Anomaly Detection on the Rail Lines Using Semantic Segmentation and Self-supervised Learning

Kanwal Jahan, Jeethesh Pai Umesh and Michael Roth

Abstract—**This paper introduces a novel application of anomaly detection on the rail lines using deep learning methods on camera data. We propose a two-fold approach for identifying irregularities like coal, dirt, and obstacles on the rail tracks. In the first stage, a binary semantic segmentation is performed to extract only the rails from the background. In the second stage, we deploy our proposed autoencoder utilizing the self-supervised learning techniques to address the unavailability of labelled anomalies. The extracted rails from stage one are divided into multiple patches and are fed to the autoencoder, which is trained to reconstruct the non-anomalous data only. Hence, during the inference, the regeneration of images with any abnormalities produces a larger reconstruction error. Applying a predefined threshold to the reconstruction errors can detect an anomaly on a rail track. Stage one, rail extracting network achieves a high value of 52.78% mean Intersection over Union (mIoU). The second stage autoencoder network converges well on the training data. Finally, we evaluate our two-fold approach on real scenario test images, no false positives or false negatives were found in the the detected anomalies on the rail tracks.**

Index Terms—**Semantic segmentation, self-supervised learning, autoencoders, deep learning, mIoU, UNet, anomaly detection, feature extraction, railway environment**

I. Introduction

Camera data is a rich source of information for performing the tasks like semantic segmentation and object detection in a given environment. The recent advances in modern hardware and GPU technology have enabled the adoption of deep learning methods on camera data for knowledge extraction and automated decision-making. The neural networks can be optimally tuned, are accurate and robust against environmental factors like illumination and weather conditions. On the other hand, traditional image-processing methods have lower accuracy for tasks like semantic segmentation, object detection, and classification, which are more susceptible to the slightest changes in environmental conditions. We use image data from two industrial cameras installed on a locomotive running in one of the harbors in Germany to perform anomaly detection on the rail tracks. In this paper, we propose a deep learning-based approach to process the collected image data for detecting dirt, stones, tree leaves, coal, and any other obstacles which are the unwanted objects found on the rail tracks. Knowledge about the presence of such objects is critical for the smooth operation of the railways. Especially, the port environment, where locomotives operate at slow speed of 20 km/h, has the presence of irregularities like coal, debris, and dirt, such rail lines if not cleaned timely can incur huge maintenance costs. There is a similar need for high-speed railway operation too, to detect any obstacles on the rail lines and avoid any unnecessary delays, accidents, or large-scale damage to the infrastructure. Obstacle detection is an important component for making informed decisions on the run time to achieve the goal of autonomous driving too.

Deep learning-based approaches rely mostly on quality labelled datasets, which are unfortunately very few in the railway domain. Generally, the process of labelling image data is tedious and time-consuming. The only dataset which is publicly available for the railway environment is known as RailSem19 [1]. It has two schemes of labelling, semantic labels (pixel-wise labelling of an image) and 2D bounding boxes. We find the semantic labels from RailSem19 [1] are useful for extracting the rails from the background. In some cases, getting the labelled data is difficult or impossible as the object can be of any shape, size and may appear rarely. Our task of anomaly detection deals with finding abnormalities, inconsistencies, and deviations from the standard behavior. Since such cases are rarely occurring, it is not only difficult to document and annotate them but also time-consuming to gather enough instances.

Our main contribution is, we introduce a novel way of using a self-supervised learning method to reduce the dependency on labelled data. We propose a method to automate the detection of anomalies or any unwanted foreign objects on the rail tracks, which can cause potential delays or accidents in railway operations. The high value (52.78%) of mean Intersection over Union (mIoU) for rail segmentation, an accurate and robust anomaly detection and speed of inference (27 fps) allow us to deploy our method for real-time anomaly detection in not only port areas, with a reduced vehicle speed operation, but also for high-speed rail vehicles. The uniqueness of the our work lies in combining the semantic segmentation and autoencoder trained in a self-supervised manner to detect the anomalies of any kind or shape on the rail lines.

