. The experiment was processed three times, and the mean values calculated. Not surprisingly, the result describes a linear relation between dataset sizes and time in seconds, as illustrated by Figure 4. Here, for instance, *1000* images take about *16* seconds to process, *5000* images take about *99* seconds, and *10,000* images takes about *209* seconds to process. Figure 4, furthermore illustrates the linear correspondence, and its function. It may furthermore be pointed out that even though the experiments on dataset sizes only regard time and not accuracy, still accuracy even for the dataset with a lower number of images all exceeds 96%.

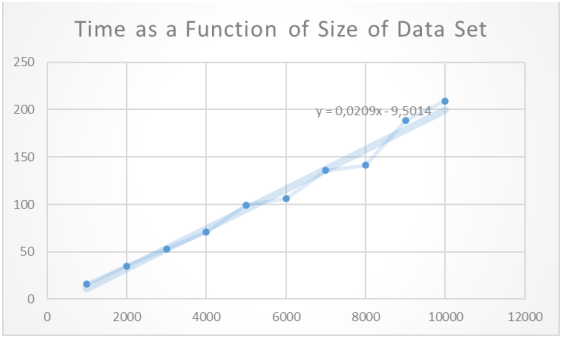

Fig. 4. Relation between number of images in dataset, and time in seconds.

With the linear function of Figure 4, *time* = ***c*** * *size-of-dataset* + ***m*** (here ***c*** = *0.02*, ***m*** = *-9.50*), experiments have shown similar relations for images, for instance, of sizes *128*, *256*, *512*, and *1024*. Here such experiments have resulted in ***c***, and ***m*** being, respectively (*0.06*, *52.18*), (*0.49*, *-310.50*), (*1.10*, *-252.02*), and (*7.36*, *-6595.56*). Certainly, while some values of ***m***, may stand out, it is still more interesting to see that the value of ***c*** obviously correspond to the size of the images, where larger image sizes imply larger values of ***c***. A reason behind this is that the image sizes have impact on the number of parameters of the CNN that have to be trained[10]. In the case of the CNN of this contribution, it can be shown that image sizes *64*, *128*, *256*, *512*, and *1024* will lead to the number of parameters of the CNN to be trained are about *16K*, *33K*, *115K*, *538K*, *and 2244K*, respectively. While CNN sub-activities, such as, *MaxPooling*, do not generate trainable parameters, the number of those parameters certainly indicates increasing time cost with respect to increasing image size. Still, in the continuation, a focus will be on experiential studies on actual time to train the CNN.

Experiments were performed in [1] on images with sizes of sides *32*, *64*, *128*, and *227*. The *227* corresponds to the full-scale image size of dataset ***a)***, still the values (*32*, *64*, *128*) were chosen to correspond more to the dataset of ***d)***, where the full-

[10] To calculate the number of parameters of CNNs, see for instance articles: Understanding and Calculating the number of Parameters in Convolution Neural Networks (CNNs) | by Rakshith Vasudev | Towards Data Science or How to Calculate the Number of Parameters in Keras Models | by Yong Cui | Towards Data Science

scale size of side is *256*. For instance, [1] showed that when the side of the image was enlarged with a factor of *2*, the time required to process the dataset seemed to increase by a factor of a bit more than *3*. For instance, image size 32 give and a process time *72* sec, *64* → *226*, and *128* → *819* seconds. Thus, *226/72* → *3.13*, and *819/226* → *3.62*.

This contribution experiments further to see the confidence of that specific assumption. Here, again, the dataset of ***b)*** has been used. Note that only the time performance is of interest here, not the accuracy. In this experiment, again, image sizes of *64, 128, 256, 512,* and *1024* have been used, and with dataset sizes of *1000, 2000, … 10000*. An excerpt from this, uses dataset sizes according to TABLE III, that shows time cost for sizes of images, and datasets.

TABLE III. TIME COST FOR IMAGES WITH SIZES 64, 128, …, AND OF DATASET SIZES 1000, 2000, …

	1000	2000	5000	8000	10 000
64	23,65668	47,74755	91,87879	213,5121	257,4784
128	88,18373	182,0169	315,5408	545,3614	592,6076
256	275,9084	569,6021	2313,174	3944,792	4865,823
512	1102,966	2046,988	5501,197	6874,302	11823,87
1024	5444,371	9469,534	29415,38	50985,02	72618,64

To measure the growth of time with respect to bigger image sizes the relative difference is here introduced. That is, this represents the value of the time cost for one image size divided by the time for the smaller image, such as time for *128x128* images divided by time for *64x64* sized images. Again, the dataset sizes *1000, …10 000* are considered, as presented in TABLE IV. What can be seen from TABLE IV is that the previous estimated factor being a bit more than *3*, is rather a bit more than *4*, where a choice may be factor, for instance, $f = 4.2$, to also take into account results of previous studies of [1].

TABLE IV. THE RELATIVE DIFFERENCES OF TIME COSTS OF TABLE I

	1000	2000	5000	8000	10000
128/64	3,727646114	3,81206727	3,434315968	2,554241052	2,30158145
256/128	3,12879011	3,129392074	7,330825446	7,233355228	8,210869741
512/256	3,997580698	3,593716284	2,378202508	1,742627075	2,429983789
1024/512	4,936118224	4,626080922	5,347087741	7,416755859	6,141697038
MEAN	3,947533787	3,790314137	4,622607916	4,736744803	4,771033005
MEAN of MEANS:		4,37364673			

While only estimations, the above tables and diagrams may be guiding according to trends regarding time cost to train the CNN based on sizes of, on one hand images, and on the other hand datasets. An attempt towards formalizing a function that generally provides the value of the time cost, should certainly also be considered only an estimation, still with a value of being guiding regarding further studies.

