Available online at www.eccomasproceedia.org
Eccomas Proceedia COMPDYN (2023) 1303-1320

ECCOMAS
Proceedia

COMPDYN 2023
9th ECCOMAS Thematic Conference on
Computational Methods in Structural Dynamics and Earthquake Engineering
M. Papadrakakis, M. Fragiadakis (eds.)
Athens, Greece, 12-14 June 2023

A MACHINE LEARNING LEVERAGED AND MINIMAL HARDWARE-BASED TRAIN AND LOAD RECOGNITION SYSTEM

J. Brötzmann[1], C.-D. Thiele[1], M. M. Rupp[2], M. Xing[1], and U. Rüppel[1]

[1] Institute of Numerical Methods and Informatics in Civil Engineering (IIB)
Technical University of Darmstadt, Germany
broetzmann@iib.tu-darmstadt.de

[2] Institute of Structural Mechanics and Design (ISMD)
Technical University of Darmstadt, Germany
rupp@ismd.tu-darmstadt.de

Abstract

In the field of structural health monitoring, it is very important to quantify the impacting loads to better estimate and track the remaining service life of the structure. For railway bridges, it is therefore necessary to identify the passing trains. In current research projects, such as ZEKISS (www.zekiss.de), this is often done using sensors, e.g., acceleration sensors. The collected data is then analysed with respect to the impacting loads and possible changes in the supporting structure. However, the determination of the passing train is complex and the instrumentation of bridges with multiple sensors and a corresponding edge device is very expensive and time consuming. In addition, due to the large number of ageing bridges, an economical solution is needed. For those reasons, it is investigated how passing trains can be identified using minimal hardware at minimal costs.
The following concept is proposed and tested. The first step is to classify different types of trains as freight or passenger trains. This can be done using image recognition algorithms. The hardware used should be as simple and inexpensive as possible so that it can be replicated for many bridges. Therefore, a single board computer with a camera module for image acquisition and expandability for other components is used. In addition, the hardware is extended to include lasers or photo sensors to classify train and wagon types using light barriers. This can be done by the length and bogie of each wagon and using databases to match the corresponding train. Distance and time can also be used to calculate the speed of the train, which is useful for dynamic studies of the bridge and calibration of the sensors. With all this together, a train and load recognition system is created.

Keywords: minimal hardware, machine learning, predictive maintenance

ISSN:2623-3347 © 2023 The Authors. Published by Eccomas Proceedia.
Peer-review under responsibility of the organizing committee of COMPDYN 2023.
doi: 10.7712/120123.10478.20409

1 INTRODUCTION

1.1 Motivation

In the rail sector, Germany has around 33,400 kilometres of track with more than 25,000 bridges [1]. The average bridge is about 85 years old and over 11,000 bridges are older than 100 years [2]. Thus, Germany must invest a lot of money in the rail sector in the next decades. The agreement *LuFV III* provides for the complete or partial renewal of a total of 2,000 bridges by 2029 at a cost of around 86 billion euros [3]. This means that many bridges will still need to be modernized after this period and the investment backlog will continue to grow. For this reason, the other existing bridges must be closely monitored. This will ensure that they can be used for as long as possible until they can be renewed themselves. To ensure an effective and, if possible, predictive monitoring, the digitalisation of monitoring and forecasting concepts is crucial. The application of the latest technologies, which can help to maintain or extend the service life of bridges, is of great economic importance in this sector [4].

Artificial Intelligence (AI) is one of the latest technologies that is disrupting a wide range of industries [5]. AI-supported predictive maintenance is a promising solution for monitoring a great number of bridges. This is the subject of ongoing research such as the research project ZEKISS, in which the authors are intensively engaged [6]. ZEKISS is a German acronym, meaning "condition assessment of railway bridges and vehicles with AI methods for the evaluation of sensor data and structural dynamic models". Within this project, bridges in Germany are instrumented with sensors, mainly acceleration sensors, to collect data about the structure. This data is then processed and analysed with the help of AI to monitor the bridge and detect defects. AI respectively machine learning is the study and development of algorithms that can learn from the data collected. With the aim of predictive infrastructure diagnostics, the trained algorithms can make conclusions and predictions based on the data. For example, the algorithm can decide that the data set shows a defect on the bridge [7]. For humans this task is complex and needs a lot of time whereas predictive maintenance algorithms can deal with this quite good. Predictive maintenance approaches are designed to estimate the condition of equipment to decide when maintenance should be performed, reducing costs, and increasing productivity by performing tasks only when they are needed [8].

Nevertheless, instrumenting bridges with sensors is a time consuming and very expensive undertaking. Experiences within the research project ZEKISS are showing this. That is why this paper proposes another way of collecting data and determining the condition of the structure. To better diagnose the structure, it is very important to know the loads operating on it. If the loads are recorded continuously, a better analysis of their impacts on the structure can be done. By knowing respectively calculating the effect of the loads, a better prediction of the remaining service life is possible [9]. At present, among other methods, costly and time-consuming axle load measuring points are used to directly determine the impacting loads [10], [11]. But these points cannot be installed near every bridge. Or it would be very expensive to do so on the large number of ageing bridges. A cost-effective approach is therefore needed. In addition, because of the distributed nature of the railway operators, a central database with details of passing trains, such as loads and time-location information, is also required.

Here, the single board computer (SBC) comes to mind for applications that include machine learning and connectable sensors or IoT (Internet of Things) devices. Low-cost yet powerful hardware is becoming increasingly available, in part due to the mass production of these products. And with the high accessibility and applicability of these devices because of the included operating system, even non-technical users can utilize them for their applications [13]. The high sales figures of the Raspberry Pi, for example, prove the success of SBCs [14].

1.2 Research goals, contributions and methodology

The main goal of this paper is to help with predictive maintenance by collecting as much data as possible about trains passing through. Economy and convenience need to be considered, as the data collection system proposed should be repeatable for many bridges. SBC and machine learning help to reduce costs and make projects easy to implement. The SBC can be equipped with a camera module for image capture and can be expanded with other components such as lasers. Machine learning algorithms can analyse the collected images of the trains to determine the type. The light barriers of the lasers can help with this task by identifying the length or bogie of each wagon. Using publicly available information on train types, the initial axle loads can be inferred. This data is then available for further structural analysis as part of the predictive maintenance. The following information may be collected as a contribution to predictive maintenance:

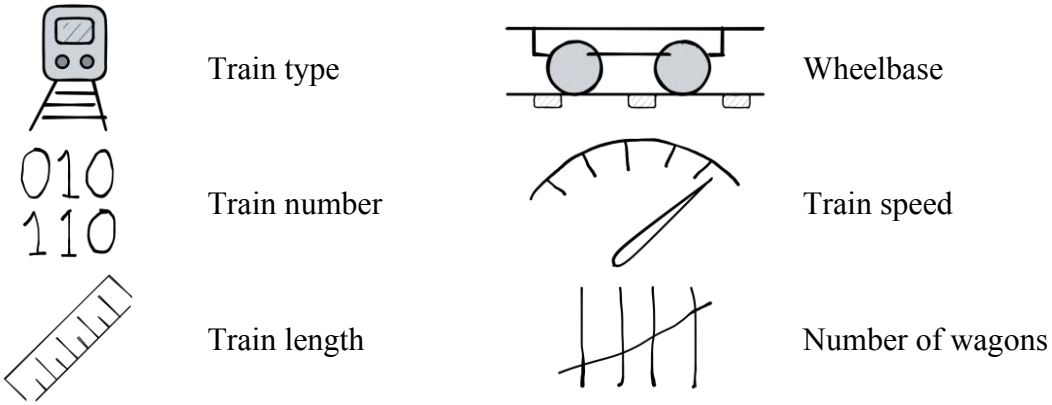


Figure 1: Collectable information.

