Railway track fault detection

Thesis work

Mathematics Expert in Data Analytics and Machine Learning

Author:

Tamás Demus

Supervisor:

Dr. András Lukács

lecturer

Department of Computer Science

Eötvös Lóránd University

Faculty of Science

AI Research Group

Budapest, 2023

Abstract

Railway track monitoring is an essential part of infrastructure maintenance since the beginning to ensure safety of railway operation. Several track faults are identified throughout the time, starting from minor surface defects up to major rail cracks and more severe damages. During normal railway operation the task of the maintenance personnel is to detect, identify and monitor the propagation of such defects, so when it comes to a certain degradation the necessary prevention steps can be taken to ensure railway safety in a sustainable level. Among many, a key inspection method is still visual inspection due to the high variety of possible faults and many aspects that need to be considered during evaluation of necessary steps. So far this has not been fully replaced with instrumented monitoring or any highly accurate approach that is prevalent in rail industry. Current study aims to develop such method by utilizing machine learning algorithms that went through severe development in the past decades. The actual case study relies on real life data received from MÁV Central Rail And Track Inspection Ltd., the responsible company for rail monitoring in Hungary. The dataset contains video recordings of the rails done by the measurement vehicles during regular track investigations. These video files are subjected to a first experiment, application of an autoencoder model followed by anomaly detection methods to detect frames of the video that might contain rail faults. The anomaly detection was performed by applying two methods, the Isolation Forest algorithm and a loss-based approach over a set of autoencoders with four different type of encoders. The project is limited to a first trial of a sample video file to gain first experiences with such models and to perceive application possibilities. In this dataset the outliers were defined as rails covered with grass and video frames containing double rail sections, and were successfully identified by most of the algorithms. The internal behavior of the model is visualized and examined by a subsequent PCA and t-SNE methods to reduce the dimensionality of the dataset. Current work is a continuation of the Mathematic Modeling Practice subject as part of a training held by the AI Research Group of Eötvös Lóránd University and considered as thesis work.

Acknowledgement

I would like to express my deepest gratitude to my advisor Dr. András Lukács, who supported me with his guidance and valuable insights throughout my project work. Also, I would like say thank you to the colleagues of AI Research Group of Eötvös Lóránd University who provided the possibility to attend on this special one-year training and gave their best to enlighten us as students to the world of data analytics and data science, Machine Learning and Artificial Intelligence.

I am also grateful to Mr. Ákos Marosi granting the possibility to have a look into the world of railway track inspection and providing access to the video footage used as dataset. His glowing eyes always represent the deepest passion some can feel towards the railway. I am thankful to Ms. Ágnes Kemény for allowing the joint work taken with the colleagues from MÁV Central Rail And Track Inspection Ltd.

I am also thankful for my partner, friends and coworkers who endured me on this journey while I was balancing on the borders of insanity and obsession.

Furthermore, I do hope that the path taken is not the end, but only a beginning.

Contents

1 Introduction **5**
1.1 Current use case . . . 5
1.2 Railway track fault detection . . . 5
1.3 Previous work and results . . . 5
1.4 MÁV CRTI Ltd. . . . 6
1.5 Modelling approach and problem statement . . . 7
1.6 Structure of the document . . . 8

2 The dataset **8**

3 Description of the model **9**
3.1 The Autoencoder . . . 9
3.2 Anomaly detection . . . 10
3.3 Data preprocessing . . . 10
3.4 Model training . . . 10
3.5 Performance evaluation . . . 11

4 Software implementation **11**

5 Results **12**
5.1 Encoding via VGG19 . . . 12
5.2 Encoding via VGG19 with batch normalization . . . 14
5.3 Encoding via ResNet50 . . . 17
5.4 Encoding via EfficientNetV2L . . . 19

6 Discussion **22**
6.1 Learning curves . . . 22
6.2 Planar visualization . . . 23
6.3 Classification metrics . . . 24
6.4 Detected anomalies . . . 25

7 Conclusion **27**

1 Introduction

1.1 Current use case

Visual inspection is inherent part of railway track monitoring activities to ensure proper condition of the infrastructure and thus provide a safe railway operation. Besides many possibilities of track monitoring, such as ultrasonic or laser-based measurements, vehicle riding parameters (or ride comfort) analysis, still visual inspection is evitable. During regular track investigations video recordings done from the rails itself from different views which allows off-line evaluation. An accurate and reliable evaluation method is desirable that would help the maintenance personnel to automatically find the track failures in these video recordings, or at least limit the duration of these to reduce required human resources. Usually 60-70 rail defects found every year during hundreds of kilometers of inspection rides. This already highlights the difficulties of the given problem.

1.2 Railway track fault detection

Image based railway track fault detection is narrowed down to certain set of faults in terms of applied methods. There are two key areas, the first one is surface defect detection and the second one is the identification of missing elements of the rail, mostly focusing on fastening elements.

Several tools are already applied with good success, starting from pure image processing approaches (edge detection, threshold application, etc.) up to traditional and machine learning based algorithms (e.g.: SVM, random forests). Image processing offers the opportunity to apply convolutional neural networks that could describe the content of these images in a compressed format capturing their main features both in (image-wise) local and global scale. These compressions can be used for further tasks such as classification or anomaly detection.

A deeper overview of the literature is provided in [1] with examples of the key areas and applied tools.

1.3 Previous work and results

Current research started as part of the studies on the Mathematics Expert in Data Analytics and Machine Learning postgraduate specialization program [2] held by the AI Research Group of Eötvös Lóránd University [3]. During the one-year program a thesis work has to be created that is often preceded by a modeling practice in the first semester. This project follows the same approach as railway track fault detection models were already built in the first semester. However, the characteristics of the dataset used at that time did not allow a successful application of a model however valuable experience is gained together with a boost of motivation to follow up the topic and deepen the knowledge in the mentioned field.

During the first semester basic convolutional neural networks were built with classification head to identify images with defective railway tracks [1]. LeNet-5, AlexNet, VGG16 and ResNet50 were applied, partially utilizing transfer learning as well (indicated with _p suffix in the following).

The dataset was taken from the Kaggle webpage [4] consisting of a limited number of images with defective and non-defective railway tracks that were already annotated. The images combine a high variety of failures (such as rail cracks, missing fastening elements, complete track failures) with very limited examples. Additionally, the photos are taken from a wide range and angle, lacking any kind of standard positioning. This leads to a very specific dataset. Furthermore, the low amount of images and imbalanced characteristics of the dataset limits the possibility of effectively apply neural networks due to the high number of parameters that need to be optimized in such models.

The results of the modeling is shown on Figure 1, where the performance of different bootstrapped models on the splits of the dataset is given. Bootstrapping technique was applied in the end to investigate how far the validation and test set could describe the initial training dataset. As shown in the figure, the high range of the obtained results of each model already indicates that the distribution of the training images differ from the other splits of the dataset.

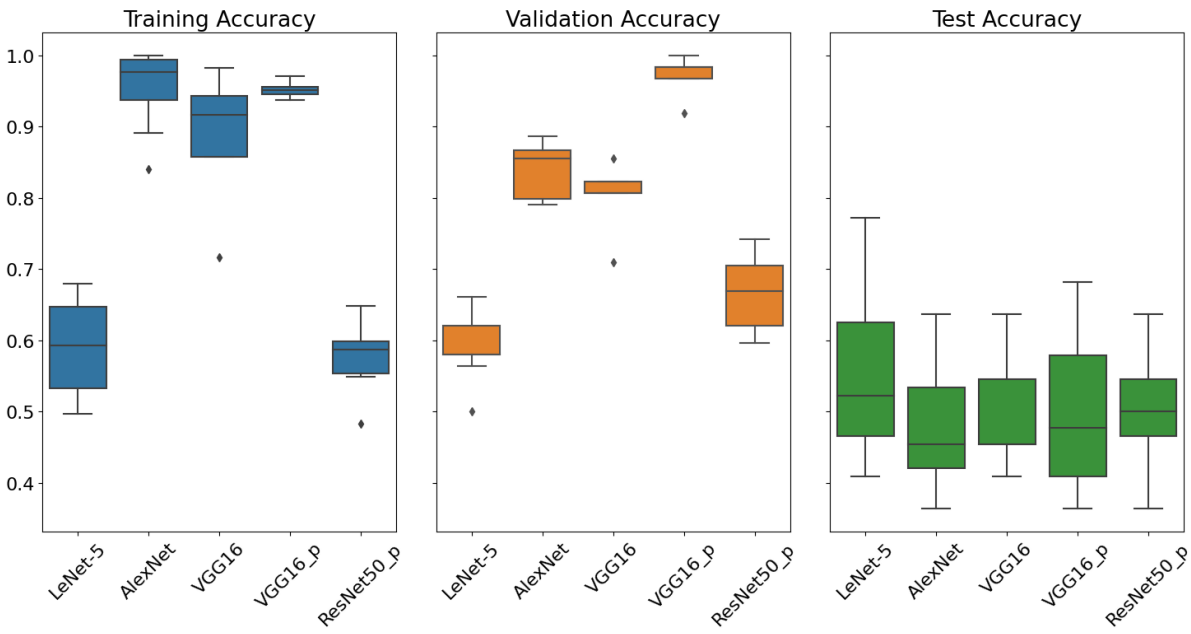


Figure 1: Results of first classification approach on the Kaggle dataset

Most of the models successfully learned the features of the training and validation datasets, however on the test dataset all failed to classify the images in an acceptable level, mostly achieving the level of a random classifier only. This lead to the realization that the low number of images compared with the high variety of the track failures will result in a situation where the model can easily learn the specifics of the training and validation dataset preventing of any generalization to the test dataset.

