**Chapter 5**

**Methodology**

This chapter consists of the work flow throughout the research. It includes data collection, building databank for training and validation set, developing deep CNN for training, classifying crack and non crack images.

**5.1 Data Source**

The datasets are collected [55]. The datasets contains images of various concrete surfaces including cracks and non cracks. The image data are divided into two as negative(non crack) and positive(crack) in separate folder for classifying images. Each class contains 20000 images with a total of 40000 images with 227 x 227 x 3 pixels (RGB). No data augmentation is applied.

Fig 5.1 No of Crack and Non crack images

So the number of positive and negative images is same. All the images are in JPG format. Figure 5.2 some crack and non crack images sample from dataset.

****

Fig 5.2: Non Crack and Crack Images [55]

**5.1.2 Data Preprocessing:**

As mentioned earlier we have a large number of dataset (40,000). Feeding this number dataset in to CNN will increase the complexity and computational time. To reduce computational time and complexity the images are reshaped into 150 x 150 pixels. And after that the images are again rescaled by multiplying each pixel with (1/255). Randomly 2000 images were taken for validation. And the rest were used for training the architecture. Figure 5.3 shows a crack reshaped image.

** **

227 x 227 pixel crack image 150 x 150 pixel crack image

Fig 5.3: Example of a reshaped crack image

**5.2 Model Architecture and Training**

The implemented architecture consists of 6 convolutional layers where maxpooling layer was inserted after each convolutional layer which is shown in figure 8.4. The number of 256 filters of different sizes is used in each convolutional layers. For example, 1x1 size was used in convolution 1, 2, 3 and 6. 3x3 filter is used in convolution 4 and 5. ReLU non linearity was used in each convolution layer. Three hidden dense layer was used in the fully connected layer. The final fully connected layer consists of only one neuron with “sigmoid” non-linearity to produce the class score.

The weights are initialized with ‘he\_uniform’ initialization method [46]. It draws samples from a uniform distribution within [-limit, limit], where limit = sqrt(6 / fan\_in) (fan\_in is the number of input units in the weight tensor). . The network is trained all over using Adam optimizer with initial standard parameters. The model is trained with mini batches of size 64. Figure 5.4 shows the architecture of implemented convolutional neural network. No data augmentation was needed. Besides binary cross entropy loss function was used.

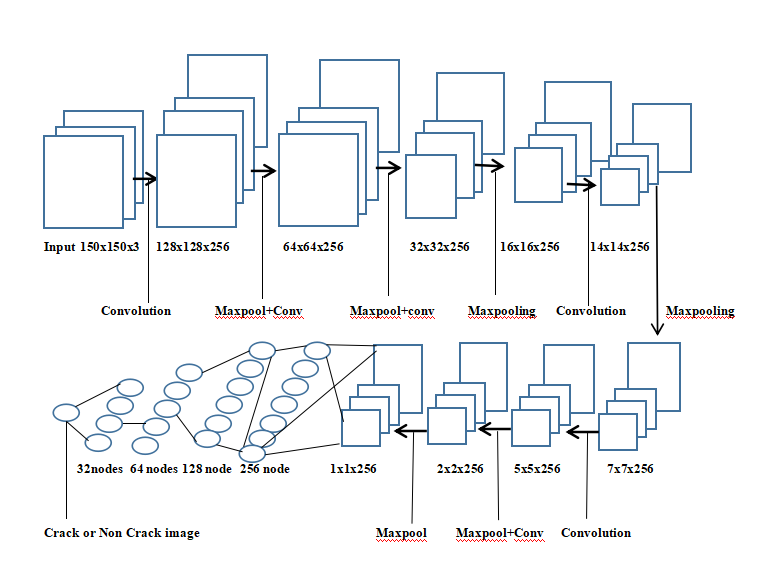


Fig 5.4: The architecture of implemented CNN

From the figure we can see the input of the first layer was 150x150x3. In each layer 256 filters were used. It was initialized with ‘he\_uniform’ kernel initializer. 6 convolutional layers were used. After that, fully connected layer was added. The final output was 1 node that indicates either a crack or non crack image. When the model was implemented completely, several crack or non crack images were tested. Table 5.1 shows the model summary.

