Classification and Quantification of Concrete Cracks using Deep Learning

Paulson J, Seshadhri M, Varshini P

Kongu Engineering College

Perundurai,Erode

paulsonj.19it@kongu.edu

seshadhrim.19it@kongu.edu

varshinip.19it@kongu.edu

***Abstract*— To ensure functionality and durability, engineering constructions must have fractures identified and repaired. Since cracks are typically discovered through visual inspection, the examiner's personal judgment plays a vital part in both the detection of concrete cracks and the estimation of their characteristics. In the past, this has helped with both the evaluation of the crack phenomenon and the repair of various structures. The number of engineering structures has expanded as a result of industrial development, but relative to that, the level of skill in the crack detecting field has not improved. Therefore, a quicker yet more precise method is required. The image processing method detects the cracks and their characteristics simultaneously. The formulation of the algorithm and the implementation is simple. But other flaws, such mistaking noises for fractures and a lack of accuracy in spotting micro-cracks, have turned into major obstacles for this technology. Finding transverse cracks in concrete structures is another critical problem. Therefore, a thorough analysis of the most recent articles is required to establish an accurate technique. In order to comprehend the potential of this technology, a critical study of the image processing phenomenon and a thorough literature review have been done in this paper. A universal DenseNet121 classification based algorithm with camera pictures for crack identification could be an effective strategy with greater accuracy.**

***Keywords*— Concrete cracks, Classification, Image Processing, Measurement, Deep learning**

I. Introduction

The reliability, safety, and longevity of concrete buildings and bridges are all impacted by cracks, which are a direct manifestation of concrete deterioration. Cracks are a significant consideration in the majority of studies on concrete structures. The measurement of damaged parts of concrete panels and beams at various states of loading were associated by several researchers with specific fracture patterns [1]. Therefore, crack detection is a crucial duty to do on roads, bridges, and buildings to assess the condition of these structures.For example, In the US, more than 100,000 bridges have deck cracks from an early age [2]. Furthermore, the Federal Highway Administration (FHWA) categorised 40% of the 570,000 bridges in the USA as being substandard and in need of repair or renovation, with an estimated cost of $50 billion[2].

A best-known methodology for checking and assessing the condition of concrete structures is human-based visual examination. The human-based manual inspection and result evaluation, however, take a lot of time and are subjective [3]. The inspectors' degree of expertise and experience have a major impact on how accurately damage is diagnosed. Therefore, achieving objectivity and efficiency in damage assessment requires autonomous damage detection [4].

To identify concrete structural deterioration and cracks in photos automatically, photos were collected from various buildings and bridges and digital technologies have built computer models. In this context, numerous earlier studies have created defect detection models, and the majority of these algorithm models have concentrated on the concrete cracks detection in infrastructures, such as clustering, image processing, filter-based models, edge detection, and image filtering based machine learning (ML) [13–25]. The majority of the outstanding efforts enhanced detection of crack through the use of computer vision techniques. Detection of cracks was retrieved by image processing techniques (IPTs), and the extracted features were assessed using computer vision-based approaches.

Concrete cracks are a rather regular occurrence. However, unanticipated concrete cracking that is not expected during the design stage impacts the structure's strength and integrity. As a result, the structure performs poorly over the course of its service life.Depending on the underlying reason of cracking, the type and pattern of cracks will change. The main causes of cracks are exposure to the outdoors, bad construction techniques, inadequate designs, and poor details. Therefore, it is crucial to comprehend the cracks that can develop in concrete. But there are risks in identifying and measuring the cracks in high or dangerous concrete structures and also visual inspection techniques are subjective, laborious and time-consuming. It is not possible to visit buildings frequently for remote area buildings. Manual inspection can also be difficult to perform in case of high risk areas.

Convolutional Neural Network (CNN) architectures are the best machine learning technologies used to tackle challenging issues including image identification, video analysis, and natural language processing. Here, using a crack dataset, three distinct CNN architectures for cracks classification are contrasted. Examination was done on the performance of ResNet50, DenseNet121, and a standard CNN on a dataset with over 40,000 photos. There are two groups of images in the dataset: cracked images and non-cracked images.The 30% of the photos are used for testing, leaving 70% of them for

network training. Then the quantification part was done by using thresholding in Image Segmentation. Thresholding is a sort of image segmentation in which we modify a picture's pixel composition to facilitate analysis. Otsu's thresholding is used to calculate the length ,breadth and angle of the cracks.

