Fault Detection in Railway Tracks using Artificial Neural Networks

Aalhad Welankiwar
Department of Computer Sciecne and Engineering
Prof Ram Meghe College of Engineering and Management, Badnera, Amravati
Badnera, Amravati, India
aalhad13@gmail.com

Shubham Sherekar
Department of Computer Sciecne and Engineering
Prof Ram Meghe College of Engineering and Management, Badnera, Amravati
Badnera, Amravati, India
shubhamsherekar9@gmail.com

Amol P Bhagat
Department of Computer Sciecne and Engineering
Prof Ram Meghe College of Engineering and Management, Badnera, Amravati
Badnera, Amravati, India
amol.bhagat84@gmail.com

Priti A Khodke
Department of Computer Sciecne and Engineering
Prof Ram Meghe College of Engineering and Management, Badnera, Amravati
Badnera, Amravati, India
priti.khodke@gmail.com

***Abstract*—Faults in railway tracks cause majority of the train accidents. We present an automated fault detection technique using Machine Learning via Artificial Neural Network. The Camera will capture images of railway track which will be provided as an input to the system consisting of neural network. The Artificial Neural Network(A.N.N) will be trained on the basis of Dataset (made from the scratch) and on the basis of this training A.N.N will then provide absolute output whether the track is faulty or not. And hence proper measures can be taken by the authorities.**

Keywords— Railway Tracks; Fault Detection; Artificial Neural Network; Image Processing

I. INTRODUCTION

We found that using Image processing and putting that as an input to the neural network is more efficient and productive than other methods. Using Long Short Term Memory (LSTM) Neural Network is a better option than Convolution Neural Network (CNN) [1]. While the LSTM network outperforms the convolutional network for the track circuit case, the convolutional networks are easier to train. Sensors are often located away from energy supplies and hence require either batteries or some form of energy supply to power them. If there are errors in transmission across WSN, then data may be missing. The claim in [2] contradicts itself as it needs to minimize the energy consumption but maximize communication efficiency. For this proper arrangement will be need to be made. The images may be blurred and/or the track may not be visible [3]. It is even possible that two cameras capture the same crack. We need to analyze the proper distance of the cameras from one another and the track so as to avoid the incorrect data and get proper output.

Ultrasonic sensors [4] measure the distance from track to sensor and if the distance is greater than the assigned value the Microcontroller detects a crack and also it tells the location by the formula D=S*Time. IR sensors [5] give the status to Arduino controller. If crack is found it immediately sends the location of crack to mobile phone via BT. Periodic images [6] are taken of the tracks and compared with the existing database of non-faulty track images on a continuous basis. If fault occurs in the track it can be detected and proper actions can be taken.

The LED will be attached to one side of the rails and the LDR to the opposite side. During normal operation, when there are no cracks, the LED light does not fall on the LDR and hence the LDR resistance is high. Subsequently, when the LED light falls on the LDR, the resistance of the LDR gets reduced and the amount of reduction will be approximately proportional to the intensity of the incident light. As a consequence, when light from the LED deviates from its path due to the presence of a crack or a break ,a sudden decrease in the resistance value of the LDR ensues. This change in resistance indicates the presence of a crack or some other similar structural defect in the rails.

Track condition and presence of cracks in tracks play an important factor in the derailment of the trains. Manual checking of the tracks takes up a huge amount of time and resources. Also it is open to human error. To address this issue of derailment and ensure proper investigation of the tracks, an automated system is needed. Here, the objective is to design a neural network capable of computing on the given inputs and providing absolute output of whether the track is damaged or not. The scope of the paper is to develop a robust neural network and train it using Backtracking Algorithm which will compute on the input images of tracks and give an output whether the image is cracked or not.

Considering the current situation of Indian Railways regarding recent accidents, we have formulated the following problem definition. Indian Railway is one of the biggest Railway Network across the globe! But recently it is going through a major derailment crisis. Keeping in mind that the railway tracks are the most crucial factor responsible for derailment, we have developed a scheme to detect faults in the railway tracks using ANN. The objective here is to detect faults in railway tracks such as broken/cracked rails, rusted fishplates and bolts, unusual track interactions and improper track curves.

II. Mechanism for Fault Detection

This paper proposes a proper method to detect cracks in railway tracks using Image Processing and Neural Networks. The input images of the tracks go through a series of Image Processing functions, discussed later, to the Neural Networks. The Network then computes whether the track is cracked or not. Backpropagation [9-12] is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network. It is commonly used to train deep neural networks, a term referring to neural networks with more than one hidden layer.

