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.

A comprehensive overview about visual inspection technologies used in the railway sector is given in [1]. It focuses on different technologies including machine learning and non machine learning applications and covers the overall railway sector including track, vehicle and infrastructure elements. The following major application categories identified: the railway track, the pantograph-catenary subsytem, the train

body and rail infrastructure. The railway track is discussed in several subpoints: the detection of rail surface defects, wear and deformation of track components, identification of rail components and the detection and extraction of rails. Current summary follows this review focusing on the track inspection methods applying machine learning algorithms based on image processing.

1.1 Surface defect detection

Xue Wang et al. presents a rail surface defect detection method [2] based on a liner CCD followed by image processing and applying a Quantum Neural Network (QNN). The accuracy results were evaluated and compared between QNN, Artifical Neural Network (ANN) and Support Vector Machine (SVM). First it detects region of interests (ROI) in the captured images by selecting pixels where the gray-level quantity is above a certain threshold. Then it searches for edges to identify the pixels that belong to certain edges. Then comparing this two set of points, a region of interest is defined by the presence of a set of pixels in both sets. Once this are marked, a rectangle can be drawn to enclose the region. Next step if to extract the features from the single ROIs. The features are defined based on the geometry of the flaw inside the ROI, the grey-level of the flaw pixels, and a so-called tansform-space, where a discrete cosine transform filter is applied. The obtained recalls are ranging between 86.7% and 100% depending on the type of flaw.

Another solution for rail surface defect detection is described in [3]. Images taken by maintenance vehicles from the top and evaluated for surface defects via two SVM classifiers applied one after the another. Overall 8 defect categories differentiated including the non-defect case. 939 manually annotated images was used for the experiments. A Random Forest based edge detection is followed by a Generalized Hough transform to detect the position of the rails. This overperformed the Canny edge detection method. Defect severity level classification was traced back to a texture classification task for that a Bag of Words model is used. A set of filters are used to create the feature vectors for each pixel in each image. From this the dictionary is created via a k-means clustering resulting in a histogram of k bins containing the number of pixels in each bin. These descriptors were fed for a χ^2-kernel SVM for training. Besides the dictionary texton forests were built. Instead of predicting the class, texton forests predict the image descriptors. The two methods were combined to an ensemble method by applying stacking. Overall 5 texton forest SVM classifier and 4 texton dictionary SVM classifier provides 9 probability vector with 8 elements for the 8 classes. A second level linear-kernel SVM was fitted to these outputs. An accuracy of 82% is achieved in the end.

Santur et al. presents a laser based imaging technology combined with deep learning model [4]. The rail profile is aquired via laser scanning and afterwards a convolutional neural network is applied. The article distinguishes between 4 defects: cracks, abrasion, corrugation and headchecks, however only binary classification is implemented (faulty or not faulty image). The rail imaging is done via stereovision, time of flight and laser triangulation to obtain a 3d image. Convolutional Neural Network (CNN) was applied on the image. The mentioned system achieved 98% accuracy.

Li et al. introduces a corrugation identification system that consists of an image acquisition system and the corresponding machine learning algorithm [5]. The images are taken from the top by equipment fitted to the underframe of the vehicle. The image processing covers the localization of the rail, feature extraction and corrugation recognition steps. Rail localization is done on the observation that the rail brightness is high and even and mostly located on the center of the picture. Features extracted by applying Fourier transformation for each column (lengthwise part of the rail surface) to determine the Fourier energy spectrum after truncation and sampling to reduce the dimensionality. Observation taken that the longitudinal average gray value is relatively high and distributed uniformly. Corrugation lines represented as periodic changes of the gray value that is a sparse energy distribution in the frequency domain. Corrugation afterwards can be detected by an SVM trained on these features. The proposed solution reaches 99% accuracy.

1.2 Component detection

Li et al. presents a methodology in his work in [6] extended in [7] and [8] to detect missing rail components focusing on the tie plate and it's surrounding localization. A fixed position camera is used in the research that allows a steady assessment of the track components. Mostly relying on edge detection methods on grayscale images, at first detecting the tie plate itself and then the other components can be identified based on the geometrical positioning.

Connection element binary classification is presented in [9] that is explained more in detail in [10].

The method is able to detect the presence of fastener bolts or hooks. Fastening elements are extracted as features from grayscale pictures taken from top view. Wavelet transformation and principal component analysis (PCA) performed to describe the features in low dimensionality. During the wavelet transformation a high-pass and low-pass filter is applied in every combination, extended by applying the same combinations on the low-pass-low-pass result. This results in a multilevel (depending on the number of extensions) wavelet transform. Two type of neural networks considered: Multilayer Perceptron and Radial Basis Function. A training set of 173 images used for the bolt detection and 221 images for the hook detection. Depending on the object to detect the wavelet transformation with levels of 3, 4, and 5 performed best, reaching accuracy rates of above 98% in almost all cases.

A commercial solution is presented in [11] utilizing GPU based hardware. It consists of an integrated image acquisition system followed by a prediction modul to identify image areas for classification and a bolt detection module. The bolt detection module uses Multilayer Perceptual Network Classifiers (MLPNC) fed by features generated through wavelet transformation. Training dataset consists of approx. 1000 images of 24x100 pixels window, that are extracted from recorded videos. The MLPNCs have a structure of 150:10:1 layers. The results of different MLPNCs combined with and AND operator. The prediction and detection modules are built on a GPU based hardware for fast execution. An accuracy rate of 99.9%, 0.1% and 95% was reached for visible, occluded and absent bolts respectively.

Another component detection is introduced in [12]. Top view rail images used for detecting the fastening elements taken by the image acquisition system taken in a way to minimise surroundings of the ROIs. The alogrithm applies preprocessing, grayscale conversion, feature point detection, feature extraction and feature matching to detect rail anchors. The images are resized to 200x200 and then cropped to 170x67 to minimise the noise caused by the surroundings. Harris-Stephen and Shi-Tomasi feature detectors were applied. The features of the train and test images were matched. If the number of matchings is above a certain threshold, than presence of a rail anchor is assumed. The method achieved an average success rate of 83.55%.

In [13] a robust fastener detection method proposed based on combination of SVMs on grayscale images. To avoid image segmentation a sliding window approach was used using Histograms of Oriented Gradients. It differentiates between missing, broken and good fastener classes. Furthermore it assignes the fastener type. For training 30 good quality images ware manually tagged from each class. A total of 3805 images were used for training, among that 1069 good fasteners, 714 broken ones, 33 missing and 1989 background patches were included. The method was compared with intensity and HOG based Optimal Tradeoff Maximum Average Correlation Height (OT-MACH) filter and SVM.

A follow up work of Gibert, Patel, and Chellappa in [14] presents classification via CNN. A CNN of 4 convolutional layers were applied with ReLU activation functions combined with max pooling units. Dropout was not necessary in the study. As there were no preprocessing of the images a normalization was applied on the input via a median filter. The mean intensity was subtracted from each image. The network was trained with Stochastic Gradient Descent (SGD) on batches of 64 images of size 75x75. Data augmentation was applied in means of flipping and cropping. Train data was resampled to contain at least 50% of adverse environment (mud, oil, etc.) images. The total training data was 50.000 patches of each class. Total iterations were set to 300.000 with a momentum of 0.9 and weight decay of $5x10^{-5}$. Learning rate was set to 0.01 and a decay factor of 0.5 every 30.000 iterations were applied. The CNN reached accuracy of 93.35%.

Detection of hexagonal fastener elements is described in [15]. Pixel similarity based approaches (principal component analysis, linear discriminant analysis, random forest, sparse representation, multi-template matching) and histogram-based similarity approaches (histogram matching, depth peeks) were compared considering their combinations as well. Performance was evaluated based on false alarm rates, that is the ratio of false negative hits compared to total negative hits. Best performance was reaching a rate of 4.72%.

An Adaboost based component classification method is presented in [16]. Top view pictures taken and evaluated by the algorithm. Considering that the gray value of the sleeper is bigger than the value for the ballast, differentation can be done by summing the weighted gray values row-wise. First the sleeper is localized, then the fastener is detected. Haar-like characteristics were calculated to select the features of the fastener. The state of the fastener is then estimated by the Adaboost algorithm. 4111 images were processed by this algorithm, containing 7059 fasteners from which 20 was broken. 18 broken fastener was correctly recognized. The method is sensitive to gravel on fastener and for oil stains and variant illumination.

1.3 Further detection methods

In their study Kumar M. et al. compares crack detection methods that are based on different measurement solutions and image processing algorithms [17]. The preprocessing steps are explained in detail. They refer to conversion grayscale images as main approach. Noise reduction and the sharpening of the picture is taken as first step. For further evaluation several edge detection techinques referred that might be applied.

A comprehensive method is described in [18]. Utilizing a multi-camera system and extended image processing fault detection on rail track components performed. Taken images are subject to grayscale conversion, canny edge detection and filtering that is followed by identifing line sections via Hough transform. Classification of each line is performed via the angle and horizontal position to determine rail and sleeper positions. Rail surface monitoring is done on binary format of the rail section. The binary format allows the differentiation of normal rail surface and defect surface. Calculating the number of pixels with defect (considered as black pixel) defects can be identified. The study uses fixed camera system, therefore fixed input orientation for the processing algorithm.

A general CNN classification is presented in [19]. A neural network was trained to classify 1.2 million images downsampled to a size of 256x256 up to 1000 different classes. The CNN consists of 5 convolutional layers and 3 fully connected layers. ReLU activation function was applied followed by normalization and overlapping max pooling layer. The network is fitted to parallel GPU computation by splitting each of the layers to the GPUs. Data augmentation and dropout were applied to reduce overfitting. The training is done with a batch size of 128, momentum 0.9 and weight decay of 0.0005. The network reached up to 36.5% top1 error rates.

