TECHNISCHE UNIVERSITÄT WIEN

Institut für Computertechnik
Institute of Computer Technology

Master's Thesis

submitted by

David Breuss

Vision-Based Anomaly Detection for Railroad Systems

In partial fulfillment of the requirements for the degree of

Master of Science (M.Sc.)

Vienna, Austria, 2023

Study code:	UE 066 504
Field of study:	Embedded Systems
Supervisor:	Univ.Prof. Dipl.-Ing. Dr.techn. Jantsch Axel
Co-Supervisor:	Dipl.-Ing. Dr. Maximilian Götzinger

```
@Thesis{Breuss2023,
 type         = {Master’s Thesis},
 author       = {David Breuss},
 title        = {Vision-Based Anomaly Detection for Railroad Systems},
 school       = {Vienna University of Technology (TU Wien)},
 year         = {2023},
 address      = {Gusshausstrasse 27--29 / 384, 1040 Wien},
 month        = {November},
}
```

Contact me:

david.breuss@tuwien.ac.at

Abstract

Railroad systems play an essential role in the transportation of people and goods in today's society worldwide [1, 2, 3, 4, 5, 6]. The rail sector contributes to society in multiple ways besides transporting, including jobs, innovation, and industrial growth [3]. Furthermore, combined road and rail transportation can significantly reduce emissions and operating costs compared to road-only transportation [7]. Due to the wear-out of the railroad infrastructure and other damages, maintenance of the infrastructure is necessary to ensure the safe and reliable transportation of people and goods [8, 9]. During the last decades, the progress in Neural Network (NN)-based algorithms [10] and increases in computational power [11] enabled the usage of machine vision systems in several application domains like robotics, medical application, manufacturing and production, and security surveillance [12, 13, 14, 15]. Anomaly detection refers to the task of recognizing data and patterns that deviate significantly from some concept of normality or expected behavior [16, 17]. In many real-world applications, no or only a small number of labeled anomalous data is available [16]. This thesis aims to develop an image-based anomaly detection algorithm for rail tracks to detect rail damages, foreign objects, and heavy vegetation on the trackbed without relying on anomalies in training data. For the training and evaluation of the proposed algorithm, a dataset consisting of gray-scale images taken from a bird's-eye view perspective by a monochrome camera is used. Since this camera system is mountable on regular trains, this could pose a cost-effective approach for an automated railroad inspection system. The proposed Vision-based Anomaly Detection Algorithm for Railroads (VADAR) algorithm consists of three Autoencoders (AEs), a rail segmentation network, and a One-Class Classifier (OCC) to detect anomalies on the trackbed and rails. Utilizing these networks enables VADAR to lower the number of false positives resulting from the gravel and relatively rare infrastructure elements like switch-frogs or attachments to the rails and crossties. The experimental results in this thesis show that VADAR can detect trackbed anomalies with sizes down to roughly 10,000 mm^2 with a recall rate of more than 51% while achieving a false positive rate of 4.8%. Rail damages with sizes down to roughly 100 mm^2 were detected with a recall rate of more than 53% while maintaining a false positive rate of less than 4.2%. The detection method of VADAR allows a tradeoff between the recall rate and false positive rate to adjust for the needs of a specific task. The experiments also show a strong positive correlation between the size of anomalies and the recall rate. VADAR can detect certain trackbed anomalies larger than roughly 20,000 mm^2 with up to 100% while maintaining a false positive rate of 5.3%. This thesis also includes a comparison of VADAR with a state-of-the-art approach. VADAR's recall rate, accuracy, and false positive rate outperform this approach's results while also enabling the detection of significantly smaller anomalies like rail damages. This algorithm demonstrates that a vision-based anomaly detection system utilizing only one camera mounted on regular trains could potentially be used for an automated railroad inspection system. Since VADAR can detect even small rail damages, further developments might enable VADAR to be used for the predictive maintenance of rail tracks.

Kurzfassung

Eisenbahnsysteme spielen weltweit eine wesentliche Rolle bei der Beförderung von Personen und Gütern [1, 2, 3, 4, 5, 6]. Der Eisenbahnsektor leistet einen vielfältigen Beitrag zur Gesellschaft, u. a. auch in Form von Arbeitsplätzen, Innovation und industriellem Wachstum [3]. Außerdem kann der kombinierte Straßen- und Schienentransport die Emissionen und Betriebskosten im Vergleich zum reinen Straßentransport erheblich senken [7]. Aufgrund der Abnutzung der Eisenbahninfrastruktur und auftretenden Schäden ist die Instandhaltung der Infrastruktur notwendig, um den sicheren und zuverlässigen Transport zu gewährleisten [8, 9]. In den letzten Jahrzehnten haben die Fortschritte bei den auf neuronalen Netzen basierenden Algorithmen [10] und die Zunahme der Rechenleistung [11] den Einsatz von Bildverarbeitungssystemen in Anwendungsbereichen wie Robotik, Medizin, Fertigung sowie Überwachungssystemen ermöglicht [12, 13, 14, 15]. Eine der Anwendung befasst sich mit der Anomaliedetektion, also dem Erkennen von Daten und Mustern, die von einem Konzept der Normalität abweichen [16, 17]. In vielen Anwendungen sind keine oder nur wenige annotierte Anomalie-Daten verfügbar [16]. Ziel dieser Arbeit ist es, einen bildbasierten Algorithmus zur Erkennung von Anomalien im Gleisbereich zu entwickeln, um bspw. Schienenschäden und Fremdkörper zu erkennen, ohne auf Anomalien in den Trainingsdaten angewiesen zu sein. Für das Training und die Evaluierung des vorgeschlagenen VADAR-Algorithmus wird ein Datensatz verwendet, der aus Graustufenbildern besteht, die von einer Monochromkamera aus der Vogelperspektive aufgenommen wurden. Da dieses Kamerasystem auf regulären Zügen montiert werden kann, könnte dies einen kosteneffektiven Ansatz für ein automatisiertes Eisenbahninspektionssystem darstellen. VADAR besteht aus drei Autoencodern, einem Schienensegmentierungsnetzwerk und einem One-Class Classifier zur Erkennung von Anomalien am Gleisbett und auf den Schienen. Diese Netzwerke ermöglichen es VADAR, Fehlalarme zu reduzieren, die durch Schotter und Infrastrukturelemente wie bspw. Weichenherzen entstehen. Die Ergebnisse in dieser Arbeit zeigen, dass VADAR Gleisbettanomalien bis zu einer Größe von etwa 10.000 mm^2 mit einer Recall-Rate von mehr als 51% und einer False-Positive-Rate von 4,8% erkennen kann. Schienenschäden bis zu einer Größe von ca. 100 mm^2 wurden mit einer Recall-Rate von mehr als 53% bei einer False-Positive-Rate von weniger als 4,2% erkannt. VADAR ermöglicht den Kompromiss zwischen Recall- und False-Positive-Rate, um an Spezifikationen einer Anwendung anzupassen. Die Experimente zeigen eine starke Korrelation zwischen der Größe der Anomalien und der Recall-Rate. VADAR kann bestimmte Anomalien die größer als ca. 20.000 mm^2 sind, mit bis zu 100% erkennen, während die False-Positive-Rate bei 5,3% liegt. Diese Arbeit zeigt auch einen Vergleich von VADAR mit einem State-of-the-Art Ansatz. Sowohl Accuracy als auch Recall- und False-Positive-Rate von VADAR übertreffen die Ergebnisse dieses Ansatzes und ermöglichen die Erkennung von deutlich kleineren Anomalien — wie zum Beispiel Schienenschäden. Diese Arbeit zeigt, dass ein bildbasiertes System zur Anomaliedetektion, mit nur einer an regulären Zügen angebracht Kamera, potenziell für ein automatisiertes Eisenbahninspektionssystem verwendet werden kann und interessant für Predictive Maintenance sein könnte.

Contents

List of Tables

List of Figures

Acronyms

AE	Autoencoder.
ANA image	image containing only anomaly annotations.
ANINA image	image containing anomaly and infrastructure annotations.
BRTI	Basic Rail Transportation Infrastructure.
CLAHE	Contrast Limited Adaptive Histogram Equalization.
CNN	Convolutional Neural Network.
DAE	Denoising Autoencoder.
DCGAN	Deep Convolutional Generative Adversarial Network.
DNN	Deep Neural Network.
GAN	Generative Adversarial Network.
GPU	Graphics Processing Unit.
HDAE	High Detail Autoencoder.
IAE	Infrastructure Autoencoder.
INA image	image containing only infrastructure annotations.
LCA	Large Coherent Area.
ML	Machine-Learning.
NN	Neural Network.
NOA image	image containing no annotations.

OCC	One-Class Classifier.
PC	Principal Component.
PCA	Principal Component Analysis.
RAAE	Rail Anomaly Autoencoder.
ReLU	Rectified Linear Unit.
t-SNE	t-distributed Stochastic Neighbor Embedding.
TAAE	Trackbed Anomaly Autoencoder.
VADAR	Vision-based Anomaly Detection Algorithm for Railroads.
VAE	Variational Autoencoder.

Chapter 1

Introduction

Railroad systems have been crucial to the emergence of modernization and progress. The transportation of people and goods is a central part of today's urbanized society [1]. For Example, between the years 2018 and 2020, approximately 100 million tons of goods were transported annually by the railroad system in Austria [18, 19, 20]. Nevertheless, also in the rest of the European Union, the rail sector is a vital contributor to innovation, industrial growth, jobs, and society [3]. The same goes for many countries around the globe. Railroad systems also play an increasingly significant role in the economic development of other countries like India [5] and countries in Africa [4]. Hong Kong has acknowledged "the railroad as the backbone of the public transport system" [21], and other Chinese cities such as Beijing, Guangzhou, and Shanghai identify the railroad as one of the main assets of their transport systems [6]. According to European development economists, a country's economic progress can be estimated by the Basic Rail Transportation Infrastructure (BRTI) index, which informs on the modernity and expansion of its railroad infrastructure [22].

Expanding rail infrastructure is today more important than ever because it allows for more transportation of people and goods via trains and is an effective way to combat the climate crisis. Studies like that of Torres *et al.* [7] show that transitioning from road-only transport to intermodal transport with road and rail significantly reduces emissions, fuel consumption, and operating costs.

1.1 Motivation and Problem Description

Since wear-out [8], climatic influences, or other damages to the infrastructure lead to the deterioration of the railroad system [9], maintenance is necessary. In order to secure reliable and safe transportation, maintenance is essential, and the inspection of rails and the trackbed is necessary to plan and coordinate maintenance. Regular maintenance is crucial to ensure the safety of transportation, guarantee a reliable railway system, and extend the rail service life. The maintenance work consists of activities like inspecting the rails and trackbed, grinding, and the power system maintenance [23]. These necessary activities are resource heavy and expensive. The EIM-EFRTC-CER Working Group reported an annual 15 to 25 billion EUR budget for European infrastructure managers for the maintenance of their railroad infrastructure [24]. The inspection of the total track length of 370,000 km in the European Union is responsible for an estimated cost of 70 million EUR per year [25]. This estimation assumes annual vehicle ultrasonic inspections and additional manual verification. Today either expensive and slow measuring vehicles infrequently inspect rail tracks or trained personnel continuously inspect the rail infrastructure superficially [26, 27, 28]. Defects and damages to the rails and other anomalies include cracks [29], head checks [30], squats [31], wheel burns [32], missing or faulty hardware [33], heavy vegetation [34], or unexpected objects [35]. According to conversations with domain experts measuring vehicles inspect some parts of the railroad system only up to two times a year. This infrequent measuring can be problematic if, e.g., a rail is damaged briefly after such an inspection. Since the inspection of railroad systems is an essential part of the according maintenance process, and several kinds of damages to the system and other anomalies can be detected visually, this presents an opportunity for automation by a machine vision system. By installing a camera system on regular trains, an automatic visual inspection of the rails and trackbed could be done more frequently and cost-effectively.

Over the last decades, increasing computational power [11], advanced algorithms, and the progress in Neural Networks (NNs) [10] led to the automation of many processes. Nowadays, computer and machine vision systems have a central role in various application domains like medical imaging, robotics, and surveillance systems [12]. Besides image segmentation, image classification, and object detection, anomaly detection is another task machine vision systems are used for [36]. The detection of anomalies is an important part of several applications like manufacturing and production [14], security surveillance [15], inspection-related tasks [37] or medical applications [13].

In general, anomaly detection is a challenging task. Especially the detection of anomaly frames in vision-based applications is a complex problem. For humans, detecting anomalies often seems like a rather easy task because years of evolutionary training and adaption allow us to detect patterns and to distinguish between relevant information and noise easily [38]. Developing an explicitly programmed algorithm to detect anomalies in images for a specific application can be very difficult because there might be no simple definition of anomalous and normal data. Furthermore, changes in contextual information like the background and lighting conditions in images may lead to a larger variety within normal data [39]. This might introduce even more challenges for an anomaly detection system that must be considered. Therefore, popular approaches to deal with these problems are NNs. Appropriate training procedures allow NNs to learn what features and characteristics represent normal data or anomalous data [36].

In the development of many practical applications, anomaly detection algorithms face a common problem: the scarcity of anomalous data. [16]. Furthermore, often the used dataset is not even labeled. Generally, anomaly detection systems are designed to find non-conforming patterns with respect to the data set normality. Section 2.2 gives a more precise definition of anomalies. For example, depending on the application and the used data type, anomalies can include damages to a product or system, unexpected or missing objects, or simply outlier data points. In general, anomaly detection poses a challenge to machine vision because there is no easy definition of anomalies for a particular set of images. However, engineers and researchers were able to take advantage of the potential of machine vision systems, neural networks, and other algorithms to design anomaly detection systems for a variety of problems like the detection of cyber attacks in communication networks [40], the detection of anomalous grapevine berries [41], flight anomaly detection during the take-off phase [42] or even anomaly detection in various medical application domains [43].

This thesis introduces a NN-based anomaly detection system called Vision-based Anomaly Detection Algorithm for Railroads (VADAR). A camera system allows for analyzing images of the rail tracks to detect anomalies like damages, foreign objects, and vegetation. Since the camera system can be installed on regular trains, the proposed system may enable an automatic continuous inspection of the rails and trackbed.

1.2 Methodology

The problem description is approached with different machine learning methods. Machine learning is an interesting approach for anomaly detection problems since it can work with large datasets to learn from past experiences to apply the learned concepts to new data [44]. Like in many real-world anomaly detection tasks, labeled anomalous samples are rare. Therefore, this work mainly focuses on anomaly detection approaches suitable for unsupervised training. Reconstruction-based anomaly detection approaches, where a model tries to reconstruct given input images, are particularly interesting for vision-based problems because the pixel-wise comparison of input and output data enables a straightforward interpretation of the anomaly detectors' decision. Several anomaly detection systems utilize reconstruction-based approaches like Autoencoders (AEs) [40], [41], [45]. Therefore, this work mainly focuses on reconstruction-based methods, and several different architectures are tested and analyzed. Since the dataset this work is focused on does not contain many anomalous rails, some of these annotated rail damages were extracted and replicated on rails in other images. In addition to the actual annotated rail damages, the evaluation of VADAR considered these synthetic rail damages for analysis. This work also presents a performance comparison of VADAR to a state-of-the-art approach. All neural network models are implemented with CUDA 11.3 [46] within PyTorch [47], which is an open-source machine learning framework (Version 1.11.0).

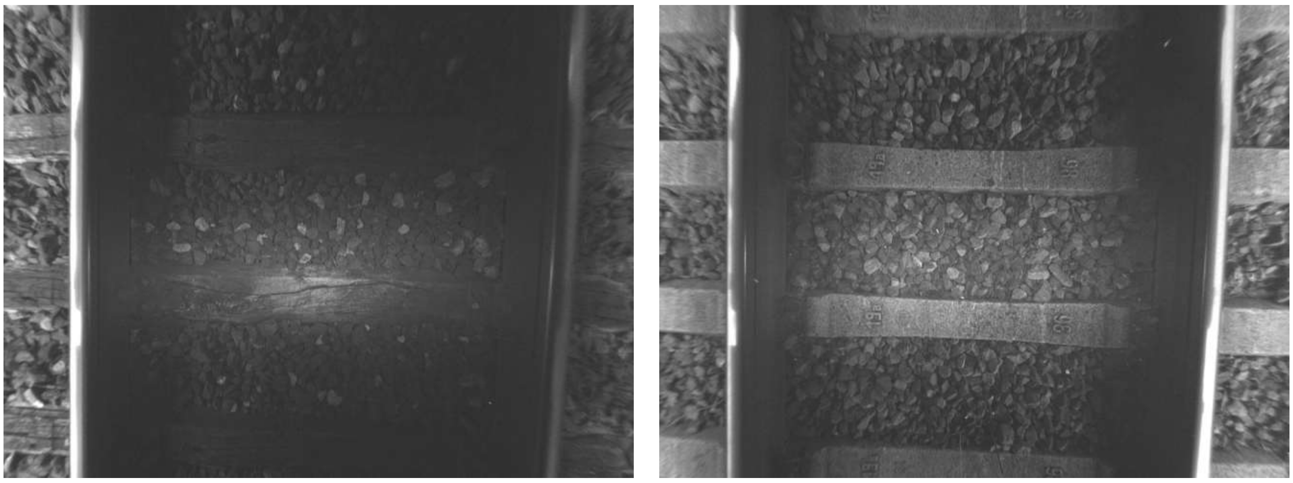

(a) Image of trackbed with wooden crossties. (b) Image of trackbed with concrete crossties.

Figure 1.1: This dataset includes images with varying lighting conditions and different crosstie materials.

1.3 Objectives and Research Questions

The main objective of this thesis is to investigate image-based anomaly detection approaches based on deep neural networks and their applicability to a specific dataset. The dataset consists of grey-scale images of rails and trackbeds of railroad systems, where the images were taken from a birds-eye perspective. Figure 1.1 shows two example images of different sequences of the dataset. The dataset contains images from sequences with varying lighting conditions, crosstie materials, and gravel. Since the dataset consists of images taken by a monochrome camera, the anomaly detection approach must be robust regarding varying lighting conditions. Anomalies of interest include objects placed on the track bed, damages to the rails and trackbed, or vegetation covering the track bed. Furthermore, the anomaly detection accuracy for the dataset with varying rails, trackbed, and lighting conditions are investigated, and constraints and limitations of the approaches and the dataset are analyzed.

This work aims to explore vision-based anomaly detection methods to find an approach that is robust to changes in the scenery, like varying lighting conditions. Like many other anomaly detection tasks, this work faces common challenges, such as the lack of knowledge of the anomalies' nature before encountering them, the diversity of possible anomalies, the rarity of anomalies, as well as the resulting class imbalance between normal and anomalous data [48].

The main research questions this thesis focuses on are the following:

- What challenges do the rail track images recorded by a monochrome camera from a birds-eye perspective introduce, and how can they be tackled?
- What properties and characteristics of anomalies influence the recall rate?
- Down to what size can anomalies be detected at a reasonable false positive rate?

1.4 Contributions and Organization

The main contribution of this work is VADAR, a vision-based anomaly detection algorithm capable of detecting anomalies like damages, foreign objects, and heavy vegetation on rail tracks. A cost-effective monochrome camera mounted on regular trains captured images of rail tracks from a birds-eye view perspective to train and evaluate this anomaly detection method. This camera system recorded images of different parts of rail tracks under varying lighting conditions. VADAR explicitly focuses on these grey-scale birds-eye view images. Therefore, the training and evaluation of VADAR only include such images. The different camera perspectives and focus on rail damages, foreign objects, and heavy vegetation on the trackbed while utilizing images of just one monochrome camera differentiates this system from state-of-the-art. Several experiments demonstrate VADAR's performance and what anomalies' properties and characteristics impact the recall rate. Experiments show that VADAR can detect certain trackbed anomalies larger than roughly 10,000 mm^2 with up to 100% recall rate while achieving a false positive rate of 5.3%. Since detecting smaller anomalies like rail damages is particularly interesting, the influence of anomalies' sizes upon VADAR's performance is investigated in more detail. VADAR achieved a recall rate of 53% for rail damages with sizes down to 50 mm^2 while maintaining a false positive rate of less than 4.2%. This thesis also includes a comparison between VADAR and a state-of-the-art anomaly detection system. These comparisons show that VADAR outperforms this state-of-the-art system in recall rate, false positive rate, and accuracy.

Chapter 2 of this thesis overviews the background, state-of-the-art, and related works. Specifically, this chapter introduces NNs, essential aspects and challenges of anomaly detection, multiple anomaly detection approaches, and existing work on anomaly detection in railroad systems. Then Chapter 3 introduces the dataset this work focuses on, explains the annotations within the dataset, and the challenges this dataset poses. The introduction to this dataset is followed by an analysis of various NN-based architectures in Chapter 4. Furthermore, this chapter explains the implementation of VADAR. Then Chapter 5 demonstrates VADAR's performance through several experiments. This chapter also compares VADAR to a state-of-the-art anomaly detection system for railroad systems. Finally, the thesis ends with Chapter 6, which gives a conclusion and an outlook.

Chapter 2

Background and State of the Art

Section 2.1 summarizes some Neural Networks (NNs) fundamentals and training methodologies. Then Section 2.2 presents definitions used in the context of anomaly detection and different approaches for anomaly detection. Section 2.3 goes into more detail about the challenges anomaly detection approaches encounter. Section 2.4 gives information about methods especially interesting for vision-based applications. The chapter closes with Section 2.5, which presents related work in the field of automatic inspection and anomaly detection systems for railroad systems.

2.1 Artificial Neural Networks

Originally the concept of artificial NNs was inspired by the human brain and was first used for classification tasks to learn from training data [49]. Nowadays, Convolutional Neural Networks (CNNs), a specific NN-type, is a popular and well-established tool in image recognition, segmentation, classification, and other computer vision tasks [50, 51, 49, 52]. This section introduces the basic principles of NNs based on the books [53, 54]. A short introduction to the fundamental ideas behind NNs and an analysis of fully connected NNs is given. Then the focus shifts to CNNs, the layers such networks are built upon, and how to work with them.

2.1.1 Fully Connected Neural Networks

Since a so-called fully connected NN has a relatively simple architecture, it is an excellent example to discuss the fundamental principles of NNs. Such networks consist of many nodes (neurons) with several inputs and outputs. These nodes are grouped in layers, and the nodes of neighboring layers are connected to each other [55]. These networks' three different layer types are the input, hidden, and output layers. Networks containing multiple hidden layers are often called Deep Neural Networks (DNNs) [56]. Figure 2.1 shows the structure of such a network. Every neuron of layer i has a weighted sum of all the outputs of layer $i-1$ as its input, where i is a number between 1 and the total number of hidden layers P. A bias can be added to this weighted sum for every neuron. Each neuron then applies a so-called activation function, which can be nonlinear, to this value. According to [57], three common examples of activation functions are the Sigmoid (or Logistic curve) function

$$f(x) = \frac{1}{1+e^{-x}}, \tag{2.1}$$

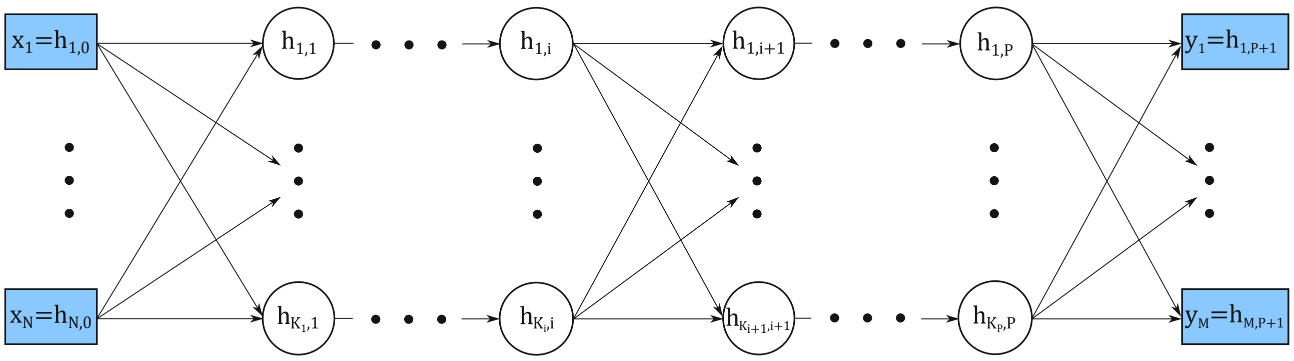


Figure 2.1: This graphic is a visual representation of a fully connected NN. The input layer includes the nodes $x_1, \ldots, x_N$ and the P hidden layers each consist of the nodes $h_{1,i}, \ldots, h_{K_i,i}$, where K_i refers to the number of neurons for the layer i. The output layer consists of the neurons $y_1, \ldots, y_M$ [53].

the Tanh (Hyperbolic Tangent) function

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1, \tag{2.2}$$

and the ReLU (Rectified Linear Units) function

$$f(x) = max(0, x). \tag{2.3}$$

For example, the output value of the neuron $h_{k,l}$ for a network, with a structure as in Figure 2.1, can be calculated as

$$h_{k,l} = f\left(\sum_{j=1}^{K_l} (w_{j,l-1} h_{j,l-1}) + b_{k,l}\right), \qquad 1 \leq k \leq K_l, 1 \leq l \leq P \tag{2.4}$$

where $f(\cdot)$ refers to the activation function of the neuron $h_{k,l}$, $w_{i,l-1}$ is the weight for the neuron output of $h_{i,l-1}$, and $b_{k,l}$ is the bias of neuron $h_{k,l}$ [58]. The weights and biases of the neurons are learnable parameters. During the training of a NN model, those parameters are adapted to minimize a loss function, which depends on the specific task and approach. Usually, the loss function describes the deviation between the output of the model and the expected or true value [50], which is either given implicitly through the task or explicitly by the labeled data.

2.1.2 Convolutional Neural Networks

In contrast to the before mentioned fully connected NNs, CNNs preserve the locality of the input data, which is especially important for image processing tasks. The loss of locality of the input data in fully connected NN is a direct consequence of the idea that each neuron operates on the weighted sum of all neuron outputs of the previous layer. Therefore, a specific neuron has no information about which part a neuron has contributed to the weighted sum at its input. However, CNNs use convolutional layers instead of fully connected layers which was heavily inspired by mathematical operations already used for several computer vision tasks [59, 60, 61]. Every convolutional layer consists of multiple kernels, where each kernel extracts features by a convolution operation on the so-called feature maps produced by the previous layer. A visual representation of an input tensor, the kernels of one convolutional layer, and the resulting

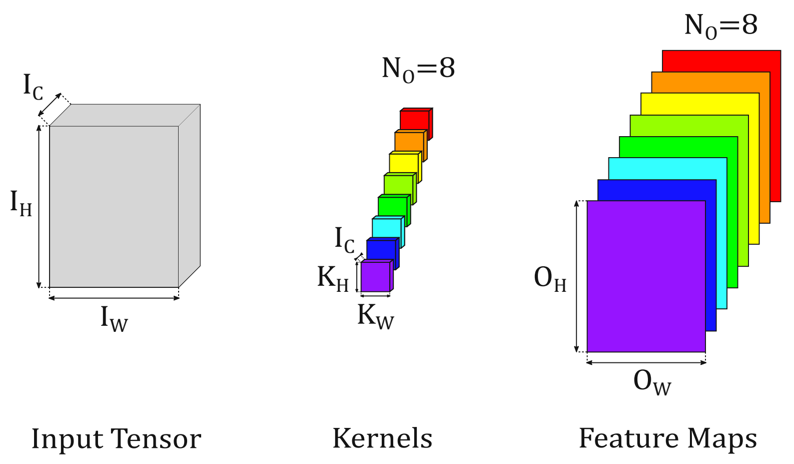


Figure 2.2: The dimensions of the input tensor, the kernels, and resulting feature maps are visualized for the case of $N_O = 8$ kernels of a convolutional layer [53].

feature maps are shown in Figure 2.2. The dimensions of the output feature-maps O_W and O_H depend on the dimensions of the input tensor I_W and I_H, the dimensions of the kernels K_W and K_H, the stride, the dilation, and the padding of the convolutional layer. Stride refers to the number of pixels by which the kernel slides along the height (S_H) and width (S_W) of the input tensor. By adjusting dilation, the field of view of the kernel can be increased without adding additional parameters. Dilation refers to the spacing between the kernel weights in the height (D_H) and width (D_W) dimensions. Through padding, dummy pixels can be added to the width (P_W) and height (P_H) of the output feature maps. Padding can be helpful if O_W and O_H should match I_W and I_H since, without padding, the output dimensions will be slightly smaller depending on the kernel size and dilation. As described in [53], the dimensions of the output feature-maps O_W and O_H can be calculated by using the equations

$$O_H = \left\lfloor \frac{I_H + 2P_H - D_H\left(K_H - 1\right) - 1}{S_H} + 1 \right\rfloor \text{ and} \tag{2.5}$$

$$O_W = \left\lfloor \frac{I_W + 2P_W - D_W\left(K_W - 1\right) - 1}{S_W} + 1 \right\rfloor . \tag{2.6}$$

Besides convolutional layers, so-called pooling layers are often used in CNNs. Pooling layers can reduce the size of the response map or convert spatial response maps to vectors to be compatible with the following linear classifier layers [53]. In CNNs, *max*- and *average* pooling are two of the most commonly used pooling layers. However, there exist multiple different pooling methods that rely on different pooling techniques and combinations of various pooling layers [62]. Another type of layer that is commonly used in modern CNNs is the batch normalization layer. In this context, batches refer to multiple instances of data. Instead of using a single data instance, batches can be fed to a NN simultaneously. Batch normalization was introduced in [63] and revolves normalizing a layer input for each training mini-batch. According to [63], this method allows significantly higher learning rates. Additionally, this increases the robustness regarding the initialization of the model. Furthermore, batch normalization can introduce a regularization effect and sometimes eliminates the need for dropout as a regularization scheme [63]. Extracting features using convolutional layers is computationally expensive and involves many floating-point operations. Nowadays, CNNs are usually run on Graphics Processing Units (GPUs) because the convolution operations are performed through multiple matrix multiplications, which are easily parallelizable through GPUs [64].

2.1.3 Features and Subspaces

Especially in image-based applications, input data may contain a large amount of information. NNs are valuable for extracting complex information from high-dimensional data for a specific task. Some NNs are using these features for powerful classification abilities [65, 66]. Typically this classification is accomplished by extracting certain features from data. These feature-based methods learn during a training procedure which features correspond with a given selection of classes or try to cluster instances according to the extracted features [67, 66]. Other approaches try to reduce the dimensionality of the input data to represent it in a lower dimensional space. In this context, this lower dimensional space is often referred to as subspace or latent space [68, 69]. This transformation of input data to a latent space is sometimes used to allow for a more efficient way to analyze the data further or simply as a compression mechanism [68, 70]. Section 2.4 discusses these approaches in more detail.

2.1.4 Training Neural Networks

The values of the weights and biases of each neuron in a NN are responsible for the error of a model, which is defined by the value of a loss function. The calculated loss forms a hyperplane, and the objective of the training is to adapt the weights and biases to reach a minimal error. This adaptation of the model parameters, specifically the weights and biases of each neuron, is accomplished by backpropagation. In principle, the weights and biases undergo slight changes to come closer and closer to a local minimum of the error by analyzing the gradients of the loss function with respect to weights and biases. A more detailed description of the backpropagation process and optimization techniques can be found in [53].

Data augmentation

Through the training process, a CNN shall learn useful features representing the training data. In general, the training data should represent the real-world data the model will encounter during its usage. In many cases, the amount of available data for training is limited, or the dataset is imbalanced. Typically, the performance of CNNs can be improved by expanding the training dataset [71]. Expanding a training dataset comes with additional financial costs or delays and might not be feasible for some applications. Instead, an expansion of the dataset can also be achieved by augmenting the already available data. Furthermore, the performance of some NN models can be significantly improved when the input data is augmented or augmented versions of the original data are fed to the network additionally [72]. In several works, multiple ways of augmenting input data and sometimes combinations of multiple augmentation methods are applied simultaneously [71, 72, 73]. In the following, a list of data augmentation methods is given. Without a claim of completeness, this list mainly focuses on methods relevant to gray-scale images.

- **Rotation**: Applying different rotations to an image. In some applications, multiples of $90°$-rotations are used to quadruple the number of training data effectively [73], but in general arbitrary angles can also be used in some applications.
- **Flipping**: By flipping an image horizontally or vertically, additional data for training is generated.
- **Cropping**: Using different crops from the original data is another way of generating additional training data. For some applications, the cropped image might need to be resized to match the specific dimensions.

- **Noise**: Adding different kinds of noise to images. Popular noise choices are Gaussian or salt and pepper noise, which alter the values of separate pixels within an image. In some works, the process of masking patches of images is referred to as adding noise as well [74].

- **Blurring**: Blurring the input image can lead to a more robust model regarding the camera system. [72].

- **Brightness adaptation**: The brightness of images can vary due to the lighting conditions. The robustness of NNs regarding different lighting conditions can be improved by adapting the brightness distribution of the dataset. One popular method to equalize the brightness distribution is Contrast Limited Adaptive Histogram Equalization (CLAHE) [75].

- **Synthetic Data**: Although this method does not augment existing data but is based on the generation of synthetic data, in literature, this approach is often referred to as a data augmentation strategy [43]. Nowadays NNs, like Generative Adversarial Networks (GANs), are used to learn from the available dataset to generate synthetic data.

Depending on the application, developers need to consider if a specific type of augmentation potentially changes the meaning of the data and its consequences. For example, rotating an image of a handwritten number six by roughly 180° will look like an image of a number nine.

Regularization

The goal of training any NN model should be to enable the model to perform well on unseen data samples. Usually, a NN is provided with samples from a training dataset during training. In order to check the performance of the trained model, a validation dataset is used. The validation dataset only contains samples the model has not encountered during training. Usually, after the initialization of a model, both the training and validation performances improve. However, overfitting describes the problem where a model's performance on training data continues to improve throughout the training but simultaneously stagnates or declines on unseen data [53]. To ensure this generalization capability, the problem of overfitting needs to be addressed. The deviation between the training and validation performance is often referred to as the generalization error. The objective of any regularization mechanism is to reduce this generalization error. The generalization error is reduced by preventing the network from reaching the absolute minimum of its loss function because the model might have learned concepts that are fine-tuned to some specific training samples and will not be beneficial for unseen samples. The most straightforward regularization approach is to stop training when the validation performance stops increasing. Parameter norm penalties are another form of regularization. Some of the most commonly used penalty norms are the so-called L_1 and L_2 norm penalties [76]. The L_1 norm penalty induces sparsity in the weights, while the absolute values of the weights are constrained in the optimization problem. The constraints on the norm of the model parameters by the L_2 norm penalty ensure that the linear transformations are bounded [77]. Another regularization method is *dropout*. *Dropout* refers to the idea of randomly selecting some elements of input tensors and setting them to zero during the training process. Because this random selection of elements changes with every sample, multiple different network configurations are used, and neurons are trained to perform well in the absence of surrounding neurons, leading to more robust features learned [53]. *Dropout* is most effective for fully connected layers and is often used for those layers only. However, there also exist specialized *dropout* implementations for convolutional layers [78]. As briefly mentioned before, batch normalization also has a regularization effect that enables omitting the *dropout* strategy in some cases [63]. In many problems, data augmentation is also used as a regularization method. Some images may be flipped, rotated, or cropped and still represent valid data that can be used to train the model. Of course, valid augmentations are dependent on the specific dataset, task, and application. In some applications, adding noise to the input data may also result in a more robust model.

Optimization algorithms

In NNs, loss functions give a metric for measuring training progression. This loss indicates how well the model can generate the expected output. The larger the deviation of the output from the target values, the higher the value of the loss function is. An optimization algorithm is needed to minimize this loss function to guide the model during training to model parameters that lead to better results. There exist multiple algorithms specifically designed for the optimization of NNs. The goal of optimization algorithm design for NNs includes fast convergence and the improvement of a particular metric of interest [79]. We can write the minimization problem as a finite-sum optimization problem, thus

$$\min_{\theta} F(\theta) = \frac{1}{B} \sum_{i=1}^{B} F_i(\theta) \tag{2.7}$$

where each $F_i(\theta)$ is the sum of training loss for a mini-batch of training samples, and B represents the total number of mini-batches [79]. In practice, B usually is set to powers of two. Stochastic gradient descent (SGD) and its variants are a prevalent class of optimization methods. SGD performs an update for the parameters

$$\theta_{new} = \theta_{old} - \eta \nabla_\theta \left(F_i(\theta_{old}) \right) \tag{2.8}$$

where $F_i(\theta_{old})$ refers to the loss function of the i-th minibatch of the model with the old parameter values, and η is the learning rate [80]. If no mini-batches but rather every single sample of the training dataset is used as a separate input, B is 1. Some papers distinguish between the cases where B is 1, B is the total number of training samples (i.e., there is only one minibatch), or B is some value in between [80], but in this section, all of these cases are referred to as SGD. With this basic approach, some problems arise. Choosing the optimal learning rate can be challenging since small learning rates lead to slow convergence, and if it is too large, it can hinder convergence or cause fluctuations in the loss function. Better results can be accomplished by using learning rate schedules to adjust the learning rate during training. However, the schedules and necessary thresholds are defined in advance and cannot adapt to the dataset's characteristics. Furthermore, the same learning rate is used for all parameter updates. Since the data used can have features with different frequencies, there might be better solutions [80]. There are multiple variations and adaptations to this vanilla SGD algorithm to deal with these challenges. One way of improvement is the introduction of a so-called momentum term $\gamma\nu_{old}$, where γ is a value between 0 and 1. This momentum term is a fraction of the update vector of the past time ν_{old} step which is added to the current update vector

$$\nu_{new} = \gamma\nu_{old} + \eta\nabla_\theta F_i(\theta_{old}), \qquad \theta_{new} = \theta_{old} - \nu_{new}. \tag{2.9}$$

This momentum increases when the gradients point in the same directions and reduces updates for directions where the gradients change over multiple iterations. Therefore, faster convergence and reduced oscillations are achieved. The so-called *Adagrad* optimizer adapts the learning rate to parameters, making bigger updates for infrequent and smaller updates for frequent parameters. *Adaptive moment estimation (Adam)* is a different method that computes adaptive learning rates for every parameter [81]. Adam stores an exponentially decaying average of past squared gradients ν_{old} and an exponentially decaying average of past gradients m_{old} similar to momentum, as

$$m_{new} = \beta_1 m_{old} + (1-\beta_1)\nabla_\theta F_i(\theta_{old}), \qquad \nu_{new} = \beta_2 \nu_{old} + (1-\beta_2)\left(\nabla_\theta F_i(\theta_{old})\right)^2, \tag{2.10}$$

where β_1 and β_2 are parameters. m and ν are estimates of the mean and uncentered variance of the gradients, respectively. The Adam update rule is given as

$$\theta_{new} = \theta_{old} - \frac{\eta}{\sqrt{\tilde{\nu}_{new}} + \epsilon}\tilde{m}_{new}, \tag{2.11}$$

where $\tilde{\nu}_{old}$ and $\tilde{m}_{new}$ are bias-corrected versions of ν_{new} and m_{new}, and ϵ is usually set to 10^{-8} [81].

2.2 Anomaly detection

An anomaly is a measurement or data that shows significant deviation from some concept of normality [16]. Therefore, anomaly detection aims to detect patterns or features in data that do not match the expected behavior [17]. In some cases, this desired behavior is established by an explicit definition, but often it is implicitly given through example data. To formalize the description of anomalies, a concept of normality and what is considered to be a deviation from this normality must be specified. In multiple different works on anomaly detection, these definitions are built upon probability theory [16]. The data space given by a specific application shall be

$$\mathcal{X} \subseteq \mathbb{R}^D. \tag{2.12}$$

A concept of normality can be defined as a distribution P^+ on $\mathcal{X}$, which is the ground-truth law of normal behavior in a given task. If an observation significantly deviates from this concept of normality, then this anomaly $\tilde{x} \in \mathcal{X}$ lies in a low probability region under P^+. Under the assumption that an underlying probability density function $p^+(x)$ exists for distribution P^+, the set of anomalies can be defined as

$$\mathcal{A} = \left\{x \in \mathcal{X} \mid p^+(x) \leq \tau\right\}, \quad \tau \geq 0, \tag{2.13}$$

where τ is a threshold defined so that the probability for $x \in \mathcal{A}$ is sufficiently small [16]. In the context of anomaly detection, anomalies, novelties, and outliers are interesting groups of data. Although all three refer to instances $x \in \mathcal{X}$ within low probability regions of P^+, anomalies are often characterized as instances generated by a different process than normal data. Outliers usually refer to rare instances of P^+, and novelties result from new regions of a non-stationary P^+. Following this definition, anomalies are the instances of interest, outliers are usually considered measurement errors or noise, and novelties are new measurements that require updating the models. In this thesis, normal data refers to instances without anomalies. In anomaly detection, one basic assumption is that the region where normal instances occur can be bounded. Therefore, a threshold $\tau \geq 0$ can be defined such that

$$\mathcal{X} \setminus \mathcal{A} = \left\{x \in \mathcal{X} \mid p^+(x) < \tau\right\} \tag{2.14}$$

is dense and not empty. This assumption does not imply that the full support of P^+ is bounded, but a dense subset of this support is. P^+ is not explicitly known in many applications, because the underlying process that generates the data is too complex. Therefore, this distribution P^+ needs to be estimated from the data. From this typical concentration assumption, anomalies are assumed to be not concentrated. Some unsupervised methods, such as kernel density estimation [82], implicitly assume that the anomaly distribution P^- follows a uniform distribution, implying an uninformative prior to P^-. However, in semi-supervised or supervised anomaly detection methods, apriori assumptions about P^- are made to achieve better results. Unsupervised approaches are the only option for such tasks when no specific information about anomalies is known beforehand. Since this is the case for many applications, it represents the most common approach in anomaly detection [16].

