Surface Crack Detection Using Hierarchal Convolutional Neural Network

Davis Bonsu Agyemang and Mohamed Bader

Abstract. Cracks on surface walls may imply that a building possesses problems with its structural integrity. Evaluating these types of defects needs to be accurate to determine the condition of the building. Currently, the evaluation of surface cracks is conducted through visual inspection, resulting in occasions of subjective judgements being made on the classification and severity of the surface crack which poses danger for customers and the environment as it not being analysed objectively. Previous researchers have applied numerous classification methods, but they always stop their research at just being able to classify cracks which would not be fully useful for professionals such as surveyors. We propose building a hybrid web application that can classify the condition of a surface from images using a trained Hierarchal-Convolutional Neural Network(H-CNN) which can also decipher if the image that is being looked is a surface or not. For continuous improvement of the H-CNN's accuracy, the application will have a feedback mechanism for users to send an email query on incorrectly classified images which will be used to retrain the H-CNN.

1 Introduction

A variety of methods have been used to detect surface cracks in walls. One method is known as "Edge Detection" which detects lines or edges that are present in the image as stated by Amer and Absushaala in (Amer & Abushaala, 2015, p. 1). One of the common edge detection techniques is called the "Sobel Operator" which detects surface cracks by applying a filter (matrix of values) on the image that contains the crack as stated by Amer and Absushaala in (Amer & Abushaala, 2015, p. 1). Machine and deep learning methods have also been applied in this area due to its capabilities to learn the images' pixels containing the defect to classify new images according to Silva and Lucena in (Silva & Lucena, 2018, p. 1).

Classifying a building based on the walls as safe or dangerous is extremely vital for surveyors and their customers. This task requires intensive analysis of each surface wall found within the building as the thickness/thinness and shape of the crack can determine its structural condition which is severe if misinterpreted or incorrectly recorded, leading to possible persecution of the surveyor according to Danso in (Danso, 2018) and Neale in (Neale, 2018). Surveyors currently inspect surfaces manually through visual examination as there is no current tool that could aid them in recording and classifying surface conditions from our knowledge. Due to this implication, a tool must be implemented to mitigate these weaknesses according to Kunal Killemsetty in (Kunal & Killemsetty, 2014, p. 64).

This paper proposes a solution of creating an hybrid web application using Hierarchical Convolution Neural Networks(H-CNN) consisting of 2 CNNs(Convolutional Neural Networks) in which the first CNN classifies: random objects(not a surface), blank(blank surface), thick cracks, thin cracks the second CNN classifies: horizontal, vertical and diagonal cracks. This was achieved by using the combination of surface crack datasets collected by Özgenel Fırat Çağlar in (Özgenel, 2018) and SDNET2018 image dataset in (Dorafshan, Thomas, & Maguire, 2018) in which the image dataset for home objects from Caltech in (Caltech, 2006) was used for the random object's class. The reason why 2 surface crack datasets were used was to train the H-CNN on a variety of surface cracks which the 2 image datasets possessed, this would hopefully help it classify better. We decided to use the Caltech image dataset in (Caltech, 2006) as it represents the objects that would be present in a building, so when the surveyor accidentally takes a picture of a bottle, for instance, the H-CNN should be able to know that the image is not a surface. From our knowledge, no previous researchers have tried to classify the exact classifies specified for this paper, making our chosen classes novel.

The application will also have the capability to allow surveyors to give a query via the "feedback mechanism" if the application incorrectly classified an image to promote continuous improvements for the accuracy of the application. The user's query will be sent to us via email which will have the attachment of the misclassified image with the information on what classification should have been according to the user. That image will be used to retrain the appropriate H-CNN level to improve the model's accuracy. When building the H-CNN each CNN will be measured against the test accuracies. The whole application will be build using Python, Flask Keras TensorFlow backend for the back-end and the creation for the H-CNN, and HTML (Hypertext Mark-up Language), Bootstrap CSS (Cascading Style Sheet), and JavaScript will be used for the front-end(User interface).

2 Literature Review

2.1 H-CNN

One of the most challenging problems in CNN is when 2 or more classes share visual similarities according to Seo and Kyung-shik in (Seo & Kyung-shik, 2019, p. 331). This is difficult due to CNN being a discriminative neural network, meaning that it distinguishes the correct classification amongst the other classes according to the works of Dai and Nian Wu in (Dai & Nian Wu, 2015). So, classifying between an apple and an orange is much challenging than an apple and a car, due to apples and oranges looking similar. An H-CNN (also known as HD-CNN) which solves this problem by first separating the classes that are easier to differentiate from one another for instances an apple and a bus. This process is known as the initial coarse classifier according to Seo and Kyung-shik in (Seo & Kyung-shik, 2019, p. 331). Once the image has passed the coarse classifier is then passed to the fine classifier to generate the final classification.

However, from our research, many researchers who dealt with surface crack detection used 1 deep CNN model which is in the next sub-sections of this chapter. This could be due to the main disadvantages of the H-CNN having to train each CNN within the hierarchy while also having to fine-tune the hyperparameters for each CNN architecture such as deciding the number of convolution filters. This could be time-consuming due to the empirical nature of configuring hyperparameters. H-CNN can only have a maximum of 2 CNNs in its hierarchy as it only uses a coarse and fine classifier category to perform classifications. Nevertheless, this leads to Branch Convolution Neural Network(B-CNN) or Multi-branch Convolutional Neural Network (MB-CNN) which is multi-layered H-CNN that produces multiple output layers probability predictions from the coarse classifier levels to the fine classifier levels and based on the total predictions a final classification can be made with more granular information according to Zhu and Bain in (Zhu & Bain, 2017, p. 2) and Aslani et al in (Aslani, et al., 2018, pp. 1-2).

