

# **TRAIN TRACK CRACK CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS**

Submitted in partial fulfillment of the requirements for the  
Award of Bachelor of Engineering Degree in Computer  
Science and Engineering

By

**Rahul B (Reg.no.38110442)**

**Mukilan V M (Reg .no.No.38110337)**



**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING SCHOOL OF COMPUTING  
SATHYABAMA INSTITUTE OF SCIENCE AND  
TECHNOLOGY JEPPIAAR NAGAR, RAJIV GANDHI  
SALAI,  
CHENNAI – 600119, TAMIL NADU**

**AUGUST 2022**



# **SATHYABAMA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **Rahul B (Reg.no.38110442), Mukilan V M (Reg.no.38110337)** who carried out the project entitled “**TRAIN TRACK CRACK CLASSIFICATION**” under my supervision from November 2021 to March 2022.

**Internal Guide**

**Dr.S.Raja Shree ME., Ph.D.,**

**Head of the Department**

**Dr. L. Lakshmanan, M.E., Ph.D.**

**Submitted for Viva voce Examination held on \_\_\_\_\_**

**Internal Examiner**

**External Examiner**

## **DECLARATION**

We **Rahul B (38110442)**, **Mukilan V M (38110337)** hereby declare that the Project Report entitled **TRAIN TRACK CRACK CLASSIFICATION** done by me under the guidance of: **Dr.S.Raja Shree ME. Ph.D.**, (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE: CHENNAI**

**SIGNATURE OF THE CANDIDATE**

## **ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide : **Dr.S.Raja Shree .ME. Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

Train Track Crack Classification a railway crack detection system. This paper presents the classification system that classify any crack in the railway tracks by using deep learning with convolutional neural networks (CNNs). In railway network Accidents are the major concern in terms of railway track unidentified crack in rail tracks in Indian railway. About Most of the accidents are occurring at railway track crack in railway tracks resulting in loss of precious life and loss of economy It is required to monitor the track health condition frequently using an crack classify system.

This project prevents train derailment by classify cracks in railway track using image processing technology. To propose a solution for track crack recognition that uses a combination of Convolutional Neural Network and specific image pre-processing steps. It described the innovative solution that provides efficient image processing and deep learning with convolutional neural networks.

A variety of neuron-wise and layer-wise visualization methods were applied using a CNN, trained with a publicly available from given image dataset. So, it's observed that neural networks can capture the colours and textures of lesions specific to respective cracks in train tracks, loss of precious life and loss of economy It is required to monitor the track health condition frequently using an crack which resembles human decision-making.

## TABLE OF CONTENTS

| CH.NO | TITLE                        | PAGE.NO |
|-------|------------------------------|---------|
|       | Abstract                     | I       |
|       | List Of Figures              | II      |
|       | List Of Abbreviations        | III     |
| 1     | <b>1.1 Introduction</b>      | 3       |
|       | 1.2 Artificial Intelligence  | 4       |
| 2     | <b>2.1 Literature Review</b> | 6       |
| 3     | <b>Scope Of The Project</b>  |         |
|       | 3.1 Existing system          | 9       |
|       | 3.2 Drawback                 | 9       |
|       | 3.3 Data Science             | 9       |
|       | 3.4 Preparing The Dataset    | 10      |
|       | 3.5 Proposed System          | 10      |

|   |                                                                                                                                                                                                                                       |                                                      |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| 4 | <b>Working Theory Of Our Project</b><br><br>4.1 Methodology<br>4.2 Objectives<br>4.3 Scope<br>4.4 Overview of the system<br>4.5 Project Requirements<br>4.6 Splitting the dataset<br>4.7 Data Flow Diagram                            | <br><br>11<br>11<br>12<br>12<br>13<br>14<br>15       |
| 5 | <b>Feasibility Study</b><br><br>5.1 System Architecture<br>5.2 Use Case Diagram<br>5.3 Class Diagram<br>5.4 Activity Diagram<br>5.5 Sequence Diagram<br>5.6 ER Diagram<br>5.7 Collaboration Diagram<br>5.8 Environmental Requirements | <br><br>17<br>18<br>19<br>19<br>20<br>20<br>21<br>21 |
| 6 | <b>Module Description</b><br><br>6.1 Import Image From The Dataset<br>6.2 To Train The Module By the Given Image Dataset<br>6.3 Working Process of Layers in CNN Model<br>6.4 Classification Identification                           | <br><br>22<br>23<br>24<br>27                         |
| 7 | <b>Conclusion</b><br><br>7.1 Conclusion<br>7.2 Future Work                                                                                                                                                                            | <br><br>30<br>30                                     |
| 8 | <b>Reference</b>                                                                                                                                                                                                                      | 31                                                   |
| 9 | <b>Appendices</b><br><br>Source Code                                                                                                                                                                                                  | <br><br>32                                           |

|  |                    |    |
|--|--------------------|----|
|  | Output Screenshots | 46 |
|--|--------------------|----|

## LIST OF FIGURES

| SL.NO | TITLE               | PAGE.NO |
|-------|---------------------|---------|
| 1     | Data Flow Diagram   | 15      |
| 2     | System Architecture | 17      |
| 3     | Workflow Diagram    | 18      |
| 4     | Use Case Diagram    | 18      |
| 5     | Class Diagram       | 19      |
| 6     | Activity Diagram    | 19      |

|   |                       |    |
|---|-----------------------|----|
| 7 | Sequence Diagram      | 20 |
| 8 | ER Diagram            | 20 |
| 9 | Collaboration Diagram | 21 |

## LIST OF ABBREVIATIONS

CNN - Convolutional Neural Networks  
 CRFs - Conditional Random Fields  
 CTC - Connectionist Temporal Classification  
 CTR - Collaborative Topic Regression  
 MLE - Maximum Likelihood Estimation  
 MRR - Mean Reciprocal Rank  
 NER - Named Entity Recognition  
 NERQ - Named Entity Recognition in Query

# **CHAPTER – 1**

## **INTRODUCTION**

### **1.1 Introduction**

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

## **1.2 ARTIFICIAL INTELLIGENCE**

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of “intelligent agents” any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term “artificial intelligence” to describe machines that mimic “cognitive” functions that humans associate with the human mind, such as “learning” and “problem solving”, however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require “intelligence” are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years

since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an “AI winter”), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field’s long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence “can be so precisely described that a machine can be made to simulate it”. This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

## **CHAPTER -2**

### **Literature Review**

#### **2.1 Literature Review**

**Title** : Analysis of cracking on running surface of rails

**Author:** Zdenka Popovic

**Year** : 2013

Montenegrin railways are a part of the European railway network. In the scope of realization of interoperability of the European railway system, rail infrastructure managers are required to have infrastructure subsystem maintenance plans for each conventional railway line [1]. This plan should inter alia include inspection and an appropriate strategy against the rolling contact fatigue (RCF). The rail failure or damage generally results from fatigue cracks and reduces the rail service life, increases the cost of maintenance, and may cause train derailment [2]. An increased traffic density, higher axle load and speed, as well as lubrication of rails, these are all factors that contribute to RCF and are a serious hazard to rail traffic. On the other hand, problems due to RCF can be reduced by applying an appropriate track geometry, correct wheel/ rail contact geometry, and better maintenance strategies. An adequate maintenance strategy should contribute to a longer rail service life, lower rail maintenance costs, and greater safety of railway traffic.

**Title:** Automatic Crack Detection and Classification Method for Subway Tunnel Safety

**Author:** Wenyu Zhang, Zhenjiang Zhang \*, Dapeng Qi and Yun Liu

**Year** : 2014

Cracks are an important indicator reflecting the safety status of infrastructures. This

paper presents an automatic crack detection and classification methodology for subway tunnel safety monitoring. With the application of high-speed complementary metal-oxide-semiconductor (CMOS) industrial cameras, the tunnel surface can be captured and stored in digital images. In a next step, the local dark regions with potential crack defects are segmented from the original gray-scale images by utilizing morphological image processing techniques and thresholding operations. In the feature extraction process, we present a distance histogram based shape descriptor that effectively describes the spatial shape difference between cracks and other irrelevant objects. Along with other features, the classification results successfully remove over 90% misidentified objects. Also, compared with the original gray-scale images, over 90% of the crack length is preserved in the last output binary images. The proposed approach was tested on the safety monitoring for Beijing Subway Line 1. The experimental results revealed the rules of parameter settings and also proved that the proposed approach is effective and efficient for automatic crack detection and classification.