Choosing factor $f = 4.2$, and $time_{64} = 23.7$ (the time cost to train the CNN for images of size *64*, and dataset size of *1000* images), then implies that a $size = 2^n * 64$ gives a time cost of $f^n * time_{64}$ (by size we here refer to the side of an image). Here, generally, with $n = log_2(size/64)$, and furthermore (with a general *size*, and $time_{size}$,) then $time_{size} = f^{log_2(size/64)} * time_{64}$.

The formalization so far, only regards differences in time cost based on image sizes, that is, not dataset sizes. As previously was outlined, this is a linear function dependent on the size of the datasets. That is, with $time = \mathbf{c} * size\text{-}of\text{-}data\text{-}set + \mathbf{m}$, and if $\mathbf{m}$ is disregarded (for reasons of simplicity), then, with $\mathbf{c} = f^{\log2(size/64)} * time_{64}$, and with image $size = i$, and dataset $size = d$, then a function $\boldsymbol{time(i, d) = f^{log_2(i/64)} * time_{64} * d / 1000}$. The last division by *1000* is explained by that the expression $\boldsymbol{f^{log_2(i/64)} * time_{64}}$ already takes *1000* into account, that is, *d* needs to be compensated with *1/1000* when multiplied with the dataset sizes (*1000, 2000, …).*

Note that the experiments of this section do not address exact values regarding the relations between sizes on the one hand, and time cost on the other hand. Still, from the results above a conclusion on an approximation of a function from size of dataset, and image size, to time can be drawn according to, the above-mentioned function. TABLE V illustrates how values of TABLE I would have been calculated on the basis of function ***time(i, d)***.

TABLE V. VALUES OF TABLE I BASED ON TIME FUNCTION **TIME(I, D)**.

	1000	2000	5000	8000	10000
64	23,7	47,4	118,5	189,6	237
128	99,54	199,08	497,7	796,32	995,4
256	418,068	836,136	2090,34	3344,544	4180,68
512	1755,886	3511,771	8779,428	14047,08	17558,86
1024	7374,72	14749,44	36873,6	58997,76	73747,2

Putting the results on time costs of previous sub-section in the context of the proposed formula for approximative time cost shows about *157* seconds *(157,34)* for *25* epochs, and (times *4*) about *630* (*629,38*) seconds for *100* epochs. While being about *177* seconds less than the actual value (*807* seconds), it still points out a time span that is reasonable for training the CNN, but certainly also a need for improving the precision in the estimation of time cost.

Certainly, there are deviations regarding the values between TABLE III and TABLE V, even though the trends can clearly be seen. While being experimental studies so far, it can also be seen that further studies are needed for higher precisions. Still, a comment on the study is that the experiments of TABLE III was done in steps of *1000* images, that is *1000, 2000, 3000, … 10000* images, and for the image sizes mentioned in TABLE III. That process, all in all, required about *425 281* seconds, which is about *118* hours, and which corresponds to almost *5* days (and nights in between). Anticipating the required amount of time using the function ***time(i, d)***, and exemplifying only through the values of TABLE V, would get a total amount of seconds = *251,469.7*, which is about *70* hours, and then corresponding to almost *3* days (and nights). With that knowledge in advance the experiment would perhaps be highly questionable. A conclusion

therefore is that while only approximative, the value of developing this kind of function should be seen as two-folded:

1. It is a part of the analyses of the efficiency of the proposed CNN.
2. Using the same CNN in similar situations should be gained by a function that outlines the required time resources

SUMMARY

This contribution has considered a hard case of detecting cracks in concrete of the Öresund Bridge between Sweden and Denmark. The dataset of images representing cracks or no-cracks of the concrete was however shown to be rather blurry which led to long development time in approaches to a Convolutional Neural Network (CNN) that process those images to a reasonable accuracy. Still, with disappointing results, new methods needed to be approached.

It has previously been shown that reducing the sizes of the images does not affect the accuracy in a negative way, thus saving a lot of time to train the CNN. Moreover, this paper especially contributes with techniques of data augmentation, that is, image preprocessing to improve performance of the CNN through extracting more information from the dataset. Based on inherent structures of the CNN, inverting the images has been shown to improve the accuracy even further. Thus, the problem of the blurry Öresund Bridge images was shown to be solvable at a level of at least 96%. Moreover, time is generally considered important, especially when development is performed at a platform of a commercial cloud service. This contribution does moreover present ways of measuring the training time in advance for the examples handled here. Therefore, a conclusion is that the high performance regarding the accuracy, as well as the timing aspects of this contribution have potentials to be guiding to similar experiments, based on the same CNN structure, or similar.

Finally, it shall also be pointed out that, while the case of blurry images of cracks is an inspiring challenge, such images may actually not address critical occasions. For finding cracks that truly are critical, there is a need for further studies supported by further guidance from experts and problem domain owners. The rather simple technique of inverting images may also be complemented with more sophisticated techniques for reducing background noise, and contrast enhancement and filtering.

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] Einarson, D., Mengistu, D., Deep Learning Approaches for Crack Detection in Bridge Concrete Structures, Proceedings of the 2022 International Conference on Electronic Systems and Intelligent Computing, ICESIC 2022. Institute of Electrical and Electronics Engineers (IEEE), 2022.

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

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

[4] 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

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

[6] 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

[7] 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.

[8] Gu S ., Pednekar M., Siater R., Improve Image Classification Using Data Augmentation and Neural Networks, SMU Data Science Review, V. 2, No 2, 2019, available at: https://scholar.smu.edu/cgi/viewcontent.cgi?article=1091&context=datasciencereview

[9] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0