When traditional CNNs are only trained on 1 topic for instance fruits dataset (banana, oranges and apple classes), it only knows that these classes exist so when a user inputs an image of a bus it can only classify it based on available classes it was trained upon, meaning that it will classify the bus image based on the class that resembles it the closest. This is problematic when it comes to defect detections in building as in future if CNNs are going to be attached to robots to capture and classify images from places that are too dangerous places for humans to enter such as severely damaged or polluted buildings, it is important that can recognise if an image does not belong to any of its classes otherwise it will give false information to the user. Having an H-CNN, one could add another class that contain random images that do not relate to the topic of the application, so the H-CNN's first layer(coarse classifier) can check if the image is related to the topic for instance fruit if it is not classified it as the random image class. The only issue with this approach is that the developer would need to ensure that none of the images in the random image class dataset resembles the topic the H-CNN wishes to classify. This would require one to manually pre-process the image dataset. Still, one of the future use-cases for H-CNN could be for anomaly detection.

2.2 Related Work

As mentioned before in this paper, neglecting a building's wall condition can pose problems for the surveyor, the customer and the environment. The problems with current literature regarding this area relates to the usability for surveyors and the lack of in-depth of information derived for the classifications made from their proposed methods.
Ellenberg et al in (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014, p. 1788) used a drone to capture images from masonry building walls to detect surface cracks by using MATLAB to apply the edge detection technique: "Prewitt edge detection" which are vertical and horizontal filters containing values that removes the background of the image to only preserve the edges(surface crack) according to Adlakha et al in (Adlakha, Adlakha, & Tanwar, 2016, pp. 1483-1484). "Percolation" was used to reduce the noise in the image for instance extremely bright images according to Ellenberg et al in (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014, p. 1793). Previously without using drones to capture images, Hu et al in (Hu, Tian, Yang, Xu, & Wang, 2012, pp. 597-598) also used "Prewitt edge detection" but combined it with "adaptive threshold" which dynamically makes some pixels values of an image more prominent if these values are above the threshold value, and less prominent if they are below the threshold. This preserves the surface cracks in the image.

One problem Ellenberg et al in (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014, pp. 1793-1794) faced was the quality of their images once the percolation algorithm was applied since it made the surface cracks appear thinner which affected their methods ability to classify accurately.
Both Ellenberg et al in (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014, pp. 1793-1794)) and Hu et al in (Hu, Tian, Yang, Xu, & Wang, 2012, p. 599) edge detection methods struggled to classify cracks when foreign objects were presents in the image, for example, a surface crack that was near the "edge of a window". As mentioned earlier in this paper, this problem could have been mitigated by using a pooling technique if CNNs were used instead of edge detection as it provides spatial invariance, meaning it is not affected by the position of the surface crack it just needs to be present. Failing to capture a defect due to spatial invariance will lead to unreliable building inspection as Ellenberg et al in (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014, pp. 1793-1794)) and Hu et al in (Hu, Tian, Yang, Xu, & Wang, 2012, p. 599) would limit how surveyors can take images of surface cracks. Using drones to capture images could also pose issues like bad weather conditions can damage the drone and limit users' control of the drone according to Ellenberg et al in (Ellenberg, Kontsos, Bartoli, & Pradhan, 2014, p. 1794) negatively influencing the quality of the images. This leads to bad quality training dataset and the cost of maintenance or being forced to purchase a new one when the drone is damaged, making this method not cost-effective.

Recently, Hoang in (Hoang, 2018, p. 1) combined the edge detection "Otsu method "and "Min-Max Gray Level Discrimination (M2GLD)" to detect surface cracks. The Otsu method automatically finds the best threshold value for an image according to Otsu in (Otsu, 1979 , p. 66) and M2GLD can reduce the grayscale intensity of the image, making the surface cracks appear darker in the image, and increase the grayscale intensity to make non-surface cracks to appear lighter, enabling the image to be distinguishable according to Hoang in (Hoang, 2018, p. 5). Hoang in (Hoang, 2018, p. 9) method produced good results as it was able to detect the test images accurately. However, for a user to use their method they would need to fine-tune 2 parameters which are the ratio and the margin parameter. Hoang (Hoang, 2018, p. 9) seemed to assume that users would have background knowledge regarding these 2 parameters which is not the case, this tool may be complex for some users which will result to them not using it.

Kim and Cho in (Kim & Cho, 2018, p. 1) use CNN for surface crack detection on walls using the AlexNet architecture consisting of 5 “convolutional layers followed by max-pooling layers, and three fully-connected layers”. The CNN was trained on a multi-class dataset containing classes such as “cracked”,” joint/edge multiple lines (ML)”, “joint/edge single line (SL)”, “intact surface” and “plant”. The results of this method were outstanding, achieving a test accuracy of 96.64% (3.36% error rate). However, the problem with the works of Kim and Cho in (Kim & Cho, 2018, p. 1) and even the researchers mentioned earlier in this chapter, is that they do not realise that user such as surveyors also need to report to their customer about the severity based on the aesthetics of the surface crack such as shape and thickness/thinness. Hoang's (Hoang, 2018, pp. 1-4) other work in (Hoang, 2018, pp. 1-4) almost achieves this by treating this as multi-class problem using SVM(Support Vector Machine) algorithm to detect "longitudinal crack”, “transverse crack"," diagonal crack","spall damage","intact wall" (Hoang, 2018, p. 1). SVM creates a hyperplane that is a line which separates each class based on their characteristics ensuring that separation between classes is at the maximal distance possible. Hoang in (Hoang, 2018, p. 1) attained a test accuracy of 85.33%(14.67% error rate) using 100 image samples per class. Even though Hoang's work in (Hoang, 2018, pp. 1-4) method treated the problem as a multi-class task, the researcher did not think about training their model to detect if the image is not a surface at all as the CNN cannot recognise if an image does not belong to its classes. For this, an H-CNN would be required to achieve that functionality.

