UNDERGRADUATE RESEARCH OPPORTUNITIES PROGRAMME (UROP)

Machine Learning for Serviceability Limit State Structural Inspection of Concrete Cracks

Ng Zhao Png Isaac

Department of Civil Engineering, Faculty of Engineering National University of Singapore

SEM1/AY2020

Abstract

This is a preliminary research and development project to develop an automatic crack detection system for width and length based on visual images. Current machine learning tools and image processing techniques are used in its development together with a database of crack images. A software program has been written to enable the computation of concrete crack width at Serviceability Limit State (SLS) criteria of 0.3mm pertaining to most global codes. It also hopes to lay the foundation of a more comprehensive program to evaluate structural health based on visual imagery and provide insights to aid current Structural Health Monitoring Systems for Concrete Structures.

Preface

This research was started with the aim of tackling the challenges in the era of aging infrastructures in developed countries. Due to the large number of aging reinforced concrete structures, it is imperative to ensure their safety and serviceability. Preliminary research is done on the use of machine learning to support the current structural health monitoring systems for concrete structures, in particular crack control in the structural SLS.

Special thanks to the National University of Singapore (NUS), Faculty of Engineering, Department of Civil and Environmental Engineering and especially Dr Kong Kian Hau, my mentor for this project. Without their guidance this research project will not be possible. In addition, to my family for their support over all these years.

Table of Contents

1. Introduction and Research Motivation

In the era of aging civil infrastructures, especially in developed countries such as the US, there is a global issue of concrete deterioration. The history of concrete stretches as far back as ancient Egypt several thousand years ago. Even today, concrete is the most used construction material in the world with 33 billion tonnes consumed (ISO, 2016) and being used to build megastructures such as the planned Jeddah tower which will be the tallest building in the world. Hence, it is here to stay, and it is crucial to maintain its durability over its structural lifecycle.

One of the leading causes of concrete deterioration is environmental degradation (Gebregziabhier, 2008). Due to long-term exposure of carbon dioxide in industrial zones leading to carbonisation of the concrete material, and chlorine attack at sea areas, these would lead to the formation of micro cracks. This is an issue due to the reinforcement of concrete with steel. Cracks having a width greater than 0.2mm have the potential to cause deterioration through seepage of water and oxygen, causing oxidation and corrosion in the underlying steel structure. Reducing its cross-sectional area and hence capacity, potentially resulting in structural issues for the member and the structure.

In civil engineering codes worldwide such as ACI 318-19 (ACI, 2019) and EC 2 (BSI, 2004), there is a mandatory requirement for crack control under the structural SLS. Under the Eurocode system, cracks greater than 0.3mm over a 50-year design lifetime would be deemed an SLS breach. To monitor such concrete cracks is costly and hence an automatic crack detection system which serves this role would be beneficial.

2. Objective

To develop a crack width detection system where critical crack regions are displayed, highlighted with the approximate crack width, and tested to determine if it is larger than 0.3mm.

3. Conceptual framework

Currently structure inspection is a manual task whereby professional engineers (PE) are required to visually inspect the condition, loading as well as any other alteration or addition works affecting the structure of the building (BCA, 2012). This involves challenges whereby it is not possible or costly to access certain areas. Recently, under Project SUAVE drones have been utilised by the Singapore Land Authority to conduct such visual inspections (SLA, 2019). However, PE`s are still required to manually sift through images gathered by the drone.

In modern Structural Health Monitoring Systems (SHM), computer vision and deep learning techniques are being implemented to great effect to automate these tasks (X. W. Ye, 2014). The methods used are mainly image processing and/or deep learning models. A review on the current state of image processing techniques on crack surfaces reveals four main approaches, integrated algorithm, morphological approach, percolation-based method, and practical technique (Arun Mohan, 2016). Whereby the processing difficulty of crack detection is solely dependent on the size of the image. This approach has a lower accuracy as compared to deep learning models but is useful to label large amounts of data for deep learning training. For deep learning training, a revised UNET-based deep learning model has been shown to have better results (98.7% precision, 91.9% recall) as compared to other deep learning models such as CrackNet-V (92.6% precision, 86.0% recall) (Lingxin Zhang, 2020).

Thus, based on their success, we would like to assess the ability of a standard UNET-based deep learning model together with image processing techniques to detect concrete cracks and their width.

4. Methodology

<u>Overview of the Crack Detection algorithm:</u>

There are three modules in the UNET Crack detection algorithm. First, the original images taken from the dataset are pre-processed, and a training dataset is formed with labels from the original image. This dataset is then sent to the UNET at the second module for training, together with a validation dataset to finetune the hyperparameters. Finally, the test dataset will be sent into the optimized UNET model to evaluate its performance.

4.1 Image pre-processing:

The dataset used in this model is the Concrete Crack images for Classification dataset (Çağlar Fırat Özgenel, 2018). With 20,000 crack images of size (227 x 227 pixels) generated from 458 high-resolution images (4032 x 3024 pixels). Due to variances in illumination and surface finishes, images need to be screened before postprocessing to have a more homogeneous dataset whereby labels generated from image processing techniques used for training would not have significant errors due to its variance. A representative sample of 420 images will be used for training, 80 images used for hyperparameter tuning and a further 80 images are used for evaluation.

The original images are first labelled for use in training the UNET model. This is done through the following procedures:

Gauss Blurring

As images from the dataset have low background lighting with high amounts of noise. A Gaussian blur is applied which generates a softened the image and enables clearer identification of prominent features. A Gaussian blur involves convolving the image with a 2D Gaussian function shown below.

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

Fig 1: 2D Gaussian function used for image pre-processing

Where *x* is the distance from the origin in the horizontal axis, *y* is the distance from the origin in the vertical axis, and *σ* is the standard deviation of the Gaussian distribution.

In this case a (3,3) kernel is used as the training images have similar dimensions in x and y axis and we want to capture more minor details where a bigger kernel might miss. The standard deviation is placed at *σ* =1.

Contrast Limited Adaptive Histogram Equalisation (CLAHE)

As some details are lost during the Gaussian blurring, contrast needs to be improved between the crack portions of the image and the surrounding background for a more accurate labelling.

In CLAHE, the contrast amplification in the vicinity of a given pixel value is limiting as opposed to normal Histogram amplification, by clipping the histogram at a predefined value before computing the CDF of the image. This limits the slope of the CDF and therefore of the transformation function. The value at which the histogram is clipped, the so-called clip limit, relies on the normalization of the histogram and thereby on the size of the neighbourhood region used.