Recent years have seen the development of another civil infrastructure fault detection and condition assessment technology based on computer vision. Convolutional Neural Network (CNN) is a modern image processing technique using image dataset, and has a lot of ways to improve the automation by utilising UAVs. By Deep Learning (DL), CNN, an image processing tool, helps to classify an object based on its type, detect an object by pinpointing its location by a rectangular box and segment the image by pixels in accordance with a predetermined pattern .

Historical buildings and structures which are adversely affected by disasters and by human influences are commonly observed in modern times. Rapid and affordable crack detection techniques are essential to track the structural condition of old infrastructures. DenseNet, a deep learning network made specifically for these problems. The effectiveness of this algorithm is assessed and contrasted with cutting-edge techniques like CNN and ResNet. The outcomes demonstrate that DenseNet outperforms the other crack classifiers in terms of accuracy performance.

Three components make up this essay. The first shows how to manipulate data. The second describes the DL classifier's which have input, output and architecture. The final discusses methods for segmenting and processing images that are used to determine the length, breadth, and orientation angle of cracks.In fourth session the result and the severity of cracks were covered based on data gathered on actual cracks and experienced staff measurements.

II. Data Manipulation

The process of preparing, processing, and labeling data in the dataset for the classification of cracks.

A. Data Preparation

Images were taken from a publicly accessible, annotated dataset at Kaggle that was used to train and evaluate concrete cracks. For classifying the images, the data is split into cracked images and uncracked images. At a size of 227 x 227 pixels with RGB channels, it has 20,000 photographs of broken concrete structures and other 20,000 images are uncracked concrete structures. Since the image data are pixel based, the focus of this study is entirely on close-up photographs obtained at near ranges (between 25 and 50 cm), each of which shows a single fracture.

B. Data Processing

Data cleansing and visualisation are the first and most important steps to ensuring a reliable classifier. To create an accurate and realistic model, it is essential to provide high-quality photos with a range of obstacles, such as shadows, roughness, scale, hard edges, tiny holes, and dark background detritus in the dataset. The CNN classifier's accuracy can be significantly impacted by low-quality photos. This problem has been fixed by setting manual selection to exclude any blur, fuzzy, or low-pixel photographs. Additionally, no image data with cracked edges were chosen since the image data will get reduced as it goes through the neural network, which suggests images with cracked edges have a lower likelihood of being detected by the network during training than image data with flaws in the center of the image. It is impossible to tell if such cracks are actually cracked or uncracked, which may cause the training data to establish inaccurate annotations. Each image was downsized to a 227 x 227 pixel input image. This is because the network can scale any desirable pattern greater than the planned pattern when it is trained on small images, but not the other way around.

C. Data Labeling

Every image that is presented to the classifiers will reduce its RGB channels over a time and is converted to a particular level of grayscale to speed up processing since the colour characteristic is not necessary. The 40,000 images used to train the CNN classifiers included a random mixture of cracked and uncracked images. For training, validation, and testing, a split of 56:14:30 was used. Thus, out of those 40,000 images, 22,400 are used only for the training, 12,000 are used for the testing, and the other 5,600 images are used for training.

III. Classification

By training the feeded data, the algorithms which are designed for classification by supervised learning approach will categorise the objects. The algorithm makes use of the dataset provided to learn, grow and how to identify one object from others and group them accordingly.

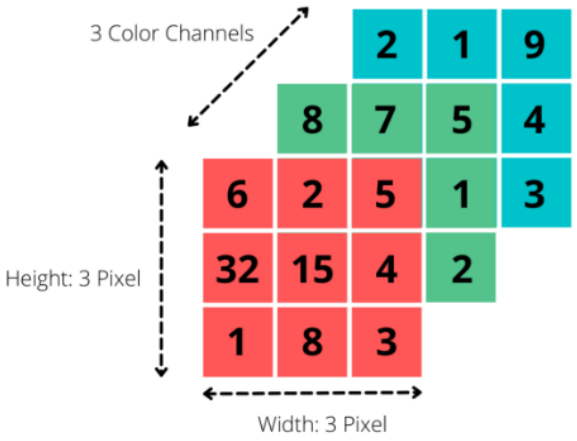

Fig. 1. Configuration of input image

A. CNN Classification

A section of neural networks called Convolutional Neural Networks (ConvNet or CNN) are used for image identification applications. Without losing its data, the convolutional layer reduces the higher dimensionality of given images. The fig.1 illustrates the configuration of the input images.