3 Methodology

3.1 H-CNN Architecture Used to Detect surface Crack

This section illustrates the H-CNN architecture that will be used for this paper as well as how it will be integrated with the front-end using Python, Flask, HTML, Bootstrap CSS and JavaScript. This architecture uses tradition CNN components such as the convolutional layer, max-pooling and fully connected layer for each H-CNN level. The CNN at each level transforms any uploaded image input to the size of 90x90. The rationale behind this size was to decrease the latency of the computations performed by the trained H-CNN when users upload a new image, as mobile systems have less computational power in comparison to the desktop. Using large image sizes such as 224x224 can increase the accuracy of the CNN according to Wu et al in (Wu, Yan, Shan, Dang, & Sun, 2015, p. 2). However, this requires many computational resources which will increase the individual CNNs training time due to having to handle multiple images at a large size. This will also slow down the application’s ability to classify when used in real-time as performing the CNN operations takes longer on larger images.

Rectifier Linear Unit(RELU) activation function was used in each convolution layers and hidden layers of the fully-connected layers for the purpose of non-linearity to allow the 2 CNNs to learn complex features within an image.

Since the application handles multiple classes, a SoftMax layer is used within the final fully-connected output layers. SoftMax distributes the probabilities throughout each class in which the class with the highest probability will be CNN’s classification result according to (Lucke & Sahani, 2007, pp. 657-659). Below in figure 1 is a depiction of the H-CNN architecture and table 1 and 2 shows hyperparameter structure and values used in the CNN levels:

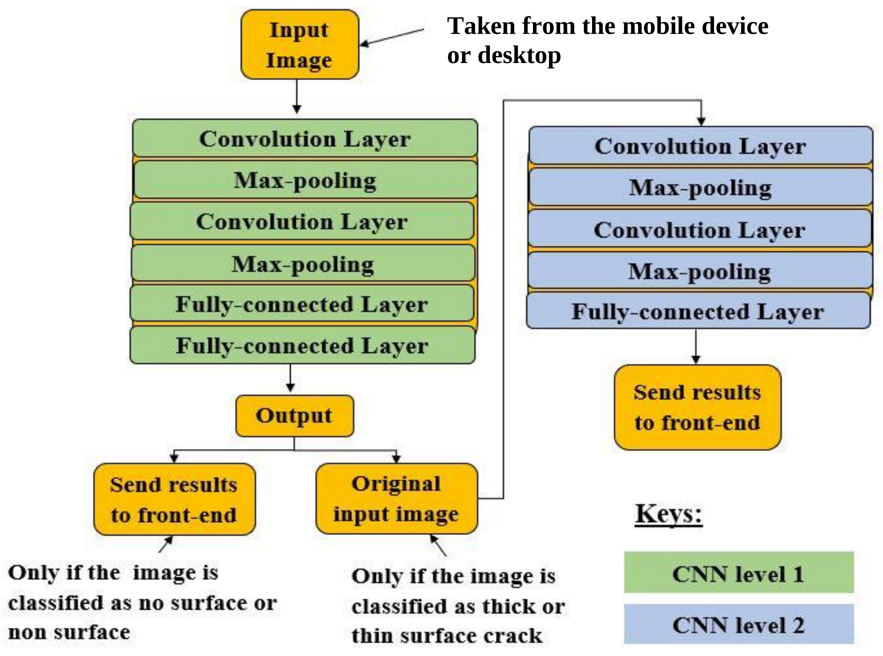
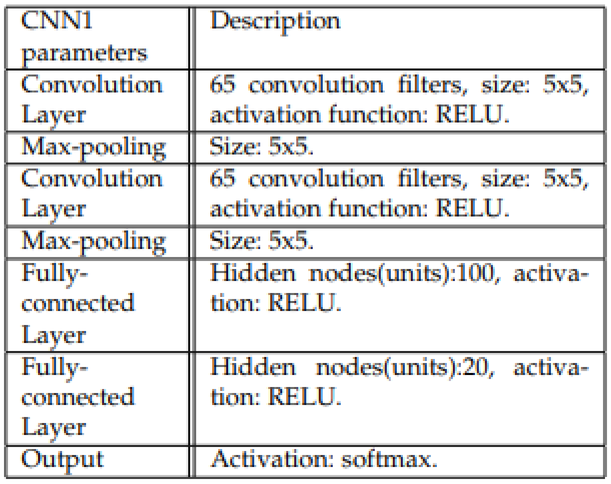
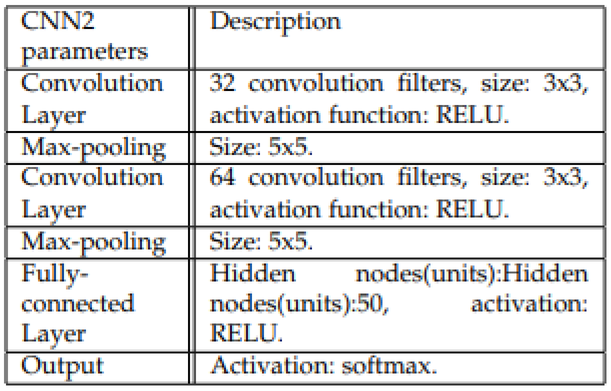


Figure 1. H-CNN architecture interaction with the surveyor's mobile device or desktop.



CNN1 parameters	Description
Convolution Layer	65 convolution filters, size: 5x5, activation function: RELU.
Max-pooling	Size: 5x5.
Convolution Layer	65 convolution filters, size: 5x5, activation function: RELU.
Max-pooling	Size: 5x5.
Fully-connected Layer	Hidden nodes(units):100, activation: RELU.
Fully-connected Layer	Hidden nodes(units):20, activation: RELU.
Output	Activation: softmax.

Table 1. CNN1(coarse classifier) parameters are orders as the architecture in figure 1.



CNN2 parameters	Description
Convolution Layer	32 convolution filters, size: 3x3, activation function: RELU.
Max-pooling	Size: 5x5.
Convolution Layer	64 convolution filters, size: 3x3, activation function: RELU.
Max-pooling	Size: 5x5.
Fully-connected Layer	Hidden nodes(units):Hidden nodes(units):50, activation: RELU.
Output	Activation: softmax.