1.4 Convolutional Neural Networks

The intention of the study is to construct a CNN for the classification task. As it will be underlined later in Section 5.2 the use of traditional approaches are very limitied in the selected dataset. One of today's most favoured approach is to apply CNNs for image classification task, therefore some introduction on the topic on constructing the neural network shall be presented besides the articles already cited in Section 1.

In [20] the general approach on finding the proper CNN architecture is explained. The main steps are the following: define the Search Space, select a Search Strategy and estimate the performance of a selected model. Due to high computational needs of the different models a particular focus is put to training speed as key factor in finding the proper model. More details can be found in [21].

Deep convolutional neural network (DCNN) was proposed in [22] to detect surface defects of rails. Video recordings of rail tracks are processed by DCNNs with different sizes related to the size and number of the filters, number of convolutional layers and the size of the fully connected layers. Hyperbolic tangent and ReLU activation functions used. The applied dataset consits of more than 22.000 manually labelled images of 6 classes (normal surface, weld, light, moderate or severe squat and joint defects). Input image size is 100x50 pixels and converted to grayscale. The multi class accuracy is ranging between 91.17% and 92.47%. The performance increases by increasing the size of the DCNN and by applying ReLU activation functions.

Another model is described in [23] for surface defect detection. A CNN is applied after preprocessing of the images. Canny edge detection is applied to the grayscaled images, that is followed by removing the false edge points. The point removal is based on the dynamic range of the edgepoints that is adaptively narrowed to obtain a set of points with low standard deviation. Once the false points removed, linear fitting performed to obtain the coefficients of the rail edge linears. Inception-v3 CNN was applied with transfer learning. More than 5000 images were used for the training, more than 2000 for the validation and around 1500 for the final test. The image size for the CNN is 960x1280 pixels. The model results in 92.54% recall rate and 92.08% precision.

A track crack detection approach is presented in [24]. The image preprocessing covers an adaptive histogram adjustment to increase the contrast of the picture followed by a grayscale conversion. Gabor transform was applied and the magnitude picture was taken to extract the model features. The following statistical values are calculated: mean, variance, skewness, kurtosis, energy and entropy. A CNN classifier was applied reaching an accuracy rate of 94.9%. One possible approach to achieve high accuracy and reduce development (and training) time is to used pre-trained models. This is realized by the so called transfer learning approach, where already trained models can be applied for specific tasks. These models are accessible and can be partially retrained to fit to the exact problem. It's applications are growing in te field of image processing.

The model of EfficientNet is brought up in [25] by describing a scaling method of different CNNs to increase accuracy whilst maintaining acceptable training times. It defines a ratio between width, depth and resolution (image size) of the model and proposes to compoundly scale the models to find the optimal size. By scaling ImageNet the top-1 accuracy rate of 84.3% maintained whilst being 8.4x smaller and 6.1x faster than the best existing CNN.

The performance of VGG16, VGG19 and ResNet50 is compared in [26]. The validation accuracy of 51.45%, 83.01% and 75.41% is achieved respectively. To avoid overfitting regularization as dropout layers with 30% and data augmentation was added. 35.000 images were used for the training and additional 10.000 and 15.000 for testing and validation.

A case study of rail track crack detection is presented in [27] utilising pre-trained VGG16 model. 1905 images were used for the training with a size of 256x256. The dataset is from Kaggle, another variant of the railway track detection consisting of much more images (https://www.kaggle.com/datasets/ashikadnan/railway-track-fault-detection-dataset2fastener) [28]. The achieved accuracy on the validation set is 90.62% without and 98.61% with data augmentation.

1.5 Summary

There are several approaches present in the current scientific methodology to identify rail track defects. A clear differentation can be done based on the nature of the defect, one significant part of the studies deal with surface defects and another part deals with component identification. Currently there is no superior approach in the field, several different solutions present that are fitted to the exact problem definition and circumstances of the issue. However certain major steps can be identified in the different aspects. Image acqusition, image processing, feature extraction, model application form the four major pillars.

In most cases the problem is narrowly defined. The task is to estimate whether a 'well-known' type of defect is present on the images or not. This could be a rail corrugation, a missing screw or a broken fastening clip.

Image acquisition is done by a clearly defined system that is mostly equipped to a rail vehicle. In some cases the origin of these is not known, at least in terms of the acquisition method. It is common that the general layout, by means of orientation, distance from the track, position of the components can be assumed as standard. The pictures are often deducted from video recordings.

Preprocessing of the images show a high variety from simple grayscale conversion to edge and line detection methods. With a few exceptions, all methods start with grayscale conversion. Normalization, contour enhancement and noise filtering are common tools however not universally used. Different edge detection technics (Canny, Sobel, Laplacian, etc.) are applied, in most cases followed by Hough line transform.

The extraction of the features is the most diversified part. Further image processing tools can be employed, such as FFT, wavelet or Gabor transform. Image feature descriptors belong here, namely the different histograms, corner detection, SIFT transform. For example detecting the rail surface based on the low variation of the grayscale values can be easily identified. Additionally the localization of components via feature matching between the reference and target images or by the detected edges and lines is supplying valuable features of the model.

Once the rail surface or the components in question identified or the necessary features computed the application of machine learning algorithm could take place. PCA, LDA and SVM is widely used, in several case not only a single model is built but a group of models which supply a joint estimation of the classes. A special group is the neural network (mostly convolutional deep neural network) that applies very limited image processing and the feature extraction is done by the neural network itself.

Another notable aspect is the dataset corresponding to the issue. The size of the training set consists mostly thousands of tagged images. The size of the pictures, variety in terms of environmental influences, illumination and resolution changes in a wide range. In several cases data augmentation was done, flipping, rotating, cropping of images is a common approach.

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 [29] and can be downloaded directly from https://www.kaggle.com/datasets/salmaneunus/railway-track-fault-detection [30]. 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.

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

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 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 study 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 several Jupyter notebooks and can be found on the following repository:
https://github.com/demustamas/Mathemathical-Modeling-Practice

The following jupyter notebooks and utilities are differentiated:

1. ./data_cleaning.ipynb Processes data from raw to data state.
2. ./data_explorer.ipynb Contains the data exploration steps.
3. ./classification_model_* The overall CNN based classification model, where * stands for the applied network model.
4. ./playground.ipynb Contains the algorithms that are under development.
5. ./toolkit/classes.py Stores class definitions.
6. ./hp_tuner.ipynb Tool used for hyperparameter tuning.

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. (Created by the scripts.)
3. ./augmented/ Additional images as result of data augmentation. (Created by the scripts.)
4. ./preprocessed/ The data after image processing, preprocessed for the machine learning algorithm, includes both the original and augmented dataset. (Created by the scripts.)
5. ./models/ Stores the trained classification models. (Created by the scripts.)

During data processing the following utility folders were used:

1. ./logs/ Log files created during the training of neural networks. (Created by the scripts.)
2. ./tuner/ Log and model files created by the hypertuner. (Created by the scripts.)

The project documentation covers the following files and directories:

1. ./documentation.tex LaTeXfile of the overall documentation.
2. ./presentation.tex LaTeXfile of the presentation.
3. ./tex_graphs/ Folder of the autegenerated images to include in the documentation.
4. ./tex_refs/ Reference files for the LaTeXdocumentation
5. ./build/ Build directory for the documentation, contains the rendered pdf files. (Created by the scripts.)

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, correcting 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. Considering that the rail surfaces show low standard deviation in the gray values combined with distinct values compared to the mean it makes the identification possible. Although no other characteristics mentioned in the reviewed articles in Section 1. This is investigated on the RGB color model and the images were converted to HSV model to get a more comprehensive view.

4.3 Model structure

The construction of the CNN follows two main approaches. The first one is to build a model from scratch and try to find a setup that learns the features of the given dataset. As basis the structure of LeNet-5, AlexNet and VGG16 is considered as these were the first networks used for image classification and these bear a structure that is not too complex for implementation. A detailed view on these networks can be found in [31]. Based on the reviewed literature and the very high number of variables in a CNN present an extended dataset is required for this method. Therefore the expectation about the achievable accuracy shall not be raised too high.

The second approach is to use a pre-trained model and apply transform learning. The models of VGG16 and Resnet50 are chosen as these are available in the Tensorflow Keras ecosystem [32]. The models were downloaded pretrained on the Imagenet dataset without the dense layers on the top of the network. These layers then added to the network and trained manually on the dataset whilst keeping the core convolutional layers freezed (feature extraction only). Once the training is done, than the layers were unfreezed (except the BatchNormalization in case of the ResNet50) for fine-tuning the model. As this is becoming more and more a common way and also already studies done in this direction (refer to [28]), an acceptable accuracy range is expected, at least in the range of what is already achieved in different studies.

The model applies Adam optimizer, EarlyStopping and ModelCheckpoint to monitor validation accuracy. In all cases the output layers are modified to supply binary values as we need to classify between defective and non defective images with a corresponding sigmoid activation function and binary crossentropy loss function.

The input images are subject to some limited image processing that covers grayscale conversion, histogram equalization, noise filtering and resizing to the applicable input size. All manipulation is done with the OpenCV module, version 4.7.0. [33].

Data augmentation was performed to increase the dataset size by taking defective and non defective images from the training set and applying one of the following: flipping, with a random angle between +90°and -90 and random zoom with a ratio between 0.8 and 1.2. Due to computational limitations the data augmentation is separated from the training of the model and done prior to any training step. This leads to a the situation that the dataset is extended with a very limited number of new samples.

4.4 Model selection and evaluation

Selecting the best model is based on the best validation accuracy achieved, that is monitored by the EarlyStopping callback. Once the training is done, the best model weights are reloaded and the final performance estimation is done on the test dataset. For discussion the following metrics are used:

1. Accuracy of the model on the training and validation dataset.
2. Loss function (learning curve) of the training and validation dataset.
3. Confusion matrix built on the test dataset.
4. ROC curve based on the test dataset.

