

ONLINE RAIL SURFACE CRACK DETECTION

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering by

CHITTURI AKHILA

(37110158)

&

CHEEMALA ASTHA DIAMOND

(37110132)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119**

APRIL - 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work **Chitturi Akhila** (37110158) & **Cheemala Astha Diamond** (37110132) who carried out the project entitled "**Online rail surface crack detection**" under our supervision from November, 2020 to April 2021.

Internal Guide

Dr. A. Mary Posonia, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Dr.L.LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **Chitturi Akhila and Cheemala Astha Diamond** hereby declare that the Project Report entitled “ **Online rail surface crack detection**” is done by us under the guidance of Associate Professor **Dr. A. Mary Posonia, M.E., Ph.D.**, of the Department of CSE at Sathyabama Institute of Science and Technology 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 THECANDIDATES

1.

2.

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the 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, Dept. 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. A. Mary Posonia, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved the 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 who were helpful in many ways for the completion of the project.

ABSTRACT

In this system, we use CNN to detect the faults in the railway tracks by images. And acquire their characteristics to extract the default of the railway track. This model will help to reduce the manual working for inspection. While performing the task it lowers the time utilization and cost and increases the safety of passengers. To train and validate the CNN algorithm we use the groundtruth database on images of masks. To This end, This CNN maintains the both space and time coherence of the defects in the tracks, and the final output throw of the false prediction effectively.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	V
	LIST OF FIGURES	IX
	LIST OF TABLES	X
1.	INTRODUCTION	11
	OUTLINE OF THE PROJECT	11
	LITERATURE REVIEW	11
	SYSTEM STUDY	12
	OBJECTIVE	13
2.	AIM AND SCOPE OF PRESENT INVESTIGATION	14
	PROPOSED SYSTEM	14
	EXISTING SYSTEMS	18
3.	SOFTWARE DEVELOPMENT METHODOLOGY	19
	CONVOLUTIONAL NEURAL NETWORKS	19
	CONVOLUTIONAL LAYERS	19
	POOLING LAYERS	20
	FULLY CONNECTED LAYERS	20
	INCEPTION V3	20
	ALGORITHMS USED	21
	DEEP LEARNING	21

	NUMPY	21
	PANDAS	23
	MATPLOTLIB	23
	SYSTEM REQUIREMENTS	24
	SOFTWARE	
	REQUIREMENTS	24
	HARDWARE	
	REQUIREMENTS	24
	REQUIREMENT ANALYSIS	25
	FUNCTIONAL REQUIREMENTS	25
	NON-FUNCTIONAL	
	REQUIREMENTS	26
4.	RESULTS AND DISCUSSIONS	27
	4.1 DEPENDENCIES	27
	4.2 TESTING	27
	4.3 TEST STRATEGY AND	27
	APPROACH	
	INTEGRATION TESTING	28
	ACCEPTANCE TESTING	28
	APLHA TESTING	28
	4.4 TRAIN AND VALIDATION	28

5.	CONCLUSION	31
	REFERENCES	31
	APPENDIX	
	A. SOURCE CODE	33
	B. SCREEN SHOTS	45
	C.PUBLISHED PAPER	47

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
2.1	SYSTEM ARCHITECTURE	15
3.2	INCEPTION V3	21
4.4	EPOHS GRAPH	29

LIST OF TABLES

TABLE No.	TABLE NAME	PAGE No.
4.4	PREDICTION TABLE	25

CHAPTER 1

INTRODUCTION

OUTLINE OF THE PROJECT

The primary aim of this proposed system and CNN based image processing based method for detecting faults on railway track surfaces is proposed. The rail surface is detected by taking the rail surface images from the railway. Otsu segmentation method is used in rail surface detection processes. The signal is obtained by calculating the variance of the obtained rail surface image. By taking the signal obtained, it is determined whether there is a fault on the rail surface. The type of fault is determined by extracting features from variance signals belonging to faulty images.