Since the approach here is being developed and tested in a practical way, the main research methodology of this paper is applied research. However, descriptive research is also used, as the fundamental topics covered here are researched and described to gain a better understanding of the solution being developed. In general, building a new approach based on an understanding of current approaches is at the heart of this paper. However, not only the theoretical development, but also the testing under real conditions is carried out.

1.3 Research challenges

The aim is to collect as much data as possible with a low budget, while ensuring the accessibility of the developed system. For the machine learning part there are no own datasets available. Finding enough data is therefore a challenge. Another challenge is to measure the speed of the train with inexpensive and usable sensors and algorithms that go with them. For computer vision activities, the capabilities of a low-cost camera need to be intensively tested. The integration of software and hardware concepts and optimisation are also major challenges.

1.4 Structure of the paper

The structure of this paper is as follows. The research background is described in chapter two. Based on that, an own concept and methodology is developed. With the concept in mind, the project is implemented in chapter four. In the next chapter, some short notes about the deployment of the system can be found. A conclusion is given in chapter six. Finally, a discussion about the results and the approach as well as the future work is presented.

2 RESEARCH BACKGROUND

This chapter presents the ongoing research and relevant background information on the topics covered here. As there are several topics involved, the aim is to keep each part short.

2.1 Predictive maintenance

Predictive maintenance has become an increasingly important research topic in the field of structural health monitoring. This is due to the greater availability of continuous monitoring data and the increased computing power of the hardware used. More data is available because data collection has become easier with the help of IoTs and sensing technologies. As a result, maintenance can be done data-driven and predictive, rather than reactive and problem-oriented. Machine learning also helps with this task [15]. This method of maintenance reduces costs by increasing availability and performing repairs only when they are needed [16].

There are still challenges with this method. Implementing and training deep learning algorithms to detect anomalies in structures like bridges is a manual and individual task for each object. Initially, anomalies must be defined. That is why a generalisation is still needed [17].

2.2 Single board computers

A basic SBC consists of a single personal computer board with memory, a processor, and some sort of input/output to interact with it. Many newer SBCs can compete with modern PCs and tablets. Rapid commercial adoption and success reduced the prices dramatically, while development has continued. The support of a large open-source community is accelerating this process. The cost of peripheral products such as sensors and cameras has also fallen. As a result, more and more applications and products are emerging from this development [18].

The evolution of today's AI, IoT and 5G in combination with the SBC has brought infinite possibilities to edge computing. E.g., NVIDIA is releasing SBCs with embedded software development kits so that the deployment of trained algorithms is much easier [19]. This is why applications like the one proposed in this paper are being researched more and more, as there are many areas in which they can be used, such as agriculture, construction sites or logistics.

2.3 Machine learning and deep learning

Artificial intelligence is one of the latest technologies to disrupt many industries and is a large field of research in its own. For this reason, this topic will be kept short. Only the basics of the necessary tasks, i.e., image and text recognition, will be discussed here. For further information, e.g., on differences between AI, machine learning, and deep learning, please refer to the general literature such as [20] or [21].

2.3.1 Image recognition

Determining what kind of train is contained in a captured image is a classic problem in computer vision. Computer vision is a field that does not just try to process and analyse pictures, but also to understand them. If an object in an image is recognised, it can also be identified, e.g., what type of train (passenger or freight) it is [22].

At the moment, the best algorithms for this kind of task are based on convolutional neural networks (CNN). The *ImageNet Large Scale Visual Recognition Challenge* provides an example of their capabilities [23]. Based on artificial neural networks, CNNs are part of deep learning algorithms [20]. In this paper, the *MobileNet* architecture is used. It is an efficient model for both mobile and embedded image processing applications, as shown here with the SBC [24].

2.3.2 Optical character recognition

In recent years, Optical Character Recognition (OCR) has been one of the most popular research fields in pattern recognition. It is a subject of active research in industry and academia due to its huge application potential. OCR is a technology for converting handwritten, typed, or printed text characters into machine readable text. So, this technique can be used to translate the train numbers into text that the computer is able to process. Image processing is recognised as a superset of OCR [25]. Therefore, it is related to the content described under 2.3.1.

After a pre-processing, the first step is to detect text in the image. The deep learning model *EAST* can be used for this, as it is a quick and accurate method for text detection in natural scenes. *EAST* stands for Efficient and Accuracy Scene Text and is based on a fully convolutional network [27]. The referenced literature provides additional information. Once the text has been detected, the *Tesseract* model is used to recognise the text to get an output string. *Tesseract* is an open-source OCR engine. It was first developed at HP between 1984 and 1994 [28]. In the newer version, *Tesseract* is using a neural network to improve the text recognition [29]. Figure 2 summarizes the above-described workflow.

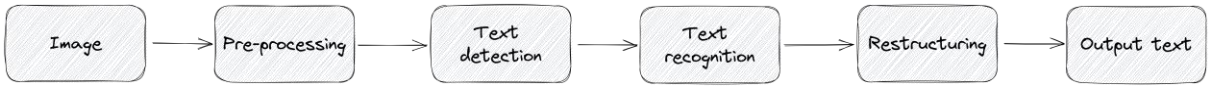


Figure 2: Process of OCR [26].

2.4 Sensor technology

There are many different types of sensors. They can detect the characteristic quantity of a measurement object and convert this quantity into an understandable signal. Sensors and sensor systems perform their function through the interplay of sensor structure, manufacturing technology and signal processing algorithms [30]. In recent years, there have been significant advances in sensor technology that have helped to make sensors lighter, smaller and more power efficient. Sensors with these characteristics can be easily deployed at a lower cost. Sensor technology is widely used in the railway industry. For the different measurement tasks different sensors are useful. An overview can be found in [31].

2.5 Understanding the railway system

To analyse the collected data as accurately as possible, e.g., to identify the train type, an understanding of the German railway system is necessary.