2.2.1 Unsupervised Anomaly Detection

For unsupervised anomaly detection, there is only unlabeled data available. The data-generating distribution P of this dataset is often assumed to be the same as the normal distribution P^+. However, in practice, P can contain noise or contamination. In this context, noise refers to some randomness that alters the outcome of the data-generating process, like measurement errors. On the other hand, unknown anomalies contaminate the set of unlabeled data and represent another disturbance. Therefore, in an unsupervised setting, P includes contributions of both P^+ and P^-. One way of formally describing the data-generating distribution is

$$P = x + \epsilon, \quad x \sim (1 - \eta)\, P^+ + \eta P^-, \tag{2.15}$$

where x refers to a noise-free data instance, ϵ is the noise distribution, and η represents the contamination rate [16]. Unsupervised anomaly detection approaches are especially interesting since real-world data comes in an unlabeled form, and the labeling process is expensive. Furthermore, deep learning methods rely on high volumes of data to learn good generalizations of a concept, and in many applications, anomaly data is rare.

2.2.2 Semi-Supervised Anomaly Detection

In semi-supervised anomaly detection, unlabeled and labeled data are available to train a model. Since the labeling process often depends on the specific knowledge of experts, which causes high costs, the amount of unlabeled data is usually more extensive than the labeled data [16, 83]. Including known anomalies can lead to a significant improvement in detection performance [83]. Generally, semi-supervised approaches can be put into one of two groups. Either the training data consists of labeled normal data only, and the core idea is to learn patterns of normal data. Alternatively, the training data contains large-scale unlabeled data and a few annotated anomalies. Then unlabeled data and the labeled anomalies can be exploited to learn a generalization [83]. In some cases, unsupervised anomaly detection methods allow for incremental updates when some anomalies are already found. In many real-world anomaly detection applications, a few labeled data can be collected relatively cheaply [83].

2.2.3 Supervised Anomaly Detection

In supervised anomaly detection, completely labeled data is available for training a model. Under the assumption that both normal and anomalous data are representative of the normal distribution P^+ and the anomalous distribution P^-, this problem can be described as a supervised classification task instead. However, in practice, certain anomalies might be rare or unknown, and depending on the application, the anomalies can drastically vary. Therefore, the labeled anomalous example data is usually not fully representative of all possible anomalies, making it difficult for a supervised learning model to gather sufficient statistical information about anomalous patterns [36].

2.3 Challenges in Anomaly Detection

The challenges of an anomaly detector depend on the application domain and the specific application. In many real-world scenarios, no or only a few labeled anomalous data samples are available during the development of the anomaly detection algorithm [38, 16]. Furthermore, even when anomalous sample data is available, an approach relying on training with those samples will often fail to detect unknown anomalies [36]. The availability of labeled data is a limiting factor regarding the choice of anomaly detection approach. Sections 2.2.1, 2.2.2, and 2.2.3 highlight the main characteristics and differences between unsupervised, semi-supervised, and supervised approaches. As described in [38], the key challenges resulting from the lack of or limited amounts of labeled data can be formulated like this:

- Machine-Learning (ML) methods typically rely on large amounts of data to allow for solid generalizations about the training data. Therefore, the sparsity of training data, often the sparsity of anomalous instances, poses a challenge.
- A significant challenge for training NNs is that real-world data usually comes without any annotations. If left unchanged, this limits the types of approaches. Otherwise, an additional labeling process is necessary.

Nowadays, many anomaly detection approaches are based on different NNs and use ML techniques to solve problems in various application domains [44, 48, 43]. Many ML solutions, especially more complex models, rely on large training datasets [84] to encourage the model to learn useful representative features instead of noises. The term *big data* is often used to refer to datasets' large and distributed nature [85]. Although the performance of various ML approaches improves with larger datasets, *big data* also poses challenges. The five key challenges that come with *big data* are referred to as the five "Vs" in literature [86]:

- **Volume**: Refers to the amount of data that has been gathered and is often viewed as the product of instance size and the dimensionality of the data. Therefore, an approach must consider many instances and extract relevant information from high-dimensional data to take advantage of high-volume datasets.
- **Velocity**: Refers to the acquisition or update rate of the data. This point is particularly interesting for real-time problems and needs to be considered in such applications.
- **Variety**: Refers to different data types like image, audio, or text.
- **Veracity**: Refers to the range of data quality. Of special interest is incomplete or noisy data.
- **Value**: Refers to the value and the gained insights that can be extracted from *big data* analysis and are usually the primary motivation of the analysis in the first place.

2.3.1 Challenges for Deep Learning in Anomaly Detection

Generally, deep learning approaches based on NNs can work with large amounts of data, and often larger training datasets lead to better generalization capabilities of the model and prevent overfitting [48, 71]. In [48], the main challenges tackled by deep learning approaches for anomaly detection are summarized in six points:

- **Low anomaly recall rate:** In many applications, a critical problem in anomaly detection is the rarity of anomalies. Typically, a tradeoff must be considered between the number of normal instances that are wrongly identified as anomalies and the number of rare anomalies that are misinterpreted as normal instances.

- **High dimensional and not-independent data:** Sometimes anomalous data have abnormal characteristics in lower dimensional representations hidden in the original higher dimensional space. For example, assume a four-dimensional point cloud of normal and abnormal data is given. By projecting these points onto a two-dimensional space, it might be easier to find a line separating these projected points than finding a hyperplane separating the points in the original four-dimensional space. Performing the anomaly detection task in a lower dimensional representation has been a popular solution in subspace-based and feature selection-based methods. However, the complex interactions and couplings between multiple features in high dimensional data might be the potential for further performance improvements but still poses a key challenge for anomaly detection. Furthermore, for anomaly detection, the scenarios where instances may depend on each other introduce another source of challenges.

- **Data-efficient learning of normality and abnormality:** Because large datasets containing annotated normal and anomalous data are unavailable for many applications due to high costs, fully-supervised approaches are often unsuitable for anomaly detection. Therefore, much research has recently focused on unsupervised anomaly detection. One major drawback of unsupervised anomaly detection is that no true anomaly data is known, and the models rely entirely on anomaly distribution assumptions. However, in some applications, collecting labeled normal data and a few labeled anomalous data is not difficult and reasonable to further enhance the anomaly detection performance utilizing a semi-supervised approach. The remaining significant challenges are learning normality or abnormality models from a relatively small amount of labeled anomalous data that allow the detection of novel anomalous instances not included in the labeled data.

- **Noise resiliency:** Many semi-supervised approaches are vulnerable to wrongly labeled instances due to the assumption of clean labeled data. Furthermore, large-scale anomaly-contaminated unlabeled data also poses a problem. In this context, mislabeled or unlabeled anomalous data is called noise. Both pose a challenge to semi-supervised approaches. The amount of noise can vary significantly within datasets, and noisy instances can be irregularly distributed in the data space.

- **Complex anomalies:** Most existing methods only consider point anomalies and do not consider conditional or group anomalies. Point anomalies refer to single anomalous data points or instances. If specific data points or instances only are anomalous under certain circumstances, we call them conditional anomalies. Group anomalies are groups of data not conforming to expected behaviors. Furthermore, most approaches focus on data from one data source. One main challenge is that some complex anomalies can only be detected when considering multiple data sources.

- **Anomaly explanation:** Any model might be biased towards a particular subset of a dataset. There may be certain risks when an anomaly detection model is directly used as a black-box model in several domains. Anomaly explanation algorithms provide information about why a specific instance was detected as an anomaly. Experts can then monitor those instances, detect biases and adapt the models accordingly. Developing such explanation algorithms, especially for complex anomalies, is still challenging.

2.4 Visual Anomaly Detection

Since anomaly detection poses a problem in various applications in different domains, researchers and engineers analyzed and implemented several anomaly detection algorithms [17, 87, 42]. Especially in image processing tasks, anomaly detection algorithms are a valuable tool for many applications [88, 89, 45]. Generally, anomaly detection approaches try to extract, characterize, and model patterns from available data. As for all anomaly detection approaches, visual anomaly detection methods can be grouped into either supervised, semi-supervised, or unsupervised approaches. This work mainly focuses on unsupervised methods for anomaly detection. Therefore, the following sections present anomaly detection approaches suitable for unsupervised training. While some of these approaches are exclusively designed for the detection of anomalies, others additionally enable an effective way of anomaly localization. Some anomaly detection methods relevant to machine vision systems are discussed in the following paragraphs. Even though these approaches are not only used for vision-based tasks, the following sections discuss their applicability to visual anomaly detection.

2.4.1 Probabilistic Anomaly Detection

Probabilistic approaches try to estimate the underlying probability density function of the training dataset, where the training data does not include anomalies. Defining thresholds for the resulting distribution and analyzing data helps to determine whether the sample data stems from the same underlying distribution [90]. A requirement for applying such probabilistic approaches is that all training data instances come from the same probability density function. Multiple techniques utilize various underlying statistical properties of datasets that rely on apriori knowledge or data assumptions. One of the main advantages of probabilistic approaches is the statistically justifiable solution of the anomaly detection algorithm. However, choosing appropriate statistical models for a given application is complex, especially for high dimensional data [91]. Furthermore, these methods assume that data follows a particular probability distribution. For many high-dimensional real-world datasets, this premise does not hold [91]. It should be noted that many vision-based applications work with such high-dimensional data. The following sections provide a brief overview of such statistical methods.

Parametric Approaches

Parametric approaches are based on the assumption that all data instances without anomalies are generated from the same probability density function

$$p(x, \theta), \tag{2.16}$$

where x represents one observation, with parameters $\theta \in \Theta$, where θ is finite [90]. During training, appropriate values for the parameters θ are determined. The most popular underlying distribution of continuous variables used is the Gaussian distribution. Many applications need more complex distributions or mixtures of distributions such as the Gamma, Poisson, or Weibull distributions [90]. However, some applications utilize the Gaussian Mixture Model, a combination of multiple normal distributions [92, 93].

Non-Parametric Approaches

Non-Parametric approaches do not rely on assumptions regarding the underlying probability distribution of the training data. Instead, the model changes during the training process to fit the dataset [90]. A rather simple variation of this approach is the use of histograms. The kernel density estimator is another probabilistic technique that does not rely on assumptions of the probability density function. Multiple kernels distributed over the data sample are used to estimate the probability distribution. This approach was used to estimate background probability density functions as proposed in [94].

2.4.2 Reconstruction-based Anomaly Detection

Reconstruction-based algorithms compare the original input image with a reconstruction of this image. In ML-based approaches, the training dataset for such a model ideally only contains normal data or is contaminated by a relatively small number of anomalous instances [38]. This encourages the model to improve at reconstructing normal data, and it will struggle with reconstructing anomalous data. The cumulative difference between the original and reconstructed image usually serves as an indicator of anomalies. This is based on the assumption that the algorithm should struggle with reconstructing anomalies in the input image; therefore, the difference should be significantly larger than in images without such anomalies [38]. Since reconstruction-based methods enable the pixel-wise comparison of the input and output images, regions with large errors indicate an anomaly. Therefore, besides anomaly detection, such approaches can also accomplish anomaly localization efficiently [95]. Typically, reconstruction-based methods like Autoencoders (AEs) or Principal Component Analysis (PCA) perform a dimensionality reduction of input data [96]. These approaches project the input data onto a lower-dimensional space, often called latent space. . This makes them useful for processing high dimensional data and enables the usage of other existing anomaly detection approaches on this latent space representation [91]. Within this category of anomaly detection methods, approaches applicable to images are discussed in more detail.

Principal Component Analysis

PCA is another kind of reconstruction-based algorithm that is utilized in several applications for anomaly detection [97], [98], [99]. It is a data-driven-based approach to anomaly detection, and its basic idea is to eliminate noise and uncommon features [98]. PCA is used for dimensionality reduction and returns a lower dimensional representation of the given dataset. This dimensionality reduction is accomplished by projecting the input data onto a predefined lower dimensional subspace [99]. The new axes, called Principal Component s (PCs), define this lower-dimensional subspace. Usually, there is a collection of PCs that contribute most of the variance in the input data [99]. The number of PCs within this collection is an important parameter that must be fine-tuned to accomplish optimal results [99]. Each PC points in the direction of the maximum variance left in the data after the variance was already accounted for in the preceding PCs [99]. Traditionally PCA is based on linear transformations and is significantly faster than AE-based approaches for the same task, although AEs potentially perform better [70].

Autoencoder

An AE is one kind of NN and a popular choice for many applications. Different variations of AEs were specialized for a wide variety of tasks. Some examples of use cases of AEs are image compression [100], enabling efficient data transmission [101], or the detection of printed circuit board defects [45]. Figure 2.4 demonstrates how reconstructing

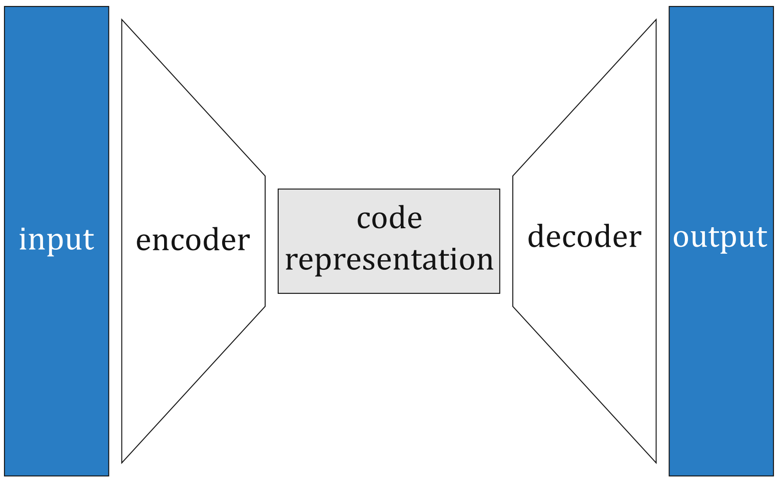


Figure 2.3: This graphic is a visual representation of an under-complete AE, where the encoder generates a lower dimensional code representation of the input.

an anomalous image by an AE enables detecting and localizing anomalies [45]. AEs try to extract meaningful content of the input data without losing essential information [70]. AEs have shown promising results in image anomaly detection [38]. They compress high dimensional input data to a lower dimensional representation to effectively reconstruct the input data by focusing on capturing useful information from this low dimensional representation [38]. In this context, dimensionality refers to the number of values representing the data. For an image, the dimension is the number of pixels, while for a tensor, the dimension refers to the number of elements. AEs consist of an encoder and a decoder. The encoder generates some sort of code representation of the input image, and the decoder tries to reconstruct the original image from this code representation. The space where the encoder projects the input into is often referred to as latent space. Figure 2.3 shows a visualization of such an AE. In general, the encoder and decoder of an AE consist of several layers. The encoder's compression level depends on multiple aspects of the architecture, such as the number of layers. When working with images, usually convolutional layers are utilized, where the number of layers, number of feature maps, and the stride for each layer impact the number of output neurons of the encoder. In convolutional AEs, convolutional layers are used in the encoder and decoder. Every convolutional layer utilizes an activation function, its own weights, and biases. For the most basic convolutional AE, where the encoder and decoder consist of just one layer each, the reconstructed data and latent representation can be described as

$$\tilde{x} = \sigma_d \left(h * W_d + b_d \right) \text{ and} \tag{2.17}$$

$$h = \sigma_e \left(x * W_e + b_e \right), \tag{2.18}$$

where $\tilde{x}$ is the reconstructed image, h is the latent vector, x is the input image, $*$ represents the convolution operator, and $\sigma_e, W_e, b_e, \sigma_d, W_d$, as well as b_d, are the activation function, weight matrix and bias vector of the encoder and decoder network, respectively [102]. For anomaly detection, so-called under-complete AEs, where the encoder calculates a code representation with a smaller dimension than the input, are commonly used [41], [35]. Therefore, the encoder of an under-complete AE performs a dimensionality reduction. However, the encoders of over-complete AEs increase the dimension of the input data. Some sort of regularization has to be implemented to ensure that the overcomplete AE does not perform the identity function. Often noise is added to the input data or random input nodes of the encoder are turned off to accomplish the regularization [45]. For anomaly detection, an AE learns the features and structures of the relevant dataset, which does not contain any anomalies or usually a relatively small number of anomalous data.

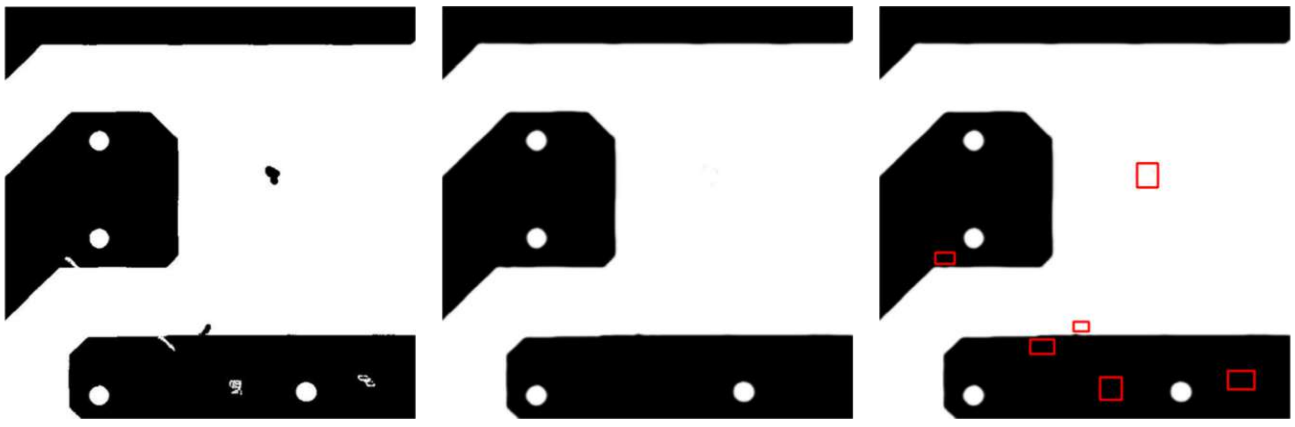

(a) Anomalous input (b) Reconstruction (c) Localization of anomalies

Figure 2.4: Example images for reconstruction-based anomaly detection and localization in printed circuit boards [45].

In contrast to strictly linear methods like PCA, the AE's ability to learn non-linear feature representations depends on the activation functions used after each layer of the model. The performance of an AE for a specific task is influenced by many design choices regarding the architecture of the encoder and decoder networks, such as the number of layers and the kernel size and stride in convolutional layers. Choosing the optimal architecture is not trivial and is still a topic of study [70].

Denoising Autoencoder

Denoising Autoencoder (DAE) is a special case of AE. While the architecture of an AE and DAE can be identical, their training is different. The purpose of DAEs is to remove noise from the input image. During the training of a DAE, noise is intentionally added to the input images. Like a conventional AE, the DAE is encouraged to reconstruct the input image through an appropriate loss function. However, the reconstructed image is not compared with the noisy input image but with the original image without noise. This training methodology encourages the model to extract meaningful features from the training dataset and to ignore the noise. Typically pixel-based noise like Gaussian- or salt and pepper noise is added to the images [45, 101, 103]. However, there are other ways of augmenting input data, which are sometimes referred to as adding some sort of noise. One example is masking certain regions of images to encourage the DAE model to replace those regions with values from contextual information [74]. A more detailed description of the noise sources, their purpose, and examples are given in Section 4.1.2.

2.4.3 Variational Autoencoder

Variational Autoencoders (VAEs) also are a special kind of AE. Although the basic structure of the VAE is similar to a regular AE, the computation of the reconstructed input is non-deterministic. Instead of learning how to construct a latent representation of the input for the decoder, the encoder of a VAE learns to compute vectors that represent the mean and variance of a multivariate Gaussian distribution from which the latent vector is sampled. The decoder then transforms this latent vector back to the original input data space. VAEs are sometimes referred to as both a reconstruction and probability-based approach since the reconstruction process depends on the learned data's probability density function. Because the latent vector is sampled from a probability density function, the VAE is a non-deterministic approach. Figure 2.5 shows a visual representation of a VAE.

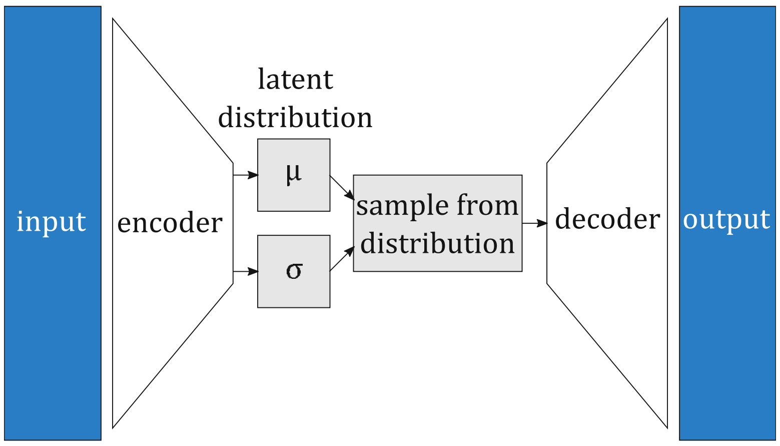


Figure 2.5: This graphic is a visual representation of an VAE. The μ and σ blocks represent the extracted mean and standard deviation vectors.

Diederik P. Kingma and Max Welling published a detailed mathematical description of Auto-Encoding Variational Bayes, the basis for VAEs [104]. The basic idea behind this approach is that a dataset

$$X = \left\{ x^{(i)} \right\}_{i=1}^{N} \tag{2.19}$$

that consists of N independent and identically distributed samples of a continuous or discrete variable x is assumed to be generated by a random process dependent on the unobserved random variable z. This random process consists of two steps. A value $z^{(i)}$ is generated from a prior distribution $p_{\theta^*}(z)$, and a value $x^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(x \mid z)$. Both the prior $p_{\theta^*}(z)$ and likelihood $p_{\theta^*}(x \mid z)$ are instances of parametric families $p_\theta(z)$ and $p_\theta(x \mid z)$, and their probability density functions are differentiable almost everywhere w.r.t. θ and z. The true parameters θ^* and values of the latent variables $z^{(i)}$ are unknown. The proposed approach solves the possible intractability of the integrals involved in calculating of the marginal likelihood $p_\theta(x)$ and the true posterior density $p_\theta(z \mid x)$. Furthermore, the approach is applicable to large datasets. A VAE is a generative model that optimizes the lower bound on the marginal likelihood of each data point in an image [87]. This marginal likelihood consists of the sum over the marginal likelihoods of all data points $\log p_\theta\left(x^{(1)}, \cdots, x^{(N)}\right) = \sum_{i=1}^{N} \log p_\theta\left(x^{(i)}\right)$. In [104], Kingma *et al.* rewrote this expression to

$$\log p_\theta\left(x^{(i)}\right) = D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \parallel p_\theta\left(z \mid x^{(i)}\right)\right) + \mathcal{L}\left(\theta, \phi, x^{(i)}\right). \tag{2.20}$$

In their solution, they introduce a recognition model $q_\phi(z \mid x)$, which approximates the intractable true posterior $p_\theta(z \mid x)$. In [104], the authors refer to $\mathcal{L}\left(\theta, \phi, x^{(i)}\right)$ as the variational lower bound on the marginal likelihood of datapoint i. They introduce a method for learning the recognition parameters ϕ jointly with the generative model parameters θ. The unobserved variable z is comparable to a latent code representation of the input. Therefore, the recognition model $q_\phi(z \mid x)$ can be interpreted as a probabilistic encoder. Similarly, $p_\theta(x \mid z)$ can be viewed as a probabilistic decoder. In a VAE, the latent variables shall be a centered isotropic multivariate Gaussian $p_\theta(z) = \mathcal{N}(z; 0, I)$. Under the assumption

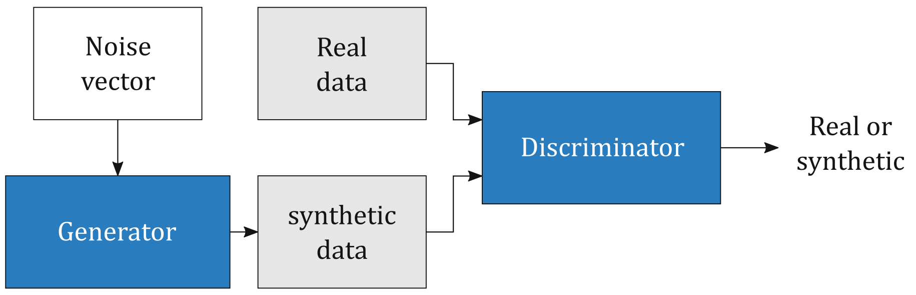


Figure 2.6: This graphic is a visual representation of the basic structure of a GAN. While the generator generates synthetic data, the discriminator network distinguishes between real and synthetic data samples.

that the true posterior takes on an approximate Gaussian form with an approximately diagonal covariance, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure

$$\log q_\phi(z \mid x^{(i)}) = \log \mathcal{N}\left(z; \mu^{(i)}, \sigma^{2(i)} I\right), \tag{2.21}$$

where the mean $\mu^{(i)}$ and the standard deviation $\sigma^{2(i)}$ of the approximate posterior are outputs of an encoding fully connected NN. Now a latent vector can be sampled from the posterior $z^{(i,l)} \sim q_\phi\left(z \mid x^{(i)}\right)$ using $z^{(i,l)} = g_\phi\left(x^{(i)}, \epsilon^{(l)}\right) = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}$ where $\epsilon^{(l)} \sim \mathcal{N}(0, I)$ ($\odot$ represents an element-wise product). This reparameterization is used to rewrite an expectation w.r.t. $q_\phi(z \mid x)$ such that the estimate of the expectation is differentiable w.r.t. ϕ [104]. Therefore, we can obtain a differentiable estimator of the variational lower bound

$$\mathcal{L}\left(\theta, \phi, x^{(i)}\right) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left(\left(\sigma_j^{(i)}\right)^2\right) - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta\left(x^{(i)} \mid z^{(i,l)}\right). \tag{2.22}$$

Similar to AEs, VAEs are used for anomaly detection in various applications like anomaly detection for solder joints [87], linear motion guides [105], and the detection of ships in synthetic aperture radar images [106].

2.4.4 GANs

The basic idea behind GANs is two-person zero-sum game theory [107]. There are different specialized versions of GANs for specific purposes and applications, but generally, a GAN consists of two networks: a generator and a discriminator. Figure 2.6 shows the basic structure of a GAN. The generator generates synthetic data samples to fool the discriminator, while the latter tries to distinguish this generated from real data. Both of these modules try to improve their task during the training process. [108]. GANs have been used for a variety of applications, like the segmentation within brain tumor images [109], face detection applications [110], and a variety of anomaly detection tasks [111, 107]. Depending on the data type, dimensionality, and task at hand, the architectures of the generator and discriminator networks vary. For example, Radford *et al.* proposed a class of CNNs specialized for unsupervised learning for various image datasets [112]. Since these GAN models are based on deep CNNs, Radford *et al.* named this class of CNNs Deep Convolutional Generative Adversarial Network (DCGAN).

Since the generator network is trained to generate data to fool the discriminator, the generator may be used to produce synthetic data. This ability to generate data may be advantageous for some applications where only insufficient data is available for training and evaluating a model or when real data should not be used for testing due to security or privacy risks [113, 114]. There are different approaches to utilizing the models of a GAN for anomaly detection. For example, inverse GANs for anomaly detection utilize the generator network [115, 116]. In principle, such inverse GAN approaches fall into the category of reconstruction-based methods. The anomaly detection task is tackled by solving an optimization problem. An input vector for the generator must be found to minimize the error between a given image and the generated image. Since the generator learns the distribution of normal data during training, it should struggle to generate anomalous images. One disadvantage of such approaches is that this optimization task is typically computationally expensive. This disadvantage renders such approaches impractical for applications with high dimensional data and low inference times. Since Vision-based Anomaly Detection Algorithm for Railroads (VADAR) should run on an embedded system installed on a moving train and analyze images in real-time such approaches are infeasible for this task. Another GAN-based anomaly detection approach utilizes the discriminator. Since the discriminator sees only normal and synthetically generated images that imitate normal data during training, the discriminator might also be able to detect anomalous images. A big advantage of this method is its relatively fast inference time. However, according to [116], this approach typically does not perform well in comparison with approaches utilizing the generator network.

2.5 Anomaly Detection and Inspection for Railroad Systems

Several research groups have already worked on different methods for automatically inspecting railroad tracks [35, 117, 118, 119]. Most of the reviewed works focus on one specific track element and have specialized sensor systems that enable the inspection of the relevant regions of the rail tracks. Figure 2.7 shows basic rail track elements and regions of the trackbed relevant to these systems. First, this section gives a brief overview of railroad datasets. Then the section closes with comparisons and analysis of several state-of-the-art railroad inspection systems.

2.5.1 Railroad Datasets

For the development of anomaly detection algorithms, datasets are essential. To the best of the author's knowledge, several unlabeled and three annotated railroad datasets were already introduced in the literature. In contrast to supervised approaches, unsupervised methods do not rely on annotated data for training. However, annotated data is necessary to evaluate any anomaly detection algorithm. Therefore, datasets without annotations, like the *Nordland*-dataset [120], are unsuitable for developing an anomaly detection system. Therefore, this section focuses on the three labeled datasets.

- **RailSem19**: Zendel *et al.* introduced a publicly available dataset [121]. Its original purpose is semantic scene understanding for trains and tramways. It consists of 8,500 annotated sequences recorded out of the cabin of trans. Unfortunately, this dataset does not contain annotations for anomalies but for elements like rails, switches, traffic signals, trains, and platforms.

- **Kaggle Railroad Track Fault Detection**: This publicly available dataset contains 384 images of rails and other railroad infrastructure. Half of the images contain anomalies [122]. However, the images are recorded from several different perspectives. Because VADAR is supposed to work with images recorded from one fixed perspective, this dataset is not suitable.

- **Vesuvio**: For the development of a railroad inspection system, Gasparini *et al.* recorded the *Vesuvio*-dataset [35]. A thermal camera, a stereo camera system, and an industrial RGB camera recorded images of rail tracks at night. Since their anomaly detection system should detect large foreign objects on the rail track, such as pickaxes, traffic lights, and LPG tanks, these objects were placed on the rail tracks for the recordings. Unfortunately, this dataset lacks annotations for smaller objects, vegetation, and rail damages. Furthermore, the dataset is not publicly available and was not provided to us upon request.

Because these datasets are unsuitable for developing VADAR, the *Kombi* dataset was used. This dataset is the property of the company *Mission Embedded* [123]. Section 3 describes this dataset in detail.

2.5.2 State-of-the-Art Anomaly Detection for Railroad Systems

In [117] and [124], methods for inspecting railroad fasteners were introduced. Railroad fasteners are used to fix the rails to the crossties. Fasteners will loosen, get damaged, or even detach under the long-term effects of vibration and temperature changes. These changes could potentially result in catastrophic events like derailment accidents. Inspecting and monitoring railroad fasteners is important to ensure the safety and efficiency of railroad systems. In [124], a system of six line scan cameras from a top-view perspective with different angles was used to observe the railroad fasteners. A fastener localization module with an average detection rate of 99.36% and a similarity-based deep CNN for classification was introduced. Another system introduced in [117] used a 3D camera system. It approached the defect detection task with a combination of PCA and a histogram-based similarity approach, where the peak difference values of the fastener images were analyzed and used as an indicator for damages. A system described in [89] focuses on inspecting crossties and fasteners and utilizes single-view line-scan cameras. A deep CNN was used for detecting good, broken, or missing fasteners, and semantic segmentation for detecting chips and crumbling concrete ties and other material classes. Du *et al.* introduced a system for detecting railroad plug defects [119]. Plugs are important components that are used to transmit control information signals, and their defects can impact the safety of railroad systems. The hardware part includes two high-speed digital cameras that capture a series of images. The images are analyzed by a change detection framework which consists of an object location, image alignment, and similarity computation module. The work in [118] focuses on rail damage detection. High-speed or intensive rolling of wheels significantly increases the probability of rail damage and poses a major safety concern. Therefore, detecting rail rolling contact fatigue cracks is necessary to ensure the safety of railroad transportation. Instead of conventional cameras, this system relies on dynamic electromagnetic thermography for the sensor system. While an excitation coil is close to the rail, eddy-current will be induced in the surface of the rail and generate Joule heat in the process. Since the heat distribution of the surface will be disturbed by cracks on the rail, an infrared camera is used to record this surface heat distribution. The paper [125] presents a method for monitoring the vegetation on railroad embankments. The main reason for vegetation control on and along rail tracks is safety for passengers and staff members. A camera was used as a sensor to acquire images of the tracks from a birds-eye view. The detection method is based on color space transformations and filters applied to separate channels of the hue-saturation-value space. In [126], a method for the railroad track gauge inspection was introduced. Railroad track gauge irregularities negatively impact the service life of rails and the vehicles using the rails. Such irregularities can even result in driving accidents like wheel trapping or derailing. The sensor system consists of four cameras and two red laser sector lights that are used to obtain spatial information on both sides of the railroad track rails. This information is then used to determine the gauge parameter. The system introduced in [35] can detect construction tools lying on the track bed of a railroad system. Obstacles, especially larger ones, placed on the rail tracks could cause

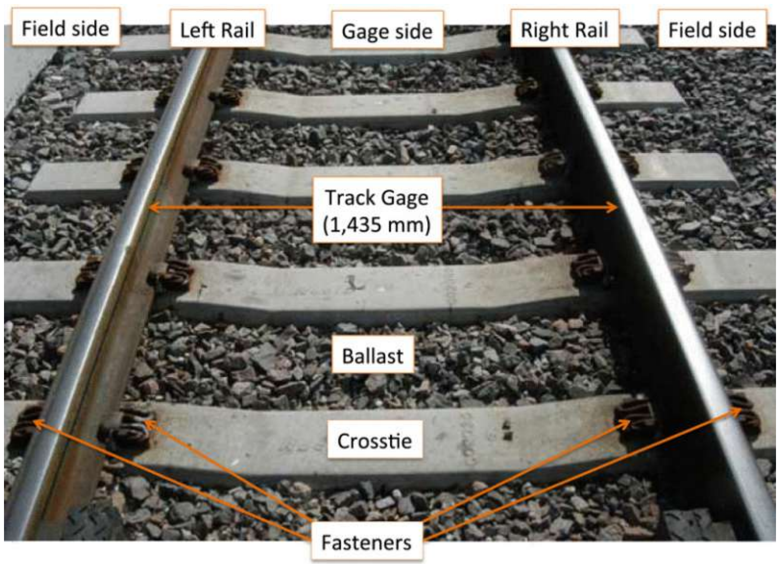


Figure 2.7: Definition of basic track elements [89]. Fasteners connect the rails to the crossties. Ballast refers to the stones underneath and between the crossties and forms the trackbed.

damage to the infrastructure or even the derailment of trains. The camera system was mounted on a rail drone, and images were taken from a front-view perspective of the rail tracks. The detection method combines an AE and a CNN-based classifier. The method was tested with multiple RGB and infrared cameras. Table 2.1 gives an overview of the before-mentioned railroad inspection systems, the anomalies of interest, and some information regarding the sensor system and the detection method.

In [35] and [127], Gasparini *et al.* propose an anomaly detection system for detecting construction tools on the trackbed of railroad systems. This approach seems to be a promising basis for further improvements because it works with only one camera, effectively deals with large amounts of data, and allows the analysis of the entire trackbed. Therefore, the following paragraph explains and discusses this approach in more detail.
RGB and infrared cameras mounted on a rail drone obtain images of the trackbed from a front-view perspective. Figure 2.8 gives an overview of the proposed anomaly detection algorithm. This approach analyzes the reconstruction error of an AE. Both the absolute error image and gradient error image between the original input image and the reconstruction image are fed to a binary classification network. This network decides if the image contains an anomaly or not. They trained the AE using images from the *Vesuvio* dataset, which is explained in more detail in Section 2.5.1, specifically created for their use case. This dataset contains more than 30,000 images obtained during the night from several camera systems, including thermal and RGB cameras. Although the dataset includes anomalous instances, these anomalies are large objects like construction tools. It does not include annotations for rail damages or smaller objects on the trackbed. For their RGB camera setup, they achieved an accuracy of 81.1%, a precision of 97.9%, a recall rate of anomalies of 71.9%, and a resulting F1-score of 82.5% [127]. However, during the night setting, the infrared camera setup outperforms the RGB camera setup with an achieved accuracy of 96.6%, a precision of 98.9%, a recall rate of anomalies of 95.7%, and a resulting F1-score of 97.3%.

Table 2.1: Comparison of multiple railroad inspection systems.

Railroad Inspection Systems				
Paper	Anomalies	Sensors	Perspective	Detection Method
[119]	Plug defects	Cameras	Top view, multiple angles	Change Detection
[89]	Ties and fasteners	Cameras	Top view	Deep CNN
[118]	Rail damages	Electromagnetic Thermography	Top view	Corner-Net
[124]	Fasteners	Cameras	Top view, multiple angles	Deep CNN
[125]	Vegetation	Cameras	Top View	Colour filtering
[117]	Fasteners	3D Camera	Angled	PCA + DP
[126]	Track gauge	Cameras + Laser sector lights	Top view, multiple angles	Segmentation, Extraction
[35, 127]	Construction tools	Cameras (infrared and RGB)	Front view	AE + CNN

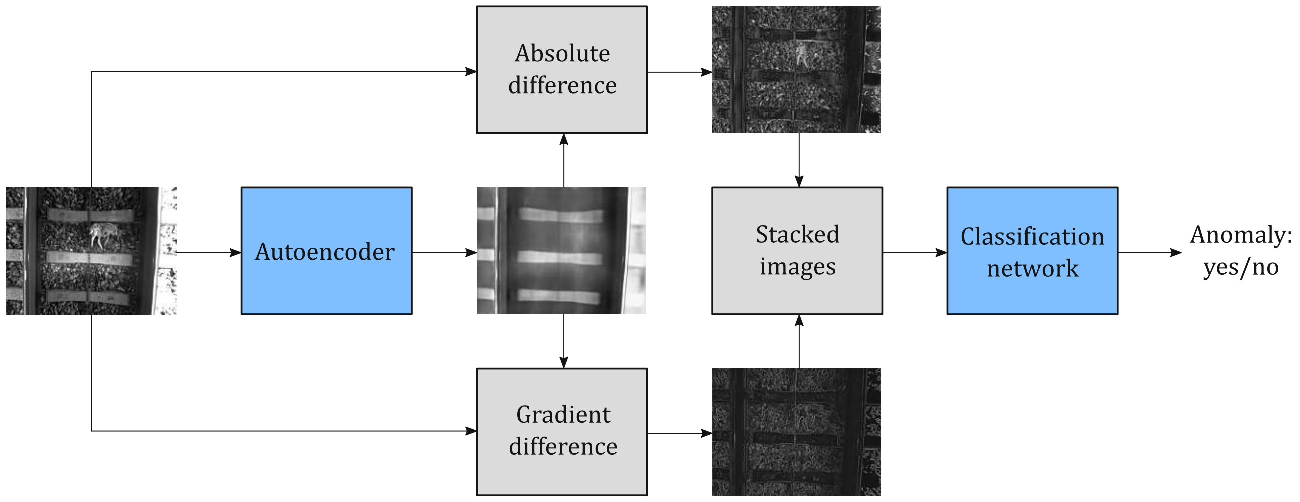


Figure 2.8: For this approach two NNs are used. In the first step, an AE is used to reconstruct the input image. Then the absolute and gradient error images are fed to a binary classification network. This classification network decides if the input image is anomalous [35].