LITERATURE REVIEW

Convolution neural networks for steel have proposed the progress in this field of neural networks was maintained by many achievements. enabling the architectures of deep learning by increasing the power of computing and providing higher and more levels of representation. Data augmentation is a powerful tool for not allowing the overfitting in this proposed method. Error rates can be reduced by committee methods by several networks combination, especially the uncorrelated in the individual predictions. unsupervised methods for representations and learning features became very popular and solved problems with supervised training. The convolutional neural network (CNN) is mainly designed for recognizing. CNN is performed outstandingly including both MNIST and SVHN datasets. Where it was

usually trained in the Neocognitron.

Broken Rail Detection System using RF Technology is proposed. Mainly this system is used for the detection of the broken rail and provides the information exactly about the tracks. To prove this model practically over the wooden plank the metallic track is simulated and these will be divided into fewer sections and these will be numbered and joined with metal plates. If any of these joints are removed and isolated that information will quickly be passed to the remote unit. Firstly the alarm is raised by the receiver for the staff alert after that the tracking number which was broken will be displayed by LCD with the microcontroller. The receiver is located near the station to alert the staff of the station and pass the information to the trains which are traveling in that route. This entire process is maintained by railway signaling.

A review on computer vision-based defect detection and condition assessment of concrete and asphalt civil are proposed. This model presents a synthesis of the art of computer vision that mainly explains the categories of the methods in computer vision methods, they are used to automate the damage and process detection. The computer vision pipeline starts processing from a lower to a higher level. These methods can be considered completely separately. Extracted features were studied to support the process of classification based on pattern methods. afterward, it is classified how the methodologies are used, validated, tested to find the unique defect and observing the damage in patterns by remote and morphology in the images of concrete surfaces, underground concrete pipes, precast concrete tunnels, and asphalt pavements.

SYSTEM STUDY

In many countries the majority of people are depending on the railways for the transportation. Railways were highly encouraged for rapid development. Although the frequent improvement of railway technology decreases the occurrence of the internal

defects, if a crack or any defects in the railway tracks are not detected may lead to heavy loss in the human life. That So far, the defects are detected by the three methods namely manual inspection, physical detection and computer vision methods In this paper, we propose first the histogram equalization is used to reduce the effect of uneven illumination on the detection of cracks. Secondly, the improved Canny edge detection algorithm is used to draw a frequent crack profile with adaptive dual thresholds. Then, an algorithm based on the contour of the fracture shape is used to determine the fracture characteristics and achieve accurate crack location. Finally, a criterion for determining the level of cracking faults based on the area of cracks is proposed. The algorithm can effectively improve the accuracy of railway track crack detection under complex environment. And the rail images are divided into two categories - defective or not.

OBJECTIVE

The proposed model is CNN based on the image processing which makes the whole process efficient. It is going to be very important and profitable for the whole Railways especially. This system will provide the railway images are defective or not more accurate information like time & date of the incident, place of the incident etc. which will be very helpful for the Railway management system to and thus it will take less time to complete the process.

CHAPTER 2

AIM AND SCOPE

PROPOSED SYSTEM

In the proposed system, the image processing and deep neural network-based CNN model is used for detecting the faulty images in the railway crack detection. This inspection process is performed by collecting the images of railway tracks by Otsu segmentation model mainly used in railway surface detection. When variance is calculated by rail surface image the signal is obtained it will determine whether there is any fault in the particular image of the railway track surface. The type of the fault is determined by considering the features from signals of variance for faulty images.

Initially data is loaded, the process will start from image processing. The image will be read and size is adjusted, all the denoise will be eliminated. Then segmentation followed by morphology will take place. Where segmentation is a process of dividing image visuals into smaller segments and morphology is a process of collecting image structures. With all these information image will be filtered with necessary features and segmented crack will be identified.

MERITS OF PROPOSED SYSTEM

It works well with high dimensionality.

reduces the consumption of hardware resources.

Improve prediction accuracy effectively.