Backpropagation is a special case of an older and more general technique called automatic differentiation. In the context of learning, backpropagation is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function. This technique is also sometimes called backward propagation of errors, because the error is calculated at the output and distributed back through the network layers. The backpropagation algorithm has been repeatedly rediscovered and is equivalent to automatic differentiation in reverse accumulation model.

Backpropagation requires a known, desired output for each input value—it is therefore considered to be a supervised learning method (although it is used in some unsupervised networks such as auto-encoders). Backpropagation is also a generalization of the delta rule to multi-layered feedforward networks, made possible by using the chain rule to iteratively compute gradients for each layer. It is closely related to the Gauss–Newton algorithm, and is part of continuing research in neural backpropagation. Backpropagation can be used with any gradient-based optimizer, such as L-BFGS or truncated Newton.

To implement the proposed algorithm, explicit formulas are required for the gradient of the function

```
{\displaystyle  w\mapsto  E(f_{N}(w,x),y)}
{\displaystyle w\mapsto E(f_{N}(w,x),y)}
```

where the function is

```
{\displaystyle         E(y,y')=|y-y'|^{2}}
{\displaystyle E(y,y')=|y-y'|^{2}}.
```

The learning algorithm can be divided into two phases: propagation and weight update. Each propagation involves the steps: Propagation forward through the network to generate the output value(s) , Calculation of the cost (error term) Propagation of the output activations back through the network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons. For each weight, the steps must be followed: The weight's output delta and input activation are multiplied to find the gradient of the weight.

A ratio (percentage) of the weight's gradient is subtracted from the weight. This ratio (percentage) influences the speed and quality of learning; it is called the learning rate. The greater the ratio, the faster the neuron trains, but the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates whether the error varies directly with, or inversely to, the weight. Therefore, the weight must be updated in the opposite direction, "descending" the gradient. Learning is repeated (on new batches) until the network performs adequately.

The following is pseudocode for a stochastic gradient descent algorithm for training a three-layer network (only one hidden layer):

```
initialize network weights (often small random values)
do
   forEach training example named ex
    prediction = neural-net-output(network, ex)
                                        // forward pass
    actual = teacher-output(ex)
    compute error (prediction - actual) at the
                           output units
    compute {\displaystyle \Delta w_{h}} \Delta w_h
    for all weights from hidden layer to output
       layer // backward pass
    compute {\displaystyle \Delta w_{i}} \Delta w_i
    for all weights from input layer to hidden layer
    // backward pass continued
    update network weights // input layer not
    modified by error estimate
    until all examples classified correctly or
    another stopping criterion satisfied
return the network
```

The lines labeled "backward pass" can be implemented using the backpropagation algorithm, which calculates the gradient of the error of the network regarding the network's modifiable weights.

III. Proposed Approach and Synthetic Dataset

In photography, computing, and colorimetry, a grayscale or greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. Images of this sort, also known as black-and-white or monochrome, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only two colors, black and white (also called bilevel or binary images). Grayscale images have many shades of gray in between.

Grayscale images can be the result of measuring the intensity of light at each pixel according to a particular weighted combination of frequencies (or wavelengths), and in such cases they are monochromatic proper when only a single frequency (in practice, a narrow band of frequencies) is captured. The frequencies can in principle be from anywhere in the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.). The railway track images are converted to grayscale using following pseudo code.

```
ConversionFactor = 255 / (NumberOfShades - 1)
AverageValue = (Red + Green + Blue) / 3
Gray = Integer((AverageValue / ConversionFactor) + 0.5) * ConversionFactor
```

NumberOfShades is a value between 2 and 256. Technically, any grayscale algorithm could be used to calculate AverageValue; it simply provides an initial gray value estimate

-the "+ 0.5" addition is an optional parameter that imitates rounding the value of an integer conversion; YMMV depending on which programming language you use, as some round automatically.

A co-occurrence matrix or co-occurrence distribution is a matrix that is defined over an image to be the distribution of co-occurring pixel values (grayscale values, or colors) at a given offset. The texture filter functions provide a statistical view of texture based on the image histogram. These functions can provide useful information about the texture of an image but cannot provide information about shape, i.e., the spatial relationships of pixels in an image. Another statistical method that considers the spatial relationship of pixels is the gray-level co-occurrence matrix (GLCM), also known as the gray-level spatial dependence matrix.