As a followup a search for a more extended dataset and more refined models were started. The company MÁV Central Rail And Track Inspection Ltd. (MÁV CRTI Ltd.) [5] was approached as they are proficient in railway track inspection and have the equipment and data that could be used for building such a model.

The expectation towards the dataset is to contain much more image than the set obtained from the Kaggle webpage and the type of defect should be limited (for example only to surface defects). The variance of the images should be narrowed down, for example the positioning, view angle or the distance of the camera from the rails shall be restricted.

Due to the nature of the problem, any real life dataset is not expected to have any kind of annotation, therefore the modeling approach needed to be change to unsupervised learning. The problem statement refers to find certain defects of the rails, that is basically a result of a deterioration process, which end up in a certain deviation from the normal condition. This leads to the idea to apply anomaly detection models and try to identify the video frames that contain such deviations.

1.4 MÁV CRTI Ltd.

The MÁV CRTI Ltd. was established in 1996 by MÁV Hungarian State Railways Co. The scope of the company covers the fields of technical inspection and analysis related railway tracks, rails and corresponding structures:

- Geometric measurement of tracks and geometric condition survey
- Measurement, examination and qualification of railway rails
- Qualification of new and used superstructure materials
- Examination of bridges
- Examination of substructures
- Development related to rail measurement, examination and line maintenance

A comprehensive overview about the company, it's activities and history can be found at [5] and in [6].

The Rail Diagnostic Department carries out regular measurements on the railway tracks in order to determine the overall condition of the rails and thus ensuring the safety of railway operation. These inspections are carried out by special railway measurement equipment and inspection vehicles, Starting from the simplest visual inspections carried out by the maintenance personnel up to special ultrasonic examinations and rail profile measurements. With such equipment rail surface and internal defects can be detected and detailed information about the track profile can be obtained along hundreds of kilometers of railway tracks.

The Rail Diagnostic Department already has some experience with machine learning based rail defect detection providing options for benchmarking the models created through this project. It also reflects the importance of such approach, as today visual inspection demands heavy efforts let that be the work done by the maintenance personnel (intrinsically walking along the tracks) or the monitoring of the video footage taken during the inspection rides.

Currently, MÁV CRTI Ltd. operates two vehicles for rail inspection purposes, the SDS and FMK-008 shown on Figure 2, both equipped with different measurement and inspection systems. Fortunately both of them is equipped with video recording, however with different systems. After a first view on the video files the system of the SDS vehicle is selected for a first modelling approach.

SDS FMK-008

Figure 2: Railway inspection vehicles of MÁV CRTI Ltd.

1.5 Modelling approach and problem statement

During the first part of the project convolutional neural networks with classification head were applied on annotated dataset. A step forward was taken by obtaining real world data, however this comes with the burden of missing labeling. Therefore, the modeling approach was also changed from supervised learning to unsupervised learning.

Autoencoders form a subgroup of algorithms of unsupervised learning. The main target of such models is to reconstruct the original input on the output. The basic idea behind an Autoencoder model is to generate feature vectors from the input dataset (encode the dataset), this is the so-called Encoder part. This set of feature vectors, called as latent space contains the representation of the dataset. In the second part, the Decoder parts tries to recreate the input data based on the feature vector generated by the Encoder. In current use case this means that the single images of the rail is translated to feature vectors and then the original image is recreated with some error using the Decoder.

The general structure of Autoencoders is shown in Figure 3.

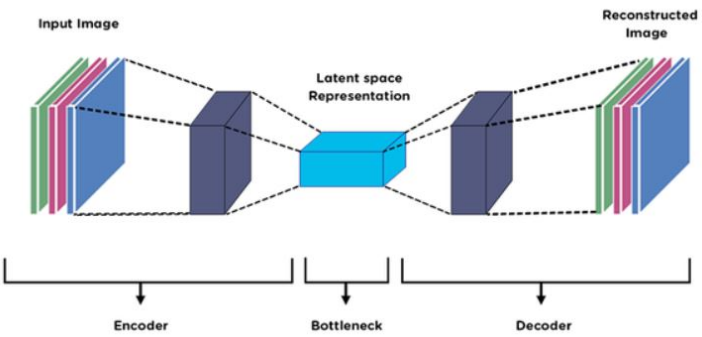


Figure 3: General structure of Autoencoders [7]

A comprehensive overview about Deep Learning algorithms is given in [8]. Chapter 14 of this book explains the mathematical structure of Autoencoders. Another introduction can be found at [9].

Autoencoders allow two basic type of anomaly detection methods. First approach is to compare the input and output information, in general the autoencoder can learn the features of the common images effectively, meaning that the decoded part will be quite comparable with the input. This means that the distance (or loss) defined between the input and output should be lower than the case, when there is an outlier that contains features that are not seen by the model, resulting in a not precise decomposition and decoding.

Secondly, the representation of the images stored in the latent space can be searched for outliers. As rail defect happen very seldom, their feature vector should occur very rarely as well. If the encoder is able to extract the differentiating features, then the outlier characteristics can be preserved in the latent space. Any general algorithm can be used to find these vectors, such as Isolation Forest, One-Class SVM or Local Outlier Factor (please see [10] or [11] for additional examples). Furthermore, the latent space can be clustered by any general clustering algorithm, for example: DBSCAN, OPTICS, and so on, please refer to [12]. These are also capable to identify outliers or clusters with limited number of elements.

In this way the problem was reformulated and traced back to anomaly detection, thus resulting in the following key questions:

Q1 Can anomaly detection algorithms used to detect rail defects?

Q2 What models could be applied on the given dataset?

Q3 What accuracy rate can be achieved with the models?

1.6 Structure of the document

The Section 2 gives a deep view on the sample dataset. The structure of the models introduced in Section 3. The realization of the models, structure of the software code is explained in Section 4. The results are interpreted in Section 5 and discussed in Section 6. Further steps and possible improvement options covered with a summary of the results in Section 7.

2 The dataset

The dataset that is studied is derived from videos of track inspection vehicles recorded during regular inspections. The video system of the SDS vehicle records both rails from two angles resulting in four video footage parallel. A single footage was selected as it provides a static positioning relative to the tracks with good protection against changes of the lightning of the surroundings. The video system records with a resolution of 720x288 (width x height) with RGB channels at 50 fps rate. The video files are binary files without any compression applied on them.

A single footage sample video of approximately 3 minutes was taken as a starting point. This video contains a side view of a rail, that is defined as *normal* rail along with a few seconds of rail covered with *grass* or showing a *double rail* section. Latter two is considered as outlier to conclude on the first experiment building up the anomaly detector model. These are not comparable to the real life rail defects, however they allow an easy model setup and evaluation. Some examples of the images extracted from the video is shown on Figure 4.

Figure 4: Sample images from the dataset

This sample video was sliced to images, resulting in the dataset shown in Table 1. The resulting dataset is imbalanced, and the so-called outliers can be easily identified. In general annotation can not be assumed for such problems, however in this case it was provided manually to grant the possibility of performance evaluation.

Image type	Number of images
Normal rail	8640
Rail covered with grass	64
Double rails	29
Total	8733

Table 1: Dataset obtained from sample video

Observing the quality of captured video frames more in detail we can see that there is no slurring on the images that is remarkable considering that the recording was taken with a vehicle speed up to even 80 to 100 kilometers per hour. However, a slight fish eye distortion can be seen that is noticeable mostly in the rail itself, as it is not tend to follow a straight line, a small bending effect is given. Also, the lighting of the pictures result in a brighter spot in the center. The image quality remains stable in the sample video, including the illuminance that is secured by the shrouds applied on the vehicle around the cameras.