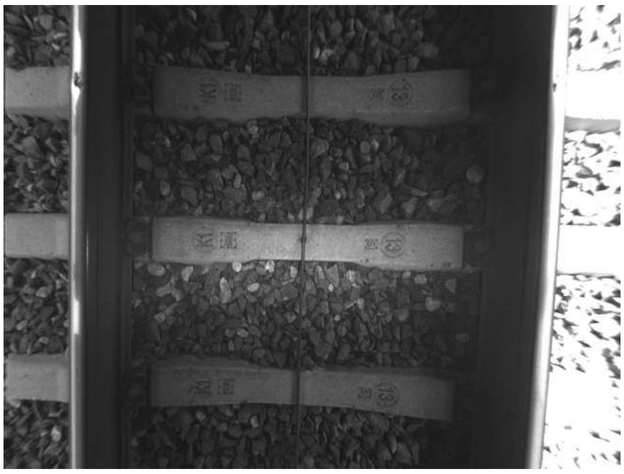
Chapter 3

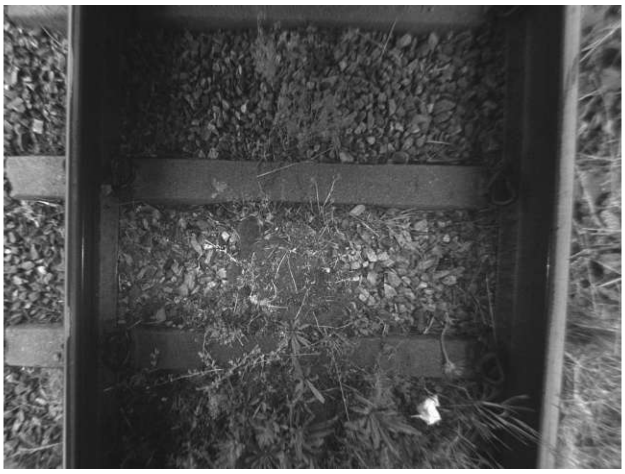
Experimental Dataset

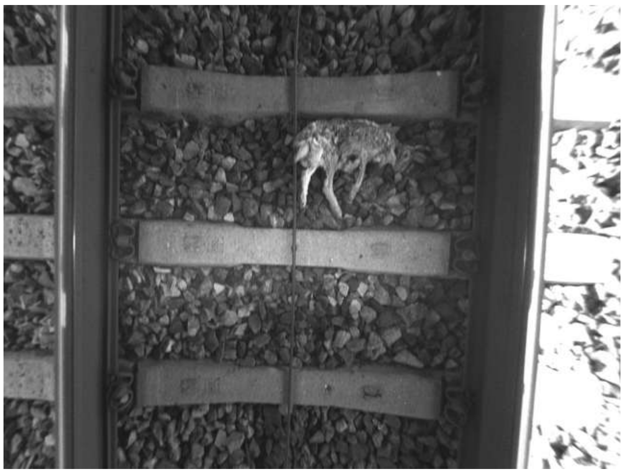
This work focuses explicitly on a dataset containing images of rails and the trackbed of railroad systems from a birds-eye view perspective. The sensor system should be cost-efficient for installation on regular trains and enable the monitoring of railroad infrastructure. All system parts must satisfy railroad industry standards to be permitted for usage [128]. There are constraints regarding the size of the sensors to be installed on regular trains. Sensors installed on the outside of the train need to withstand the maximum train speed and simultaneously provide useful data under different lighting and weather conditions. From now on, the dataset obtained from this camera system is referred to as *Kombi* dataset. It should be noted, that the training and evaluation of the algorithm introduced in Section 4.3 only take images of the *Kombi* dataset into consideration. The following sections present more information about the *Kombi* dataset and applied data augmentation techniques. This chapter closes with some thoughts regarding suitable anomaly detection approaches.

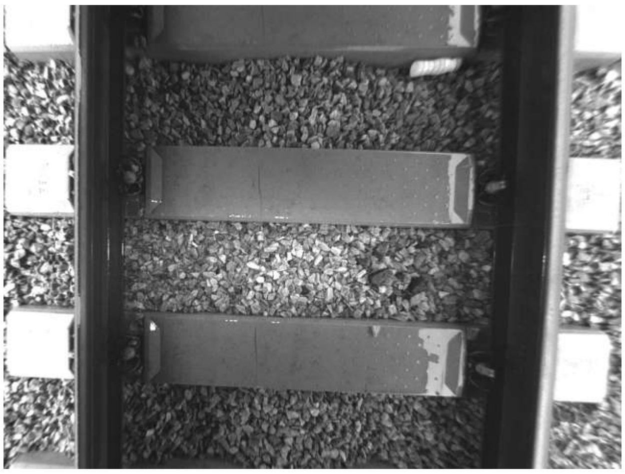
3.1 Kombi-Dataset

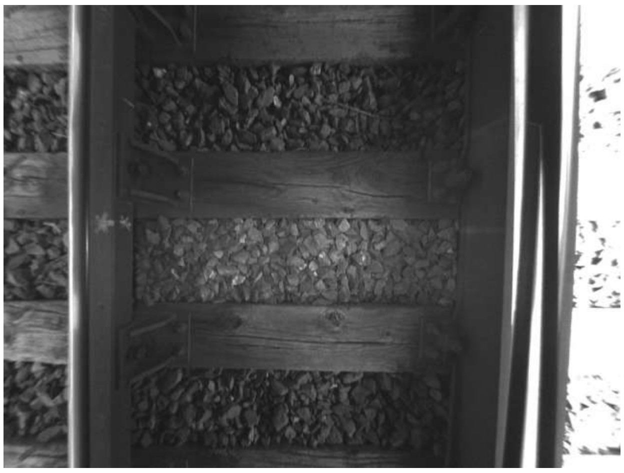
This dataset was collected in the context of a research project in 2019 [129]. The project aimed to determine a cost-efficient sensor system suitable for installation in regular trains. Since the used camera has a fisheye lens, some regions of the images far away from the center are distorted or contain black regions. These regions can be cropped and ignored. Figure 3.4 shows two example images of the dataset before cropping. In a pre-processing step described in Section 4.3.1, these images are prepared for Vision-based Anomaly Detection Algorithm for Railroads (VADAR). The *Kombi* dataset consists of roughly 2,000,000 images and is only partially labeled. Two cameras simultaneously recorded the images of this dataset. Therefore, half of the images show the same rail tracks but from a slightly different perspective. The resolution of the uncropped images is 1,600 by 1,200 pixels. A rough estimation for the area one pixel corresponds with is 2 mm^2. Besides anomalies of interest like damages to the rails and crossties, unexpected objects like trash or animal corpses, and heavy vegetation, infrastructure elements like switch frogs and sensors were also labeled. In total, roughly 220,000 frames were labeled. More than 55,000 infrastructure elements (Table 3.1) and roughly 22,000 anomalies (Table 3.2) were annotated within this labeled part of the dataset. Infrastructure elements include any element of the infrastructure, but elements that are seen in (almost) every image, like the left and right rail, crossties, fasteners, and


(a) Rail damages are of particular interest for maintenance.


(b) Heavy vegetation on the trackbed and next to the rails.


(c) An animal corpse lies on the trackbed and could attract other and bigger animals.


(d) Some frames of the dataset include trash, e.g., bottles, on or next to the trackbed.


(e) At the center of a switch between two tracks is a so-called switch frog.

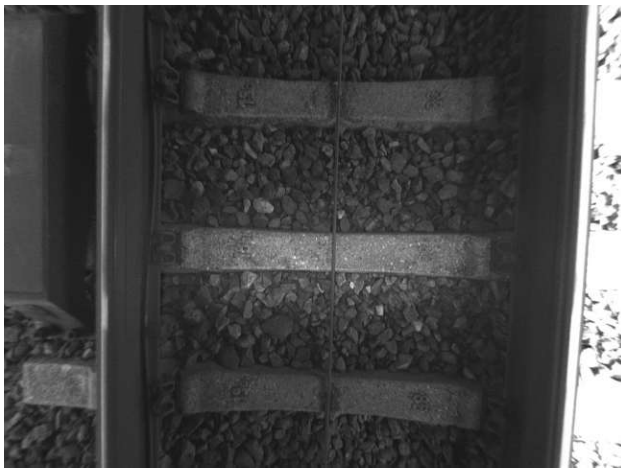

(f) In some frames of the dataset boxes are placed next to the rails.

Figure 3.1: The images 3.1a and 3.1b show examples of rail damages and vegetation on the trackbed. Different kinds of unexpected objects like trash or dead animals can be found on the trackbed. Two examples of such objects are shown in the frames 3.1c and 3.1d. Along a railroad track some additional infrastructure elements like switch frogs 3.1e or boxes 3.1f are installed.

cables. Frames can have multiple annotations associated with them. A large portion of the anomalies corresponds to image errors that are not of interest or small-scale anomalies like small pieces of trash or little vegetation. Tables 3.1 and 3.2 list the number of annotations for each class within the *Kombi* dataset.

The two classes of annotated infrastructure elements are *Crosstie-* and *Rail attachments*. These classes include elements that are permanently fixed to the crossties and rails. Boxes on the left or right side of the trackbed belong to the class *Box*. Figure 3.1e shows an image of a *Switch frog*. Images close to a switch frog, where a second rail track crosses the trackbed, belong to the class *Switch*. If a third rail appears on the rail track's left or right side, this rail is annotated as a *Different rail*. *Spacers* are infrastructure elements placed to keep a defined distance from surrounding infrastructure. Sensors mounted on the side of rails belong to the *Sensor* class. All other infrastructure elements of interest are annotated as *Other*. The differences between normal images and images with infrastructure elements can be considerable. These differences can be more significant than those between normal and anomalous frames. Therefore, infrastructure elements pose a challenge to the anomaly detection method in achieving a reasonable balance between the number of false positives and correctly detected anomalies. Another challenge for anomaly detection is the random nature of the ballast in the track bed. Since in most frames of this dataset, ballast (gravel) covers the trackbed, the anomaly detection method should be able to distinguish between gravel and unexpected objects on the trackbed. Since the stones' shape, size, and brightness vary, the anomaly detection method needs to consider that.

Figure 3.1 shows some example frames of the dataset: damaged rails, heavy vegetation, and unexpected objects on the trackbed. Even without any damages, objects, or additional infrastructure, there is a wide variety within frames because of different lighting conditions, the appearance of ballast, and the crossties' shapes and material.

The following subsections explain the main challenges this dataset poses to designing VADAR and briefly describe how anomalies and other image characteristics are annotated.

3.1.1 Varying Appearance of the Trackbed

Since the dataset consists of frames from different railroad tracks, the appearance of the trackbed varies drastically. The size, shape, and material of crossties do change within the *Kombi* dataset, and the same is true for the size and color of the ballast on the trackbed. Therefore, normal data is vast, and the anomaly detection algorithm must be able to deal with that to avoid an unreasonably high false positive rate. While the *Kombi* dataset holds annotations for anomalies and infrastructure elements, it also includes annotations for crosstie material, lighting conditions, and the type of ground. Table 3.3 lists the number of images falling into each category. All images were put into one of three categories for ambient lighting: *Daylight*, *Dark*, or *Mixed*. The category *Mixed* contains images at the entrance or exit of a tunnel where only one part of the image is in bright daylight, and the rest is dark.

Ballast

One big challenge this dataset poses is the gravel. The random nature of each stone's size, shape, brightness, and arrangement significantly increases the difficulty of detecting objects placed on the ballast. The size of the separate stones may limit the size of detectable anomalies. Furthermore, if significantly darker or brighter stones surround multiple stones of similar brightness, this group of stones could also lead to a false positive of an anomaly detector.

Lighting conditions

In many frames, direct sunlight makes the right or left side next to the trackbed very bright. The partial shadows of the gravel on those brighter sides lead to irregular patterns of high contrast and, therefore, introduce additional complex features within the dataset. Furthermore, in some frames lighting artifacts may appear on the rails or the trackbed when light is coming from a specific angle. The ambient lighting conditions of each image were categorized into one of three groups *Daylight*, *Dark*, and *Mixed*. Information regarding the lighting conditions of frames is listed in Table 3.3. Since all images were taken during the day, most images fall into the *Daylight* category. A small portion of images taken inside tunnels was categorized as *Dark*. At the entry or exit of a tunnel, frames were annotated with a *Mixed* lighting condition.

3.1.2 Rail anomalies

Rail damages are especially interesting for maintenance because they need to be repaired in order to ensure the safety of transportation, prevent more severe damages, and save maintenance costs [30, 25]. Even small rail damages can lead to more significant damages over time [30]. In order to detect even small rail anomalies, rails could be analyzed separately from the rest of the trackbed. Table 3.2 lists the number of annotated rail damages and other anomalies like foreign objects and vegetation.

3.1.3 Infrastructure elements

Most samples in the dataset are frames of a single rail track where only two rails and the trackbed with the crossties and ballast are visible. However, a smaller but significant portion of the dataset contains frames with additional infrastructure elements. The interpretation of such additional infrastructure elements as anomalies should be prevented when these elements are not damaged. Table 3.1 lists the annotated infrastructure classes and the corresponding number of instances labeled in the *Kombi* dataset.

Class	Instances	Class	Instances
Crosstie attachment	23,965	Switch	13,313
Rail attachment	4,720	Switch positioner	2,123
Different rail	3,468	Switch frog	1,075
Box	2,900	Spacer	674
Other	3,684	Sensor	370

Table 3.1: The classes *Crosstie-* and *Rail attachments* include elements that are permanently fixed to the crossties and rails. Boxes on the left or right side of the trackbed belong to the class *Box*. Figure 3.1e shows an image of a *Switch frog*. Images close to a switch frog, where a second rail track crosses the trackbed, belong to the class *Switch*. If a third rail appears on the rail track's left or right side, this rail is annotated as a *Different rail*. *Spacers* are infrastructure elements placed to keep a defined distance from surrounding infrastructure. Sensors mounted on the side of rails belong to the *Sensor* class. All other infrastructure elements of interest are annotated as *Other*.

Damages		Foreign objects	
Class	Instances	Class	Instances
Vegetation	11,408	Bottle	242
Image error	4,833	Can	195
Damage to crosstie or ground	1,473	Animal	19
Rail damage	507	Other	3.294
Other	13		

Table 3.2: The group of anomalies, like damages and foreign objects, consists of five and four different classes, respectively.

3.1.4 Annotations

For each image in this dataset that includes either an infrastructure element of interest or an anomaly like some sort of damage or an unexpected object on the trackbed an annotation file exists. This file contains the class and location information of the elements of interest with coordinates and the width and height of the bounding box. Figure 3.4 shows two example images of the *Kombi* dataset with bounding boxes around a box and a can. Tables 3.1 and 3.2 list

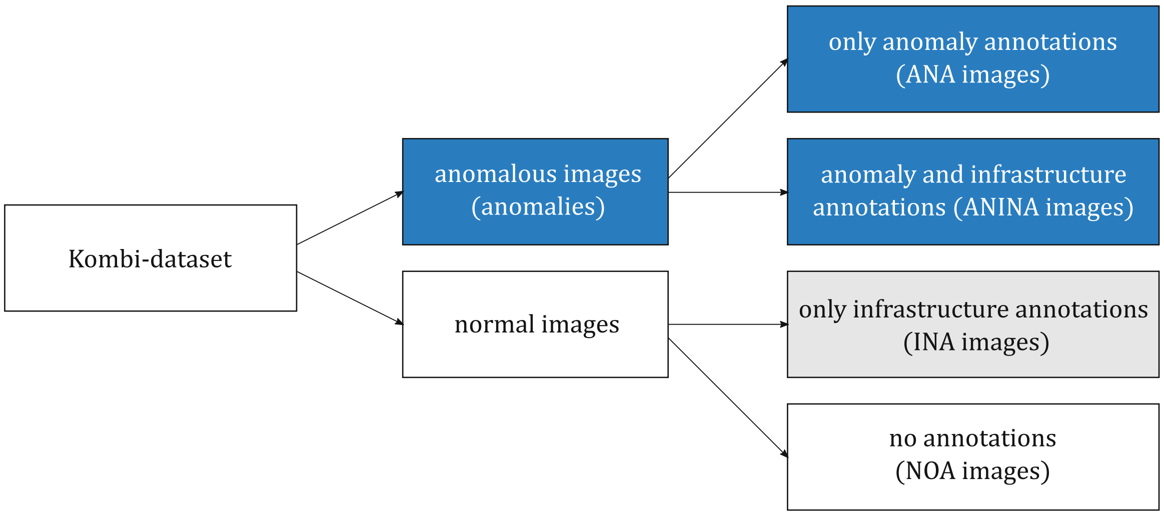

Kombi-dataset
anomalous images (anomalies)
normal images
only anomaly annotations (ANA images)
anomaly and infrastructure annotations (ANINA images)
only infrastructure annotations (INA images)
no annotations (NOA images)

Figure 3.2: In case an image of the *Kombi* dataset contains at least one anomaly annotation the image is considered to be anomalous and is referred to as an anomaly. If an image does not contain any anomaly annotations it is a normal image. Within these two groups, we further distinguish between images with and without infrastructure annotations. Therefore, four additional subgroups are defined.

this dataset's different infrastructure elements, damages, and foreign objects. Figure 3.3 shows additional information regarding the sizes of anomalies. The anomaly sizes are defined as the number of pixels within the according bounding box. This bounding box size can be misleading for a few anomalies, though. Since the *Kombi* dataset contains a few sequences of images where hundreds of consecutive images show some vegetation across the whole image, often one single bounding box was defined to include the entire trackbed. Therefore, the anomaly sizes are just an approximation.

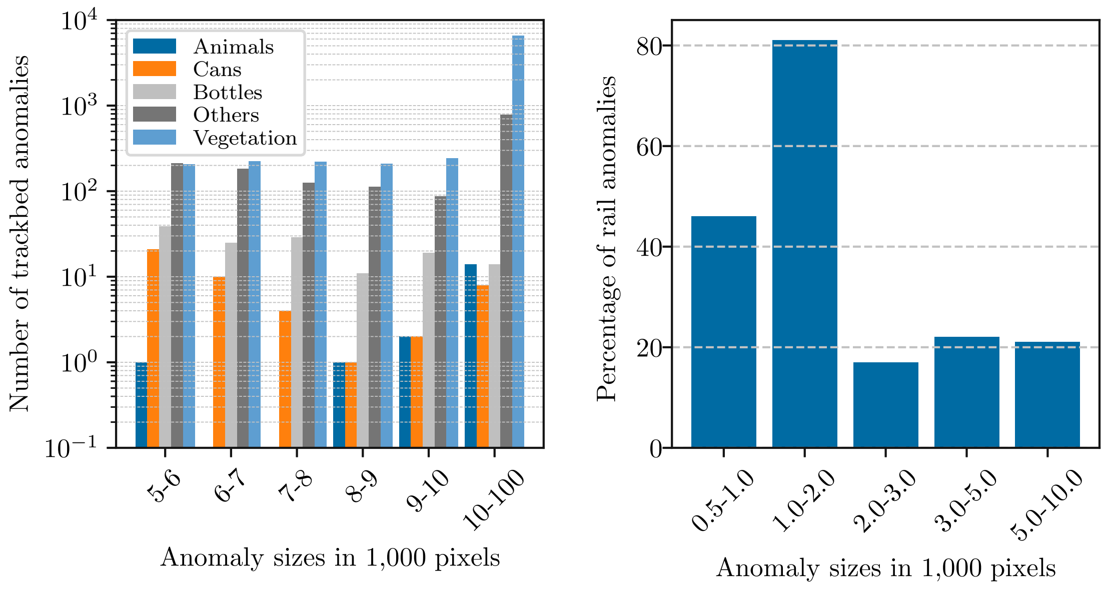
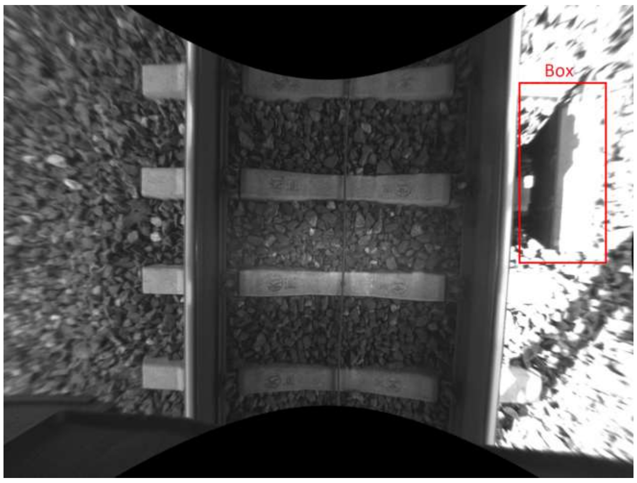
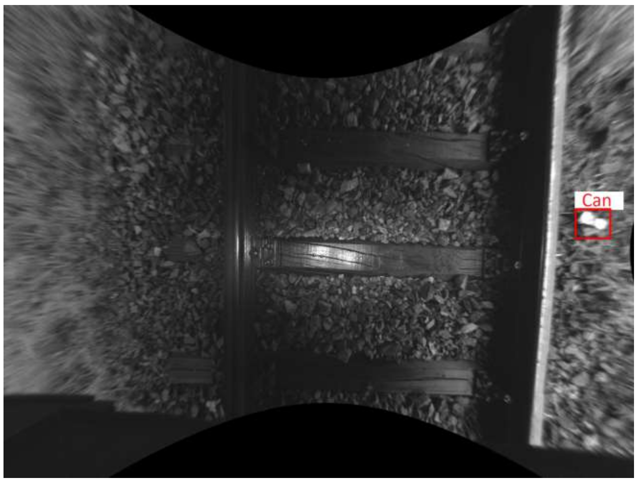


Figure 3.3: The left figure shows the relative numbers of objects and other anomalies on the trackbed with specific sizes. On the right side, the distribution of rail anomaly sizes is shown.

The largest rail damages with anomaly sizes of several thousands of pixels correspond to corrugations–these quasi-sinusoidal irregularities on the rails with a wavelength of less than one meter [130]. Corrugations are typically visible on several tens or hundreds of consecutive frames. The bounding box was defined on the entire rail head section in these cases. The left part of Figure 3.3 shows the size distribution of several anomaly object classes on the trackbed. By far, most anomaly instances belong to the anomaly class vegetation. As mentioned before, in many cases, only one bounding box was used to annotate multiple separate instances of vegetation. Therefore, only one large bounding box was used instead of multiple smaller ones.



(a) Bounding box for an infrastructure element.



(b) Bounding box for an anomaly.

Figure 3.4: Figures (a) and (b) show example images from the *Kombi* dataset with an annotated box and can, respectively. Due to the fisheye lens of the camera system, some regions on the border of the images are distorted or include black regions. Therefore, images were cropped to ignore these parts of the images.

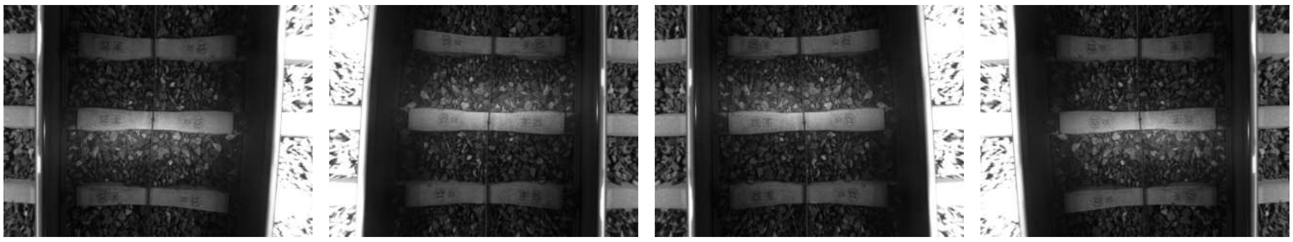
Depending on the annotations, an image of the *Kombi* dataset has it belongs to specific groups of images. Figure 3.2 shows how these groups are defined. Every anomalous image belongs to the group of image containing only anomaly annotationss (ANA images) or image containing anomaly and infrastructure annotationss (ANINA images). Normal images are a member of the group image containing no annotationss (NOA images) or image containing only infrastructure annotationss (INA images).

Furthermore, one additional file stores information regarding the lighting conditions, type of crossties, and the ground for each frame. The total number of images for each such scenario is listed in Table 3.3. The overwhelming majority of images show crossties made of concrete or wood on gravel in daylight.

Railroad Crossties		Ground Types		Ambient Lighting	
Class	Instances	Class	Instances	Class	Instances
Concrete	185,128	Gravel	226,845	Daylight	227,311
Wood	42,552	Railroad Crossing	2,255	Dark	2,324
Mixed	1787	Mixed	466	Mixed	8
Metal	74	Bridge	50		
Hardrubber	26	Asphalt	9		
None	8	Unknown	18		
Unknown	68				

Table 3.3: This table lists the number of annotated images within the *Kombi* dataset that fit the defined categories of railroad crossties, ground types, and ambient lighting conditions.

3.2 Data Augmentation



(a) Original image. (b) Image rotated by 180°. (c) Vertically flipped image. (d) Horizontally flipped image.

Figure 3.5: Applying a 180° rotation and vertically and horizontally flipping the original image quadruples the available data for training.

Several different data augmentation methods were applied to expand the training data set and improve the training procedure's quality. Images were rotated by 180° and added to the training data set to increase training data. Furthermore, the images were flipped vertically and horizontally. In total, the number of available training data was quadrupled by these three augmentations. In Figure 3.5, the resulting augmented images obtained from one original image are shown. Figure 3.5a shows the original unaltered image, while Figures 3.5b to 3.5d show the rotated, vertically flipped, and horizontally flipped images, respectively.

Because of varying lighting conditions, some parts of images are dark and result in low-contrast regions. Generally, low-contrast regions pose a challenge and often limit the feature extraction process. Contrast Limited Adaptive Histogram Equalization (CLAHE) was applied as a preprocessing step to combat these problems. Two images with and without this preprocessing step are shown in Figure 3.6.

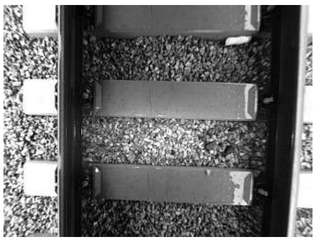
3.2.1 Synthetic Data

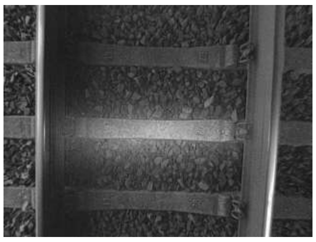
Synthetic data can be used to train or validate models and is a valuable option when a given dataset does not contain enough samples. One method of creating synthetic data is to use the generator network of a Generative Adversarial Network (GAN). As explained in Section 2.4.4, during training, the generator of a GAN learns how to create samples that look realistic to fool the discriminator. Therefore, the generator gradually improves in creating synthetic data. The GAN approach of generating synthetic data seems to work for generating synthetic normal data. However, the generation of synthetic images with rail damage was not successful. One reason could be that rail damages are typically small and contained within less than 1% of the image pixels. Furthermore, the training relies on a rather large number of available real data samples. While the *Kombi* dataset contains more than 220,000 labeled images, only a few hundred images contain damaged rails. Rail damages of special interest, like breakouts and indentations, are even rarer. The labeled part of the *Kombi* dataset includes roughly 150 such damages. For the validation of an anomaly detection algorithm, the more anomalous samples available, the better because several aspects of the anomaly most likely influence the algorithm's performance. The position, size, and orientation of the anomaly and the surroundings and lighting conditions might influence the recall rate of an anomaly.

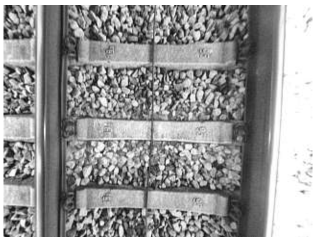
Another option to replicate known rail damages is to copy and paste such damages onto images with rails without damages. However, due to different lighting conditions and the rails' conditions, the inserted damages do not look realistic. To make these patches of rail damage fit in on images of rails, these images are fed to an AE with a high dimensional

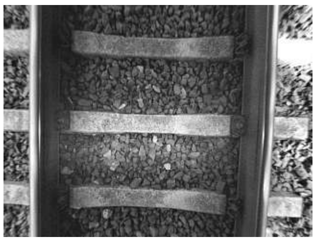
(a) Unaltered original image.

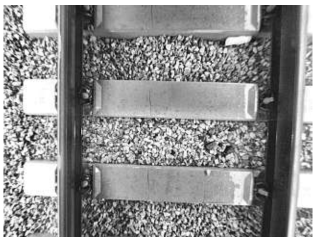
(b) Unaltered original image.


(c) Unaltered original image.


(d) Unaltered original image.


(e) CLAHE applied on (a).


(f) CLAHE applied on (b).


(g) CLAHE applied on (c)

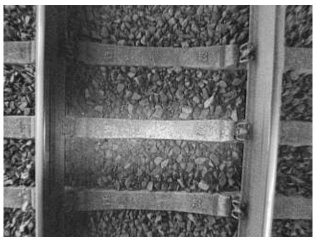

(h) CLAHE applied on (d).

Figure 3.6: In Subfigures e to h, the results of applying the CLAHE algorithm on the images a to d are shown, respectively. This preprocessing step helps equalize the image's brightness and increase the contrast.

Table 3.4: All layers are two-dimensional convolutional layers implemented in PyTorch. A batch normalization layer follows each convolutional layer. The input image has a width of 1,025 and a height of 769.

Encoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1	4	3	1	1	769	1025
2	4	8	3	2	1	385	513
3	8	16	3	2	1	193	257
4	16	32	3	2	1	97	129
5	32	32	3	2	1	49	65

bottleneck. This Autoencoder (AE) reconstructs the given image with high detail. Therefore, this model is named High Detail Autoencoder (HDAE). Tables 3.4 and 3.5 show the HDAE's encoder and decoder architectures, respectively. The damage blends into the surrounding rail better within the output image, and a small patch of this synthetic damage is copied and inserted into the original image. The resulting output image of this procedure is the original normal image with synthetic rail damage that looks more realistic than when the real damage is simply copied and directly inserted in another image. Figure 3.7 visualizes this generation process of synthetic rail damages. Additionally, by rotating, mirroring, resizing, and adapting the brightness values, the original damage patch can be augmented to change the appearance of the damage.

Figures 3.8 and 3.9 show examples of synthetic images with replicated rail damages. A small rectangular crop around the original damage in the top left image was inserted in various positions on the rail of normal images. Additionally, this damage crop was rotated by multiples of 90° and mirrored randomly.

Table 3.5: All layers are two-dimensional transposed convolutional layers implemented in PyTorch. A batch normalization layer follows each transposed convolutional layer. The decoder network is symmetric to the encoder network regarding the layer structure. The latent representation, which is the input of this network, has a width of 49 and a height of 65.

Decoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	32	32	3	2	1	97	129
2	32	16	3	2	1	193	257
3	16	8	3	2	1	385	513
4	8	4	3	2	1	769	1025
5	4	1	3	1	1	769	1025

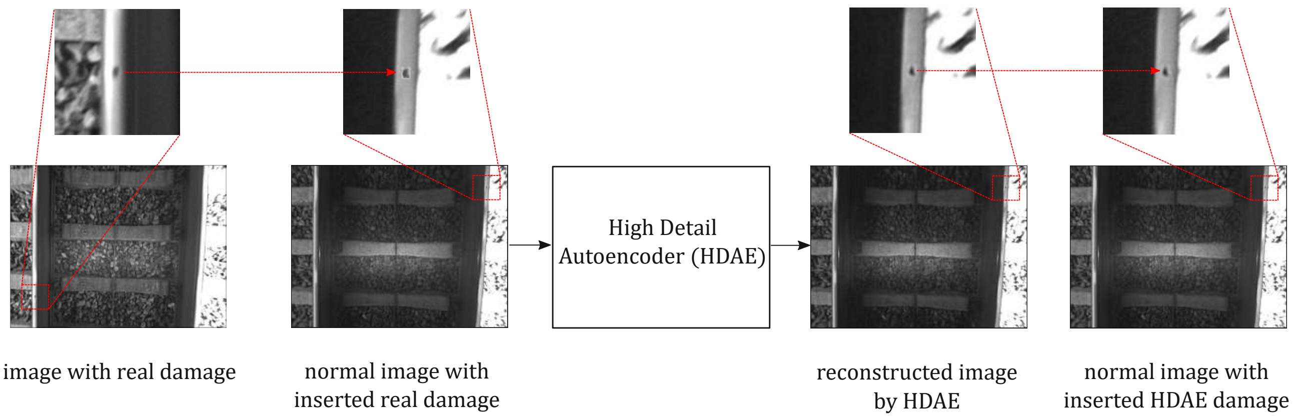


Figure 3.7: The left-most images show an image with real rail damage. A small patch containing this damage is copied and inserted on the rail of a normal image. The damage in the output image of an AE looks more realistic than in its input image. Finally, a small patch, including this reconstructed damage, is inserted at the same position in the normal image.

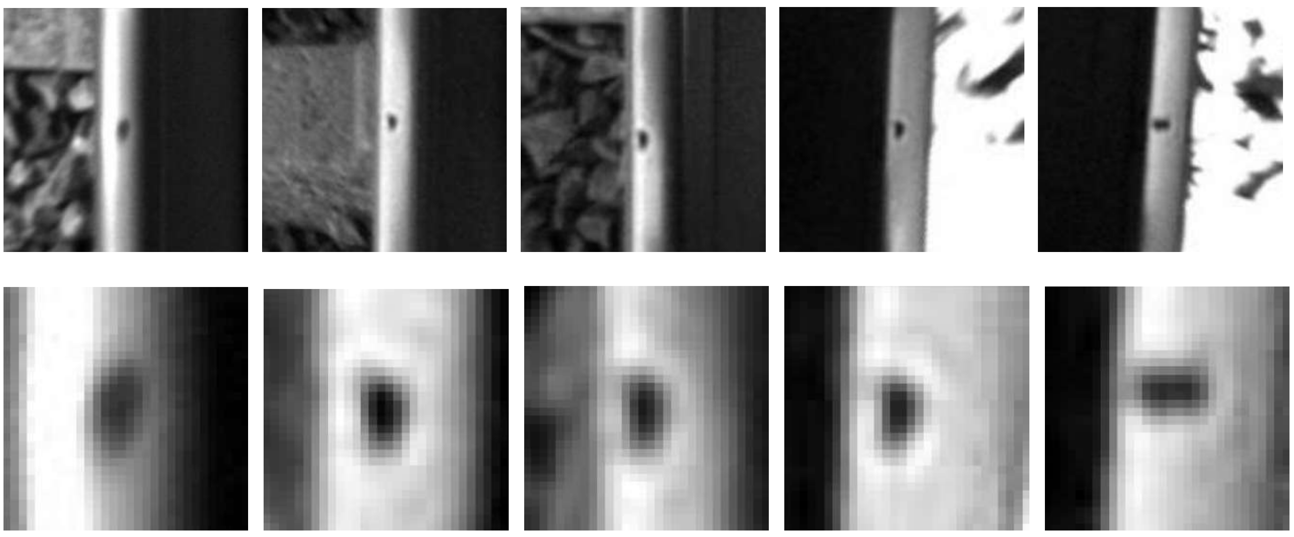


Figure 3.8: The images in the top row show indentations on different rail images; the second-row images are close-up versions of the same damages. The damaged rail displayed on the left-most column is a real image from the *Kombi* dataset, while the other four damages are replicated versions of this indentation.

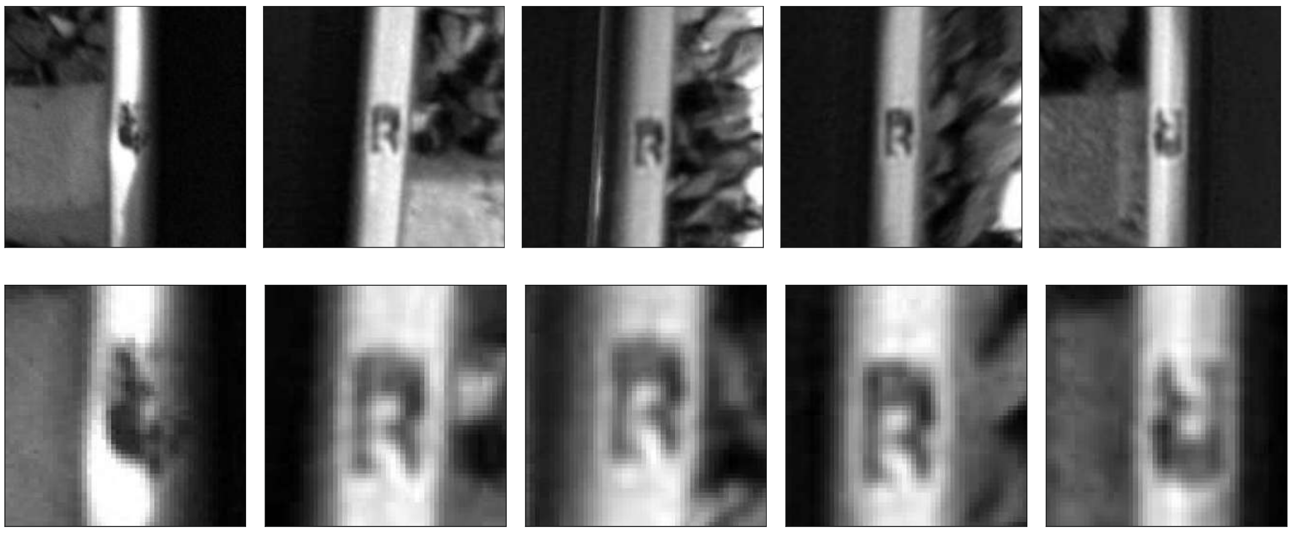


Figure 3.9: The images in the top row show breakouts on different images of rails, and the second-row images are close-up versions of the same damages. Third-row images display the according patch of the reconstruction error images. The damaged rail displayed on the left-most column is a real image from the *Kombi* dataset, while the other four damages are replicated versions of this breakout.

Chapter 4

Architecture

This chapter introduces several neural network models for anomaly detection approaches. First, anomaly detection approaches introduced in Section 2.4 are compared and analyzed regarding their applicability to this thesis's task. Then, Autoencoders (AEs), Denoising Autoencoders (DAEs), one Variational Autoencoder (VAE), and one Deep Convolutional Generative Adversarial Network (DCGAN) are analyzed further. The following sections analyze the applicability and effectiveness of approaches based on Neural Networks (NNs) for this specific application. Several approaches are analyzed and compared to find an appropriate detection method. First, the summed-up reconstruction errors of normal and anomalous images for the different AEs, DAEs, and the VAE are analyzed. Secondly, the latent representations of the AE and DAE models are analyzed to allow an interpretation of the learned features. Furthermore, the reconstruction-probability measure of the VAE and the effectiveness of the DCGAN's discriminator for anomaly detection are investigated. If not otherwise specified, every layer, except the final layer of the decoder networks, utilizes the Rectified Linear Unit (ReLU) activation function described in Equation 2.3. The sigmoid activation function defined in Equation 2.1 follows the final layers of the decoder networks.

4.1 Anomaly Detection Methods Analysis

A Machine-Learning (ML)-based approach is beneficial to take advantage of the available rather large dataset. Therefore, within this thesis, the focus lies on these approaches. Although using information about already annotated anomalies can improve the performance of an anomaly detection algorithm, Vision-based Anomaly Detection Algorithm for Railroads (VADAR) should not rely on this data because there is no guarantee that the low number of known anomalies is representative of all possible anomalies. Therefore, unsupervised and semi-supervised anomaly detection approaches are of interest. VADAR must be capable of analyzing image data captured from cameras mounted on a train. Although the real-time capability of VADAR is not the focus of this thesis, VADAR, potentially with slight adaptions, should allow for frame rates of roughly ten frames per second on an embedded Graphics Processing Unit (GPU) like a unit from the NVIDIA Jetson Series [131]. Furthermore, an anomaly detection algorithm giving some explanation for why an instance is considered to be anomalous is valuable. Table 4.1 lists ML-based methods for vision-based anomaly detection and their learning approaches. Furthermore, this table lists additional information on each approach and if a technique allows for some anomaly explanation. Since VADAR should be able to detect anomalies without entirely relying on anomalous instances for training, this table only lists approaches that allow for unsupervised or semi-supervised learn-

Table 4.1: Comparison of multiple ML-based anomaly detection approaches.