Table 2. CNN2(fine classifier) parameters are orders as the architecture in figure 1.

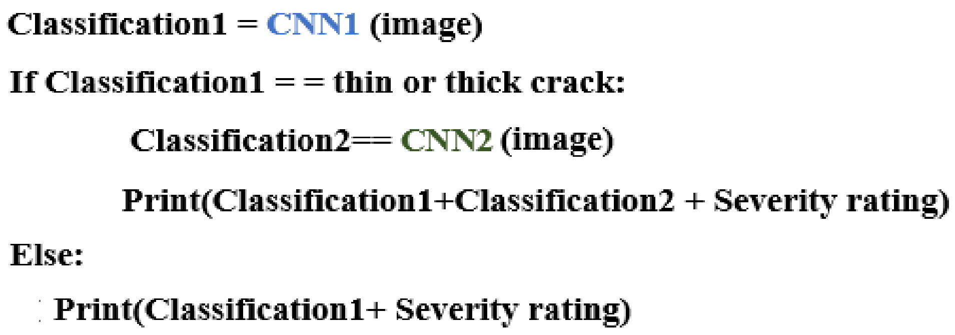
The image classes for CNN1 were:

- Random image: this is any image that is not related to surfaces. This will be used to help the application to detect images that are not related to surfaces for instance a spoon or a shoe.
- Blank: this is a surface which possess no cracks.
- Thin cracks: these are surfaces that contain thin hairline cracks.
- Thick cracks: these are surfaces that contain large thick cracks.

If the CNN1 detects a thin or thick crack in the image, CNN2 will then perform classification to determine the shape of the crack. The image classes of CNN2 are:

- Vertical crack
- Horizontal crack
- Diagonal crack

Below in figure 2 shows pseudo code of how the H-CNN will classify images from users:



```
Classification1 = CNN1 (image)
If Classification1 = = thin or thick crack:
        Classification2== CNN2 (image)
        Print(Classification1+Classification2 + Severity rating)
Else:
   Print(Classification1+ Severity rating)
```

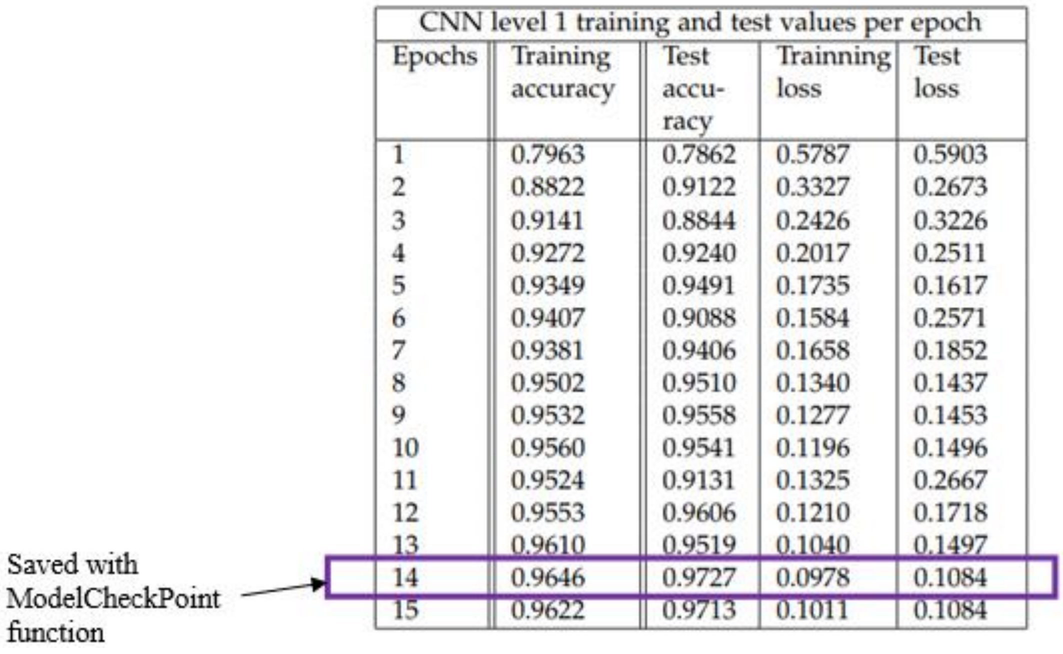
Figure 2. Pythonic pseudo code for the H-CNN.

3.2 Training and Testing Process for the CNNs for the H-CNN

The H-CNN was trained on 3 images datasets, containing a total of 920 (230 per class) images for CNN1 and a total of 390(130 per class) images for CNN2. The batch size of CNN1 was 20 which meant that CNN1 would learn the patterns of 20 images at a time during training (Radiuk, 2017, p. 20). CNN1's number of epochs(cycles) was 15 in which in 1 cycle it would learn 20 images from the training set. CNN2 used a batch size of 10 for 10 epochs during the training process. The batch sizes for the test set was the same size for CNN1 and CNN2 as Keras TensorFlow backend simultaneously classifies the images in the test set during training which meant that the epochs were also the same. We chose these batch sizes and epochs to rapidly train the CNNs as in future it needs to be able to retrain on the incorrectly classified images from the "feedback mechanism" at a fast rate so that users do not have to wait for long to use the updated H-CNN. Data augmentation was used on the training set images, which transformed the images to help the CNNs learn the patterns of the images from a different position which is important as the user may take a photo from different angles. The optimizer used for the 2 CNNs was Adam at learning rate 0.001 to speed up the training process, this is important for retraining the H-CNN in the future at a fast rate from the images gathered from the feedback mechanism. The best fully connected layer weights and convolution filter values for each CNN will be saved using keras' "ModelCheckPoint" function. This ensures that the CNN with the highest test accuracy is saved for the H-CNN.