Later on the course of the work MÁV CRTI Ltd. provided further videos from both vehicles ranging up to 450 GB of raw data that can be used for further training, evaluation or optimization the models. In these videos 68 rail defects are recorded, and their approximate location is also known. These defects are hidden in recordings ranging up to a duration of some hundred hours. This already indicates the main challenge of this problem, that failure rate (faulty images relative to all images) is well below the value of 1%.

3 Description of the model

3.1 The Autoencoder

Current problem requires images to be processed, therefore the latent space representation shall be created by a neural network fitted to work with images as this is expected to learn the features that might differentiate between non-defective rail images and outliers. The same applies for the decoder part as well. Four type of Encoders used as listed below with the consideration to keep the model simple, allow comparably fast training (which is driven more by the size of the dataset).

The VGG19 is one of the first deep convolutional networks introduced in [13]. In the second model batch normalization was added after each convolutional layer. The ResNet50 [14] overcomes the issue of the vanishing gradient by adding the skip connections. EfficientNetV2 [15] is developed to reduce training time and provide better parameter efficiency than previous models.

During the encoding the input image sizes are gradually reduced in four steps, resulting different shapes of the latent space due to the different number of filters applied. An overview of the resulting

latent space considering an input size of 704 x 288 is shown in Table 2. For comparison the original image of size 720 x 288 x 3 has a total number of 622080 to describe a single instance.

Encoder type	Width x Height x Filters	Number of parameters
VGG19	22 x 9 x 512	101376
VGG19 with batch normalization	22 x 9 x 512	101376
Resnet50	22 x 9 x 2048	405504
EfficientNetV2L	22 x 9 x 1280	253440

Table 2: Latent space shape of different Encoders

Once the feature vectors obtained, the original image can be reconstructed. In current study VGG19 is used as basis decoder model. As the processing of the data has to be done backwards, the whole structure is reversed and the convolutional layers are replaces with transposed convolution.

In order to fit the Decoder input to the Encoder output a FilterMatching part is introduced. As the difference between the latent space and decoder input is only the number of filters, the shape is adjusted via additional convolutional layers as listed below. This was done to avoid that during stepping forward on the first layer of the decoder a lot of filters should be compressed into a much smaller set. Ideally the decoder and encoder part shall form an hourglass shape, but in this case only a single model was used for decoding to maintain simplicity.

- In case of the VGG models, an identity mapping is applied
- For the ResNet50, a two-step approach is followed, each is halving the number of filters
- When applying the EfficientNetV2L model, also a two-step decomposition is introduced first the number of filters is reduced to 896 and then to 512.

3.2 Anomaly detection

In the loss-based approach the loss function is set as the pixel-wise mean squared error of the input and decoded image. The threshold to designate an image as outlier is to have higher loss value (distance between input and output image) than the mean loss overall the whole dataset and three times the standard deviation added together. This is an initial guess, depending on what outliers are found, the multiplier of the standard deviation might need to be changed. Such optimization of the multiplier (or threshold) is only possible when annotation is given. As this is not the usual case in this problem, an optimization is only possible on manually labeled dataset and with the assumption that further dataset contains images from the same distribution. Practically this means that the most common *normal* rails should be included in such optimization process, including normal track, bridges, junctions, road crossings, etc. When it is known that no defect are present in the training set, then a limited fine-tuning is possible, lacking the information on how losses linked to certain type of *normal* rails differ from each other.

The sklearn implementation of Isolation Forest algorithm described in [16] is used when looking for outliers in the latent space. The default setting were applied, parameter optimization could be a possibility to improve the model further, however for comparison purposes this first approach is accepted.

3.3 Data preprocessing

During modeling besides augmentation no image manipulations were applied except the normalization during entering the neural network and resizing to 704 x 288 x 3 to ensure decoding of the images to the same original size.

3.4 Model training

Transfer learning is applied in case of the Encoders to ensure an efficient approach, the model weights from the Imagenet pretraining is selected for this purpose. In case of the Decoders no transfer learning is available due to the reversed structure of the network, there only a random initialization was used.

The loss function is defined as pixel-wise mean squared loss between the original and replicated images. Adam optimizer is applied with a learning rate of $5 \cdot 10^{-6}$. The batch size for training is set to 8.

The dataset is split to *train*, *validation* and *test* set, considering the imbalance nature of the dataset, during split a stratified approach was applied. The training of the models were done on the *train* dataset for 100 epochs, the *validation* set was used to monitor how well the model generalize and to detect possible overfitting or other training issues. The *test* dataset was used for independent analysis of the model, how the results presented in Section 5 is interpreted over the whole dataset offering the opportunity to compare which images detected as outliers (or missed) by the different detectors models (loss based or Isolation Forest).

3.5 Performance evaluation

The first approach on the performance evaluation is the general classification metrics based on the confusion matrices of each approach. Furthermore, a visualization is provided via a two-step dimensionality reduction approach: first a PCA was used to determine the 50 most important features of the dataset, then a t-SNE was applied to find the two most important feature. This allows a graph representation of the dataset. This approach was applied on the input images, latent space vectors, filter matched vectors and on the decoded images, allowing visualization how outliers behave over the whole model.

4 Software implementation

The project is written in Python language partially in python scripts directly, partially in Jupyter notebooks. The documentation is created in LaTeX. The overall software code is stored at https://github.com/demustamas/project_work. Due to storage limitations the trained model files and the original dataset is not uploaded to GitHub. The basic folder and file structure is as follows:

- ./build - Build directory of LaTeX, contains project documentation
- ./data - Contains the dataset (not available on GitHub)
- ./results - Contains resulting models, images, tables
- ./tex_content - LaTeXdocumentation files
- ./tex_images - LaTeXimages for documentation
- ./tex_refs - Style files, bibliography
- ./toolkit - Contains main scripts
- ./toolkit/classes.py - Class definitions for loading the dataset
- ./toolkit/pytorch_tools.py - Class definitions for PyTorch
- ./project_work.ipynb - Jupyter notebook project file
- ./project_work.tex - Main LaTeXdocumentation file
- ./project_work_pres.tex - Final presentation of the project
- ./video_slicer.ipynb - Jupyter notebook used for slicing the videos to images

The raw video files (binary files) were sliced using the video_slicer.ipynb. The anomaly detection is then performed by project_work.ipynb and the results, including trained models, extracted plot images, csv files are stored in ./results.

The documentation is created in LaTeX, the main file is project_work.tex, a final presentation will be stored under project_work_pres.tex.

The main script files are classes.py and pytorch_tools.py. These contain the class definitions used to import the dataset, define the models of the Autoencoder and AnomalyDetector with the corresponding methods used for training, predicting and evaluating the results.

The scripts are prepared to run either on CPU or on GPU, depending on the available IT infrastructure. The computation expensive steps were run on the servers of AI Research Group of Eötvös Lóránd University.

5 Results

5.1 Encoding via VGG19

The learning curves of the Autoencoder training is presented on Figure 5. The initial high loss values are reduced significantly during the first few epochs. The train and validation losses converge quite close to each other due to the fact that the train dataset (as the overall dataset itself) contains mostly *normal* rail images. Also, the variation of the curves are very small, the validation set represents well the training set. The validation curve is slightly above the train losses, there is no sign of overfitting or abnormal training of the model.

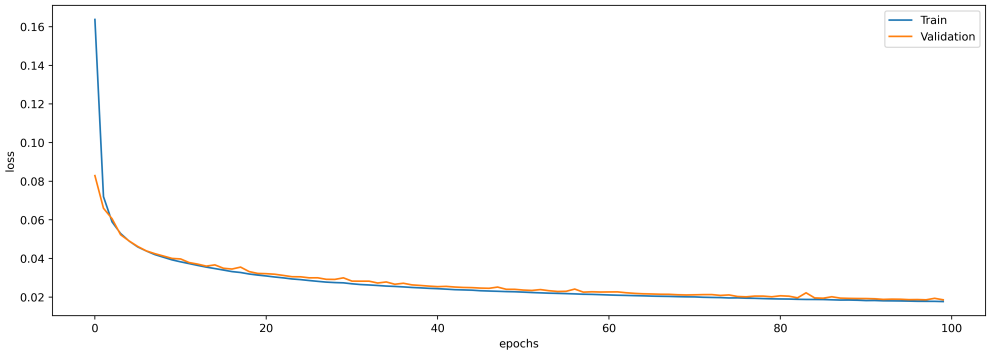


Figure 5: Learning curve of the VGG19 Encoder

After training some example images are predicted (encoded and decoded) to visualize the performance of the Autoencoder. This is represented on Figure 6. The sample autoencoded images show the main characteristics that is recorded by the model. The vertical shiny edge of the rail is easily identified along with the side surface. The details that are close to the center of the image or are brighter parts of the image are more likely to be captured. Even the ballast stones can be recognized demonstrating the power behind such models. It is to be noted that the reproduced image is not fully complete, the more towards the image edge, the more black areas given resulting in high loss of information.