5 Results

5.1 Data cleaning

The raw data from the Kaggle webpage [30] 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.

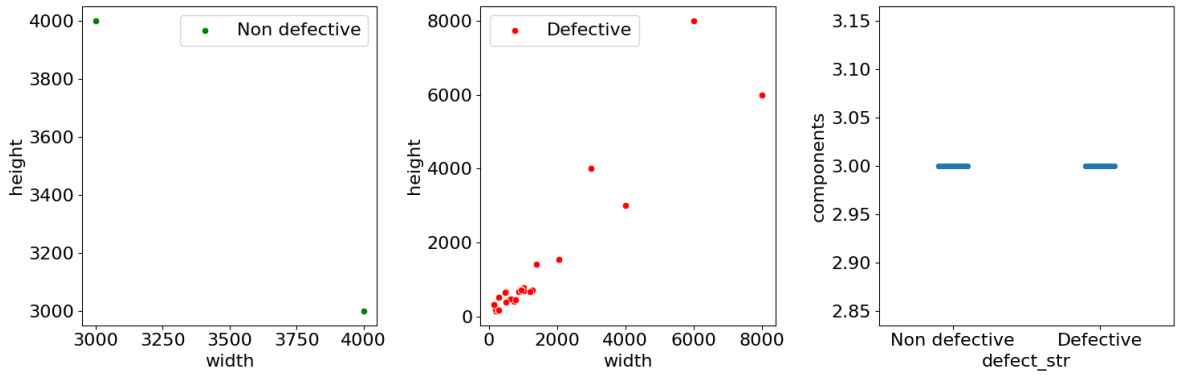


Figure 3: Distribution of the image shapes

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 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. However a differentiation based on the image shape might no be wise as it is missing any content related information. A new image taken with different resolution might lead to improper prediction in the end. The spread over the sizes indicate that during image processing if resizing is considered than not only downscaling but upscaling might be necessary.

The distribution and correlation of the mean of the color components with respect to the images is shown on Figure 4.

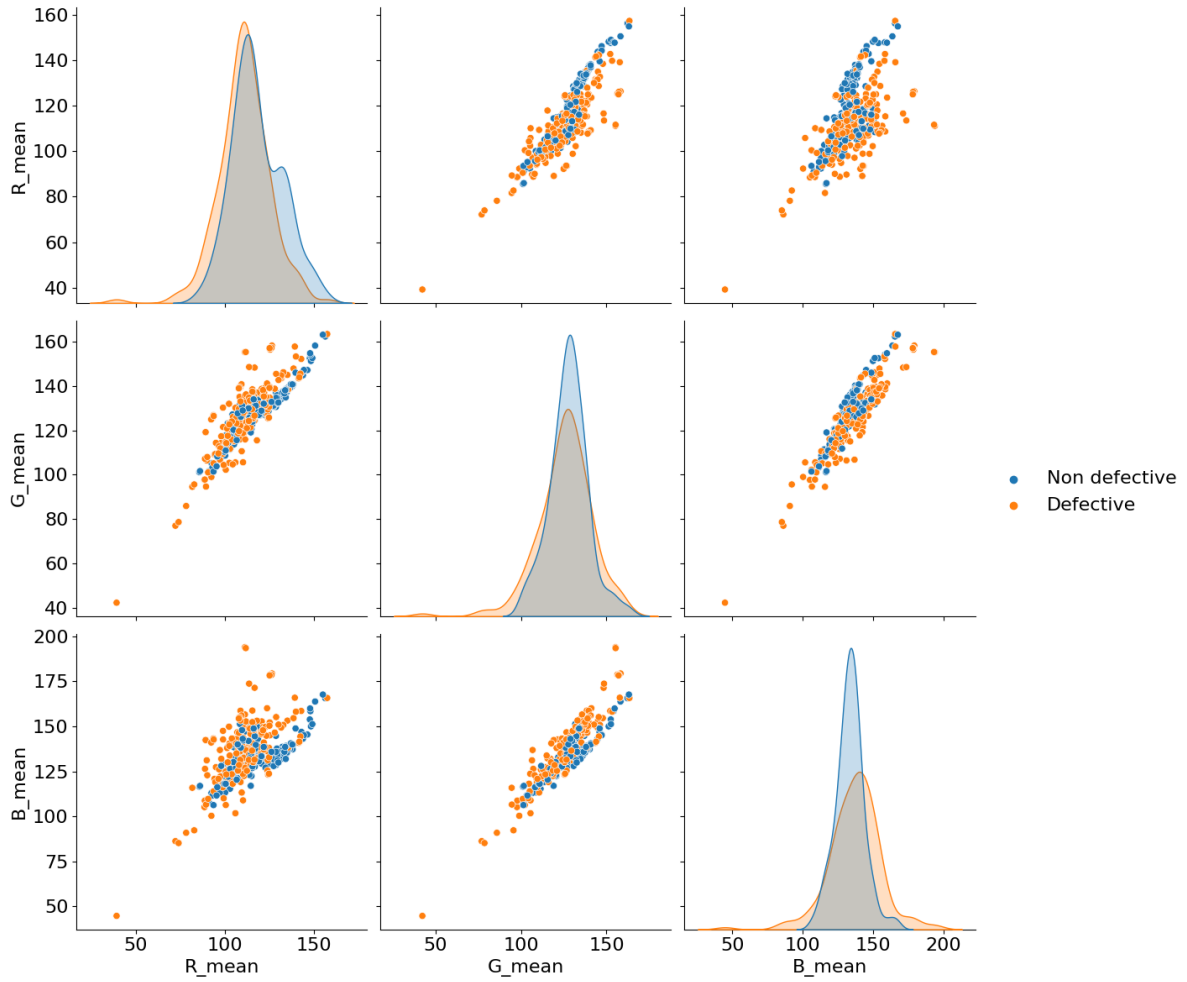


Figure 4: Color component pair analysis

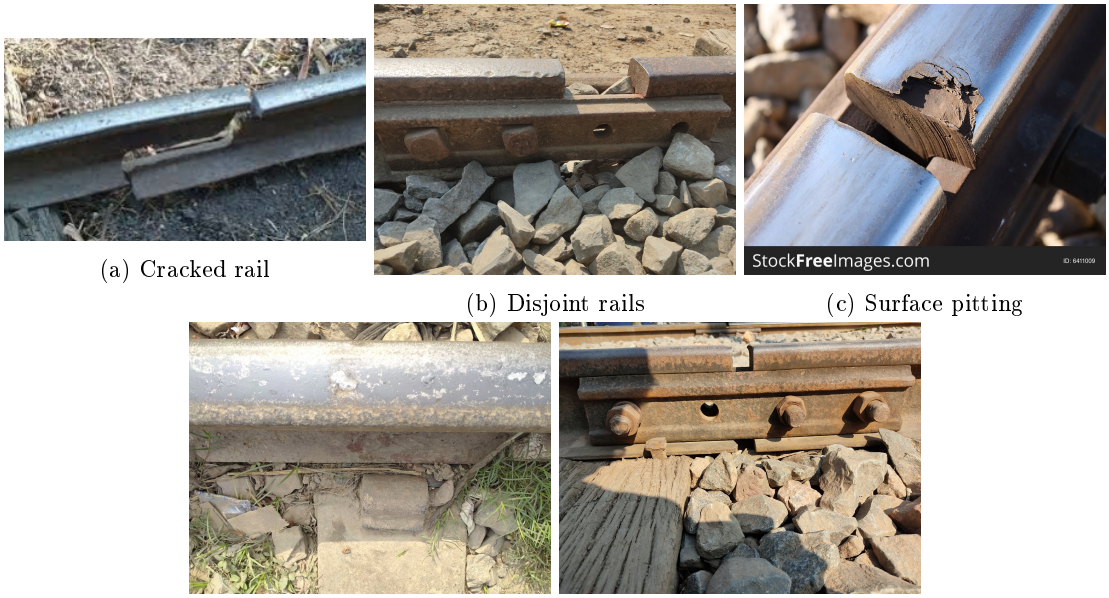
The different color components show comparable histograms. The case of the green and blue component the non defective components show a more narrow distribution. The paired color components in both classes represent similar feature spaces. Out of this no clear differentiation between defective and non defective classes can be done based on the color components.

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. Defect fastener. (Figure 5d)
5. Missing elements, e.g.: screws, springs. (Figure 5e)

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. In several cases components are missing that can be screws, fastening hooks or similar. There are cases when these components are covered with ballast rocks.

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.



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

(d) Fastener defect (e) Missing element

Figure 5: Identified rail defects

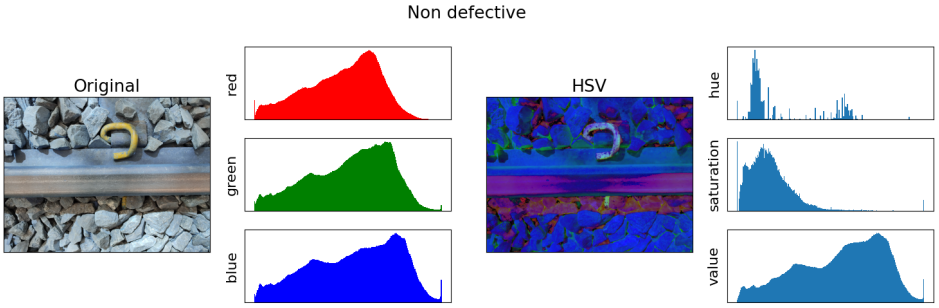
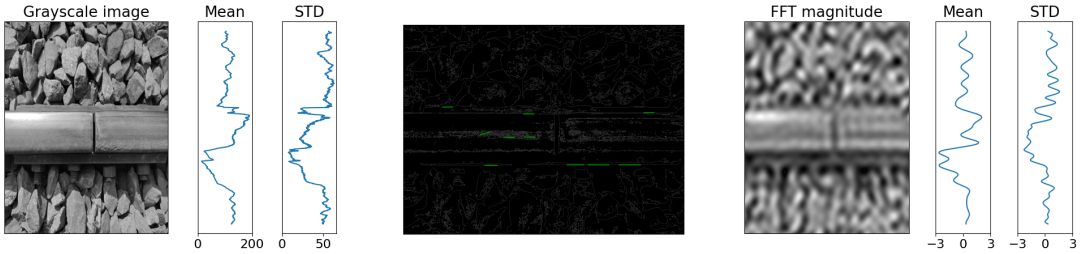


Figure 6: Color component analysis on RGB and HSV color modes



(a) Grayscale analysis (b) Edge detection (c) FFT based detection

Figure 7: Rail surface recognition

An example of the analysis is shown on Figure 6. These histograms are taken for the overall picture, that is sorting every pixel into bins depending on their color values.