A 227 x 227 x 3 pixel image is sent to the input layer's first layer (a RGB image with 227 rows and 227 columns).

The final output is predicted by the Sigmoid layer based on the type of classification.The table displays each layer and operation's precise dimensions.

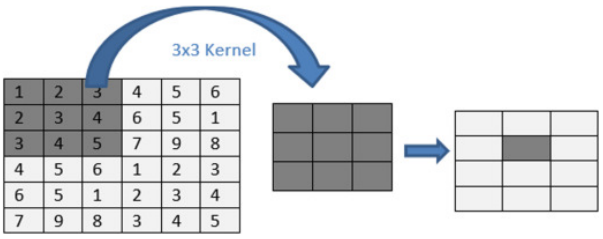

Fig. 2. Kernel

1) *Convolutional Layer:* The bulk of the network is carried by the convolutional layer. A linear procedure known as a convolution involves multiplying an input with a collection of weights. Kernel or filter is a two dimensional weighted array which is multiplied with the image matrix. The filter's size is smaller than the input data, thus when it is multiplied by the input image data, the result is a dot product.

Filters/kernels (K(i,j)) gets multiplied with the input image (I(i,j)) to produce (F(i,j)) the feature map :

$$F(i, j)=(I\times K)(i, j)=\sum m\sum nIm,nK(i-m,j-n)$$

2) *Pooling Layer:* Pooling layer is the second primary structure. It creates a single neuron from each group of the outputs from the preceding layer. Average pooling and maximum pooling are the two most used forms of pooling processes. An average pooling layer takes the mean of its input data and averages it. Max pooling, however, captures the greatest benefit.

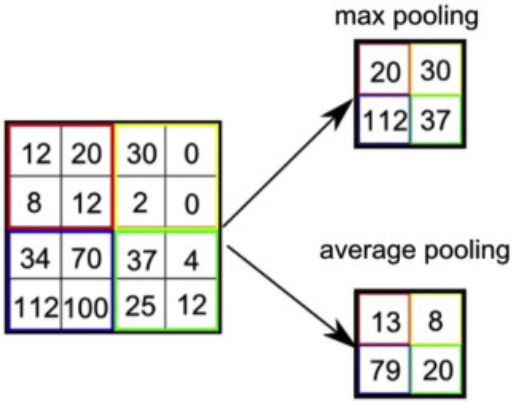

Fig. 3. The Pooling Layer

3) *Activation Layer:* As an activation function, ReLU and sigmoid functions were employed. This yields the typical ReLU activation with default values: max(x, 0).

Sigmoid(x) = 1/(1+exp(-x)) is the activation function called Sigmod. Sigmoid returns a value that is almost equal to zero for small values (<-5), and nearly one for high values (>5).

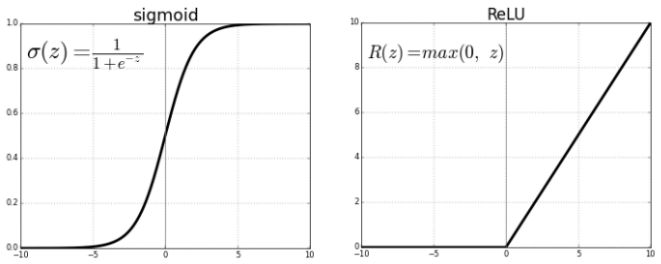

Fig. 4. Comparison of Sigmoid and ReLU

4) *Global Average Pooling 2D:* Global Average Pooling 2D does something different from Flatten.It average pools the spatial dimensions until they are all equal to one. In this instance, averaged values are not retained.The other dimensions are unaffected.

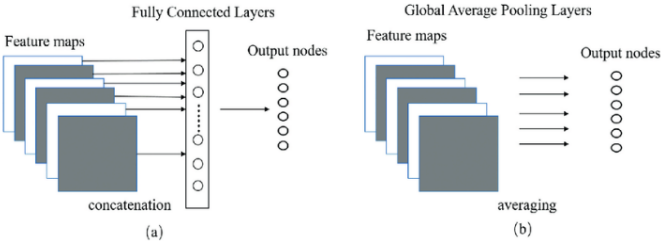

Fig. 5. Fully connected layer and Global Average Pooling layer

5) *CNN Classifier Consideration:* Here, data shuffling was used to try to improve performance. By distributing the data among itself, data shuffling works to avoid overfitting and lower the statistical distribution variance.

