CrackWeb : A modified U-Net based segmentation architecture for crack detection

Sandeep Ghosh, Subham Singh, Amit Maity and Hirak Kumar Maity
College of Engineering and Management, Kolaghat
E-mail: hmait@cemk.ac.in

Abstract. Classical image processing methods demands heavy feature engineering, as well as they are not that precise, when it comes to manual extraction of relevant features in real life scenarios amid to various lighting conditions and other factors. Thus, detection of cracks using methods based on classical image processing techniques fails to provide satisfactory results always. Hence, we have proposed a deep convolutional neural network, that is not based on manual extraction of features as mentioned above. We proposed a modified U-Net architecture, and replaced all of its convolutional layers with residual blocks, inspired from the ResNet architecture. For evaluation of our model Dice Loss is used as our objective function and F1 score as a metric. Other than that, for better convergence and optimization, a learning rate scheduler and AMSGRAD optimizer was utilized.

1. Introduction
Sudden outbreaks or fractures that occurs through the member completely or partly due to various natural and man-made reasons results in formation of crack in concrete structure. More precisely, when the tensile stress exceeds tensile capacity formation of crack takes place. The utility of concrete can be seen in almost everywhere, from buildings, bridges, to other structures. Thus, when a crack occurs in a concrete slab, it can be an indication of some major structural defects in the whole architecture, which holds a potential risk of some serious accident. There are some existing ways to detect cracks, starting from visual inspection and monitoring to other non-destructive techniques (NDT) which uses various image processing methods to segment cracks, but due to unavoidable noise in images, the segmentation of those cracks from its background, isn’t that precise.

In this paper, we tried to design a simple deep learning algorithm, that holds the potential solution to detect cracks of any thickness. As a result, it bypasses the need of manual feature extraction, by learning the essential features necessary for segmenting the crack from its background by classifying each pixel as a crack or not.

The arrangement of the paper is as follow: The scope and motivation for the work is mentioned in Section 2. A brief literature survey is mentioned in Section 3. The proposed work is discussed in Section 4 along with network architecture and mathematical modelling. In Section 5, we have mentioned the benchmark data sets that are used for this work along with the image augmentation methods. The training set-up along with the computation complexity is mentioned in section 6. The performance evaluation is done in Section 7. Finally the conclusion and scope of the future advancements is mentioned in Section 8.
2. Motivation
Formation of cracks in a structure can be termed as a significant indication of deterioration. Thus, early detection of cracks plays a crucial role in keeping the architectural integrity intact. Since, visual inspections as well as NDTs both lacks precision and objectivity in the qualitative analysis. Hence, an automated crack detection technique stands out to be a mandatory replacement. The existing literature presents a variety of techniques using classical as well as AI (Artificial Intelligence) based image processing methods. The motivation here is to compare and present an automated technique based on a Novel architecture which bypasses the need of manual feature extraction and turns out to be an improvement over the existing automated techniques too.

3. Related Works
This portion provides a brief literature review about existing methods and recent works on the segmentation of crack images in concrete structures and pavements. In the existing literature, there are two given methods: Classical methods [1, 2, 3, 4] and Deep learning methods [5, 6, 7, 8, 9]. Oliveira and Correia [1] presented a method for automatic crack detection using survey images captured at high driving speeds. They used a pre-processing technique using morphological filters for reducing pixel intensity variance and then applied dynamic thresholding to identify dark pixels in images, as compared to potential crack pixel. Ai et al. [2] mapped the probabilities of each crack pixel. Later they used a SVM(Support Vector Machine) based method and mapped the probability of crack predictions using multi scale-neighborhood information. Shi et al. [3] manually extracted the features of the annotated dataset and created crack detection network using random forest. Li et al. [4] created a model using multi variate hypothesis test for segmenting the extracted candidate cracks using a method of windowed intensity path.

Bang et al. [5] used fully connected layers as decoders and used deep residual blocks as encoders. Fan et al. [6] distributed the crack images into small patches and labelled each patch as positive or negative based on the presence of crack on the patch and then used Convolutional Neural Networks (CNN) to detect or classify those patches. As a result, the complete inference time for each testing image was very high. Zang et al. [7] used two different CNN for segmentation. Their approach was to first extract the relevant features and then feed the same into a different CNN. Yang et al. [8] used two modules namely pyramid and boosting module. The pyramid module merged two feature maps from two convolutional layers into the decoder block and at the same time, assignment of weights from easy to hard samples respectively was done using hierarchical boosting modules. Jenkins et al.[9] and Nguyen et al. [10] utilised a U-Net based architecture. Lau et al. [11] they have modified the basic U-Net architecture by replacing the encoder blocks with pre-trained ResNet-34 blocks.