**Title** : Crack detection using image processing: A critical review and analysis

**Author:** Arun Mohan a, \*, Sumathi Poobal

**Year** : 2017

Cracks on the concrete surface are one of the earliest indications of degradation of the structure which is critical for the maintenance as well the continuous exposure will lead to the severe damage to the environment. Manual inspection is the acclaimed method for the crack inspection. In the manual inspection, the sketch of the crack is prepared manually, and the conditions of the irregularities are noted. Since the manual approach completely depends on the specialist's knowledge and experience, it lacks objectivity in the quantitative analysis. So, automatic image-based crack detection is proposed as a replacement. Literature presents different techniques to automatically identify the crack and its depth using image processing techniques. In this research, a detailed survey is conducted to identify the research challenges and the achievements till in this field. Accordingly, 50 research papers are taken related to crack detection, and those research papers are reviewed. Based on the review, analysis is provided based on the image

processing techniques, objectives, accuracy level, error level, and the image data sets. Finally, we present the various research issues which can be useful for the researchers to accomplish further research on the crack detection.

**Title:** Crack Detection in Concrete Tunnels Using a Gabor Filter Invariant to Rotation

**Author:** Roberto Medina 1, José Llamas 1 ID

**Year:** 2017

In this article, a system for the detection of cracks in concrete tunnel surfaces, based on image sensors, is presented. Both data acquisition and processing are covered. Linear cameras and proper lighting are used for data acquisition. The required resolution of the camera sensors and the number of cameras is discussed in terms of the crack size and the tunnel type. Data processing is done by applying a new method called Gabor filter invariant to rotation, allowing the detection of cracks in any direction. The parameter values of this filter are set by using a modified genetic algorithm based on the Differential Evolution optimization method. The detection of the pixels belonging to cracks is obtained to a balanced accuracy of 95.27%, thus improving the results of previous approaches.

**Title:** Automotive Crack Detection for Railway Track Using Ultrasonic Sensorz Article

**Author:** Sopanharith Sam, V.Ganesh

**Year:** 2016

In the fast developing country, people are facing many accidents; it would be undesirable for any nation to losing their life for unwanted cause. Railways are one of the important transports in India. There is a need for manual checking to detect the crack on railway track and always railway personnel takes care of this issue, even though the inspection is made regularly. Sometimes the crack may unnotice. Because of this the train accident or derailment may occur. In order to avoid this situation and automate the railway crack detection has been proposed. Here ultrasonic sensor is used to detect the crack in the

railway track by measuring distance from track to sensor, if the distance is greater than the assigned value the microcontroller identifies there is a crack, also it tells the exact location of the crack by the formula “ $\text{DISTANCE}=\text{SPEED}*\text{TIME}$ ”. While the checking process is going on, the train may approach, it is identified by the vibration sensor and gives alert to the microcontroller, thereby shrinks the size of the robot between the two tracks. After the train has crossed it returns to its normal position and continue its checking process.

## **CHAPTER 3**

### **SCOPE OF THE PROJECT**

#### **3.1 Existing system**

Accurate train positioning is crucial for train safety. In this paper, we propose a train positioning method which fuses vision and millimetre-wave radar data. The proposed method contains two parts: loop closure detection (LCD) and radar-based odometry. The loop closure detection part fuses the convolutional neural network (CNN) features and the line features to achieve accurate key location detection. In this paper, we proposed a train positioning method which fuses vision and millimetre-wave radar data. The proposed framework includes the loop closure detection part and the radar-based odometry part. . Loop closure detection eliminated the cumulative error when the train detects a key position, and the radar-based odometry can achieve the train position estimation of the train, and therefore realize the positioning of the train on the whole railway line.

#### **3.2 Drawback**

It has not focused on increasing the recognition rate and classification accuracy of severity of the crack.

#### **3.3 Data Science**

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

#### **Required Skills for a Data Scientist:**

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

### **3.4 Preparing The Dataset**

This dataset contains approximately 300 train and 20 test image, which were then classified into 2 classes:

- DEFECTIVE
- NON-DEFECTIVE

### **3.5 Proposed System**

The recent success of convolutional neural networks (CNNs) in tasks such as object classification extends to the problem of train track crack classification. In the following sections, we will present an overview of our problem to classify images of train track into track crack identification. Many established image recognition systems use standard deep learning and extracted features, which do not have

significant performance when applied to previously unseen data

We implemented three different classifiers from scratch A baseline classifier with one convolutional layer A CNN with a fixed size of five convolutional layers A deeper CNN with a parameter number of convolutional layers, filter dimensions, and number of filters. For each of these models, we tuned parameters including learning rate, regularization, and dropout.

## **CHAPTER 4**

### **WORKING THEORY OF OUR PROJECT**

#### **4.1 Methodology**

Load the data

- Loading the given dataset
- Import required libraries packages

**Pre-process the data**

- Reshape, data augmentations

**Define model**

- Sequential or Functional
- Number of layers to be used, Number of nodes to be used in the model, Evaluation metrics

**Compile the model**

- Define loss function, optimizer, weights and bias.

## **4.2 Objectives**

The goal is to develop a deep learning model for Train track crack image classification by convolutional neural network algorithm for potentially classifying the results in the form of best accuracy by comparing the CNN architectures.

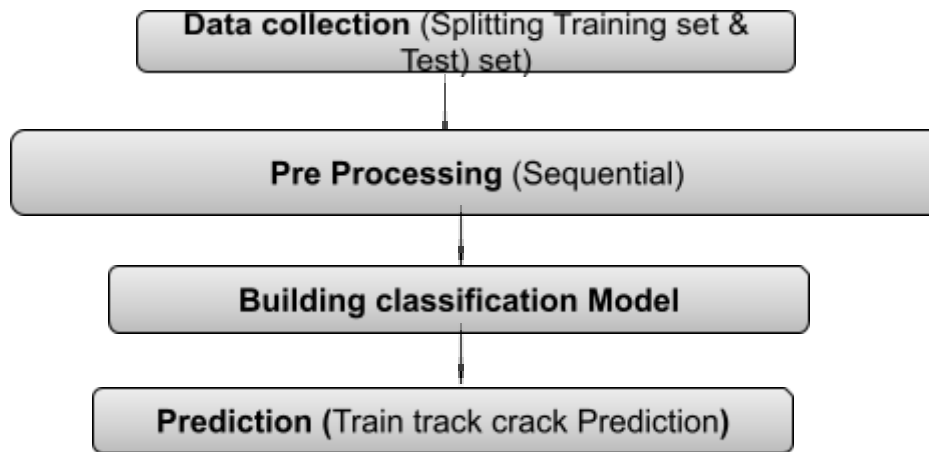
## **4.3 Scope**

The scope of this paper is to implement and investigate how different CNN methods impact the prediction. The scope in our research is to split this classes into Train track crack and normal images. This can be classified into 2 classes. Besides, extracting hand crafted features from raw images after different processing is one of the best scopes we have worked in this research. The objective of using this method is to get some features which are specifically responsible for the train track crack prediction.

## **4.4 Overview of the system**

- Define a problem
- Gathering image data set
- Evaluating algorithms
- Detecting results

The steps involved in Building the data model is depicted below.



#### **4.5 Project Requirements**

##### **General:**

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
  - A. Hardware requirements
  - B. software requirements

##### **Functional requirements**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists

requirements of a particular software system. The following details to follow the special libraries like tensorflow, keras, matplotlib.

### **Non-Functional Requirements**

Process of functional steps,

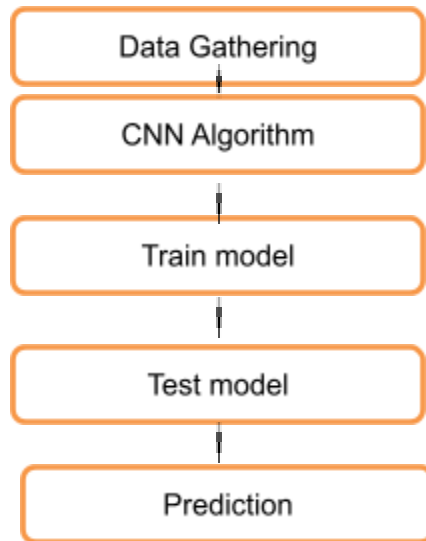
1. Problem define
2. Preparing data
3. Evaluating algorithm
4. Improving results
5. Prediction the result