A function that converts the one or more network values parameters into a number is called loss function. It represents how good those parameters can enable the CNN to do the task for which it is designed. Here the Binary cross entropy is used as the loss function.

TABLE I
CNN ARCHITECTURE

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 227, 227, 3)]	0
conv2d (Conv2D)	(None, 225, 225, 16)	448
max_pooling2d (MaxPooling2D)	(None, 112, 112, 16)	0
conv2d_1 (Conv2D)	(None, 110, 110, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 55, 55, 32)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0
dense (Dense)	(None, 32)	1056
dense_1 (Dense)	(None, 1)	33

Total params: 6,177
Trainable params: 6,177
Non-trainable params: 0

None

The first block is a convolutional block, which has a size of 3 x 3, the filters were set to 16, the stride set to 1, and the padding remained the same. The second convolutional block, on the other hand, was composed of 32 filters with the same stride and a 3 x 3 size. A stride represents how many steps were made along the activation map that the filter is moved in; its default value is 1. In other terms, the padding is a method for preserving the input and output dimensions of a convolutional procedure. It entails symmetrically by adding the zeros to the input data matrix.

The stride of the pooling layer was 2 for both the first and second blocks. 10 epochs were picked, and a batch size of 32 was determined. For a standard CPU processor, it took two to three hours to train this network, but using a GPU just took a few minutes. The study's mentioned tasks are all carried out using Google Collab.

B. DenseNet121 Classification

The output is passed on to the next convolutional neural network without the first convolutional layer, which receives the input. By this, there are "L" number of connections for each layer, one from one to the next.

However, when the neural network has a lot of layers there raises the vanishing gradient problem. It denotes that the information traveling from one input to the output layer is stretched out and some information might vanish or disappear. this will reduce the network learning capacity..

DenseNets solves this problem by replacing the typical neural network architecture and the connectivity between layers. Every layer in a DenseNet architecture is mapped to another layer.

1) DenseNet121 classifier consideration: Using the table, we can comprehend the DenseNet-121 design. Here, we can see that every dense block has various layers (repetition), each of which includes two convolutions; the bottleneck layer is a 1x1-sized kernel, and the convolution operation is performed by a 3x3 sized kernel.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Fig. 7. DenseNet121 architecture model

A convolutional layer of size 1x1 and an average pooling layer 2x2 and stride size of 2 are in every layer. Consequently, the layers at play are as follows

The 3x3 maximum pooling and a stride size of 2 after a basic convolution layer of 64 7x7 filters and a stride size of 2, followed by two convolutions in Dense Block 1 are repeated six times and layer 1 of the Transition (1 Convolutional + 1 AveragePooling). Again, an other Dense Block with two convolutions every 12 iterations is followed by the another Transition layer (1 Convolutional + 1 AveragePooling), then an other Dense Block with two convolutions every 24 iterations is followed by the another Transition layer , then again another Dense Block with two convolutions every 16 iterations is followed by the GlobalAveragePooling layer, which gets all of the feature maps to perform classification and finally the output layer.

Simplifying, a DenseNet-121 has 4 AvgPool and 120 Convolutions. DenseNets, as opposed to their regular CNN or ResNet equivalents, have attained good performances and better outcomes among competing datasets because they require less parameters and permit reusing features, resulting in better models.