4 H-CNN Test Accuracy Results

Testing the architectures through the performance metrics specified in the introduction was beneficial as it enabled us to see if the CNNS would classify correctly most of the time. Table 3 and figure 3 and 4 shows the training and test accuracy and loss for CNN 1 and Table 4 and figure 5 and 6 shows the training and test accuracy and loss for CNN 2:



CNN level 1 training and test values per epoch				
Epochs	Training accuracy	Test accu-racy	Trainning loss	Test loss
1	0.7963	0.7862	0.5787	0.5903
2	0.8822	0.9122	0.3327	0.2673
3	0.9141	0.8844	0.2426	0.3226
4	0.9272	0.9240	0.2017	0.2511
5	0.9349	0.9491	0.1735	0.1617
6	0.9407	0.9088	0.1584	0.2571
7	0.9381	0.9406	0.1658	0.1852
8	0.9502	0.9510	0.1340	0.1437
9	0.9532	0.9558	0.1277	0.1453
10	0.9560	0.9541	0.1196	0.1496
11	0.9524	0.9131	0.1325	0.2667
12	0.9553	0.9606	0.1210	0.1718
13	0.9610	0.9519	0.1040	0.1497
14	0.9646	0.9727	0.0978	0.1084
15	0.9622	0.9713	0.1011	0.1084

Saved with ModelCheckPoint function

Table 3. CNN1 training and test values.

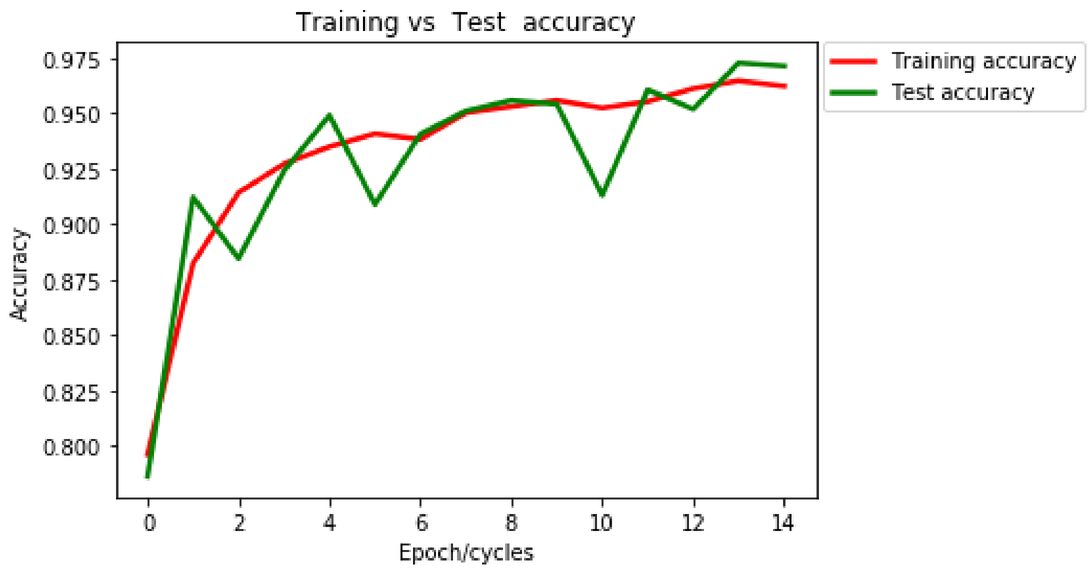


Figure 3. Training vs test accuracy for CNN1.

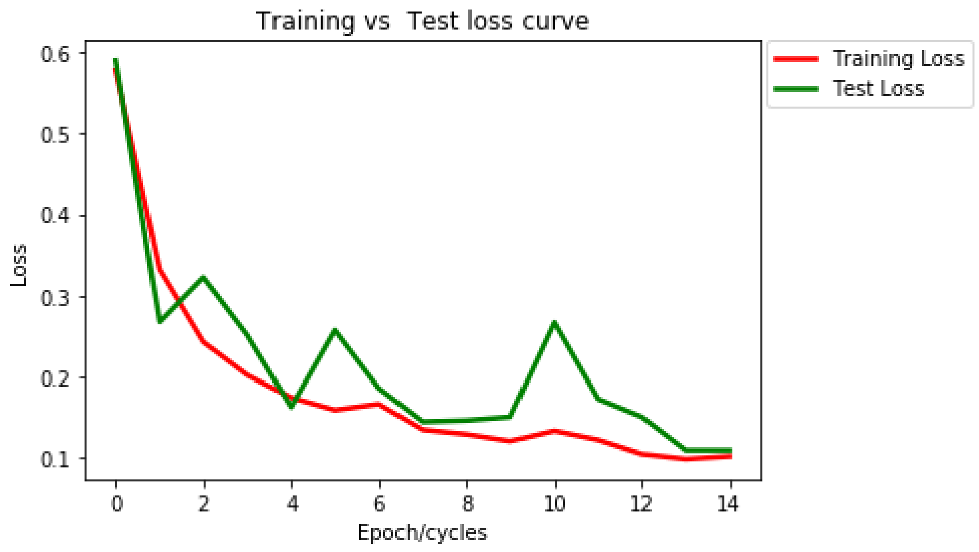
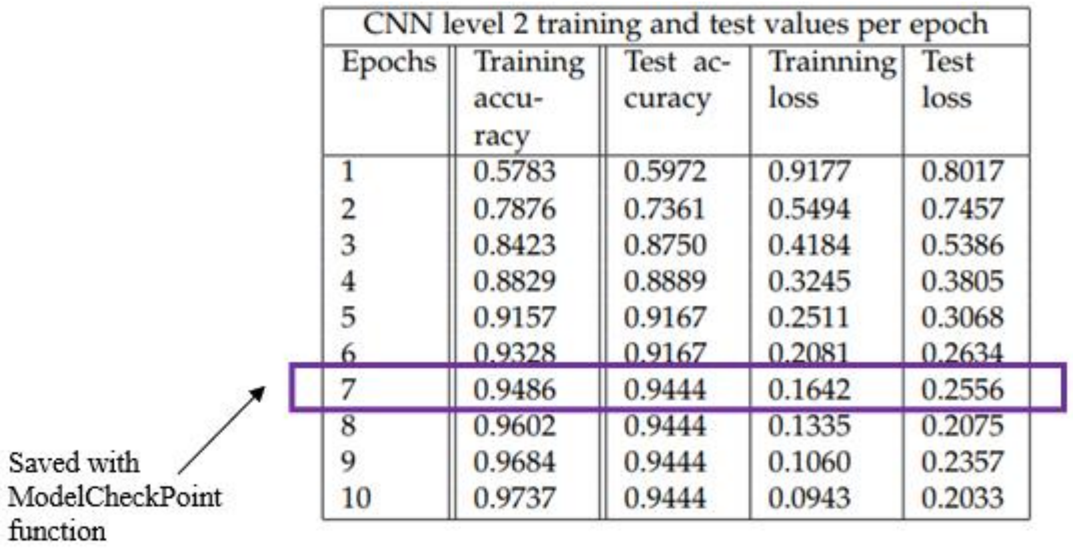


