Railway Track Fault Detection

Mathemathical Modeling Practice

Tamás DEMUS
XP4B9D

Fall Semester 2022

Contents

1 Introduction

Analyzing images and processing the information stored within is a key field of machine learning. Classification of different images, identifying and localizing different objects are basic problems. Several real-life cases prove the usability of such approach such as traffic sign recognition, object detection or face recognition. The target of current research is to endeavour to create an algorithm that is able to classify images taken from parts of the rail track to classify whether the rail is defect or not.

In Section 2 the dataset is introduced without any detailed analysis to give a first glance where the study is started. Section 3 the main research questions stated. Following in Section 4 a detailed dexription of the applied methodology and toolkits introduced. Section 5 presents the results of the study divided according to the main steps of the algorithm. In Section 6 the results are reflected to the research questions, hopefully leading to positive outcomes. Finally Section 7 concludes in the overall achievements.

2 Dataset description

The dataset used for this study is taken from Kaggle webpage [1] and can be downloaded directly from `https://www.kaggle.com/datasets/salmaneunus/railway-track-fault-detection` [2]. The dataset is stored in different directories related to their purpose: Train, Validation, or Test dataset. Inside each directory the classes also splitted to separate directories: Defective or Non defective. The directory structure along with the number of images can be seen in Table 1. The images are taken from different perspectives, either from the top or from the side or from any other direction or angle. The photos show different track sections, that could be a close view on the single rail or a full picture taken from the entire rail section. The quality of the dataset is spreading between different formats, mostly

Folder	Number of images
./Train/Defective	150
./Train/Non defective	150
./Validation/Defective	31
./Validation/Non defective	31
./Test/Defective	11
./Test/Non defective	11

Table 1: Dataset directory structure

consisting of jpg files. The size of the images are different ranging from very low to very high resolutions. The dataset contains only color pictures. Some examples are shown in Figure 1 and Figure 2. The total size of the dataset is 2.14 GB.

(a) Non defective (b) Non defective (c) Non defective

Figure 1: Example images of non defective track

(a) Defective (b) Defective (c) Defective

Figure 2: Example images of defective track

3 Problem statement

The algorithm targets to identify from a picture whether it represents a defective or a non defective track section. For this purpose image manipulation technics together with machine learning algorithms, in more detail neural networks applied. The problem formulation defines the main questions of the research to be answered:

Q1 What kind of defects are represented in the images?

Q2 Can these defects detected by applying image manipulation and machine learning approach?

Q3 What accuracy rate can be achieved with the algorithm?

4 Methodology

The research can be divided to several major steps, these are described in the subsequent sections. The algorithm is developed in Python programming language in a form of a Jupyter notebook. In current

research the aim was to store the different stages of the processing in different file folders to allow single analytics of each processing step. In each folder the same original structure is retained. Therefore the following data folders are differentiated.

1. raw Contains the raw dataset directly copied from Kaggle.
2. data The images after data cleaning in standard format and naming.
3. augmented Additional images as result of data augmentation.
4. preprocessed The data after image processing, preprocessed for the machine learning algorithm. Includes both the original and augmented dataset.

4.1 Data cleaning

The images should be brought to the same quality in order to efficiently process them. This covers the identification of corrupted files, correct the wrong file formats and the resolution of all issues that prevents the processing of the data along the same pipeline. The result of this step should be a data structure that is easy to handle and process further. This includes the following information.

1. Type of image, whether it is intended for training, validation or testing.
2. Indicator of defective or non-defective class, both as integer and as string.
3. Path of the image directory, filename and their joints as full path.

4.2 Data exploration

The target of this step is to get a deep understanding of the pictures contentwise. The main properties, such as size, color composition, orientation, what is shown and similar need to be checked with respect to characteristic differences between the distinct classes. The size and the mean of the color components is extracted from the images and stored in the dataframe of the images. The distribution and correlation of the images color components considering the mean values analyzed. A random sample of the images is taken for visualization to study the different defects of the defective class. As a final step, the color components were investigated in detail considering the histogram of the component values taken for each pixels. This is investigated on the RGB color model and the images were converted to HSV model to get a more comprehensive view.