4. Proposed technique
The method that we have used here is a supervised learning algorithm, where we tried to optimize an universal function approximator such that it fits as, \( f : x \rightarrow y \). We mapped the crack images \((x)\) to their respective labels, i.e, masks \((y)\). We then optimized the parameterized neural network comprising of multiple layers of different types, having different weights and biases using a gradient descent based optimizer such that a minimised loss function can be obtained by altering the theta \( \theta \) parameter upto a satisfactory level \( L = g(x, y, \theta) \).

4.1. Model Architecture
We have created an U-Net based model architecture. U-Net [12] was classically used for segmenting microscopic cells with low data quantity, but when used for detection of cracks, it produced satisfactory outputs. Therefore, this architecture was ideally suited for the mentioned
work. According to the official U-Net paper “There is large consent that successful training of deep networks requires many thousand annotated training samples”. U-Net represents a network with a training strategy that relies highly on image augmentation for more efficient use of the given data. The architecture involves a symmetric down sampling path that captures the context and an adjacent up sampling path that allows the precise localization of the ROI (Region of Interest). We have customised the original network by replacing all the classic convolutional layers with residual blocks, inspired by the famous ResNet [13]. The main intuition behind this alteration was “identity shortcut connection” that skips one or more layers as shown in Figure 1(a). By virtue of which the vanishing gradient problem that comes with deep neural networks was overcome. Along with that, a full pre-activation layer (shown in Figure 1(c)) with skip connections (Figure 1(b)) were introduced that provides an alternative easier path for the gradients to back propagate.

The proposed network receives a three channel (RGB) image of dimension (256,256,3) and generates an output of dimension (256,256,3). Encoder loads with a stem (Figure 2(a)), that consists of a convolutional layer with 16 kernels of size (3 × 3) each with stride of 1 and padding of zero, followed by a Batch Normalization, a ReLu activation layer and another convolution layer identical to the above. On the other hand, an identical convolutional layer with 16 kernels of size (1 × 1) followed by a batch normalization layer without activation, is fed to the successive layers below. A full pre-activated layer along with a skip connection constitutes a successive block as shown in Figure 2(b). The presented architecture used the following numbers of kernels, \( l \): \([16 \times t = 1, 32 \times t = 2, 64 \times t = 3, 128 \times t = 4 & 256 \times t = 5]\), respectively. With the kernels of size (3 × 3) and stride 2 at each level (\(l=\text{level}\)), followed by a final bottleneck layer with 256 filters and stride 1. The output from this layer is fed to the decoder network, where a single decoder block (as shown in Figure 2(c)) consists of concatenated activations along the channel of dimensions equivalent to its respective encoder block along with up-scaled activations of its previous convolutional
blocks. Each decoder block consist of a UpSampling2D layer with filter size \((2 \times 2)\) followed by a residual block. The complete network architecture is shown in Figure 3.

4.2. Loss Function

The network’s output layer is connected to a sigmoid function such that each pixel in the output layer ranges from zero to one. Which determines the probability of crack in each pixel. As a result, it was noticed that the non-crack pixels outnumbered the crack pixels by a huge ratio. Hence, a huge class imbalance was observed. Thus, to deal with this issue, dice coefficient loss [14] was selected as it directly optimizes the dice score. Moreover, its equivalent to the F1 score too. Dice coefficient is actually twice the area of overlap divided by the total number of pixels in the image. Thus, the dice loss stands out to be:

\[
L = \frac{1}{N} \sum_{y=1}^{N} 1 - \frac{2|y' \odot y|}{|y'| + |y|}
\]

where, prediction and ground truth is depicted as \(y' \in \mathbb{R}^{h \times w}\) and \(y \in \mathbb{R}^{h \times w}\) respectively. The operator \(\odot\) denotes the element-wise multiplication operation. The loss function produces a
scalar value between '0' and '1' such that when the output matches the ground truth it becomes zero, else one.

4.3. Parameter Optimization
The optimizer that we have used here to optimize our model is ADAM and Beyond [15]. Which uses a new exponential moving average AMSGRAD. The AMSGRAD uses a smaller learning rate in comparison to ADAM [16]. In case of ADAM the decrement or decay of learning rate is not guaranteed whereas AMSGRAD uses smaller learning rates, thus it maintains the maximum of all the learning rates until the present time step and uses that maximum value for normalizing the running average of the gradient unlike the learning rates in ADAM or RMSPROP [17]. Thus, it converges better than ADAM or RMSPROP.