The paper is structured in the following way. We describe prior developments of the methods used in this paper in section II followed by a detailed description of our proposed method and contribution in Section III. The training procedures, metrics, and training results are shown in Section IV and lastly, we

[1]Kanwal Jahan, Jeethesh Pai Umesh and Michael Roth are with the Institute of Transportation Systems, German Aerospace Center (DLR), 38108 Braunschweig, Germany `firstname.lastname@dlr.de` except m.roth@dlr.de

[2]1st and 2nd both authors contributed equally.

conclude and discuss the possible future research in Section V.

II. Related Work

Semantically segmenting the image is necessary to extract rail coordinates, as that is the only area of interest to detect the presence of anomalies in our application. Initial approaches of image segmentation were performed using traditional binary threshold, K-Means clustering, and Graph-based segmentation methods. But these approaches often lacked the semantic meaning in the labels. Since then, researchers have started to focus on neural network-based segmentation methods. The problem with these networks is that it requires costly pixel-wise labelling of image data, which were not initially available until a few years ago. With the formation of datasets like KITTI [2], PASCAL VOC [3], Cityscapes [4] there has been advancement in this genre of computer vision, specifically for automotive sector. Early works of semantic segmentation using neural networks date back to SegNet [5] which is based on encoder and decoder network inspired from VGG-Net [6]. Later works like U-Net [7] showed that using skip connections of earlier feature maps with the upstream feature maps can help to retain the quality of segmentation while upsampling. A similar approach is used in FRRN (Full Resolution Residual Networks) [8] in which residual streams carry the information about the upstream layers and a pooling stream extracts the features sequentially. The FRRN network is inspired by the design of ResNet [9] which also has a similar residual carryover of intermediate feature maps. The challenge related to such network architecture is that it is computationally expensive and has large memory requirements while training. Therefore, it requires efficient algorithms to calculate gradients and store them on GPU even for low batch sizes to avoid out-of-memory errors (OOM). Despite its structure and computational cost, FRRN performs very well on Cityscapes. Other networks like FCN (Fully Convolutional Networks) [10] as well DeepLab [11] are also popular for the semantic segmentation tasks. A model like U-Net extracts features from the given image with help of Convolutions and max-pooling layers. During each pooling, a copy of the feature map generated is stored and concatenated with upsampling layers. This concatenation of old feature maps preserves spatial features or edges when compared to vanilla upsampling of pooled layers. Pooled layers are responsible for extracting features that are necessary for semantic segmentation tasks. However, these networks have been mostly used for the automotive sector, medical diagnosis and credit and fraud detection.

The availability of datasets is abundant in the case of road scenes, but there was not much research in the case of railway sector until the RailSem19 dataset [1], introduced in 2019, which gives a weakly supervised annotation of semantic segmentation in the railway domain using networks trained on road scenes. The scenes are mainly taken with a camera in ego perspective view and are ideal for our application.

Anomaly detection has been researched upon for a few decades in different domains. A short survey by Chandola et al. [12] discusses anomaly detection using various techniques in real-world situations. It also covers several applications like an anomaly in OCR (Optical Character Recognition) and gives a brief introduction of types of anomalies that can be detected. A theoretical framework by Fei Ye et al. [13] presents the concepts related to anomaly detection using deep encoder-decoder networks on huge datasets like ImageNet [14]. Earlier, anomaly detection tasks were dealt with, with the help of clustering, nearest neighbor classification, support vector machines, and Bayesian networks. But owing to the changes in illumination, noise, and changing environments of our application, such conventional techniques fail to capture the nuances of the anomalies correctly and therefore produce unstable results. Therefore, we look into methods that are robust against such environmental factors as well as illumination changes. To distinguish an anomaly from a given image, it is necessary to find out features of some specific regions of the image, which eventually help us to classify if the region appears normal or anomalous.

Extraction of features from the image is an evolving topic for decades. Some conventional methods used in image feature extraction are given in a survey of feature extraction methods by G.Kumar et al. [15] which mainly comprise of properties of gray level co-occurrence matrix-like energy, Gabor filters [16] and Haar wavelets [17]. However, due to changing environments like lighting, noise, and weather conditions it is difficult to devise a robust method using hand-crafted features which can reliably extract features irrespective of the environment. However, in the works of R. Roslan et al. [18], an interesting application using the Gabor filter shows that conventional methods are reliable only to some extent. Also based on the fact that the first stages of convolutional filters of convolutional neural networks (CNN) imitate Gabor filters, we analyze some deep learning-related feature extraction procedures which are robust in variable lighting conditions and noise.