5 Results

5.1 Data cleaning

The raw data from the Kaggle webpage [2] is copied to the folder raw. This folder is not changed during the data processing, it is used as a starting point and to preserve the original data. During data cleaning three different file formats detected, namely webp, jpeg and jpg. The filenames show a much wider spread including random names, regular photo labeling (e.g.: DCIM) and others. As Tensorflow is not able to work with webp files and to maintain the same standard along the dataset, all files that are not in jpg format are converted to jpg. During this conversion all the files were renamed applying an integer counter. The modified dataset is then stored in the data folder in the same structure as in the original data.

Second step is to sum all the main information of each image in a single Python DataFrame to allow easy processing.

5.2 Data exploration

The distribution of the image sizes are shown in Figure 3. Please note that higher detail images can be found in the corresponding Jupyter notebook, the aim in this document is to explain the evaluation approach and summarize the results. Non defective pictures have only 2 shapes, either 3000x4000 or 4000x3000, depending whether it has portrait or landscape orientation. Defective images have wider distribution in the shapes reaching as high as 8000 pixels and as low as 148 pixels. The minimum image

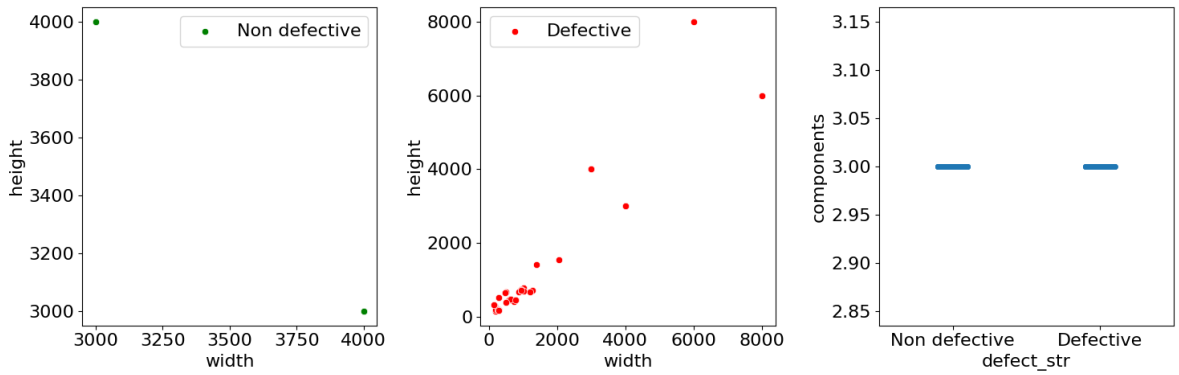


Figure 3: Distribution of the image shapes

size in this case is 148x194 (height x width) and the maximum is 8000x6000. All images have 3 color components according to RGB color model.

The distribution and correlation of the mean of the color components with respect to the images is shown on Figure 4. The different color components show comparable histograms. The case of the green