The gradient computation is done by using

\[ g_t = \frac{1}{m} \sum_{i=1}^{m} \Delta \theta \mathcal{L}(x, y, \theta_t) \]  \hspace{1cm} (2)

Then the moment estimation is computed as

\[ m_t = \rho_1 m_{t-1} + (1 - \rho_1) g_t, v_t = \rho_2 v_{t-1} + (1 - \rho_2) g_t^2 \]  \hspace{1cm} (3)

where, \( \hat{v}_t = \max(\hat{v}_t-1, v_t) \) and \( \theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{\hat{v}_t}} \). We kept the information of the maximum value of all \( v_t \) until current time step and later used that in the update rule compared to just the current \( v_t \) in ADAM algorithm.

4.4. Learning Rate
The learning rate used here, was not kept constant throughout the training of the data, instead we used a learning rate scheduler, that increases/decreases the learning rate gradually after every fixed set of epochs such that the function can attain the optimum convergence by the end of the training.

5. Dataset and Preprocessing
The performance of the proposed model is evaluated with two different datasets as mentioned below:

- **Crack500** [18] The dataset consists of 500 images and corresponding masks of size (3264 x 2448). 250 images were used for training, 50 for validation and 200 images were used for testing.
- **DeepCrack** [19] The dataset consists of 537 manual annotated images. Divided into 300 training samples and 237 test samples.

Two sample data are shown in Figure 4(a to d) and Figure 5(a to d) from Crack500 and DeepCrack datasets respectively.

5.1. Image Augmentation
All the data are augmented on the fly using the albumentation [20] library. Random flips and rotations are applied with random changes in lighting by increasing/decreasing contrast, gamma & brightness. Random distortions are also applied like elastic distortion, grid distortion and optical distortion.
Figure 4. Sample images and their corresponding masks from Crack500 dataset

Figure 5. Sample images and their corresponding masks from DeepCrack dataset

6. Training
The estimated time taken by our architecture to train on Crack500 and DeepCrack dataset were 46 mins and 10 mins respectively. We trained our model on a Google Colab instance with a 16GB Tesla P100 paired up with an Intel Xeon processor and 12GB of RAM. The model was trained end-to-end with a batch size of 10 and input image size of (256, 256, 3) to compensate for the hardware limitations. The deep learning framework used here is a high level API known as Keras along with Tensorflow 2.2 as backend. Keras is a fast, easy and optimized API for prototyping research works like the one presented in this paper.

7. Results and Discussion
The proposed network is trained on Crack500 and DeepCrack datasets. F1 score is the harmonic mean of precision and recall. The required expressions are shown below:

$$\text{Precision} = \frac{TP}{(TP + FP)}; \text{Recall} = \frac{TP}{(TP + FN)}; \text{F1 - Score} = \frac{(2\text{PrecRec})}{(\text{Prec} + \text{Rec})}$$ (4)

where TP = True positive, FP = False Positives, FN = False Negatives. The experimental results are shown in Figure 6 and Figure 7 for the above mentioned datasets, respectively. A comparative study of our network with other well known existing methods on the mentioned datasets is shown below:

| Method                        | Precision | Recall   | F1     |
|-------------------------------|-----------|----------|--------|
| U-Net by Jenkins et al. [9]   | 0.6811    | 0.6629   | 0.6788 |
| U-Net by Nguyen et al. [10]   | 0.6954    | 0.6744   | 0.6895 |
| CNN by Fan et al. [6]         | 0.7123    | 0.6955   | 0.7056 |
| Split-Attention Network [21]  | 0.7368    | 0.7165   | 0.7295 |
| Lau et al. [11]               | 0.7426    | 0.7285   | 0.7327 |
| **Our method**                | **0.8095**| **0.7409**| **0.7737**|
Table 2. Comparison of various methods on DeepCrack dataset.

| Method           | Precision | Recall | F1   |
|------------------|-----------|--------|------|
| Y. Liu et al. [19] | 0.861     | 0.869  | 0.865|
| Our method       | 0.8535    | 0.905  | 0.878|

Figure 6. Original Image, Ground truth and predicted mask are shown in (a - c) and (d - f), respectively from Crack500 dataset.

Figure 7. Original Image, Ground truth and predicted mask are shown in (a - c) and (d - f), respectively from DeepCrack dataset.