Non-local Means Denoising:

From the resulting Gauss and Histogram adapted image, there remains background noise that has resulted from the Equalisation. As the colours of the image are now equalised. Non local means denoising would be useful as opposed to local means denoising to reduce the background noise by replacing the colours of a pixel with an average of the colours of similar pixels around the neighbourhood, as well as to keep the features we have enhanced through the histogram equalisation.

The denoising of a color image u = (u1, u2, u3) at a certain pixel p is expressed through:

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q)\, w(p,q), \qquad C(p) = \sum_{q \in B(p,r)} w(p,q),$$ (Eqn. 2)

Fig 2: Non-local means denoising used for image pre-processing

where i = 1, 2, 3 and B (p, r) indicate a region centred at point p with size (2r + 1) × (2r + 1) pixels. The weight w(p, q) depends on the squared Euclidean distance $d^2 = d^2$ (B(p, f), B(q, f)) of the (2f + 1) × (2f + 1) colour patches centred respectively at p and q.

$$d^2(B(p,f), B(q,f)) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{j \in B(0,f)} (u_i(p+j) - u_i(q+j))^2.$$ (Eqn. 3)

Fig 3: Squared Euclidean distance at p and q used for the weightage function

In this case, the recommended sizes are followed at 21x21pixels for the search window,7 for the block size and 10 for h.

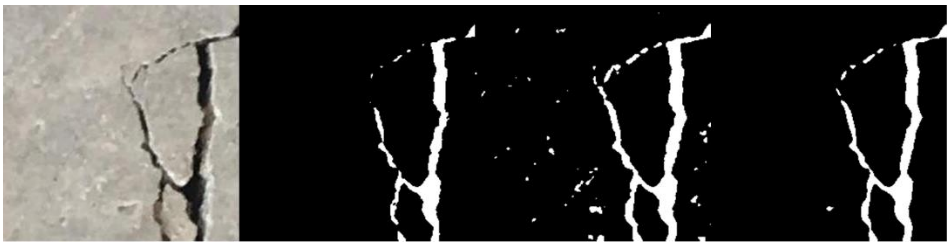

Original
Gauss Blurring
Gauss blur + Histogram equalisation
Gauss blur + Histogram equalisation + Non local means filtering

Fig 4: Steps taken during image pre-processing to support labelling

4.2 Architecture of UNET model:

The classic UNET architecture as proposed by Ronneberger et. al, (Olaf Ronneberger, 2015) is used with the addition of normalisation layers applied to the output of each unit. To increase the speed of the training process. Inputs are of size (255,255,1).

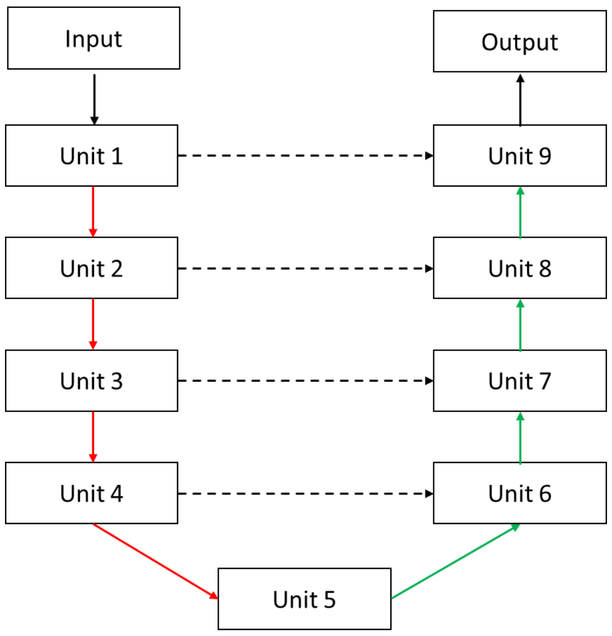


Fig 5: Standard UNET model used (Refer to Appendix A for unit structure)

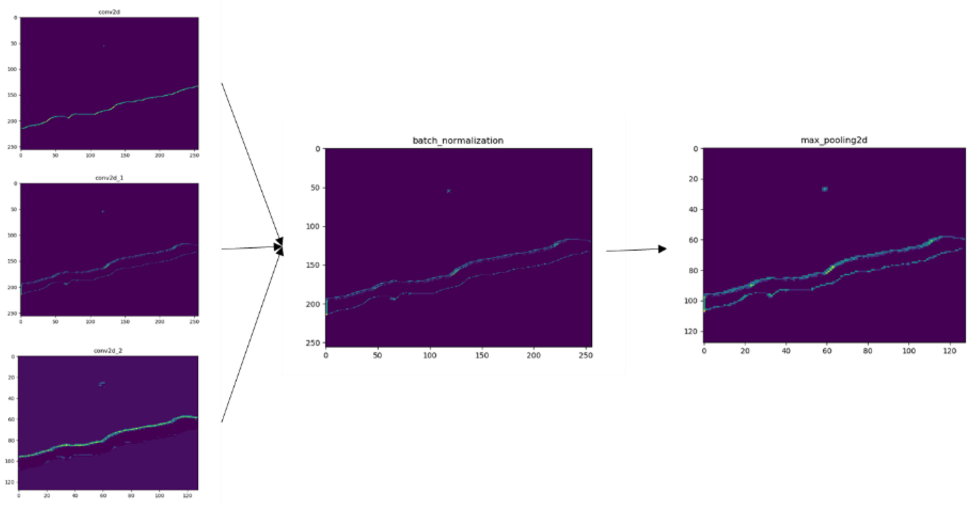


Fig 6: Basic representation how the layers interact from convolution to pooling

Convolution layer

The convolution layer contains learnable filters which transforms the image into an output volume. The filters cover a small area of the input image but extend throughout its colour channels. They slide (convolve) across the input and compute dot products between the entries of the filter and the input at any position. The number of pixels for which they slide is the stride.

Intuitively the network will learn filters that activate when a visual feature is detected on the image as shown in the first process of Figure 7 whereby different activations occur at each unit. In this case, to keep the consistency of input and output sizes during convolution the same padding is applied. In the UNET architectures, the convolution kernel and the stride speed are fixed at 3x3 and 1, respectively.