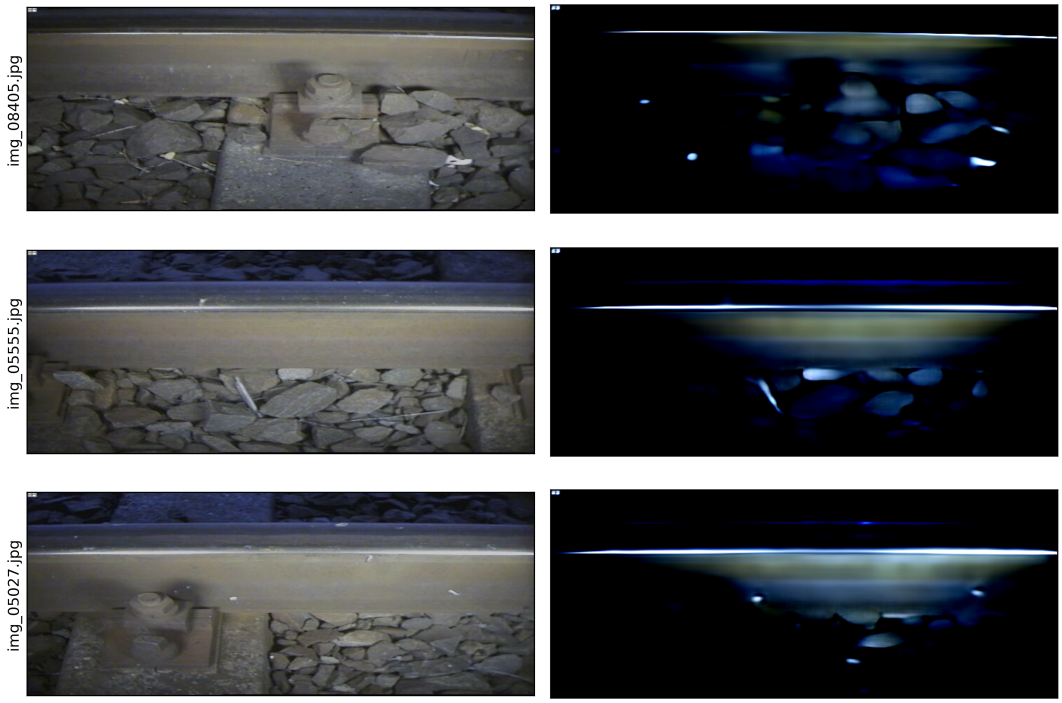


Figure 6: Predicted images in case of VGG19 Encoder

The visualization of the different model stages based on the PCA / t-SNE approach is depicted on Figure 7. The colors indicate whether we observe a *normal*, *grass covered* or *double rail* image, the latter two is considered as outlier. The alpha value of the markers indicate the loss value of the image, the higher the alpha the less transparent the marker is. The outlier images form a certain clustering already among the input images, that is, at least to some extent, retained after encoding. This can be seen on the input phase already in the top right and center bottom part. These clusters are separated partially after encoding. There is a clear cluster made of outliers after encoding in the bottom right part, but this cluster does not contain all members, some are still embedded among the *normal* images, in the top right part, however slightly moved to the boundary of it. It can be also seen that after decoding these outliers are structured back to their original position in the visualization space. This indicates that the encoding is somehow clustering the dataset, whilst the decoding part readjusts this clustering back to its origin, basically proving that the main idea of the Autoencoders is represented in the model.

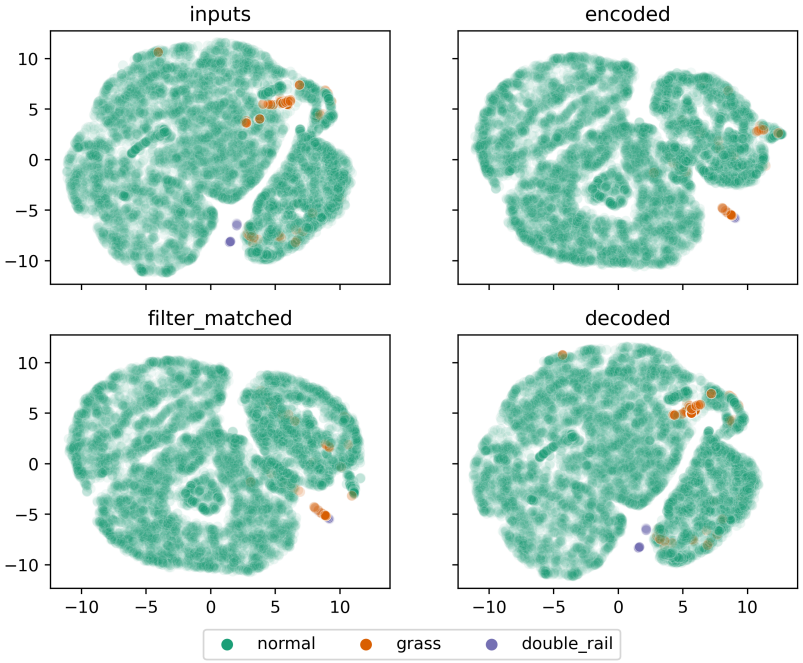


Figure 7: PCA / t-SNE visualization of the VGG19 Encoder

The loss values of each image of the whole dataset together with a marking of the outliers is shown on Figure 8. The peaks in the loss value are remarkably indicating the positions of the outlier images. The concentration of the outliers are also confirmed based on the video footage, first a few seconds of grass covered section can be seen, then in the second part a longer period of grass coverage with the appearance of the double rails is present in the video. It is to be noted that, for example in case of the first *grass* image the loss value is not so well distinguishes the image from its neighboring images. Similarly, but in the opposite direction, at some *normal* pictures the loss value is significantly higher than the values of the preceding or following images. This denotes that this approach most likely will not a perfect identification of the anomalies.

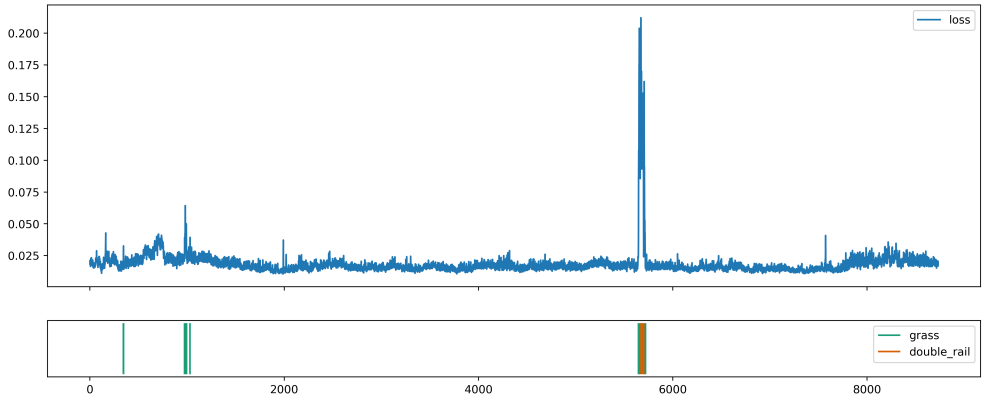


Figure 8: Loss values of the dataset with VGG19 encoding

The confusion matrices for the loss-based and Isolation Forest anomaly detection techniques can be seen on Figure 9. The loss-based approach identifies the ⅔ of the outliers correctly, whilst providing no false positive. The threshold of three times the standard deviations seems to be a quite conservative approach, considering the actual use case that the target is to identify all the outliers, even with the cost of increasing the false positive rate, lowering this threshold might be useful. The Isolation Forest gives slightly different results, finding approximately half of the outliers, and providing some number of false positives.