Method	Learning Approach	Anomaly Explanation	Additional Information
Probabilistic	Unsupervised/Semi-supervised	Limited	Very difficult to design effective prior for an application, performance heavily depends on this design or model [48]
AE	Unsupervised	Yes	May reduce false positives over traditional methods if learned representations are expressive [48]
DAE	Unsupervised	Yes	May reduce false positives over traditional methods if learned representations are expressive [48], denoising capabilities [45]
VAE	Unsupervised	Yes, but limited in latent space	May reduce false positives over traditional methods if learned representations are expressive [48]
AE/DAE with further analysis in latent space	Unsupervised/Semi-supervised	Limited	Efficacy of analysis dependent on features extracted by AE/DAE [48]
GAN discriminator	Unsupervised	Limited	According to [116] performance often worse than generator-based approaches, but faster
GAN generator	Unsupervised	Yes	Iterative optimization process is computationally expensive, longer inference [116]

ing approaches. Probabilistic methods rely on the assumption that all data stems from a specific data distribution. For many high-dimensional real datasets, this assumption does not hold [17]. Furthermore, according to [88], designing an effective prior for different anomaly detection applications is challenging. Although the generator of a Generative Adversarial Network (GAN) can be used for anomaly detection, the long inference time due to the involved iterative optimization process makes the approach not feasible for VADAR. The effectiveness of a given method always depends on the specific application and data. Therefore, multiple approaches were analyzed to find a suitable method for this application and dataset. Specifically, this thesis focuses on multiple AEs and DAEs, a VAE, and the discriminator of a DCGAN. Chapter 4 begins by analyzing these approaches with different underlying architectures.

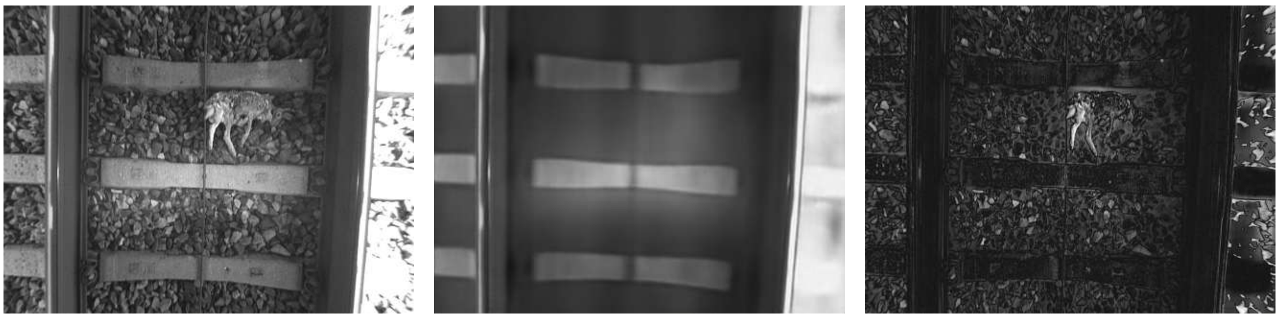

(a) Original image (b) Reconstructed image (c) Reconstruction error image

Figure 4.1: Subfigure (a) shows the input image of the AE with the architecture described in Tables 4.2 and 4.3. The reconstructed image of the AE and the resulting reconstruction error image are shown in Figure (b) and Figure (c), respectively.

4.1.1 Autoencoders

This section introduces two different AE architectures; both were inspired by models proposed in previous works of other research groups [35, 45]. However, the layer count, number of channels, and kernel sizes were modified to adapt the models to our dataset.

Ten-Layer Autoencoder

Tables 4.2 and 4.3 list the encoder's and decoder's layers and parameters of the first model. The encoder consists of ten convolutional layers, and the decoder contains ten transposed-convolutional layers. Figure 4.1 shows the reconstruction of an example image containing an anomaly and the resulting reconstruction error image. The reconstruction error image

$$I_{RE} = |I_{input} - I_{recon}| \tag{4.1}$$

represents the pixel-wise absolute difference between the input image I_{input} and the reconstructed image I_{recon}. The AE can reconstruct the rails and crossties well. Because darker pixels represent smaller reconstruction errors, the crossties and rails in the reconstruction error image are dark. On the other hand, this AE model struggles with reconstructing other image details, like the ballast or the anomaly. Parts of the anomaly and some bright stones are visible in the reconstruction error image due to the higher reconstruction errors.

Table 4.2: All layers are two-dimensional convolutional layers implemented in PyTorch. The input image has a width of 1,025 and a height of 769.

Encoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1	4	7	1	3	769	1025
2	4	8	7	2	3	385	513
3	8	16	7	2	3	193	257
4	16	32	7	2	3	97	129
5	32	64	7	2	3	49	65
6	64	64	7	2	3	25	33
7	64	64	7	2	3	13	17
8	64	64	7	2	3	7	9
9	64	64	7	1	3	7	9
10	64	64	7	1	3	7	9

Figure 4.2 visualizes the distribution of the summed-up absolute reconstruction error

$$\Sigma_{RE} = \sum |I_{input} - I_{recon}| \tag{4.2}$$

of all labeled images. Although the mean value for anomalous images is slightly higher than normal images, both distributions are very broad and overlap. Therefore, anomalous images can not be separated from normal images by simply defining a threshold value since this would lead to many false positives.

Table 4.3: All layers are two-dimensional transposed convolutional layers implemented in PyTorch. The decoder network is symmetric to the encoder network regarding layer structure. The latent representation, which is the input of this network, has a width of 7, a height of 9, and 64 channels.

Decoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	64	64	7	1	3	7	9
2	64	64	7	1	3	7	9
3	64	64	7	2	3	13	17
4	64	64	7	2	3	25	33
5	64	64	7	2	3	49	65
6	64	32	7	2	3	97	129
7	32	16	7	2	3	193	257
8	16	8	7	2	3	385	513
9	8	4	7	2	3	769	1025
10	4	1	7	1	3	769	1025

Eight-Layer Autoencoder

Tables 4.4 and 4.5 list the encoder's and decoder's layers and parameters of the second AE model. The encoder and decoder have eight convolutional and eight transposed-convolutional layers, respectively. In contrast to the model described in Tables 4.2 and 4.3, each layer is followed by a batch-normalization layer. Due to the different architectures, the reconstructions of input images differ in quality and detail. Figure 4.3 shows the reconstruction of an example image and the resulting reconstruction error image. The reconstruction error represents the pixel-wise absolute difference

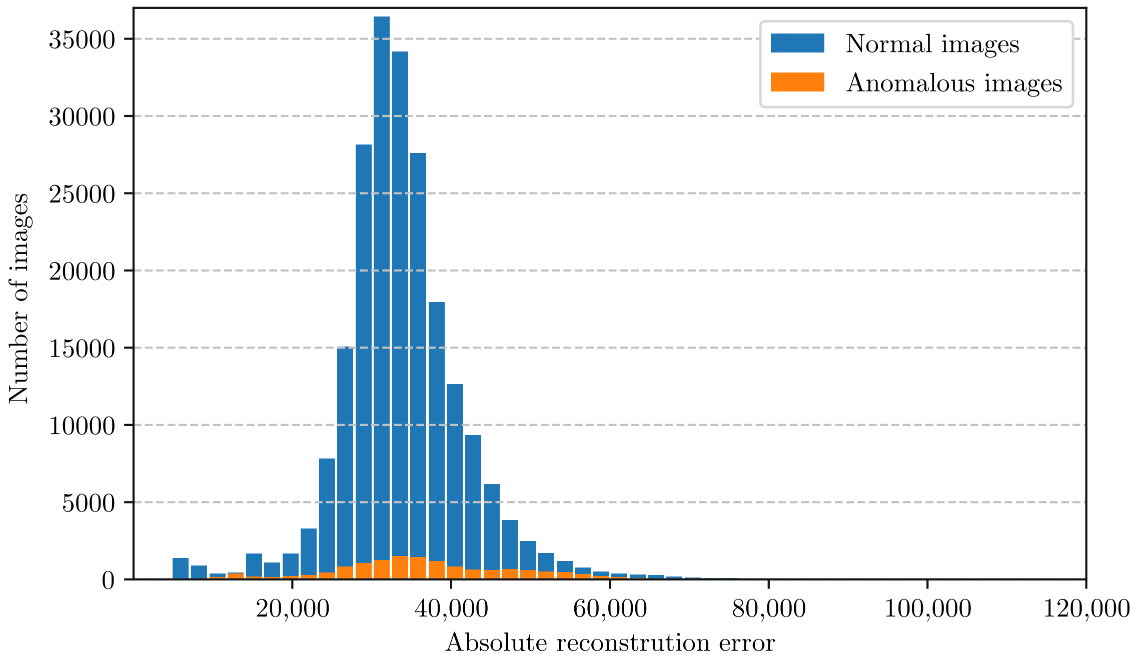


Figure 4.2: This figure shows the distribution of the summed-up absolute reconstruction error of images with and without anomalies. These reconstruction error values result from the ten-layer AE with the encoder architecture described in Table 4.2. The mean absolute reconstruction error of samples with and without anomalies is 37,175 and 33,999, respectively.

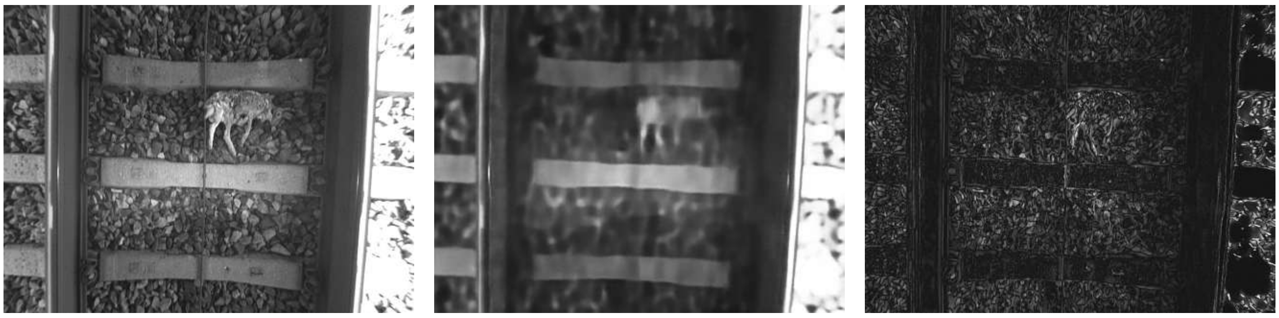

(a) Original image (b) Reconstructed image (c) Reconstruction error image

Figure 4.3: Figure (a) shows the input image of the AE with the architecture described in Tables 4.4 and 4.5. The reconstructed image of the AE and the resulting reconstruction error image are shown in Figure (b) and Figure (c), respectively.

between the input and reconstructed image. Compared with the ten-layer AE described in Section 4.1.1, this AE model can reconstruct more details of the image. Although some brighter stones are now visible in the reconstruction image, parts of the anomaly are also reconstructed in more detail. One factor influencing the reconstruction capabilities of an AE is the dimension of the latent representation. The higher the number of channels, width, and height of the latent vector, the more information about the image can be stored and used by the decoder to reconstruct the input. This aspect needs to be considered for an anomaly detection task to allow an AE to generate good reconstructions of normal images while failing to reconstruct anomalies.

The distribution of summed-up absolute reconstruction errors of images with and without anomalies achieved with this AE architecture can be seen in Figure 4.4. The reconstruction errors are significantly lower compared to the ten-layer AE analyzed in Section 4.1.1. The mean value of absolute reconstruction errors is roughly 24% smaller. This is also expected since the reconstructed images show many details of the original images, as demonstrated in Figure 4.3, that are not visible in Figure 4.1. However, the distributions of absolute reconstruction errors for images with and without anomalies still overlap. Therefore, the absolute reconstruction error is not a reliable indicator of anomalies for this AE either.

Table 4.4: All layers are two-dimensional convolutional layers implemented in PyTorch. A batch normalization layer follows each convolutional layer. The input image has a width of 1025 and a height of 769.

Encoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1	16	3	1	1	769	1025
2	16	16	3	2	1	385	513
3	16	32	3	2	1	193	257
4	32	32	3	2	1	97	129
5	32	32	3	2	1	49	65
6	32	32	3	2	1	25	33
7	32	32	3	2	1	13	17
8	32	32	3	2	1	7	9

Table 4.5: All layers are two-dimensional transposed convolutional layers implemented in PyTorch. A batch normalization layer follows each convolutional layer. The decoder network is symmetric to the encoder network regarding the layer structure. The latent representation, which is the input of this network, has a width of 13 and a height of 17.

Decoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	32	32	3	2	1	13	17
2	32	32	3	2	1	25	33
3	32	32	3	2	1	49	65
4	32	32	3	2	1	97	129
5	32	32	3	2	1	193	257
6	32	16	3	2	1	385	513
7	16	16	3	2	1	769	1025
8	16	1	3	1	1	769	1025

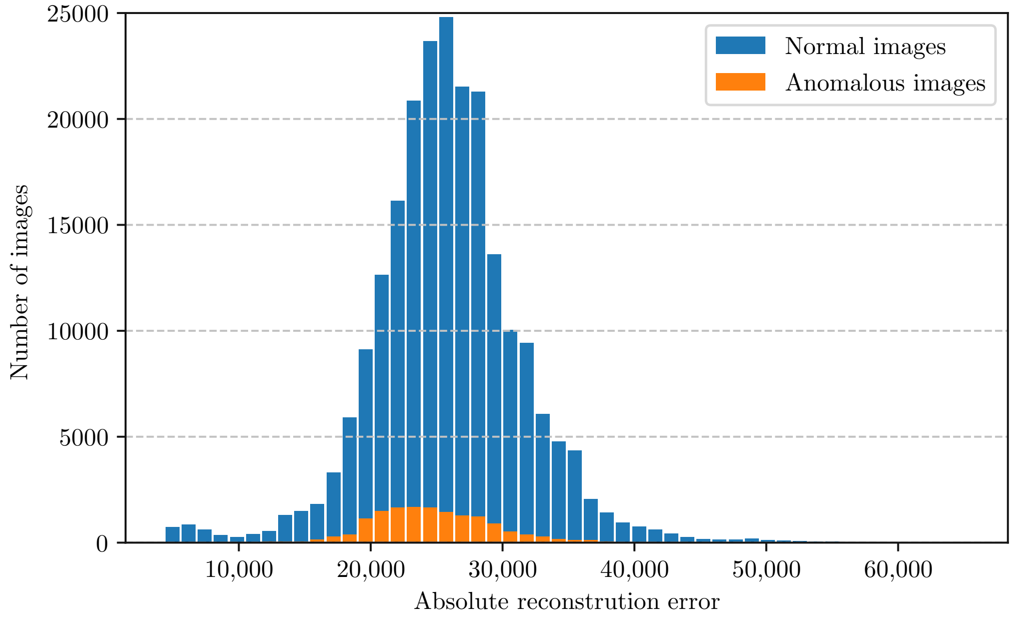
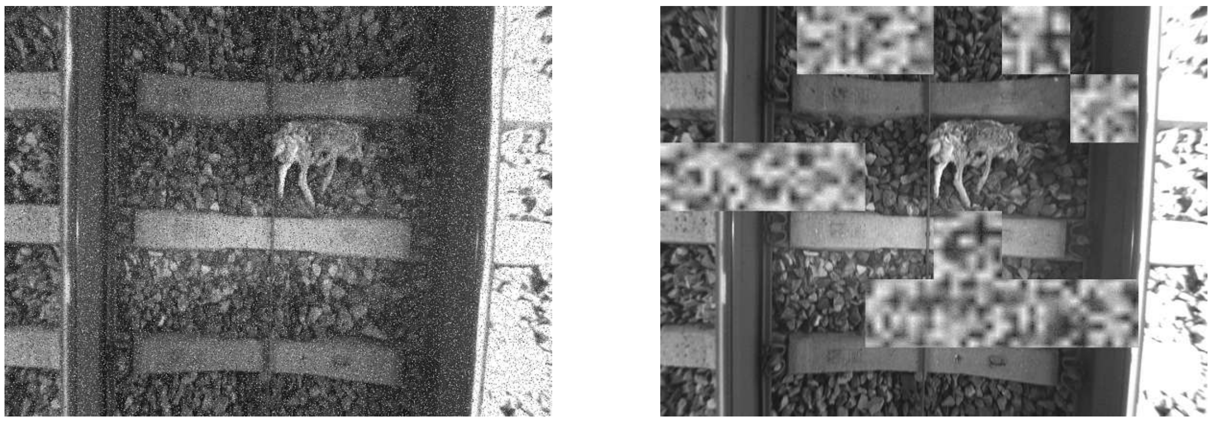


Figure 4.4: This figure shows the distribution of the summed-up absolute reconstruction error of images with and without anomalies. These reconstruction error values result from the ten-layer AE with the encoder architecture described in Table 4.4. The mean absolute reconstruction error of samples with and without anomalies is 25,368 and 25,793, respectively.

4.1.2 Denoising Autoencoders

In principle, DAEs differ from conventional AEs regarding their training procedure. As described in Section 2.4.2, AEs receive the original images as input during training, and DAEs only get to see augmented versions of the images. However, DAEs are trained to reconstruct the unaltered images because the loss function is computed between the reconstructed and original images. Typically, this augmentation of the input data is accomplished by adding noise. Popular choices of noise sources for DAEs are Gaussian and salt and pepper noise or up-scaled versions of these [45, 103]. Another way of adding noise is to mask patches of the input. This can be accomplished by replacing these patches with a constant value or with random values [74]. Figure 4.5 demonstrates what salt and pepper noise and masking an image looks like. These noise sources influence the feature extraction capabilities of a DAE in different ways. Salt and pepper noise makes a model more robust regarding noise and more resilient against small-scale deviations. This small-

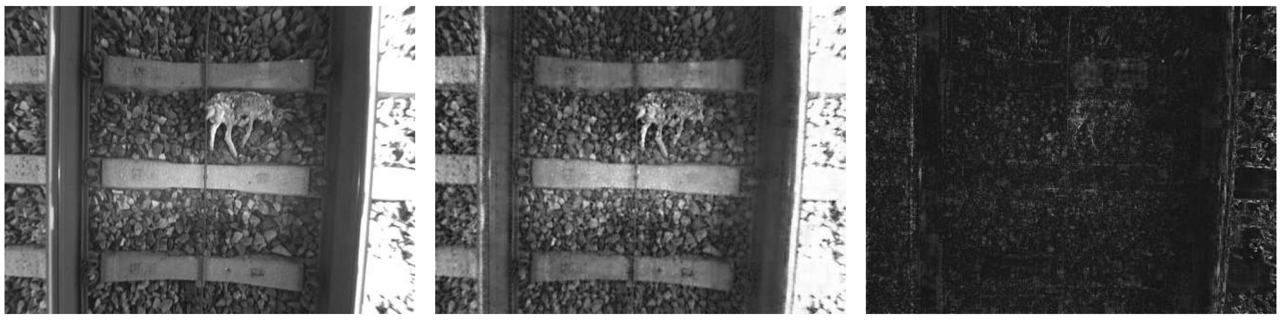


(a) Image with salt and pepper noise

(b) Image masked with up-scaled noise

Figure 4.5: Figure (a) and Figure (b) show an image with added salt and pepper noise and masked with up-scaled noise, respectively.

scale noise forces the model to extract meaningful features from the noisy images and ignore small-scale deviations. However, masking patches of an image forces a DAE to ignore unexpected new features by using contextual information. This allows the model to ignore larger regions of an image during training that include information unexpected from surrounding regions and replace it with data more likely for the given context. The next two subsections discuss and analyze two different DAE architectures with different noise sources inspired by other papers [45, 103, 74]. The layer count, number of channels, and kernel sizes were changed to adapt the models to our dataset.



(a) Original image

(b) Reconstructed image

(c) Reconstruction error image

Figure 4.6: Figure (a) shows the input image of the five-layer DAE with the architecture described in Tables 4.6 and 4.7. The reconstructed image of the DAE and the resulting reconstruction error image are shown in Figure (b) and Figure (c), respectively.

Five Layer Denoising Autoencoder

Tables 4.6 and 4.7 show the encoder and decoder architectures of the first DAE, respectively. During the training, 10% of salt and pepper noise was added to all input images. Figure 4.5a illustrates an image with added salt and pepper noise. In contrast to the before mentioned architectures, the dimensionality reduction of the latent representation is accomplished by maximum pooling layers while the stride of each convolutional layer is set to 1. Figure 4.3 shows the reconstruction of an example image and the resulting reconstruction error image. The reconstruction error represents the pixel-wise absolute difference between the input and reconstructed image. Clearly, this model performs well in the reconstruction of the image since many details are recognizable in the reconstructed image shown in Figure 4.6b. Most of the remaining reconstruction errors stem from edges of separate stones on the trackbed, ballast in very bright image regions, and parts of the anomaly.

Table 4.6: All layers are two-dimensional convolutional or maximum pooling layers implemented in PyTorch. A batch normalization layer follows each convolutional layer. The input image has a width of 1025 and a height of 769.

Encoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1	16	5	1	2	769	1025
MaxPool 1	16	16	5	4	2	193	257
2	16	16	5	1	1	193	257
MaxPool 2	16	32	5	4	2	49	65
3	32	32	5	1	1	49	65
MaxPool 3	32	64	5	4	2	13	17
4	64	64	5	1	1	13	17
MaxPool 4	64	64	5	2	2	7	9
5	64	64	5	1	1	7	9

Table 4.7: All layers are two-dimensional transposed convolutional or maximum unpooling layers implemented in PyTorch. Each convolutional layer is followed by a batch normalization layer. The decoder network is symmetric to the encoder network regarding layer structure. The latent representation, which is the input of this network, has a width of 9 and a height of 17.

Decoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	64	64	5	1	2	7	9
MaxUnPool 1	64	64	5	2	2	13	17
2	64	64	5	1	2	13	17
MaxUnPool 2	64	64	5	4	2	49	65
3	64	32	5	1	2	49	65
MaxUnPool 3	32	32	5	4	2	193	257
4	32	16	5	1	2	193	257
MaxUnPool 4	16	16	5	4	2	769	1025
5	16	1	5	1	2	769	1025

Ten-Layer Denoising Autoencoder

For the second DAE, the encoder and decoder architectures of Tables 4.2 and 4.3 were used. However, each image was segmented into 48 square patches, and 12 were replaced by up-scaled random values. One example of a masked image is shown in Figure 4.5b. Figure 4.9 shows the reconstruction of an example image and the resulting reconstruction error image. The reconstruction error represents the pixel-wise absolute difference between the input and reconstructed image.

The distributions of absolute reconstruction errors for labeled images with and without anomalies obtained from this ten-layer DAE are shown in Figure 4.8. As for the other AEs and DAE, the summed-up absolute reconstruction error does not seem to be a good indicator for anomalies since the distributions of images with and without anomalies overlap.

4.1.3 Latent Space Analysis

The basic principle of AEs and DAEs is to generate a latent space representation of input data and reconstruct the input from this vector. Analyzing this transformed input may reveal more insight into the features and aspects the models are focused on. If the extracted features for anomalous images differ significantly from the features of normal images, an

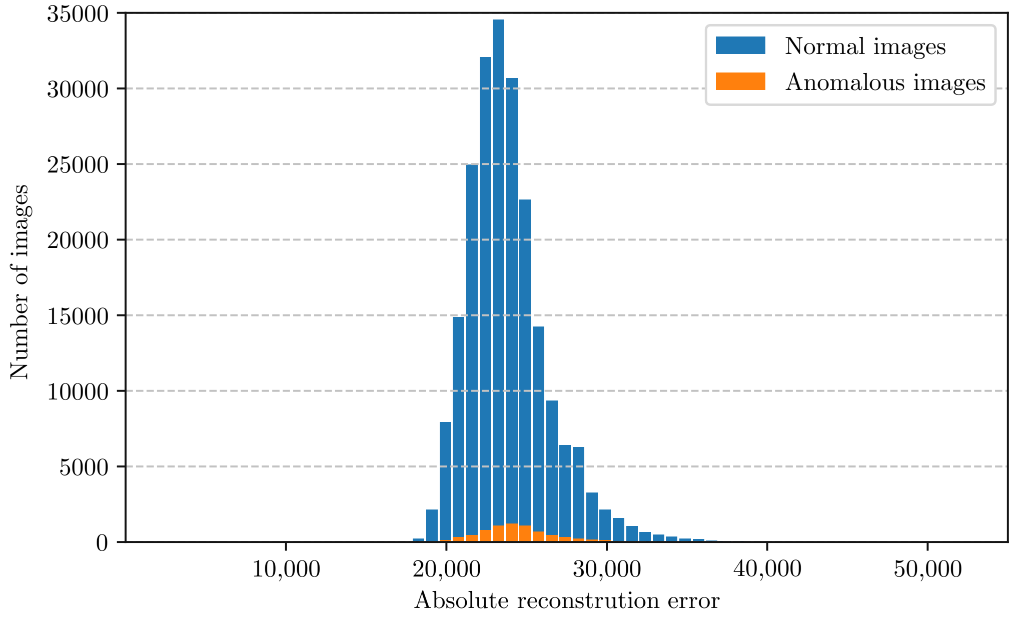


Figure 4.7: This figure shows the distribution of the summed-up absolute reconstruction error of images with and without anomalies. These reconstruction error values result from the five-layer DAE with the encoder architecture described in Table 4.6. The mean absolute reconstruction error of samples with and without anomalies is 24,781 and 23,851, respectively.

anomaly detection method applied to this latent representation could also be a promising approach. For this analysis, t-distributed Stochastic Neighbor Embedding (t-SNE) was used to visualize high-dimensional data by projecting it to a two-dimensional space [132].

Figures 4.10 to 4.14 show the two-dimensional output of the t-SNE-algorithm applied to the latent representations of hundreds of example images. Figures 4.10 and 4.11 display this projection for roughly 150 instances of anomalous

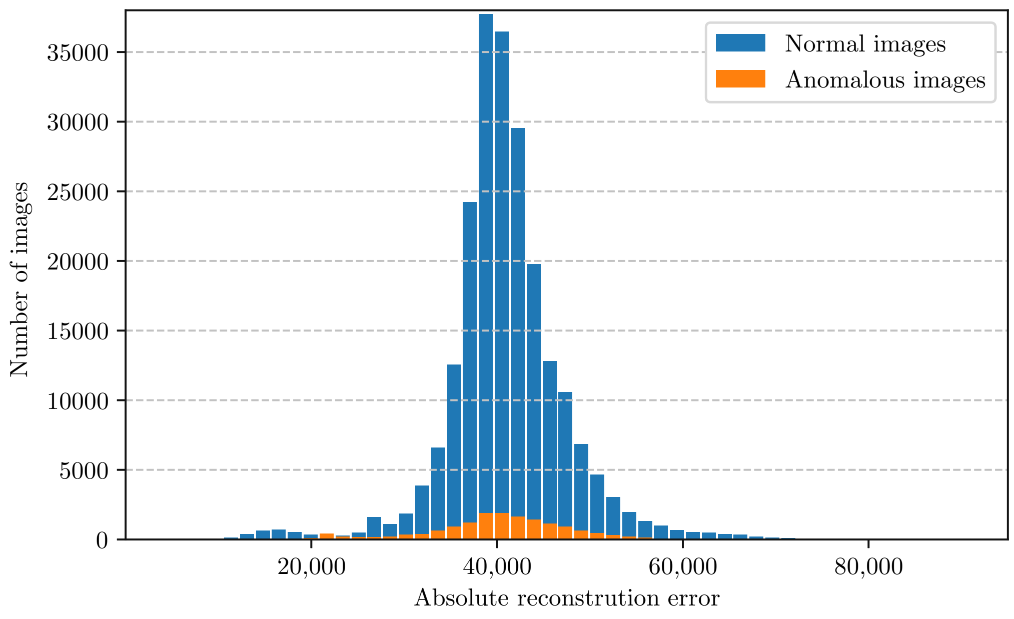
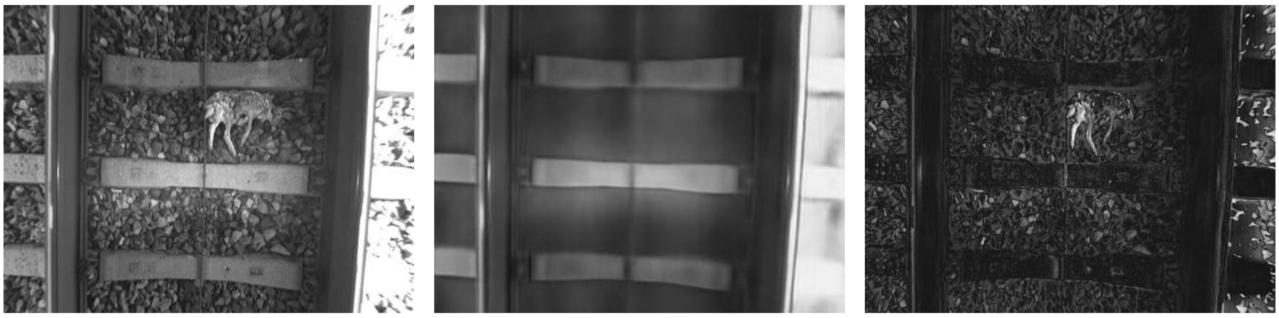
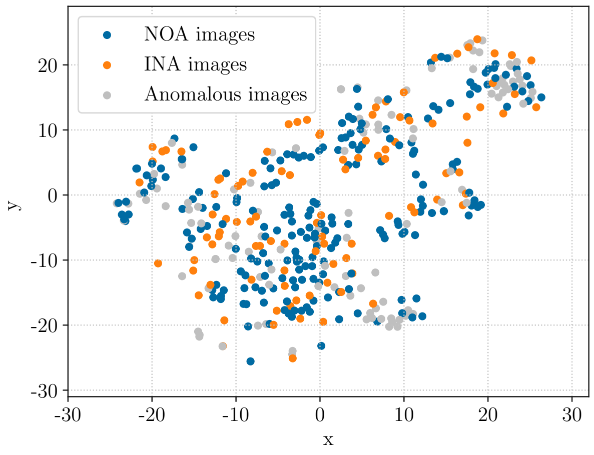


Figure 4.8: This figure shows the distribution of the summed-up absolute reconstruction error of normal images and anomalous images. These reconstruction error values result from the ten-layer DAE with the encoder architecture described in Table 4.2. The mean absolute reconstruction error of samples with and without anomalies is 40,936 and 40,971, respectively.

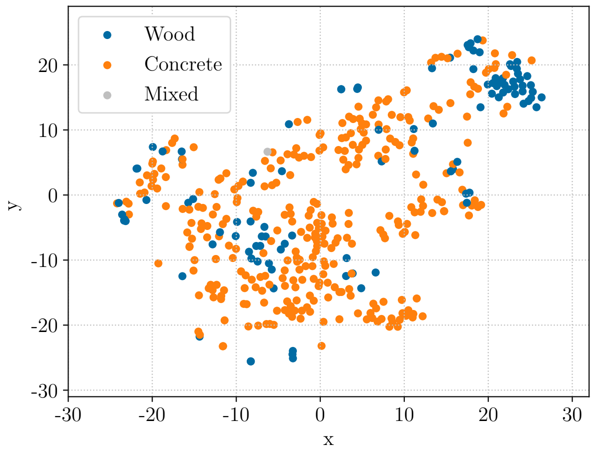


(a) Original image (b) Reconstructed image (c) Reconstruction error image

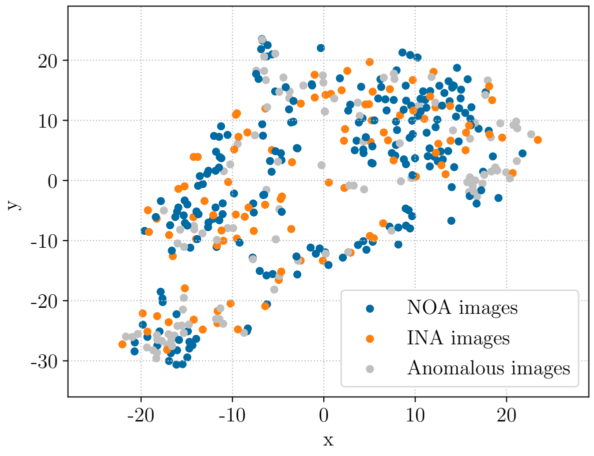
Figure 4.9: Figure (a) shows the input image of the ten-layer DAE with the architecture described in Tables 4.2 and 4.3. The reconstructed image of the DAE and the resulting reconstruction error image are shown in Figure (b) and Figure (c), respectively.



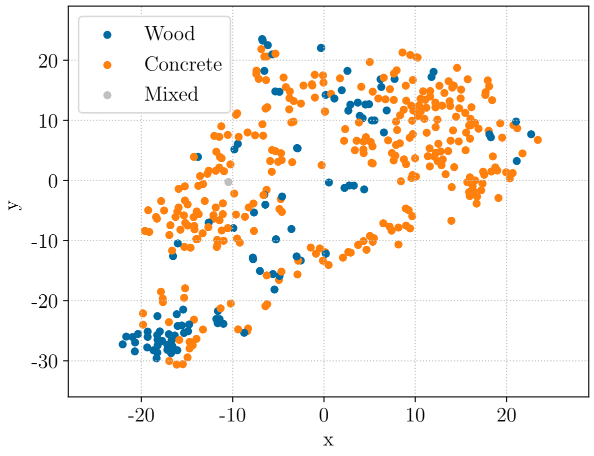
(a) This figure shows a projection of the latent representation obtained from the ten-layer encoder architecture described in Table 4.2.



(b) This figure shows a projection of the latent representation obtained from the ten-layer encoder architecture described in Table 4.2.

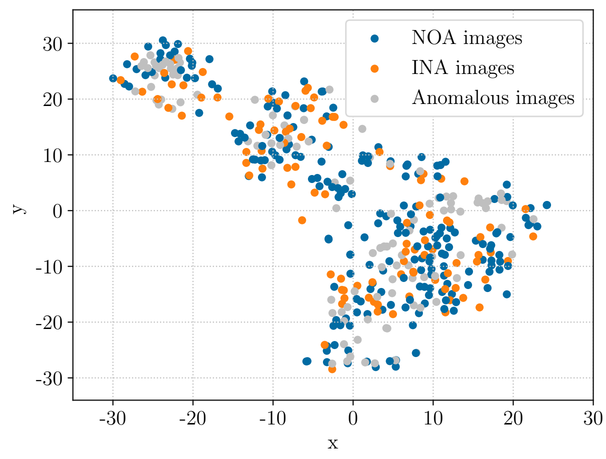


(c) This figure shows a projection of the latent representation obtained from the ten-layer encoder of the masked DAE listed in Table 4.2. Since this DAE utilizes the same architecture as in Figure (a), the plot shows similar results.

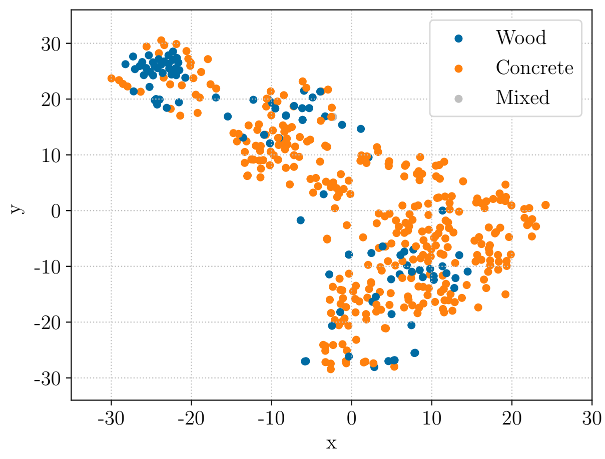


(d) This figure shows a projection of the latent representation obtained from the ten-layer encoder of the masked DAE. Since this AE utilizes the same architecture as in Figure (b), the plot shows similar results.

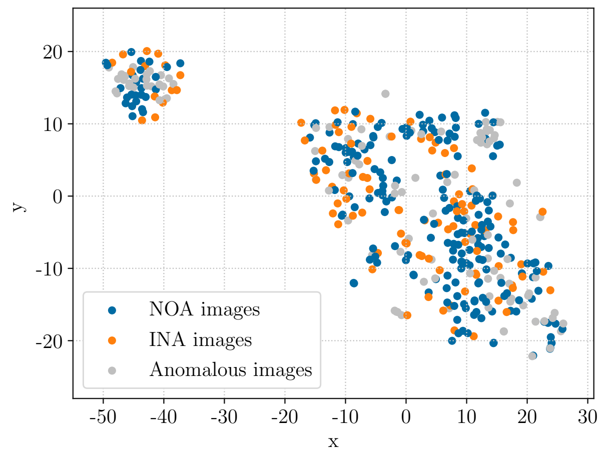
Figure 4.10: These figures show the output of the T-SNE algorithm applied to the latent space representations of the ten-layer AE and DAE with the encoder architecture shown in Table 4.2. Roughly 150 images with anomalies, infrastructure elements, and normal images without any annotations are plotted. The colors group images according to their type (normal, infrastructure, and anomaly) or the crosstie material in the image.



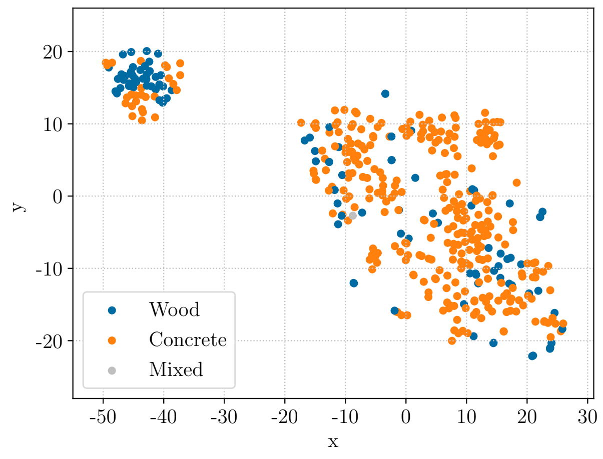
(a) This figure shows a projection of the latent representation obtained from the eight-layer encoder architecture described in Table 4.4.



(b) This figure shows a projection of the latent representation obtained from the eight-layer encoder architecture described in Table 4.4.



(c) This figure shows a projection of the latent representation obtained from the five-layer encoder of the DAE with the architecture described in Table 4.6.

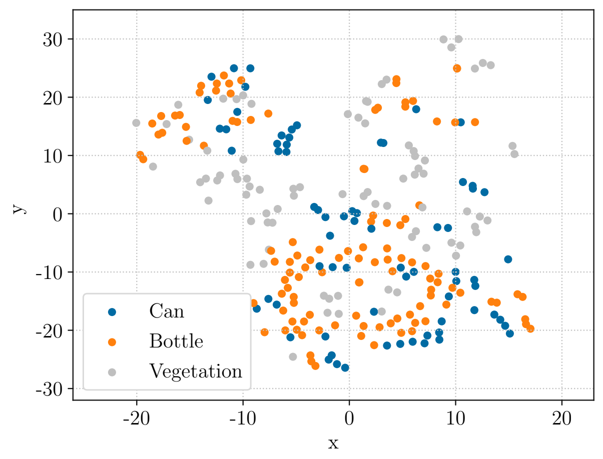


(d) This figure shows a projection of the latent representation obtained from the five-layer encoder of the DAE with the architecture described in Table 4.6.

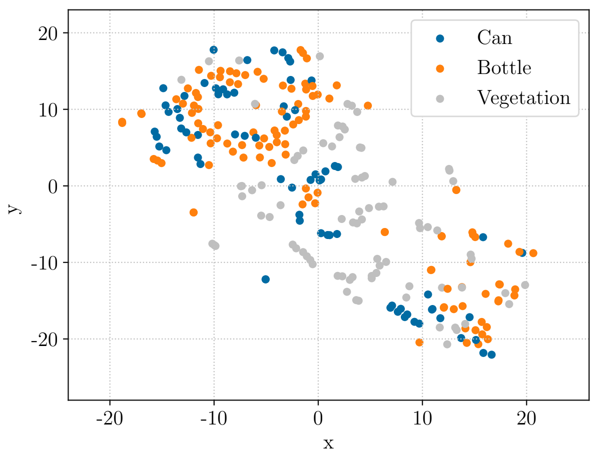
Figure 4.11: These figures show the output of the T-SNE algorithm applied to the latent space representations of the AE with the encoder architecture shown in Table 4.4 and Table 4.6. Roughly 150 images with anomalies, infrastructure elements, and normal images without any annotations are plotted. The colors group images according to their type (normal, infrastructure, and anomaly) or the crosstie material in the image.

images, image containing only infrastructure annotationss (INA images), and image containing no annotationss (NOA images) for the proposed AE and DAE models. While Figure 4.10 visualizes the projections of the ten-layer AE's and ten-layer DAE's latent space representations, Figure 4.11 shows the according eight-layer AE's and five-layer DAE's latent space projections. In Figures 4.10a, 4.10c, 4.11a, and 4.11c, points are colored depending on wether they belong to NOA images, INA images, or anomalous images. The colors in Figures 4.10b, 4.10d, 4.11b, and 4.11d represent the type of crosstie material. There is no clear clustering of normal, infrastructure, or anomalous images for any of the analyzed models. However, many images with wooden crossties are clustered together in all models.