Since we have a lack of labelled datasets for anomalies, we rely on self-supervised learning algorithms to perform feature extraction. One of the methods, related to self-supervised learning especially in $2D$ images, dates back to the work of Rumelhart et al. in 1986 [19] which introduces autoencoder as a neural network that learns to encode and decode the input fed to it. Autoencoders further paved a new way for self-supervised machine learning, as the constraint of costly labelling is removed. In cases of self-supervised learning, there is a large database without labels that can be used for training a feature extractor. In the paper by Dor Bank et al. [20] we can see multiple applications of an autoencoder. It also discusses different types of autoencoders by the nature of their structure. Autoencoders can be used in fields like clustering [21] [22], recommendation systems [23], classification [24] and also anomaly detection [25] [26] where autoencoders are trained on normal data and are used to detect abnormal data.

III. Proposed Method

Our anomaly detection framework is based on the analysis of image data collected from RGB cameras installed on an operational locomotive, capturing ego-motion, in one of the ports of Germany. On the collected video frames, a multi-stage image processing approach, as shown in $Algorithm.$ 1 is applied to detect any inconsistencies and anomalous objects on the rails.

Algorithm 1 Algorithm for anomaly detection on the rail track

```
Require: 3-Channel image with rail tracks
 1: Extract rails using semantic segmentation
 2: if Rail lines found then
 3:     Perform pixel clustering using DBscan
 4:     Associate estimated clusters using Euclidean distance
        to form the individual rail lines
 5:     Divide each rail line into patches of $16 \times 16$ pixels
 6:     Use trained autoencoder to reconstruct the patches
 7:     Apply threshold on the reconstruction errors of the
        patches
 8:     Declare an anomalous object on path with reconstruc-
        tion error above defined threshold
 9: else
10:     Print  Image does not contain any rails
```

At the first stage, semantic segmentation is performed to extract all the rail lines from the given image, as there can be multiple parallel railway tracks. Next, clustering is performed to link the pixels belonging to one rail together and to distinguish rail lines from each other. In the final step, an autoencoder fitting to our requirements is designed to predict the occurrence of anomalies on the extracted rails.

A. *Semantic segmentation*

For the extraction of rail tracks, we use U-Net [7] to semantically segment the images into a rail and non-rail object. The network has a contracting part and an expansive part, which gives it a U-shaped structure. The architecture introduces copy and crop connections that concatenate each up-sampling convolution layer with the respective downsampling convolution layer. For example, the first downsampling layer is concatenated with the last upsampling layer and similarly the second downsampling layer with the second last upsampling layer. The contracting part consists of repeated convolution layers followed by ReLU and max-pooling layers. To further improve the performance of U-Net, we introduce VGG-Net [6] and ResNet-50 [9] as the backbone network, as suggested in [27]. We further increase the accuracy with the help of transfer learning [28]. Instead of initializing the learning weights to zero or random normal, they are initialized by pre-trained weights of VGG-Net or ResNet-50 with ImageNet [14]. As our deployed U-Net and VGG-Net share the architecture we use the weights of the encoding layers only.

The number of pixels in the image belonging to rail lines is a small fraction as compared to the rest of the image. Due to the presence of a huge class imbalance, we use and compare the losses such as weighted binary cross-entropy loss ($\mathcal{L}_{wbce}$) and focal loss [29] ($\mathcal{L}_{focal}$) to train our network. Weighted binary cross-entropy loss, as well as focal loss, penalizes the network more for loss contribution from foreground pixels, rail pixels, than from the background. The weighted binary cross-entropy loss as described in Eq.1 introduces the penalty terms as weights assigned to foreground and background loss contributions. The class imbalance is taken care of by increasing the foreground class weight.

$$\begin{aligned}\mathcal{L}_{wbce} = -\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} w_1 I_{xy} log(P_{xy}) + \\ w_0(1-I_{xy})log(1-P_{xy})\end{aligned} \tag{1}$$

Where, P_{xy} = predicted probability of the pixel to be target label (rail pixel), w_0 = normalized weight for background class, w_1 = normalized weight for foreground class W, H = width and height of the image respectively. I_{xy} is target label for training example

$$\begin{aligned}\mathcal{L}_{focal} = -(1-P_{xy})^{\gamma}\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} \alpha I_{xy} log(P_{xy}) \\ +(1-\alpha)(1-I_{xy})log(1-P_{xy})\end{aligned} \tag{2}$$