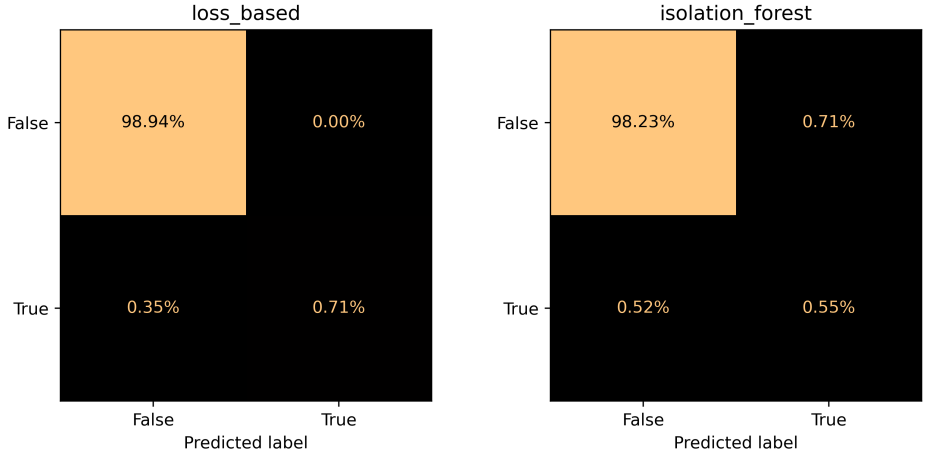


Figure 9: Confusion Matrices of the VGG19 Encoder

5.2 Encoding via VGG19 with batch normalization

The batch normalization did not deliver a significant change to the characteristics of the learning curves as can be seen on Figure 10. Similar observations can be done as for the model without batch normalization. After the first few epochs the loss values significantly decrease and continues getting reduced indicating that the model is capable to learn new features. The train and validation losses run quite close to each other due to the many very similar images, and indicating a good representation capability of the train dataset. No overfitting or training issues can be seen, however the validation loss curve seem to show slightly higher variance. At 100 epoch the loss value is slightly higher than it was with the VGG19 model. The difference is minor, normally the batch normalization shall decrease the learning times of a model, however the resulting loss values highly depend on the weight initialization as well. In this case the initial validation loss was also higher at start compared to the pure VGG19 model (0.099 vs 0.082, with and without batch normalization respectively).

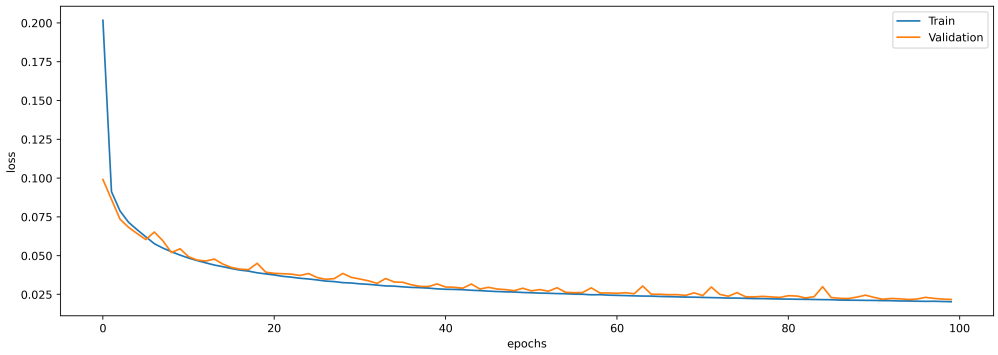


Figure 10: Learning curve of the VGG19 (BN) Encoder

The samples of the predicted images bear the some characteristics as it was seen on the VGG19 model. Some examples are given on Figure 11. More details are reconstructed in the center of the picture or in brighter part of the image. As basic feature, the main edge of the rail is also visible together with the side surface. Smaller details such us outlines of ballast rocks are also captured to a certain extent. This shows that the introduction of batch normalization did not change significantly the quality of the decoded pictures. However, still not recreating the whole image.

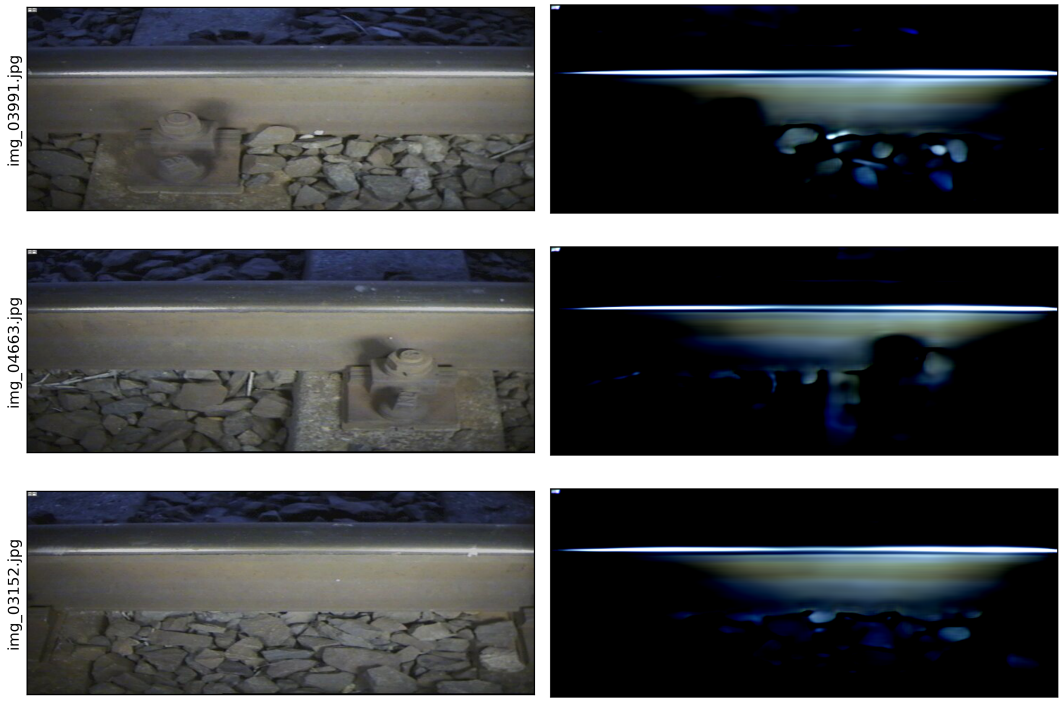


Figure 11: Predicted images in case of VGG19 (BN) Encoder

A more interesting change can be identified on the PCA / t-SNE visualization on the Figure 12. The change of the cluster follows the same character as without batch normalization, but the separation of the outliers happens differently. The outliers that were still part of the *normal* cluster, close to the boundary, now show a more accented deviation based on their loss values as can be seen on the top right part of the encoded phase. But some outliers are stuck deeper inside the clusters of the *normal* images. This can be seen on the bottom left part of the encoded phase. The outliers on the bottom right part are still well isolated from the main *normal* cluster. During reconstruction of the images, the decoded visualization shows a clear difference to the input characteristics based on the position of the outlier images. As the PCA and the t-SNE and their joint effect is not optimized, it can not be excluded that this change is resulting only from the visualization technique, however we shall keep in mind, that this could be also an indication of a different model behavior.

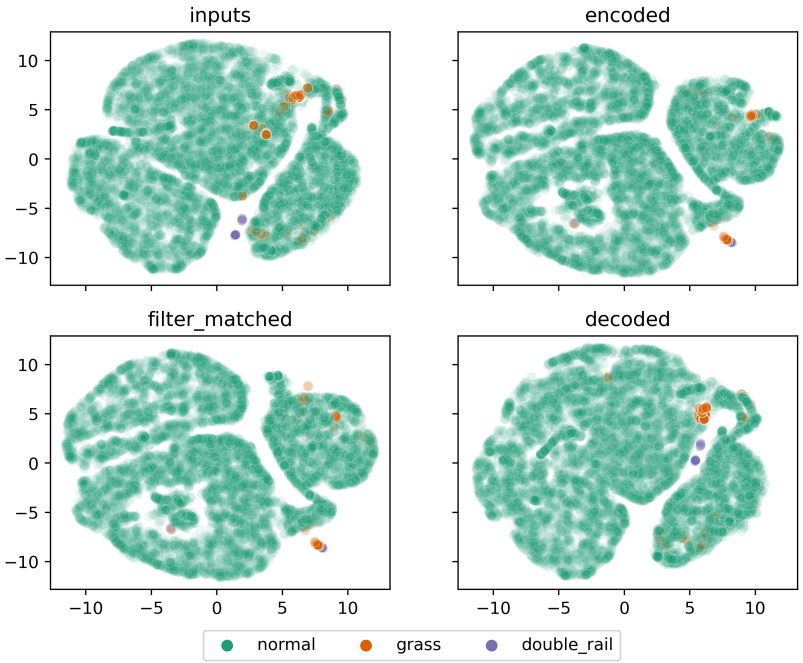


Figure 12: PCA / t-SNE visualization of the VGG19 (BN) Encoder

The loss values of the images show a very comparable series as the VGG19 model as shown on Figure 13. Emphasizing that the effect of the batch normalization can not be really identified through the comparison of the loss values. Most of the outliers are indicated with high loss values, especially the part with the *double rail* cluster. On the other hand some outliers still *hide* inside the loss values as it is not deviating from the others. An example for this is the first *grass* outlier.