As there was no dataset available for this particular work we have built our own dataset from the scratch with the help of *GLCM technique*. In matlab , graycomatrix(I) creates a gray-level co-occurrence matrix (GLCM) from image I. Another name for a gray-level co-occurrence matrix is a *gray-level spatial dependence matrix*. Also, the word co-occurrence is frequently used in the literature without a hyphen, cooccurrence. graycomatrix creates the GLCM by calculating how often a pixel with gray-level (grayscale intensity) value *i* occurs horizontally adjacent to a pixel with the value *j*. (You can specify other pixel spatial relationships using the 'Offsets' parameter.) Each element (*i*,*j*) in glcm specifies the number of times that the pixel with value *i* occurred horizontally adjacent to a pixel with value *j*.

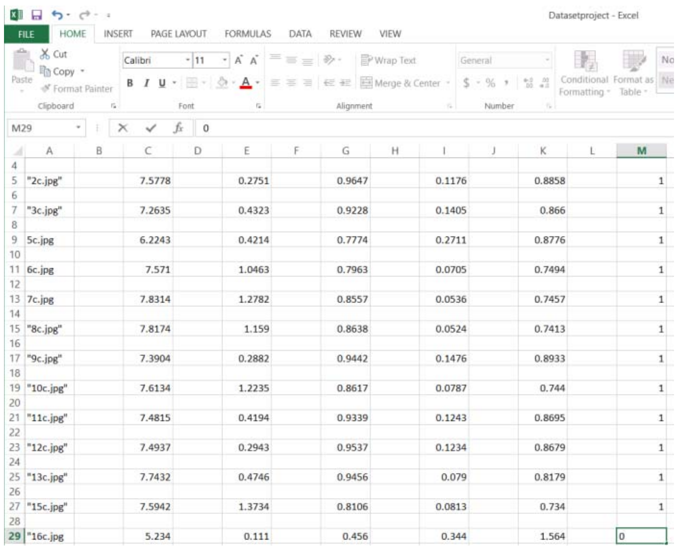
glcms = graycomatrix(I, Name, Value, ...) returns one or more gray-level co-occurrence matrices, depending on the values of the optional name/value pairs. Parameter names can be abbreviated, and case does not matter. [glcms, SI] = graycomatrix(**___**) returns the scaled image, SI, used to calculate the gray-level co-occurrence matrix. The values in SI are between 1 and NumLevels.

Using GLCM we created an array of 8*8 matrix then we used texture analysis function of MATLAB to find the entropy of gray scale image using :- " e = entropy(I)"

Where, I is input image which is one of the features which we extracted from the image. Similarly, we obtained properties of gray-level co-occurrence matrix from 'graycoprops' graycoprops(glcm,properties) calculates the statistics specified in properties from the gray-level co-occurrence matrix glcm. glcm is an *m*-by-*n*-by-*p* array of valid gray-level co-occurrence matrices. If glcm is an array of GLCMs, stats is an array of statistics for each glcm.

graycoprops normalizes the gray-level co-occurrence matrix (GLCM) so that the sum of its elements is equal to 1. Each element (*r*,*c*) in the normalized GLCM is the joint probability occurrence of pixel pairs with a defined spatial relationship having gray level values *r* and *c* in the image. graycoprops uses the normalized GLCM to calculate properties. Which is used using command:- "s = graycoprops(glcm)" The features which we obtained from above were:- Entropy, Contrast, Correlation, Energy, and Homogeneity.

Output was set to 1 for image having cracked rails and 0 for no cracks. This data was stored in an Excel Sheet for multiple images and this Excel sheet shown in figure was used as an dataset for training Neural Network. Figure 1 is the database sheet containing the extracted features of the images for the dataset.



M29 fx 0

	A	B	C	D	E	F	G	H	I	J	K	L	M
4													
5	"2c.jpg"		7.5778		0.2751		0.9647		0.1176		0.8858		1
6													
7	"3c.jpg"		7.2635		0.4323		0.9228		0.1405		0.866		1
8													
9	5c.jpg		6.2243		0.4214		0.7774		0.2711		0.8776		1
10													
11	6c.jpg		7.571		1.0463		0.7963		0.0705		0.7494		1
12													
13	7c.jpg		7.8314		1.2782		0.8557		0.0536		0.7457		1
14													
15	"8c.jpg"		7.8174		1.159		0.8638		0.0524		0.7413		1
16													
17	"9c.jpg"		7.3904		0.2882		0.9442		0.1476		0.8933		1
18													
19	"10c.jpg"		7.6134		1.2235		0.8617		0.0787		0.744		1
20													
21	"11c.jpg"		7.4815		0.4194		0.9339		0.1243		0.8695		1
22													
23	"12c.jpg"		7.4937		0.2943		0.9537		0.1234		0.8679		1
24													
25	"13c.jpg"		7.7432		0.4746		0.9456		0.079		0.8179		1
26													
27	"15c.jpg"		7.5942		1.3734		0.8106		0.0813		0.734		1
28													
29	"16c.jpg		5.234		0.111		0.456		0.344		1.564		0

Figure 1: Created synthetic dataset for fault detection in railway tracks.