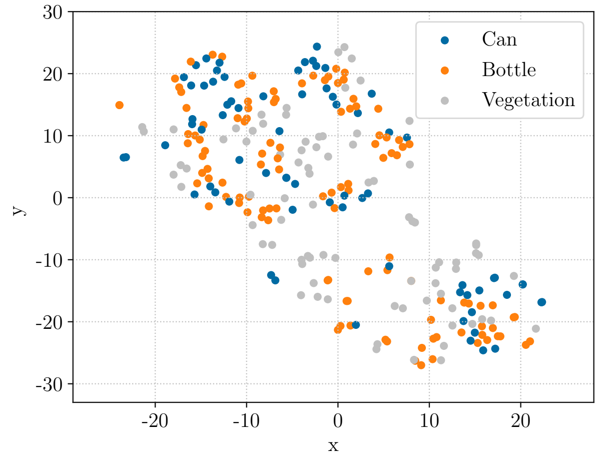
Figure 4.12 shows the t-SNE-projection for images of three anomaly classes with 50 instances each for all proposed AE and DAE models. Although a few instances of the same anomaly class have values close to each other, those instances come from the same dataset sequences. Therefore, these images share similar crossties and lighting conditions.



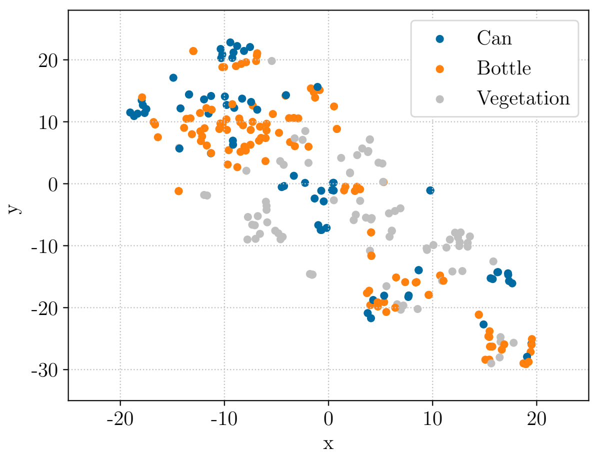
(a) This figure shows a projection of the latent representation obtained from the ten-layer encoder of the ten-layer AE described in Table 4.2.



(b) This figure shows a projection of the latent representation obtained from the ten-layer encoder of the masked DAE described in Table 4.2.



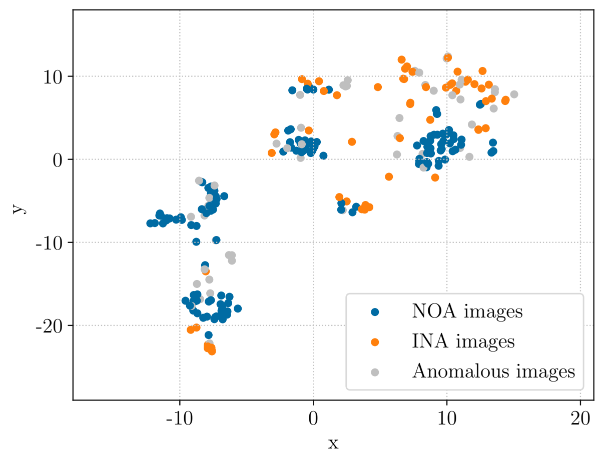
(c) This figure shows a projection of the latent representation obtained from the encoder of the eight-layer AE described in Table 4.4.



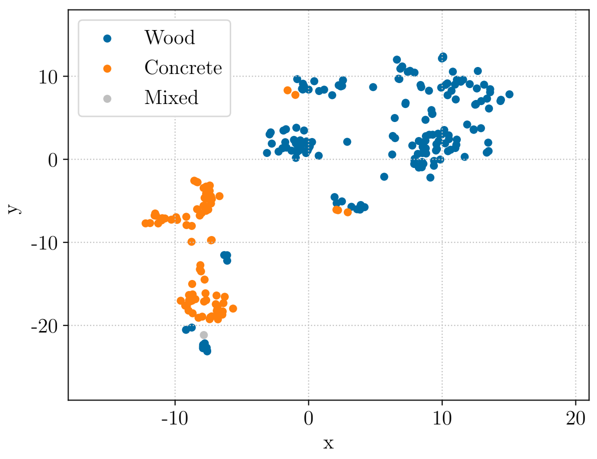
(d) This figure shows a projection of the latent representation obtained from the five-layer encoder of the DAE described in Table 4.6.

Figure 4.12: These figures show the output of the T-SNE algorithm applied to the latent space representations of the two AEs and DAEs. Roughly 150 images with cans, bottles, and vegetation are plotted. The colors group images according to their anomaly class in the image.

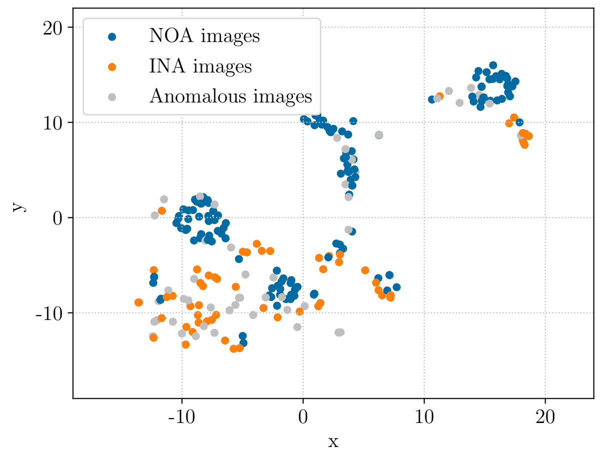
To analyze how this clustering changes for images within the same sequence, Figures 4.13 and 4.14 shows the projections of all latent representations of images from one sequence of the dataset. While Figure 4.13 shows the projections of the ten-layer AE and ten-layer DAE latent space representation, Figure 4.14 shows the same for the eight-layer AE and the five-layer DAE. The lighting conditions for every image in this sequence are similar and should not greatly influence the clustering. Furthermore, this sequence includes a relatively high number of infrastructure elements and anomalies. In Figures 4.13b, 4.13d, 4.14b, and 4.14d, points are colored according to the crosstie material. In Figures 4.13a, 4.13c, 4.14a, and 4.14c, points are colored to their corresponding image type (NOA images, INA images, or anomalous images). Apparently, the clustering is mainly influenced by the crosstie material since points representing wooden and concrete crossties are separated from each other in Figures 4.13b, 4.13d, 4.14b, and 4.14d. Anomalous instances seem to be spread across all other points. There is no clear separation of points corresponding to anomalous images from other points in Figures 4.13a, 4.13c, 4.14a, and 4.14c.



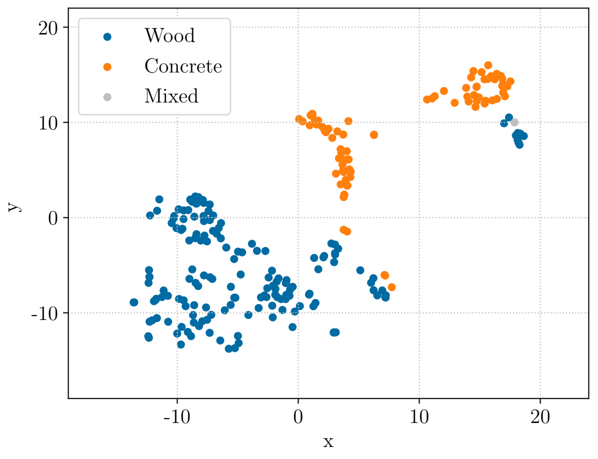
(a) This figure shows a projection of the latent representation obtained from the ten-layer encoder architecture described in Table 4.2.



(b) This figure shows a projection of the latent representation obtained from the ten-layer encoder architecture described in Table 4.2.



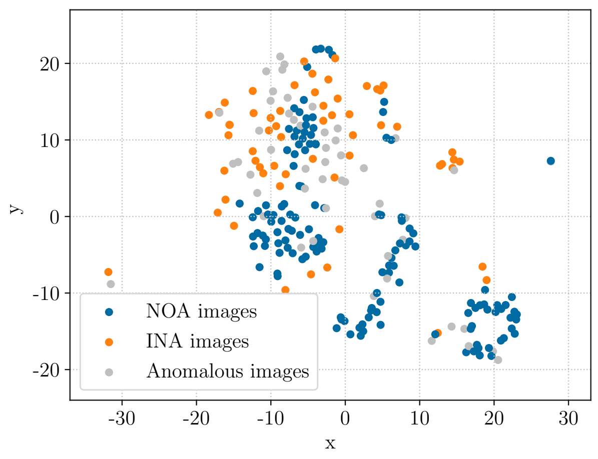
(c) This figure shows a projection of the latent representation obtained from the ten-layer encoder of the masked DAE. Since this DAE utilizes the same architecture as in Figure (a), the plot shows similar results.



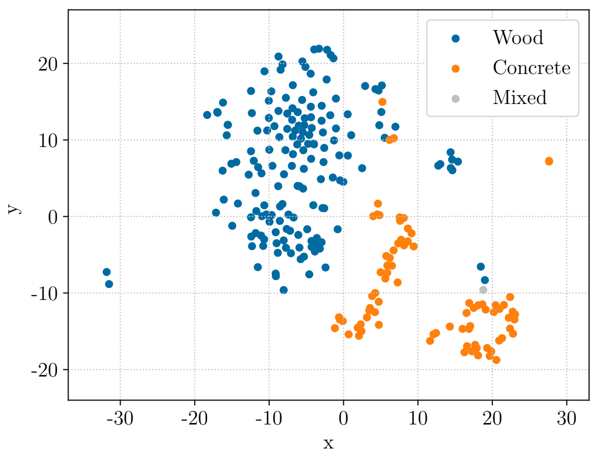
(d) This figure shows a projection of the latent representation obtained from the ten-layer encoder of the masked DAE. Since this DAE utilizes the same architecture as in Figure (b), the plot shows similar results.

Figure 4.13: These figures show the output of the T-SNE algorithm applied to the latent space representations of the ten-layer AE with the encoder architecture shown in Table 4.2. Only images from one sequence of the dataset were analyzed.

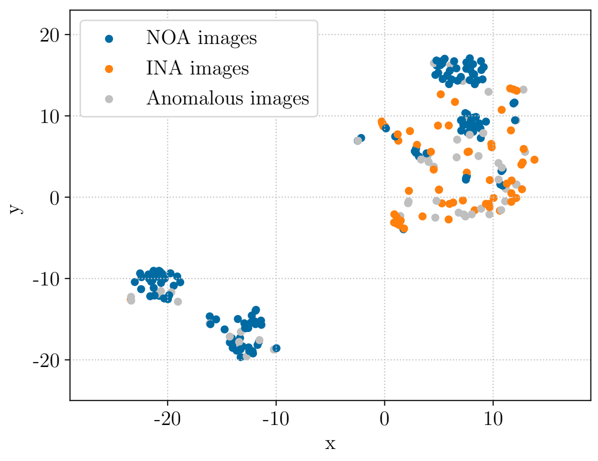
Interestingly, for all AE and DAE models, there are two sub-clusters of images with crossties out of concrete in Figures 4.13b, 4.13d, 4.14b, and 4.14d. Most likely, this is the case because all of the images belonging to one of the clusters are in a turn of the track, which leads to a horizontal shift of the rails and trackbed on the images. The points corresponding with instances with wooden crossties are further apart than the points of instances within the crosstie cluster. One reason for that could be that most infrastructure elements like switches or switch frogs have wooden crossties in this dataset sequence. Furthermore, there are two main clusters of normal images with wooden crossties since there appear two different kinds of wooden crossties with different widths. Much anomaly-specific information seems already lost after the encoder network of each AE and DAE. However, some anomalous instances with large anomalies, like heavy vegetation, have a relatively large distance to clusters of normal data. This might allow for an additional filter step to reduce the number of false positives.



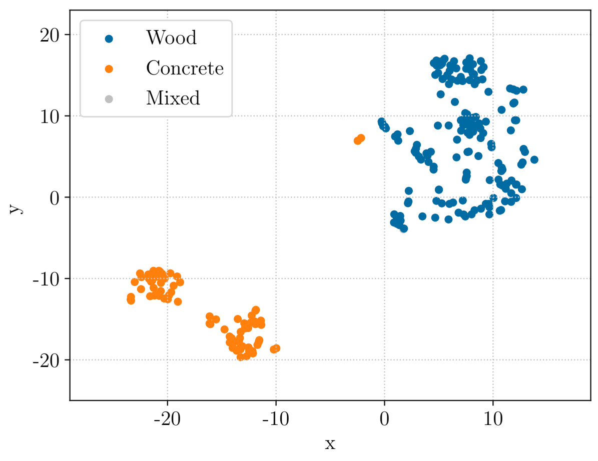
(a) This figure shows a projection of the latent representation obtained from the eight-layer encoder architecture described in Table 4.4.



(b) This figure shows a projection of the latent representation obtained from the eight-layer encoder architecture described in Table 4.4.



(c) This figure shows a projection of the latent representation obtained from the five-layer encoder of the DAE listed in 4.6.

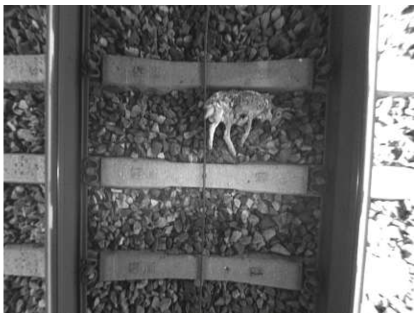


(d) This figure shows a projection of the latent representation obtained from the five-layer encoder of the DAE listed in 4.6.

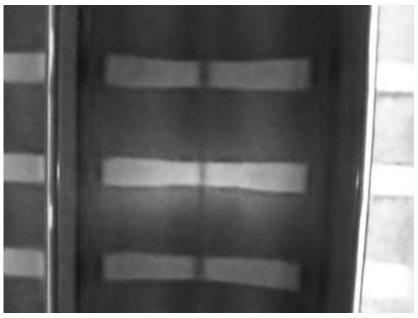
Figure 4.14: These figures show the output of the T-SNE algorithm applied to the latent space representations of the eight-layer AE with the encoder architecture shown in Table 4.4 and the five-layer DAE shown in Table 4.6. Only images from one sequence of the dataset were analyzed. The influence of the lighting conditions on the output of the t-SNE is minimal because the lighting conditions barely change throughout this sequence.

4.1.4 Variational Autoencoder

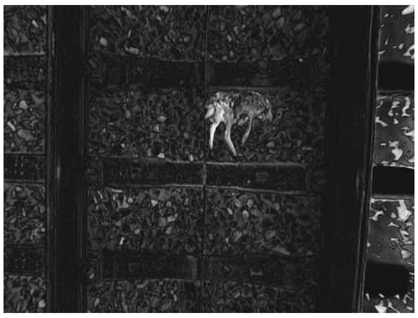
While AEs and DAEs only distinguish themselves by the training procedure and can have identical architectures, the architecture of a VAE is more complex. Instead of just using an encoder and decoder network, which are used to extract features from the input data and reconstruct the data from these features, a VAE introduces additional fully connected layers. These layers are used to give a probabilistic description of observations in the latent space. A random sample from the observed probability distribution is used and fed to the decoder network for the reconstruction of the data. Section 2.4.3 explains this process in more detail, and Figure 2.5 visualizes the architecture. The convolutional encoder and decoder architectures used to extract features from the original image and reconstruct it from a sampled latent vector are shown in Table 4.8 and Table 4.9, respectively. The output tensor of the encoder network is flattened to be used as an input for the following fully connected linear layer network. The first linear layer has 4,032 input and 2,048 output neurons and utilizes the hyperbolic tangent activation function. This output is fed to two separate linear layers with 1,024 output elements. The outputs of these two layers are interpreted as the mean values and logarithmic standard deviation values of a distribution from which a vector will be sampled and forwarded to the decoder network.



(a) Original image

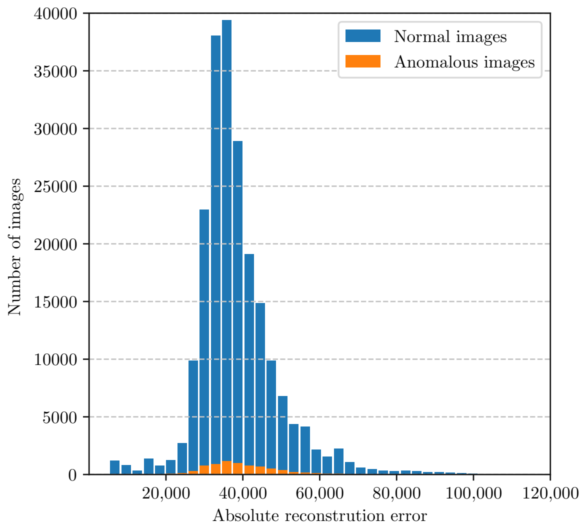


(b) Reconstructed image

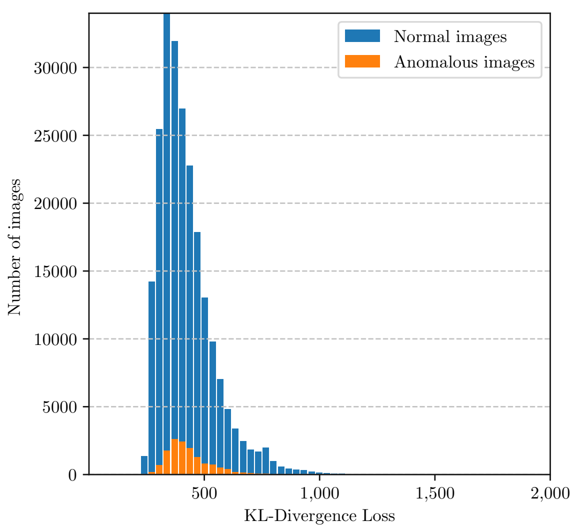


(c) Reconstruction error image

Figure 4.15: Figure (a) shows the input image of the VAE with the convolutional encoder and decoder architectures described in Tables 4.8 and 4.9. The reconstructed image of the VAE and the resulting reconstruction error image are shown in Figure (b) and Figure (c), respectively.



(a) Histogram of the absolute reconstruction error for images with and without anomalies.



(b) Histogram of the KL-divergence loss for images with and without anomalies.

Figure 4.16: Figures 4.16a and 4.16b show the distribution of the reconstruction error and KL-divergence loss. The mean value of the reconstruction error for images with and without anomalies is 34,781 and 33,852, respectively. For images with and without anomalies, the mean value of the KL-divergence loss is 438 and 420, respectively.

These two layers used for extracting the mean and logarithmic standard deviation values are visualized in Figure 2.5 by the μ and σ blocks. This sampled vector is then fed to a two-layer linear network, where the first layer has 1,024 inputs and 4,032 outputs, and the second one has 4,032 inputs and outputs. After resizing the output tensor, this network is followed up by the decoder network described in Table 4.9. Although a VAE allows us the calculation of a loss value from the latent representation and uses this as an indicator for anomalies, it also generates a reconstruction of the input image. An example image's reconstruction and reconstruction error is shown in Figure 4.16. At this point, it should be mentioned that this is just one possible reconstruction. In contrast to AEs and DAE, the reconstruction image of a VAE is non-deterministic due to the introduced random sampling in the bottleneck.

Table 4.8: The convolutional encoder of the VAE. All layers are two-dimensional convolutional layers implemented in PyTorch. A batch normalization layer follows each convolutional layer. The input image has a width of 1025 and a height of 769.

Encoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1	4	7	1	3	769	1025
2	4	8	7	2	3	385	513
3	8	16	7	2	3	193	257
4	16	32	7	2	3	97	129
5	32	64	7	2	3	49	65
6	64	64	7	2	3	25	33
7	64	64	7	2	3	13	17
8	64	64	7	2	3	7	9

Table 4.9: The Convolutional decoder of the VAE. All layers are two-dimensional transposed convolutional layers implemented in PyTorch. A batch normalization layer follows each convolutional layer. The decoder network is symmetric to the encoder network regarding layer structure. The latent representation, which is the input of this network, has a width of 7 and a height of 9.

Decoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	64	64	7	2	3	13	17
2	64	64	7	2	3	25	33
3	64	64	7	2	3	49	65
4	64	32	7	2	3	97	129
5	32	16	7	2	3	193	257
6	16	8	7	2	3	385	513
7	8	4	7	2	3	769	1025
8	4	1	7	1	3	769	1025

4.1.5 Deep Convolutional Generative Adversarial Network

While AEs, DAEs, and VAEs are trained to reconstruct given input images, the training of GANs is based on a different idea. A discriminator is trained to distinguish between real and synthetic data, and a generator network learns how to generate synthetic data to fool the discriminator. For the analysis of the applicability of GANs to the given problem description, the architectures for the generator and discriminator network were defined following the guidelines for stable DCGANs proposed by Radford *et al.* in [112]:

- Pooling layers in the discriminator and generator should be replaced by strided convolutional and fractional-strided convolutional layers, respectively.
- In both networks, batch normalization layers should be used.
- Fully connected hidden layers should be replaced for deeper architectures.
- ReLU activation function should be used for all layers of the generator network but the output, which should use the hyperbolic tangent function.

- Leaky ReLU activation should be used for all layers of the discriminator.

According to [112], following these guidelines helps to design stable DCGANs, especially for higher-resolution modeling. It should be noted that Radford *et al.* recommend using the hyperbolic tangent activation function for the output layer of the generator network because bounded activation functions lead to quicker training of the network. Instead of the hyperbolic tangent activation function, the sigmoid function was chosen, which allows mapping the output values in the range between 0 and 1 instead of -1 and 1. Following these guidelines, taking inspiration for the generator and discriminator network from the already proposed encoder and decoder networks of the various AEs, and varying hyperparameters of the network, the generator and discriminator architectures listed in Tables 4.10 and 4.11 lead to reasonable synthetic images. For this GAN training, roughly 10,000 labeled normal images of the dataset without any anomalies were used.

Table 4.10: All layers are two-dimensional transposed convolutional layers implemented in PyTorch. Each transposed convolutional layer, except the output layer, is followed by a batch normalization layer and a ReLU activation function. The output layer utilizes a sigmoid activation function. The input noise vector has 1,000 elements.

Generator Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1000	384	(7,9)	1	-	7	9
2	384	384	3	2	1	13	17
3	384	384	3	2	1	25	33
4	384	192	3	2	1	49	65
5	192	96	3	2	1	97	129
6	96	48	3	2	1	193	257
7	48	48	3	2	1	385	513
8	48	1	3	2	1	769	1025

Table 4.11: All layers are two-dimensional convolutional layers implemented in PyTorch. Each convolutional layer, except the last one, is followed by a batch normalization layer and uses a leaky ReLU activation function. The last layer uses a sigmoid activation function. Input images for this network have a height of 769 and a width of 1,025.

Discriminator Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
1	1	48	3	2	1	385	513
2	48	48	3	2	1	193	257
3	48	96	3	2	1	97	129
4	96	192	3	2	1	49	65
5	192	384	3	2	1	25	33
6	384	384	3	2	1	13	17
7	384	384	3	2	1	7	9
8	384	1	(7,9)	1	-	1	1

For the 100 epochs long training, a batch size of 16 and the *Adam*-optimizer [81] with a learning rate of 10^{-4} were chosen. Figure 4.17 shows four generated images of this generator network. Details like repeating gravel patterns or the edges of some crossties still allow the discriminator to distinguish some of the synthetic images from normal data. However, the four example images also reveal some interesting aspects and the potential of synthetic image generation.

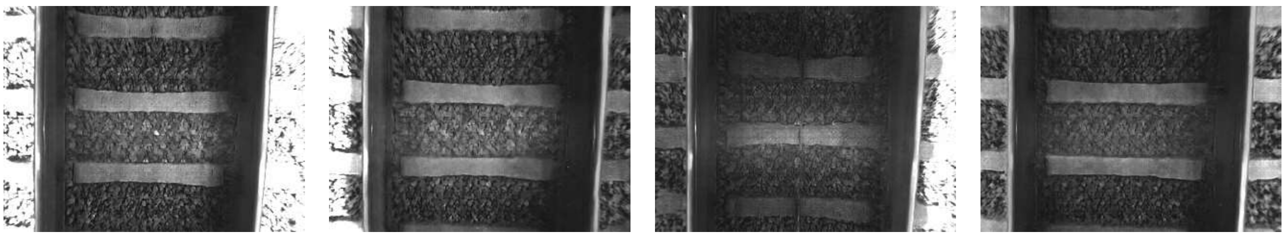


Figure 4.17: The DCGAN's generator created these four images from random noise input vectors. Although these four generated images might look realistic initially, some details reveal their synthetic nature. In some parts of the trackbed, the same pattern of gravel seems to repeat itself. Furthermore, some edges of crossties are not straight lines like in normal images.

The overwhelming majority of labeled images in the *Kombi* dataset show tracks where the left and right sides of the trackbed are roughly as bright as the middle, or only the right side is very bright due to direct sunlight. The leftmost image shows a scenario where both the right and left sides seem to be under direct sunlight, which does not occur except the sun is exactly above the train. Moreover, the second image shows a scenario where only the left side is bright, which resembles conditions underrepresented in the given dataset. As described in Section 2.4.4, the generator and discriminator can be exploited for anomaly detection. Approaches based on the generator network try to create a given image by adapting its input vector through an optimization process. This process is computationally expensive and is not further investigated in this thesis. On the other hand, the discriminator can be utilized as an One-Class Classifier (OCC) when the training dataset of the GAN includes no anomalies. This approach leads to significantly faster inference times. However, according to [116], this approach typically performs poorly compared to approaches utilizing the generator network. Figure 4.18 shows the distribution of discriminator output values. Even though normal images tend to lead to discriminator output values closer to 1, many anomalous samples also have a high discriminator

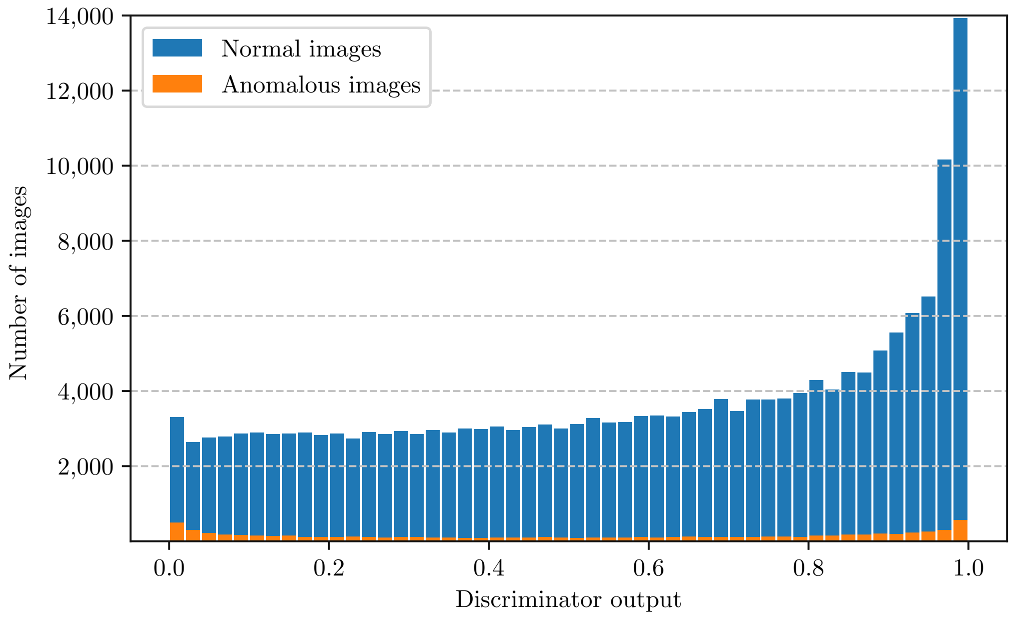


Figure 4.18: This figure shows the distribution of discriminator output values for images with and without anomalies. Normal images lead to higher discriminator output values than anomalous images, but there are also many anomalous images with high associated output values. Using a threshold value for the discriminator output to detect anomalies will inevitably lead to a high number of false positives.

output value associated with them. It turns out that the discriminator output value for anomalous samples is correlated with the size of the anomaly. Typically, larger anomalies like widespread vegetation on the trackbed lead to discriminator output values close to zero, while the output value is closer to one for smaller anomalies.

4.2 Architecture Decisions

The main goal of the proposed anomaly detection system VADAR is to detect unknown anomalies in images of rail tracks. Therefore, several NN-based approaches were analyzed, trained, and tested on the *Kombi* dataset. Since this dataset includes way more NOA images than anomalous images, the dataset is heavily imbalanced, posing a problem for binary classification approaches. As described in Section 2.4.2, reconstruction-based anomaly detection methods like AEs, DAEs, and VAEs are a good choice for anomaly detection approaches on unbalanced datasets and are often used for completely unsupervised approaches.

Maxi: Willst du wirkl
der Abkürzung haben
"NAO"/"NAOs" oder "

Figures 4.2, 4.4, 4.8, and 4.7 show the distributions of absolute reconstruction errors of images with and without anomalies for the proposed AE and DAE models. For all models, both distributions have mean values close to each other, and the distributions manifest within the same absolute reconstruction error range. One reason for this could be that the random nature of the ballast contributes a large portion of the total reconstruction error in an image, regardless of whether anomalies are included. Furthermore, this background reconstruction error value underlies a rather big variance due to varying ballast and lighting conditions. Since separate stones of the ballast come in different sizes, shapes, and brightness values, the AEs and DAEs seem to struggle with learning features to represent these stones without simultaneously improving the reconstruction of anomalies. Therefore, a clear separation of anomalous images from normal ones by introducing a threshold value for the absolute reconstruction error is not feasible. In addition to the sum of the absolute reconstruction error, the mean squared error loss [133], the structural similarity- and the multiscale structural similarity index [134] were considered. However, these different loss functions did also not enable an easy separation between normal and anomalous images. The VAE has the same issues, and neither the absolute reconstruction error nor the KL-divergence loss seem to be reliable indicators for anomalies. Although the discriminator of the DCGAN tends to output higher values for normal images, a similar trend for output values of anomalous images was not observed. However, the ten-layer AE and DAE seem to ignore objects lying on the trackbed and only focus on reconstructing the mean value of the ballast in the background. This is beneficial for an anomaly detection task since the anomalies lead to unusually large reconstruction errors in local regions. More sophisticated ways of analyzing reconstruction error images must be considered to detect and focus on such areas. This should allow VADAR to keep the number of false positives low while maintaining a high recall rate of anomalies. As an additional feature, AE-based approaches also enable anomaly localization with a relatively small additional effort. Since these regions of large reconstruction errors are the indication of anomalies, they also hold information about the position of the anomaly. This localization aspect significantly increases the interpretability of AE-based anomaly detection methods since the regions that were interpreted as anomalous can be highlighted. The following sections introduce the proposed anomaly detection system, called VADAR, andgive details about the design decisions, architectures, and training methodologies.

In [35], an AE-based anomaly detection method was introduced for a similar problem description. This paper focused on the detection and classification of large construction tools that were placed on the tracks of railroad systems. They used an unsupervised approach to train an autoencoder to specialize in reconstructing normal frames. The reconstruction error image is then fed to a second neural network to decide if this reconstruction error is the result of an anomalous data sample. In contrast to the dataset used in this thesis, the researchers in [35] worked with a collection of images taken from different cameras from a front-view perspective of the rails and trackbed. Besides RGB cameras, they focused on infrared cameras as well. Although there are significant differences in the problem description, the camera system, and the perspective of the cameras, they were facing similar challenges, and their approach could be adapted to also work for smaller anomalies like smaller unexpected objects on the trackbed or damages to the rails and crossties.

4.3 VADAR

Since the *Kombi* dataset contains close to 2,000,000 images (Section 3.1), the anomaly detection method of choice needs to be able to work with large quantities of data. The training and evaluation of the proposed algorithm, called VADAR, is limited to the original and augmented images of the *Kombi* dataset. Furthermore, parallelization of the computations should be supported to exploit the computational power of GPUs to reduce the computation time [64]. Machine learning algorithms, specifically neural networks, allow for high parallelization and can deal with large datasets [88, 64]. The total number of anomalies in the dataset is expected to be small because anomalies are rare in the labeled part of the dataset. Although different classes of anomalies like damages, trash, or dead animals were identified, another kind of object or damage could always appear in the unlabeled part of the dataset or the potential future use case of the system. Therefore, a binary classification-based approach is not a feasible method because only a small number of labeled anomaly images are available. Furthermore, there is no guarantee that this limited set of anomalies covers all or even the majority of interesting anomalies. This suggests an unsupervised or semi-supervised anomaly detection approach. Section 4.1.5 already summarized the analysis of several ML-based anomaly detection approaches for the given task, and AE-based approaches seem to be a suitable choice for this specific application.

Furthermore, Gasparini *et al.* described an AE-based approach for railroad inspection in [35, 127]. An AE is used to reconstruct the input images, and the reconstruction error is analyzed to decide whether the image contains any anomalies. In [35], a supervised approach was used to train a convolutional neural network that decides whether the image is anomalous. This work specialized in detecting construction tools that lie on the trackbed, and example images were available to train this classification network. A supervised approach like this does not lead to satisfying results for many anomaly detection tasks because not enough anomalous instances are available [17]. Some of the characteristics and limitations of supervised approaches are summarized in Section 2.2.3. Because of this significant drawback, different methods of detecting anomalies were explored. An additional rail segmentation network is used to differentiate between reconstruction errors of the rails and the rest of the image. Setting all pixel values of an image to one if its original value is at least as big as a threshold and to zero if it is smaller is referred to as thresholding. After thresholding in the rest of the reconstruction error image, only large coherent error regions within this image are further analyzed by a neural network. This network distinguishes between the gravel of the trackbed and parts of anomalous example objects. A block diagram of the proposed anomaly detection approach is shown in Figure 4.19. First, the next subsection discusses the necessary pre-processing steps for the input images. The following sections explain the separate blocks of this block diagram in more detail.

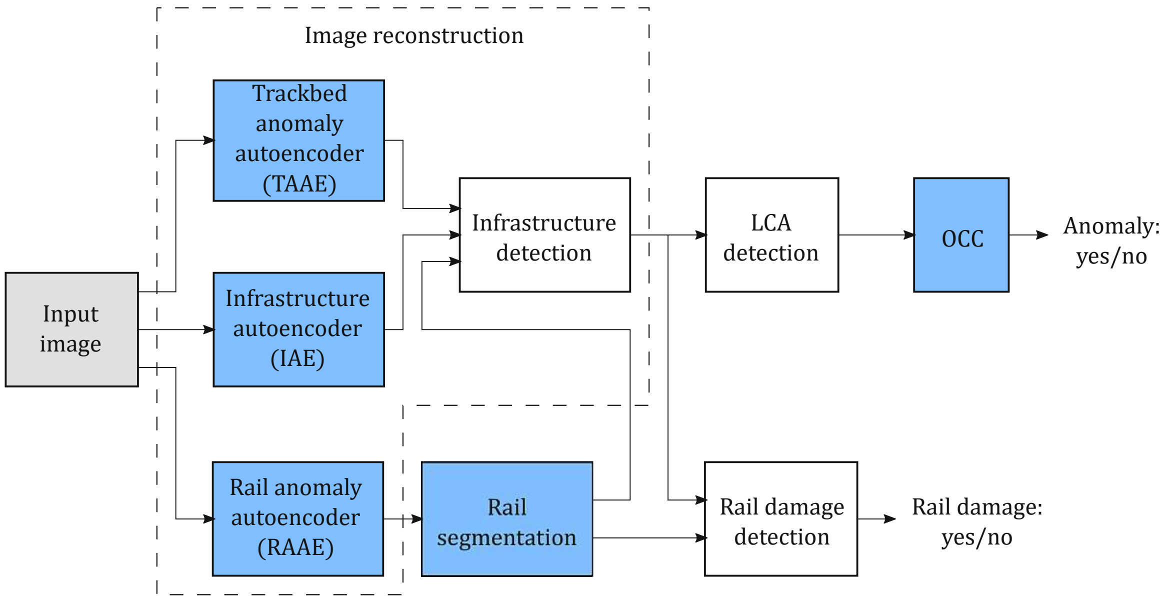


Figure 4.19: Three AEs are used for this approach. The Trackbed Anomaly Autoencoder (TAAE) and Infrastructure Autoencoder (IAE) share the same architecture, while the Rail Anomaly Autoencoder (RAAE) has an architecture with a wider bottleneck. The TAAE and RAAE are trained with NOA images, and the IAE is exclusively trained with INA images. To detect damages to the rails, a rail segmentation network is used, and the reconstruction error of the RAAE is analyzed. Large Coherent Areas (LCAs) within the reconstruction error image are then fed to a One-class classifier (OCC) to decide if this part of the image is anomalous.

4.3.1 Pre-Processing

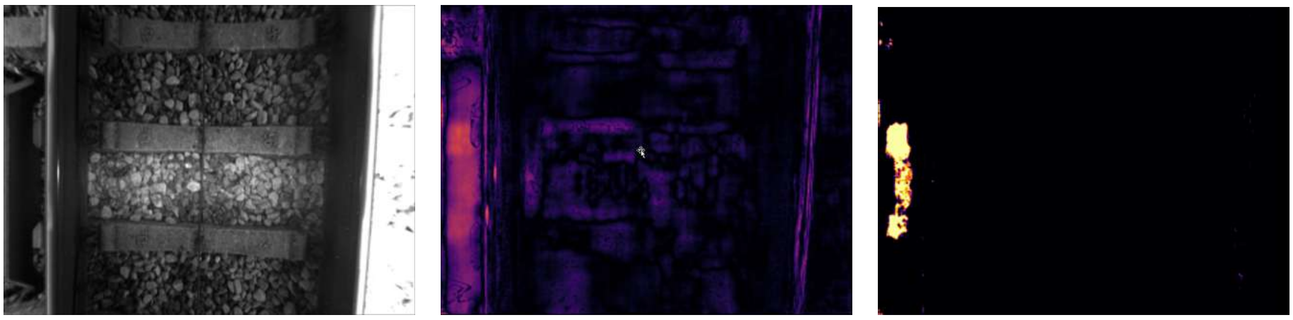
Because of the wide view angle of the camera system used to take the images for the dataset, some regions of the images are distorted and do not include useful information for the anomaly detection algorithm. Therefore, the images are cropped and only include the relevant parts, specifically the trackbed with the rails and crossties. After cropping, the images have a height of 769 pixels and a width of 1,025 pixels. Before any further processing, the images were normalized to values between 0 and 1. It should be noted that VADAR only accepts gray-scale images as input.

4.4 Trackbed Anomaly Detection

Trackbed anomalies are all anomalies except rail anomalies. This includes unexpected objects on the trackbed, like trash or animals, damage to the fasteners, and heavy vegetation. Unfortunately, a loss function like mean squared error or the combination of mean squared error and gradient error (described in section 4.4.2) could not be used as a reliable indication for anomalous frames. The main reason for this is the large contribution to the overall reconstruction error by the ballast of the trackbed and its large variation, which depends on the size of separate stones, their color, and the lighting conditions. Instead, only large coherent areas within the reconstruction error are seen as potential anomalies and analyzed in an additional step to lower the false positive rate of this anomaly detection approach. Another source for false positives is infrastructure elements. The following sections explain how VADAR detects anomalies on the trackbed and how the contribution of false positives by the ballast and infrastructure elements is lowered.

4.4.1 Image Reconstruction

The first step of the proposed anomaly detection method is to reconstruct the input image by two AEs. The trackbed anomaly detection uses the Infrastructure detection block to ignore infrastructure elements. Two AEs with the same



(a) Original image with an infrastructure element.

(b) Difference of TAAE's and IAE's absolute reconstruction error image.

(c) Detected and located infrastructure element.