ARCHITECTURE:

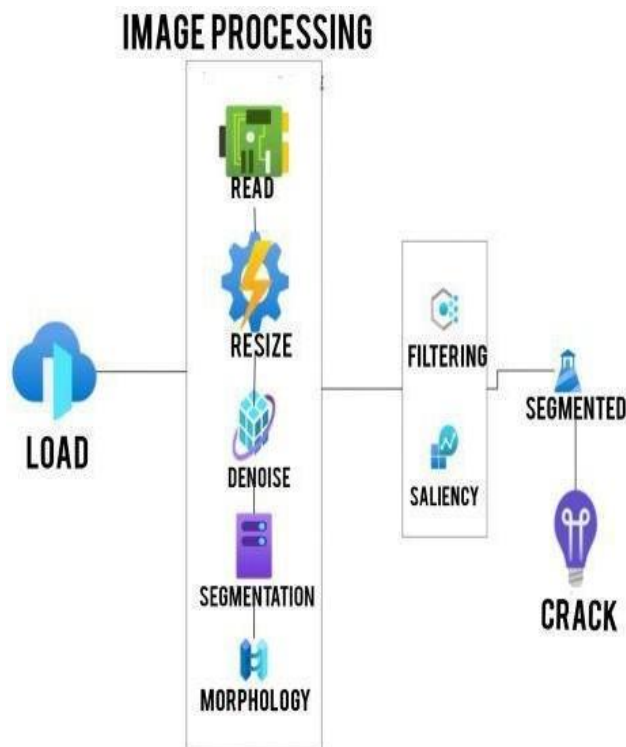


Fig 2.1 : System Architecture.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

MODULES

Image Preprocessing

The railway surface is predicted by collecting various railway surface images from rail tracks. So here, for the railway surface detection the OTSU segmentation method is used. The signal acquired by calculating variance of obtained rail surface image. By considering the obtained signal it will determine whether there is any fault on railway surface. This faults being extracted the features from faulty images variance signals.

The rail surface is detected by collecting the rail surface images from the railway. Otsu segmentation method is used in the rail surface detection process. The signal is acquired by calculating the variance of the obtained rail surface image. By taking the signal obtained, it is determined whether there is a fault on the rail surface. The type of fault is determined by extracting features from variance signals belonging to faulty images

Border detection is calculated by the grayscale of depth image sum for every row and the disparity ratio of two rows. When the ratio is greater than a threshold, that row will be separated border table and floor areas

The transformation in the brightness modifies the brightness in the pixel also, the transformation completely depends on the properties of the pixel. Generally, brightness transformation includes brightness corrections and greyscale transformations. Brightness correction takes pixel brightness into account by its location or position of the image and the original brightness and modified. Where grayscale transformations change brightness despite to position of the image.

Consider this equation

$$G(x,y) = T\{ f(x,y) \}$$

In this equation,

$F(x,y)$ = input image on which transformation function has to be applied.

$G(x,y)$ = the output image or processed image.

T is the transformation function.

This relation between input image and the processed output image can also be represented as.

$$s = T(r)$$

where r is actually the pixel value or gray level intensity of $f(x,y)$ at any point. And s is the pixel value or gray level intensity of $g(x,y)$ at any point.

The basic gray level transformation has been discussed in our tutorial of basic gray level transformations.

Feature Extraction

Feature extraction provides two functions, they are data compression, invariance. Feature vector contains selected features, they are extracted from a sample set of measurements. The feature step is to eliminate redundancy in representation. A cautiously selected vector should contain most of the preferred information and it should be smaller in dimensions compared to the master copy of the signal vector. This improves the accuracy of a classification and also reduces the computational effort.

Prediction

The principal of this methodology is to detect the cracks on the railway track with pictures. This model is used for image processing. Mainly segmentation is carry out in view of the density pixels of the image. Considering K number of class and $(k-1)$

Threshold pixel value and the image pixel K is divided

After all the segmented cracks in the data set are trained and validated.