While Eq. 2, focal loss [29] not only takes care of class-imbalance with the factor α but also takes care of complex instances by penalizing more the hard examples over the simple cases similar to w_1 in Eq. 1. It reduces the loss contribution from easily classified target pixels and focuses on hard examples through the the modulating factor-γ. The trained network performing semantic segmentation generates masks of the original image as predictions, where rail track pixels are assigned a pixel value of 1 and non-rail pixel values with 0.

B. *Pixel Clustering*

The result of semantic segmentation is a binary mask with all the rail tracks marked with a pixel value of 1. The next logical step is to co-relate all the white pixels into separate rail lines. We find a pixel thickness of one to be good enough to form the rail line. Oftentimes, the generated mask will have a rail pixel thickness of 2 or more pixels, which we reduce to only one pixel by taking a mean of them. On the other hand, there can be multiple rail lines in one image whose coordinates need to be distinguished from each other to extract the features robustly.

We associate the pixels based on Euclidean distance. Firstly, we use DBScan clustering algorithm [30] which yields a total number of clusters of white pixels in the segmented image. Each cluster obtained from DBScan represents a part of the rail of the segmented image. Next, we combine those clusters, using Euclidean distance, which are the closest based on the horizontal axis difference of the endpoints of the clusters. Conclusively, we extend the ends of the clusters (rail line) to the region of interest boundary which is ideally the bottom half of the given image, if they are long enough. The extension of the line is done using the slope of the last 20 points of each cluster.

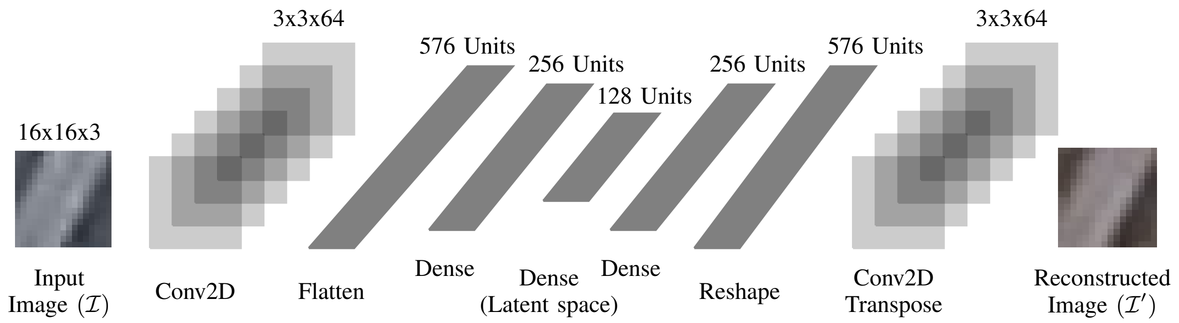


Fig. 1: Autoencoder architecture designed for feature extraction.

C. Anomaly detection

Once the coordinates containing the rail tracks in an image are known and pixels belonging to a single rail are clustered together, the next step is to extract the feature along the rail track on the original image. The rail tracks are distorted by the perspective of the camera with varying width as the camera is capturing the ego-motion of the vehicle. i.e, farther lines appear thinner and closer lines appear thicker. To address this, the rail line is divided along length into the square patches of varying size. For the patches closer to the installed cameras, i.e, part of the rail immediately under the camera, we start with patch of 20×20 pixels and reduce the size by two pixel for every fifth patch along the rail, keeping the aspect ratio ≈ 1. The patch size ranges between maximum size of 20×20 to a minimum size of 14×14 pixels and is re-scaled to a dimension of 16×16 to maintain the uniformity in the feature extraction. We chose the maximum size to be 20 which incorporates the size of rail lines appearing nearest to the camera. The size of the patch is chosen to design an algorithm sensitive to anomalies, on the rails only, which are small in size too. However, the size of the patch can be adjusted to detect anomalies on sleepers as well, utilizing the symmetry in the design of rail tracks.