Figure 4.20: Defining a threshold for the difference of TAAE's and IAE's absolute reconstruction error image enables the detection and localization of infrastructure elements.

architecture, but different types of training images (see Figure 3.2) are used to detect anomalies on the trackbed. The reason for two instead of one AE is that the reconstruction of infrastructure elements in INA images typically results in high reconstruction errors. Therefore, this leads to a larger number of false positives. The Infrastructure detection block utilizes the reconstruction error images of both AEs, which are called TAAE and IAE, to detect and ignore such infrastructure elements. The Infrastructure detection block is explained in more detail in 4.4.3.

4.4.2 Trackbed Anomaly Autoencoder and Infrastructure Autoencoder

Within the roughly 220,000 labeled images of the dataset, more than 180,000 images are images without annotations and are called NOA images, as described in Section 3.1.4. Furthermore, roughly 28,000 frames contain no anomalies but additional infrastructure elements like switch-frogs, other rails close to a switch-frog, crosstie attachments, or sensors. Since such infrastructure elements could falsely be identified as anomalous objects besides the TAAE, which is trained exclusively with NOA images, a second AE called IAE was developed that is specialized in the reconstruction of images containing infrastructure elements. The training procedure of the IAE only contains INA images. Because this specialized IAE performs better on INA images than the TAAE, the reconstruction error images of both AEs show relatively large differences where the infrastructure element is positioned. Since every infrastructure frame still contains many of the same fundamental features as normal images, the IAE performs well in reconstructing normal images. Therefore, the differences in reconstruction errors of the two AEs show only small differences in regions without additional infrastructure. This allows for localizing infrastructure elements that could otherwise be interpreted as anomalies.

Architecture

The AE consists of an encoder and decoder network, where the decoder network is symmetric to the encoder network regarding the layer structure. The AE was implemented in PyTorch, and the architectures of the encoder and decoder are shown in Tables 4.2 and 4.3, respectively. The architecture of the encoder consists of ten convolutional layers. Every layer uses a square kernel with a size of seven and zero padding of three on each side. All the layers are defined with a stride of two except the first and the last two, with a stride of one. Although some aspects of the architecture, like the number of layers, were inspired by the work [35], some parameters were adapted for this anomaly detection approach. For any AE design, the output dimensionality of the encoder, equal to the decoder's input dimensionality, must be considered. The higher the dimensionality of this latent representation of the input image, the more information is available for the decoder network to reconstruct the image [135]. Therefore, typically the output of the AE will be a more detailed reconstruction of the input image the higher the latent dimensionality is. Multiple factors of the chosen architecture influence this dimensionality. The dimensionality is the product of the number of output channels, width, and height of the last layer of the encoder. The number of output channels is just a parameter of the last encoder layer and can be defined directly, but all encoder layers influence the height and width. They can be computed by iteratively applying equations 2.5 and 2.6 for each layer. The resulting output widths and heights for each layer of the encoder and decoder are listed in Tables 4.2 and 4.3, respectively. This specific encoder architecture leads to a dimensionality reduction ϕ computed as

$$\phi = \frac{O_{C,Enc} \cdot O_{W,Enc} \cdot O_{H,Enc}}{I_{C,Enc} \cdot I_{W,Enc} \cdot I_{H,Enc}} = \frac{64 \cdot 9 \cdot 7}{1 \cdot 1025 \cdot 769} \approx 0.0051, \tag{4.3}$$

where $O_{C,Enc}$, $O_{W,Enc}$, and $O_{H,Enc}$ refer to the output number of channels, output width, and height of the encoder, respectively, and $I_{C,Enc}$, $I_{W,Enc}$ and $I_{H,Enc}$ represent the number of input channels and the width and height of the input image respectively. The dimensionality of the latent space representation is an important factor in dealing with the random nature of the ballast, which covers most of each image. Although the total reconstruction error of a frame can be reduced by increasing the latent space dimensionality, the reconstruction error is still dominated by the contributions of the gravel. Both AE models utilize the same architecture. This leads to an almost identical performance on normal frames.

Figure 4.21 shows the reconstruction and reconstruction error images of a frame of two AEs with different latent space dimensionalities. The reconstruction of some parts of the trackbed, like the rails, fasteners, and crossties, is far better than the rest of the trackbed, which is indicated in the reconstruction error images as black regions with minor reconstruction errors. Notably, the higher the latent space dimensionality, the more detailed the reconstruction image is. The reconstruction quality of the gravel is better for higher dimensional latent space representations, but this also leads to a better reconstruction of an anomaly. Since the ballast's contribution to the overall reconstruction error underlies a significant variation depending on the size and shape of the gravel and the lighting conditions, the summed-up absolute reconstruction error is not a reliable indicator for anomalies, as already discussed in Section 4.1.5. A more reliable indicator for anomalies was found in analyzing large coherent error areas within the reconstruction image. By using an AE with a lower dimensional latent space, the structure and texture of the ballast are entirely lost in the reconstruction image, but this allows objects to have a significantly larger impact on the reconstruction error image in the form of a large coherent error area as depicted in Figure 4.21. This indicator limits the threshold for large coherent error areas and, therefore, the size of anomalous objects to the size of separate gravel stones because unusually bright or dark stones will also result in similar error areas. Since most objects considered anomalies, like animals, heavy vegetation, or trash, like bottles, are significantly larger than a separate gravel stone, this limiting factor is not a significant drawback for this specific use case.

Training

For the TAAE, the utilized part of the dataset contained roughly 7,500 labeled NOA images. By rotating the images by 180°, mirroring them, and adding Contrast Limited Adaptive Histogram Equalization (CLAHE), the number of training images was effectively multiplied by 8 (see Section 3.2). This ensures that the TAAE is trained on a small portion of the available labeled data and still learns useful features for the reconstruction of NOA images only. 80% of this utilized part of the dataset was used as training data, and the remaining 20% as validation data. Similar splits of training data

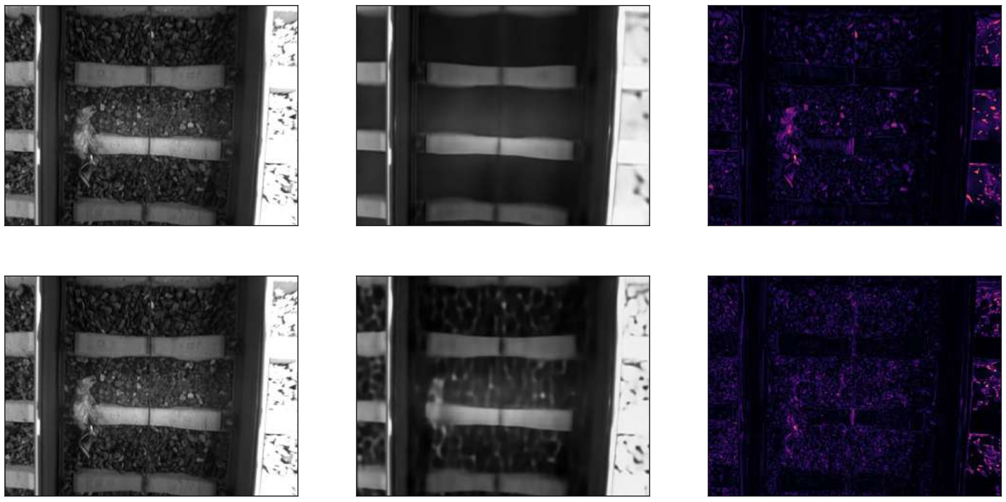


Figure 4.21: The two images in the leftmost column are the original input image. All images regarding the AE with a narrow bottleneck dimension of $(64, 7, 9)$ are shown in the top row. The images were produced by TAAE. The images in the bottom row correspond to the AE with a wider bottleneck dimension of $(64, 13, 17)$, where the image was reconstructed by the RAAE. The reconstruction images of both AEs are shown in the middle column, and the reconstruction error images are in the rightmost column.

are common practice in machine learning applications [35, 136, 137]. The *Adam*-optimizer was used with a learning rate of 0.001, and the AE model was trained for 100 epochs since the training and validation loss started stagnating. To evaluate the current model's performance, the deviation of the reconstructed image from the input image is considered by a loss function. Like in [35], a loss function was used that considers both the mean squared error loss L_{MSE} and a gradient loss L_G. The mean squared error loss is defined as

$$L_{MSE} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \parallel I(m,n) - O(m,n) \parallel_2^2, \tag{4.4}$$

where I and O represent the input and output image of the AE, and the gradient loss is

$$L_G = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \parallel G_I(m,n) - G_O(m,n) \parallel_2^2, \tag{4.5}$$

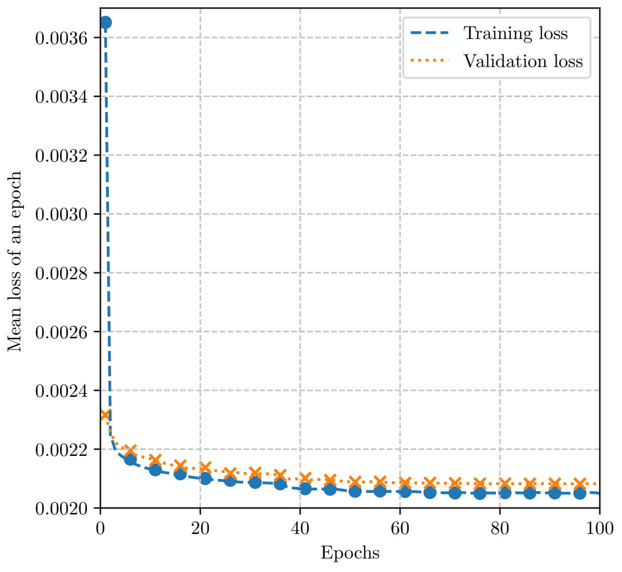
where $G(\cdot)$ represents the following gradient computation

$$G_x(I) = I * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \quad G_y(I) = I * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad G_I = \sqrt{G_x(I)^2 + G_y(I)^2}. \tag{4.6}$$

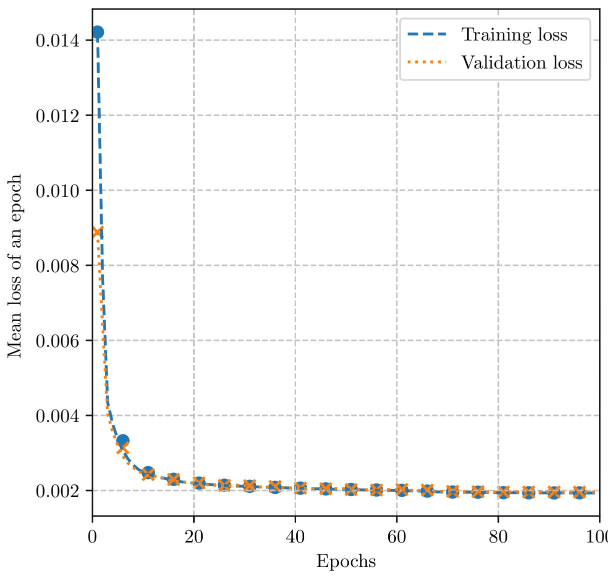
Here " $*$ " refers to a two-dimensional convolution. The overall loss L was then defined as

$$L = L_{MSE} + L_G. \tag{4.7}$$

The same optimizer, learning rate, number of epochs, and loss function were used for training the IAE. However, this AE is trained with INA images. From the available 28,675 labeled INA images, roughly 6,000 were used for training and split up to 80% training and 20% validation data. In Figure 4.22, the training progress of the TAAE and IAE are shown, respectively.



(a) Training progress of the TAAE.

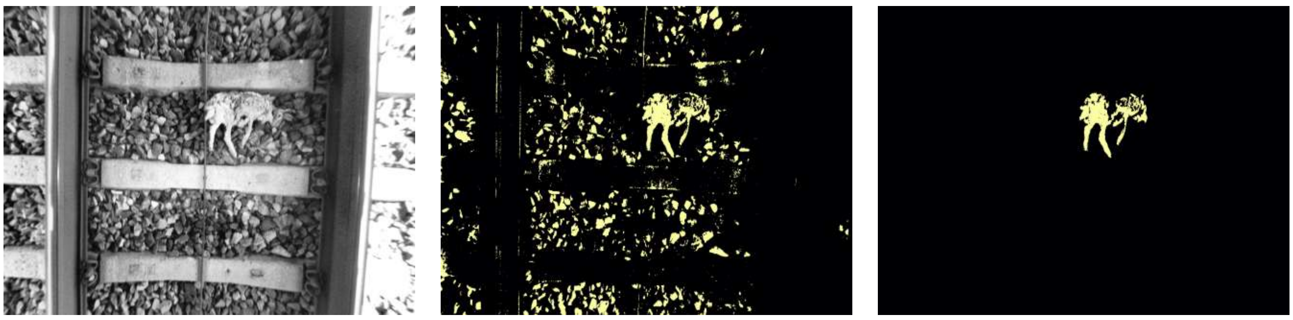


(b) Training progress of the IAE.

Figure 4.22: This figure shows the training and validation loss of the IAE during the training. The dashed and dotted lines show the smoothed training and validation loss.

4.4.3 Infrastructure Detection

Some infrastructure elements are responsible for the anomaly detection approach's increased false positive rate. Therefore, two AEs, namely the TAAE and IAE, are used to identify additional infrastructure elements within a frame to prevent it from being detected as an anomaly. To prevent these false positives, the TAAE is exclusively trained with NOA images, and the IAE is only trained with INA images. Therefore, the TAAE struggles with reconstructing anomalies and infrastructure elements. However, the IAE was only trained with INA images. Because the infrastructure elements are just a small part of the otherwise normal images, this IAE can reconstruct images with and without infrastructure elements. The reconstruction errors of each AE are the absolute difference between the original input image and the reconstructed image, and Figure 4.21 shows these images for two different AEs. Because there is a wide variety of infrastructure elements and only limited training data available, those elements might still be associated with a relatively large reconstruction error, but this reconstruction error is considerably smaller than when no infrastructure elements were included in the training. The infrastructure correction block in Figure 4.19 eliminates reconstruction errors resulting from infrastructure elements that are not damaged. One example of such a detected infrastructure element is shown in Figure 4.20. Even though the resulting mask does not cover the entire infrastructure element in every case, this still leads to a significant performance improvement. Section 5.1.1 discusses experiments to quantify this improvement. Looking at the absolute difference between the reconstruction error images of both AEs, regions with larger errors result from infrastructure elements that are not damaged and can be ignored for the following steps of anomaly detection.



(a) Original image with an anomaly. (b) Thresholded reconstruction error. (c) Large coherent area found in (b).

Figure 4.23: In the original image (a) an animal is on the trackbed. In (b) the thresholded reconstruction error image is shown. Only if a large coherent area within the reconstruction error image is found, the image is considered to be potentially anomalous. If this is the case, the output is the mask shown in (c).

4.4.4 Large Coherent Area Detection

Instead of using the full reconstruction error image, only large coherent areas are considered potentially anomalous regions. First, the reconstruction error image is thresholded to obtain a binary image. Pixel values smaller than a predefined threshold are set to zero, and all other pixel values are set to one. This threshold value is calculated using two parameters, the quantile q, and the minimum threshold value θ_{min}. The threshold value

$$\theta = \max\left(\text{torch.quantile}(I_{RE}, q), \theta_{min}\right), \tag{4.8}$$

where the *torch.quantile*-function of the PyTorch-framework [47], which returns the q'th quantile of the reconstruction error image, is used. Section 5.1.1 includes experiments that demonstrate the impact of these threshold parameters q and θ_{min} on the performance of VADAR. The *label* and *regionprops* functions of the *skimage.measure* package are used to obtain the largest areas within this binary image. If an area was found that is larger than a threshold value, this might be an indication of an anomaly and will be investigated further by an OCC. Figure 4.23 shows the thresholded reconstruction error image and the detected large coherent area as an example. This visualization demonstrates that anomalies leading to a coherent area significantly larger than separate stones of the gravel can be successfully detected. Experiments in Section 5.1.1 demonstrate the effectiveness of this approach.

4.4.5 One-Class Classifier

When the output of LCA detection is used as an anomaly detection approach, the largest contribution to false positives is gravel regions. Either parts of the gravel are exposed to significantly different lighting conditions than the rest, or some unusually bright stones are grouped close together on the trackbed. Therefore, an additional neural network is used as a one-class classifier to decide if the detected large coherent area within the reconstruction error resulted from gravel or an anomaly. A 64 × 64-pixel patch from the original image that includes the LCA or parts of it is used as the input for the OCC.

Architecture

A relatively simple convolutional network, consisting of only four convolutional and three pooling layers, is used to extract features of the image patch. After an adaptive average pooling layer, a small, fully connected network of three linear layers is used as the head of the classifier. Table 4.12 gives an architectural overview.

Table 4.12: All layers are implemented in PyTorch. The last linear layer uses the sigmoid activation function, and all other convolutional and linear layers utilize the rectified linear unit as an activation function. Every convolutional and linear layer, except Linear 3, is followed by a batch-normalization layer.

OCC Architecture					
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding
Convolutional 1	1	32	3	1	1
Convolutional 2	32	32	3	1	1
MaxPool	32	32	2	2	0
Convolutional 3	32	64	3	2	3
Convolutional 4	64	64	3	2	3
MaxPool	64	64	2	2	0
AdaptiveAveragePool	64	64	-	-	-
Flatten	-	-	-	-	-
Linear 1	256	64	-	-	-
Linear 2	64	32	-	-	-
Linear 3	32	1	-	-	-

Training

This classifier network was trained with both patches of anomalies and patches of gravel on the trackbed. Roughly 30% of the available anomalous samples were used to extract patches. From larger anomalies like animals, multiple patches were extracted to increase the total amount of anomalous data available for training. For the training procedure, roughly 6, 000 images containing only gravel and 6, 000 images containing anomalies were used. Through the combinations of mirroring, 0°, 90°, 180°, 270°, and CLAHE from each original patch 15 additional augmented samples were used for training. Figure 4.24 shows the training progress of the OCC.

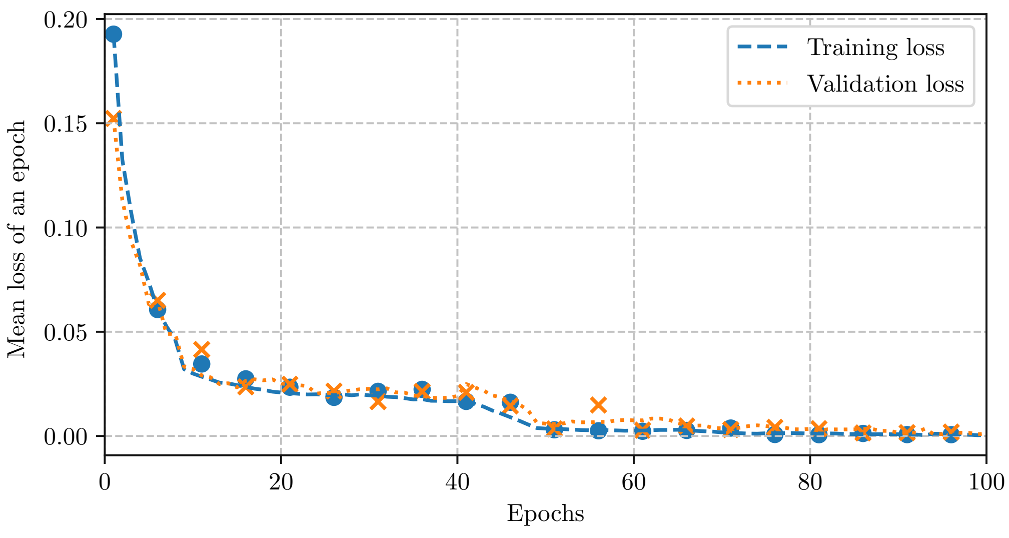


Figure 4.24: This figure shows the training and validation loss of the OCC during the training. The dashed and dotted lines show the smoothed training and validation loss.

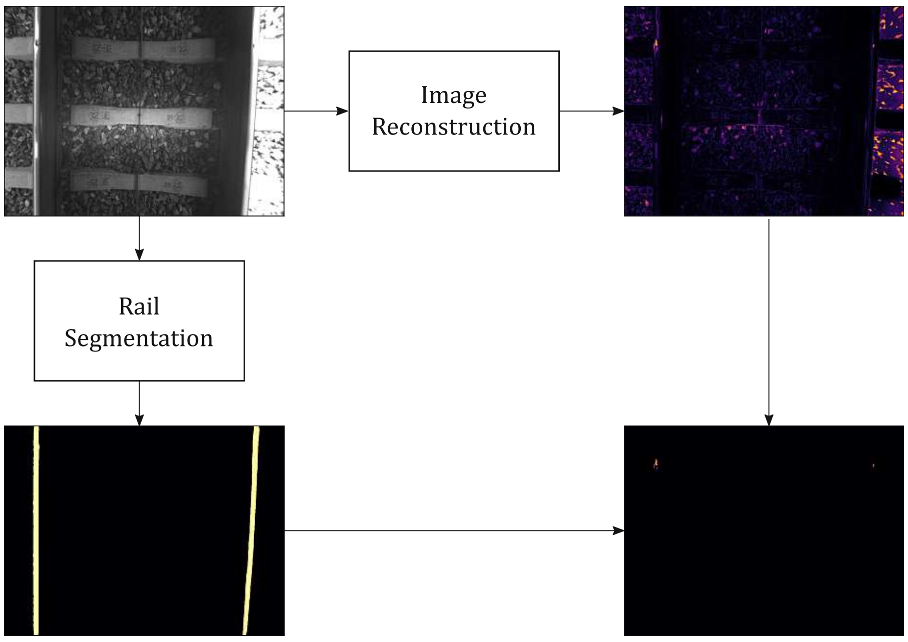


Figure 4.25: The original image with rail damage is shown in the upper left corner. Through the image reconstruction process, the reconstruction error image is obtained. The output of the rail segmentation network allows VADAR to focus on reconstruction errors of the rails.

4.5 Rail Anomalies

Unlike the detection of trackbed anomalies, the image reconstruction for rail anomaly detection is accomplished by one AE. This third AE, referred to as RAAE, has a different architecture than the TAAE and IAE and is exclusively used for the detection of anomalies on the rails. Detecting rail anomalies is of special interest since even small damages to the rail can lead to larger, more severe damages that can pose problems for safe and reliable transportation. The size of separate stones of the gravel sets a limit for detectable anomalies. In order to overcome this limitation for the detection of rail anomalies, an additional NN is used. The output of a rail segmentation network is applied to the reconstruction error of the RAAE. This network allows analysis of reconstruction errors of the rails. For rail damage detection, the summed-up rail anomaly pixel values are considered. Figure 4.25 shows an example image of a damaged rail and how it can be detected. The following sections describe the RAAE, the segmentation network, and rail damage detection.

4.5.1 Rail Anomaly Autoencoder

An AE with a wider bottleneck leads to reconstructed images with smaller reconstruction errors, as discussed in Section 4.1.1. However, if the bottleneck of an AE is too wide, even small image details are reconstructed. Since most images within the *Kombi* dataset show a trackbed covered in ballast, an AE with a wider bottleneck also learns features during the training to reconstruct these random stones (see Figure 4.6). Unfortunately, the features helping to reconstruct random stones also enable an AE to reconstruct small rail damages better. Therefore, there is also an upper limit for the bottleneck width of an AE to be useful as an anomaly detection approach. An appropriate AE architecture must be chosen for optimal performance of rail damage detection. Further experiments discussed in Section 5.1.2 show that the RAAE with the slightly wider bottleneck outperforms the TAAE in rail damage detection.

Architecture

In Tables 4.4 and 4.5, the architecture of the RAAE is described. Compared to the TAAE and IAE, the dimensionality of the RAAE's latent representation is higher. This encoder architecture results in a dimensionality reduction ϕ defined as

$$\phi = \frac{O_{C,Enc} \cdot O_{W,Enc} \cdot O_{H,Enc}}{I_{C,Enc} \cdot I_{W,Enc} \cdot I_{H,Enc}} = \frac{32 \cdot 13 \cdot 17}{1 \cdot 1025 \cdot 769} \approx 0.0090, \tag{4.9}$$

which is almost twice as large as for the other autoencoder architecture shown in Equation 4.3. This architecture allows for a more detailed reconstruction of images. Figure 4.21 shows the reconstructed images and the resulting reconstruction error images of the TAAE and RAAE.

Training

The training procedure for the RAAE is the same as for the TAAE, which is described in Section 4.4.2. The RAAE is trained with roughly 7,500 labeled NOA images. Rotating the images by 180°, mirroring them, and adding CLAHE multiplied the number of images by a factor of eight. The training procedure utilizes the *adam*-optimizer with a learning rate of 0.001, and the training lasts for 100 epochs. Figure 4.26 shows the training progress of the RAAE.

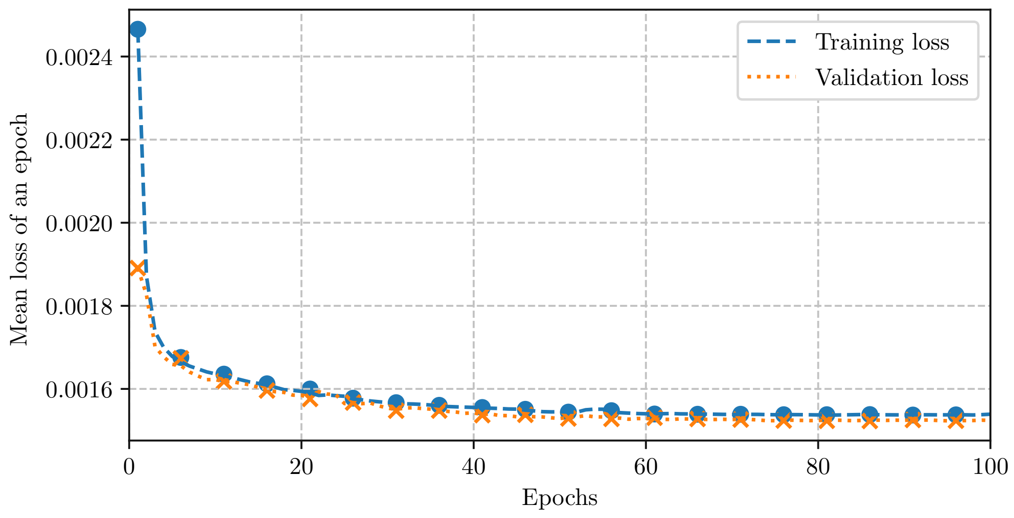


Figure 4.26: This figure shows the training and validation loss of the RAAE during the training. The dashed and dotted lines show the smoothed training and validation loss.

4.5.2 Rail Segmentation

A rail segmentation network is used in addition to the RAAEs to detect rail anomalies like damages. Figure 4.25 shows an output mask for one example image and how it is used to detect damages. The output of this network is a mask of the rails and is used to focus on the rail sections of an image for rail damage detection. This segmentation network allows separating the reconstruction errors of the rails from the reconstruction errors in all other areas.

Architecture

The architecture of this segmentation network is identical to the AE architecture discussed in section 4.4.2. Therefore, it also consists of an encoder and decoder network. Although the same architecture is used, this NN learns different features through the training process to extract the input image's rail segments and generate a rail mask instead of reconstructing the input image.

Training

For the training of this segmentation model, truth masks of the rails are needed for all training images. Since the camera system is mounted on the train, the rails do not change their positions relative to the camera system, except the train is turning. Therefore, the rails stay in the same position for multiple, sometimes even hundreds of consecutive images. The truth masks were obtained for roughly 5,000 frames of the dataset for different lighting conditions and fasteners to make the segmentation model more robust regarding different scenarios. These masks were defined manually, but a rail mask is often identical for hundreds of consecutive images. Additionally, the images and corresponding truth masks were shifted horizontally by a value between -10 and 10 pixels to make the model more robust to minor variations of the rail positions. These slight variations typically occur in real data during a turn.

Instead of training a completely new model, a transfer model approach was used to benefit from the already learned features of the trained AE. Unlike the AEs, the output pixel of the rail segmentation model should always be zero or one to indicate the position of a rail. Therefore, the binary cross-entropy loss was used in contrast to the AE training. Furthermore, the *Adam*-optimization algorithm with a learning rate of 0.001 was applied. After 40 epochs, the training was stopped since the training and validation loss stopped improving further. Figure 4.27 shows the training and validation loss.

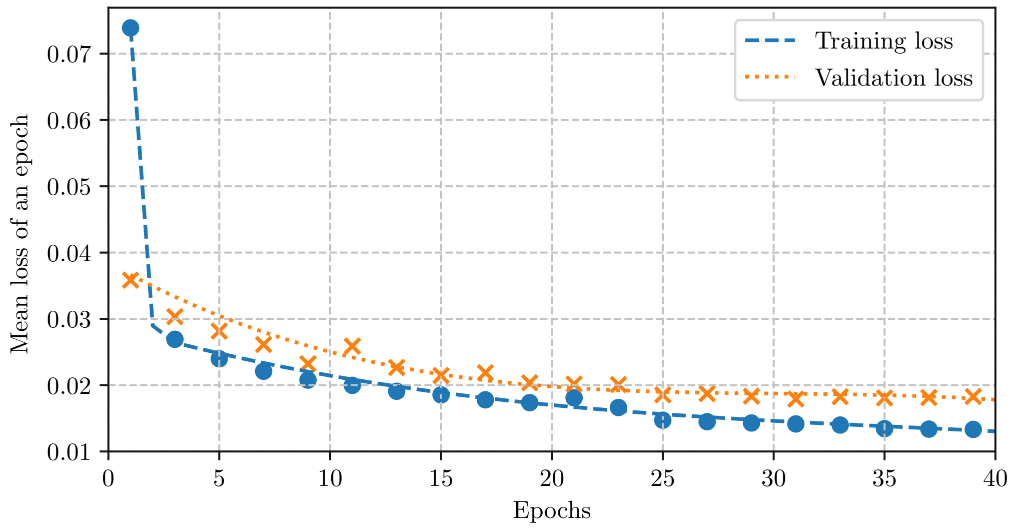


Figure 4.27: This figure shows the training and validation loss of the rail segmentation model during the training. The dashed and dotted lines show the smoothed training and validation loss.

4.5.3 Rail Damage Detection

For rail damage detection, the reconstruction error image of the RAAE and the rail mask generated by the rail segmentation network are used. Only pixels within this rail segmentation mask can contribute to the rail anomaly value. Although the RAAE reconstructs rails without damage well, small errors do occur. To keep the number of false positives low, threshold values are introduced to ignore small reconstruction errors of rails without damages. The first threshold value θ_{rp} marks rail pixels as potentially anomalous. Only if the reconstruction error value of a pixel is higher than θ_{rp}, this pixel is considered anomalous. In Figure 4.25, the image on the right bottom side highlights these pixels. Otherwise, the reconstruction error of this rail pixel is ignored. The second threshold value is referred to as *threshold* due to its significant role in the experiments presented in Section 5.1.2. If the summed-up reconstruction error value of all potentially anomalous pixels is above this *threshold*, the rails within this image are considered anomalous.

Chapter 5

Evaluation, Results, and Comparison

In this chapter, results of Vision-based Anomaly Detection Algorithm for Railroads (VADAR) are presented. The influence of several parameters of the approach upon performance factors like overall accuracy, recall rate, and false positive rate is analyzed. Furthermore, the influence of certain characteristics of anomalies on the recall rate is investigated. In the last section, the performance of the approach described in [35] is compared to VADAR. This work from Gasparini *et al.* focuses on a similar use case where their system detects anomalies on rail tracks by using cameras. Therefore, this seems to be a system suitable for a direct comparison. Additionally, the differences in both approaches are analyzed to explain some performance differences.

5.1 Evaluation Methods

To test the performance of VADAR, the algorithm's predictions on the whole labeled part of the *Kombi* dataset were analyzed. In Section 3 is a detailed description of the more than 220,000 annotated images, their characteristics, and annotations. Because only a few images are available in the dataset with damaged rails, it is difficult to grasp the characteristics and features of such damages that the algorithm is sensitive to. Therefore, artificial rail anomalies were added to normal images to analyze what characteristics of rail damage and its surroundings the anomaly detection algorithm is sensitive to. In conversations with domain experts and by reviewing state-of-the-art (described in Section 2.5.2), the main performance metrics analyzed are the accuracy

$$acc = \frac{TP + TN}{TP + FP + TN + FN}, \tag{5.1}$$

the recall rate

$$rec = \frac{TP}{TP + TN}, \tag{5.2}$$

and the false positive rate

$$fpr = \frac{FP}{FP + TN}, \tag{5.3}$$

where TP, FP, TN, and FN refer to the true positives, false positives, true negatives, and false negatives, respectively. Analyzing the recall rate of different types of anomalies separately allows a better understanding of what characteristics of anomalies VADAR is sensitive to. For anomalies on the trackbed, the areas of their bounding boxes in pixels were used to estimate the sizes of the according anomalies. Therefore, the influence of the size of anomalies on the recall rate was analyzed. The replication of rail damages enables a more detailed analysis. Since the proposed replication of rail damages described in Section 3.2.1 gives complete control over the size and position of these damages, the impact of damage size and background brightness on the recall rate was analyzed. It should be noted that one pixel in an image of the *Kombi* dataset corresponds with a size of roughly 2 mm^2 as already mentioned in Chapter 3. The following Sections present the results of the trackbed and rail anomaly detection algorithms. Besides a performance analysis of the proposed method, results obtained from only one Autoencoder (AE) are discussed as well to demonstrate the advantages of utilizing the combination of Trackbed Anomaly Autoencoder (TAAE), Infrastructure Autoencoder (IAE), and Rail Anomaly Autoencoder (RAAE). Furthermore, results are presented for multiple different parameters (e.g., thresholds) to demonstrate how these parameters influence the performance metrics. For evaluation, an *NVIDIA A100 40GB* [138] Graphics Processing Unit (GPU) was used.

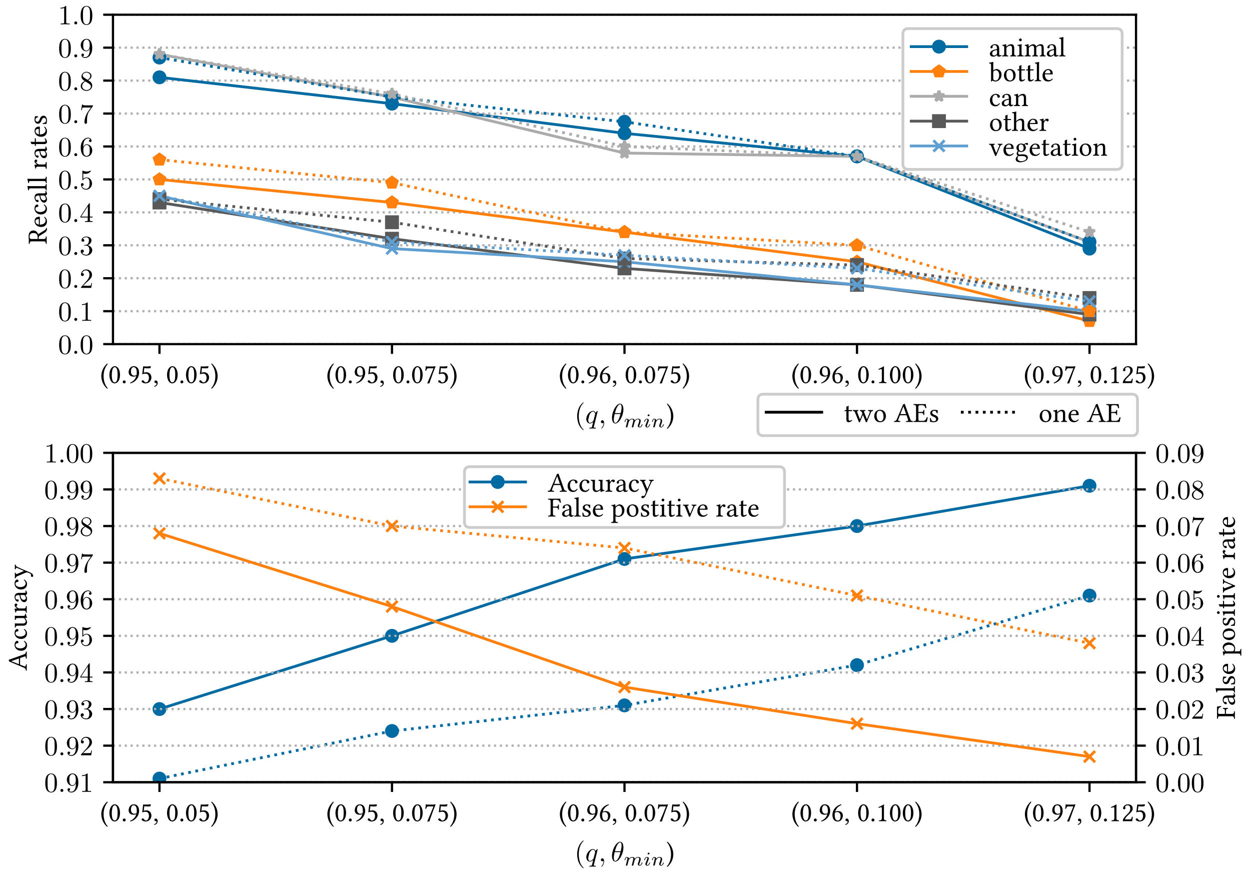
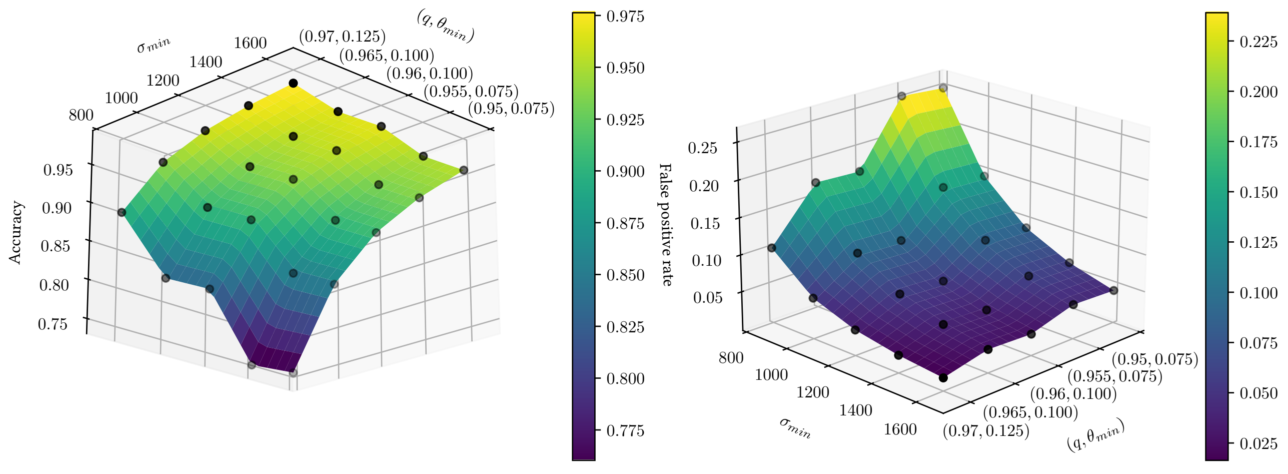


Figure 5.1: The upper plot shows the recall rates for different classes for five different threshold parameters, q, and θ_{min}, of the trackbed anomaly detection approach. The lower figure visualizes the achieved accuracy and false positive rate for these five threshold values. Furthermore, both plots also show the achieved results when only one AE and, therefore, no infrastructure detection is used.

5.1.1 Trackbed Anomaly Detection

Figure 5.1 shows the recall rates for each class of anomalies on the trackbed and the according accuracy and false positive rate for different threshold parameters q and θ_{min}, where q is the quantile and θ_{min} defines a minimum value



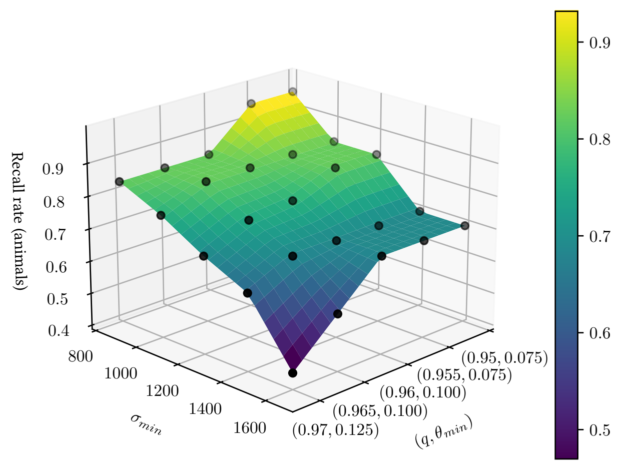
(a) Achieved overall accuracy.

(b) Achieved false positive rate.