The grayscale analysis makes it possible to recognize rail surface as mentioned before. The standard deviation and the mean value of the grayscale values correspond to the surface. In cases when the rail orientation is alongside with the rows or columns, the identification is relatively easy. If the means and standard deviations of the grayscale values are calculated row or columnwise, the resulting function will bear the characteristics of the rail surface. This is indicated on Figure 7a.

In similar cases the edge detection works quite efficiently as the rail itself bears a couple of well-defined edges. This makes it possible to detect line segments on the edge map which allows detection of the rail

orientation. Once this is known, the image can be rotated in an angle that the longitudinal position of the rail will be perpendicular to the calculation direction as shown in Figure 7b. This leads to the easy detection of the rail edges and makes the line detection (based on the Hough probability tansform) very robust as indicated with the green sections that show the longest lines detected. The surface detection can be improved by applying a FFT transform to the image and standardising the mean and standard deviation values. Such composition is shown on Figure 7c.

Due to the high variety of the image content (as shown in Figure 8), orientation and defect method, none of the mentioned algorithm lead to a clear segmentation of the image. This prevents the localization of any classification algorithm and also prevents the application of any approach that is mentioned in Section 1. To cope with the classification task we turn then to CNNs.

Figure 8: Variety of dataset

5.3 LeNet-5

LeNet-5 is one of the earliest models used originally for the handwritten digit recogniton. It suffered from computational limitations due to the low performance of the computers at the time of being constructed, but it defined the basic structure of a convolutional networks used afterwads. The structure of the neural network according to Tensorflow Keras is shown in Table 2. As the input is limited to a single channel, the images were converted to grayscale followed by adaptive histogram normalization (CLAHE) and noise filtering with a median filter with kernel of 11x11. After all the changes applied, the images were resized to 32x32 pixels.

Layer	Filters / Neurons	Size / Rate	Padding	Stride	Feature map	Activation
Input	-	-	-	-	32 x 32 x 1	-
Rescaling	-	-	-	-	32 x 32 x 1	-
Conv2D	6	5 x 5	valid	2	28 x 28 x 6	tanh
AveragePooling2D	-	2 x 2	-	1	14 x 14 x 6	-
Conv2D	16	5 x 5	valid	2	10 x 10 x 16	tanh
AveragePooling2D	-	2 x 2	-	1	5 x 5 x 16	-
Flatten	-	-	-	-	400	-
Dense	120	-	-	-	120	tanh
Dense	84	-	-	-	84	tanh
Dense	1	-	-	-	1	sigmoid

Table 2: LeNet-5 layer structure

A learning rate of 0.0001 was used as initial value which was decreased by the ReduceLROnPlateau callback to 0.000005 by a factor of 0.5 and with a patience set to 2. The accuracy peak on the validation dataset reached with a low number of epochs. The resulting metrics are shown in 9.

The accuracy increases rapidly in the first few epochs topping at around the 10th epoch at 59.67% on the validation dataset. Afterwards the training accuracy slightly decreases and the validation accuracy decreases as well. The latter is showing bigger fluctuation, mostly due to the limited number of validation data points, a single change in the classification in one of the images leaves to a bigger relative change. The loss function decreases continuously for both datasets, howevere an increasing distance between the two is observed. The confusion matrix and the ROC curve shows a classifier with 59% accuracy. However a deeper look into the learning curves reveal that the increasing gap between the training and validation data shows that the training data might be unrepresentative to the test data. It has to be noted that

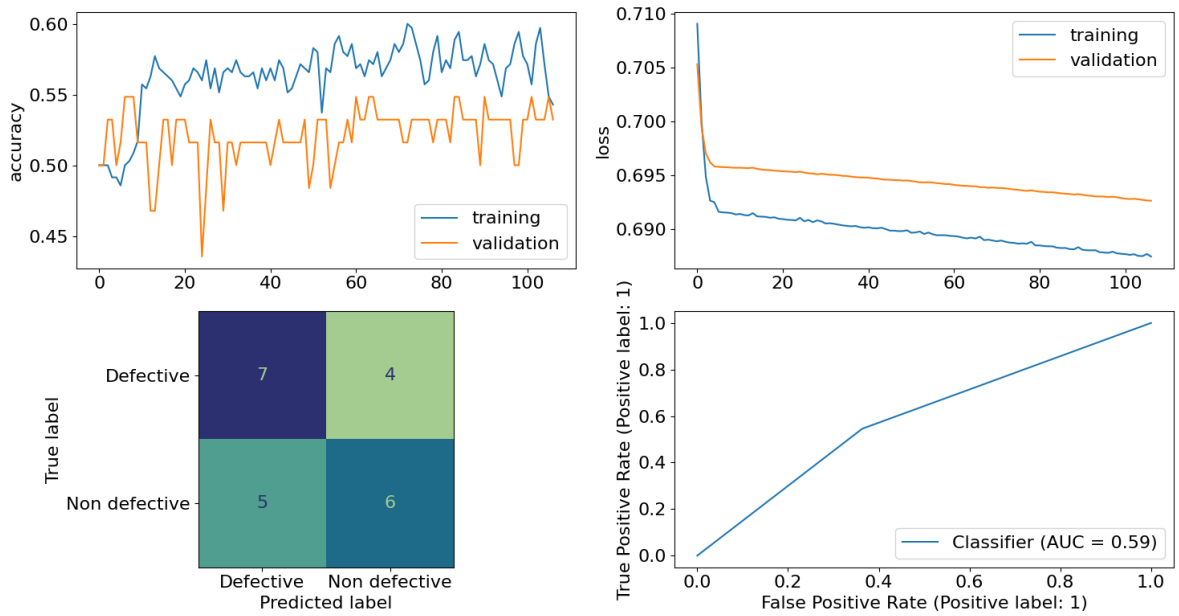


Figure 9: LeNet-5 metrics

for different training sessions the confusion matrix and ROC curve also show some variance of accuracy ranging down to 45%. The resulting trained model did not prove to be stable. This could be due to the fact that the layer structure is very limited and failed to learn the proper features or the training data is not representative to the test data.

5.4 AlexNet

AlexNet is the second major convolutional network borught up several years after LeNet-5 was introduced. Compared to LeNet-5 it's depth, width and resolution is increased significantly. The structure is described in Table 3. It introduces ReLU activation function together with DropOut layers and overlapping pooling. The input accepts color pictures, therefore no manipulation of the images were done, except resizing to correct size.

Layer	Filters / Neurons	Size / Rate	Padding	Stride	Feature map	Activation
Input	-	-	-	-	227 x 227 x 3	-
Rescaling	-	-	-	-	227 x 227 x 3	-
Conv2D	96	11 x 11	valid	4	55 x 55 x 96	ReLU
MaxPooling2D	-	3 x 3	valid	2	27 x 27 x 96	-
Conv2D	256	5 x 5	same	1	27 x 27 x 256	ReLU
MaxPooling2D	-	3 x 3	valid	2	13 x 13 x 256	-
Conv2D	384	3 x 3	same	1	13 x 13 x 384	ReLU
Conv2D	384	3 x 3	same	1	13 x 13 x 384	ReLU
Conv2D	256	3 x 3	same	1	13 x 13 x 256	ReLU
MaxPooling2D	-	3 x 3	valid	2	6 x 6 x 256	-
Dropout	-	0.5	-	-	6 x 6 x 256	-
Flatten	-	-	-	-	9216	-
Dense	4096	-	-	-	4096	ReLU
Dropout	-	0.5	-	-	4096	-
Dense	4096	-	-	-	4096	ReLU
Dense	1	-	-	-	1	sigmoid

Table 3: AlexNet layer structure

During training the same optimization settings were applied as for LeNet-5. The training time increased significantly due to the increase of the trainable parameters. The results are shown in Figure

10.