Figure 4. Training vs test loss for CNN1.



CNN level 2 training and test values per epoch				
Epochs	Training accu-racy	Test ac-curacy	Trainning loss	Test loss
1	0.5783	0.5972	0.9177	0.8017
2	0.7876	0.7361	0.5494	0.7457
3	0.8423	0.8750	0.4184	0.5386
4	0.8829	0.8889	0.3245	0.3805
5	0.9157	0.9167	0.2511	0.3068
6	0.9328	0.9167	0.2081	0.2634
7	0.9486	0.9444	0.1642	0.2556
8	0.9602	0.9444	0.1335	0.2075
9	0.9684	0.9444	0.1060	0.2357
10	0.9737	0.9444	0.0943	0.2033

Saved with ModelCheckPoint function

Table 4. CNN2 training and test values.

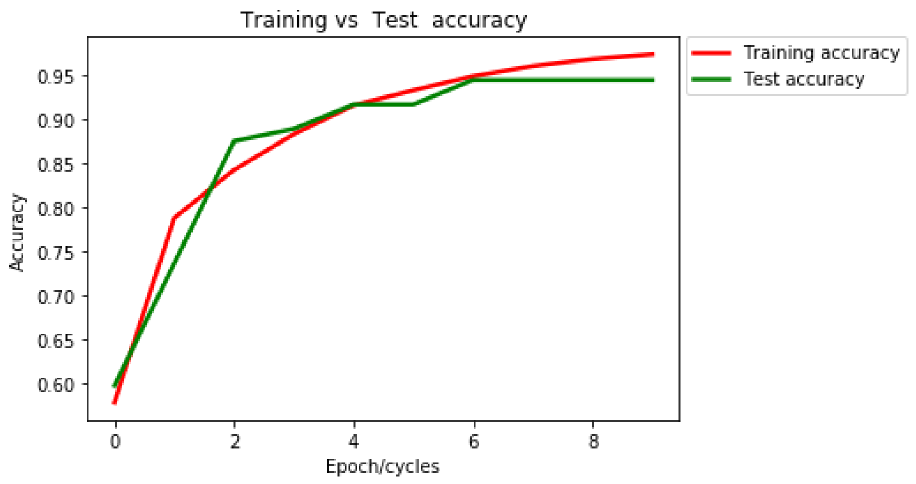


Figure 5. Training vs test accuracy for CNN2.

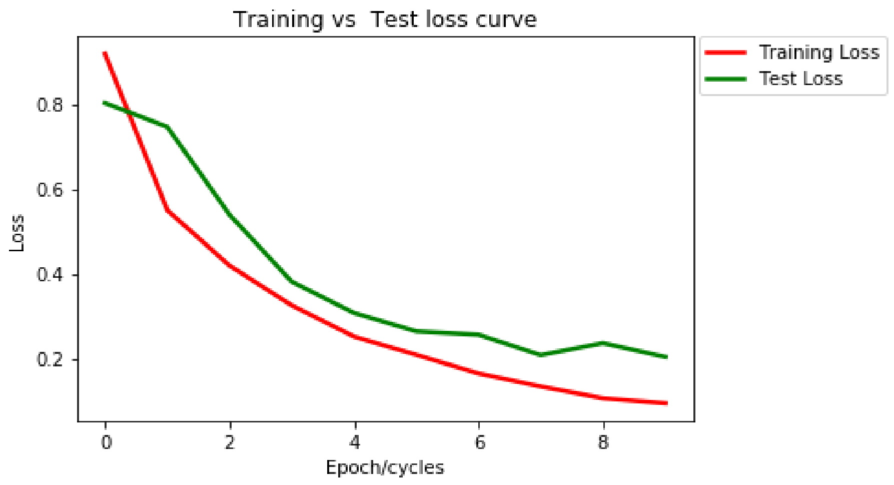


Figure 6. Training vs test loss for CNN2.