IV. Experimental Results

The neural network structure shown in figure 2 will have 5 inputs which are: Entropy, Contrast, Correlation, Energy, and Homogeneity. Figure gives us an idea of how the features(Extracted from input images) are given as an input to the neural network and absolute output is provided. A hidden layer which will be trained on the basis of dataset created and absolute output where, 1 will represent crack in the railway track and 0 will represent no crack.

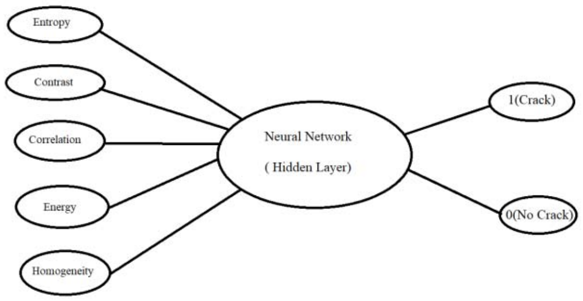


Figure 2: Proposed neural network based model for fault detection in railway tracks.

Figure 3 is the input image of a cracked track. This image is input to the Neural Network where the functions are performed on the image. This image is obtained from the cameras placed near the tracks. The image is converted to a Gray Scale Image. This is necessary for Feature Extraction since it cannot be performed on RGB image. It is done with the help of Image Processing Toolbox.

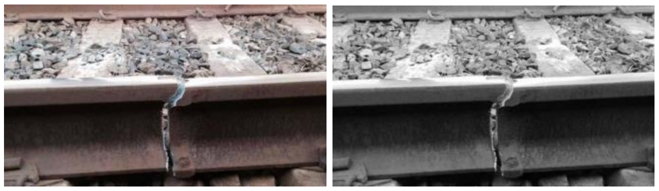


Figure 3: Input image and input image converted to grayscale.

Cropping function is used to crop the crack out of the track image. It is used to train the Neural Network. The features are extracted for matching and training the Neural Network. These specify the 100 strongest features of the image. Figure 4 shows the cropped image and marked features.

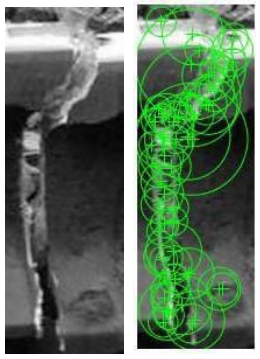


Figure 4: Cropped image and strongest features of crack image.

In figure 5 the features of the cropped image and input image are matched. Outlier and Inliers are matched with each other. The Outliers or the unrequired points are casted aside. Only the inliers are matched. Based on the matched points, the crack is detected in the input image if present shown in figure 6. A box encircles the crack. Neural Network will go a step further and directly give us an absolute output of whether the crack is present or not and thus classify the images in two different stacks.

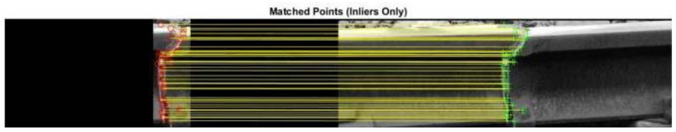


Figure 5: Feature matching of input image and stored image.

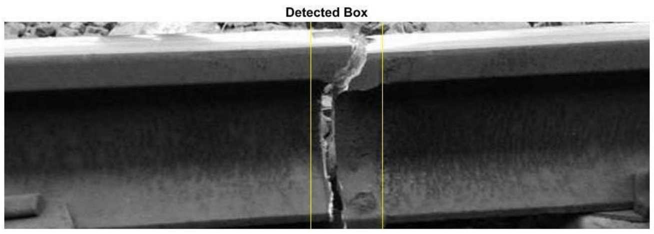


Figure 6: Crack detected in railway track.

In this system an image of railway track is given as an input to find whether there is fault or not. The features are extracted of that input image using same GLCM technique. These features are extracted from input image , will be given as input to the neural network i.e entropy , contrast , correlation , energy , homogeneity and with the help of hidden layer which is trained using dataset absolute output will be generated. 1 will represent crack and 0 will represent no crack. In this way the neural network is used to detect crack in railway track.