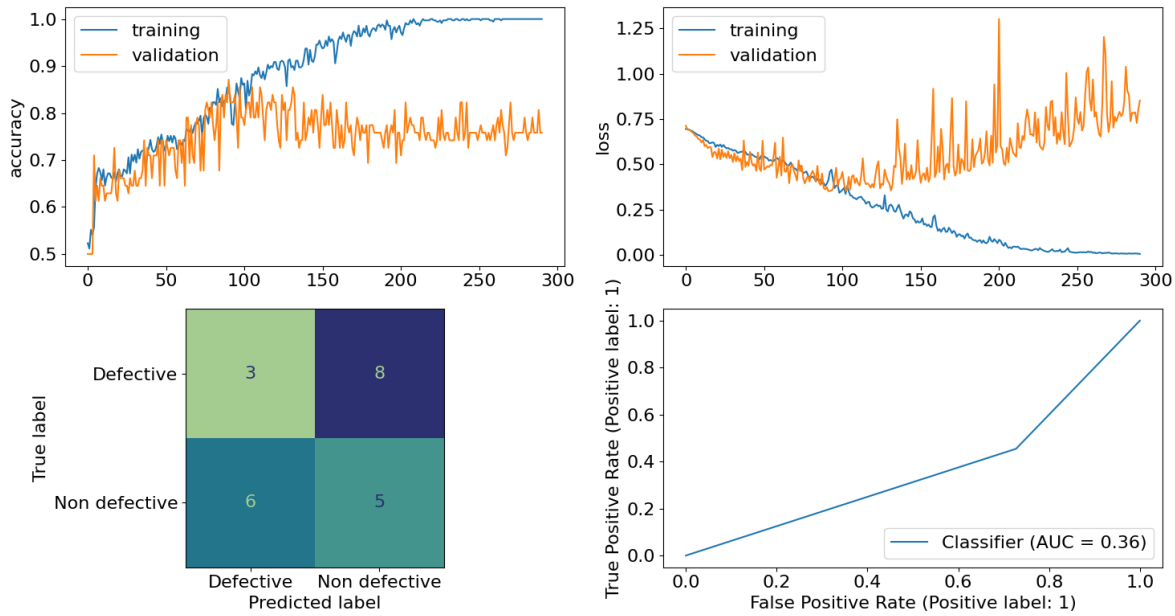


Figure 10: AlexNet metrics

An accuracy rate of 87% achieved on the validation dataset, whilst the training accuracy reaches 100%. This indicates that the network is able to learn the training dataset, but an overfitting can be seen after approx. 120 epochs. Same can be seen on the learning curves as well. The confusion matrix understates that the training data is not representative to the test data. A fairly good fitting validation set did not result in a good fit on the test data, only an accuracy of 36 % achieved.

5.5 VGG16

The VGG model brings a different approach to the structure of the CNNs. Instead of using single big filters (like a 11 x 11 filter in AlexNet), several smaller filters applied consecutevily. Applying such small filter on the input layer results in a larger feature map on the input. Additionally increasing the number of consecutive layers reduce the tendency to overfit. The implemented structure of VGG16 is shown on Table 4. As input image, resized color images used without any further alteration.

The initial learning rate is reduced to 0.00001 and the ReduceLROnPlateau patience was set to 1. This helped to find the local minimum only in a few starting epochs. This model required the longest training time. The resulting metrics can be seen on Figure 11.

The training needed much less epochs than in previous case, reaching the validation accuracy of 83.87% at the 64^{th} epoch. An overfitting can be seen from the learning curve, which indicates optimal epoch number of 45, where the validation accuracy was 77.41%. The model fitted to the test data resulting in an accuracy of 59%, although due to long running time only a single training is performed, therefore the assumption that the training data is not representative to the test data can not be dropped.

5.6 Pretrained VGG16

In order to improve our results, pretrained models are applied. To allow comparison on the effect of prelearned features the same VGG16 model was selected. An initial learning rate is set to 0.00001 with a ReduceLROnPlateau callback set with a patience of 1 and minimal learning rate of 0.000001.

The model shows significant improvement in the performance, the top accuracy of 91.93% is reached after 6 epochs on the validation set. The unrepresentativeness of the training data can be observed slightly on the learning curves after approximately 10 epochs. The test data shows a classifier with an accuracy of 59%. During fine-tuning both the initial and the minimal learning rate was reduced to the one tenth. Unfortunately no improvement to the model achieved.

Layer	Filters / Neurons	Size / Rate	Padding	Stride	Feature map	Activation
Input	-	-	-	-	224 x 224 x 3	-
Rescaling	-	-	-	-	224 x 224 x 3	-
Conv2D	64	3 x 3	same	1	224 x 224 x 64	ReLU
Conv2D	64	3 x 3	same	1	224 x 224 x 64	ReLU
MaxPooling2D	-	2 x 2	valid	2	112 x 112 x 64	-
Conv2D	128	3 x 3	same	1	112 x 112 x 128	ReLU
Conv2D	128	3 x 3	same	1	112 x 112 x 128	ReLU
MaxPooling2D	-	2 x 2	valid	2	56 x 56 x 128	-
Conv2D	256	3 x 3	same	1	56 x 56 x 256	ReLU
Conv2D	256	3 x 3	same	1	56 x 56 x 256	ReLU
Conv2D	256	3 x 3	same	1	56 x 56 x 256	ReLU
MaxPooling2D	-	2 x 2	valid	2	28 x 28 x 256	-
Conv2D	512	3 x 3	same	1	28 x 28 x 512	ReLU
Conv2D	512	3 x 3	same	1	28 x 28 x 512	ReLU
Conv2D	512	3 x 3	same	1	28 x 28 x 512	ReLU
MaxPooling2D	-	2 x 2	valid	2	14 x 14 x 512	-
Conv2D	512	3 x 3	same	1	14 x 14 x 512	ReLU
Conv2D	512	3 x 3	same	1	14 x 14 x 512	ReLU
Conv2D	512	3 x 3	same	1	14 x 14 x 512	ReLU
MaxPooling2D	-	2 x 2	valid	2	7 x 7 x 512	-
Flatten	-	-	-	-	25088	-
Dense	4096	-	-	-	4096	ReLU
Dense	4096	-	-	-	4096	ReLU
Dense	1	-	-	-	1	sigmoid

Table 4: VGG16 layer structure

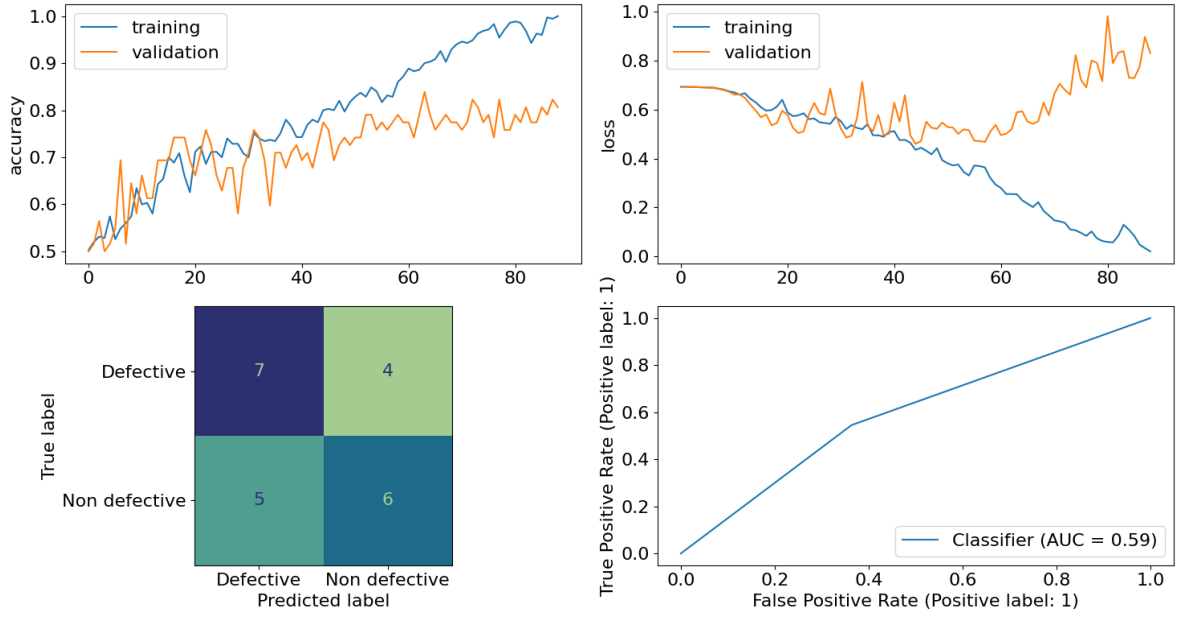


Figure 11: VGG16 metrics

5.7 Pretrained ResNet50

Second pretrained model used is the ResNet50 that applies fewer filters with lower complexity. Shortcut connections added between sets of convolutional layers to perform identity mapping.

The training with feature extraction settings done with an initial learning rate of 0.0001 and a minimum learning rate of 0.000005 added with ReduceLROnPlateau.

The model performed fairly well reaching an accuracy rate 67.74% on the validation set. The loss