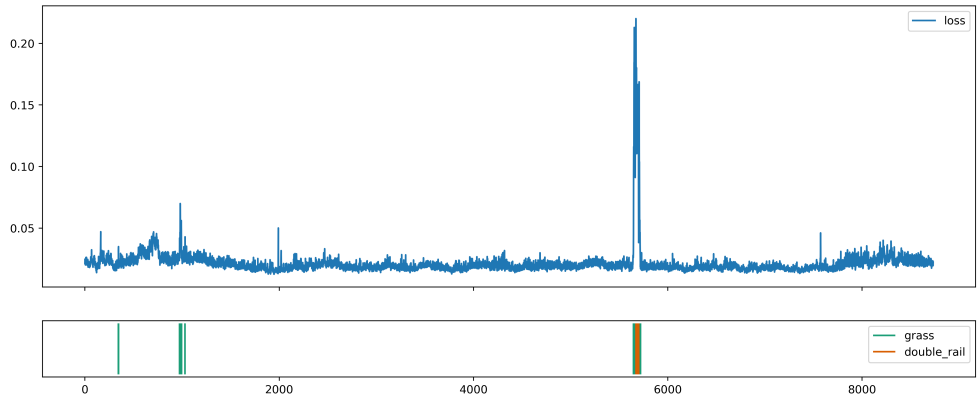


Figure 13: Loss values of the dataset with VGG19 (BN) encoding

Interestingly the confusion matrices deliver different results than without batch normalization. These matrices are shown on Figure 14 The loss-based approach delivers almost exactly the same results, there is a slight change that can be captured in the increase of the true positive indications and thus on the reduction on the false positives (missed outliers). Applying the Isolation Forest algorithm on the bottleneck vectors shows a slight performance increase, the true positive predictions increased, whilst the false positive and true negative rate is reduced. This reduction is significant in case of the true negative indications.

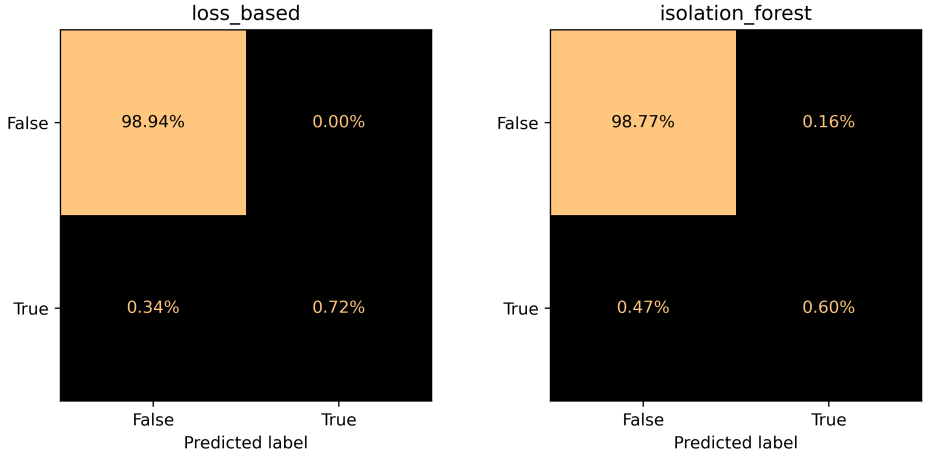


Figure 14: Confusion Matrices of the VGG19 (BN) Encoder

5.3 Encoding via ResNet50

The learning curves as shown in Figure 15 act in a very similar manner as with the VGG19 models, the difference that can be identified is the variation of the validation losses over the epochs. Apart from that, the same starting and resulting loss is reached by the model, without any overfitting or training problems. The validation curve follows the training curve, supporting that the training dataset is representative to the whole dataset. It is to be noted that the training was done in different dataset split ups for all models.

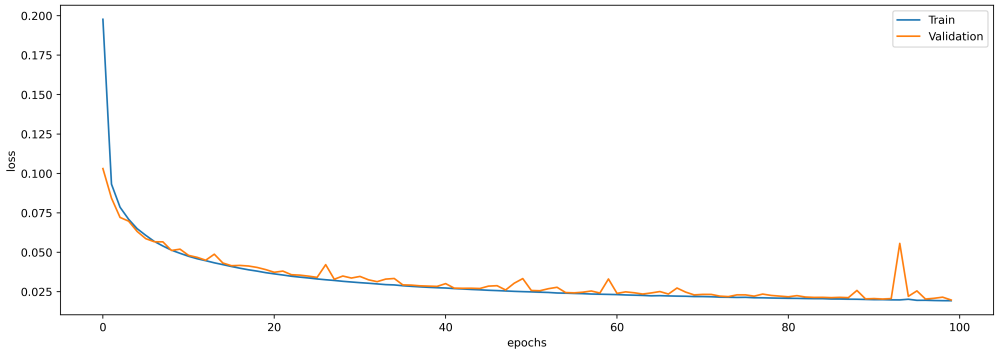


Figure 15: Learning curve of the ResNet50 Encoder

The autoencoded images follow the same pattern as before, major features of the rail is regenerated, the details are concentrating in the center or brighter parts of the images, the main edge of the running surface of the rail and the side surface both can be identified. The representation of the ballast rocks also present. These are presented on Figure 16. It can be concluded that the model learned to encode the images to feature vectors and then the decoding is also successful, however it has to be noted that the resulting image is far from complete.

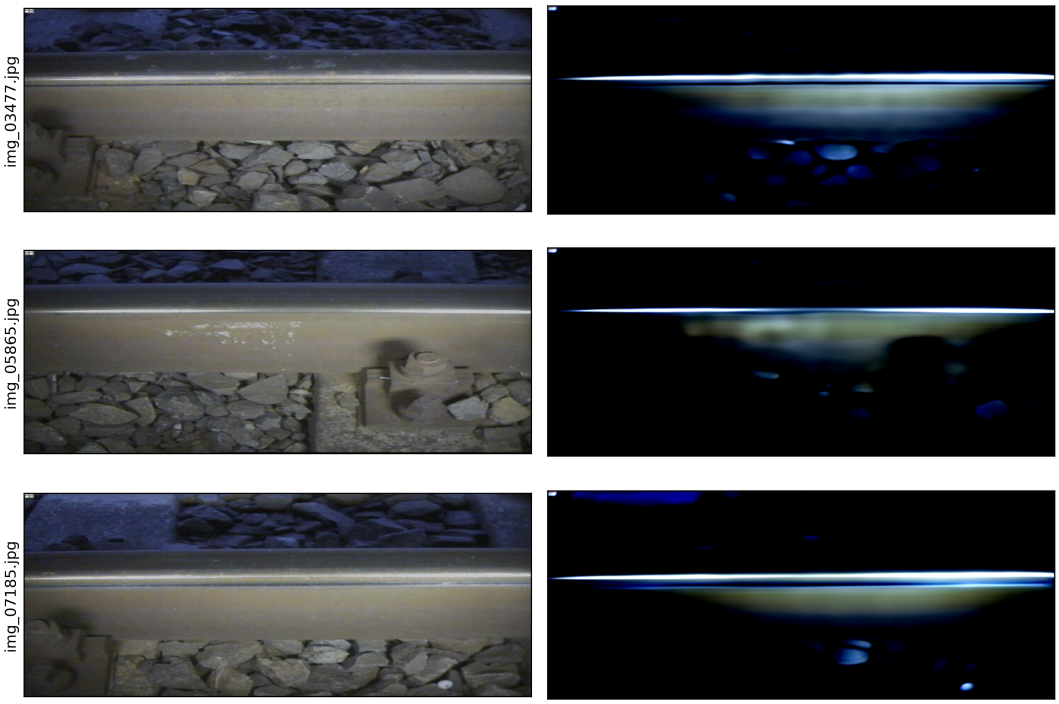


Figure 16: Predicted images in case of ResNet50 Encoder

The PCA / t-SNE visualization shows a little enhancement in the behavior of the model on Figure 17. The outliers of the input images are encoded in a way that they tend to separate from the major *normal* cluster, even the ones that remained in this cluster are closer to the boundary as shown on the top right part of the encoded sub-figure. Even it can be seen that this region or the boundary is *opening up* and the *normal* images do not encompass these outliers so much. The bottom right concentration of the outliers are separating clearly with this model, as seen with the other models as well. The decoded images retain the clustering in a strict manner, compared to the distribution of the outliers on the input visualization, the decoded outliers form a more closed cluster. Some outliers (that usually have low loss values compared to the other anomalies) still remain inside the *normal* cluster through the whole model.