V. CONCLUSION

Currently, the derailment of the trains is a National issue that is gripping our nation. 90% of the derailments are caused due to the broken or cracked railway tracks. A proper method for detecting cracks needs to be devised so as to avoid the loss of lives and property caused due to the derailments of the trains.

The current crack detection schemes are either manual inspection or using LED/LDR. Manual inspection uses up a lot of time, resources and manual labor. It is also very prone to human error. There is no automation in this method. The LED/LDR method is also not very feasible since there can be lots of errors due to various things such as a stone or a leaf coming in between the sensors. It is very cumbersome since the sensors need to be placed after a very short regular interval. If a bot is devised and made to move along the tracks, then the trains running on the track are delayed until the inspection is done. Also continuous inspection is not possible since we cannot move the bot everyday on the tracks.

The main intention of this work was to create a more feasible and robust method for the detection of the cracks on the railway tracks. We decided to go with Image processing since cameras are present everywhere and a single camera can cover a lot of track. Also there is no manual labor and the trains can work without any disruption even during the inspection. The cameras can inspect the tracks for 24 hours everyday without disturbing anything since they don’t need to be placed on the tracks or such.

The proposed approach provides much faster and reliable inspection of the tracks. The input image from a camera of the track is given to the Neural Network which is trained using the dataset of cracked tracks images. The neural network then performs the operations on it and gives us an absolute output of whether the track is cracked or not. Two separate data structures are formed to directly classify the cracked and non cracked images.

The future work of this paper is to include various other parameters like Metal Quality, presence of Water or Snow above accepted levels, etc of the track and provide a more diverse output about the quality and usability of the track.

REFERENCES

[1] J. Chen, C. Roberts, and P. Weston, “Fault detection and diagnosis for railway track circuits using neuro-fuzzy systems,” Control Eng. Pract., vol. 16, no. 5, pp. 585–596, 2008.

[2] K. Verbert, B. De Schutter, and R. Babuška, “Exploiting spatial and temporal dependencies to enhance fault diagnosis: Application to railway track circuits,” in Proc. Eur. Control Conf., Linz, Austria, Jul. 2015, pp. 3047–3052.

[3] S. Ntalampiras, “Fault identification in distributed sensor networks based on universal probabilistic modeling,” IEEE Trans. Neural Netw. Learn. Syst., vol. 26, no. 9, pp. 1939–1949, Sep. 2015.

[4] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun. (2015). “Deep image: Scaling up image recognition.” [Online]. Available: http://arxiv.org/abs/1501.02876

[5] L. Oukhellou, A. Debiolles, T. Denoeux, and P. Aknin, “Fault diagnosis in railway track circuits using Dempster–Shafer classifier fusion,” Eng. Appl. Artif. Intell., vol. 23, no. 1, pp. 117–128, 2010.

[6] L. Oukhellou, A. Debiolles, T. Denoeux, and P. Aknin, “Fault diagnosis in railway track circuits using Dempster–Shafer classifier fusion,” Eng. Appl. Artif. Intell., vol. 23, no. 1, pp. 117–128, 2010.

[7] Z. L. Cherfi, L. Oukhellou, E. Côme, T. Denoeux, and P. Aknin, “Partially supervised independent factor analysis using soft labels elicited from multiple experts: Application to railway track circuit diagnosis,” Soft Comput., vol. 16, no. 5, pp. 741–754, 2012.

[8] M. A. Sandidzadeh and M. Dehghani, “Intelligent condition monitoring of railway signaling in train detection subsystems,” J. Intell. Fuzzy Syst., Appl. Eng. Technol., vol. 24, no. 4, pp. 859–869, 2013.

[9] S. Sun and H. Zhao, “Fault diagnosis in railway track circuits using support vector machines,” in Proc. 12th Int. Conf. Mach. Learn. Appl., vol. 2. Miami, FL, USA, Dec. 2013, pp. 345–350.

[10] Z. Lin-Hai, W. Jian-Ping, and R. Yi-Kui, “Fault diagnosis for track circuit using AOK-TFRs and AGA,” Control Eng. Pract., vol. 20, no. 12, pp. 1270–1280, 2012.

[11] S. Ntalampiras, “Fault identification in distributed sensor networks based on universal probabilistic modeling,” IEEE Trans. Neural Netw. Learn. Syst., vol. 26, no. 9, pp. 1939–1949, Sep. 2015.

[12] M. M. Gardner et al., “Equipment fault detection using spatial signatures,” IEEE Trans. Compon., Packag., Manuf. Technol. C, vol. 20, no. 4, pp. 295–304, Oct. 1997.