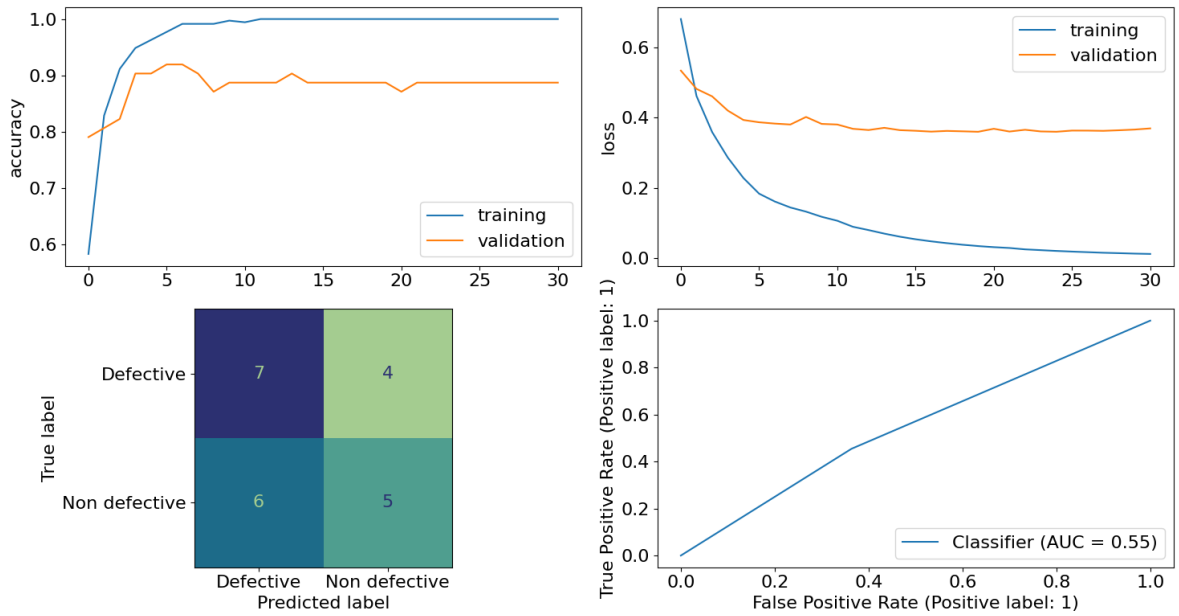


Figure 12: Pretrained VGG16 metrics

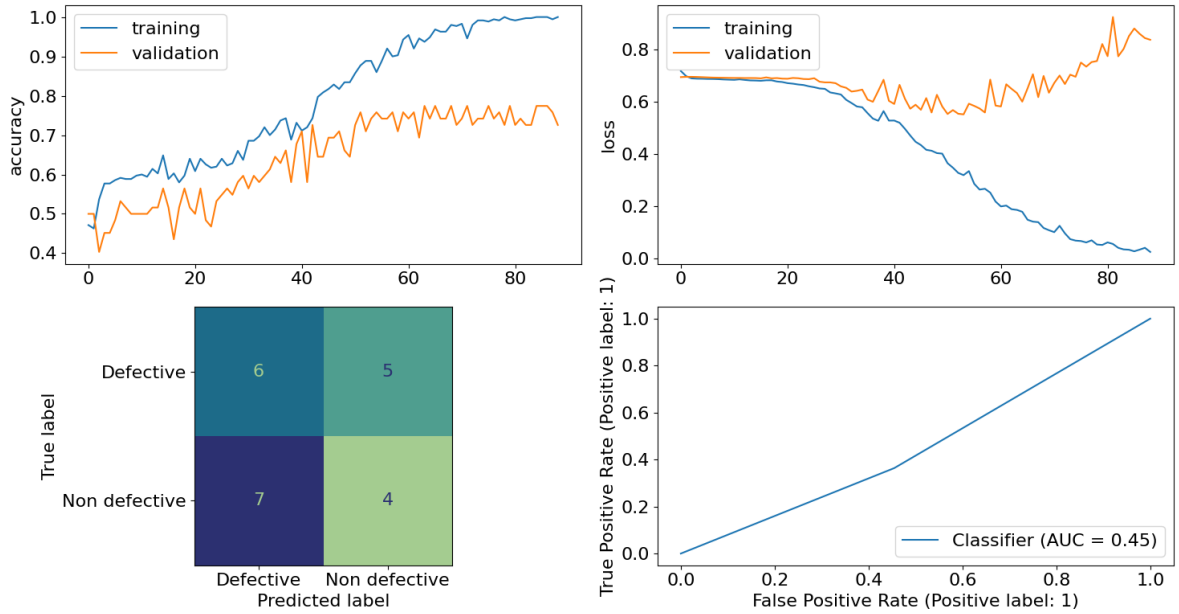


Figure 13: Pretrained ResNet50 metrics

function of the validation set is lower than of the training set indicating that the model was fitted better on the validation set than to the training set. This might happend due to regularization applied during training, low size of validation set, the validation set is not representative to the training set or due to the applied data augmentation. In our case all of the above (except the regularization) might contribute to this effect.

Applying fine-tuning by reducing the learning rates to the one tenth resulted in a siginificatn improvement of the model performance. The accuracy on validation set reached 79.03%. However the unrepresentative train dataset can be once again observed on the learning curves. For both results (feature extraction only and fine tuning applied) the classifier performed poorly on the test dataset with 41% and 45% accuracy rates respectively.

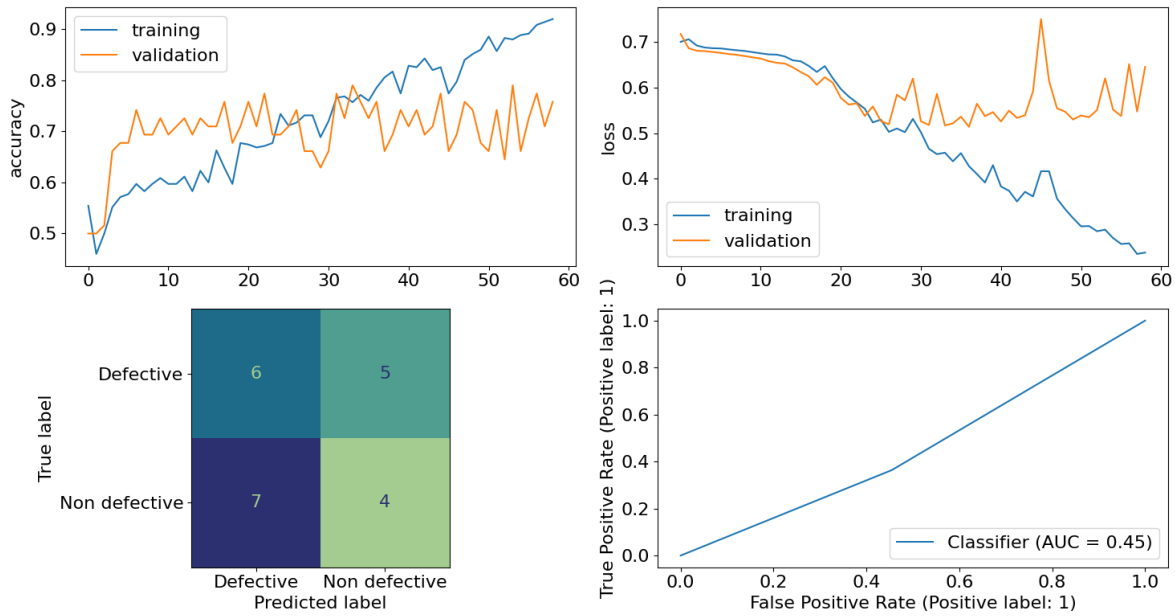


Figure 14: Fine-tuned ResNet50 metrics

6 Discussion

The results of the applied models are summarized in Table 5. Two major observations present, from one side the model performance highly depends on the selected learning rate, and on the other side the representativeness between test and training dataset needs to be checked more in detail. To address these two points further tuning of the models done by applying a random search algorithm on the learning rates and performing resampling to assess the accuracy rates better.

Model	Training		Validation		Test Accuracy
	Accuracy	Loss	Accuracy	Loss	
LeNet-5	59.14%	0.6858	59.68%	0.6884	59%
AlexNet	84.86%	0.3921	87.10%	0.3749	36%
VGG16	90.00%	0.2541	83.87%	0.5895	59%
Pretrained VGG16	97.71%	0.1840	91.94%	0.3871	55%
Pretrained ResNet50	50.29%	0.7003	67.74%	0.6771	45%
Fine-tuned ResNet50	57.71%	0.6741	70.97%	0.6764	45%

Table 5: Model results at optimal number of epochs

A deeper look into the test dataset reveals a possible root cause of the unrepresentativity issue. Figure 15 shows the image that is part of the test data and shows basically the same image to be classified. Furthermore this particular image is not an easily understandable case for the model, it is a very specific track failure with very limited comparable image in the training dataset. A misclassification of these images could lead to a test accuracy deviation of 13.64%. The test and validation accuracy is comparable only in case of the LeNet-5. The other models show much higher deviation between these two metrics than 13.64%, therefore this might be not the only reason.

6.1 Fine tuning of learning rates

Initially the learning rates were manually selected and the ReduceLROnPlateau callback was applied. Due to the fact that the size of the validation dataset is very limited, the classification of a single image leads to an change of 1.6% in terms of validation accuracy. In case the value of the learning rate is set to too low, several epochs need to be run to achieve a change in the validation accuracy score. Therefore the chance that the ReduceLROnPlateau reduces further the learning rate is very high. To avoid this behaviour, during fine tuning constant learning rate was applied. To realize the random search

Figure 15: Unrepresentative samples from test dataset

the modul keras_tuner was used. This allows easy hypertuning of the parameters, that is realised in ./hp_tuner.ipynb. The range of learning rate was set to cover an interval one magnitude higher and one magnitude lower compared to the manually selected learning rate. The number of epochs was set to be at least as much as the optimal epoch number in the manual case and overall 10 randomly selected values were run. In this way the accuracy rate of LeNet-5 and AlexNet was significantly improved, and the pretrained ResNet50 show some improvement, however in latter case the final result after fine-tuning of the model did not lead to any further increase in the validation accuracy. The results are presented in Table, 6, changes shown in red.

Model	Training		Validation		Test Accuracy
	Accuracy	Loss	Accuracy	Loss	
LeNet-5	60.00%	0.6842	62.90%	0.6892	59%
AlexNet	100.00%	0.0039	91.94%	0.5422	45%
VGG16	90.00%	0.2541	83.87%	0.5895	59%
Pretrained VGG16	99.71%	0.1436	91.94%	0.3830	23%
Pretrained ResNet50	52.86%	0.6882	74.19%	0.6734	45%
Fine-tuned ResNet50	75.71%	0.4569	79.03%	0.5169	45%

Table 6: Results after hypertuning the learning rates

In case of simpler neural networks (LeNet-5, AlexNet) the tuning of the learning rate proved to be an efficient approach. However in case of more deeper networks (VGG16, ResNet50) no significant improvement is achieved, except the pretrained ResNet50 prior to finetuning of the network parameters, that is considered as an intermediate status when building a CNN model. The reason behind might be that a more sophisticated approach on learning rates is used during manual setup. An adaptive setting might result in a situation when a local minimum can be approached much closer that in case of a constant learning rate as the number of epochs was limited during retraining the models with the hypertuned learning rate. All in all, a few bigger steps in the beginning followed by a sequential lowering of the learning rate might be a more useful approach for these networks. Due to the limited computational capacity this is not investigated further in the scope of this study.