Table 5.1 Model Summary

|  |  |  |
| --- | --- | --- |
| **Layer(Type)** | **Output Shape** | **No of Parameter** |
| **Input** | (150, 150, 3) | 0 |
| **Convolution1** | (128, 128, 256) | 1024 |
| **Maxpooling** | (64, 64, 256) | 0 |
| **Convolution2** | (64, 64, 256) | 65792 |
| **Maxpooling** | (32, 32, 256) | 0 |
| **Convolution3** | (32, 32, 256) | 65792 |
| **Maxpooling** | (16, 16, 256) | 0 |
| **Convolution4** | (14, 14, 256) | 590080 |
| **Maxpooling** | (7, 7, 256) | 0 |
| **Convolution5** | (5, 5, 256) | 590080 |
| **Maxpooling** | (2, 2, 256) | 0 |
| **Convolution6** | (2, 2, 256) | 65792 |
| **Maxpooling** | (1, 1, 256) | 0 |
| **Flatten** | (256, 1) | 0 |
| **Dense** | (128, 1) | 32896 |
| **Dense** | (64, 1) | 8256 |
| **Dense** | (32, 1) | 2080 |
| **Sigmoid** | (1) | 33 |

The crack detection is a binary classification task. The output is a binary label ; representing the absence of crack or not respectively. In the training set, weighted binary cross entropy loss is optimized. The loss is defined as:

Where ti and si are the groundtruth and the CNN score for each class i in C. As **usually an activation function (Sigmoid / Softmax) is applied to the scores before the CE Loss computation**, we write f(si) to refer to the activations [56].

**5.3 Conclusion**

The chapter is about building a deep CNN architecture. A number of total 1,421,825 parameters are used.

**CHAPTER 6**

**Results and Performance Analysis**

**6.1 Environment**

The CNN model was trained in kaggle notebook [57]. The notebook editing session allows 9 hours execution time. It has 20 Gigabytes of auto saved disk including 4 CPU cores, 16 Gigabytes of RAM. And the GPU specs with 2 CPU cores and 13 Gigabytes of RAM. It provides NVIDIA Tesla P100. Keras API on top of TensorFlow (CUDA toolkit 9.0, cuDNN SDK v7 and python 3.6) were used [58].

**6.2 Experimental Analysis**

The data set is divided into training and validation set randomly. 2000 images are selected for validation and the rest for training. The results presented in this work is based on accuracy and f1 score [59] which are described by the following equations:

(20)

(21) 

Where tp, tn, fp, fn represent true positive, true negative, false positive and false negative respectively. Recall is defined as the fraction of the relevant instances in a dataset that is successfully retrieved and precision expresses the proportion of the data points the model says is relevant actually are relevant.

CNN has the advantage of learning features automatically instead of manual feature extraction techniques. The self-learning ability of CNN model makes it more convenient than the traditional learning system. Figure 6.1 shows annotated concrete crack images.