TABLE 2
DENSENET121 ARCHITECTURE

conv5_block16_1_conv (Conv2D)	(None, 7, 7, 128)	126976	['conv5_block16_0_relu[0][0]']
conv5_block16_1_bn (BatchNormalization)	(None, 7, 7, 128)	512	['conv5_block16_1_conv[0][0]']
conv5_block16_1_relu (Activation)	(None, 7, 7, 128)	0	['conv5_block16_1_bn[0][0]']
conv5_block16_2_conv (Conv2D)	(None, 7, 7, 32)	36864	['conv5_block16_1_relu[0][0]']
conv5_block16_concat (Concatenate)	(None, 7, 7, 1024)	0	['conv5_block15_concat[0][0]', 'conv5_block16_2_conv[0][0]']
bn (BatchNormalization)	(None, 7, 7, 1024)	4096	['conv5_block16_concat[0][0]']
relu (Activation)	(None, 7, 7, 1024)	0	['bn[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0	['relu[0][0]']
batch_normalization (BatchNormalization)	(None, 1024)	4096	['global_average_pooling2d[0][0]']
dropout (Dropout)	(None, 1024)	0	['batch_normalization[0][0]']
dense (Dense)	(None, 1024)	1049600	['dropout[0][0]']
dense_1 (Dense)	(None, 512)	524800	['dense[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 512)	2048	['dense_1[0][0]']
dropout_1 (Dropout)	(None, 512)	0	['batch_normalization_1[0][0]']
dense_2 (Dense)	(None, 1)	513	['dropout_1[0][0]']

Total params: 8,618,561
Trainable params: 8,531,841
Non-trainable params: 86,720

C. ResNet50 Classification

Residual Networks also known as ResNet act as a neural network for many computer applications. This model won the ImageNet challenge in 2015. It represented an advanced deep neural network with 150 layers. Before ResNet, implementing the deep neural network was challenging because of the vanishing gradient.

However, just adding layers on top of one another does not increase network depth. Because of vanishing gradient, which occurs when a gradient is repeatedly multiplied as it moves back to its previous layers, deep networks can be challenging to train. By this, the performance becomes super saturated or even begins to fall quickly as it penetrates further.

1) ResNet50 classifier consideration: A 64 kernel convolution layer, each stride of 2, and a kernel of 7x7 size gives 1 layer. then the max pooling size 2 stride. The next convolution has 3 layers: 64 kernels of 1x1 size, 64 kernels of

3x3, and lastly 256 kernels of 1x1. These levels are repeated three times altogether.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 9. ResNet121 architecture model

Then 11,128 kernels of size 1x1 followed by 3x3, 128 kernels and lastly 1x1, 1,512 kernels. This was done totally 4 times for 12 layers. Followed by 1x1, 256 kernels then two more kernels with 3x3, 256 and 1x1 ,1024. Now it is repeated 6 times for 18 layers.

TABLE 3
RESNET50 ARCHITECTURE

activation_45 (Activation)	(None, 7, 7, 2048)	0	['add_14[0][0]']
res5c_branch2a (Conv2D)	(None, 7, 7, 512)	1049088	['activation_45[0][0]']
bn5c_branch2a (BatchNormalization)	(None, 7, 7, 512)	2048	['res5c_branch2a[0][0]']
activation_46 (Activation)	(None, 7, 7, 512)	0	['bn5c_branch2a[0][0]']
res5c_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	['activation_46[0][0]']
bn5c_branch2b (BatchNormalization)	(None, 7, 7, 512)	2048	['res5c_branch2b[0][0]']
activation_47 (Activation)	(None, 7, 7, 512)	0	['bn5c_branch2b[0][0]']
res5c_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	['activation_47[0][0]']
bn5c_branch2c (BatchNormalization)	(None, 7, 7, 2048)	8192	['res5c_branch2c[0][0]']
add_15 (Add)	(None, 7, 7, 2048)	0	['bn5c_branch2c[0][0]', 'activation_45[0][0]']
activation_48 (Activation)	(None, 7, 7, 2048)	0	['add_15[0][0]']
average_pooling2d (AveragePooling2D)	(None, 4, 4, 2048)	0	['activation_48[0][0]']
flatten (Flatten)	(None, 32768)	0	['average_pooling2d[0][0]']
fc1 (Dense)	(None, 256)	8388864	['flatten[0][0]']
fc2 (Dense)	(None, 128)	32896	['fc1[0][0]']
fc3 (Dense)	(None, 1)	129	['fc2[0][0]']

Total params: 32,009,601
Trainable params: 31,956,481
Non-trainable params: 53,120

Lastly , a kernel of size 1x1, 512 was added, then two more kernels of size 3x3, 512 and 1x1, 2048. This was done 3 times, for 9 layers. Then an average pool finishes it completely with a softmax layer.

IV. IMAGE SEGMENTATION

A method to divide an image into various groups known as segments which are used for future processing and analysis is called Image Segmentation. It has used a variety of algorithms. There are numerous methods for separating images, including threshold-based, edge-based, region-based methods, artificial neural network-based Segmentation and clustering-based Segmentation. The segmentation method employed here to handle the tested image was the thresholding approach.