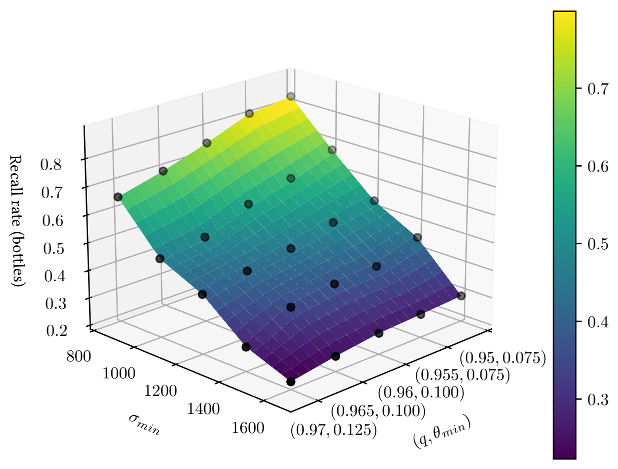
Figure 5.2: While the left figure shows the achieved overall accuracy, the right figure visualizes the false positive rate for various parameter settings. The black spheres represent the measured recall rates in experiments for the threshold parameters. Other points on the surfaces do not correspond with measurements but are colored for visualization purposes.

for the threshold θ as described in Equation 4.8. For better illustration, the size threshold σ_{min} was fixed to 1,200 pixels for these five experiments. The dotted data points represent the achieved results without infrastructure detection. Although the recall rates are slightly better without the infrastructure detection, the achieved overall accuracy and false positive rate are significantly worse. Since infrastructure detection aims to lower the number of false positives by ignoring regions that most likely contain infrastructure elements and no anomaly, a substantially lower false positive rate and higher accuracy are expected. As a negative side effect, the recall rate might also be lower when certain anomalies are falsely identified as infrastructure elements and ignored. However, the drop in recall rate seems to be small compared to an achieved drop in false positive rate, especially for the higher threshold settings. For the experiment $(q, \theta_{min}) = (0.97, 0.125)$, the false positive rate was reduced by a factor of four, while the recall rates for the various classes were only slightly reduced.

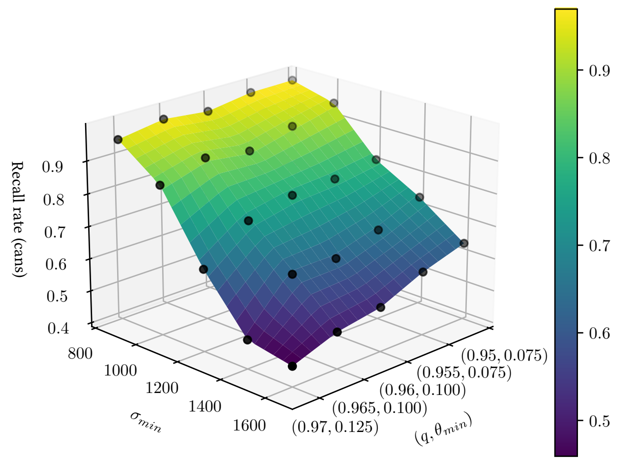
Figure 5.2 shows the dependencies of the overall accuracy and false positive rate on the threshold parameters (q, θ_{min}) and the size threshold σ_{min}. The higher the size threshold or threshold parameters, the higher the overall accuracy and the lower the resulting false positive rate. Simultaneously, the recall rates for anomalies decrease with higher threshold values, as depicted in Figure 5.3. Figures 5.2 and 5.3 visualize the trade-off between the achievable false positive rate and the anomaly recall rates. Although the recall rates for different anomaly classes vary, the dependencies on the parameters show the same trend. This trade-off needs to be considered for a specific application. Interestingly, the recall rates for anomaly classes like animals and cans are significantly higher than for other anomalies like vegetation. There seem to be particular characteristics about animals and cans the algorithm is more sensitive to. One significant characteristic of anomalies for the anomaly detector is the size of anomalies. Figure 5.4 visualizes how the minimum object size in pixels, the threshold parameters q and θ_{min}, and the size threshold σ_{min} influence the recall rate of trackbed anomalies. For this analysis, all trackbed anomalies except vegetation were considered. Images with vegetation were ignored because there are many instances where small plants cover large parts but not all of the trackbed. Since such plants are separate from each other, the bounding boxes are not suitable as a size estimation for the individual plants. For all σ_{min}, the recall rate for trackbed anomalies rises with the object size. For σ_{min} values up to 1,200 pixels depicted in Figures 5.4a to 5.4c, there are some parameter settings where all objects larger than 10,000 pixels are successfully detected. For $\sigma_{min} = 800$,



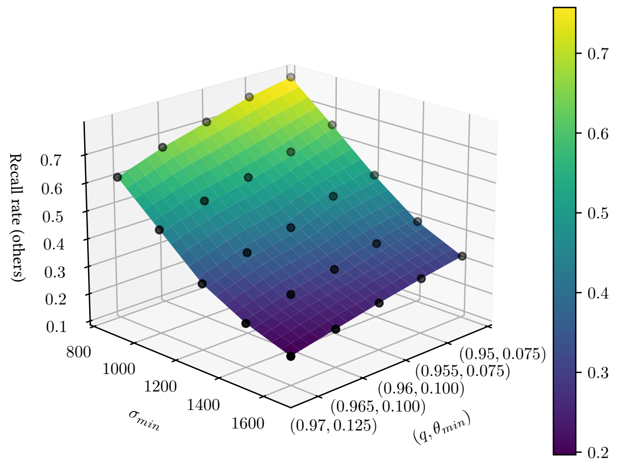
(a) Recall rate for animals.



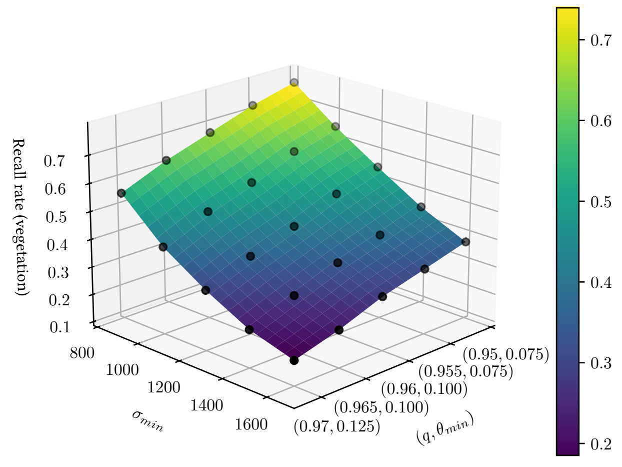
(b) Recall rates for bottles.



(c) Recall rates for cans.



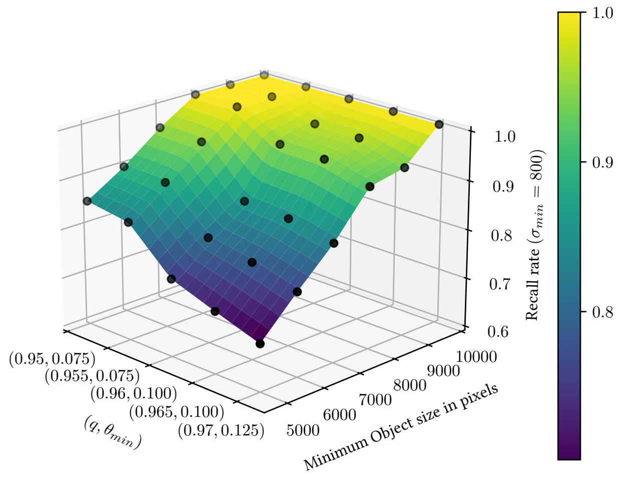
(d) Recall rates for others.



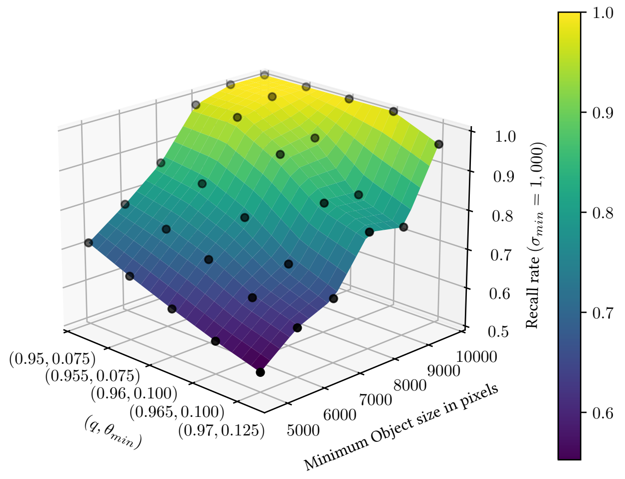
(e) Recall rates for vegetation.

Figure 5.3: These five 3D plots show the dependencies of the recall rates for different trackbed anomaly classes on the size threshold σ_{min} and the threshold parameters (q, θ_{min}). The black spheres represent the measured recall rates for experiments with the according threshold parameters. Other points on the surfaces do not correspond with measurements but are colored for visualization purposes.

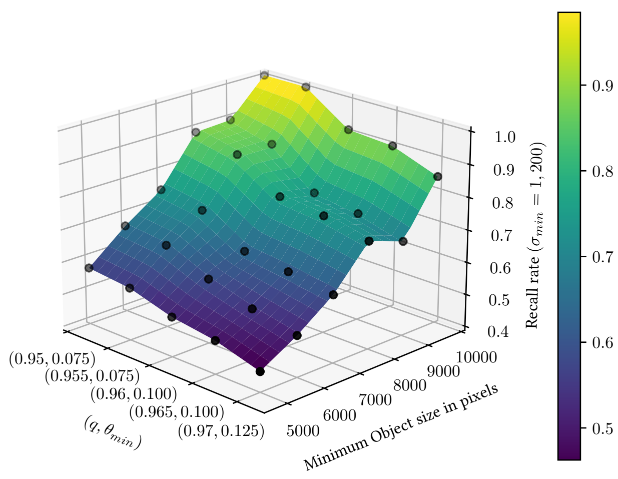
all tested q and θ_{min} pairs led to a recall rate of 100% for objects larger than 10,000 pixels. It should be noted that 10,000 pixels correspond with roughly 20,000 mm^2. By raising σ_{min} to 1,000, the only q and θ_{min} pair leading to a recall rate for objects larger than 10,000 pixels lower than 100% is $(q, \theta_{min}) = (0.97, 0.125)$. Setting σ_{min} to 1,200 leads to a 100%



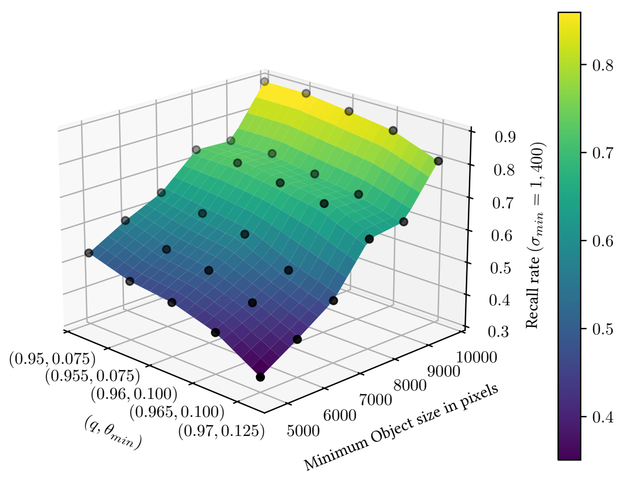
(a) Recall rates for the size threshold $\sigma_{min} = 800$ pixel.



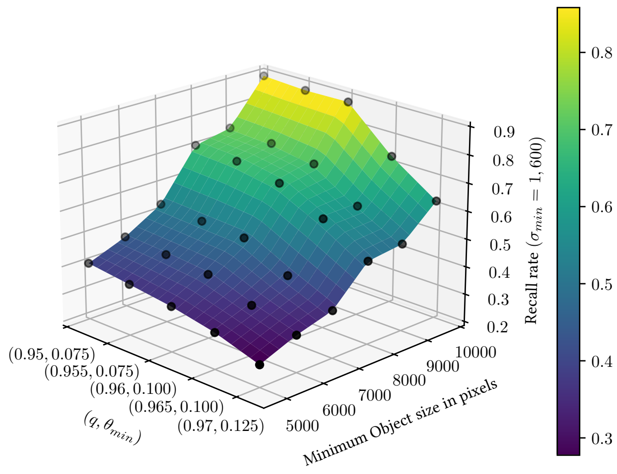
(b) Recall rates for the size threshold $\sigma_{min} = 1,000$ pixel.



(c) Recall rates for the size threshold $\sigma_{min} = 1,200$ pixel.



(d) Recall rates for the size threshold $\sigma_{min} = 1,400$ pixel.



(e) Recall rates for the size threshold $\sigma_{min} = 1,600$ pixel.

Figure 5.4: These six 3D plots show the dependencies of the recall rates for different minimum object sizes on the size threshold σ_{min} and the threshold parameters (q, θ_{min}). The pixels within a bounding box are used to estimate the object size. The black spheres represent the measured recall rates for experiments with the according threshold parameters. Other points on the surfaces do not correspond with measurements but are colored for visualization purposes.

recall rate of the largest objects for the two parameter pairs $(0.95, 0.075)$ and $(0.955, 0.75)$. Furthermore, VADAR can detect anomalies on the trackbed with bounding boxes larger than 5,000 pixels, with a recall rate of over 51% and a false positive rate of 4.8%. This is achieved for a σ_{min} of 1,200 and $(q, \theta_{min}) = (0.965, 0.100)$.

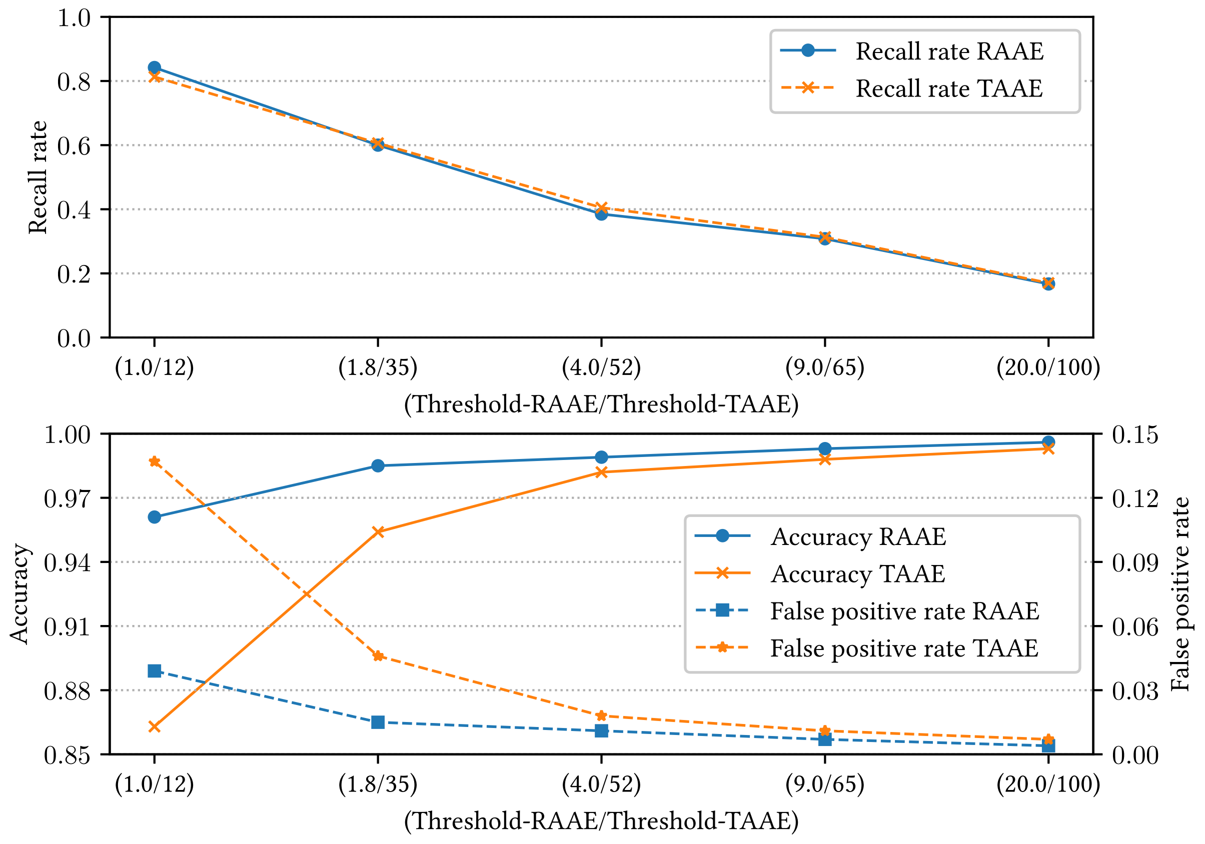
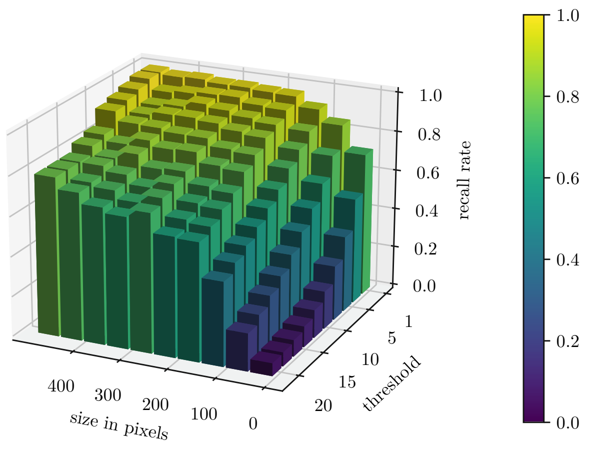


Figure 5.5: The upper plot shows the recall rates for rail damages within the labeled part of the *Kombi* dataset. Results are shown for the RAAE and TAAE. The lower plot visualizes the according accuracy and false positive rate, again, for both AEs.

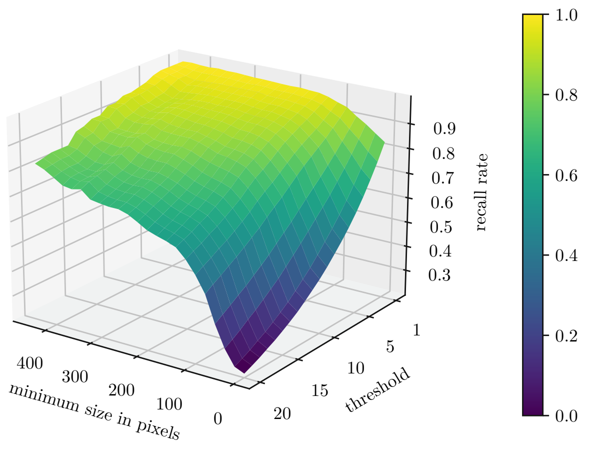
Figure 3.3 shows the number of objects of trackbed anomaly classes with bounding boxes of different sizes. Notably, 15 out of 19 animals have an according bounding box larger than 10,000 pixels. This influences the recall rate and explains why the recall rate is higher than for the other trackbed anomaly classes. One reason for the higher recall rates of cans might be the unusually high brightness differences in the background caused by light reflections on the metal cans. On the other hand, one reason for the lower recall rate of vegetation could be the inaccurate size estimation based on the bounding boxes. Most annotated vegetation is different kinds of grass that might only fill a small part of the according bounding box. Since there is a positive correlation between the recall rate of trackbed anomalies and their size, this might negatively influence the recall rate of thin and sparse vegetation. Additionally, many vegetation instances appear to have similar brightness levels as the ballast in the background. It should be noted that the parameters q and θ_{min} determine the threshold value for the pixel reconstruction error. Therefore, they control how sensitive the trackbed anomaly detection is to brightness differences between the anomalies and the background. Since reconstruction error pixels are only considered anomalous if their error value is larger than the threshold, the trackbed anomaly detection struggles with anomalies with smaller brightness differences.

5.1.2 Rail Anomaly Detection

This section presents experiments conducted to demonstrate the performance of the rail anomaly detection of VADAR. Figure 5.5 shows the recall rates, false positive rates, and accuracies for five different threshold values. Those metrics are plotted for two different AEs, specifically the RAAE and TAAE. To achieve the same recall rates with the TAAE, the threshold values were increased. This is necessary because the overall reconstruction error of the TAAE is significantly



(a) Recall rate for different rail damage sizes and several threshold values.



(b) Recall rate for different minimum rail damage sizes and several threshold values.

Figure 5.6: Figure (a) shows the achieved recall rate for different sizes of damages at various threshold values, and Figure (b) visualizes the recall rate for given minimal damage sizes. The recall rates increase with the size of the damages. Simultaneously, the lower the threshold value, the higher the recall rate.

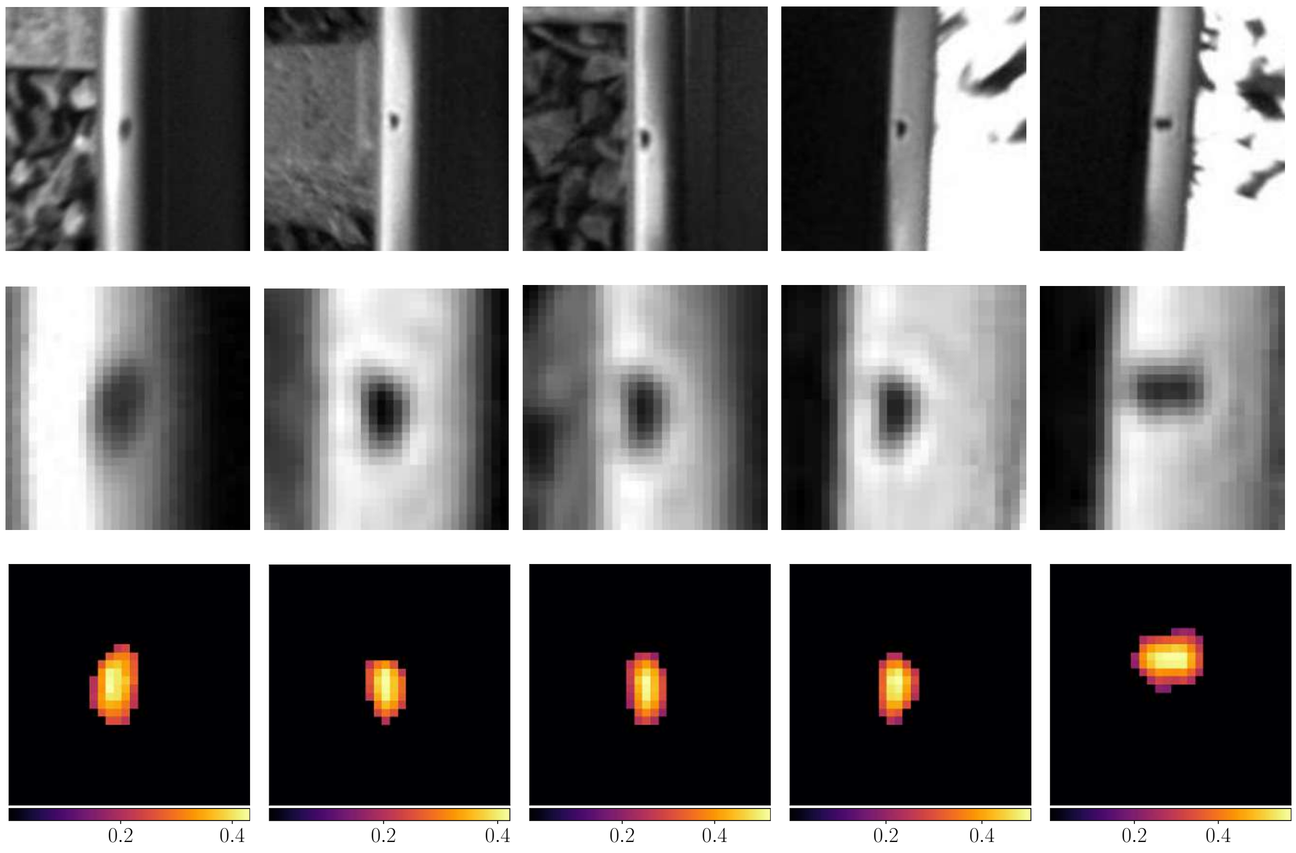


Figure 5.7: The images in the top row show indentations on different images of rails, and the second-row images are close-up versions of the same damages. Third-row images display the according patch of the reconstruction error images. The damaged rail displayed on the left-most column is a real image from the *Kombi* dataset, while the other four damages are replicated versions of this indentation.

larger than that of the RAAE. The accuracy and false positive rate plot in Figure 5.5 shows the significant performance differences between the two AEs for this specific task. While the RAAE and the TAAE achieve almost identical recall rates of rail damages, the RAAE outperforms the TAAE for every setting in accuracy and false positive rate. The most significant performance differences can be observed for smaller threshold values, which correspond with higher recall rates. The *Kombi* dataset's labeled part only includes a low number of rail damages. A higher number of damages of different sizes, with various surroundings and at multiple positions on the rail, allow a more detailed analysis. Therefore, some labeled rail damages were extracted, modified, and inserted on other images at different positions on the rails. By utilizing an AE with a high dimensional bottleneck, the inserted damages were adapted to the new surroundings of the image. A more detailed description of this synthetic image generation is given in Section 3.2.1.

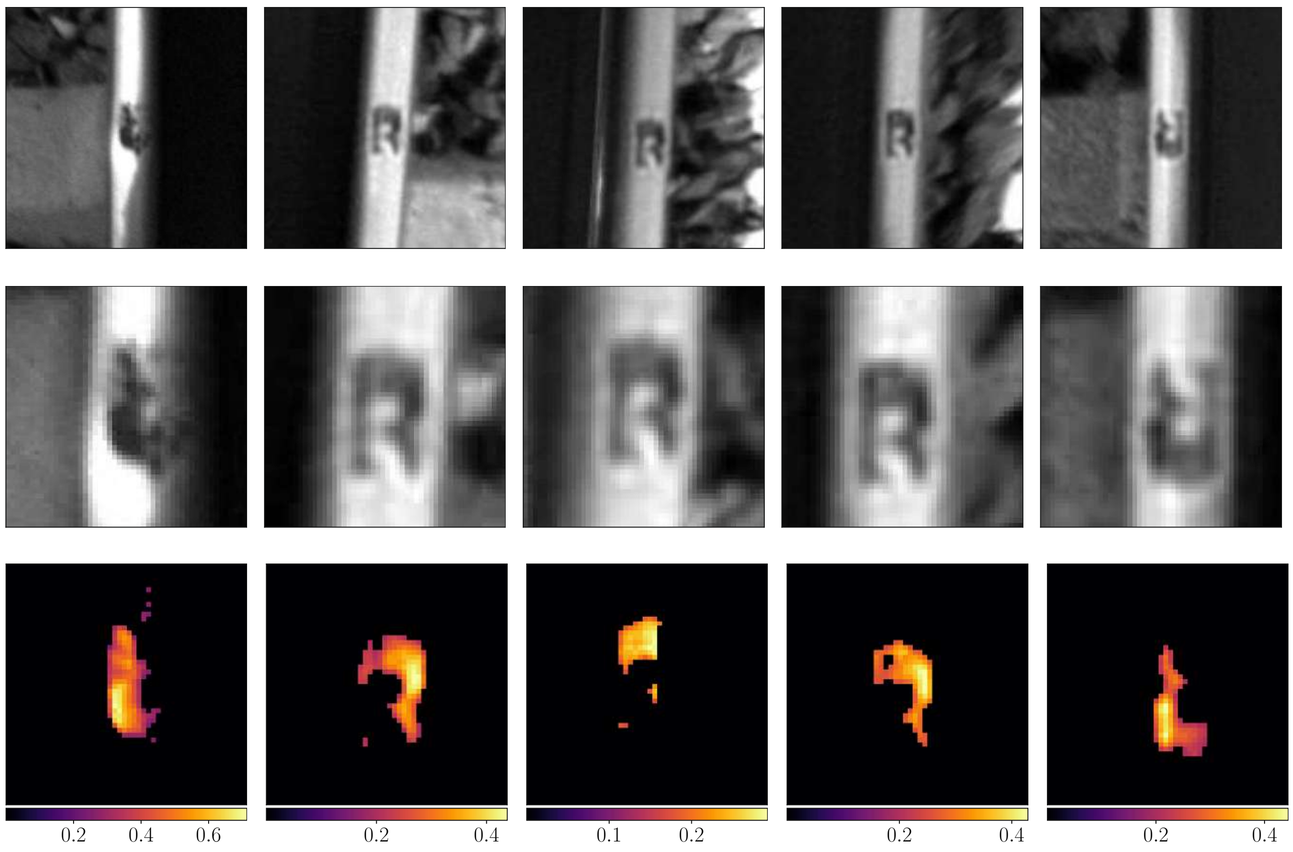
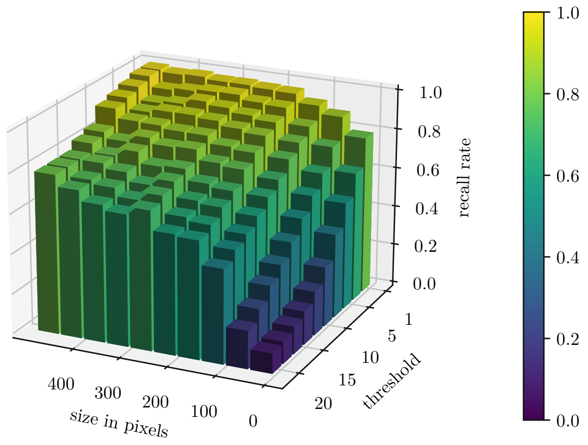


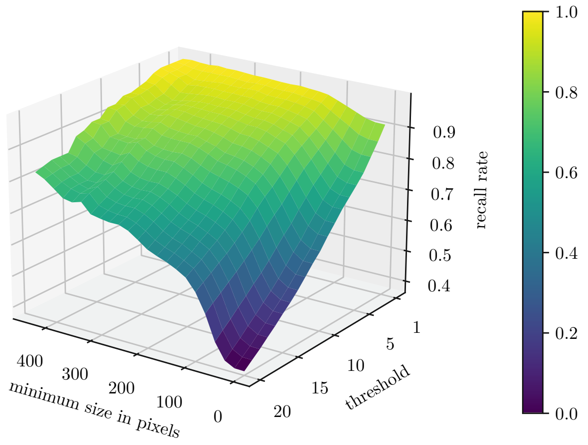
Figure 5.8: The images in the top row show breakouts on different images of rails, and the second-row images are close-up versions of the same damages. Third-row images display the according patch of the reconstruction error images. The damaged rail displayed on the left-most column is a real image from the *Kombi* dataset, while the other four damages are replicated versions of this breakout.

Experiments with Synthetic Data

Figure 5.6 shows the influence of the damage size and the chosen threshold value on the recall rate of the proposed rail damage detection method. Larger damages correlate with higher recall rates. Additionally, the influence of the damage size on the recall rate seems to be the largest for damages of a size up to 100 pixels. The synthetic anomaly instances can be grouped into two sorts of damages: indentations and breakouts. While indentations are typically small-scale damages, breakouts are usually larger rail damages. Figures 5.7 and 5.8 show examples of replicated rail damages, and a patch of the resulting reconstruction error image the anomaly detection is sensitive to. In both Figures, the left-most column shows images of the original image with a damaged rail. The second-row images show close-up versions of the damages shown in the first row. Images in the third row show the reconstruction error image and highlight the pixels the anomaly detection algorithm considers to be anomalous. For small damages like indentations shown in Figure 5.7, the AE-based seems to detect the majority of pixels of the rail damage and ignores all surrounding healthy pixels. In images of rails with larger-scale damages like breakouts shown in Figure 5.8, some parts of the damage are not fully detected. One reason for this is that some pixel error values are below the pixel threshold and, therefore, are not considered to be anomalous. Therefore, by lowering the pixel threshold, the number of anomalous pixels VADAR considers rises. Although this would increase the recall rate, this would also increase the number of false positives.

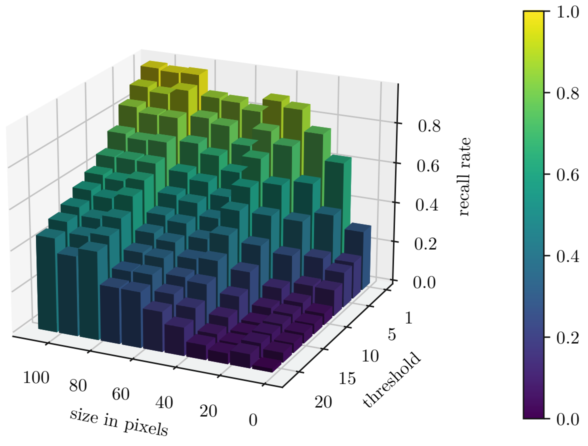


(a) Recall rate for different breakout sizes and several threshold values.

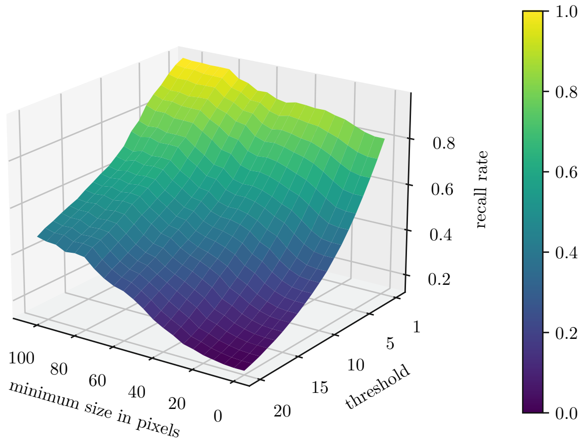


(b) Recall rate for different minimum breakout sizes and several threshold values.

Figure 5.9: Figure (a) shows the achieved recall rate for different sizes of rail breakouts at various threshold values, and Figure (b) visualizes the recall rate for given minimal damage sizes. Clearly, the recall rate increase with the size of the damages. Simultaneously, the lower the threshold value, the higher the recall rate.



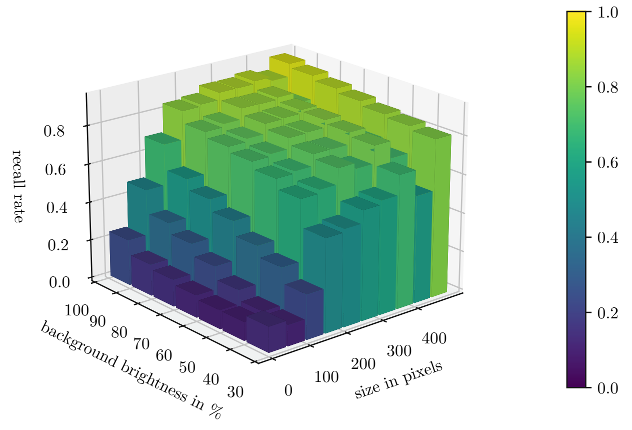
(a) Recall rate for different indentation sizes and several threshold values.



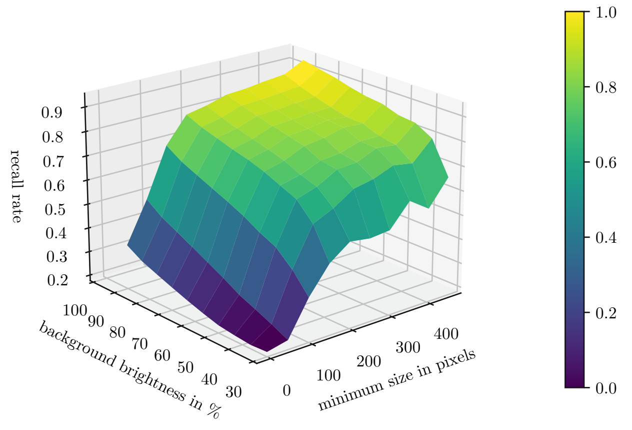
(b) Recall rate for different minimum indentation sizes and several threshold values.

Figure 5.10: Figure (a) shows the achieved recall rate for different sizes of indentations on rails at various threshold values, and Figure (b) visualizes the recall rate for given minimal damage sizes. Clearly, the recall rate increases with the size of the damages. Simultaneously, the lower the threshold value, the higher the recall rate.

Additionally, in some cases, the rail segmentation does not cover the entire head of the rail when rail damage covers a large area. Furthermore, in some instances, a few pixels corresponding to regions without damage can be treated as anomalous pixels. This can be seen in the reconstruction error image of the original breakout. Despite these inaccuracies in the pixel-level localization of large damages, all the shown damages are still successfully detected. Figures 5.9



(a) Recall rate for different rail damage sizes and several background brightness values.



(b) Recall rate for different minimum rail damage sizes and several background brightness values.

Figure 5.11: Figure (a) shows the achieved recall rate for different sizes of indentations on rails at positions with various background brightness values, and Figure (b) visualizes the recall rate for given minimal damage sizes. The recall rate increases with the size of the damages. Simultaneously, the brighter the background, the higher the recall rate.

and 5.10 visualize the influence of the damage size on the recall rate. As expected, there is a positive correlation between the size of damages and their recall rates. VADAR can detect rail damages larger than 50 pixels, corresponding with roughly 100 mm^2, with a recall rate of more than 51% while maintaining a false positive rate of roughly 4.2%. By lowering the *threshold* value, the recall rate increases even further at the cost of a higher false positive rate. Higher false positive rates negatively impact the practicality of VADAR because of an increased manual workload of reviewing VADAR's decisions to ignore false positives. However, other factors, like the brightness of the background, also impact the recall rate. Therefore, the mean brightness values of the position where the synthetic damages are inserted were also considered. Figure 5.11 shows how this background brightness influences the recall rate for various damage sizes. For

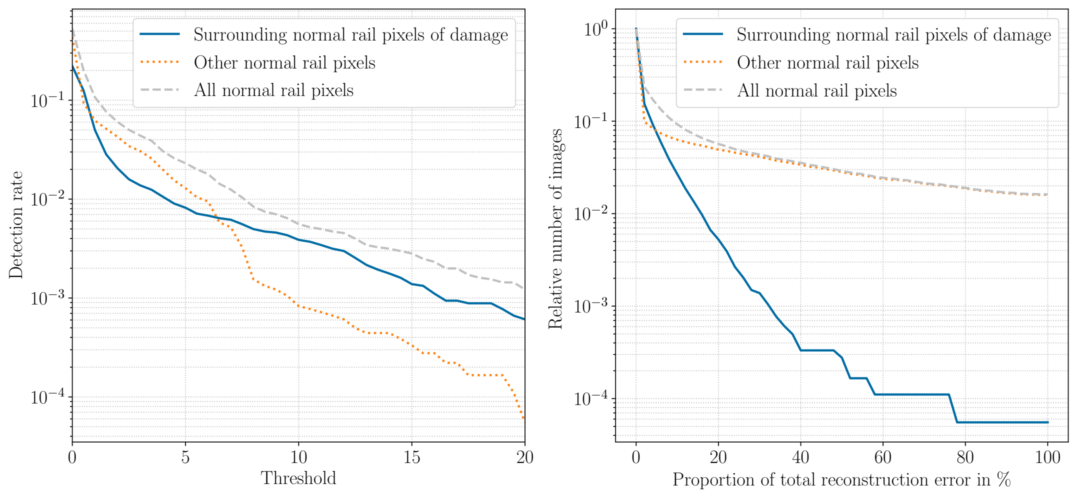


Figure 5.12: The left figure visualizes the influence of all types of rail pixels, except the actual damage pixels, on the detection rate. With larger threshold values, the detection rate drops significantly. At higher threshold values, the surrounding normal rail pixels of damage seem to lead to higher detection rates than other normal pixels. The right figure shows the relative number of images whose different regions of normal rail pixels contribute at least a certain proportion to the total reconstruction error. In only roughly 0.5% of images with synthetic damages, the surrounding pixels of damages contribute more than 20% of the total reconstruction error.

this experiment, the threshold value was fixed to a value of 10. There is a positive correlation between the background brightness and the recall rate. One likely reason for this is that brighter regions allow for larger brightness differences between the background and an inserted damage. This is the case because all observed damages in the *Kombi* dataset are significantly darker their its surrounding region on the rail. The larger the brightness difference between regions in the input image and its corresponding regions in the reconstructed image, the larger the chance of detection. Within the synthetic damage images, only a handful of images had a background brightness level lower than 30% and are not represented in Figure 5.11. For smaller damages, this correlation between the background brightness and the recall rate is stronger than for larger damages. Probably this is the case because the smaller the damage, the bigger the impact of the absolute reconstruction error on the anomaly detection decision. Larger damages are more likely to be detected because there is a larger number of corresponding error pixels. As long as the individual error pixel values are larger than the pixel threshold, they contribute to the reconstruction error sum. Since for smaller damages, the number of detectable error pixels is significantly smaller, the individual error pixel values seem to have a bigger impact on the recall rate.