Batch Normalisation layer

In the research done by Sergey et. al, it is shown that the training of deep learning networks is slowed by the changes in the distribution of internal nodes in each layer (Sergey Ioffe, 2015). Batch normalisation for each unit of the UNET resolves this difficulty by standardising the activations of the previous layers whereby assumptions made about the spread of the inputs during weight updating remains unchanged, increasing the efficiency and speed of the training process.

Pooling layer

The Pooling layer is used to control overfitting the model to the training. This is done by decreasing the size of the input tensor, in this case by half, extracting only the important features from each region which reduces the number of parameters. There are four pooling layers used in this model, each providing a greater receptive field, enabling more complex features to be extracted at each stage. The size of the Pooling layer used in this case is a 2x2 filter, with a stride of 1.

Up sampling layer

Opposite of the pooling layer, the up-sampling layer increases the size of the input tensor, restoring features which are lost during down sampling. This is done through transposed convolution of the input to produce a one-to-many relationship. By learning the optimal parameters through backpropagation, the input is then able to be converted by the filter to the desired output size with its restored features. Size of the filter used is same as the pooling layer.

Activation function

Activation functions are used to determine the activation of the neuron in the convolution layer. As the network is trained, the weighted sums and biases in the neuron are optimised based on backpropagation. Activation functions enables this to happen by having gradients along with the error enabling it to update the weights and bias in the neuron instead of a linear output that would be expected.

Rectified Linear Activation Function (ReLu) is used in this model in comparison to Sigmoid and tanh. This is because of the nature these functions to snap to extremes. In a study done by Hara et al, (Kazuyuki Hara, 2015). ReLu is shown to improve the performance of a recognition system as compared to sigmoid functions due to its derivative term leading it to avoid plateaus when using a gradient descent algorithm and allows it to converge at a faster rate, speeding up the training process. An improvement in the performance compared to other activation functions is also seen in the facial recognition system by Wang et al, where ReLu based functions provide better accuracy in CNN models (Yingying Wang, 2020).

E

$$\text{ReLU}(x)=\max(0,x)$$

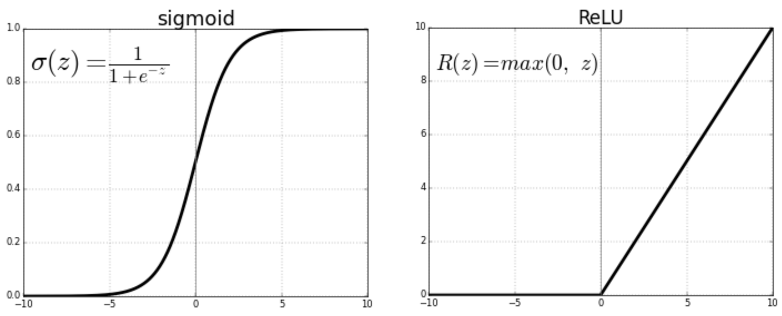


Fig 6: Comparison between sigmoid and ReLu functions.

Loss Function

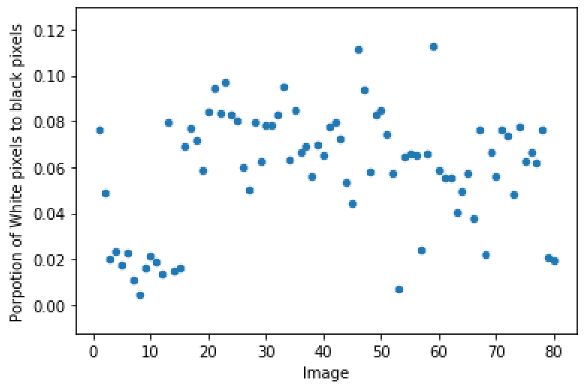


Fig 7: Proportion of white to black pixels in the test dataset

A loss function is used as the target for the model, penalising deviations between prediction and ground truth. As the crack detection problem is a Binary classification problem (crack and background areas). From the proportion shown in Fig 7, it is also a class imbalance problem. Even though cross-entropy loss (CE) does not perform as well in class imbalanced problems as compared to other functions such as dice coefficient, the difference is marginal, 0.968 vs 0.97 in dice loss using dice coefficient (Jadon, 2020).

In addition, CE loss will provide a more even gradient as opposed to other binary loss functions like dice coefficient hence the time of training is more predictable. The class imbalance problem in the dataset is resolved through assigning loss multipliers to each class so the network is disincentivized to ignore outliers such as small cracks in the image.

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1) \quad \text{(Eqn. 4)}$$

Fig 8: Cross entropy Loss Function (Log loss)

The Binary (two classes) CE function is shown in Fig 8 with classes C1 and C2. t1 [0,1] and s1 are the ground truth and the predicted probability of C1, and t2=1−t1 and s2=1−s1 are the ground truth and the predicted probability for C2.

Optimiser

Adam is used for the optimiser as it is comparatively more efficient and adapts learning scale rates for different layers as opposed to manual assignment by other optimisers like Stochastic Gradient Descent. (Diederik P. Kingma, 2015) It uses estimations of the mean and uncentered variance of the loss function to adapt the weight of each weight in the neural network.

4.3 UNET Model Parameters

Table 1: Parameters Used during training phase

Parameters	Value
Learning rate	0.0001
First moment decay rate	0.9
Second moment decay rate	0.999
Epsilon	0.001
Batch size	2

The Adam hyperparameters in this study are set up referring to Kingma and Bal, with the learning rates of set as 0.0001 (Diederik P. Kingma, 2015). The default exponential decay rates for the mean and uncentered variance estimates are used at 0.9 and 0.99, respectively. The harmonic parameter in the Batch normalisation is set at $1e^{-3}$. With a batch size of 2.

4.4 UNET Training procedure

The UNET Model was trained with the labels and images for training and testing. The classifier was trained until no improvement to the validation loss is made after 2 consecutive epochs. This was determined to be at 106 epochs, with the training data randomly split between 2 batches, achieving a validation accuracy of 97.9% and a validation loss of 12.7%.