Mathematically, the center points of one rail surface can be denoted as (i_p, j_p) given that one rail line is divided into p patches. Generally, in our case each rail line surface has $p \in [1, 16]$. The formula for center points of square patch p is given by the equation.

$$C_p = (\frac{l}{2}, pl - \frac{l}{2}) \tag{3}$$

For the used dataset, l starts with 20 and changes after every $4th$ patch and is reduced by 2 pixels. No patch in given p overlaps with the adjacent patch. After converting a rail line into p symmetric patches, having same aspect ratio of rail to background pixel, as shown in Fig. 2, the features of each patch are extracted and analyzed for the presence of an anomaly on the tracks. Whenever an obstacle appears on a rail line, the extracted features of that square patch appear different from the rest of the rail surface. For feature extraction, we use deep autoencoders. This autoencoder takes an input image and tries to reconstruct the same input image. Autoencoders

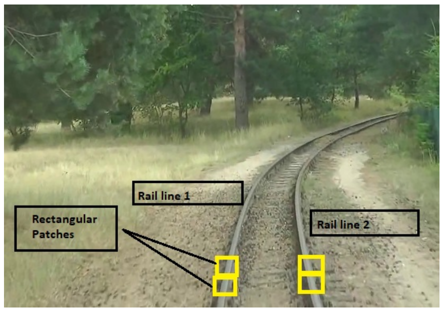


Fig. 2: Extracting features along the rail line.

[31] have an encoder-decoder structure to extract the hidden features of an image. The encoder structure ensures that the image fed in is converted into a compressed and compact form called code. The decoders learn to reconstruct the generated code into the input image again. During the encoding and decoding of the training set, the network learns to extract useful features. The architecture of the autoencoder we have designed designed for our algorithm is given in Fig. 1

The encoder structure consists of a Conv2D layer and a dense layer. The Conv2D (Convolutional layer) has a filter size of 64 and kernel size 3×3. Using the flatten layer the output of the convolutional layer is converted into a $1D$ array which is passed on to ReLU activation and a batch normalization layer. The dense layer consists of 256 units, its output is also passed on to ReLU activation and batch normalization layer resulting in code/latent space. The latent space dimension of the autoencoder is 128. The number and properties of the layers are the same in encoder and decoder except for the order of layers is reversed in decoder, i.e. latent space is fed to a dense layer and then a Conv2D layer to reconstruct the image. Due to the unavailability of any labelled data, the autoencoder is trained only on normal images which do not have any obstacles or dirt covering the rail tracks. To classify a patch as an anomaly or normal rail track, we calculate the reconstruction error $(\mathcal{L}_r)$ obtained from it. The reconstruction error between ground truth image $(\mathcal{I})$ and reconstructed output image $(\mathcal{I}')$ from the autoencoder is given by,

$$\mathcal{L}_r(\mathcal{I}, \mathcal{I}') = \sqrt{(\sum_{x=0}^{W}\sum_{y=0}^{H}(\mathcal{I}(x,y) - \mathcal{I}'(x,y))^2} \tag{4}$$

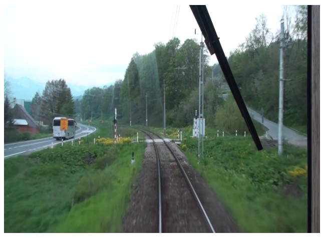
where, $\mathcal{I}$ i.e, ground truth image is same as the input image of the autoencoder. To calculate the threshold, we run the trained auto-encoder on the test dataset which does not have any obstacles or abnormalities, and record the reconstruction error values of all the patches in a list for every image, which we consider as normal range of reconstruction error values. Then we find out minimum and maximum thresholds using the minimum and maximum occurring value in this recorded list. During inference, We treat reconstruction errors falling in this range as normal and those falling outside this range as an anomaly. The following section discusses details regarding the training configurations and the respective results of our experiments.

IV. Experiments

Our approach consists of training the two networks. One for semantic segmentation and the second for anomaly detection. This section elaborates the used datasets, training configurations, and evaluation of the network performances.

A. Datasets

The data used for training the network to perform semantic segmentation is publicly available as RailSem19 [1]. The RailSem19 dataset consists of total 8500 images with 19 semantic labels mapped to JSON (JavaScript Object Notation) files. We use only the rail semantic label and represent the rest of them as background class. We split the dataset into 8390 training images, 60 validation images, and 50 test images. The used dataset with rail image and its corresponding label can be seen in Fig. 3. The data needed for training our second


(a) Image with rail track.