The proposed rail damage detection method is sensitive to the summed-up reconstruction error of the image regions covered by the rail segmentation output. For the synthetic rail damages, a more detailed analysis of which pixels the anomaly detector considers to be anomalous and how much these pixels contribute to this summed-up reconstruction error is important. The surrounding pixels of synthetic damages are of special interest. If the anomaly detector is sensitive to the surrounding pixels rather than the pixels of the damage, this might indicate low-quality synthetic damage. Figure 5.12 shows the influence of normal pixels on the detection rate and the contribution of the normal pixel's reconstruction error to the total summed-up reconstruction error. The pixels represented by the solid blue line in Figure 5.12 refers to a rectangular region surrounding rail damage, excluding the pixels of the damage. The rectangle's border is expanded by five pixels in each direction to include the transition from the damage to the healthy rail. The left plot in Figure 5.12 shows how often the summed-up reconstruction error of normal rail pixels is larger than the threshold value, leading to damage detection. The right plot in Figure 5.12 visualizes the relative number of images where different regions of normal rail pixels contribute at least a certain percentage of the total reconstruction error. Overall, normal rail pixels other than surrounding rail pixels of damages dominate the total contribution to the reconstruction error. While in only roughly 0.5% of images with synthetic damages, the surrounding normal rail pixels of damages contribute more than 20% of the total reconstruction error, in less than 6% of images, all normal rail pixels contribute at least the same amount. Interestingly, the total reconstruction error stems exclusively from normal rail pixels in roughly 1.5% of all images with synthetic rail damages. Therefore, these images' artificial rail damage pixels do not contribute to the reconstruction error. Either the brightness difference between the damage and the rail head is too small, or the rail segmentation mask does not include the artificial rail damage pixels.

5.2 Comparison with State of the Art

In [35], Gasparini *et al.* propose an anomaly detection approach for inspecting railroad systems (see Section 2.5.2). Their system also analyzes the reconstruction error of an AE. However, their approach utilizes a binary classification network to decide whether the absolute and gradient error images indicate an anomalous data point. An overview of this anomaly detection method is shown in Figure 2.8 and is described in more detail in Section 2.5.2 and in [35]. The results Gasparini *et al.* achieved on their *Vesuvio* dataset are summarized in Section 5.2.1. Since this dataset is not publicly available, their models were also implemented and trained on the *Kombi* dataset. This way, a fair comparison between the performances of the two approaches on the *Kombi* dataset can be presented. A slightly different resolution was chosen to apply their approach to the *Kombi* dataset. In [35], each image of the *Vesuvio* dataset was cropped to a square image with a resolution of 192×192. However, the crops of the images from the *Kombi* dataset that still contain the rails and crossties are rectangular and have a resolution of 768×1024. A maximum pooling layer was applied to adapt the resolution to 192×256 without distorting the images. The first stage of this anomaly detection approach is an AE. Tables 5.1 and 5.2 show the encoder and decoder architectures. The absolute difference between the original and the reconstructed image and their gradient differences are used as inputs for the last stage of the anomaly detection algorithm. In Table 5.3, the architecture of this binary classification network is shown.

Table 5.1: All layers are two-dimensional convolutional layers implemented in PyTorch. Each convolutional layer is followed by a leaky Rectified Linear Unit (ReLU) activation function.

Encoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
LAY 1	1	16	3	1	1	192	256
LAY 2	16	32	3	2	1	96	128
LAY 3	32	64	3	2	1	48	64
LAY 4	64	128	3	2	1	24	32
LAY 5	128	256	3	2	1	12	16
LAY 6	256	512	3	2	1	6	8
LAY 7	512	1024	3	2	1	3	4
LAY 8	1024	1024	3	1	1	3	4
LAY 9	1024	1024	3	1	1	3	4

Table 5.2: All layers are two-dimensional transposed convolutional layers implemented in PyTorch. Each convolutional layer, except the last one, is followed by a leaky rectangular linear unit activation function. A sigmoid activation function is applied after the last layer. The decoder network is symmetric to the encoder network regarding layer structure.

Decoder Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
LAY 1	1024	1024	3	1	1	3	4
LAY 2	1024	1024	3	1	1	3	4
LAY 3	1024	512	3	2	1	6	8
LAY 4	512	256	3	2	1	12	16
LAY 5	256	128	3	2	1	24	32
LAY 6	128	64	3	2	1	48	64
LAY 7	64	32	3	2	1	96	128
LAY 8	32	16	3	2	1	192	256
LAY 9	16	1	3	1	1	192	256

Table 5.3: The layers *Conv 1* to *Conv 8* are two-dimensional convolutional layers, and layers *Linear 1* and *Linear 2* represent fully connected layers. Each of these layers is followed by a rectangular linear unit activation function except the last one, which is followed by a sigmoid activation function.

Binary Classifier Architecture							
Layer	Input Channel	Output Channel	Kernel Size	Stride	Padding	Output Height	Output Width
Conv 1	2	8	3	1	1	192	256
Conv 2	8	16	3	2	1	96	128
Conv 3	16	32	3	2	1	48	64
Conv 4	32	64	3	2	1	24	32
Conv 5	64	128	3	2	1	12	16
Conv 6	128	256	3	2	1	6	8
Conv 7	256	256	3	2	1	3	4
Conv 8	256	256	3	1	1	3	4
Flatten	-	-	-	-	-	-	-
Linear 1	3072	48	-	-	-	-	-
Linear 2	48	2	-	-	-	-	-

5.2.1 Results

In [127], Gasparini *et al.* executed all of their experiments on the *Vesuvio* dataset. With an RGB camera setup, they achieved an overall accuracy of 81.1%, a precision of 97.9%, a recall rate for anomalies of 71.9%, and a resulting F1-score of 82.5%. Because this dataset is not publicly available, their approach was implemented and tested on the *Kombi* dataset to compare the results. Figure 5.13 shows the recall rates for various anomalies and the recall rates for different object sizes.

The approach described in [35] was developed for the *Vesuvio* dataset. The proposed approach consisting of an AE and a binary classification network was implemented, trained, and tested with the *Kombi* dataset to compare the performance to the proposed approach of this thesis. However, some significant differences between these two datasets should be noted. The anomalies in the *Vesuvio* dataset are large objects placed on the trackbed. Construction tools and objects common in construction sites near railroad tracks were placed on the trackbed. Although in [35], it is not explicitly mentioned how many anomalous frames are within this dataset, at least the percentage of true positives, false positives, true negatives, and false negatives of the total number of outcomes can be calculated from their presented results. Specifically, from Equations (5.1) to (5.3), the ratio between normal and anomalous images can be calculated. It turns out that roughly 53% of the dataset contains anomalous images. Since this dataset seems to have an almost equal number of normal and anomalous images, a binary classification approach seems reasonable and leads to good results. However, the *Kombi* dataset is unbalanced since there are far more normal than anomalous images. Therefore, a direct comparison of precision for the two different datasets is unreasonable. Furthermore, the *Kombi* dataset contains images that include additional infrastructure elements like switches that are not considered anomalous. This aspect of railroad tracks was not considered in [35, 127] and might not be represented in the *Vesuvio* dataset.

Figure 5.13 shows a comparison of the recall rates of various anomaly classes and object sizes between the state-of-the-art approach by Gasparini *et al.* and two settings of the VADAR method are shown. The parameter setting for VADAR

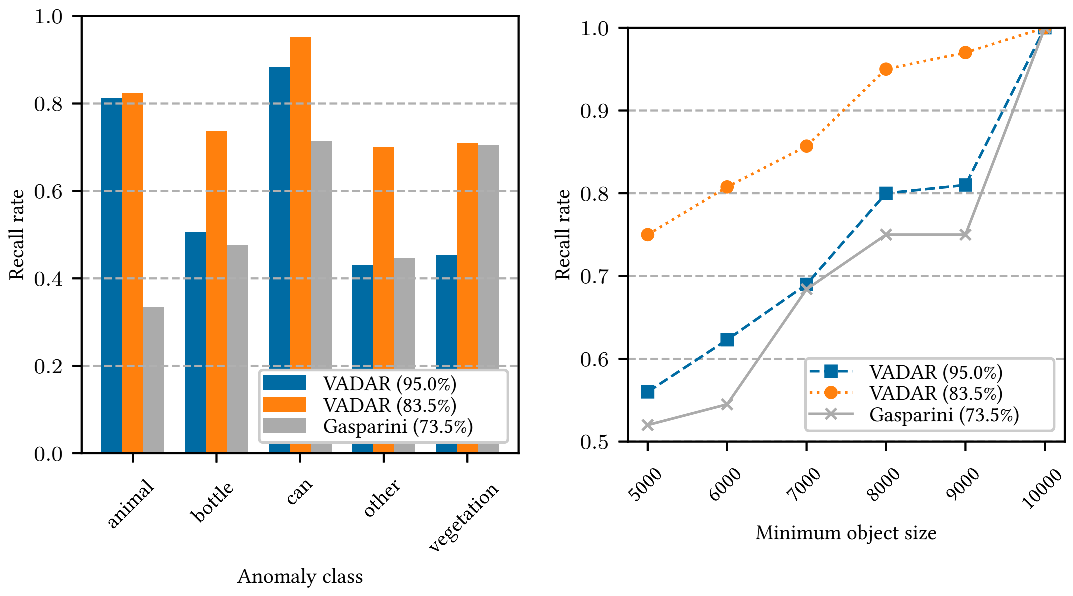


Figure 5.13: The left and right figures show the recall rates for different anomaly classes and object sizes for the state-of-the-art approach by Gasparini *et al.* and two settings of the VADAR method, respectively. The percentages within the brackets refer to the achieved accuracy of the methods and parameter settings. The recall rates vary for different anomaly classes and continuously increase with the object size.

that leads to an overall accuracy of 95.0% achieves better or equal recall rates for every minimum object size and anomaly class but vegetation. This performance difference is probably due to the supervised approach that the Gasparini method is based on. Since most anomaly instances belong to the class vegetation (see Table 3.2), the classification network is best at detecting these anomalies. Simultaneously, this approach performs worst with the instances of the animal class because this is the anomaly class with the lowest number of instances within the *Kombi* dataset. By changing the threshold parameters of VADAR, the overall accuracy decreases to 83.5%, which is still significantly higher than the accuracy of the state-of-the-art approach of 73.5%, and the recall rates outperform the state-of-the-art approach for every anomaly class and object size.

Chapter 6

Conclusion

Maintenance of the rails and the trackbed of railroad systems is necessary to ensure the safety and reliability of railroad tracks. Nowadays, rails and the trackbed are inspected thoroughly but infrequently by slow and expensive measurement vehicles or continuously but superficially by trained personnel. Therefore, the inspection of rail tracks is an expensive task. Due to the high costs, rail track measurement vehicles check some rail tracks only twice a year. Installing a cost-effective camera and computing system on regular trains Vision-based Anomaly Detection Algorithm for Railroads (VADAR) could pose a cost-efficient opportunity for continuous automated inspection of rail tracks. An automated system for inspecting rail tracks takes on a key challenge and could potentially enable predictive maintenance. Rail damages are of special interest to maintenance teams because appropriate actions can lead to increased reliability of the railroad system. Therefore, detecting even small damages, and scheduling maintenance tasks accordingly, can prevent such damages from getting more severe. This can further increase the safety and reliability of rail tracks. Besides foreign objects like bottles and dead animals on the trackbed, heavy vegetation can also lead to problems.

The proposed anomaly detection algorithm VADAR analyzes gray-scale images taken from a birds-eye view monochrome camera mounted on the bottom of a regular train. The images taken from this perspective, combined with a system based on three Autoencoders (AEs), allow for detecting foreign objects on the trackbed and even smaller rail damages. Since this camera system and additional hardware for computations and data transmission can be installed on regular passenger and freight trains, this vision-based anomaly detection system poses a comparably cost-effective solution for railroad inspection. Furthermore, this system enables inspecting rail tracks on every train ride, which is not feasible with specialized maintenance trains due to high costs.

Varying lighting conditions, several different crosstie materials, additional infrastructure elements like switches, and the random nature of ballast on the trackbed pose challenges to an anomaly detection algorithm. In the course of this thesis, various AE architectures were tested. Additionally, two Denoising Autoencoders (DAEs) with different architecture and noise models and a Variational Autoencoder (VAE) were implemented and analyzed. While neither the summed-up reconstruction error nor the latent space analysis by t-distributed Stochastic Neighbor Embedding (t-SNE) proved to be suitable indicators for anomalies, the Rail Anomaly Autoencoders (RAAEs) performed well on the reconstruction of rails, and the Trackbed Anomaly Autoencoder (TAAE) proofed to be useful for the detection of anomalies on the trackbed. A rail segmentation network helps to separate rails' reconstruction errors from the trackbed's reconstruction errors. Therefore, the rail segmentation enables an independent analysis of the trackbed and rails for anomalies. Indicators for anomalies on the trackbed are large coherent areas within the thresholded reconstruction error image. An additional One-Class Classifier (OCC) decides if the input image patch leading to this large coherent area includes an anomaly or only contains ballast. While the minimum size of detectable anomalies on the trackbed correlates with the size of separate ballast stones, this limitation does not apply to the rails due to its independent analysis. Since the reconstructions of rails without damages show only minor reconstruction errors, the summed-up reconstruction error is a reliable indicator for rail anomalies. Instead of using only one AE, the performance of the anomaly detection algorithm was improved by introducing two additional AEs. The Infrastructure Autoencoder (IAE) was exclusively trained with images that include additional infrastructure elements to reduce the reconstruction errors of such elements. Combined with the TAAE, which was trained with image containing no annotationss (NOA images) only, the differences in the reconstruction error images indicate regions belonging to such infrastructure elements and are ignored. While the RAAE and TAAE were trained exclusively with images without any annotations, the RAAE led to significant performance improvements on the rail damage detection task.

Threshold parameters of the trackbed anomaly detection allow for a tradeoff between high recall rates of anomalies and low false positive rates. Trackbed anomaly detection experiments for accuracy values between roughly 75% and 99% were conducted. While there is a positive correlation between the size of trackbed anomalies and their recall rate, other characteristics of anomalies also influence the recall rate. Experiments showed that for several threshold settings, the recall rate for trackbed anomalies with bounding boxes covering more than 1.27% of the input image, which equals 10,000 pixels or roughly 20,000 mm^2, exceeds 80% while maintaining an accuracy of more than 95%. Dead animals and cans are detected more reliably than vegetation and anomalies annotated as other objects. Most likely, VADAR struggles with vegetation more than with animals or cans because of the sparsity of some vegetation and the low brightness difference to the background. Generally, VADAR is sensitive to brightness differences between an anomaly and the background. While achieving an accuracy of more than 95%, trackbed anomalies of particular interest, like dead animals, were detected with a recall rate of more than 73%, and all objects with a recall rate of more than 51%. False positive rates larger than 5% are considered less interesting for this specific application because it would require too much human effort to filter out the high number of false positives. VADAR detects anomalies on the trackbed with a bounding box size larger than 5,000 pixels, corresponding to the size of roughly 10,000 mm^2, with a recall rate of more than 51% while achieving a false positive rate of 4.8%. The main limiting factor for the size of trackbed anomalies is the size of the ballast stones.

Similar to trackbed anomaly detection, threshold parameters can be used to define a tradeoff between the recall rate of rail anomalies and the false positive rate. By varying these parameters, the rail anomaly detection achieved accuracies between roughly 96% and 99.5% and recall rates for annotated rail damages of up to 83%. While achieving a recall rate of more than 53% for rail damages larger than 50 pixels, VADAR maintained a false positive rate of 4.2%. In contrast to trackbed anomalies like foreign objects and vegetation, there is only a small number of annotated rail damages in the labeled section of the *Kombi* dataset. Some of these real damages were manually cropped to be automatically changed in size and orientation and replicated on different positions in healthy images. This procedure allows the generation of thousands of synthetic images containing rail damages of different types, sizes, orientations, and positions. This enabled a more detailed analysis of the impact specific characteristics of rail damages have upon the recall rate. While larger damages are clearly associated with higher recall rates, damages on brighter sections of rails are also correlated with higher recall rates. In contrast to trackbed anomaly detection, the minimum size of rail damages VADAR can detect is mainly limited by the highest tolerable false positive rate and the camera's resolution.

A comparison of VADAR with a state-of-the-art anomaly detection system for railway inspection shows that VADAR achieves better results in every analyzed performance metric. While maintaining a 10pp higher accuracy VADAR simultaneously detects every investigated class of anomaly with a higher recall rate than the state-of-the-art algorithm.

6.1 Outlook

In this thesis, VADAR was only applied to the *Kombi* dataset, which consists of gray-scale images taken from a birds-eye view perspective. Introducing an RGB instead of a monochrome camera could increase performance. The additional color information could be exploited to increase the recall rates of anomalies further. Most likely, this would improve the detection of objects with different colors than the ballast. This could be especially interesting for certain vegetation types because their RGB representation should be dominated by the green channel, which is unusual for ballast or other infrastructure elements on the trackbed. Taking control over lighting conditions by introducing different lighting systems could improve performance. Within the *Kombi* dataset sequences where the train rides through a tunnel are very dark. Because of this, most details are not visible, and only small sections of the images are bright enough to allow anomalies to be detected. Therefore, additional lamps, different lighting systems, or an infrared camera could enable the detection of anomalies in tunnels or during the night.

The proposed approach could also be adapted to work with images taken from a different perspective. Instead of using birds-eye view images, a camera could record images from a front-view perspective of the train. This would allow the images to cover more space around the track and enable the detection of unnoticed anomalies along a rail track, like heavy vegetation leaning into the structure gauge of a track or signs of vandalism like graffiti. This front-view perspective would also have the side-effect of different lighting conditions. The whole trackbed, including the areas on both sides of the rails, would typically have similar brightness levels and should not be overexposed like in many instances of the *Kombi* dataset. Furthermore, getting permission from official authorities to mount a camera system within a train instead of installing it on the outside of a train is less complicated.

While VADAR detects anomalies, a remote analyst still needs to look at the potentially anomalous images and decide if the anomaly is interesting for maintenance. Although the described experiments aimed at high accuracy and a low number of false positives, further reducing the number of false positives by an additional algorithm could be beneficial. Because this algorithm would only analyze the potentially anomalous images reported by VADAR, a remote analyst could use more powerful computational units. This would enable using algorithms like iterative reconstruction-based Generative Adversarial Network (GAN) methods that might further reduce the number of false positives. Furthermore, an analysis of the latent space representation of an AE or a DAE could be used to cluster certain types of anomalies.

Bibliography

[1] European Rail Supply Industry Association. “Establishing rail as the backbone of future mobility”. In: 24.5 (2018).

[2] *Fresh view on railways*. https://www.wko.at/service/aussenwirtschaft/fresh-view.166-railways.pdf. Accessed: 2023-05-04.

[3] *Draft proposal for a European Partnership under Horizon Europe Transforming Europe’s Rail System*. https://rail-research.europa.eu/wp-content/uploads/2020/07/20200705_Partnership_High-Level-Paper.pdf. Accessed: 2023-05-17.

[4] *Rail Infrastructure in Africa Financing Policy Options*. https://www.afdb.org/fileadmin/uploads/afdb/Documents/Events/ATFforum/Rail_Infrastructure_in_Africa_-_Financing_Policy_Options_-_AfDB.pdf. Accessed: 2023-05-17.

[5] *Importance of Railway Intelligent Transportation Systems and Architectural Design Issues of Indian Railway Network Scenario*. Vol. 2011 Joint Rail Conference. ASME/IEEE Joint Rail Conference. Mar. 2011, pp. 433–441.

[6] Hua Hu et al. “Effect of integrated multi-modal transit information on modal shift”. In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2010, pp. 1753–1757.

[7] Julian Torres de Miranda Pinto et al. “Road-rail intermodal freight transport as a strategy for climate change mitigation”. In: *Environmental development* 25 (2018), pp. 100–110.

[8] Mats Andersson. *Marginal cost of railway infrastructure wear and tear for freight and passenger trains in Sweden*. 2010.

[9] Michael A Rossetti. “Potential impacts of climate change on railroads”. In: *The Potential Impacts of Climate Change on Transportation: Workshop Summary*. 2002.

[10] Eralda Nishani and Betim Çiço. “Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation”. In: *2017 6th Mediterranean Conference on Embedded Computing (MECO)*. 2017, pp. 1–4.

[11] Kirtan B. Patel. “A Review: Machine vision and its Applications.” In: *IOSR Journal of Electronics and Communication Engineering* 7 (2013), pp. 72–77.

[12] Abdullah Ayub Khan et al. “Machine Learning in Computer Vision: A Review”. In: *ICST Transactions on Scalable Information Systems* (Apr. 2021).

[13] Andre Esteva et al. “Deep learning-enabled medical computer vision”. In: *npj Digital Medicine* 4.1 (Jan. 2021), p. 5.

[14] Bassem Zohdy et al. “Machine Vision Application on Science and Industry: Machine Vision Trends”. In: (Jan. 2019), pp. 233–254.

[15] Ansam Abdulhussein, Hasanien Kariem, and Alaa Alanssari. “Computer Vision to Improve Security Surveillance through the Identification of Digital Patterns”. In: May 2020, pp. 1–5.

[16] Lukas Ruff et al. “A Unifying Review of Deep and Shallow Anomaly Detection”. In: *CoRR* abs/2009.11732 (2020).

[17] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly Detection: A Survey”. In: *ACM Comput. Surv.* 41.3 (July 2009).

[18] STATISTIK AUSTRIA. “Verkehrsstatistik 2020”. In: (2020).

[19] STATISTIK AUSTRIA. “Verkehrsstatistik 2019”. In: (2019).

[20] STATISTIK AUSTRIA. “Verkehrsstatistik 2018”. In: (2018).

[21] Transport and Housing Bureau. *Public Transport Strategy Study.* Tech. rep. Government of Hong Kong, 2017.

[22] M Nicolas J Firzli and J Nicola. “Transportation infrastructure and country attractiveness”. In: *Revue Analyse Financière* 48 (2013), Q2.

[23] Tomas Lidén. “Railway Infrastructure Maintenance - A Survey of Planning Problems and Conducted Research”. In: *Transportation Research Procedia* 10 (2015). 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands, pp. 574–583.

[24] EIM-EFRTC-CER Working Group on Market Strategies for Track Maintenance & Renewal. “Report from the EIM-EFRTC-CER Working Group on Market Strategies for Track Maintenance & Renewal”. In: (2012).

[25] Zdenka Popović et al. “Rail inspection of RCF defects”. In: *Metalurgija -Sisak then Zagreb-* 52 (Oct. 2013), pp. 537–540.

[26] Andrew Dow. *The railway: British track since 1804.* Wharncliffe, 2014.

[27] Guido HANSPACH. “UST 02: Schienenprüfzug der neuen Generation für die europäischen Bahnen”. In: *Der Eisenbahningenieur (Hamburg)* 57.1 (2006), pp. 26–28.

[28] Haoyu Wang et al. “Study of loaded versus unloaded measurements in railway track inspection”. In: *Measurement* 169 (2021), p. 108556.

[29] Pranav Lad and Mansi Pawar. “Evolution of railway track crack detection system”. In: *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA).* 2016, pp. 1–6.

[30] Zdenka Popović, L. Puzavac, and Luka Lazarević. “Rail defects due to rolling contact fatigue”. In: 54 (Apr. 2016), pp. 17–29.

[31] Maria Molodova et al. “Automatic Detection of Squats in Railway Infrastructure”. In: *IEEE Transactions on Intelligent Transportation Systems* (Oct. 2014).

[32] Madalina Ciotlaus et al. “Rail-wheel Interaction and Its Influence on Rail and Wheels Wear”. In: *Procedia Manufacturing* 32 (Jan. 2019), pp. 895–900.

[33] Xavier Gibert, Vishal M. Patel, and Rama Chellappa. “Deep Multitask Learning for Railway Track Inspection”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.1 (2017), pp. 153–164.

[34] Roger Nyberg et al. “Monitoring Vegetation on Railway Embankments : Supporting Maintenance Decisions”. In: June 2013.

[35] Riccardo Gasparini et al. “Anomaly Detection for Vision-Based Railway Inspection”. In: *EDCC Workshops.* 2020.

[36] Jie Yang et al. “Visual Anomaly Detection for Images: A Survey”. In: *CoRR* abs/2109.13157 (2021).

[37] Ricardo Silva et al. “Machine Vision Systems for Industrial Quality Control Inspections: 15th IFIP WG 5.1 International Conference, PLM 2018, Turin, Italy, July 2-4, 2018, Proceedings”. In: July 2018, pp. 631–641.

[38] Vincent Wilmet et al. “A Comparison of Supervised and Unsupervised Deep Learning Methods for Anomaly Detection in Images”. In: *CoRR* abs/2107.09204 (2021).

[39] Tri Cao, Jiawen Zhu, and Guansong Pang. *Anomaly Detection under Distribution Shift.* 2023.

[40] Zhaomin Chen et al. "Autoencoder-based network anomaly detection". In: *2018 Wireless Telecommunications Symposium (WTS).* 2018, pp. 1–5.

[41] Laurenz Strothmann, Uwe Rascher, and Ribana Roscher. "Detection of Anomalous Grapevine Berries Using All-Convolutional Autoencoders". In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium.* 2019, pp. 3701–3704.

[42] Yunpeng Jiang et al. "Research on the Flight Anomaly Detection During Take-off Phase Based on FOQA Data". In: *2019 CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS).* 2019, pp. 756–760.

[43] Tharindu Fernando et al. "Deep Learning for Medical Anomaly Detection - A Survey". In: *CoRR* abs/2012.02364 (2020).

[44] Sonali B. Wankhede. "Anomaly Detection using Machine Learning Techniques". In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT).* 2019, pp. 1–3.

[45] Saeed Khalilian et al. "PCB Defect Detection Using Denoising Convolutional Autoencoders". In: *2020 International Conference on Machine Vision and Image Processing (MVIP).* 2020, pp. 1–5.

[46] *PyTorch: Installing previous versions of PyTorch.* `https://pytorch.org/get-started/previous-versions/`. Accessed: 2022-11-11.

[47] *An open source machine learning framework.* `https://pytorch.org/`. Accessed: 2022-11-11.

[48] Guansong Pang et al. "Deep Learning for Anomaly Detection: A Review". In: *CoRR* abs/2007.02500 (2020).

[49] Guangxin Lou and Hongzhen Shi. "Face image recognition based on convolutional neural network". In: *China Communications* 17.2 (2020), pp. 117–124.

[50] Divya Arora, Mehak Garg, and Megha Gupta. "Diving deep in Deep Convolutional Neural Network". In: *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN).* 2020, pp. 749–751.

[51] S Kaushik, Abhishek Raman, and K.V.S Rajeswara Rao. "Leveraging Computer Vision for Emergency Vehicle Detection-Implementation and Analysis". In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT).* 2020, pp. 1–6.

[52] Eralda Nishani and Betim Çiço. "Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation". In: *2017 6th Mediterranean Conference on Embedded Computing (MECO).* 2017, pp. 1–4.

[53] KC Santosh, Nibaran Das, and Swarnendu Ghosh. "Chapter 2 - Deep learning: a review". In: *Deep Learning Models for Medical Imaging.* Ed. by KC Santosh, Nibaran Das, and Swarnendu Ghosh. Primers in Biomedical Imaging Devices and Systems. Academic Press, 2022, pp. 29–63.

[54] M.A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015.

[55] S Agatonovic-Kustrin and R Beresford. "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research". In: *Journal of Pharmaceutical and Biomedical Analysis* 22.5 (2000), pp. 717–727.

[56] Wojciech Samek et al. "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications". In: *Proceedings of the IEEE* 109.3 (2021), pp. 247–278.

[57] John Pomerat, Aviv Segev, and Rituparna Datta. "On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 6183–6185.

[58] Clara Catanese et al. "A Survey of Neural Network Applications in Fiber Nonlinearity Mitigation". In: *2019 21st International Conference on Transparent Optical Networks (ICTON)*. 2019, pp. 1–4.

[59] David Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (Nov. 2004), pp. 91–.

[60] John Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698.

[61] L. Seed et al. "A generalised convolver for computer vision". In: *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No.97CB36136)*. Vol. 2. 1997, 1522–1526 vol.2.

[62] Nilakshi Devi and Bhogeswar Borah. "Cascaded pooling for Convolutional Neural Networks". In: *2018 Fourteenth International Conference on Information Processing (ICINPRO)*. 2018, pp. 1–5.

[63] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015).

[64] Mohit Pandey et al. "The transformational role of GPU computing and deep learning in drug discovery". In: *Nature Machine Intelligence* 4 (Mar. 2022), pp. 211–221.

[65] Stephen Notley and Malik Magdon-Ismail. "Examining the Use of Neural Networks for Feature Extraction: A Comparative Analysis using Deep Learning, Support Vector Machines, and K-Nearest Neighbor Classifiers". In: *CoRR* abs/1805.02294 (2018).

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012.

[67] Liron Bergman and Yedid Hoshen. "Classification-Based Anomaly Detection for General Data". In: *CoRR* abs/2005.02359 (2020).

[68] Felix Leeb, Stefan Bauer, and Bernhard Schölkopf. "Interventional Assays for the Latent Space of Autoencoders". In: *CoRR* abs/2106.16091 (2021).

[69] Jack Klys, Jake Snell, and Richard S. Zemel. "Learning Latent Subspaces in Variational Autoencoders". In: *CoRR* abs/1812.06190 (2018).

[70] Jasem Almotiri, Khaled Elleithy, and Abdelrahman Elleithy. "Comparison of autoencoder and Principal Component Analysis followed by neural network for e-learning using handwritten recognition". In: *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. 2017, pp. 1–5.

[71] Jia Shijie et al. "Research on data augmentation for image classification based on convolution neural networks". In: *2017 Chinese Automation Congress (CAC)*. 2017, pp. 4165–4170.

[72] Wang Jinyeong and Sanghwan Lee. "Data Augmentation Methods Applying Grayscale Images for Convolutional Neural Networks in Machine Vision". In: *Applied Sciences* 11 (July 2021), p. 6721.

[73] Timofey A. Korzhebin and Alexey D. Egorov. "Comparison of Combinations of Data Augmentation Methods and Transfer Learning Strategies in Image Classification Used in Convolution Deep Neural Networks". In: *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*. 2021, pp. 479–482.

[74] Kaiming He et al. *Masked Autoencoders Are Scalable Vision Learners*. 2021.

[75] Karel J. Zuiderveld. "Contrast Limited Adaptive Histogram Equalization". In: *Graphics gems*. 1994.

[76] Andrew Y. Ng. "Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78.

[77] Akshay Badola, Vineet Padmanabhan Nair, and Rajendra Prasad Lal. "An Analysis of Regularization Methods in Deep Neural Networks". In: *2020 IEEE 17th India Council International Conference (INDICON)*. 2020, pp. 1–6.

[78] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. "DropBlock: A regularization method for convolutional networks". In: *CoRR* abs/1810.12890 (2018).

[79] Ruoyu Sun. "Optimization for deep learning: theory and algorithms". In: *CoRR* abs/1912.08957 (2019).

[80] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.04747 (2016).

[81] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[82] JooSeuk Kim and Clayton D. Scott. "Robust Kernel Density Estimation". In: *Journal of Machine Learning Research* 13.82 (2012), pp. 2529–2565.

[83] Feng Gao et al. "ConNet: Deep Semi-Supervised Anomaly Detection Based on Sparse Positive Samples". In: *IEEE Access* 9 (2021), pp. 67249–67258.

[84] Xue Ying. "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168 (Feb. 2019), p. 022022.

[85] Srikanth Thudumu et al. "A comprehensive survey of anomaly detection techniques for high dimensional big data". In: *Journal of Big Data* 7 (July 2020).

[86] Yiteng Zhai, Yew-Soon Ong, and Ivor W. Tsang. "The Emerging "Big Dimensionality"". In: *IEEE Computational Intelligence Magazine* 9.3 (2014), pp. 14–26.

[87] Furkan Ulger, Seniha Esen Yuksel, and Atila Yilmaz. "Anomaly Detection for Solder Joints Using beta-VAE". In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 11.12 (2021), pp. 2214–2221.

[88] Ahad Alloqmani et al. "Deep Learning based Anomaly Detection in Images: Insights, Challenges and Recommendations". In: *International Journal of Advanced Computer Science and Applications* 12 (Jan. 2021).

[89] Xavier Gibert, Vishal M. Patel, and Rama Chellappa. "Deep Multitask Learning for Railway Track Inspection". In: *IEEE Transactions on Intelligent Transportation Systems* 18.1 (2017), pp. 153–164.

[90] Marco A.F. Pimentel et al. "A review of novelty detection". In: *Signal Processing* 99 (2014), pp. 215–249.

[91] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41 (July 2009).

[92] Elnaz Bigdeli et al. "A fast noise resilient anomaly detection using GMM-based collective labelling". In: *2015 Science and Information Conference (SAI)*. 2015, pp. 337–344.

[93] Wenbo Liu et al. "Outlier Detection Algorithm Based on Gaussian Mixture Model". In: *2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*. 2019, pp. 488–492.

[94] Stefania Matteoli et al. "Background Density Nonparametric Estimation With Data-Adaptive Bandwidths for the Detection of Anomalies in Multi-Hyperspectral Imagery". In: *IEEE Geoscience and Remote Sensing Letters* 11.1 (2014), pp. 163–167.

[95] Kaitai Zhang, Bin Wang, and C.-C. Jay Kuo. "PEDENet: Image Anomaly Localization via Patch Embedding and Density Estimation". In: *CoRR* abs/2110.15525 (2021).

[96] Monica Casella et al. “Autoencoders as an alternative approach to principal component analysis for dimensionality reduction. An application on simulated data from psychometric models”. In: *Symposium on Psychology-Based Technologies*. 2021.

[97] I. K. Savvas et al. “Increasing the Quality and Performance of N-Dimensional Point Anomaly Detection in Traffic Using PCA and DBSCAN”. In: *2018 26th Telecommunications Forum (TELFOR)*. 2018, pp. 1–4.

[98] Fanwu Chu. “An improved PCA algorithm for anomaly detection of hydropower units”. In: *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)* (2017), pp. 494–498.

[99] Christian Callegari et al. “A Novel PCA-Based Network Anomaly Detection”. In: *2011 IEEE International Conference on Communications (ICC)*. 2011, pp. 1–5.

[100] Zhengxue Cheng et al. “Deep Convolutional AutoEncoder-based Lossy Image Compression”. In: *CoRR* abs/1804.09535 (2018).

[101] Faisal Nadeem Khan and Alan Pak Tao Lau. “Robust and efficient data transmission over noisy communication channels using stacked and denoising autoencoders”. In: *China Communications* 16.8 (2019), pp. 72–82.

[102] Sanyapong Youkachen et al. “Defect Segmentation of Hot-rolled Steel Strip Surface by using Convolutional Auto-Encoder and Conventional Image processing”. In: *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*. 2019, pp. 1–5.

[103] Antanas Kascenas, Nicolas Pugeault, and Alison Q O’Neil. “Denoising Autoencoders for Unsupervised Anomaly Detection in Brain MRI”. In: *Medical Imaging with Deep Learning*. 2022.

[104] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013.

[105] Min Su Kim et al. “Unsupervised Anomaly detection of LM Guide Using Variational Autoencoder”. In: *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. 2019, pp. 1–5.

[106] Nuno Ferreira and Margarida Silveira. “Ship Detection in SAR Images Using Convolutional Variational Autoencoders”. In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. 2020, pp. 2503–2506.

[107] Xuan Xia et al. “GAN-based anomaly detection: A review”. In: *Neurocomputing* 493 (2022), pp. 497–535.

[108] Jie Gui et al. “A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications”. In: *CoRR* abs/2001.06937 (2020).

[109] Marco Domenico Cirillo, David Abramian, and Anders Eklund. “Vox2Vox: 3D-GAN for Brain Tumour Segmentation”. In: *CoRR* abs/2003.13653 (2020).

[110] Alankrita Aggarwal, Mamta Mittal, and Gopi Battineni. “Generative adversarial network: An overview of theory and applications”. In: *International Journal of Information Management Data Insights* 1.1 (2021), p. 100004.

[111] Thomas Schlegl et al. “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery”. In: *Information Processing in Medical Imaging*. Ed. by Marc Niethammer et al. Cham: Springer International Publishing, 2017, pp. 146–157.

[112] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: (Nov. 2015).

[113] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *CoRR* abs/1812.04948 (2018).

[114] Alvaro Figueira and Bruno Vaz. “Survey on Synthetic Data Generation, Evaluation Methods and GANs”. In: *Mathematics* 10.15 (2022).

[115] Dominik Narnhofer et al. “Inverse GANs for accelerated MRI reconstruction”. English. In: *Wavelets and Sparsity XVIII*. SPIE Optics+Photonics 2019 ; Conference date: 23-08-2019 Through 27-08-2019. 2019.

[116] Lucas Deecke et al. “Image Anomaly Detection with Generative Adversarial Networks”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michele Berlingerio et al. Cham: Springer International Publishing, 2019, pp. 3–17.

[117] Çağlar Aytekin et al. “Railway Fastener Inspection by Real-Time Machine Vision”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.7 (2015), pp. 1101–1107.

[118] Hao-ran Li et al. “Dynamic Electromagnetic Thermography System for Rail Inspection”. In: *2021 IEEE Far East NDT New Technology And Application Forum (FENDT)*. 2021, pp. 99–103.

[119] Xinyu Du, Yu Cheng, and Zichen Gu. “Change Detection: The Framework of Visual Inspection System for Railway Plug Defects”. In: *IEEE Access* 8 (2020), pp. 152161–152172.

[120] Daniel Olid, José M. Fácil, and Javier Civera. “Single-View Place Recognition under Seasonal Changes”. In: *CoRR* abs/1808.06516 (2018).

[121] Oliver Zendel et al. “Railsem19: A dataset for semantic rail scene understanding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[122] *Kaggle Railway Track Fault Detection*. `https://www.kaggle.com/datasets/salmaneunus/railway-track-fault-detection`. Accessed: 2022-10-19.

[123] *Mission Embedded*. `https://mission-embedded.com/`. Accessed: 2023-05-17.

[124] Junbo Liu et al. “Learning Visual Similarity for Inspecting Defective Railway Fasteners”. In: *IEEE Sensors Journal* 19.16 (2019), pp. 6844–6857.

[125] Roger Nyberg et al. “Monitoring Vegetation on Railway Embankments : Supporting Maintenance Decisions”. In: June 2013.

[126] Shubin Zheng et al. “Railway track gauge inspection method based on computer vision”. In: *2012 IEEE International Conference on Mechatronics and Automation*. 2012, pp. 1292–1296.

[127] Riccardo Gasparini et al. “Anomaly Detection for Vision-Based Railway Inspection”. In: *Dependable Computing - EDCC 2020 Workshops*. Ed. by Simona Bernardi et al. Cham: Springer International Publishing, 2020, pp. 56–67.

[128] T Hoppe, G Matschke, and R Müller. “Homologation of Trans-European Rolling Stock: An Integrated Approach”. In: *Final Proceedings of the 7th World Congress on Railway Research (WCRR)*. 2006, pp. 4–8.

[129] *Kontinuierliches On-board Monitoring der Bahn Infrastruktur –Technische und ökonomische Analyse*. `https://projekte.ffg.at/projekt/3019460`. Accessed: 2023-05-17.

[130] SL Grassie. “Rail corrugation: characteristics, causes, and treatments”. In: *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit* 223.6 (2009), pp. 581–596.

[131] *NVIDA Jetson*. `https://www.nvidia.com/de-de/autonomous-machines/embedded-systems`. Accessed: 2023-05-03.

[132] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605.

[133] *MSELoss*. `https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html`. Accessed: 2023-05-04.

[134] *pytorch-msssim*. `https://github.com/jorge-pessoa/pytorch-msssim`. Accessed: 2023-05-04.

[135] Yasi Wang, Hongxun Yao, and Sicheng Zhao. “Auto-Encoder Based Dimensionality Reduction”. In: *Neurocomputing* 184 (Nov. 2015).

[136] Qinkun Xiao and Yang Si. “Human action recognition using autoencoder”. In: *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. 2017, pp. 1672–1675.

[137] Jimin Tan et al. “A critical look at the current train/test split in machine learning”. In: *CoRR* abs/2106.04525 (2021).

[138] *NVIDA A100 Tensor-Core-GPU.* `https://www.nvidia.com/de-de/data-center/a100/`. Accessed: 2022-11-11.