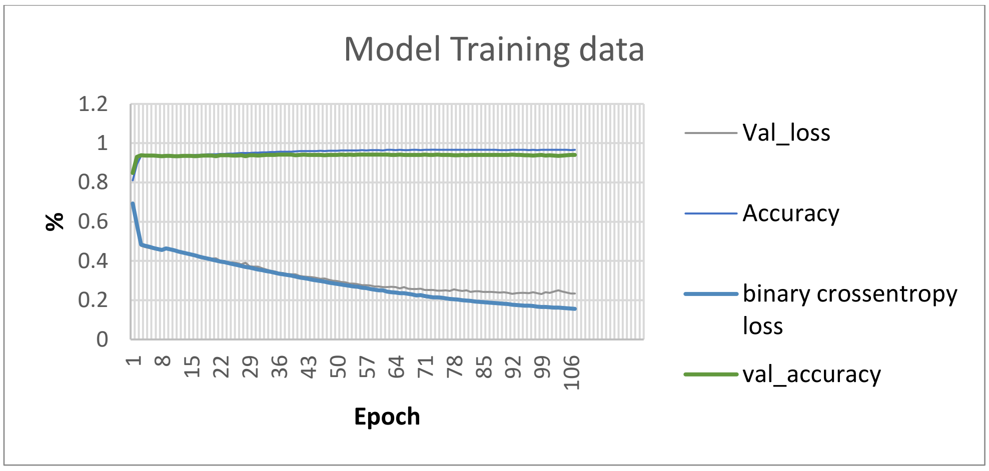


Fig 9: Accuracy and cost functions of UNET during training

From the model results during training, accuracy has hit a plateau early during the training phase and has seen incremental increases as the model trains. However, there is still room for improvements to be made in the loss function.

5. UNET Test results

<u>Image composition</u>

Firstly, proportion of crack (black) to white pixels is analysed to ensure the composition of the image has not been significantly altered and crack details have been captured. The results can be seen in Figure 10. The model has a Mean Squared Error (MSE) of 0.137, representing a good representation of the crack features in the original image.

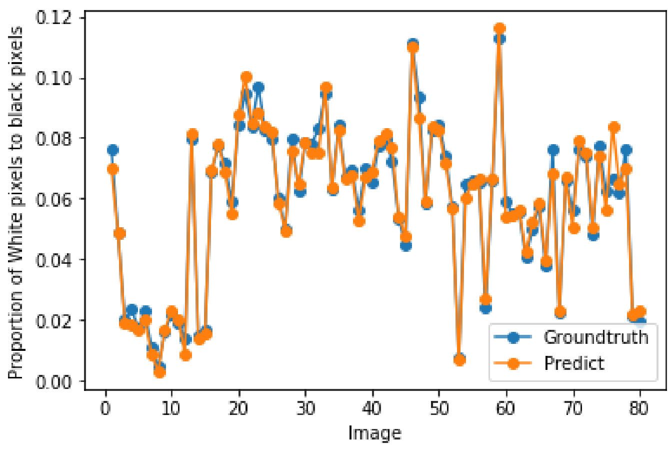


Fig 10: Proportion of White to Black (Crack) pixels predicted vs Groundtruth (actual)

An analysis of the Image segmentation gives insights into how crack features are formed during prediction. In Figure 11 below, an inverse relationship is observed between the proportion of the image and the percentage of image which is segmented. This helps to confirm that the model has not been overfitting and ignoring crack pixels.

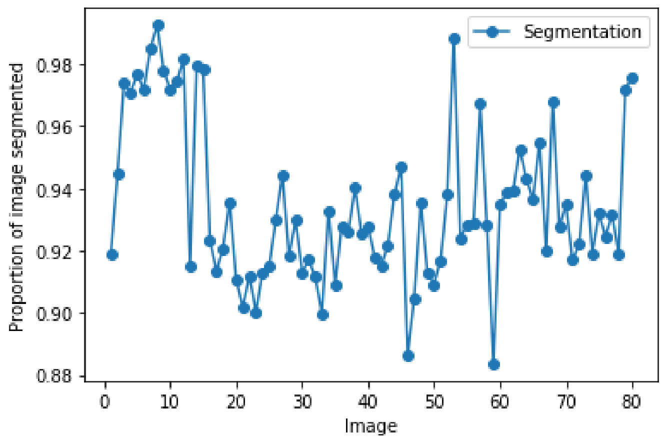


Fig 11: Proportion of image segmented through the test data

Convolutional Layer activation

Analysis of the convolutional layer activations during testing in Appendix D, shows that even at the deepest depth, there are no unused layers. This is a sign that it is making use of the entire depth of the UNET for prediction. To learn more about the nature of the crack class, we might need to go deeper into the network, however for our purposes, it is enough for crack prediction.

Performance

Performance of the UNET is then assessed on the test batch through Precision, Recall and F1 metrics, to ensure that performance is holistically captured. The mean results from the dataset are 92.0%, 70.7% and 43.7% for precision, F1 and recall, respectively. This shows that while the current UNET model has a relatively high rate of precision, the model is selective in returning crack features from the image and might have issues dealing with smaller cracks.

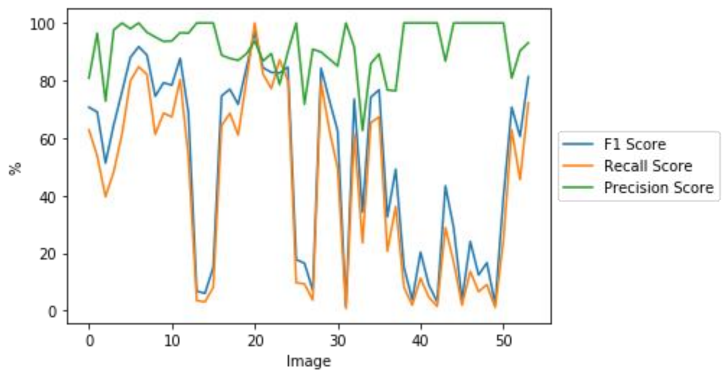


Fig 12: Scores by the UNET Model on the dataset

6. Critical Features Extraction

Overview

To obtain critical features of the predicted crack, segmentation must occur to isolate critical areas and general details of the predicted crack.

Some key assumptions made are that images are assumed to be taken from the same height and angle, without accounting for perception error. A crack width of 0.1mm is assumed to be 1-pixel length, and any image surpassing a pixel width of 3 pixels is deemed to have failed the SLS criteria.

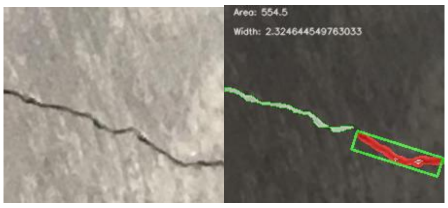


Figure 13: Image showing a 0.2mm wide crack

Segmentation Procedure

Based on research done by Thouraya et. al, on ultrasonic crack detection (Thouraya Merazi-Meksen, 2014). A watershed technique is proposed to extract regions of interest from connected objects. Then skeletonization is used to determine the effective length of the region.