A. Thresholding

Image segmentation method known as thresholding is used for converting original image into binary image or a multi color image by adding threshold value to pixel intensity. All the pixels have values from 0 to 255 range where 0 denotes black and 255 denotes white.

Otsu's Thresholding is an automatic thresholding algorithm that has the following steps. First is processing the input image first, followed by obtaining the image histogram (pixel distribution), computing T(threshold value), and replacing image values with white in places where saturation is more than threshold value T and black in the opposite circumstances.

The aim is to divide the image into two clusters using a threshold value that is determined by minimising the weighted variance of both classes, which are represented by $\sigma_w^2(t)$.

The complete equation can be stated as follows: $\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)$, where t is a threshold , having a value between 0 and 255 inclusive, that divides the probabilities of the two classes $w_1(t), w_2(t)$.

For each pixel value in the two distinct clusters C1, C2, the probability P is determined using the cluster probability functions written as follows: $w_1(t) = \sum_{i=1}^{t} P(i)$, $w_2(t) = \sum_{i=t+1}^{I} P(i)$.

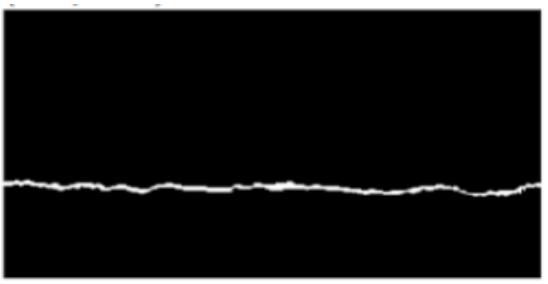

Fig. 10 Thresholded image

The global threshold from a gaussian image is returned via a built-in CV2 function named "THRESH_OTSU," which is utilised in this work.

V. MEASUREMENTS

Before starting the measurement process of the cracks, the given image undergoes preprocessing techniques or methods like changing image channel and size, thresholding, and noise removal.

A. Length of the Crack

The first and most important step in calculating a crack's length is pinpointing the crack's exact location. The maximum separation between the edges of the cracks (A,B,C,B) would be used to represent the crack length. d(i,j) is the maximum length between any endpoints with the crack's position when d(i,j) is taken into account as the length between any two points. Following a comparison of the distances on all sides to calculate the largest length. The Euclidean distance, found in Eq. , is used to determine all of these numbers.:

$$d(i,j) = \sqrt{(p2 - p1)2 + (q2 - q1)2}$$

where p1, p2 are coordinates of p and q1,q2 are coordinates of q.

B. Width of the Crack

The size of a crack significantly influences how much structural damage it will cause to a structure. The largest length between 2 sides on the crack is considered to be the crack width. Determining the crack's width is based on the crack's orientation and the parameters that were established to determine the crack's length. If the fracture is vertical, the height of the image and its breadth are iterated over in a loop (horizontal). These lengths are computed, and then an array is made to hold them. The crack's equivalent width is determined by the largest width and the largest length.

C. Angle of the Crack

The angle was calculated by following the mainline to the diagonal crack which determines its crack length, and the horizontal line beginning from the starting designated as starting point indicates the crack's orientation. The equation in Eq is used to calculate the crack's angle of orientation.

$$cos\theta = \left(\frac{hyp}{adj}\right)$$

TABLE 4
RANGE ALLOWED FOR CONCRETE CRACK WIDTH BASED ON ON SITE EVALUATION

Building Type	**Cracked Place**	**Width**	**Severity**
Load Bearing	Wall	2 cm	Not safe
		1.5 cm	1-2 months
		1 cm	Remedial
		0.5 cm	No effect
Load Bearing	Pillar & Beam	2 cm	Not safe
		1.5 cm	1-2 months
		1 cm	Remedial
		0.5 cm	No effect
Pillar Boarding	Wall	2 cm	Immediate remedial
		1.5 cm	Remedial
		1 cm	Remedial
		0.5 cm	No effect
Pillar Boarding	Pillar & Beam	2 cm	Not safe
		1.5 cm	1-2 months
		1 cm	Remedial
		0.5 cm	No effect

D. Damage Severity

Cracking is a sign that a material has been stretched beyond its strain tolerance. Understanding how cracking occurs requires a variety of issues, including building lifespan, exposure, basement condition, soil capacity, and more.