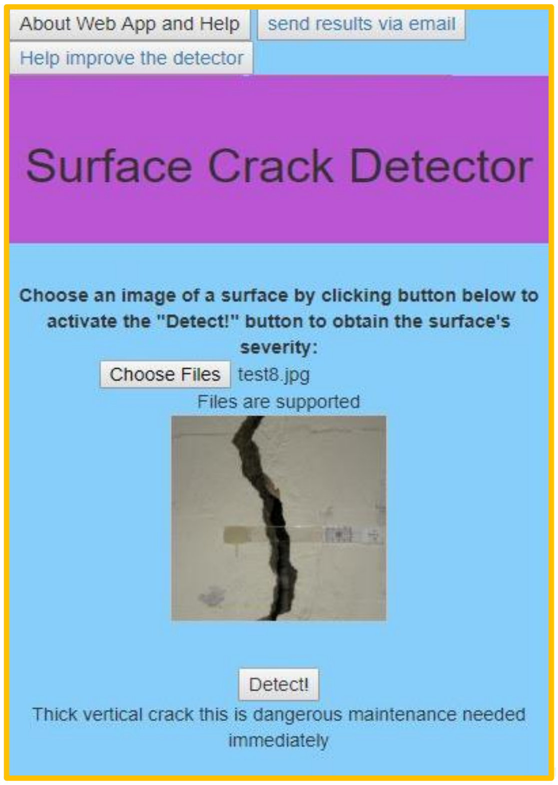
4.1 Discussion

The best test accuracy obtained using the "ModelCheckPoint" function for CNN1 and CNN2 were: 0.9727(97.3%) in epoch 14 and 0.9444(94.4%) in epoch 7. As it can be seen in table 4 from epoch 7 to 10 the test accuracy stops improving but the training accuracy continuously increased for the same epoch range having a standard deviation of 0.0109(1.09%).The possible causes of the lack of increase for the test accuracy from epoch 7 to 10 in table 4 could be due to slight overfitting. The test loss at epoch 14 for CNN1 was exceptional as it signified how close the CNN1's classification were from the actual labels of the images in the test set which meant that at epoch 14 CNN1 classified majority of test images correctly with minimal mistakes.

Using the ModelCheckPoint function was advantageous particularly for CNN1 as after the 14th epoch the test accuracy decreased by 0.14% and test loss was also increased by 0.33%, this meant if the ModelCheckPoint function was not used, the final epoch would have been saved instead, losing the opportunity of having a better performing model. The average test accuracy of the H-CNN(combination of CNN1 and CNN2) was 95.85%(4.15% error rate), considering the lack of image samples for CNN at level 2.

5 Surface Crack Detector H-CNN Capabilities

The integration process of the H-CNN to the front end was achieved through the Flask web framework. Flask enabled output of the H-CNN to be made visible in the HTML, CSS and JavaScript front-end. In figure 7 shows the classification functionality and feedback mechanism:



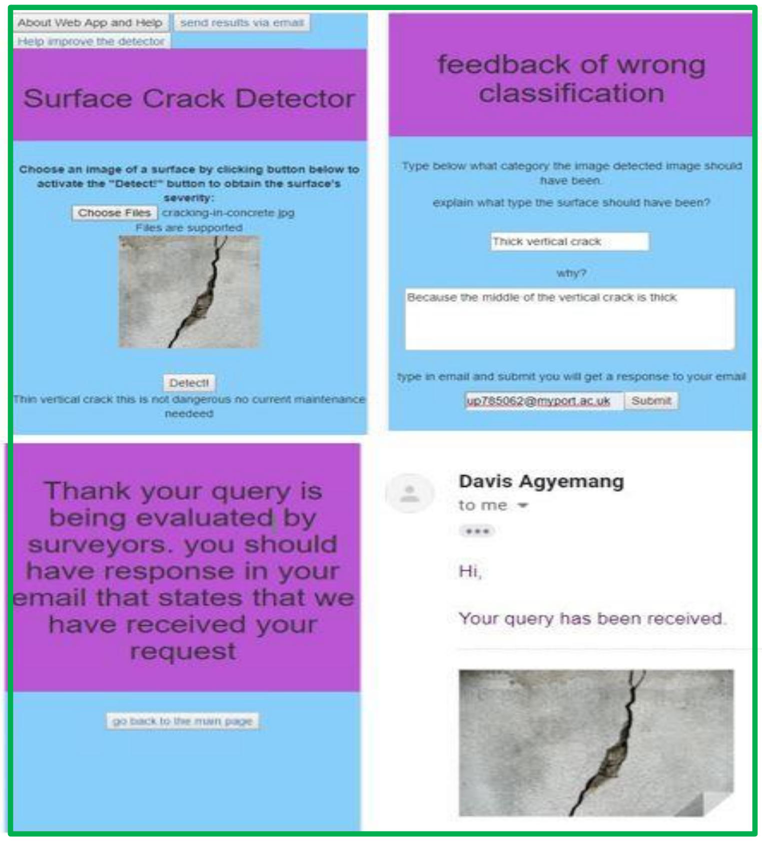
Surface Crack Detector

Choose an image of a surface by clicking button below to activate the "Detect!" button to obtain the surface's severity:

test8.jpg

Files are supported

Thick vertical crack this is dangerous maintenance needed immediately



Surface Crack Detector

Choose an image of a surface by clicking button below to activate the "Detect!" button to obtain the surface's severity:

cracking-in-concrete.jpg

Files are supported

Thin vertical crack this is not dangerous no current maintenance needeed

feedback of wrong classification

Type below what category the image detected image should have been.

explain what type the surface should have been?

Thick vertical crack

why?

Because the middle of the vertical crack is thick

type in email and submit you will get a response to your email

up785062@myport.ac.uk

Thank your query is being evaluated by surveyors. you should have response in your email that states that we have received your request

Davis Agyemang

Hi,

Your query has been received.

Figure 7. Demonstration of the Surface Crack Detector classification functionality(orange highlighted screen is the classification functionality and the green highlighted image is the feedback mechanism).

The classification functionality directly accesses the user's mobile phone camera to take the picture but if the user uses the desktop version it will access the file explorer. The feedback mechanism could have been automated in terms of allowing users to update the model without the author and the co-author checking if the H-CNN needed to be retrained based on the user's query. However, the dangerous of this approach is the users can potentially decrease the accuracy of the H-CNN if they decided to place the wrongly classified image in the wrong class for retraining, hence why it needs to be checked before to mitigate that risk. Using the feedback mechanism will enable the surface crack detector to continuously improve its accuracy by learning from its mistakes which is an area that many researchers such as the ones mentioned earlier in this paper have not ventured.