Erosion and Dilation

The foreground object is first eroded, by having pixels near the crack boundary discarded by the size of the kernel used, removing smaller crack areas. This is followed by dilating the remaining pixels where the regions of interest lie. Figure 14 below illustrates this process.

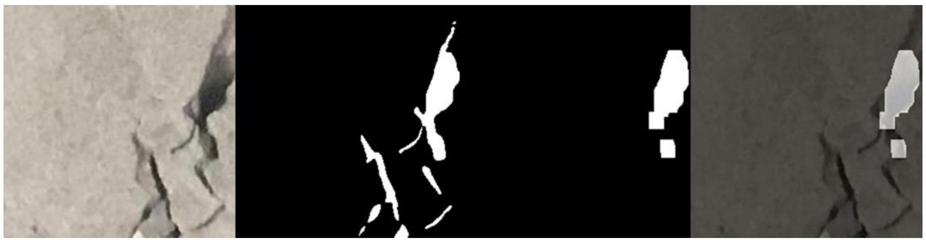


Fig 14: (Left to Right) Process of erosion and dilation until the critical regions are found

Distance transform

The background of the image is then identified and separated from the regions of interest. The remaining areas is marked and separated using the watershed algorithm to establish boundaries, segregating these regions.

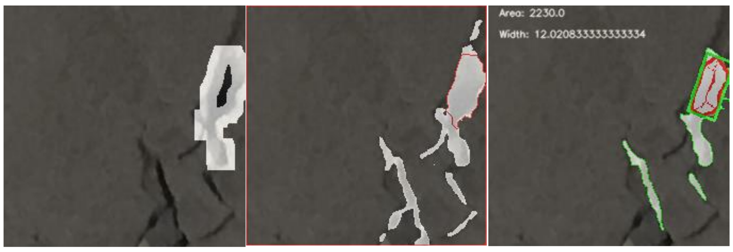


Fig 15: (Left to Right) separation of background, Watershed boundary regions, Skeletonization

Skeletonization

Skeletonization of the crack area is then carried out through repeated erosions and dilations until a single pixel length is obtained for the segment. The length of the skeleton will then be used for an approximation of crack length and hence the width is found. An iterative search throughout the image determines the location of the largest crack width from the segments are identified and the critical area is then highlighted by a box.

7. Overall Evaluation

Due to the empirical nature of crack analysis, the test results are analysed through different segments and the performance is also evaluated based on the ability or failure of the current model to extract the crack features from challenging data.

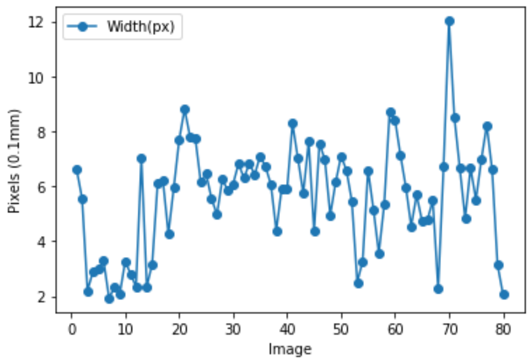


Fig 16: Crack width obtained from the Combined model on the testing dataset

Table 2: Evaluation of Errors on Combined Model

	Error	Images
1	Incorrect Critical Width location	6/80
2	Non cracks detected (False Positives)	2/80
3	Cracks >= 0.3mm missed (3px) (False Negatives)	2/80
4	Cracks <0.2mm missed (False Negatives)	13/80
5	Blank results	0/80

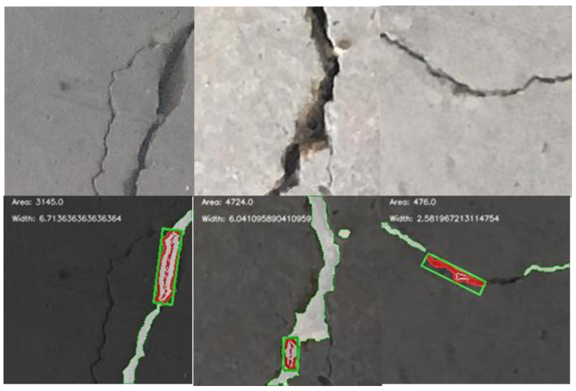


Fig 17: Test images number 60, 33, 12 respectively

Errors evaluated during testing are listed in Table 2. With images having the greatest number of errors in Fig 17, classified according to the errors in table 2. Images 60 and 12 have errors 3 and 4 with Image 33 having errors 1 and 4. Aspects of perception distortion and similar background colour to the cracked area are observed in Images 60 and 12 highlighting a weakness in the trained UNET in this region. Colour changes around the crack of image 33 are also shown to disrupt the model's ability to predict crack edges resulting in a miss in the critical width location.

Background noise and lighting evaluation

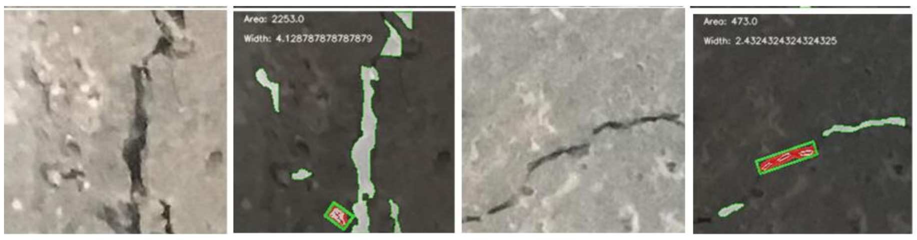


Figure 18: Performance of the model on regions with background noise.

There is also a higher incidence rate of false negatives with the increase in background noise muting prediction of crack features. Changes in the luminosity also affects the performance of the model with a better prediction in better brightness. This can be attributed to the normalisation and thresholding of the image during image processing. However, the model is shown to have detected most regions of the cracked area, making it able to detect cracks in challenging data.