6.2 Resampling

In order to achieve a reliable performance indicator in terms of accuracy, resampling is advised, in this particular case bootstrapping is used. A custom function is defined to resample the datasets including the training, validation and test dataset. The custom function shuffles the indices of the images, including training, validation and test dataset and segments the resulting list to the same three groups maintaining the same number of images for each dataset type. Afterwards augmentation applied for 25 images from the defective and non-defective training images respectively. This is repeated 10 times (except for the VGG16, where only 5 iterations done due to limited computational capacity). In this way several dataset can be created and separate models can be trained for all of them. This allows evaluating the single models on data that is not used for training at all, and furthermore allows to increase the representativeness of the training data to the test data. Once the single models are evaluated, the final metric is derived by taking the average of the test accuracies.

The following model states were considered for bootstrapping, the accuracy results for each dataset are shown on Figure 16 and quantified in Table 7.

1. LeNet−5 LeNet-5 with hypertuned learning rate
2. AlexNet AlexNet with hypertuned learning rate
3. VGG16 VGG16 with hypertuned learning rate
4. VGG16_p Pretrained VGG16 with hypertuned learning rate and without finetuning
5. ResNet50_p Pretrained ResNet50 with hypertuned learning rate and without finetuning

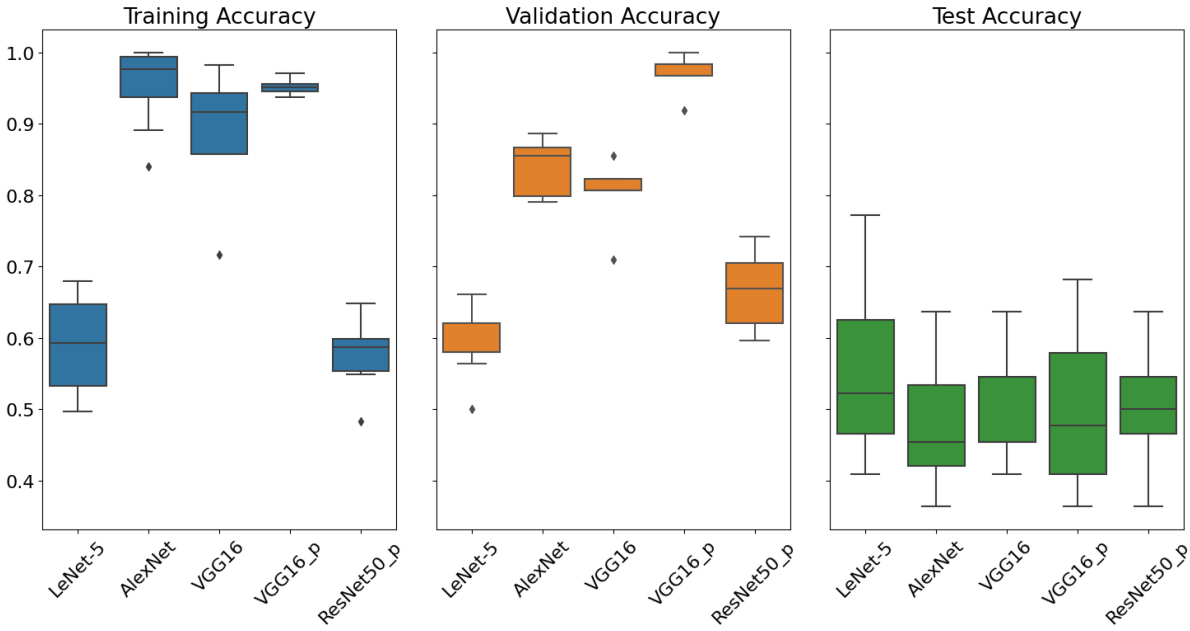


Figure 16: Results of bootstrapping

Model	Training		Validation		Test	
	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.
LeNet-5	58.77%	6.92%	59.03%	4.38%	55.00%	11.02%
AlexNet	95.54%	5.31%	83.87%	3.72%	47.73%	8.09%
VGG16	88.34%	10.35%	80.00%	5.42%	50.00%	9.09%
Pretrained VGG16	95.23%	0.98%	97.41%	2.18%	49.55%	11.43%
Pretrained ResNet50	57.62%	4.44%	66.61%	5.27%	50.91%	8.78%

Table 7: Accuracy results of bootstrapping

Unfortunately all model shows the same poor performance on the test dataset during bootstrapping. The accuracy fluctuates around 50% that is basically a random classifier. There could be several reasons for this behaviour. When the network is trained, the weights are optimized considering the loss function on the training data. In this step the network learns the training set, therefore it is expected to reach high accuracy on the training data. Then we select the model with highest validation accuracy, meaning we introduce an *artifical* learning on the validation set. This links the weights to the information stored in the validation images. Taking a look into the different metrics depcited on the Figures 9, 10, 11, 12, 13 or 14, it can be observed that the validation accuracy (and in some cases even the validation loss) vary greatly just in a few consecutive epochs. Selecting the maximum from these model instances is introducing a certain distortion to the results. Even in this case one would expect that the test dataset (especially in case of bootstrapping) shall vary around the same mean value, however this is not the case. Further analyzis shall be done to understand this behavior as proposals given in Section 7.

Figure 16 allows the comparison of the trained models. On the training accuracy the AlexNet and VGG16 models show good performance, indicating that these are capable of learning the features in the training set. The LeNet-5 seems to be not complex enough, while poor results is obtained with the

ResNet50 as well. Considering the model scaling described in [25], the number of images, number of classes shall be fitted to the depth, width and resolution of a CNN. Might be that ResNet50 is too complex for the binary classification based on this dataset size, having too much parameters in the convolutional layers to train. However in this case the pretrained model should deliver acceptable accuracy results, whilst this is not the case. The same behaviour is emphasized by the validation accuracy measures. The difference between the VGG16 models show that the pretrained model can be used as an efficient feature extractor, most probably increasing the dataset size would increase the accuracy of the pure VGG16 model as well.

7 Conclusion

During the study 5 types of convolutional neural networks were built to identify defective rail tracks on the given dataset. The networks include the LeNet-5, AlexNet, VGG16 and ResNet50, for the latter two pure models and transform learning was applied. The investigation is extended by hypertuning the learning rate to find the best fit on the validation dataset and bootstrapping was applied to obtain reliable test accuracy measure.

The results show that the AlexNet and VGG16 models achieved fairly good performance, from which the pretrained VGG16 proved to be the best one in terms of the validation accuracy, reaching up to an accuracy rate of 97.41% on average. However the performance on the test dataset show that all models behave as random classifiers. The reason for this is most probably the limited number of images in the dataset combined with several failure cases (rail cracks, missing elements, etc.) that all merged into a single defective class.

Revisiting our problem statement described in Section 3, the following answers can be given.

Q1 What kind of defects are represented in the images?

The images represent a high variety of defects in terms of type (rail crack, surface pitting, missing elements, etc.) as shown in Figure 5 and in terms of magnitude (from small defects up to major track failures).

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

Image manipulation is useful in cases when a fixed camera setup is given. Current dataset contains images taken from different perspectives, angles and distances, in some cases a close view on a rail surface is goven, in other cases a full track is captured. In this view image processing is limited to grayscale conversion and resizing to the correct input size and color mode of the particular neural network. In terms of machine learning approach, the convolutional neural networks show a certain solution to the given problem. The AlexNet and VGG16 successfully learns the training set and provides fairly good results on the validation dataset, however all attempts fail to reliably predict on the test dataset.

Q3 What accuracy rate can be achieved with the algorithm?

Manually trained network results a represented in Table 6, furthermore the consistency of each model is depicted on the bootstrapping results in Figure 16.

For further improvement several proposals can be formulated that might lead to more accurate and precise classification.

1. The hypertuning of the parameters can be extended to examine the effect of the batch size, dropout and batch normalization layers, optimizer, and so on. Besides increasing the type of parameters, the iteration number of the hypertuning can be increased to cover wider or more detailed subspace of the parameters.

2. Data augmentation can be included in the neural network. In this way all images become augmented and therefore the dataset size can be significantly increased. Further improvement is to extended the augmentation functions with cropping, patching and so on. Image rotation can be tuned by limiting the angle of rotation to a viable range, for example only $\pm 5^{\circ}$to model real life camera position together with $\pm 90^{\circ}$for horizontal or vertical position.

3. Initialization of the network weights is taken as random for the current study. Implementing a more established approach might lead to better performance.

4. Including further models, such as VGG19 or ResNet-34.

5. Deeper analysis of the ResNet network, finetuning of the parameters by unfreezing only selected convolutional blocks.

List of Figures

List of Tables

References

[1] Scarlett Liu, Quandong Wang, and Yiping Luo. "A review of applications of visual inspection technology based on image processing in the railway industry". In: *Transportation Safety and Environment* 1.3 (Dec. 2019), pp. 185–204. ISSN: 2631-4428. DOI: 10.1093/tse/tdz007. URL: https://academic.oup.com/tse/article/1/3/185/5714252 (visited on 12/24/2022).

[2] Xue Wang, Yike Tang, and Ping Cheng. "Machine-vision detection for rail-steel's surface flaws based on quantum neural network". In: *2008 7th World Congress on Intelligent Control and Automation*. 2008 7th World Congress on Intelligent Control and Automation. Chongqing, China: IEEE, 2008, pp. 5050–5055. ISBN: 978-1-4244-2113-8. DOI: 10.1109/WCICA.2008.4593749. URL: http://ieeexplore.ieee.org/document/4593749/ (visited on 12/25/2022).