2.5.1 Series schema

On the first of January 1968, the model number scheme was changed to a seven-digit number. This made the scheme computer readable. Table 1 provides an example. When a train is photographed, this number can be used to identify the locomotive. However, a text recognition algorithm is required to extract the number from the photograph. It is also necessary to analyse where the serial numbers are located on the train. They are usually printed on the front of the train. The quality of the photo must also be good enough to read the number with text recognition.

114 036-7	612 078-6
Electric locomotive of the DB series 114	Diesel locomotive of the DB series 612

Table 1: Examples of the series schema of the Deutsche Bahn (DB).

2.5.2 Bogie

Another way of identifying the train and wagon types is by the bogies. Identification is possible by using lasers to measure the distances and a database to match the corresponding type. Figure 3 provides an example. Most of the bogies used today are two-axle, but there are also those with one-axle, three-axle, and more axle bogies [32].

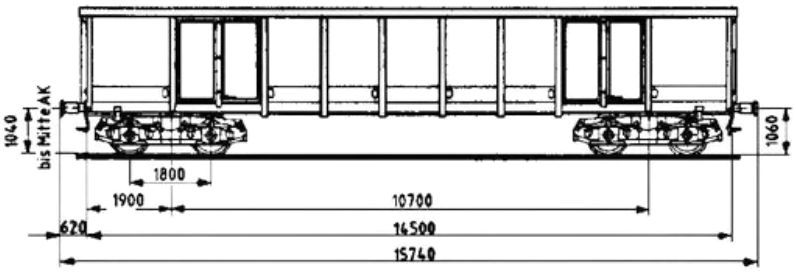


Figure 3: Example of the distance between the bogies of a Deutsche Bahn wagon [32].

2.6 Existing solutions and ongoing research in wayside monitoring

Within the Deutsche Bahn ecosystem, there are numerous analogue solutions that can help in the development of the here proposed method. The DB Systel, an information and telecommunication service supplier for Deutsche Bahn [33], is continuously training image recognition algorithms with their collected data. The resulting *Visual Recognition Service* in the *DB IoT Cloud* can be used in a wide variety of ways. E.g., it is possible to detect snow to automatically instruct the snow clearance services or recognize graffiti on trains to clean them in a more planned and targeted manner [34], [35]. The two use cases show that using cameras combined with image recognition is a feasible and efficient technique to assist the railway in early and optimal fault detection, including predictive maintenance capabilities.

There are other sensors used in the railway industry. For example, Camlin Ltd has developed a pantograph system that measures the train speed and direction using laser telemeters positioned within the acquisition enclosure. This solution has many similarities to the here proposed method, as it also uses cameras combined with machine learning to identify train types [36]. However, since it is a rather complex and large system that needs to be built permanently, the cost is likely to be much higher than the solution developed here. In addition, the Camlin system is not mobile and therefore cannot be used in different locations. The AI tasks require a good data set, which is not given by the companies. The programs also need a platform to run on and a data connection to receive the data collected. An SBC is used here, which enables a complete embedded solution that does not necessarily require an Internet connection to function.

Other measurement solutions, as briefly described in the introduction, use axle load measuring points to determine loads directly. These do not fall into the category of wayside monitoring since it is a direct measurement system. Nevertheless, it is an important part of research. [9], [10], [11], [12] give a good overview regarding different methods in this area.

3 CONCEPT

In the following chapter, a concept for an economical and mobile data collection system will be developed using the research background, related work and solutions mentioned above.

3.1 General workflow

A general workflow is provided that largely incorporates the performed analysis.

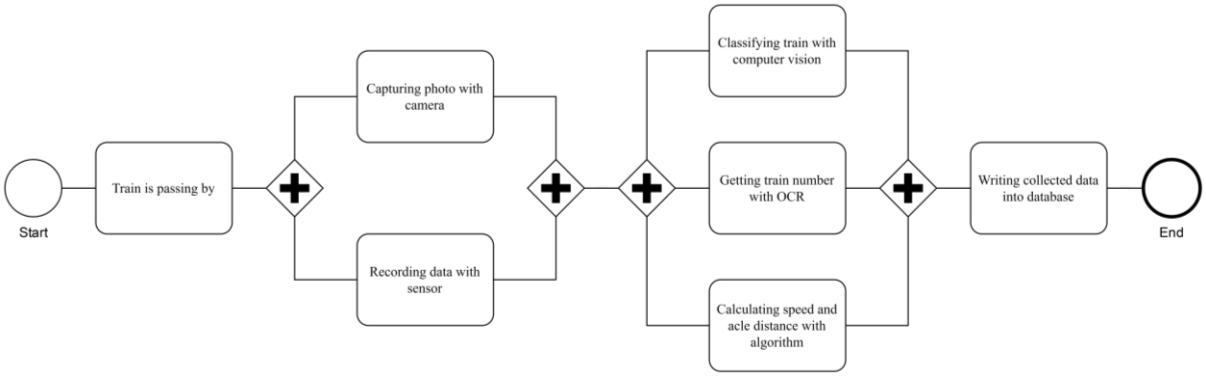


Figure 4: Workflow of the data collection and analysis system.

As visible in **Figure 4**, the procedure begins when a train is passing by. The system detects a motion, the camera module takes a picture, and the light barriers start recording. Time information is stored as well. In the third step, the data gets analysed. The different AI-algorithms try to determine the train type and the number of the vehicle. In addition, the calculation of the speed and the axis distances using the sensor information is carried out. After that, the determined data gets stored into an SQL database and the pictures stored on the SBC. The big advantages of an SQL database are that it is easy to use, easy to set up and open source, so you can use it without any costs. The entity relationship model (ERM) of the implemented database is given in Figure 5.