<u>Critical Width detection evaluation</u>

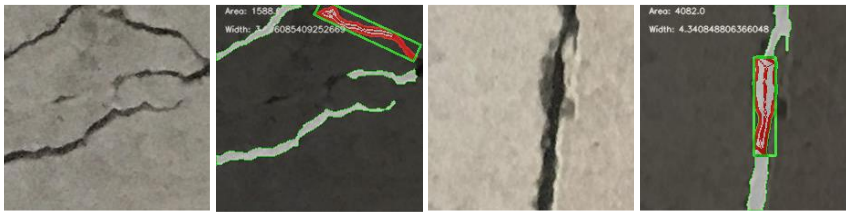


Figure 19: Performance of the model on regions with varying critical widths

The length of segments identified from the crack are however prone to errors, resulting in underestimations of width. This is more noticeable in areas where the width is uniform, and the watershed algorithm has not been able to separate the critical areas from the regions of lower pixel density. However, these errors are offset in the ability to find and highlight these areas of concern.

8. Reflections

A standard UNET model used together with image processing techniques has shown to detect cracks greater than 0.3mm with a relatively high degree of reliability and to highlighting the critical regions. There are however some flaws in the UNET model that need to be improved for real life application.

Improvements to existing model

Accuracy

The model has difficulty at detecting cracks at width 0.2mm and below, mainly due to the background noise reduction filters. To increase the accuracy of the model, further optimisation of the settings needs to be done as well as testing using more real-life environments and would need to be acquired to prove its effectiveness. Loss functions can also be experimented with to increase the accuracy of the model, such as Hinge loss. (Katarzyna Janocha, 2017)

Perspective error

Currently, the images are also assumed to be taken perpendicular to the surface. However, in real life applications this is not true. Thus, a correction needs to be implemented to take this into account, possible solutions include projecting the current 2D image into a 3D plane through affine transformation as used by Adhikari in the bridge detection model during image pre-processing (R.S. Adhikari, 2013).

Increasing current abilities

Crack length and width

Based on the skeleton of a crack, the length of the crack could be extrapolated with the longest branch of the skeleton taken for the length of the crack. In addition, to separate branches in the skeleton, possibly Harris Corner could be used to place points at edges where black pixels can be added to separate the branches in further study.

Future works

In order to achieve the greater goal of estimating the future lifespan of the concrete structure before localised failure occurs, it is necessary for the model to have a wider scope in classifying the types of crack being inspected and provide a more comprehensive overview of the structure. Several additions to the crack detection algorithm can be implemented such as a classifier to determine the kind of structural crack being observed as well as models to predict the effect on the underlying steel and be part of a more complete SHM framework.

References

ACI. (2019). *Building Code Requirements for Structural Concrete (ACI 318-19).* American Concrete Institute.

Arun Mohan, S. P. (2016). Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, 787-798.

BCA. (2012). *PERIODIC STRUCTURAL INSPECTION OF EXISTING BUILDINGS.* Singapore: BCA.

BSI. (2004). *BS EN 1992, Eurocode 2: Design of concrete structures.* Brussels: British Standards Institution.

Çağlar Fırat Özgenel, G. S. (2018). Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings. Berlin: ISARC .

Diederik P. Kingma, J. B. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations, .* San Diego.

Gebregziabhier, T. T. (2008). *Durability problems of 20th century reinforced concrete heritage structures and their restorations .* Barcelona: SAHC.

ISO. (2016). *STRATEGIC BUSINESS PLAN.* International Standards Organisation.

Jadon, S. (2020). A survey of loss functions for semantic segmentation. *IEEE*.

Katarzyna Janocha, W. M. (2017). On Loss Functions for Deep Neural Networks in Classification. *Theoretical Foundations of Machine Learning 2017.* TFML .

Kazuyuki Hara, D. S. (2015). Analysis of Function of Rectified Linear Unit used in Deep learning. *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). Killarney.

Lingxin Zhang, J. S. (2020). A research on an improved Unet-based concrete crack detection algorithm. *Structural Health Monitoring*, 1–16.

Olaf Ronneberger, P. F. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.* MICCAI.

R.S. Adhikari, O. M. (2013). Image-based retrieval of concrete crack properties for bridge inspection. *Automation in Construction*, 180-194.

Sergey Ioffe, C. S. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing.* CA: Google.

Shi, Y. (2016). Automatic Road Crack Detection Using Random Structured Forests. *IEEE transactions on intelligent transportation systems*, 3434-3445.

SLA. (2019). A SMARTER INSPECTION FOR STATE PROPERTIES. *SLA*.

Thouraya Merazi-Meksen, M. B. (2014). Mathematical morphology for TOFD image analysis and automatic crack detectionThourayaMerazi-MeksenMalikaBoudraaBachirBoudraa. *Ultrasonics*, 1642-1648.

X. W. Ye, Y. H. (2014). Structural Health Monitoring of Civil Infrastructure Using Optical Fiber Sensing Technology: A Comprehensive Review. *The Scientific World Journal*, 11.

Yingying Wang, Y. L. (2020). The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition. *Applied Sciences*, 1-20.

Appendix A: Test image data

Unit	Layer	Filter Size	Kernels	Stride	Output size
1	Conv 1_0 Conv 1_1 Batch_Normalisation Max_Pooling	3,3	64	1,1	256,256,64 256,256,64 256,256,64 128,128,64
2	Conv 2_0 Conv 2_1 Batch_Normalisation Max_Pooling	3,3	128	1,1	128,128,128 128,128,128 128,128,128 64,64,128
3	Conv 3_0 Conv 3_1 Batch_Normalisation Max_Pooling	3,3	256	1,1	64,64,256 64,64,256 64,64,256 32,32,256
4	Conv 4_0 Conv 4_1 Batch_Normalisation DropOut (0.5) Max_Pooling	3,3	512	1,1	32,32,512 32,32,512 32,32,512 32,32,512 16,16,512
5	Conv 5_0 Conv 5_1 Batch_Normalisation DropOut (0.5)	3,3	1024	1,1	16,16,1024 16,16,1024 16,16,1024 16,16,1024
6	Up Sampling Merge with Unit 4 Conv 6_0 Conv 6_1 Batch_Normalisation	3,3	512	1,1	32,32,1024 32,32,1024 32,32,512 32,32,512 32,32,512
7	Up Sampling Merge with Unit 3 Conv 7_0 Conv 7_1 Batch_Normalisation	3,3	256	1,1	64,64,512 64,64,512 64,64,256 64,64,256 64,64,256
8	Up Sampling Merge with Unit 2 Conv 8_0 Conv 8_1 Batch_Normalisation	3,3	128	1,1	128,128,256 128,128,256 128,128,128 128,128,128 128,128,128
9	Up Sampling Merge with Unit 1 Conv 9_0 Conv 9_1 Batch_Normalisation	3,3	64	1,1	256,256,128 256,256,128 256,256,64 256,256,64 256,256,64
OUTPUT	Conv 10_0	3,3	64	1,1	256,256,1