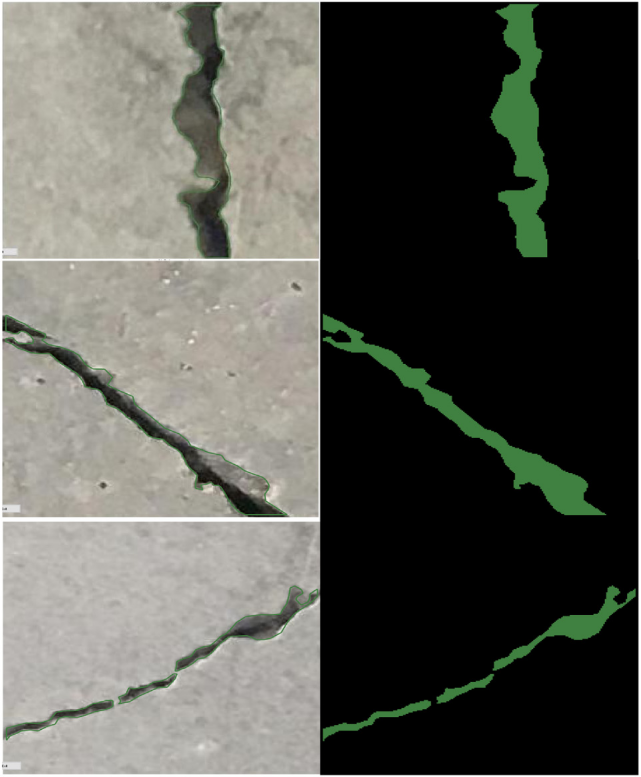


Fig 6.1: Example of annotated concrete crack images [60]

From the figure we can see, cracks are segmented by feature extraction method of CNN. The accuracy curve for the training set (38000 images) and validation set (2000 images) is showed in figure 6.2 for 50 epochs.

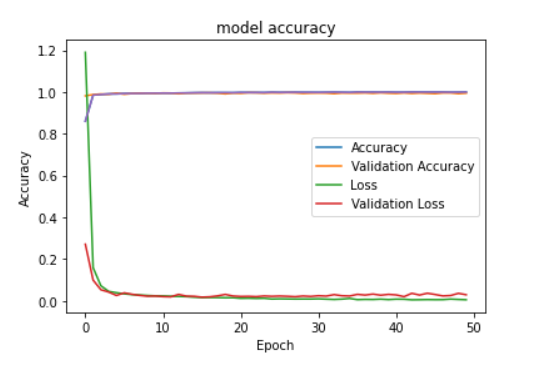


Figure 6.2: Accuracy curve for training and validation

From the figure we can see, by the increasing number of epochs the proposed model gets more accuracy with a decreasing factor of loss. It took around 233s for first epoch and gradually the time was minimized. Using equations (20) and (21), the proposed system achieved 99.45% accuracy and 99.43% f1 score(validation) which outperformed the previous state of the art which is shown in table 6.1. A recent research [10] had the validation accuracy of 99.39%. Another more recent research [11] had an accuracy of 99.98% using VGG16 and InceptionV3 classifiers.

Table 6.: The proposed system achieved the state of the art result:

|  |  |
| --- | --- |
| **Accuracy (Validation Set)** | **F1 Score (Validation Set)** |
| 99.45% | 99.43% |

The reasons behind this accuracy is, 6 convolutional layers were used including maxpooling layer in each. The accuracy of the network highly depends on the depth of CNN architecture [60]. ‘ReLU’ activation was used in convolutional layer. ‘ReLU’ is computationally efficient and converges much faster than most other activation functions [14]. It computes the function  **and the activation being threshold at zero. Compared to ‘Sigmoid’ function which has exponential operations ‘ReLU’ can be implemented by simply thresholding a matrix of activation at zero.**

**The kernels are initialized with ‘he\_uniform’ kernel initializer [46]. It simply draws samples from a uniform distribution within [-limit, limit]. Where,**

***limit = sqrt(6/fan\_in)*  (23)**

**Where *fan\_in* is the number of input units in the weight tensor. So the kernels were also balanced. Three ‘fully connected’ layers were used in the network. To reduce the overfitting L2 regularization was used.** It can be applied by explicitly penalizing the

square magnitude of all parameters in the target. For every weight w in the network, the term is added to the objective, where is the regularization strength. It regularizes the parameters that constrain, regularizes or shrinks the coefficient estimates towards zero.

‘Sigmoid’ function was used in output layer. The network was trained with Adam optimizer with initial standard parameters. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data [61]. As I used a large number of data and parameters (1,421,825) Adam optimizer was well suited for the network. The images were also noisy, so it was appropriate for those problems. And binary cross entropy function was used to reduce to the loss.

**6.3 Conclusion**

The proposed method achieved a quite good accuracy detecting crack images. This methodology can contribute a lot in detecting cracks in civil engineering sectors.