The algorithm trained an entire dataset is passed forward and backward until we get correct predictions

The class is the prediction model (CNN) When class is less than 0.5 it returns having fault in the particular railway track. When the class is more than 0.5 then there is no fault in the railway track

EXISTING SYSTEM

The existing system proposes a background-oriented defect inspector (BODI) to improve defect detection by considering specified characteristics of the track during inspection. Reformulating the inspection task in this manner offers a new way to model rail surface images. More specifically, BODI features a random sampling stage to obtain a compact background representation without any prior information. A sufficient number of random selections generates adequate and diverse background statistics, and defect-determination and a fusion of procedures then determine whether current pixel belongs to the background. We thus build a background model for each longitudinal line to distinguish defects from the normal background model, and propose a new detection method (BODI). In existing work, we process pixels in a block independently and set w to 1. It means that each transversal line is a block. This is reasonable given that the entire longitudinal line along a rail's surface image is consistent. Actually, a promising defect inspector (i.e., the definition of distance measurement shown in our reformulation) is assumed to be able to highlight the defects, so a key process is to choose a proper background representation. A simple way is to generate a global inspector always using all pixels detected as background pixels. As a result, the defect pixels contaminate the background representation. The aim of the defect inspector is to

select adequate and representative background pixels from the rail surface image and define a proper measurement for the background while limiting the effect of defects.

DISADVANTAGES OF EXISTING SYSTEM

Time Consuming
Labour Intensive
Low Accuracy
Error Prone

CHAPTER 3

ALGORITHMS AND MODEL DESIGN

CONVOLUTIONAL NEURAL NETWORKS

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that does multiplication or other dotproduct, and its activation function is commonly ReLU. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

CONVOLUTIONAL LAYERS

. A convolutional layer within a CNN generally has the following attributes:

- Convolutional filters/kernels defined by a width and height (hyper-parameters).

- The number of input channels and output channels (hyper-parameters). One layer's input channels must equal the number of output channels (also called depth) of its input.
- Additional hyperparameters of the convolution operation, such as: padding, stride, and dilation.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its field. Although fully connected can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for *each* neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper.^[15] For example, regardless of image size, using a 5 x 5 tiling region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during back propagation in traditional neural networks. Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

POOLING LAYERS

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. Global pooling acts on all the neurons of the feature map. There are two common types of pooling in popular use: max and average. *Max pooling* uses the

maximum value of each local cluster of neurons in the feature map, while *average pooling* takes the average value.

FULLY CONNECTED LAYERS

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi layer neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

INCEPTION V3

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

A high-level diagram of the model is shown below:

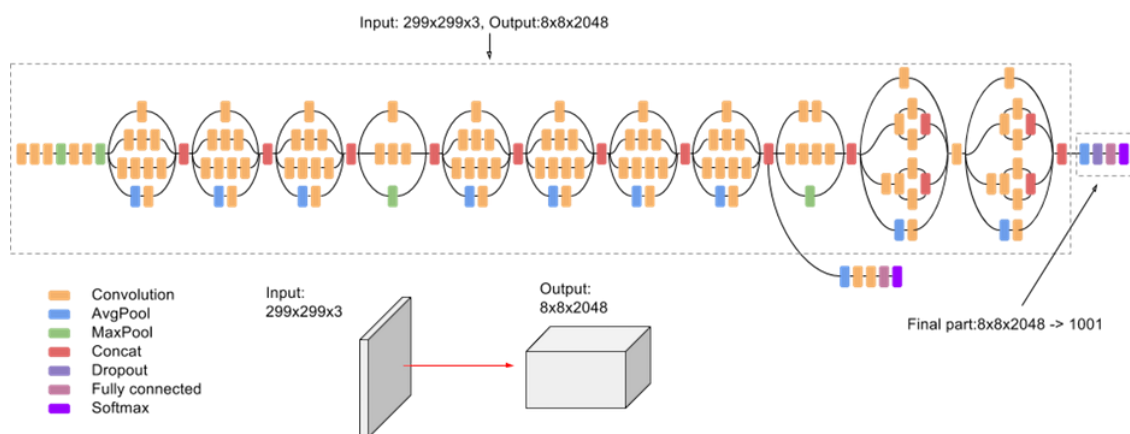


Fig. 3.2 Inception v3 Architecture

The inception v3 has more information about the Inception architecture.