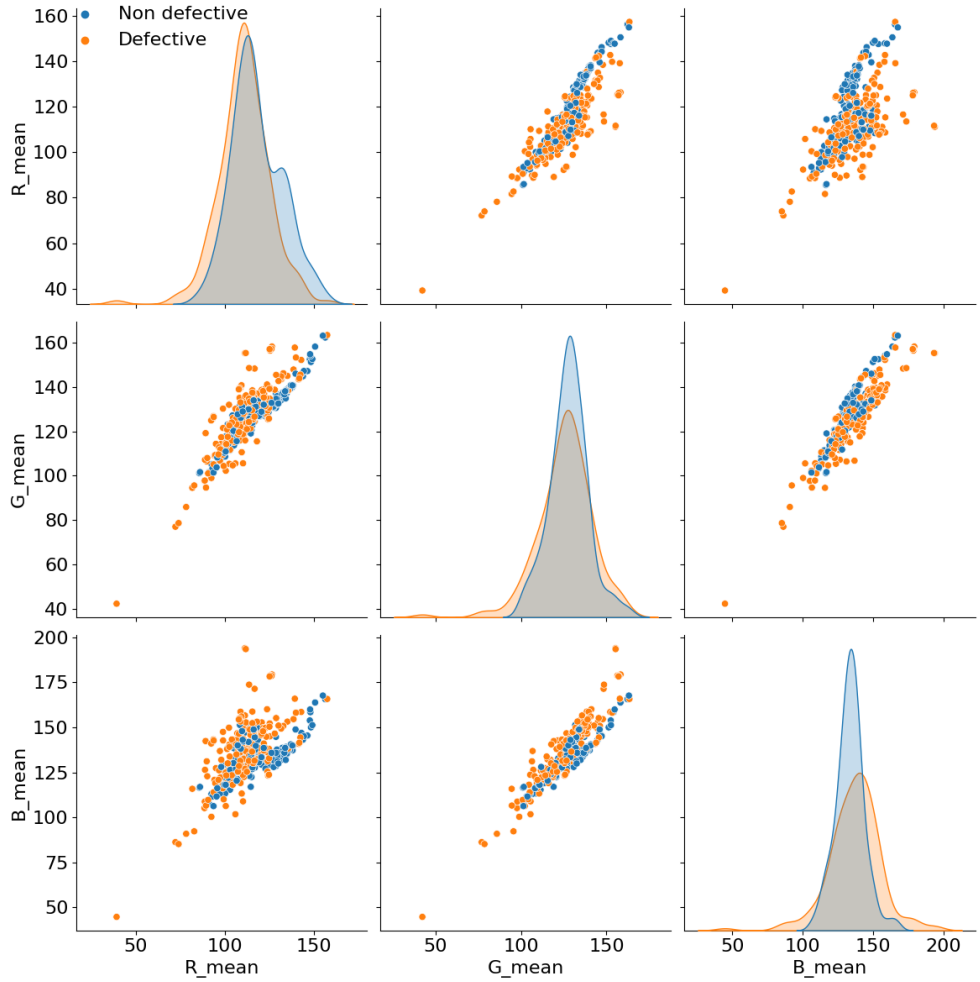
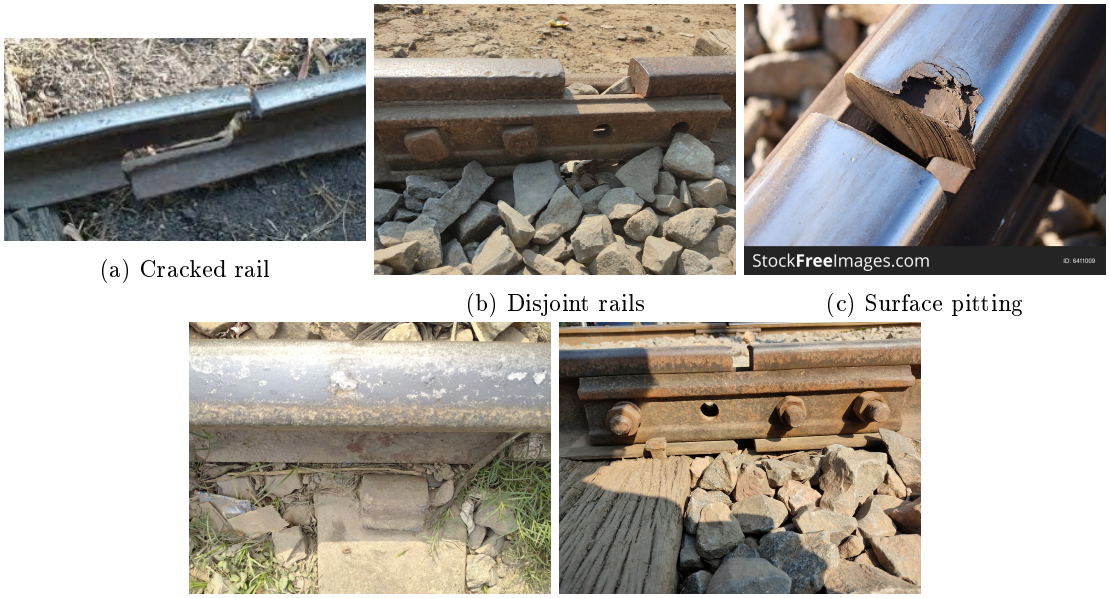