Each crack has distinct characteristics, such as breadth, length, direction, and size. These characteristics aid in determining the extent of the harm.. In general, the permissible crack width ranges between [0 to 0.5 cm] for the design regulations under consideration. Any crack width that is wider than the permitted values could jeopardise the stability of the building. Cracks in concrete are essentially unavoidable, limiting the frequency and severity of cracks can assist ensure the stability and longevity of the structure. Here, civil engineer's practical experience aids them in determining the extent of the crack.

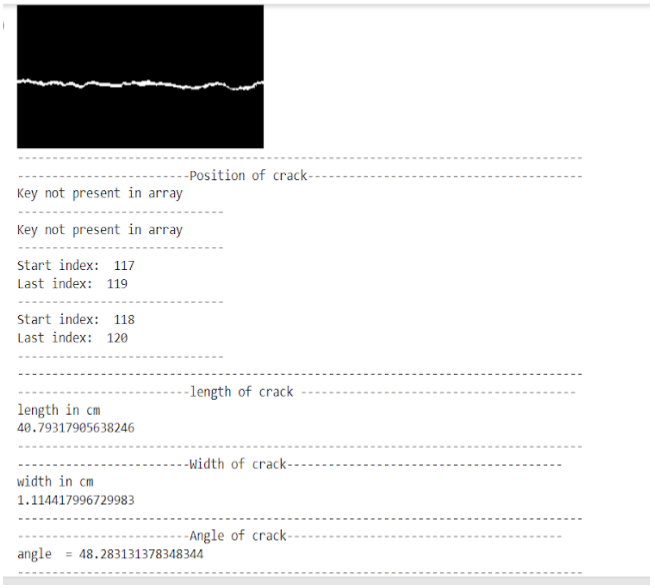

Fig. 11 Measuring the cracks

Based on details gathered form several civil staffs we develop a table 4 to understand the severity of crack.

TABLE 5
TRAIN TEST SPLIT

Class	**Positive and Negative**
No of train images	22400
No of test images	12000
No of validation images	5600

V. RESULT AND DISCUSSION

A. Data Description

An appropriate dataset was collected from Kaggle Surface Crack Detection. About 40,000 images were included in the dataset and all of them are 227 x 227 RGB images. The

images are classified into 2 categories namely, positive and negative. The table 5 describes the number of images used in each class.

B. Sample Dataset

The dataset contains two types of cracked concrete namely positive and negative. In the dataset, positive contains 20,000 images and negative contains remaining 20,000 images. Fig.13 represents the images of two types of concrete cracks.

Fig. 13 Sample Dataset

C. Parameter for Evaluation

CNN, ResNet 50 and DenseNet 121 were used to classify the concrete crack image dataset. The performances can be evaluated using accuracy. Accuracy is used as a metrics for evaluating classification models to measure algorithm's correctness. Informally, the accuracy is the guess to which the model is good or right. In formal terms, accuracy is the ratio of the number of right guesses to the total guesses. The obtained accuracy is

Accuracy = Number of correct predictions / Total number of predictions

In the proposed method, all the three models namely, CNN, ResNet50 and DenseNet121 were trained and tested with the same image dataset and with the same train test split. The results such as accuracy and loss function value obtained for the classification of concrete cracks using CNN, ResNet50 and DenseNet121 algorithms are presented in Table 6.3. The architectures run for 10 epochs.

TABLE 6
RESULT OF CLASSIFIERS

S.No	Architecture	Accuracy (In Percentage)	Loss Function Value
1	CNN	98.45	0.086
2	ResNet 50	98.11	0.123
3	DenseNet 121	98.76	0.035