ALGORITHM USED

DEEP LEARNING

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

NUMPY

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists,

such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

PANDAS

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn. Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily. Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' Data Frames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

MATPLOTLIB

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which is designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to

say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc, with just a few lines of code."

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

Backend Technologies	: Python, Numpy, Sci-learn, Eclipse ID
Frontend Technologies	: Web Technologies, Bootstrap
Operating System	:Windows

HARDWARE REQUIREMENTS

Hardware	:Minimum Requirement
Disk Space	:32 GB or more,10 GB or more for Foundation
Processor	:1.4 GHz 64 bit
Memory	:512 MB
Display	:(800 × 600) Capable video adapter.

REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that it has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

CHAPTER 4

RESULTS AND DISCUSSION

DEPENDENCIES

Pandas - to work with solid data structures

Matplotlib - to visualize data using 2D plots.

Scikit-learn - to create deep learning models

TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

Test Objectives

- ◆ All field entries must work properly.
- ◆ Pages must be activated from the identified link.
- ◆ The entry screen, messages and responses must not be delayed.
- ◆ Features to be tested
- ◆ Verify that the entries are of the correct format
- ◆ No duplicate entries should be allowed
- ◆ All links should take the user to the correct

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

ALPHA TESTING

In software development, alpha test will be a test among the teams to confirm that your product works. Originally, the term alpha test meant the first phase of testing in a software development process. The first phase includes unit testing, component testing, and system testing. It also enables us to test the product on the lowest common denominator machines to make sure download times are acceptable and preloads work.

TRAIN AND VALIDATION

For classification tasks, a supervised learning algorithm looks at the training dataset to determine, or learn, the optimal combinations of variables that will generate a good predictive model. The goal is to produce a trained (fitted) model that generalizes well to new, unknown data. The fitted model is evaluated using “new” examples from the held-out datasets (validation and test datasets) to estimate the model’s accuracy.

in classifying new data. To reduce the risk of issues such as overfitting, the examples in the validation and test datasets should not be used to train the model.

Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

In order to avoid overfitting, when any classification parameter needs to be adjusted, it is necessary to have a validation dataset in addition to the training and test datasets. For example, if the most suitable classifier for the problem is sought, the training dataset is used to train the different candidate classifiers, the validation dataset is used to compare their performances and decide which one to take and, finally, the test dataset is used to obtain the performance characteristics such as accuracy, sensitivity, specificity, and so on. The validation dataset functions as a hybrid: it is training data used for testing, but neither as part of the low-level training nor as part of the final testing.



Fig 4.4 epohs graph

Prediction	Defective	Non-Defective	ePohs(1)		ePohs(10)		ePohs(30 or more)	
			D	ND	D	ND	D	ND
Train	150	150	120	110	130	125	150	150
Test	150	150	130	122	143	132	150	150
Validation	150	150	110	111	146	149	150	150

4.4 Prediction table

The dataset in train, test and validation the images will get correct predictions after training the algorithm several times.

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

In this research we adapt the state of the art detection model Mask R-CNN to automate crack detection on concrete surfaces. We train this model using our own ground truth dataset which was built on a subset of images from a recent benchmark annotated image set aimed for training and validation of artificial intelligence based crack detection algorithms for concrete. In order to adapt our framework to scenarios with more varying surfaces and lighting conditions, in the future we aim to retrain the network on a larger and more varied dataset.