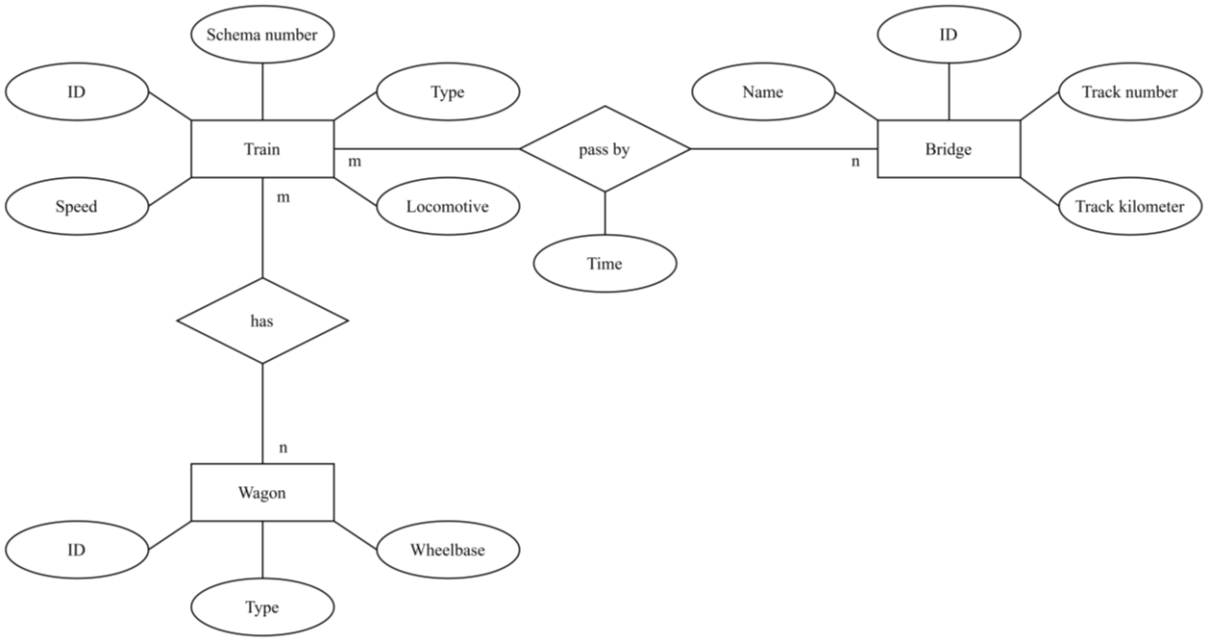


Figure 5: ERM of the database.

3.2 Hardware

3.2.1 Single board computer

The SBC, which is at the heart of this project, must be selected carefully to meet the project's requirements. For this project, a camera connection and a sensor interface are critical. In addition, the use of deep learning models, such as image recognition or OCR algorithms, also requires graphics processing power. In simple terms, GPU or NPU are suitable. The Arduino microcontroller is therefore ruled out since it does not have a GPU or a camera interface [37]. The following table compares the Nvidia Jetson Nano and the Raspberry Pi 4.

SBC	Nvidia Jetson Nano	Raspberry Pi 4 (4 GB)
I/O pins	40-pin GPIO	40-pin GPIO
Camera	MIPI CSI port	MIPI CSI port
CPU	Quad core ARM Cortex-A57 64-bit @ 1.43 GHz	Quad core ARM Cortex-A72 64-bit @ 1.50 GHz
GPU	Maxwell 128 cores @ 921 MHz	Broadcom VideoCore VI
Memory	4 GB LPDDR4	4 GB LPDDR4
Ethernet	Gigabit Ethernet / Wi-Fi	Gigabit Ethernet / Wi-Fi / Bluetooth
Price	$99	$55

Table 2: Comparison of the SBCs Nvidia Jetson Nano [38] and Raspberry Pi 4 [39].

Table 2 shows the comparison between the Jetson Nano and the Raspberry Pi. The Jetson Nano is equipped with a 128-core Maxwell GPU running at 921 MHz. In a side-by-side comparison, the Jetson Nano GPU is far superior to the Raspberry Pi. However, the Raspberry Pi's Cortex-A72 is a generation newer than the A57 in the Jetson Nano. This CPU has faster performance and a faster clock speed. But for deep learning, it may not provide enough performance benefits. Considering the significant number of machine learning applications and future development needs for this project, the more expensive Nvidia Jetson Nano is chosen.

3.2.2 Camera

Having chosen the Jetson Nano as the SBC, it offers two possible ways of connecting the camera: USB 3.0 and the MIPI CSI-2 port. MIPI has a high bandwidth of 6 GB/s and is faster than USB 3.0. It includes four picture data lanes, each capable of 1.5 GB/s. CSI-2 also uses fewer resources from the CPU thanks to its multi-core processors. The disadvantage of its limited cable length can be overlooked as the system will be an all-in-one solution [40]. With the MIPI CSI-2 port as the selected camera port, the Waveshare IMX219-160 IR-CUT camera module was chosen, shown in Figure 6. It contains 800 megapixels with an IMX219 sensor and a 160-degree field of view (160 FOV). It also has infrared (IR) night vision capabilities, which is necessary as the system needs to be able to capture photos around the clock. With a price tag of $28, the camera is still reasonably priced [41].

Figure 6: Camera connected to the Jetson Nano [41].

The image quality of the camera is important for computer vision tasks. It is important to check the quality under different lighting conditions. The implementation will check the practical implications of this camera, e.g., night vision, weather adaptability or ease of use.

3.2.3 Sensor

As discussed in the previous chapters, the aim is to select a sensor that is suitable in terms of price and installation. In form of a wayside monitoring, it is possible to install sensors close to the tracks. The *IR Break Beam Sensor* was found to meet those requirements. To detect movement, infrared break-beam sensors are an easy solution. They have an emitter that sends out a beam of infrared light. This is invisible to human eyes. Then there is a receiver on the other side that is sensitive to this light. If something passes between the two, even if it is not visible to IR, the receiver is separated from the emitter. The receiver informs the connected device. As for price, a pair of sensors costs just $6. More interestingly, compared to sensors costing over $100, the response time is only slightly increased from 1 to 2 milliseconds, with no impact on their usefulness. The sensing distance is up to 50cm / 20", so they can be placed on either side of a track [42].

To determine the wheelbase, the bogie types must first be established. For that, an algorithm must be created to perform this operation based on the acquired data. The information that can be obtained from the sensor is the signal adaptation, i.e., the interruption of the photoelectric door. Moreover, the distance between the two photoelectric gates is also important for the algorithm.

3.3 Deep learning algorithms

3.3.2 Classification of trains

Data acquisition is crucial for image classification tasks and for the initial development and training of the algorithm. There are three ways of doing this: using an existing dataset, taking photos to create a dataset, or using a scraper. In the scope of this project, photos are taken after the deployment of the data collection system. Thus, to initially train the algorithm, a web scraper is used to obtain as many images of trains as possible. The number of classes must be determined by the dataset. Two classes, passenger, or freight are more practical.

TensorFlow in combination with *OpenCV* is used as the training and deployment platform. It is important to be able to train and deploy models on different devices. Therefore, the algorithms are trained on a PC and deployed directly to an edge device such as an SBC. It is also necessary to test different models and different learning approaches, comparing training time and model accuracy.

3.3.3 Capture of train number

To not only classify the trains, but to capture as much data as possible, OCR algorithms are used to identify more information. As described in chapter two, trains can be identified by their series scheme number. The number is usually printed on the front of the locomotive. However, these numbers can also be scattered around the wagon at different angles and in different fonts. Therefore, the practical application will show how effective the capturing of train numbers works.