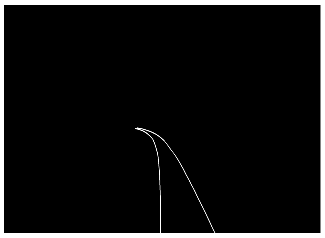

(b) Semantic label.

Fig. 3: Sample training data from RailSem19 [1] dataset.

network, autoencoder, is generated with the help of a semantic segmentation model trained on the above-mentioned dataset. We train autoencoder with the patches of the rail lines which are normal (i.e, free from dirt or any other foreign elements). The patches generated for training, which are extracted from the RailSem19 training set, are given in Fig. 4. The generated dataset has a total number of 9694 images divide into 8443 images used as training set, 373 for validation and 878 for test set. Since it is difficult to acquire data for finding a threshold for anomaly detection. We use the RailSem19 test set to generate test patches and run our trained auto-encoder on them and subsequently calculate the reconstruction errors as mentioned in the Section-III-C. The RailSem19 dataset reflects the normal conditions of railway operations and therefore, they are suitable for generating a threshold for anomalous tracks.

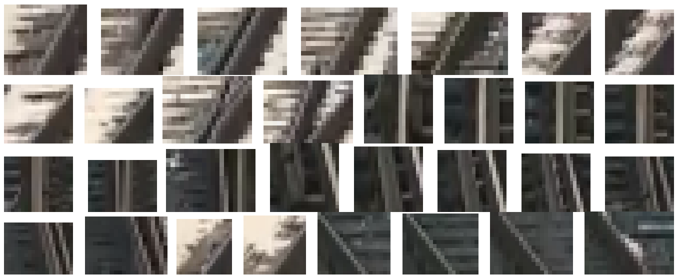


Fig. 4: Generated dataset to train the designed autoencoder.

B. Training semantic segmentation network

The network is trained on 2 NVIDIA GeForce GTX 1080 Ti GPU's with a batch size of 4 and image size of 892 x 596 x 3 (RGB image) as input. The image dimensions are chosen to reflect similar results on images captured by the camera fitted on our locomotive. Also, we use augmentation techniques like random noise, horizontal flip, random brightness, and random contrast while training the network to bridge the delta between images captured on a commercial camera, which we use on our locomotive, and the images from a digital camera, which is available in the dataset.

We keep $w_0 = 0.2$ and $w_1 = 0.8$ in the loss equation of weighted binary cross-entropy (Eq. (1)). We also train with a focal loss of $\alpha = 0.8$ with other parameters same as the weighted binary cross-entropy loss. We use Adam [32] optimizer with a learning rate of 0.0001 for 50 epochs with a batch size of 4. Since the learning can be improved by supplementing U-Net with different backbones like ResNet and VGG-Net, The following table I shows the results of our training results with different backbones. As depicted in the

TABLE I: U-Net training results

Backbone Network	**Loss function *with params***	**Test Metrics mIoU**
VGG	Weighted B.C.E	49.16%
VGG	Focal loss (Gamma=2)[b]	52.78%
VGG	Focal loss (Gamma=4)[b]	50.78%
ResNet	Weighted B.C.E	48.05%
ResNet	Focal loss (Gamma=2)[b]	50.78%
ResNet	Focal loss (Gamma=4)[b]	42.36%

[b]focal loss alpha = 0.8 for all observations

Table I, U-Net with VGG backbone performs better in terms of mean Intersection over Union (mIoU) than other networks and therefore is chosen for performing segmentation tasks. Utilization of focal loss has contributed to some accuracy but, as the hyperparameter gamma increases, we can see that the focal loss decreases the loss value for correct predictions whereas the mIoU does not improve. So we choose the U-Net with VGG-Net Backbone model trained with $\alpha = 0.8$ and $\gamma = 2$ parameters for drawing inference. We found this optimal weight at the 46th epoch which we use for further procedures.