REFERENCES

- [1]H. Oliveira and P. L. Correia, "Automatic road crack detection and characterization," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 1, pp. 155–168, March 2013, doi: <https://doi.org/10.1109/TITS.2012.2208630>.
- [2]C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth, "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure," vol. 29, no. 2, 2015, pp. 196 – 210, infrastructure Computer Vision. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1474034615000208>
- [3] K. Kang et al., "T-CNN: Tubelets with convolutional neural networks for object

detection from videos,” IEEE Trans. Circuits Syst. Video Technol., to be published, 2016.

[4]O. Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015.

[5]D. Soukup and R. Huber-Mork, “Convolutional neural networks for steel “ surface defect detection from photometric stereo images,” in Proc. Int. Symp. Visual Comput., Dec. 2014, pp. 668–677.

[6]L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, “Road crack detection using deep convolutional neural network,” in Proc. IEEE Int. Conf. Image Process., Sep. 2016, pp. 3708–3712.

[7]Y.-J. Cha, W. Choi, and O. Buy“ uk” ozt “ urk, “Deep learning-based crack “ damage detection using convolutional neural networks,” Comput.-Aided Civil Infrastructure Eng., vol. 32, no. 5, pp. 361–378, Mar. 2017

[8]S. J. Schmugge et al., “Detection of cracks in nuclear power plant using spatialtemporal grouping of local patches,” in Proc. IEEE Winter Conf. Appl. Comput. Vis., Mar. 2016, pp. 1–7.

[9]Sireesha R, Ajay Kumar B, Mallikarjunaiah G and Bharath Kumar B, “Broken Rail Detection System using RF Technology”, in “SSRG International Journal of Electronics and Communication Engineering”, volume 2 issue 4- April 2015

Prof. P.Navaraja, “CRACK DETECTION SYSTEM FOR RAILWAY TRACK BY USING ULTRASONIC AND PIR SENSOR”, in “INTERNATIONAL JOURNAL OF ADVANCED INFORMATION AND COMMUNICATION TECHNOLOGY”, Volume -1, Issue-1, May 2014

APPENDIX

A. *SOURCE CODE*

```
import os

%matplotlib inline

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import os

from tensorflow.keras import layers

from tensorflow.keras import Model

from tensorflow.keras.applications.inception_v3 import InceptionV3
```

```
import tensorflow as tf
```

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

```
from tensorflow.keras.optimizers import RMSprop
```

```
import cv2
```

```
import numpy as np
```

```
import pandas as pd
```

```
import random
```

```
from tensorflow.keras.preprocessing.image import img_to_array, load_img
```

```
train_dir = os.path.join('dataset/Train')
```

```
validation_dir = os.path.join('dataset/Validation')
```

```
train_defective_dir = os.path.join('dataset/Train/Defective')
```

```
train_nondefective_dir = os.path.join('dataset/Train/Non defective')
```

```
validation_defective_dir = os.path.join('dataset/Validation/Defective')
```

```
validation_nondefective_dir = os.path.join('dataset/Validation/Non defective')
```

```
train_defective_fnames = os.listdir(train_defective_dir )
```

```
train_nondefective_fnames = os.listdir( train_nondefective_dir)
```

```
print(train_defective_fnames[:20])
```

```
print(train_nondefective_fnames[:20])
```

```
print('total training defective images :', len(os.listdir(train_defective_dir)))
```

```
print('total          training          non-defective          images          :',  
len(os.listdir(train_nondefective_dir)))
```

```
print('total validation defective images :', len(os.listdir( validation_defective_dir
)))
```

```
print('total validation non-defective images :', len(os.listdir(
validation_nondefective_dir)))
```

```
nrows = 4
```

```
ncols = 4
```

```
pic_index = 0
```

```
pic_index += 8
```

```
next_defective_pix = [os.path.join(train_defective_dir, fname)
```

```
    for fname in train_defective_fnames[pic_index-8:pic_index]]
```

```
next_nondefective_pix = [os.path.join(train_nondefective_dir, fname)
```

```
    for fname in train_nondefective_fnames[pic_index-8:pic_index]]
```

```
fig = plt.gcf()
```

```
fig.set_size_inches(ncols * 4, nrows * 4)
```

```
for i, img_path in enumerate(next_defective_pix+next_nondefective_pix):
```

```
sp = plt.subplot(nrows, ncols, i + 1)
```

```
sp.axis('Off')
```

```
img = mpimg.imread(img_path)
```

```
plt.imshow(img)
```

```
plt.show()
```

```
local_weights_file =  
'inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'
```

```
pre_trained_model = InceptionV3(input_shape = (300,300, 3),  
                                include_top = False,  
                                weights = None)
```

```
pre_trained_model.load_weights(local_weights_file)
```

```
for layer in pre_trained_model.layers:
```

```
    layer.trainable = Falsepre_trained_model = InceptionV3(input_shape =
```

```

(300,300, 3),

        include_top = False,

        weights = None)

pre_trained_model.summary()


last_layer = pre_trained_model.get_layer('mixed7')


print('last layer output shape: ', last_layer.output_shape)


last_output = last_layer.output


train_datagen = ImageDataGenerator(rescale = 1./255.,

        rotation_range = 40,

        width_shift_range = 0.2,

        height_shift_range = 0.2,

        shear_range = 0.2,

        zoom_range = 0.2,

        horizontal_flip = True)