### **4.6 Splitting the dataset**

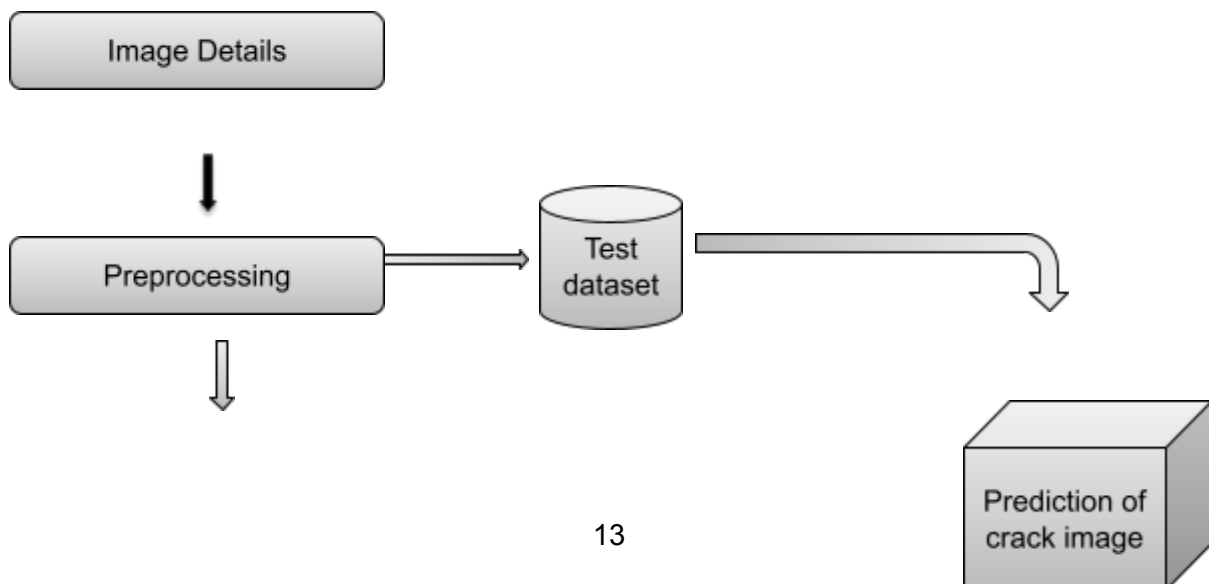
The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

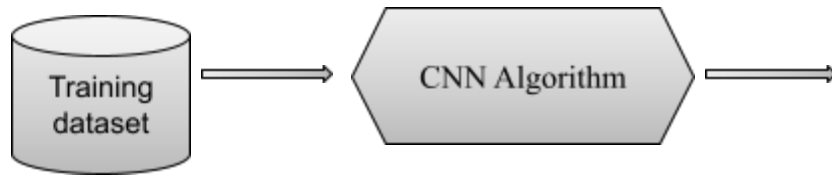
### **Construction of a Detecting Model**

**Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly.**

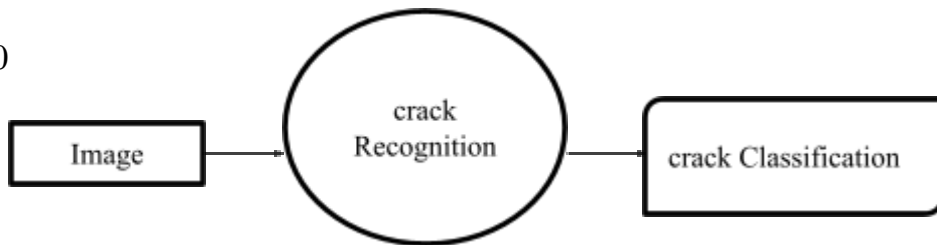


#### 4.7 Data Flow Diagram

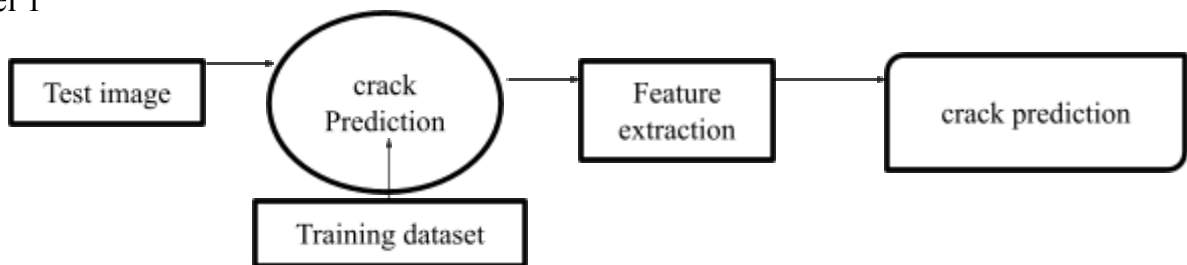




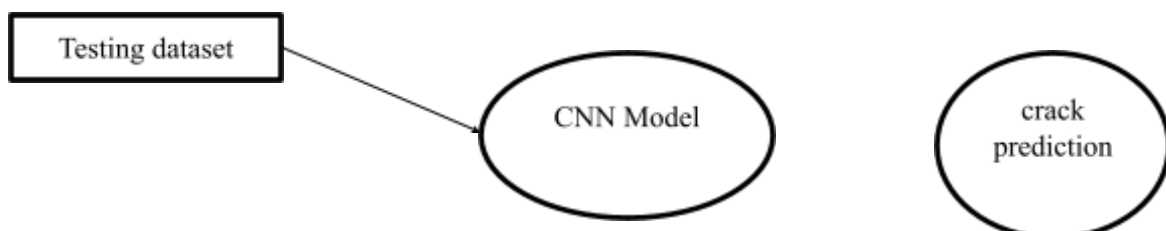
Level 0



Level 1



Level 2

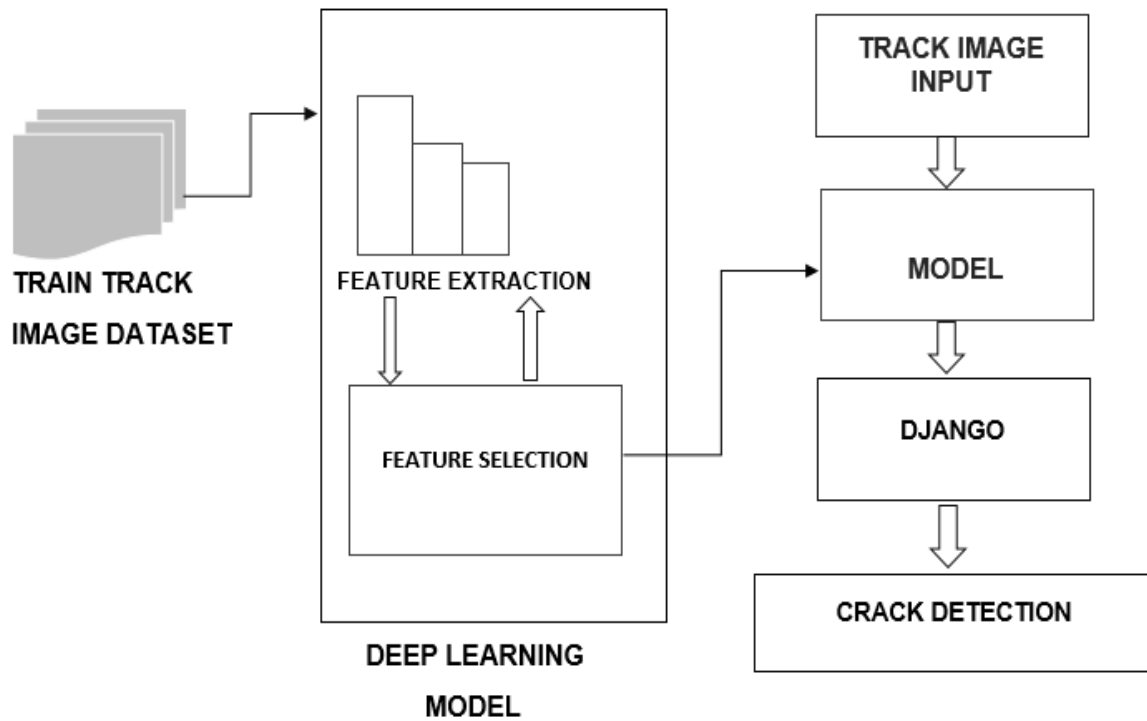




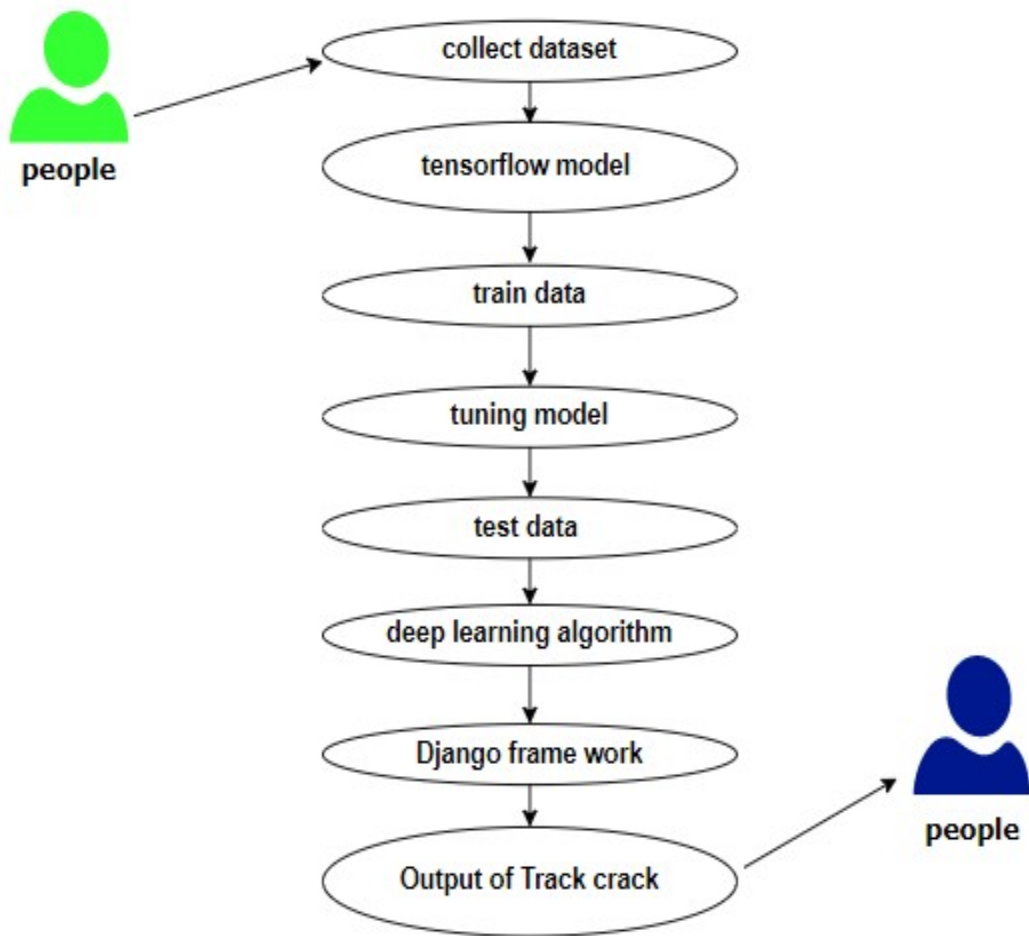
## **CHARTER 5**

### **FEASIBILITY STUDY**

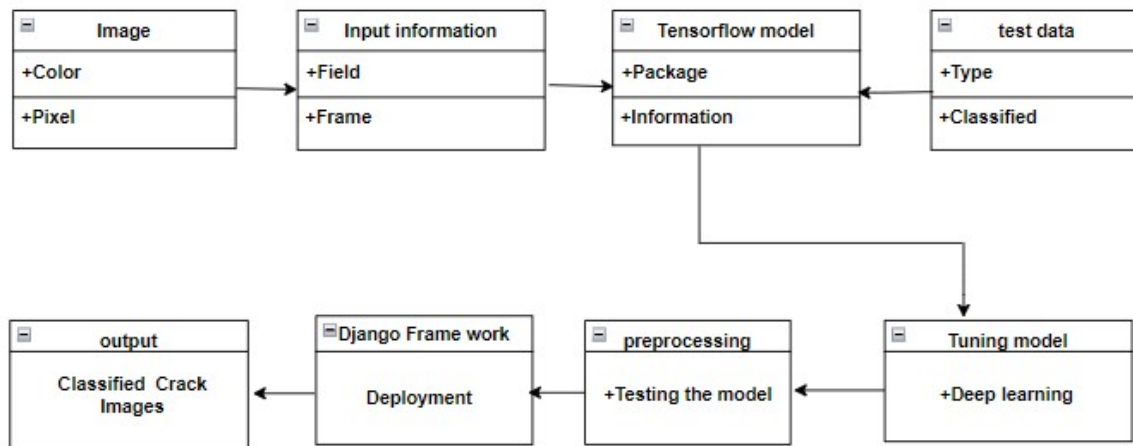
#### **5.1 System Architecture**



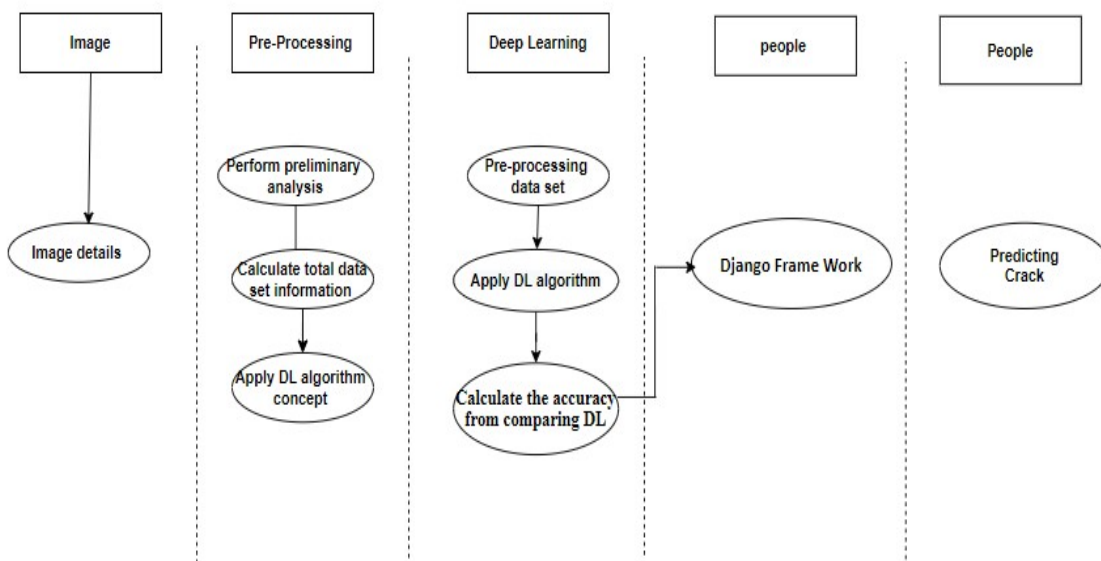
## 5.2 Use Case Diagram



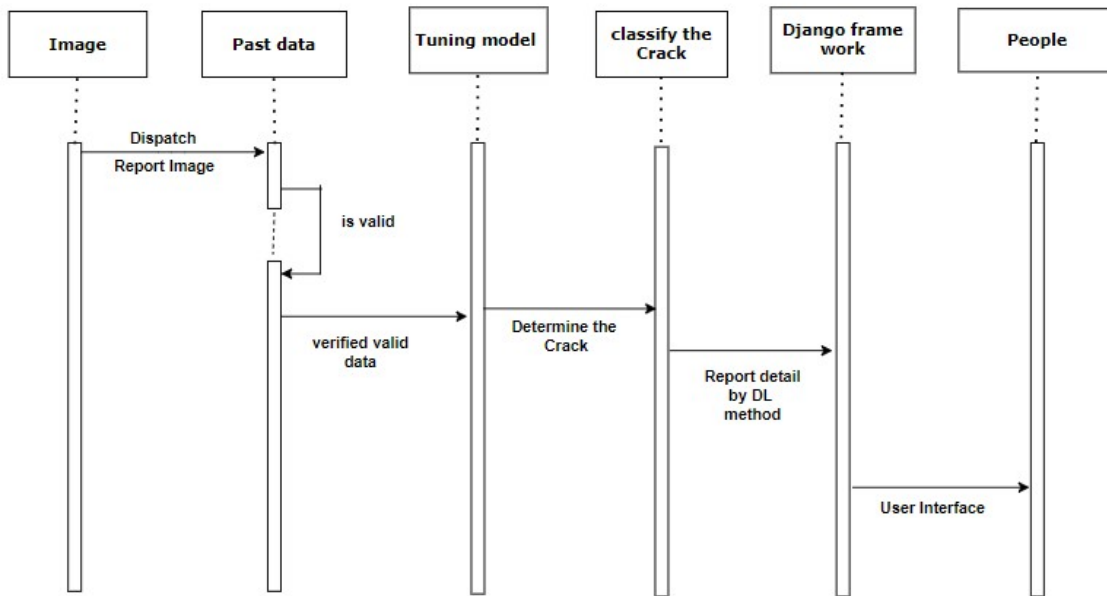
### 5.3 Class Diagram



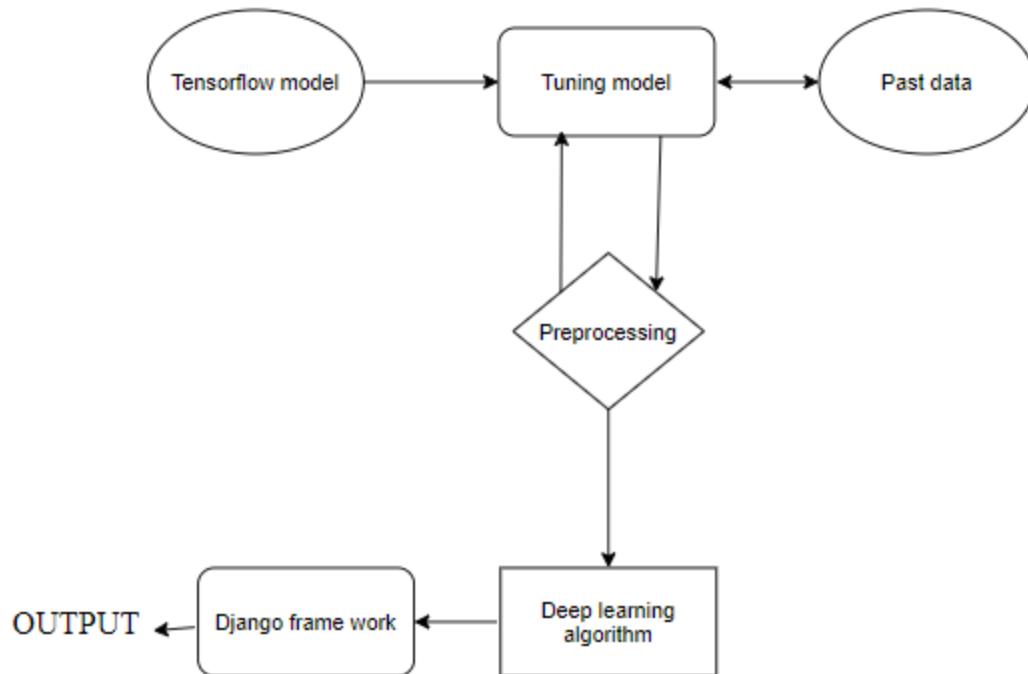
### 5.4 Activity Diagram



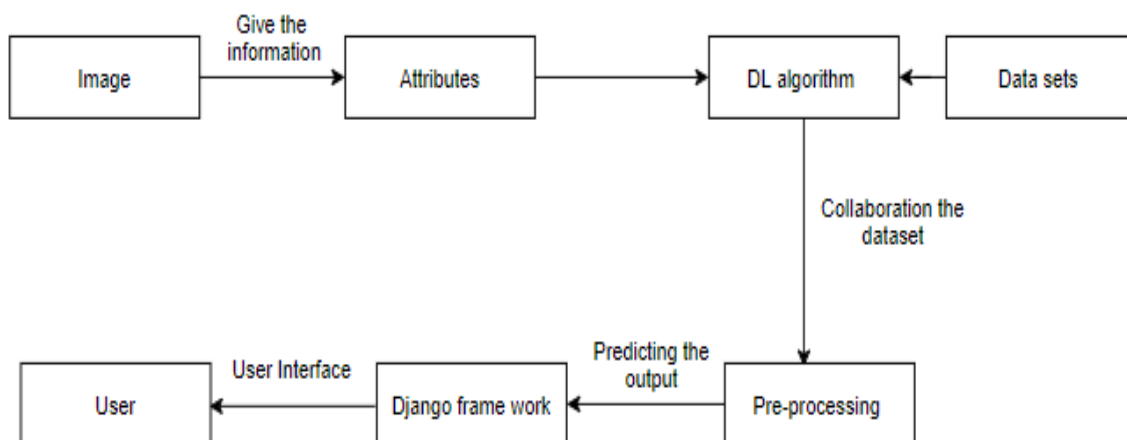
## 5.5 Sequence Diagram



## 5.6 ER Diagram



## 5.7 Collaboration Diagram



## 5.8 Environmental Requirements

### 1. Software Requirements

Operating System : Windows / Linux  
Simulation Tool : Anaconda with Jupyter Notebook

## **2. Hardware requirements**

Processor : Pentium IV/III  
Hard disk : minimum 80 GB  
RAM : minimum 2 GB

## **CHAPTER 6**

### **MODULE DESCRIPTION:**

#### **6.1 Import Image From The Dataset**

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of CNN.

#### **Defective**

Trained data for Defective train track:

```
===== Images in: dataset/Train/Defective
images_count: 149
min_width: 156
max_width: 6000
min_height: 148
max_height: 6000
```



## Non Defective

Trained data for non defective train track:

```
===== Images in: dataset/Train/Non defective
images_count: 150
min_width: 4000
max_width: 4000
min_height: 3000
max_height: 3000
```



## 6.2 To Train The Module By the Given Image Dataset

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.