C. Training Auto-encoder network

We train the auto-encoder for 50 epochs with a learning rate of 0.0001 on the images of size $16 \times 16 \times 3$. The structure of the designed auto-encoder is already mentioned and explained in Fig. 1. We use regularization techniques during the training owing to the quantity of data used in the training and also because it tends to overfit. We deploy Batch Normalization [33] for normalizing the outputs of the activation functions to prevent overfitting.

We test our approach on the video data collected from the installed cameras on the running locomotive. We run the proposed approach on a total of 132 frames (a video of 13 seconds) containing anomalous and non-anomalous images. We used a black bag and a note book, introducing different shapes and colors, as anomalies on the rail tracks. Fig. 5 illustrates the performance of our network with a high true positive rate. For a network to be deployed in the real-time application, not only the high true positive rate is important but also a low false positive rate. During our test run no false positives are detected and clean track is classified as a clean track as shown in Fig. Fig. 6-i. The Fig. 6 illustrates the results of our approach on the rail images collected from our installed set up. Fig. 6-a, Fig. 6-d and Fig. 6-g show the raw images. Fig. 6-b, Fig. 6-e and Fig. 6-h show the extracted rails using the semantic segmentation network and Fig. 6-c and Fig. 6-f with the red boxes highlights the area of anomalous objects (placed book and bag). While Fig. 6-i shows rail lines highlighted with pink color which means the rail is safe to traverse and does not contain any anomalies. The extracted rails from semantic segmentation are overlayed on the original image to show the performance of the algorithm. The threshold for reconstruction values of the used dataset are 1.1 and 0.349.

		Predictions	
		Normal	Anomaly
Ground Truth	Normal	120	0
	Anomaly	0	12

Fig. 5: Confusion matrix

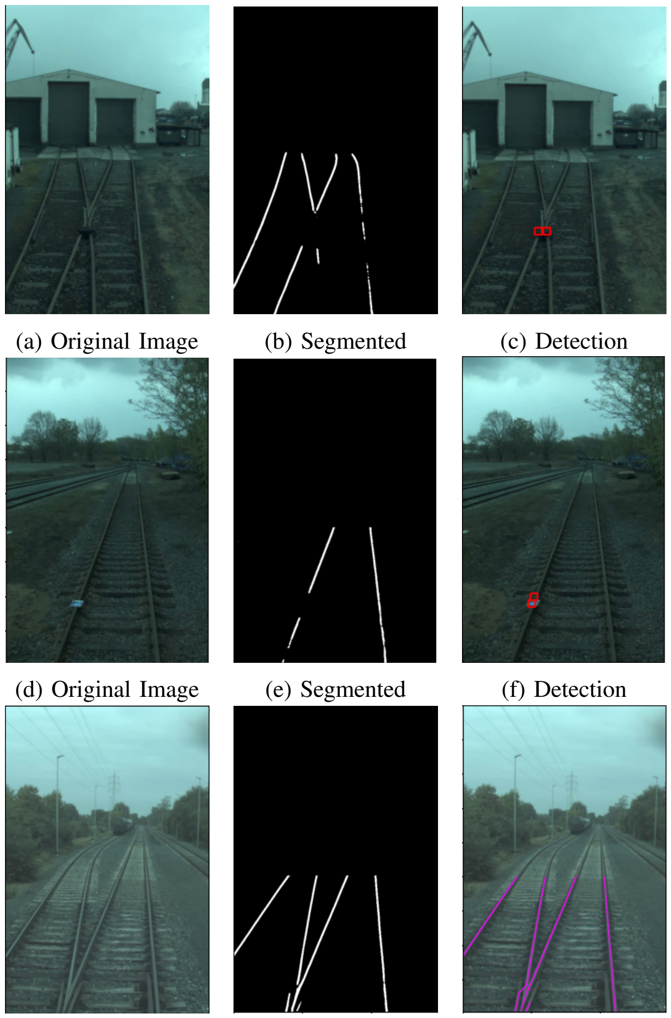

(a) Original Image (b) Segmented (c) Detection
(d) Original Image (e) Segmented (f) Detection
(g) Original Image (h) Segmented (i) Detection

Fig. 6: Anomaly detection on images from a German harbor.

V. Conclusion

This paper presents an approach to robustly detect anomalous objects, of any size and color, like obstacles, coal, and dirt present on rail tracks. The approach is functional in real-time settings with a detection speed of 27 fps and is being tested in one of the German harbors. The images collected from installed cameras on operational shunter locomotive, low-speed vehicles with a max speed of 20 km/h, are used to detect the presence of any dirt, coal, or debris on the rail tracks.