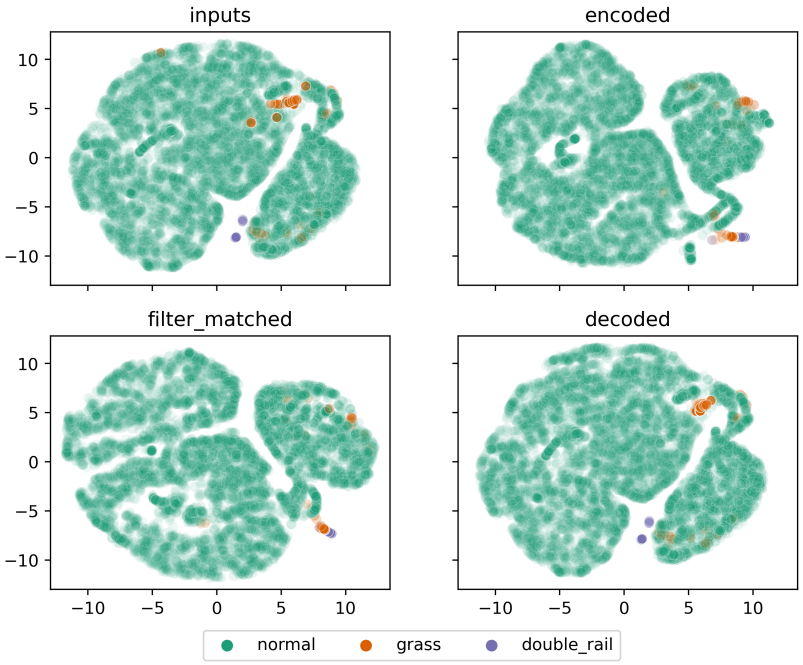


Figure 17: PCA / t-SNE visualization of the ResNet50 Encoder

The loss values depict the same variation over the images as seen for the VGG models, depicted on Figure 18. The majority of the outliers can be easily identified, although not all of them have a high loss value and some images with high loss values belong to the *normal* rail class. The loss values do not indicate a difference between the models seen so far.

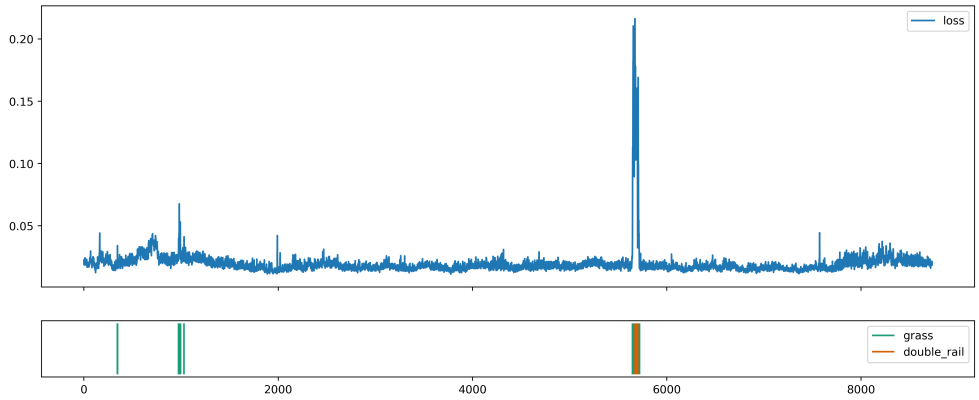


Figure 18: Loss values of the dataset with ResNet50 encoding

The classification metrics stay between the VGG19 models with and without batch normalization. The confusion matrices can be seen on Figure 19. The loss-based approach delivers the same performance as the threshold still too conservative, approximately ⅔ of the true positives identified correctly. The Isolation Forest algorithm identifies true positives in higher rate than the VGG19 without batch normalization, however the false positives remain on the level of a pure VGG19 model.

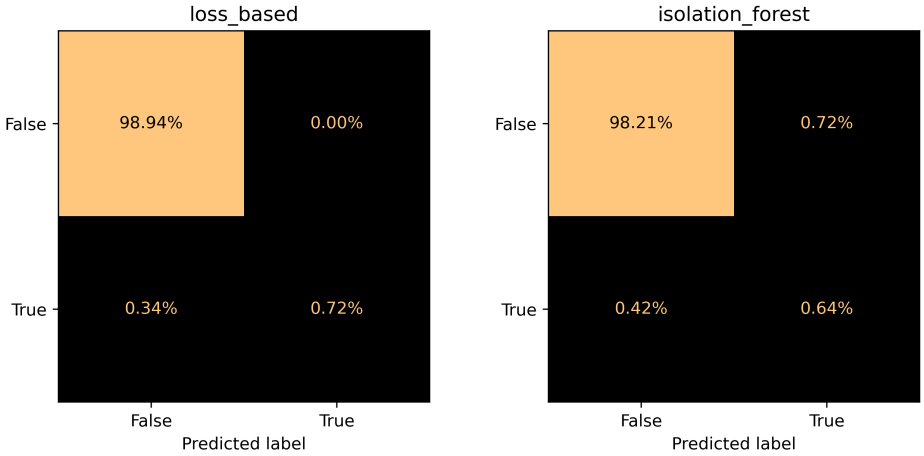


Figure 19: Confusion Matrices of the ResNet50 Encoder

5.4 Encoding via EfficientNetV2L

Observing Figure 20, the learning curves of the EfficientNetV2L model, there is a change in the patterns seen among the other models. The curves are comparable in terms of general characteristics, the starting and resulting losses are comparable, and the majority is reduced in the first few epochs. In detail however the variance of the validation losses do not show a further increase, it steadily follows the training curve. Also, for this model there is no overfitting or any training issue detected, the training data seem to represent the whole dataset, this is supported by the validation curve.

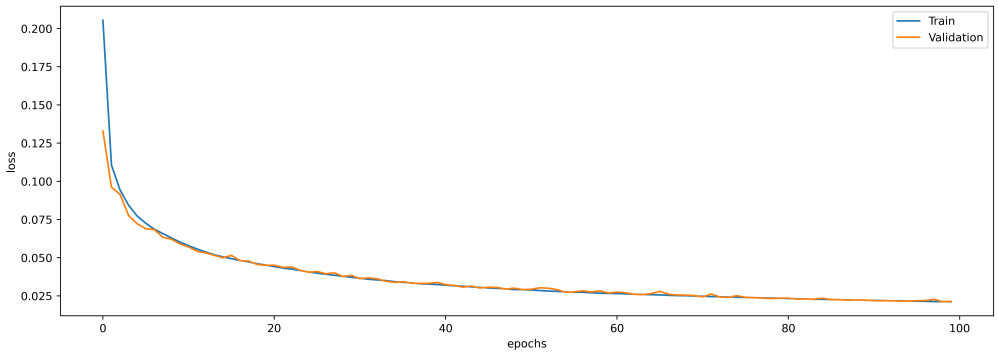


Figure 20: Learning curve of the EfficientNetV2L Encoder

The predicted images are similar to the ones seen before, the basic features are well grabbed, and the details are focusing around the center or bright part of the images. The rail itself and ballast stones can be easily recognized, the model (same as for the other models) captures the edges easily. The reconstructed images however still not deliver full pictures, loss of information due to incompleteness is assumed. The sample images are presented on Figure 21