Model: "sequential\_2"

| Layer (type)                  | Output Shape        | Param # |
|-------------------------------|---------------------|---------|
| conv2d_10 (Conv2D)            | (None, 60, 60, 96)  | 34944   |
| activation_14 (Activation)    | (None, 60, 60, 96)  | 0       |
| max_pooling2d_6 (MaxPooling2) | (None, 30, 30, 96)  | 0       |
| conv2d_11 (Conv2D)            | (None, 20, 20, 256) | 2973952 |
| activation_15 (Activation)    | (None, 20, 20, 256) | 0       |
| max_pooling2d_7 (MaxPooling2) | (None, 10, 10, 256) | 0       |
| conv2d_12 (Conv2D)            | (None, 8, 8, 384)   | 885120  |
| activation_16 (Activation)    | (None, 8, 8, 384)   | 0       |
| conv2d_13 (Conv2D)            | (None, 6, 6, 384)   | 1327488 |
| activation_17 (Activation)    | (None, 6, 6, 384)   | 0       |
| conv2d_14 (Conv2D)            | (None, 4, 4, 256)   | 884992  |
| activation_18 (Activation)    | (None, 4, 4, 256)   | 0       |
| max_pooling2d_8 (MaxPooling2) | (None, 2, 2, 256)   | 0       |
| flatten_2 (Flatten)           | (None, 1024)        | 0       |

## 6.3 Working Process of Layers in CNN Model

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training,

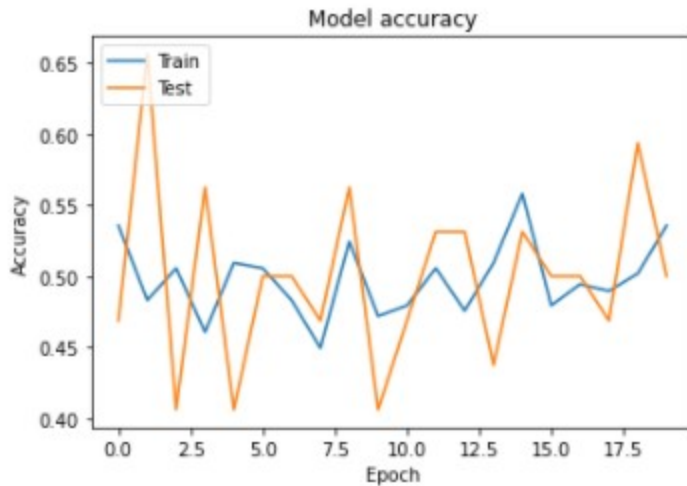
ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

### **Input Layer:**

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension  $28 \times 28 \times 3 = 784$ , it need to convert it into  $784 \times 1$  before feeding into input.

### **Convo Layer:**

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.



## Pooling Layer

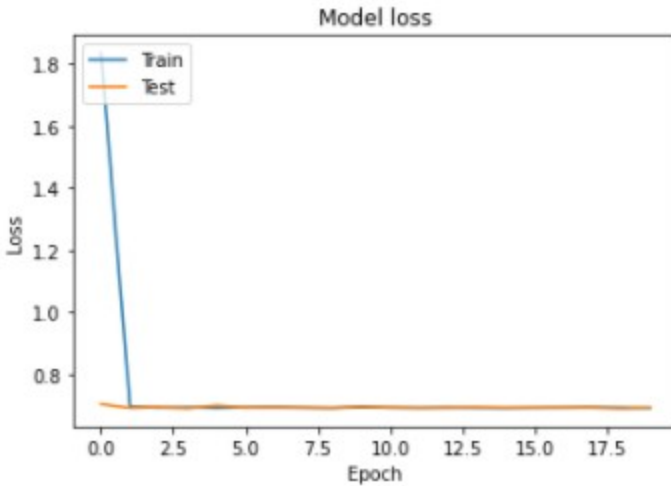
Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.