```

```

test_datagen = ImageDataGenerator(rescale = 1./255.,

    rotation_range = 40,

    width_shift_range = 0.2,

    height_shift_range = 0.2,

    shear_range = 0.2,

    zoom_range = 0.2,

    horizontal_flip = True)

train_generator = train_datagen.flow_from_directory(train_dir,

    batch_size=20,

    class_mode='binary',

    target_size=(300,300))

validation_generator = test_datagen.flow_from_directory(validation_dir,

    batch_size=20,

    class_mode = 'binary',

    target_size = (300,300))

```

```
x = layers.Flatten()(last_output)
```

```
x = layers.Dense(128, activation='relu')(x)
```

```
x = layers.Dropout(0.2)(x)
```

```
x = layers.Dense(1, activation='sigmoid')(x)
```

```
model = Model( pre_trained_model.input, x)
```

```
model.compile(optimizer = RMSprop(lr=0.0001),
```

```
              loss = 'binary_crossentropy',
```

```
              metrics = ['accuracy'])
```

```
model.summary()
```

```
history = model.fit(train_generator,
```

```
validation_data=validation_generator,  
  
steps_per_epoch=10,  
  
epochs=3,  
  
validation_steps=5,  
  
verbose=2)
```

```
acc    = history.history[ 'accuracy' ]  
  
val_acc = history.history[ 'val_accuracy' ]  
  
loss    = history.history[ 'loss' ]  
  
val_loss = history.history[ 'val_loss' ]
```

```
epochs = range(len(acc))
```

```
plt.plot ( epochs, acc )  
  
plt.plot ( epochs, val_acc )  
  
plt.title ('Training and validation accuracy')  
  
plt.figure()
```

```
plt.plot ( epochs, loss )
```

```
plt.plot ( epochs, val_loss )
```

```
plt.title ('Training and validation loss' )
```

```
img = cv2.imread('dataset/Test/Defective/IMG_20201114_102222.jpg')
```

```
plt.imshow(img)
```

```
img = cv2.resize(img,(300,300))
```

```
img = np.reshape(img,[1,300,300,3])
```

```
classes = model.predict(img)
```

```
if classes<0.5:
```

```
    print("This Railway track has no fault")
```

```
else:
```

```
    print("This Railway track has fault")
```

```
classes = model.predict(img)
```

```
if classes>0.5:
```

```
    print("This Railway track has no fault")
```

```
else:
```

```
    print("This Railway track has fault")
```

```
successive_outputs = [layer.output for layer in model.layers[1:]]
```

```
visualization_model = tf.keras.models.Model(inputs = model.input, outputs =  
successive_outputs)
```

```
defective_img_files = [os.path.join(train_defective_dir, f) for f in  
train_defective_filenames]
```

```
nondefective_img_files = [os.path.join(train_nondefective_dir, f) for f in  
train_nondefective_filenames]
```

```
img_path = random.choice(defective_img_files + nondefective_img_files)
```

```
img = load_img(img_path, target_size=(300,300))
```

```
x = img_to_array(img)
```

```
x = x.reshape((1,) + x.shape)
```

```
x /= 255
```

```
successive_feature_maps = visualization_model.predict(x)
```

```
layer_names = [layer.name for layer in model.layers[1:]]
```

```
for layer_name, feature_map in zip(layer_names, successive_feature_maps):
```

```
    if len(feature_map.shape) == 4:
```

```
        n_features = feature_map.shape[-1]
```

```
        size = feature_map.shape[1]
```

```
        display_grid = np.zeros((size, size * n_features))
```

```

for i in range(n_features):

    x = feature_map[0, :, :, i]

    x -= x.mean()

    x /= x.std()

    x *= 64

    x += 128

    x = np.clip(x, 0, 255).astype('uint8')

    display_grid[:, i * size : (i + 1) * size] = x

scale = 20. / n_features

plt.figure(figsize=(scale * n_features, scale))

plt.title(layer_name)

plt.grid(False)

plt.imshow(display_grid, aspect='auto', cmap='summer')

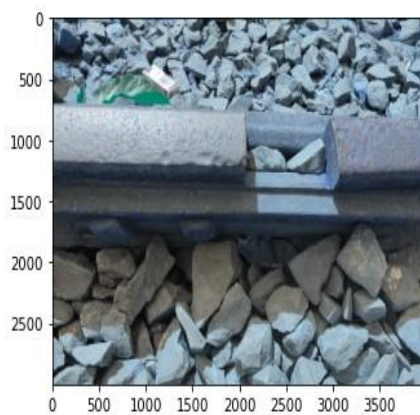
```