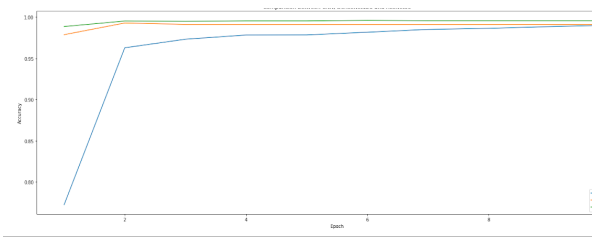

Fig. 14 Comparison of accuracy CNN, DensNet 50, ResNet121

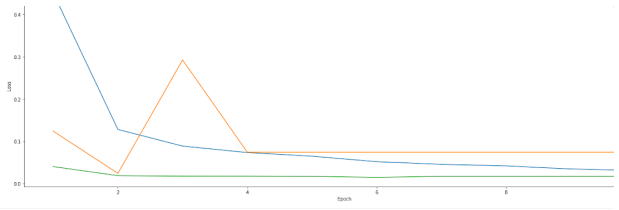

Fig. 15 Comparison of loss CNN, DensNet 50, ResNet121

D. Test and calculate accuracy of model

The above three architectures were used separately to classify the cracked images from uncracked images. The output produced from all the models were studied and displayed below. The basic CNN algorithm gives an accuracy of 98.45% and loss of 0.086.The DenseNet121 algorithm gives an accuracy of , value accuracy of 98.76%, loss of and value loss of 0.035 . The ResNet50 gives an accuracy of , value accuracy of 98.11%, loss of and value loss of 0.123.

VI. CONCLUSION

This study suggests an automated crack detection model for civil structures utilising deep learning methodologies. A neural network model was independently trained using a library of 227x227 pixels of 40,000 images The three classifiers, basic CNN, DenseNet, and ResNet, which categorise images into cracked and uncracked categories, were employed. 40,000 images were used in total, with a split of 56:14:30 (22,400 images were used for training and 5600, 12,000 images, respectively, were used for validation and testing). After undergoing many transformations, including Otsu's approach for image thresholding, to perform any noise removal, picture binarization, and image segmentation, the categorised image was measured.After locating the cracks, the geometrical parameters, such as length, width, and orientation angle, are determined.

In addition, future work will involve certain factors like concrete strength, exposure etc. and the international code for concrete cracks can be included in measuring the severity of the damage.

REFERENCES

[1] Song, L., Sun, H., Liu, J., Yu, Z. and Cui, C., 2022. *Automatic segmentation and quantification of global cracks in concrete structures based on deep learning*. Measurement, 199, p.111550.

[2] Mohammed, M.A., Han, Z. and Li, Y., 2021. *Exploring the Detection Accuracy of Concrete Cracks Using Various CNN Models*. Advances in Materials Science and Engineering, 2021.

[3] S. Li and X. Zhao, "*Automatic crack detection and measurement of concrete structure using convolutional encoder-decoder network*," IEEE Access, vol. 8, pp. 602–618, 2020.

[4] S. Li, X. Zhao, and G. Zhou, "*Automatic pixel‐level multiple damage detection of concrete structure using fully convolutional network*," Computer-Aided Civil and Infrastructure Engineering, vol. 34, no. 7, pp. 616–634, 2019.

[5] Rezaie, A., Achanta, R., Godio, M. and Beyer, K., 2020. *Comparison of crack segmentation using digital image correlation measurements and deep learning*. Construction and Building Materials, 261, p.120474.

[6] Chen, K., Reichard, G., Xu, X. and Akanmu, A., 2021. *Automated crack segmentation in close-range building façade inspection images using deep learning techniques*. Journal of Building Engineering, 43, p.102913.

[7] Zhang, C., Jamshidi, M., Chang, C.C., Liang, X., Chen, Z. and Gui, W., 2022. *Concrete Crack Quantification using Voxel-BasedReconstruction and Bayesian Data Fusion*. IEEE Transactions on Industrial Informatics.

[8] Protopapadakis, E., Voulodimos, A., Doulamis, A., Doulamis, N. and Stathaki, T., 2019. *Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing*. Applied intelligence, 49(7), pp.2793-2806.

[9] Flah, M., Suleiman, A.R. and Nehdi, M.L., 2020. *Classification and quantification of cracks in concrete structures using deep learning image-based techniques.* Cement and Concrete Composites, 114, p.103781.

[10] Ali, R., Chuah, J.H., Talip, M.S.A., Mokhtar, N. and Shoaib, M.A., 2022. *Structural crack detection using deep convolutional neural networks.* Automation in Construction, 133, p.103989.

[11] Yang, Q., Shi, W., Chen, J. and Lin, W., 2020. *Deep convolution neural network-based transfer learning method for civil infrastructure crack detection*. Automation in Construction, 116, p.103199.