[3] Ke Ma et al. "Texture classification for rail surface condition evaluation". In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). Lake Placid, NY: IEEE, Mar. 2016, pp. 1–9. ISBN: 978-1-5090-0641-0. DOI: 10.1109/WACV.2016.7477597. URL: https://ieeexplore.ieee.org/document/7477597/ (visited on 12/25/2022).

[4] Yunus Santur, Mehmet Karakose, and Erhan Akin. "A new rail inspection method based on deep learning using laser cameras". In: *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*. 2017 International Artificial Intelligence and Data Processing Symposium (IDAP). Malatya: IEEE, Sept. 2017, pp. 1–6. ISBN: 978-1-5386-1880-6. DOI: 10.1109/IDAP.2017.8090245. URL: http://ieeexplore.ieee.org/document/8090245/ (visited on 12/25/2022).

[5] Qingyong Li et al. "A cyber-enabled visual inspection system for rail corrugation". In: *Future Generation Computer Systems* 79 (Feb. 2018), pp. 374–382. ISSN: 0167-739X. DOI: 10.1016/j.future.2017.04.032. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X17307069 (visited on 12/25/2022).

[6] Ying Li et al. "Component-based track inspection using machine-vision technology". In: *Proceedings of the 1st ACM International Conference on Multimedia Retrieval - ICMR '11*. the 1st ACM International Conference. Trento, Italy: ACM Press, 2011, pp. 1–8. ISBN: 978-1-4503-0336-1. DOI: 10.1145/1991996.1992056. URL: http://portal.acm.org/citation.cfm?doid=1991996.1992056 (visited on 12/22/2022).

[7] Hoang Trinh et al. "Enhanced rail component detection and consolidation for rail track inspection". In: *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*. 2012 IEEE Workshop on Applications of Computer Vision (WACV). Breckenridge, CO, USA: IEEE, Jan. 2012, pp. 289–295. ISBN: 978-1-4673-0234-0 978-1-4673-0233-3 978-1-4673-0232-6. DOI: 10.1109/WACV.2012.6163021. URL: http://ieeexplore.ieee.org/document/6163021/ (visited on 12/25/2022).

[8] Ying Li et al. "Rail Component Detection, Optimization, and Assessment for Automatic Rail Track Inspection". In: *IEEE Transactions on Intelligent Transportation Systems* 15.2 (Apr. 2014), pp. 760–770. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2013.2287155. URL: http://ieeexplore.ieee.org/document/6662397/ (visited on 12/25/2022).

[9] P.L Mazzeo et al. "Visual recognition of fastening bolts for railroad maintenance". In: *Pattern Recognition Letters* 25.6 (Apr. 2004), pp. 669–677. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2004.01.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167865504000194 (visited on 12/25/2022).

[10] Francescomaria Marino et al. "A Real-Time Visual Inspection System for Railway Maintenance: Automatic Hexagonal-Headed Bolts Detection". In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 37.3 (May 2007), pp. 418–428. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.893278. URL: http://ieeexplore.ieee.org/document/4154946/ (visited on 12/25/2022).

[11] P. De Ruvo et al. "A GPU-based vision system for real time detection of fastening elements in railway inspection". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. 2009 16th IEEE International Conference on Image Processing ICIP 2009. Cairo, Egypt: IEEE, Nov. 2009, pp. 2333–2336. ISBN: 978-1-4244-5653-6. DOI: 10.1109/ICIP.2009.5414438. URL: http://ieeexplore.ieee.org/document/5414438/ (visited on 12/25/2022).

[12] Rubayat Ahmed Khan, Samiul Islam, and Rubel Biswas. "Automatic detection of defective rail anchors". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC). Qingdao, China: IEEE, Oct. 2014, pp. 1583–1588. ISBN: 978-1-4799-6078-1. DOI: 10.1109/ITSC.2014.6957919. URL: http://ieeexplore.ieee.org/document/6957919/ (visited on 12/25/2022).

[13] Xavier Gibert, Vishal M. Patel, and Rama Chellappa. "Robust Fastener Detection for Autonomous Visual Railway Track Inspection". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. 2015 IEEE Winter Conference on Applications of Computer Vision (WACV). Waikoloa, HI, USA: IEEE, Jan. 2015, pp. 694–701. ISBN: 978-1-4799-6683-7. DOI: 10.1109/WACV.2015.98. URL: http://ieeexplore.ieee.org/document/7045952/ (visited on 12/25/2022).

[14] Xavier Gibert, Vishal M. Patel, and Rama Chellappa. "Material classification and semantic segmentation of railway track images with deep convolutional neural networks". In: *2015 IEEE International Conference on Image Processing (ICIP)*. 2015 IEEE International Conference on Image Processing (ICIP). Quebec City, QC, Canada: IEEE, Sept. 2015, pp. 621–625. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7350873. URL: http://ieeexplore.ieee.org/document/7350873/ (visited on 12/24/2022).

[15] Caglar Aytekin et al. "Railway Fastener Inspection by Real-Time Machine Vision". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.7 (July 2015), pp. 1101–1107. ISSN: 2168-2216, 2168-2232. DOI: 10.1109/TSMC.2014.2388435. URL: http://ieeexplore.ieee.org/document/7015636/ (visited on 12/25/2022).

[16] Yiqi Xia, Fengying Xie, and Zhiguo Jiang. "Broken Railway Fastener Detection Based on Adaboost Algorithm". In: *2010 International Conference on Optoelectronics and Image Processing*. 2010 International Conference on Optoelectronics and Image Processing (ICOIP). Haiko, Hainan, China: IEEE, Nov. 2010, pp. 313–316. ISBN: 978-1-4244-8683-0. DOI: 10.1109/ICOIP.2010.303. URL: http://ieeexplore.ieee.org/document/5662900/ (visited on 12/22/2022).

[17] Maneesh Kumar M. et al. "A Survey on Crack Detection Technique in Railway Track". In: *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*. 2018 Conference on Emerging Devices and Smart Systems (ICEDSS). Tiruchengode: IEEE, Mar. 2018, pp. 269–272. ISBN: 978-1-5386-3479-0. DOI: 10.1109/ICEDSS.2018.8544319. URL: https://ieeexplore.ieee.org/document/8544319/ (visited on 12/22/2022).

[18] Mehmet Karakose et al. "A New Approach for Condition Monitoring and Detection of Rail Components and Rail Track in Railway*:" in: *International Journal of Computational Intelligence Systems* 11.1 (2018), p. 830. ISSN: 1875-6883. DOI: 10.2991/ijcis.11.1.63. URL: https://www.atlantis-press.com/article/25892537 (visited on 12/24/2022).

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. (Visited on 12/22/2022).

[20] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. *Neural Architecture Search: A Survey*. Tech. rep. arXiv:1808.05377 [cs, stat] type: article. arXiv, Apr. 2019. DOI: 10.48550/arXiv.1808.05377. arXiv: 1808.05377.

[21] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, eds. *Automated Machine Learning: Methods, Systems, Challenges*. en. The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing, 2019. ISBN: 9783030053178 9783030053185. DOI: 10.1007/978-3-030-05318-5. (Visited on 01/05/2023).

[22] Shahrzad Faghih-Roohi et al. "Deep convolutional neural networks for detection of rail surface defects". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016 International Joint Conference on Neural Networks (IJCNN). Vancouver, BC, Canada: IEEE, July 2016, pp. 2584–2589. ISBN: 978-1-5090-0620-5. DOI: 10.1109/IJCNN.2016.7727522. URL: http://ieeexplore.ieee.org/document/7727522/ (visited on 12/24/2022).

[23] Lidan Shang et al. "Detection of rail surface defects based on CNN image recognition and classification". In: *2018 20th International Conference on Advanced Communication Technology (ICACT)*. 2018 20th International Conference on Advanced Communications Technology (ICACT). Chuncheon-si Gangwon-do, Korea (South): IEEE, Feb. 2018, pp. 45–51. ISBN: 979-11-88428-01-4. DOI: 10.23919/ICACT.2018.8323642. URL: https://ieeexplore.ieee.org/document/8323642/ (visited on 12/24/2022).

[24] R. Thendral and A. Ranjeeth. "Computer Vision System for Railway Track Crack Detection using Deep Learning Neural Network". In: *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. 2021 3rd International Conference on Signal Processing and Communication (ICPSC). Coimbatore, India: IEEE, May 2021, pp. 193–196. ISBN: 978-1-66542-864-4. DOI: 10.1109/ICSPC51351.2021.9451771. URL: https://ieeexplore.ieee.org/document/9451771/ (visited on 12/22/2022).

[25] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Tech. rep. arXiv:1905.11946 [cs, stat] type: article. arXiv, Sept. 2020. DOI: 10.48550/arXiv.1905.11946. arXiv: 1905.11946.

[26] Chandeep Sharma. "Comparison of CNN and Pre-trained models: A Study". In: Apr. 2022.

[27] Shreetha Bhat et al. "Classification of Rail Track Crack using CNN with Pre-Trained VGG16 Model". In: *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*. Apr. 2022, pp. 1–6. DOI: 10.1109/ICDCECE53908.2022.9793318.

[28] *Railway Track Fault Detection | Dataset2(Fastener)*. en. URL: https://www.kaggle.com/datasets/ashikadnan/railway-track-fault-detection-dataset2fastener (visited on 01/06/2023).

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

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

[31] Raimi Karim. *Illustrated: 10 CNN Architectures*. en. Dec. 2022. URL: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d (visited on 01/06/2023).

[32] *Keras Applications*. URL: https://keras.io/api/applications/.

[33] *Home*. en-US. URL: https://opencv.org/ (visited on 01/06/2023).