4 IMPLEMENTATION

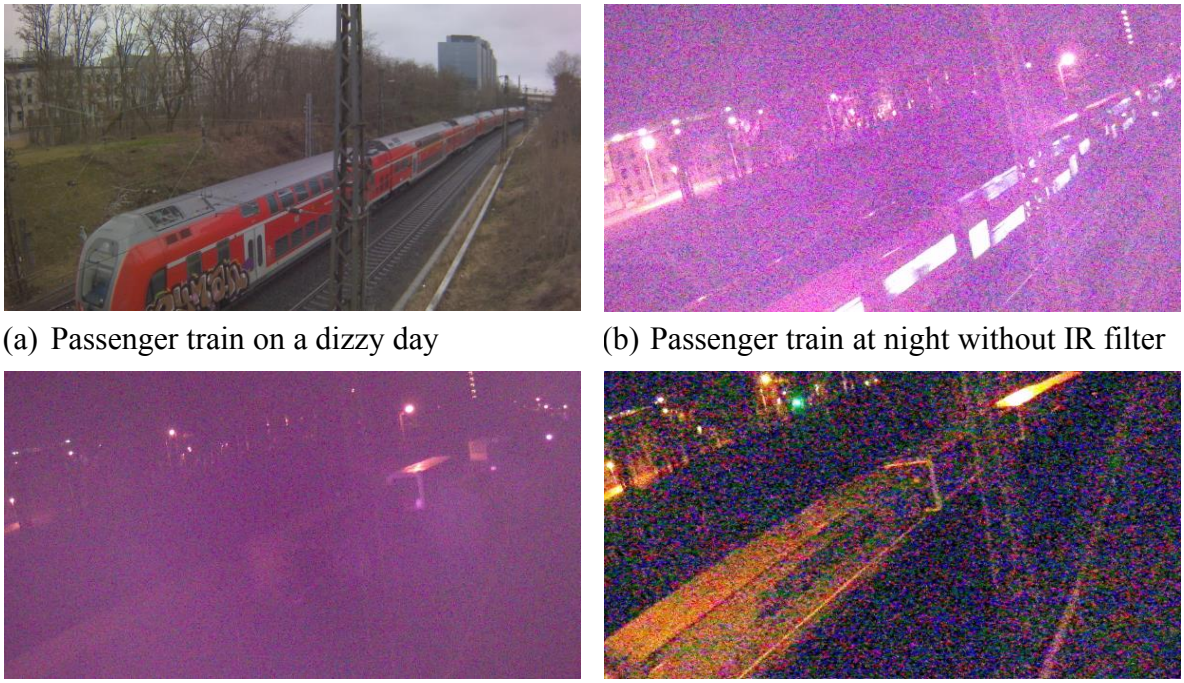
The concept developed in the previous chapter is implemented in this chapter.

4.1 Hardware configuration and testing

Before being able to use the Jetson Nano, an initial setup is necessary. Nvidia provides a setup guide for that. The Jetson Nano Developer Kit, that is being installed during the setup, is offering many useful tools. It is compatible with common sensors and peripherals, e.g., the

Adafruit sensor chosen here. It also supports popular AI frameworks such as *TensorFlow* and *PyTorch*. Jetson Nano can run neural networks in parallel to process data more efficiently [43]. The Jetson Nano Developer Kit includes the Jetson Linux Driver Package so that all further *JetPack* components and required Python libraries can be installed. The installation process is unobtrusive. However, as the Jetson Nano was chosen for its superior GPU, this advantage should be exploited. Therefore, *OpenCV*, the library used for the computer vision tasks, is installed using CUDA. CUDA is NVIDIA's set of libraries for working with their GPUs. This allows image recognition tasks to run faster on the GPU than on the CPU [44].

Taking and saving pictures with the camera also requires some initial setup to work. Additional libraries, such as *JetCam*, are essential. Some basic day and night tests were carried out with the camera, as shown in Figure 7.



(a) Passenger train on a dizzy day

(b) Passenger train at night without IR filter

(c) Freight train at night without IR filter

(d) Freight train at night with IR filter

Figure 7: Performance tests with the camera.

It is clear from Figure 7 (a) that the camera can capture images perfectly in daylight, but its performance at night is unsatisfactory, as shown in Figure 7 (b), (c) and (d). The camera includes infrared LEDs and IR-CUT, which eliminates the reddish problem of normal IR cameras. However, this still doesn't give the desired results. Also, due to the high speed of trains, shutter speeds of at least 1/500s to 1/1000s are required. Shorter shutter speeds do not work well, and the images are too dark due to the poor image quality of this camera. As a replacement, tests were carried out using a mobile phone camera with simple controls. Shutter speeds for sharp photos can be found by estimating the speed with a stopwatch. With trains travelling at 100km/h and above, most photos are taken between 1/400s and 1/500s. However, the ISO is set quite high at 400. For ICE and other high-speed trains travelling at 200km/h, shutter speeds need to be higher than 1/1000s and ISO higher than 800. Overall, even with night vision, the camera only delivers usable photos in good lighting conditions, so the scope of this project is limited to daylight. The camera does not perform well at high shutter speeds and can only produce images that meet the requirements at low shutter speeds.

4.2 Data acquisition

It takes a lot of effort to create a dataset of self-recorded images of trains that is large enough to develop and train deep learning algorithms. Almost a hundred pictures were taken over three days. However, this is still a small number, and the variety is low caused by a low number of possible different locations. Furthermore, it is not really necessary to create an own dataset. There is plenty of data available on the internet. Therefore, the more effective strategy of a web scraper is chosen. A scraper can gather and index group of web images. The website *Bahnbilder* from Thomas Wendt [45] is perfectly suited for this purpose. It provides many photographs of trains, which are well separated by their source of energy (e.g., electric or diesel), their form of use, their location of use, etc. More importantly, they even contain the first three digits of the series scheme number. So, the gathered dataset can be used for the implementation of text detection and recognition as well.

The initial aim of the train classification was to separate the pictures into freight and passenger trains. It must be ensured that the dataset is balanced so the algorithm is able to tackle the task. Since the images are well categorised, they can be easily scraped by category.

The next step is to develop the actual web scraper. A delay between requests is used to ensure that the scraper doesn't hit the site too hard. At the end of the process, each class, passenger and freight, contains a thousand photos. Not only the locomotive, but also the carriages or freight wagons were included. The reason for this is that locomotives are versatile.

4.3 Model training for train classification

First, the data set is divided into training (80%) and testing (20%). The architecture of the CNN model includes two convolutional layers, two pooling layers and two dense layers. As shown in Figure 8, the training and validation curves have opposite trends, so over-fitting occurs. Several ways were tried to address the overfitting problem, such as adding regularisation and reducing the complexity of the architecture. However, these adjustments were not sufficient, although the accuracy rate improved.