Overall, the application performed exceptionally well in terms of accuracy, but occasionally the application classified an input incorrectly especially if the uploaded image was not focused and the detecting thick or thin cracks was challenging as the way a user took the picture influenced that, hopefully in the future that will be fixed by using an algorithm that can determine depth of the image to determine the width of the crack.

This application has the potential to be beneficial for users such as surveyors, as it does not only provide increased efficiency in evaluating surface conditions, but its use cases can also be to help users who suffer from partial vision problems to still conduct inspections. This is the link for the of the application if one wishes to use the application: https://surfacecrackdetector.herokuapp.com.

5 Conclusion

The benefit of the application is that users such as surveyors will be able to record cracks at rapid rate as they would not need to manually record the conditions. The reason why that benefit is important is that users such as surveyors are required to write a report on a building which is a long process, but if the surface crack detection was used they would be able to quickly take pictures of the building, send the classification results to themselves and go back to the office to elaborate on the classification made by the application for their report. This would improve the work flow of surveyors.

This paper presents the creation of an H-CNN based surface crack detection application. The H-CNN was trained on 3 images datasets, containing a total of 920 (230 per class) images for CNN1 and a total of 390(130 per class) images for CNN2. Evaluation and analysis were achieved by testing the individual CNNs against its test set to monitor the test accuracy and loss. The CNNs were evaluated using their test accuracies to see if they would be appropriate for the H-CNN. The front-end end was built for the usability for users using Flask, HTML, Bootstrap CSS and JavaScript. It was concluded that the application has many use-cases, and with the “feedback mechanism,” this application has the potential to become more accurate over time for continuous improvement.

6 Future works

The authors hope to implement features such as to classify other defects such as damps and mould in buildings and to have the ability to classify a batch of images at once for users who wish to classify more images. A surveying Master lecturer asked the authors if in future when the application becomes more established that it can be merged with his augmented reality tool which can look at parts of a building such as a wall to inform the user on the last time it was maintained. Combining the surface crack detection application will help his augmented reality tool to see if a surface within a building needs maintenance.

References

Adlakha, D., Adlakha, D., & Tanwar, R. (2016). Analytical Comparison between Sobel and Prewitt Edge Detection Techniques. *International Journal of Scientific & Engineering*

Research, Volume 7, Issue 1, 1482.

Amer, h. M., & Abushaala, M. A. (2015). Edge detection methods. *2015 2nd World Symposium on Web Applications and Networking (WSWAN)* (p. 1). Sousse: IEEE.

Aslani, S., Dayan, M., Storelli, L., Filippi, M., Murino, V., Rocca, M., & Sonaa, D. (2018). Multi-branch Convolutional Neural Network for Multiple Sclerosis Lesion Segmentation. *Neuroimage*, 1-2.

Caltech. (2006 , December 12). *Home Objects dataset*. Retrieved from caltech: http://www.vision.caltech.edu/pmoreels/Datasets/Home_Objects_06/

Dai, J., & Nian Wu, Y. (2015). Generative Modeling of Convolutional Neural Networks. *The International Conference on Learning Representations (ICLR) 2015* (p. 1). San Diego: The International Conference on Learning Representations(ICLR).

Danso, M. (2018, October 18). Interview Validate Customer Requirements and Gain Advice. (D. Agyemang, Interviewer)

Dorafshan, S., Thomas, R. J., & Maguire, M. (2018). SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data in Brief*, 1664-1668.

Ellenberg, A., Kontsos, A., Bartoli, I., & Pradhan, A. (2014). Masonry Crack Detection Application of an Unmanned Aerial Vehicle. *International Conference on Computing in Civil and Building Engineering* (p. 1788). Florida: International Conference on Computing in Civil and Building Engineering.

Hoang, D. N. (2018). Detection of Surface Crack in Building Structures Using Image Processing Technique with an Improved Otsu Method for Image Thresholding. *Advances in Civil Engineering*, 1.

Hoang, D. N. (2018). Image Processing-Based Recognition of Wall Defects Using Machine Learning Approaches and Steerable Filters. *Computational Intelligence and Neuroscience*, 1.

Hu, D., Tian, T., Yang, H., Xu, S., & Wang, X. (2012). Wall Crack Detection Based on Image Processing. *Third International Conference on Intelligent Control and Information Processing* (p. 597). Dalian: IEEE.

Kim, B., & Cho, S. (2018). Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique. *Sensors*, 1.

Kunal, K., & Killemsetty, N. (2014). Study on control of cracks in a Structure through Visual Identification & Inspection. *IOSR Journal of Mechanical and Civil Engineering*, 64.

Lucke, J., & Sahani, M. (2007). Generalized Softmax Networks for Non-linear Component Extraction. *17th International Conference* (pp. 657-659). Porto: International Conference on Artificial Neural Networks.

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). High-resolution image classification with convolutional. *IEEE International Geoscience and Remote Sensing Symposium* (p. 2). Fort Worth: IEEE .

Neale, S. (2018, October 12). Capturing Requirements. (D. Agyemang, Interviewer)

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *arXiv:1511.08458*, 9.

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics Volume: 9 , Issue: 1*, 66.

Özgenel, F. Ç. (2018, January 15). *Concrete Crack Images for Classification*. Retrieved from data.mendeley: https://data.mendeley.com/datasets/5y9wdsg2zt/1

Radiuk, M. P. (2017). Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Information Technology and Management Science*, 20.

Seo, Y., & Kyung-shik, S. (2019). Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, 331.

Sharma, N., Vibhor, J., & Mishra, A. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. *Procedia Computer Science Volume 132*, 379.

Silva, d. L., & Lucena, d. S. (2018). Concrete Cracks Detection Based on Deep Learning Image Classification. *Proceedings* (p. 1). Brussels: Molecular Diversity Preservation International(MDPI).

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep Image: Scaling up Image Recognition. *arXiv*, 2.

Zhu, X., & Bain, M. (2017). B-CNN: Branch Convolutional Neural Network for Hierarchical Classification. *arXiv:1709.09890 (Preprint)*, 2.