## Fully Connected Layer (FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

## Soft max / Logistic Layer

Soft max or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and soft max is for multi-classification.

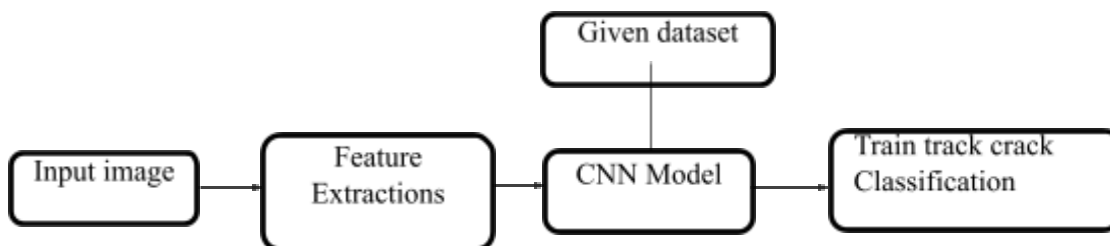


## Output Layer

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

## 6.4 Classification Identification

We give input image using keras preprocessing package. That input Image converted into array value using pillow and image to array function package. We have already classified crack of train crack in our dataset. It classifies what are the train crack. Then we have to predict the train track crack using predict function.



The Train track crack recognition method is based on a two-channel architecture that is able to recognize crack of the train track. The track crack parts are cropped and

extracted and then used as the input into the inception layer of the CNN. The Training phase involves the feature extraction and classification using convolution neural network.

## **Libraries Required**

- tensorflow: Just to use the tensor board to compare the loss and adam curve our result data or obtained log.
- keras: To pre-process the image dataset.
- matplotlib: To display the result of our predictive outcome.
- os: To access the file system to read the image from the train and test directory from our machines.

## **TensorFlow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow –

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.

- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

## **Keras**

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

### **Features**

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features –

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

## **Matplotlib**

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPython or Tkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also. Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

## OS:

The OS module in Python comes with various functions that enables developers to interact with the Operating system that they are currently working on. In this article we'll be learning mainly to create and delete a directory/folder, rename a directory and even basics of file handling.

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

## **CHAPTER 7**

### **Conclusion**

#### **7.1 Conclusion**

It focused how image from given dataset (trained dataset) and past data set used to predict the pattern of Train track crack using CNN model. This brings some of the following insights about track crack prediction. The major benefit of the CNN classification framework is the ability to classify images automatically. In this study, we have discussed the overview of methodologies for detecting the abnormalities in track images which includes collection of train track image data set, preprocessing techniques, feature extraction techniques and classification schemes.

#### **7.2 Future Work**

Railway department wants to automate the detecting of track crack from early time (real time).

To optimize the work to implement in Artificial Intelligence environment.

## **CHAPTER 8**

### **Reference**

#### **8. Reference**

- [1] L. Zhu, F. R. Yu, B. Ning, and T. Tang, "Communication-based train control (CBTC) systems with cooperative relaying: Design and performance analysis," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2162–2172, Jun. 2014.
- [2] L. Zou, T. Wen, Z. Wang, L. Chen, and C. Roberts, "State estimation for communication-based train control systems with CSMA protocol," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 843–854, Mar. 2019.
- [3] Z. Wang, X. Wu, G. Yu, and M. Li, "Efficient rail area detection using convolutional neural network," *IEEE Access*, vol. 6, pp. 77656–77664, 2018.

- [4] Z. Yang et al., "Train intelligent detection system based on convolutional neural network," in Proc. Int. Conf. Appl. Hum. Factors Ergonom. Cham, Switzerland: Springer, 2019, pp. 157–165.
- [5] Y. Wang, J. Cong, P. Wang, X. Liu, and H. Tang, "A data-fusion approach for speed estimation and location calibration of a metro train based on low-cost sensors in smartphones," IEEE Sensors J., vol. 19, no. 22, pp. 10744–10752, Nov. 2019.
- [6] A. Papp, C. Wiesmeyr, M. Litzenberger, H. Garn, and W. Kropatsch, "A real-time algorithm for train position monitoring using optical timedomain reflectometry," in Proc. IEEE Int. Conf. Intell. Rail Transp. (ICIRT), Birmingham, U.K., Aug. 2016, pp. 89–93.
- [7] P. Ernest, R. Mazl, and L. Pfeucil, "Train locator using inertial sensors and odometer," in Proc. IEEE Intell. Vehicles Symp., Jun. 2004, pp. 860–865.
- [8] M. Lauer and D. Stein, "A train localization algorithm for train protection systems of the future," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 970–979, Apr. 2014.
- [9] J. Otegui, A. Bahillo, I. Lopetegi, and L. E. Diez, "A survey of train positioning solutions," IEEE Sensors J., vol. 17, no. 20, pp. 6788–6797, Aug. 2017.
- [10] G. M. Shafiullah, A. Gyasi-Agyei, and P. Wolfs, "Survey of wireless communications applications in the railway industry," in Proc. 2nd Int. Conf. Wireless Broadband Ultra Wideband Commun., 2007, p. 65.

## APPENDICES

### A.SOURCE CODE

**To build a model for training and  
testing:**

#### **Module 1:**

```
import os
```

```
import numpy as np # linear algebra
```

```
import matplotlib.pyplot as plt
```

```
# Dl framwork - tensorflow, keras a backend
```

```
import tensorflow as tf
```

In [ ]:

```

import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU,
Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense

```

In [ ]:

```

import warnings
warnings.filterwarnings('ignore')

```

In [ ]:

```

dir_name_train_Defective = 'dataset/Train/Defective'

```

```

dir_name_train_non_defective = 'dataset/Train/Non defective'

```

In [ ]:

```

def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

```

```
plt.figure(figsize=(35, 10))
for idx, img_path in enumerate(item_files):
    plt.subplot(2, n, idx+1)
    img = plt.imread(img_path)
    plt.imshow(img, cmap='gray')
    plt.axis('off')

plt.tight_layout()
```

In [ ]:

```
def Images_details_Print_data(data, path):
    print("==== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.*", recursive=True)]
    data = {}
    data['images_count'] = len(files)
    data['min_width'] = 10**100 # No image will be bigger than that
    data['max_width'] = 0
    data['min_height'] = 10**100 # No image will be bigger than that
    data['max_height'] = 0

    for f in files:
        im = Image.open(f)
        width, height = im.size
        data['min_width'] = min(width, data['min_width'])
        data['max_width'] = max(width, data['max_height'])
        data['min_height'] = min(height, data['min_height'])
```

```
data['max_height'] = max(height, data['max_height'])
```

```
Images_details_Print_data(data, path)
```

In [ ]:

```
print("")
print("Trained data for Defective train track:")
print("")
Images_details(dir_name_train_Defective)
print("")
plot_images(dir_name_train_Defective, 10)
```

In [ ]:

```
print("")
print("Trained data for non defective train track:")
print("")
Images_details(dir_name_train_non_defective)
print("")
plot_images(dir_name_train_non_defective, 10)
```

In [ ]:

```
Classifier=Sequential()
Classifier.add(Convolution2D(32,(3,3),input_shape=(250,250,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(38, activation='relu'))
```

In [ ]:

```
Classifier.add(Dense(2, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [ ]:

```
training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(250,250),batch_size
=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('dataset/Validation',target_size=(250,250),batch_size
```

```
=32,class_mode='categorical')
```

In [ ]:

```
img_dims = 150
```

```
epochs = 20
```

```
batch_size = 32
```

In [ ]:

```
##### Fitting the model
```

```
history = Classifier.fit_generator(  
    training_set, steps_per_epoch=training_set.samples // batch_size,  
    epochs=epochs,  
    validation_data=test_set, validation_steps=test_set.samples // batch_size)
```

In [ ]:

```
def graph():
```

```
    #Plot training & validation accuracy values
```

```
    plt.plot(history.history['accuracy'])
```

```
    plt.plot(history.history['val_accuracy'])
```

```
    plt.title('Model accuracy')
```

```
    plt.ylabel('Accuracy')
```

```
    plt.xlabel('Epoch')
```

```
    plt.legend(['Train', 'Test'], loc='upper left')
```

```
    plt.show()
```

```
    # Plot training & validation loss values
```

```
    plt.plot(history.history['loss'])
```

```
    plt.plot(history.history['val_loss'])
```

```
    plt.title('Model loss')
```

```
    plt.ylabel('Loss')
```

```
    plt.xlabel('Epoch')
```

```
    plt.legend(['Train', 'Test'], loc='upper left')
```

```
    plt.show()
```

In [ ]:

```
graph()
```

## Module 2:

```
# Dl framework - tensorflow, keras a backend
```

```
import tensorflow as tf
```

```
In [ ]:
```

```
import tensorflow.keras.backend as K
```

```
In [ ]:
```

```
from tensorflow.keras.models import Model
```

```
In [ ]:
```

```
from tensorflow.keras.models import Sequential
```

```
In [ ]:
```

```
from tensorflow.keras.layers import Input
```

```
In [ ]:
```

```
from tensorflow.keras.layers import Dense
```

```
In [ ]:
```

```
from tensorflow.keras.layers import Flatten
```

```
In [ ]:
```

```
from tensorflow.keras.layers import Conv2D
```

```
In [ ]:
```

```
from tensorflow.keras.layers import Dropout
```

```
In [ ]:
```

```
from tensorflow.keras.layers import LeakyReLU
```

```
In [ ]:
```

```
from tensorflow.keras.layers import Activation
```

```
In [ ]:
```

```
from tensorflow.keras.optimizers import Adam
```

In [ ]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:

```
from tensorflow.keras.callbacks import ModelCheckpoint
```

In [ ]:

```
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

In [ ]:

```
from tensorflow.keras.callbacks import EarlyStopping
```

In [ ]:

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

In [ ]:

```
model = Sequential()
```

```
# 1st Convolutional Layer
```

```
model.add(Conv2D(filters=96, input_shape=(250,250,3), kernel_size=(11,11), strides=(4,4),  
padding='valid'))
```

```
model.add(Activation('relu'))
```

```
# Max Pooling
```

```
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
```

```
# 2nd Convolutional Layer
```

```
model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1), padding='valid'))
```

```
model.add(Activation('relu'))
```

```
# Max Pooling
```

```
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
```

```
# 3rd Convolutional Layer
```

```
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
```

```
model.add(Activation('relu'))
```

```
# 4th Convolutional Layer
```

```
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
```

```

model.add(Activation('relu'))
# 5th Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# Passing it to a Fully Connected layer
model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(4096, input_shape=(250*250*3,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))
# 2nd Fully Connected Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# 3rd Fully Connected Layer
model.add(Dense(1000))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Output Layer
model.add(Dense(2))
model.add(Activation('softmax'))
model.summary()

# Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

In [ ]:

```
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal  
_flip=True)
```

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [ ]:

```
training_set=train_datagen.flow_from_directory('dataset/Train', target_size=(250,250), batch_size  
=32, class_mode='categorical')
```

In [ ]:

```
img_dims = 150
```

```
epochs = 20
```

```
batch_size = 32
```

In [ ]:

```
##### Fitting the model
```

```
history = model.fit(  
    training_set, steps_per_epoch=trainingset.samples // batch_size,  
    epochs=epochs,  
    validation_data=test_set, validationsteps=test_set.samples // batch_size)
```

In [ ]:

```
import matplotlib.pyplot as plt
```

In [ ]:

```
def graph():
```

```
    #Plot training & validation accuracy values
```

```
    plt.plot(history.history['accuracy'])
```

```
    plt.plot(history.history['val_accuracy'])
```

```
    plt.title('Model accuracy')
```

```
    plt.ylabel('Accuracy')
```

```
    plt.xlabel('Epoch')
```

```
    plt.legend(['Train', 'Test'], loc='upper left')
```

```
    plt.show()
```

```
    # Plot training & validation loss values
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

In [ ]:

```
graph()
```

In [ ]:

```
print("[INFO] Calculating model accuracy")
scores = model.evaluate(test_set)
print(f'Test Accuracy: {scores[1]*100}')
```

### Module 3:

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
```

In [ ]:

```
from tensorflow.keras.models import Sequential
```

In [ ]:

```
from tensorflow.keras.layers import Convolution2D
```

In [ ]:

```
from tensorflow.keras.layers import Flatten
```

In [ ]:

```
from tensorflow.keras.layers import Dense
```

In [ ]:

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

In [ ]:

```
Classifier=Sequential()
```

In [ ]:

```
Classifier.add(Convolution2D(32,3,3,input_shape=(250,250,3),activation='relu'))
```

```
Classifier.add(MaxPooling2D(pool_size=(2,2)))
```

```
Classifier.add(Convolution2D(128,3,3,activation='relu'))
```

```
Classifier.add(MaxPooling2D(pool_size=(2,2)))
```

```
Classifier.add(Flatten())
```

```
Classifier.add(Dense(256, activation='relu'))
```

```
Classifier.add(Dense(2, activation='softmax'))
```

```
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
Classifier.summary()
```

In [ ]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:

```
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal  
_flip=True)
```

In [ ]:

```
training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(250,250),batch_size  
=32,class_mode='categorical')
```

In [ ]:

```
test_set=test_datagen.flow_from_directory('dataset/Validation',target_size=(250,250),batch_size  
=32,class_mode='categorical')
```

In [ ]:

```
from IPython.display import display
```

In [ ]:

```
img_dims = 150
```

```
epochs = 50
```

```
batch_size = 32
```

|                                                                                                                                                                                       |         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| Classifier.fit_generator( training_set, steps_per_epoch=trainingset.samples // batchsize,<br>epochs=epochs,<br>validation_data=test_set,validationsteps=testset.samples // batchsize) | In [ ]: |
| <b>import</b> h5py                                                                                                                                                                    | In [ ]: |
| Classifier.save('Traintrack.h5')                                                                                                                                                      | In [ ]: |
| <b>from</b> keras.models <b>import</b> load_model                                                                                                                                     | In [ ]: |
| model=load_model('Traintrack.h5')                                                                                                                                                     | In [ ]: |
| <b>import</b> numpy <b>as</b> np                                                                                                                                                      | In [ ]: |
| <b>from</b> tensorflow.keras.preprocessing <b>import</b> image<br>test_image=image.load_img('test1.jpg',target_size=(250,250))                                                        | In [ ]: |
| <b>import</b> matplotlib.pyplot <b>as</b> plt<br>img = plt.imshow(test_image)                                                                                                         | In [ ]: |
| test_image=image.img_to_array(test_image)                                                                                                                                             | In [ ]: |
| test_image=np.expand_dims(test_image,axis=0)                                                                                                                                          | In [ ]: |
| result=model.predict(test_image)                                                                                                                                                      | In [ ]: |
| prediction = result[0]                                                                                                                                                                | In [ ]: |

|                                                                                                                                                                                 |         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| <code>classes=training_set.class_indices</code>                                                                                                                                 | In [ ]: |
| <code>classes</code>                                                                                                                                                            | In [ ]: |
| <code>prediction=list(prediction)</code>                                                                                                                                        | In [ ]: |
| <code>prediction</code>                                                                                                                                                         | In [ ]: |
| <code>classes=['Defective', 'Non defective']</code>                                                                                                                             | In [ ]: |
| <code>output=zip(classes,prediction)</code>                                                                                                                                     | In [ ]: |
| <code>output=dict(output)</code>                                                                                                                                                | In [ ]: |
| <code>output</code>                                                                                                                                                             | In [ ]: |
| <code>if output['Defective']==1.0 :</code><br><code>    print("Defective")</code><br><code>elif output['Non defective']==1.0:</code><br><code>    print("Non defective")</code> | In [ ]: |

## **PyCharm:**

### **Views.py**

```
from django.shortcuts import render
```

```
from django.http import HttpResponseRedirect
from django.urls import reverse_lazy
from django.views.generic import TemplateView
from employee.forms import EmployeeForm
```

```
from django.views.generic import DetailView
from employee.models import Employee
```

```
class EmployeeImage(TemplateView):
```

```
    form = EmployeeForm
```

```
    template_name = 'emp_image.html'
```

```
    def post(self, request, *args, **kwargs):
```

```
        form = EmployeeForm(request.POST, request.FILES)
```

```
        if form.is_valid():
```

```
            obj = form.save()
```

```
            return HttpResponseRedirect(reverse_lazy('emp_image_display', kwargs={'pk': obj.id}))
```