Figure 4: Color component pair analysis

and blue component the non defective components show a more narrow distribution. The paired color components in both classes represent similar feature spaces.

During the study of the random samples, the following defects have been identified.

1. Rail cracks. (Figure 5a)
2. Disjoint rail connections. (Figure 5b)
3. Pitting of the rail surface. (Figure 5c)
4. Improper fixation of the rail. (Figure 5d)
5. Missing elements, e.g.: screws, springs. (Figure 5e)



(a) Cracked rail (b) Disjoint rails (c) Surface pitting

(d) Wrong fixation (e) Missing element

Figure 5: Identified rail defects

The magnitude of each defect is varying in a wide range. For example in case of cracked rails, from a crack of a few millimeters up to a missing rail part of several centimeters can be found. Similarly, surface pitting can range from small cracks on the surface up to severe cases, like shown in Figure 5c.

The color component analysis on RGB and HSV colormodes revealed no exact differentiation between the classes. The component values highly depend on what is represented on the picture and how the photo is taken. For example, presence of shadow has a significant impact on the component histogram. An example of the analysis is shown on Figure 6.

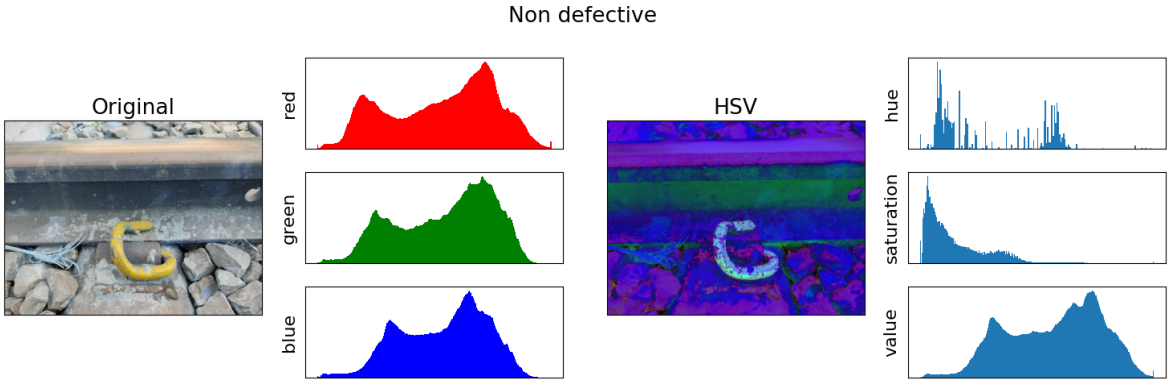


Figure 6: Color component analysis on RGB and HSV color modes

6 Discussion

7 Conclusion

List of Figures

List of Tables

References

[1] *Kaggle: Your Home for Data Science*. URL: https://www.kaggle.com/ (visited on 12/22/2022).

[2] *Railway Track Fault Detection | Kaggle*. URL: https://www.kaggle.com/datasets/salmaneunus/railway-track-fault-detection (visited on 12/22/2022).