B.SCREENSHOTS

Defected cracks are calculated by using the ratio of the correct cracks found to the number of cracks detected. Recollect how many actual cracks are identified. This will be calculated by the ratio of correct cracks to the actual number of cracks. As we know that we had used the CNN model to create this concept of prediction we should divide the images into train and validation dataset at which the images with faults and no faults. By training the model several times

we get correct predictions whether the images of tracks containing any faults or not and these return the image with defective and non-defective.

```
In [46]: img = cv2.imread('dataset/Test/Defective/IMG_20201114_102203.jpg')
plt.imshow(img)
img = cv2.resize(img, (300, 300))
img = np.reshape(img, [1, 300, 300, 3])
```

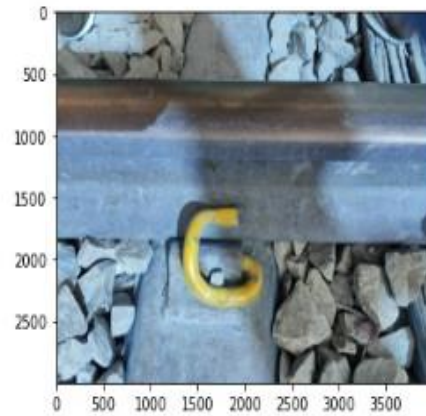


```
In [47]: classes = model.predict(img)
```

```
In [48]: if classes<0.5:
print("This Railway track has no fault")
else:
print("This Railway track has fault")
```

This Railway track has fault

```
In [49]: M img = cv2.imread('dataset/Test/Non defective/IMG_20201114_101633.jpg')
plt.imshow(img)
img = cv2.resize(img, (300, 300))
img = np.reshape(img, [1, 300, 300, 3])
```



```
In [50]: M classes = model.predict(img)
```

```
In [51]: M if classes>0.5:
print("This Railway track has no fault")
else:
print("This Railway track has fault")
```

This Railway track has no fault

Several images are collected in the data set and by using the inception v3 and CNN model it will return if the images are having defects or not.

C.PUBLISHED PAPER