8. Conclusion
The proposed algorithm requires minimal feature engineering unlike any other machine learning algorithms and it achieves state-of-the-art F1 score of about 88 on the DeepCrack data set and 77 on the Crack500 data set. Also the network uses a U-Net like architecture with residual blocks to automate the process of crack segmentation. Though achieving state-of-the-art performances on various datasets, the only drawback of our method is manual annotations of data, which makes the whole process quite expensive. Unsupervised learning holds the potential to resolve...
This issue as it doesn’t demands labelled data.

This is also noticed that the images have uneven illumination that could be addressed with techniques like Histogram Equalization (HE) or Contrast Limited Adaptive Histogram Equalization (CLAHE). Further advancements can be done using better quality images along with more precise labeling of the data. Even more data augmentation techniques may be added to resolve the issue of small dataset.

References

[1] Oliveira H and Correia P 2009 Automatic road crack segmentation using entropy and image dynamic thresholding EUSIPCO 622-626.
[2] Ai D, Jiang G, Kei L S and Li C 2018 Automatic pixel level pavement crack detection using information of multiscale neighborhoods IEEE Access 6:24452-24463.
[3] Shi Y, Cui L, Qi Z, Meng F and Chen Z 2016 Automatic road crack detection using random structured forests IEEE Trans. Intell. Transp. Syst. 17(12) 3434-3445.
[4] Li H, Song D, Liu Y and Li B 2019 Automatic pavement crack detection by multi scale image fusion IEEE Trans. Intell. Transp. Syst. 20(6) 2025-2036.
[5] Bang S, Park S, Kim H, Yoon Y S and Kim H 2018 A deep residual network with transfer learning for pixel-level road crack detection ISARC 35: 1-4.
[6] Fan Z, Wu Y, Lu J and Li W 2018 Automatic pavement crack detection based on structured prediction with the convolutional neural network arXiv:1802.02208.
[7] Zhang A, Wang K C P, Li B, Yang E, Dai X, Peng Y, Fei Y, Liu Y, Li J Q and Chen C 2017 Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network Computer-Aided Civil Infrastructure Engg. 32(10) 805-819.
[8] Yang F, Zhang L, Yu S, Prokhorov D, Mei X and Ling H 2020 Feature pyramid and hierarchical boosting network for pavement crack detection IEEE Trans. Intell. Transp. Syst. 21(4) 1525-1535.
[9] Jenkins M D, Carr T A, Iglesias M I, Buggy T and Morison G 2018 A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks EUSIPCO 2120-2124.
[10] Nguyen N T H, Le T H, Perry S and Nguyen T T 2018 Pavement crack detection using convolutional neural network, SoICT 251256.
[11] Lau S L H et al. 2020 Automated Pavement Crack Segmentation Using U-Net-Based Convolutional Neural Network IEEE Access.
[12] Ronneberger O, Fischer P and Brox T 2015 U-Net: Convolutional Networks for Biomedical Image Segmentation ICMICCAI 234-241.
[13] He K, Zhang X, Ren S and Sun J 2016 Deep Residual Learning for Image Recognition CVPR 770-778.
[14] Sudre C H et al 2017 Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentsations Lecture Notes in Computer Science 240248.
[15] Reddi S J, Kale S, Kumar S 2018 On the Convergence of Adam and Beyond.
[16] Kingma D P, Jimmy B 2015 Adam: A Method for Stochastic Optimization. CoRR.
[17] Hinton G Neural Networks for machine learning online course https://www.coursera.org /learn/neural-networks/home/welcome
[18] Yang F, Zhang L, Yu S, Prokhorov D, Mei X and Ling H 2020 Feature pyramid and hierarchical boosting network for pavement crack detection IEEE Trans. Intell. Transp. Syst. 21(4) 15251535.
[19] Liu Y, Yao J, Lu X, Xie R and Li L 2019 DeepCrack: A deep hierarchical feature learning architecture for crack segmentation Neurocomputing 338 139-153.
[20] Buslaev A, Iglovikov V I, Khvedchenya E, Parinov A, Druzhinin M and Kalinin A A 2020 Albumentations: Fast and Flexible Image Augmentations Information 11 125.
[21] Zhang H, Wu C, Zhang Z, Zhu Y, Zhang Z, Lin H, Sun Y, He T, Mueller J, Mannathu R, Li M and Smola A 2020 ResNeSt: Split-attention networks arXiv:2004.08955.