```
        context = self.get_context_data(form=form)
```

```
        return self.render_to_response(context)
```

```
    def get(self, request, *args, **kwargs):
```

```
        return self.post(request, *args, **kwargs)
```

```
class EmpImage(DetailView):
```

```
    model = Employee
```

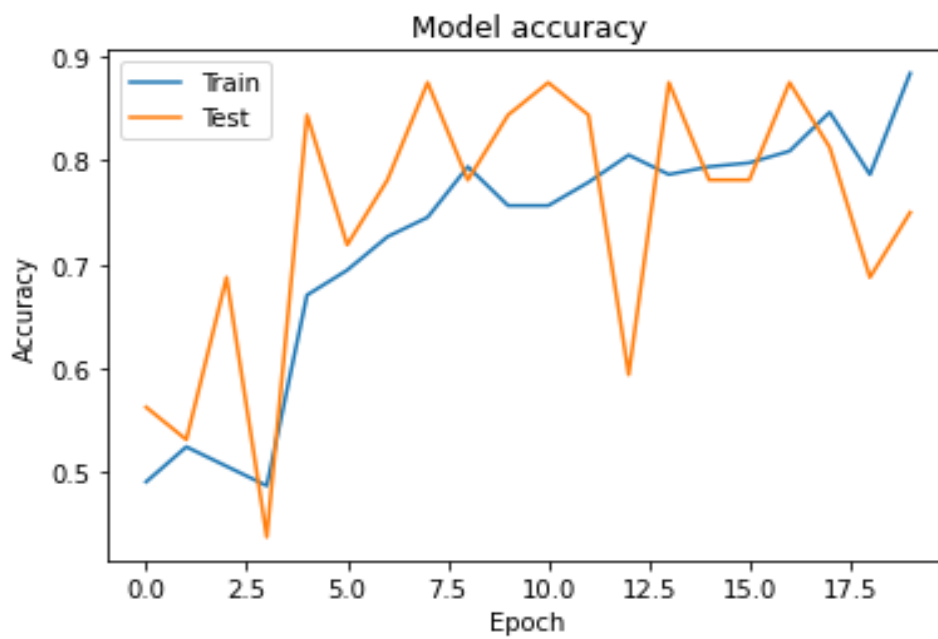
```
    template_name = 'emp_image_display.html'
```

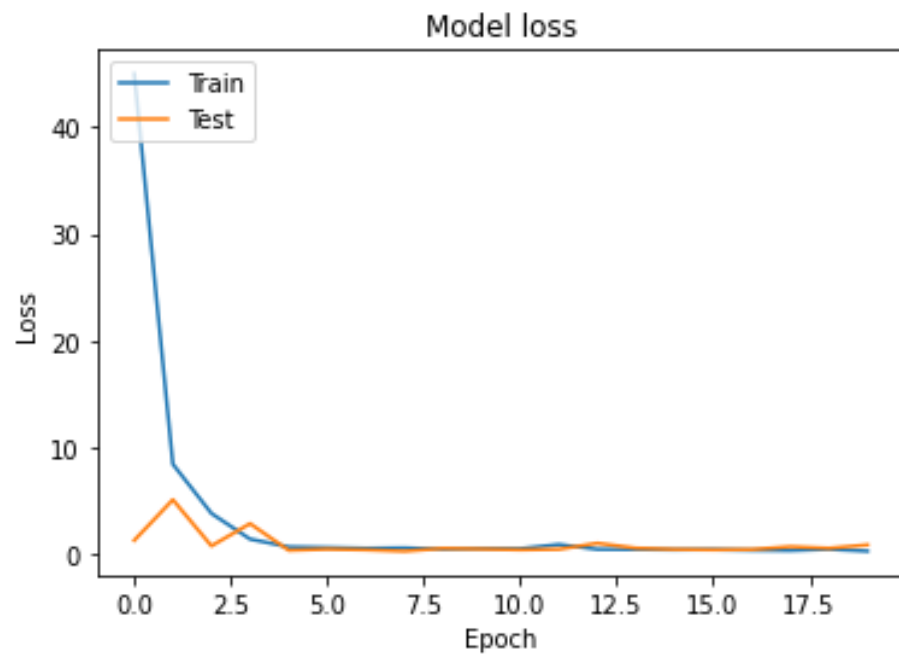
```
context_object_name = 'emp'
```

```
def traincrack(request):  
    result1 = Employee.objects.latest('id')  
    import numpy as np  
    import tensorflow as tf  
    from tensorflow import keras  
    import h5py  
    test_image = image.img_to_array(test_image)  
    test_image = np.expand_dims(test_image, axis=0)  
    result = models.predict(test_image)  
    prediction = result[0]  
    prediction = list(prediction)  
    classes=['Defective', 'Non defective']  
    output = zip(classes, prediction)  
    output = dict(output)  
    if output['Defective'] == 1.0:  
        a = "DEFECTIVE"  
    elif output['Non defective'] == 1.0  
        a = "NON DEFECTIVE"
```

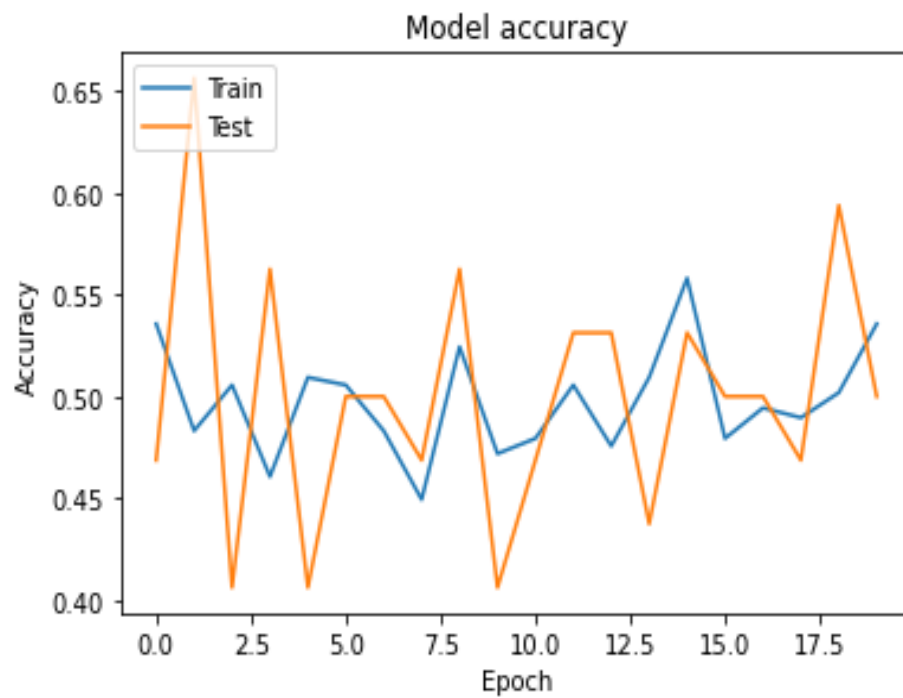
## B.OUTPUT SCREEN SHOTS

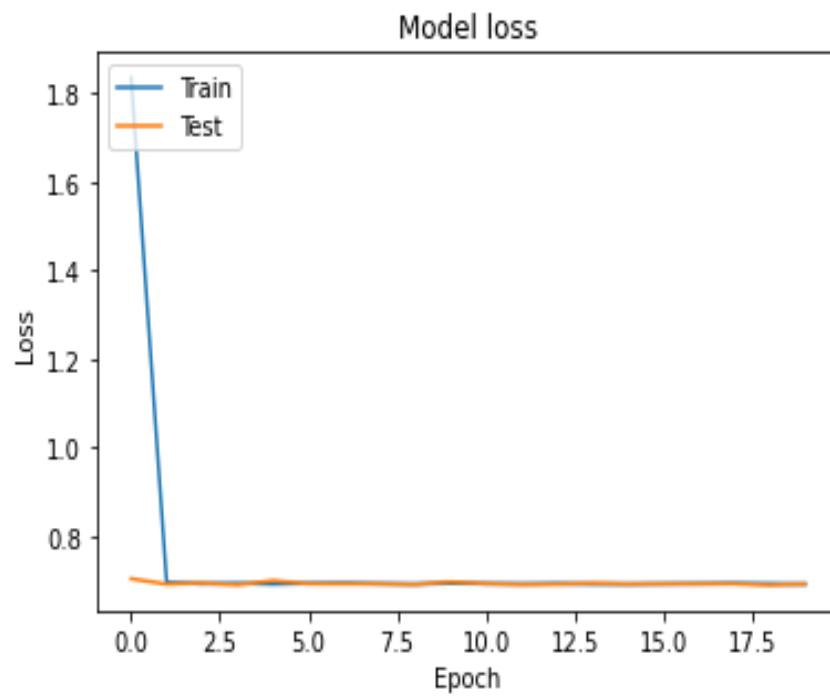
### Manual Network Architecture





## Alex Network Architecture





## Django Deployment



## TRAIN TRACK CRACK PREDICTION USING ARTIFICIAL NEURAL NETWORK

DEFECTIVE

[Go Back!!!](#)