Representation of each unit in the UNET architecture

Appendix B: Test image Prediction data

Image	Predicted Proportion	GroundTruth Proportion	Segmented	Segmented (px)	Width(px)
1	0.069884	0.076319	0.919172	47364	6.601594
2	0.04864	0.048777	0.944653	48677	5.566667
3	0.018858	0.020095	0.974053	50192	2.207944
4	0.018545	0.023249	0.971007	50035	2.875
5	0.016535	0.01734	0.976635	50325	2.983146
6	0.02028	0.022722	0.971822	50077	3.284672
7	0.008514	0.011015	0.985212	50767	1.932203
8	0.002902	0.004476	0.992645	51150	2.330882
9	0.016779	0.016062	0.97809	50400	2.087838
10	0.023188	0.021717	0.971977	50085	3.267516
11	0.020027	0.019064	0.974403	50210	2.801136
12	0.008832	0.013689	0.981486	50575	2.344595
13	0.081224	0.079422	0.915155	47157	7.022727
14	0.013741	0.01485	0.979565	50476	2.359043
15	0.015718	0.016582	0.978265	50409	3.168539
16	0.069201	0.06905	0.923461	47585	6.088889
17	0.07798	0.077098	0.913563	47075	6.193182
18	0.068819	0.071568	0.920782	47447	4.269231
19	0.055185	0.058843	0.935667	48214	5.945255
20	0.087471	0.084064	0.91071	46928	7.6875
21	0.100151	0.09462	0.901842	46471	8.811321
22	0.084935	0.083562	0.911952	46992	7.808824
23	0.088257	0.096947	0.90025	46389	7.734657
24	0.083449	0.082542	0.913117	47052	6.145631
25	0.081814	0.07992	0.914961	47147	6.45
26	0.058639	0.060093	0.929904	47917	5.548611
27	0.049507	0.050088	0.944458	48667	4.97271
28	0.075937	0.079599	0.918182	47313	6.247619
29	0.064798	0.062465	0.929709	47907	5.854251
30	0.078552	0.078338	0.913059	47049	6.085106
31	0.075162	0.078108	0.917153	47260	6.828358
32	0.07487	0.082971	0.911545	46971	6.296012
33	0.096582	0.09484	0.899629	46357	6.826087
34	0.063594	0.06305	0.932853	48069	6.424242
35	0.08255	0.08453	0.909216	46851	7.098901
36	0.066297	0.066806	0.928002	47819	6.705263
37	0.067748	0.069382	0.926003	47716	6.068182
38	0.052979	0.056232	0.940189	48447	4.386179
39	0.067093	0.069766	0.92577	47704	5.924658

40	0.069015	0.065262	0.927905	47814	5.889286
41	0.079107	0.077523	0.918007	47304	8.286408
42	0.081224	0.079422	0.915155	47157	7.022727
43	0.07684	0.072286	0.921772	47498	5.738636
44	0.054091	0.053362	0.938171	48343	7.626667
45	0.047314	0.044565	0.947175	48807	4.373494
46	0.110228	0.111251	0.886511	45681	7.559603
47	0.086706	0.093561	0.904733	46620	6.965812
48	0.0593	0.05821	0.935628	48212	4.949275
49	0.083449	0.082542	0.913117	47052	6.145631
50	0.08255	0.08453	0.909216	46851	7.098901
51	0.071361	0.074009	0.916979	47251	6.58
52	0.056755	0.057356	0.93819	48344	5.463415
53	0.006796	0.007223	0.988608	50942	2.51087
54	0.06046	0.064795	0.92412	47619	3.268204
55	0.064811	0.065626	0.928429	47841	6.596154
56	0.06629	0.065141	0.928894	47865	5.166667
57	0.02718	0.023888	0.967552	49857	3.565625
58	0.066405	0.065886	0.928526	47846	5.328571
59	0.116216	0.112741	0.883716	45537	8.71831
60	0.053857	0.058928	0.934716	48165	8.420765
61	0.054627	0.055092	0.938559	48363	7.142857
62	0.056439	0.055517	0.939277	48400	5.982143
63	0.042165	0.040452	0.952823	49098	4.545151
64	0.051967	0.049853	0.943042	48594	5.684211
65	0.058567	0.05722	0.936482	48256	4.75
66	0.039468	0.037767	0.954802	49200	4.769841
67	0.067996	0.075966	0.920006	47407	5.487633
68	0.022728	0.022179	0.968134	49887	2.261111
69	0.06694	0.066181	0.927536	47795	6.722222
70	0.050669	0.055976	0.93491	48175	12.02083
71	0.07896	0.076019	0.917037	47254	8.512346
72	0.075067	0.073657	0.922044	47512	6.675676
73	0.050545	0.048241	0.944342	48661	4.833333
74	0.074157	0.077275	0.919152	47363	6.669872
75	0.056343	0.062465	0.931961	48023	5.509615
76	0.083922	0.066649	0.924703	47649	6.963158
77	0.064763	0.062086	0.931708	48010	8.227273
78	0.069884	0.076319	0.919172	47364	6.601594
79	0.021633	0.021112	0.971996	50086	3.138655
80	0.023104	0.019302	0.975587	50271	2.086066