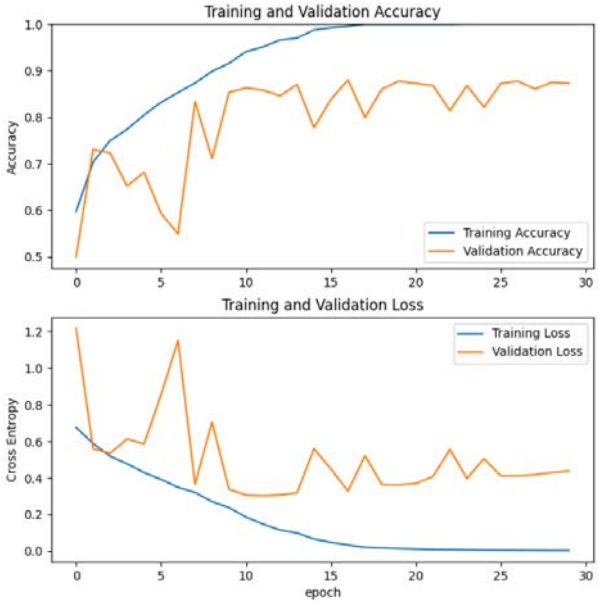


Figure 8: Training and validation of the CNN

Next, *MobileNet* is used and adjusted in the form that the base model is frozen. Only the final pooling and fully connected layers are trained. The weights from *ImageNet* and the Adam optimiser are also used. The Adam optimiser is simple and gives good results. With 30 to 50 epochs, the results are shown in Table 3. It took 200s on the GTX1060 GPU. An accuracy rate of 96 to 97 percent is satisfactory, and it would be difficult to improve by one or two percentage points. Further changes, such as changing the optimiser, did not improve accuracy.

Actual freight train	0.97	0.03
Actual passenger train	0.04	0.96
	Predicted freight train	Predicted passenger train

Table 3: Heatmap of *MobileNet*.

4.4 Text detection and recognition for train numbers

Detecting text on trains is much more complicated than the previous task of classifying trains. This is due to the different placements and different types of scheme numbers. The complexity of the text, caused by different and non-standard fonts and widely varying letter spacing, is as well an issue. Additionally, trains can be dirty. Another problem is the angle. The photos must face the front of the train with an angle of no more than 5°. Otherwise, training becomes very difficult. The scraped dataset from *Bahnbilder* shows this diversity. Because of these problems, the dataset cannot be used. Only the already trained model and programs are available.

As mentioned in chapter three, the deep neural network *EAST* is a fast and efficient tool. The pre-trained *EAST* model is loaded with *OpenCV*. A prediction function must be defined. A bounding box and confidence values are being returned.

The coordinates of the bounding box are used as input for text recognition with *Tesseract*. When running the tool, there are several configuration options that have a significant impact on the process and therefore on the output. Those options are the following:

- Page segmentation mode: Set *Tesseract* to perform only a subset of the layout analysis, and to assume a particular form of image. Mode 8 is used here, which treats the image as a single word. Since the main text in the bounding box is a different word, this is the case.
- Language: The language to use. English is assumed if none is given. The results are not different in this case as most of the information contains letters and numbers.
- OCR engine mode: Mode 1 is neural networks, which gives the best results.

An example of an output is shown in Figure 9. The non-text message in the top right-hand corner of the image is misidentified as text. Some experiments have shown that image resolution also has a large effect, with more errors occurring as the level of detail increases when processing high resolution images. To get the best results, the resolution was set to 320x320. The OCR model cannot be expected to be 100% accurate. However, good results were obtained with both the *EAST* and *Tesseract* models.

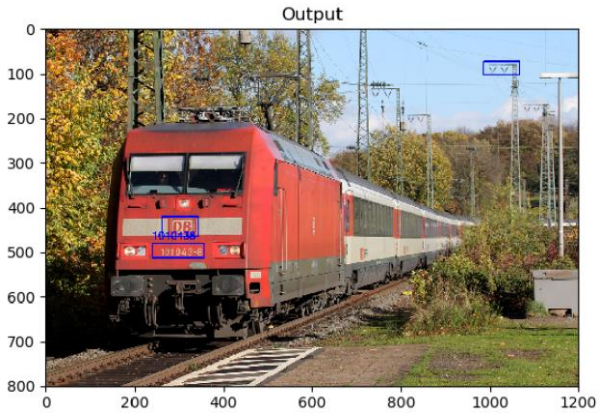


Figure 9: Text detection and recognition output.

After recognising the numbers and letters in the images, it is important to verify which part of the output is the desired train number. As explained in chapter two, the series scheme is based on seven digits. This is the first thing to check. Other verifications are necessary to make the recognition as robust as possible.

4.5 Computing wheelbase and speed of train with sensor data

The analysis in chapter two has shown that the use of lasers as sensors, e.g., for train speed detection, can be a viable option. In addition, if two sets of photogates are correctly spaced, further information about the wheelbase of the wagons can be obtained. Two signals are recorded each time a wheel passes a photogate, a high to low (falling edge) and low to high (rising edge) transition. According to [32] and shown in Figure 10, the standard container wagon wheelbase for a two-axle bogie is 1,800 mm, and 1,700 mm for a three-axle bogie. The dis-

tance between the two sets of photogates must be greater than the wheelbase but it should not be too long. It is therefore set at 2,000 mm. Vehicle wheelbases vary widely. It is an important piece of information when analysing loads. It can be compared with existing data to determine its unique application and load capacity. To confirm the wheelbase, it is important to first distinguish the type of bogie (one, two or three axles) of each wagon. As the wheel passes the sensor, its time stamp is recorded and the sensor to which it relates is recorded. Different strings are generated for different types of wagons. The string and the time array can be used to calculate the speed and wheelbase of the vehicle.

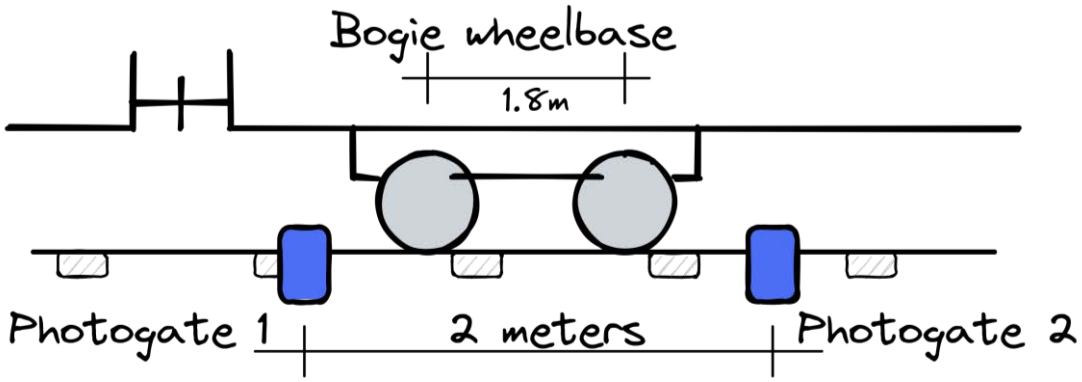


Figure 10: Sensor placement for specific wagon types.