Our data processing pipeline can achieve a segmentation mIoU accuracy of 52.78% and does not produce any false positives or false negatives on deployment. The high accuracy achieves the purpose in low-speed vehicles. Yet can be well generalized for higher-speed rail cars as a preventive maintenance routine also with advance GPUs providing higher detection speed. The approach is successfully tested on different types of rail lines i.e. embedded rail, elevated rail, etc. Due to the unavailability of labelled data for anomalies, we perform self-supervised learning and distinguish anomalies as small as 9×12 inches. Our future work includes testing the approach for high speed railway vehicles.

The current work is focused on anomaly detection at rail lines only. The work can be extended to other interesting infrastructure too, like railway sleepers as they are placed adjacent to rail lines, by adjusting the size of patch and retraining the proposed autoencoder in section III-C. The accuracy of the semantic segmentation network can be further improved by using deeper networks, but at the cost of inference time.

Acknowledgment

This research is part of the project HavenZuG funded by the IHATEC (Innovative Hafentechnologien)[1] initiative of the German Federal Ministry of Transport and Digital Infrastructure (BMVI).

References

[1] O. Zendel, M. Murschitz, M. Zeilinger, D. Steininger, S. Abbasi, and C. Beleznai, “Railsem19: A dataset for semantic rail scene understand-

[1]https://www.innovativehafentechnologien.de/

ing," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 1221–1229.

[2] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision (IJCV)*, 2018.

[3] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, p. 303–338, Jun. 2010.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," 2016.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," 2016.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[8] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," 2017.

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, 07 2009.

[13] F. Ye, H. Zheng, C. Huang, and Y. Zhang, "Deep unsupervised image anomaly detection: An information theoretic framework," 2020.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[15] G. Kumar and P. K. Bhatia, "A detailed review of feature extraction in image processing systems," in *2014 Fourth International Conference on Advanced Computing Communication Technologies*, 2014, pp. 5–12.

[16] D. Terzopoulos, Y. Lee, and M. A. O. Vasilescu, "Model-based and image-based methods for facial image synthesis, analysis and recognition," in *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, ser. FGR' 04. USA: IEEE Computer Society, 2004, p. 397–402.

[17] R. Reisenhofer, S. Bosse, G. Kutyniok, and T. Wiegand, "A haar wavelet-based perceptual similarity index for image quality assessment," *Signal Processing: Image Communication*, vol. 61, p. 33–43, Feb 2018. [Online]. Available: http://dx.doi.org/10.1016/j.image.2017.11.001

[18] R. Roslan and N. Jamil, "Texture feature extraction using 2-d gabor filters," in *2012 International Symposium on Computer Applications and Industrial Electronics (ISCAIE)*, 2012, pp. 173–178.

[19] D. Rumelhart., G. Hinton, and R. Williams, "Explorationsin the microstructure of cognition," in *Parallel distributed processing*. MIT Press, Cambridge, MA, USA, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362.

[20] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2021.

[21] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International conference on neural information processing*. Springer, 2017, pp. 373–382.

[22] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, J. Ruiz-Shulcloper and G. Sanniti di Baja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 117–124.

[23] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion. New York, NY, USA: Association for Computing Machinery, 2015, p. 111–112. [Online]. Available: https://doi.org/10.1145/2740908.2742726

[24] L. Le, A. Patterson, and M. White, "Supervised autoencoders: Improving generalization performance with unsupervised regularizers," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/2a38a4a9316c49e5a833517c45d31070-Paper.pdf

[25] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. van den Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," 2019.

[26] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," 2016.

[27] R. Zhang, L. Du, Q. Xiao, and J. Liu, "Comparison of backbones for semantic segmentation network," *Journal of Physics: Conference Series*, vol. 1544, p. 012196, 05 2020.

[28] L. Torrey and J. Shavlik, "Transfer learning."

[29] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[30] M. Hahsler, M. Piekenbrock, and D. Doran, "dbscan: Fast density-based clustering with R," *Journal of Statistical Software*, vol. 91, no. 1, pp. 1–30, 2019.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, p. 318–362.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.