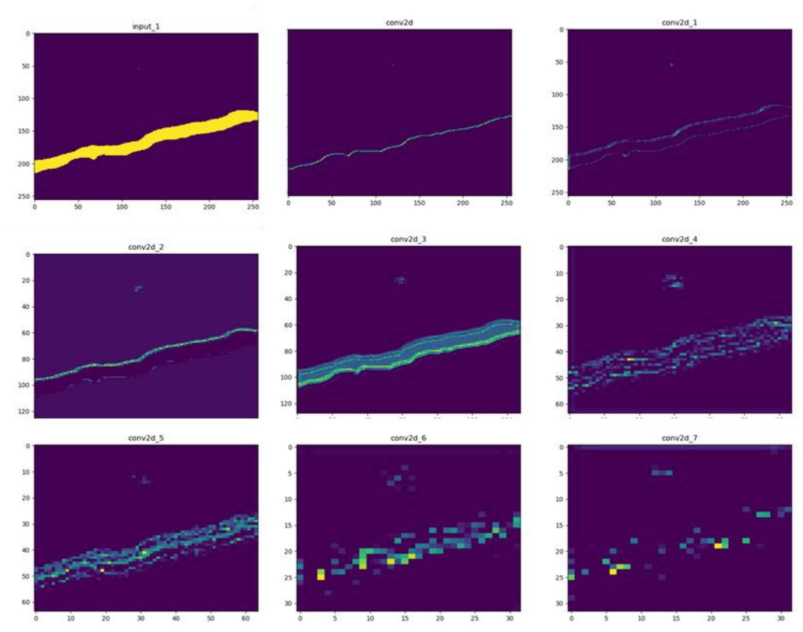
Image	F1 Score	Recall Score	Precision Score
1	0.707198494	0.628449317	0.808510638
3	0.690537084	0.537848606	0.964285714
4	0.513311688	0.396240602	0.728571429
5	0.647214854	0.484126984	0.976
6	0.762135922	0.615686275	1
7	0.880361174	0.799180328	0.979899497
8	0.917547569	0.84765625	1
9	0.888211708	0.820976559	0.96744186
10	0.745237671	0.612585366	0.951219512
11	0.792190026	0.686777644	0.935828877
12	0.783847981	0.673469388	0.9375
14	0.877192982	0.803212851	0.966183575
15	0.688956434	0.535855004	0.964539007
17	0.067924528	0.03515625	1
19	0.060606061	0.03125	1
26	0.151624549	0.08203125	1
27	0.746660189	0.643994413	0.888268156
34	0.769880823	0.686128098	0.876923077
36	0.717266088	0.610122606	0.870056497
37	0.844090475	0.800241619	0.893023256
38	0.967479675	1	0.937007874
39	0.844595658	0.822562727	0.86784141
40	0.828425209	0.772021791	0.893719807
41	0.826633375	0.871686853	0.78600823
43	0.846818182	0.799570815	0.9
44	0.177935943	0.09765625	1
45	0.165283158	0.093391703	0.717948718
46	0.071301248	0.037105751	0.909090909
47	0.842978327	0.793499164	0.899038462
48	0.731971154	0.629132231	0.875
51	0.625382428	0.494549284	0.850340136
52	0.015503876	0.0078125	1
53	0.735642016	0.61454868	0.916167665
54	0.342996607	0.23618626	0.626168224
55	0.741979738	0.653195838	0.858695652
56	0.767799642	0.673783359	0.892307692
57	0.325595736	0.20665362	0.767123288
59	0.491608425	0.362349313	0.764227642
60	0.151624549	0.08203125	1
61	0.038314176	0.01953125	1

62	0.203508772	0.11328125	1
63	0.089552239	0.046875	1
65	0.030769231	0.015625	1
66	0.43373494	0.289156627	0.86746988
68	0.287625418	0.16796875	1
69	0.038314176	0.01953125	1
70	0.240549828	0.13671875	1
73	0.124542125	0.06640625	1
75	0.166666667	0.090909091	1
76	0.023166023	0.01171875	1
77	0.394984326	0.24609375	1
78	0.707198494	0.628449317	0.808510638
79	0.605673759	0.455466667	0.903703704
80	0.813308358	0.722217822	0.930693069

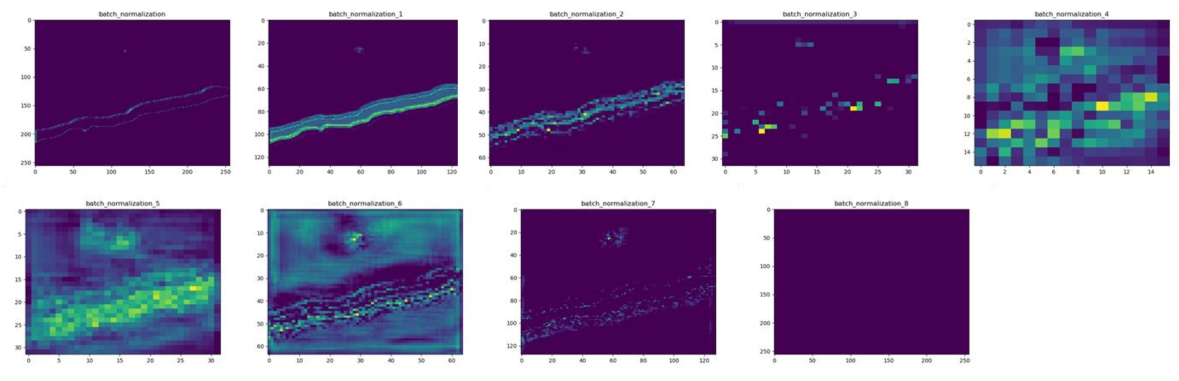
Appendix C: Errors on Images in the Test Batch

Incorrect Critical Width location	Non cracks detected	Cracks >= 0.3mm missed (3px)	Cracks <0.2mm missed
17	3	12	2
20	4	60	6
33			7
56			8
76			12
77			18
			19
			26
			29
			33
			53
			59
			60

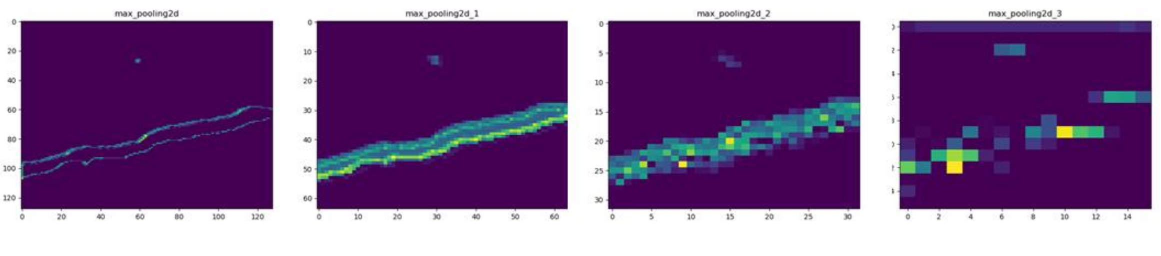
Appendix D: UNET Model Imagery



Convolutional layers sample



Batch Normalisation layers



Pooling Layers