First, the sensor must be mounted and tested as described in [42]. In terms of inputs (GPIO), Jetson Nano can read the status of buttons, switches, and dials, as well as sensors such as temperature, light, or motion. Once set up, the implemented code can be tested in conjunction with the sensor readings. The code needs to have the following functions.

When the first wheel passes the first sensor (sensor A), the photogate is interrupted and a signal is sent to the Jetson Nano. This activates the following functions. The first function (function A) records the timestamp and writes A to a defined string. The second sensor (sensor B) and the corresponding second function (function B) have the same purpose but write the letter B into the string. With the two time stamps recorded, the time difference can of course be calculated. With the set distance of two metres between the photogates, the calculation of the train's speed is trivial. Moreover, for the computer vision task, function A also initialises the photographing. An additional function then parses the string to obtain the bogie type and picks up the corresponding timestamp to calculate the speed and wheelbase of the vehicle. The number of calls of the function A or B corresponds to the number of axles and therefore as well the number of wagons attached to the locomotive. There are three different types of bogies and their corresponding strings: one-axle equals AB, two-axle equals AABB, and three-axle equals AAABBB.

Those string sequences are for axle distances smaller than two meters. For axle distances greater than 2 meters, different sequences arise. E.g., for a two-axle bogie: ABAB.

4.6 Derive further information from the data collected

One of the main goals of this work is to contribute to predictive maintenance of railway bridges by collecting as much data as possible about trains passing through. By identifying a specific type of train that has crossed the bridge, other information can be derived. The load imposed on the bridge by the train is the most important information.

For example, the image recognition part of the system identified a passenger train. The OCR part recognised the series scheme number starting with 422. The sensors detected eight

bogies and therefore four wagons including the locomotive. Using databases provided by Deutsche Bahn, the axle loads of the empty train can be accurately determined as shown in Table 4 [46].

	Axle load [t]				Axle distance [m]			
Type	First	Second	Third	…	First	Second	Third	…
DB - BR 422	15.8	15.8	17.9	…	3.015	5.215	17.905	…

Table 4: Axle loads and distances of a specific train.

This is very different from the load models used in the Eurocode, the European standards for structural design. There, a general load case is applied, which can differ a lot from the real loads, because the load case must be representative [47]. The difference increases when many short and less loaded trains cross the bridge.

5 DEPLOYMENT

As mentioned above, the data collection system has been designed with ease of deployment in mind. It is very easy to deploy the tested algorithms due to features of the AI-frameworks like *TensorFlow*. Models are trained on a personal computer and can be supplied directly on an SBC without any changes.

However, as a main program, in addition to the combination of the above parts, there are also runtime and error considerations to be taken into account. Because of the signalling block, the time interval between two trains is at least 100 seconds. This means that every 100 seconds the program must finish its work and be ready for the next train. The *TensorFlow* runtime has components that are lazily initialised, which can result in a high latency for the first request sent to a model after it is loaded. This latency can be several orders of magnitude higher than a single inference request. For this project it causes a delay on the first run after each reboot, the prediction process takes ten seconds. A warm-up is run to solve the problem. After this, the next prediction takes less than one second. The OCR task takes between ten and twenty seconds, so the program meets the runtime requirements very well. A loop ensures that the program continues to run. A defined period represents the time that has elapsed since the wheel last passed the sensor. It is greater than twenty seconds to ensure that the entire train passes through the sensor.

For the actual implementation on site, the first thing to consider is the location of the installation and the power supply. A good option is to use a second microcontroller and connect the two via a wireless network. The camera connected to the second microcontroller can then be mounted on an existing mast to save costs. The position of the camera in relation to the sensor also needs to be tested to get the right size image of the train.

Within the timeframe of this project, it was not possible to install the system close to Deutsche Bahn tracks, as this requires permission. However, the solution provides a complete technical path and demonstrates feasibility.

6 CONCLUSION

In this project, a machine learning and minimal hardware-based data collection system has been developed. Ease of deployment and low cost have been kept in mind. The total cost of the implemented system is less than $150 (SBC: $99, camera: $28, sensors: $12). The Python program running on the Jetson Nano takes pictures of passing trains and collects further data. The data is then analysed on the SBC itself, making the system an all-in-one solution.

The first result is train classification, where the deep learning model was able to identify the type of train, passenger or freight, with 95 to 97 percent accuracy. *TensorFlow* is used as a

training and deployment platform for image classification, not only on personal computers, but also on Jetson Nano. To build a dataset, a web scraper was developed to easily scrape categorised images from *Bahnbilder*. The CNN model and the *MobileNet* model were applied. Transfer learning with pre-trained weights from *ImageNet* was used on the *MobileNet* to improve the results. In addition, OCR algorithms were used to identify the train number. In the case of text detection and recognition, it was difficult to achieve a high level of accuracy under the existing conditions due to the lack of data sets and the diversity of train numbers. However, the implementation of the *EAST* model and the *Tesseract* program has shown its feasibility. Finally, two sensors allow the measurement of the, the speed and the counting of the wagons. The significant result here is an algorithm that uses the sequence of the two sensor signals to determine the bogie type based on the sequence of the two sensor signals. In addition, the calculation of the speed based on the elapsed time also gives good results. Depending on the bogie type and the speed, the wheelbase can then be determined.

7 DISCUSSION AND FUTURE WORK

Overall, the paper demonstrates the feasibility of an alternative low-cost data collection system. Collecting and analysing the data and deriving further information works quite well. However, a better data set for additional computer vision tasks would be an improvement. Cameras could be placed at different railway facilities to collect images so that the collected images are in a consistent format and can therefore be used for more AI tasks. A more complete train classification function can be implemented with a more detailed dataset, where the passenger train is divided into classes such as IC, ICE, RE, etc., and the freight train is further classified for types of freight wagons. In addition, image segmentation models can be trained to further extract textual information to complete the OCR part of the solution. Also, a better camera is required. Current CSI cameras do not meet night vision requirements. One solution is to place lights nearby, another is to use a camera with night vision capabilities. Cameras with high image quality are also needed to cope with high-speed trains. The power supply for the microcontroller is also a challenge that needs to be considered, e.g., solar panels. Programmatically, to achieve simultaneous detection of multiple trains, multiple threads, multiple processes can be considered. At the same time, a simpler C++ language can be used for programming to improve performance. Additionally, the system needs to be tested even more practically and under real conditions. This will require permission from Deutsche Bahn

For predictive maintenance, other key information needs to be further explored. Identifying trains can quantify the load of the train itself, but not the additional load such as passengers or goods. One solution could be to not only take photographs of passing trains, but also to record a video. The video could then be analysed. If the train is a freight train, the video could be analysed for its ballast. With additional AI algorithms, the type of goods, such as wood, gravel, containers, etc., that the wagons are carrying could be identified. However, some assumptions still must be made. If the train is a passenger train, it will not be possible to analyse the video for the number of passengers, as they will be inside the train. Another option is to analyse the platform camera footage. As described in chapter 2.6, Deutsche Bahn already uses this footage, for instance, to detect snow and automatically instruct the snow removal services. This footage can also be used to count passengers getting on and off the train to determine the number of people on the train.

REFERENCES

[1] Deutsche Bahn AG, Daten & Fakten 2021. Accessed: 01/23/2023.

[2] DB Netz AG, Brückenkarte. 2021. Accessed: 01/23/2023.

[3] Leistungs- und Finanzierungsvereinbarung III (LuFV III). 2020. Accessed: 01/23/2023.

[4] McKinsey Global Institute, Technology, jobs, and the future of work. 2017. Accessed: 01/24/2023.

[5] A. Agrawal, J. Gans, A. Goldfarb, *Power and Prediction: The Disruptive Economics of Artificial Intelligence.* Harvard Business Review Press. 2022.

[6] ZEKISS, research project funded by the BMDV. Accessed: 01/24/2023.

[7] I. Kalathas, M. Papoutsidakis, Predictive Maintenance Using Machine Learning and Data Mining: A Pioneer Method Implemented to Greek Railways. *Designs* 2021, 5, 5.

[8] A. Bousdekis, D. Apostolou, G. Mentzas, Predictive Maintenance in the 4th Industrial Revolution: Benefits, Business Opportunities, and Managerial Implications. *IEEE Engineering Management Review*, 48.1, 57-62, 2020.

[9] S.R. Lorenzen, H. Riedel, M.M. Rupp, L. Schmeiser, H. Berthold, A. Firus, J. Schneider, Virtual Axle Detector Based on Analysis of Bridge Acceleration Measurements by Fully Convolutional Network. *Sensors 2022*.

[10] Y. Yu, C.S. Cai, L. Deng, State-of-the-art review on bridge weigh-in-motion technology. *Advances in Structural Engineering*, *19*(9), 1514-1530, 2016.

[11] T. H. T. Chan, L. Y U, S. S. Law, T. H. Yung, Moving Force Identification Studies, I: *Theory*. *Journal of Sound and Vibration*, 247(1), 59–76, 2001.

[12] G. Kouroussis, C. Caucheteur, D. Kinet, G. Alexandrou, O. Verlinden, V. Moeyaert, Review of Trackside Monitoring Solutions: From Strain Gages to Optical Fibre Sensors. *Sensors 2015*.

[13] S. J. Johnston, M. Apetroaie-Cristea, M. Scott, S. J. Cox, Applicability of commodity, low cost, single board computers for Internet of Things devices. *IEEE 3rd World Forum on Internet of Things*, 141-146, 2016.

[14] E. Upton, Happy Birthday to us. 2022. Accessed: 01/24/2023.

[15] Y. Ran, X. Zhou, P. Lin, Y. Wen, R. Deng, A survey of predictive maintenance: Systems, purposes and approaches. *arXiv*, 2019.

[16] R. Keith Mobley, An introduction to predictive maintenance. *Elsevier*, 2002.

[17] L. Sun, Z. Shang, Y. Xia, S. Bhowmick, S. Nagarajaiah, Review of Bridge Structural Health Monitoring Aided by Big Data and Artificial Intelligence: From Condition Assessment to Damage Detection. *Journal of Structural Engineering* 146(5), 2020.

[18] C. Ortmeyer, A Brief History of Single Board Computers. *Electronic Design Uncovered*, Issue 06, 2014.

[19] NVIDIA, Embedded systems. Accessed: 01/25/2023.

[20] A. Burkov, The hundred-page machine learning book. *Andriy Burkov*, 2019.

[21] P. Buxmann, H. Schmidt, Künstliche Intelligenz. *Springer Gabler*, 2021.

[22] D. Forsyth, J. Ponce, Computer vision: a modern approach. *Pearson*, 2012.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*. 115 (3): 211–252, 2015.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv*, 2017.

[25] A. Chaudhuri, K. Mandaviya, P. Badelia, S. K. Ghosh, Optical Character Recognition Systems for Different Languages with Soft Computing. *Springer*, 2017.

[26] Optical Character Recognition Pipeline. Accessed: 01/26/2023.

[27] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J Liang, EAST: An Efficient and Accurate Scene Text Detector. *arXiv*, 2017

[28] R. Smith, An Overview of the Tesseract OCR Engine. *Ninth international conference on document analysis and recognition*, Vol. 2: 629–633, 2007.

[29] D. Sporici, E. Cuşnir, C.-A. Boiangiu, Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing. *Symmetry* 12(5):715, 2020.

[30] H.-R. Trankler, O. Kanoun, Recent advances in sensor technology. *Instrumentation and Measurement Technology Conference*. Rediscovering Measurement in the Age of Informatics 1: 309-316, 2001.

[31] V. J. Hodge, S. O'Keefe, M. Weeks, A. Moulds, Wireless Sensor Networks for Condition Monitoring in the Railway Industry: A Survey. *Transactions on Intelligent Transportation Systems* 16(3): 1088-1106, 2015.

[32] DB Cargo, Güterwagenkatalog. Accessed: 01/26/2023.

[33] DB Systel GmbH, Strong rail needs a strong digital partner. Accessed: 01/30/2023.

[34] DB Systel GmbH, (Soft) focus on anonymity. Accessed: 01/30/2023.

[35] DB Systel GmbH, Graffiti-Jagd mit künstlicher Intelligenz. Accessed: 01/30/2023.

[36] Camlin rail, Pantobot 3D: Predict, Prevent & Maintain. Accessed: 01/30/2023.

[37] Y. A. Badamasi, The working principle of an Arduino. *11th International Conference on Electronics, Computer and Computation*, 2014.

[38] NVIDIA, Jetson Nano. Accessed: 01/31/2023

[39] Raspberry Pi, Raspberry Pi 4 Tech Specs. Accessed: 01/31/2023.

[40] MIPI, CSI-2. Accessed: 01/31/2023.

[41] Waveshare, IMX219-160 IR-CUT Camera. Accessed: 01/31/2023.

[42] Adafruit, IR Break Beam Sensor. Accessed: 01/31/2023.

[43] NVIDIA, Getting Started with Jetson Nano Developer Kit. Accessed: 01/31/2023.

[44] Q-engineering, Install OpenCV 4.5 on Jetson Nano. Accessed: 02/01/2023.

[45] Bahnbilder. Accessed: 02/02/2023.

[46] Deutsche Bahn AG, Brückendynamik. Accessed: 02/05/2023.

[47] Eurocode 1: Actions on structures - Part 2: Traffic loads on bridges; German version EN 1991-2:2003 + AC:2010.

[48] mFund, Unsere Förderung für die Mobilität der Zukunft. Accessed: 01/24/2023.