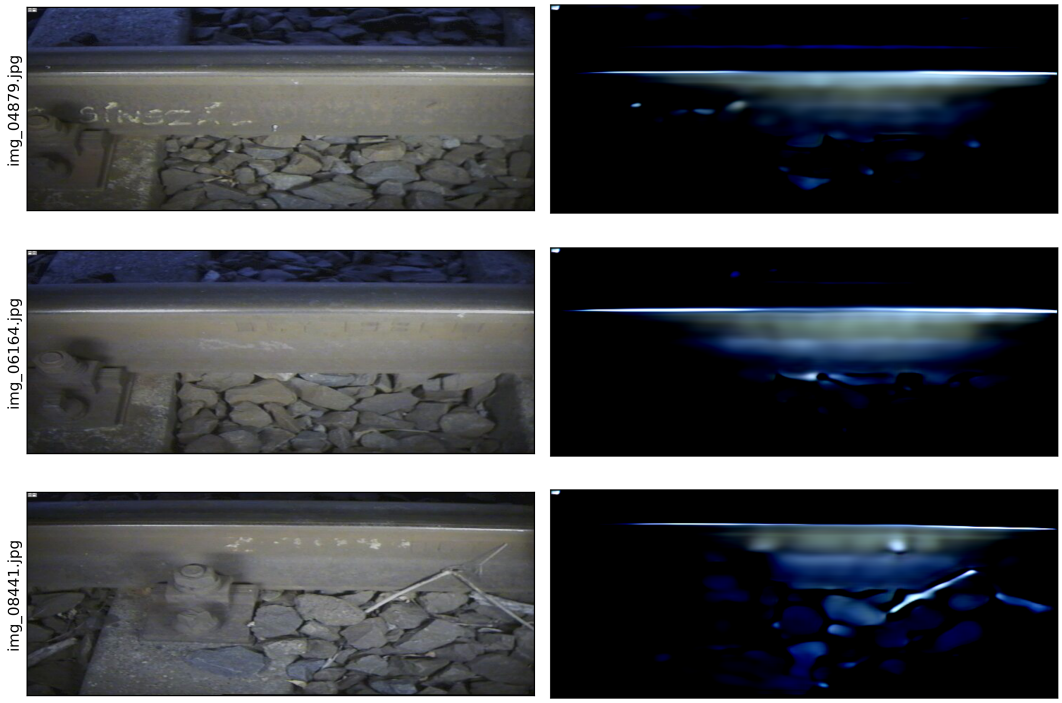


Figure 21: Predicted images in case of EfficientNetV2L Encoder

Reducing the dimensions to a plane results comparable characteristic behavior as the other models as shown on Figure 22. Although there are some differences that need to be noted. First, the outliers are captured and separated from the *normal* cluster, especially on the bottom right part of the encoded view, however the separation is not as clear as was for example in case of the ResNet50 Encoder. Secondly the outliers that are stuck on the top right part of the encoded view inside the *normal* cluster are now moved more to the boundary and even the *normal* cluster is opened up better. After decoding back the images, the resulting feature plane shows that the outliers are placed back more concentrated to their original place indicating a certain loss of information and generalization of the model.

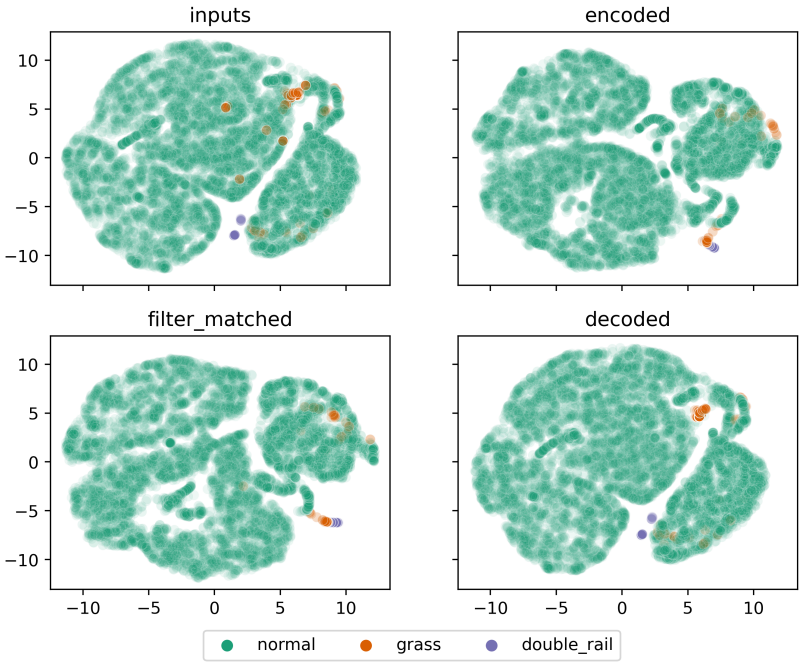


Figure 22: PCA / t-SNE visualization of the EfficientNetV2L Encoder

The loss values of the full dataset is shown on Figure 23. Once again the same characteristics can be seen as for the other models. The loss metric alone is not necessarily indicates the performance of these anomaly detectors. Most of the outliers can be found by comparing the loss values with a given threshold, however some of those remain embedded and can not be retained with this method.

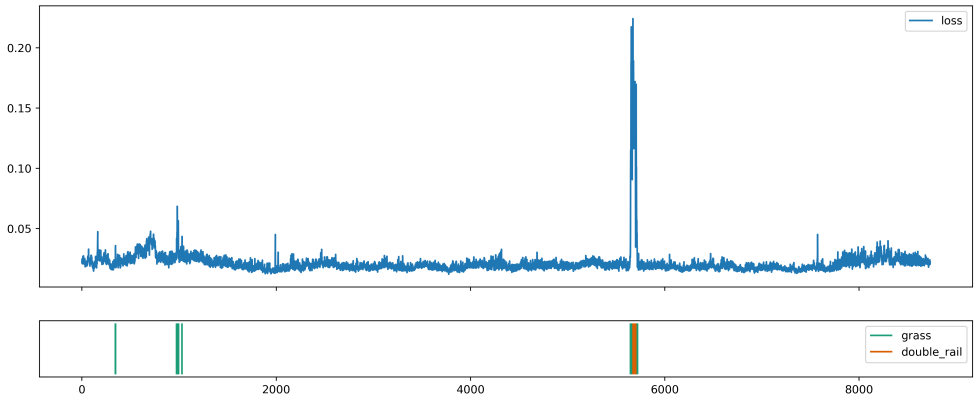


Figure 23: Loss values of the dataset with EfficientNetV2L encoding

The classification metrics as shown with the confusion matrices on Figure 24 present a similar performance of the loss-based approach, a little more than ⅔ of the outliers found. The Isolation Forest delivers one of the best true positive indications, but on the cost of the worst false negative rate.

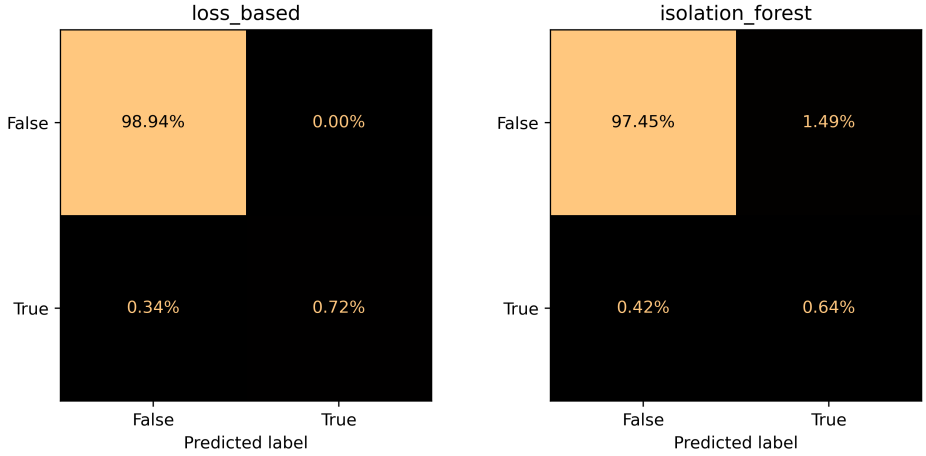


Figure 24: Confusion Matrices of the EfficientNetV2L Encoder

6 Discussion

6.1 Learning curves

As already observed in Section 5, there are clear differences between the learning curves of the different encoder models. To examine these differences the learning curves and their rolling standard deviation of the *train* and *validation* datasets are shown on Figure 25 and 26 respectively. The rolling calculation is done with a window of 10 epochs.

Direct comparison of the models and their performances can not be done as the training was done with the same parameter setting and no hyperparameter optimization was done. Current remarks serve only to understand the model behavior and outline possible future improvement options.

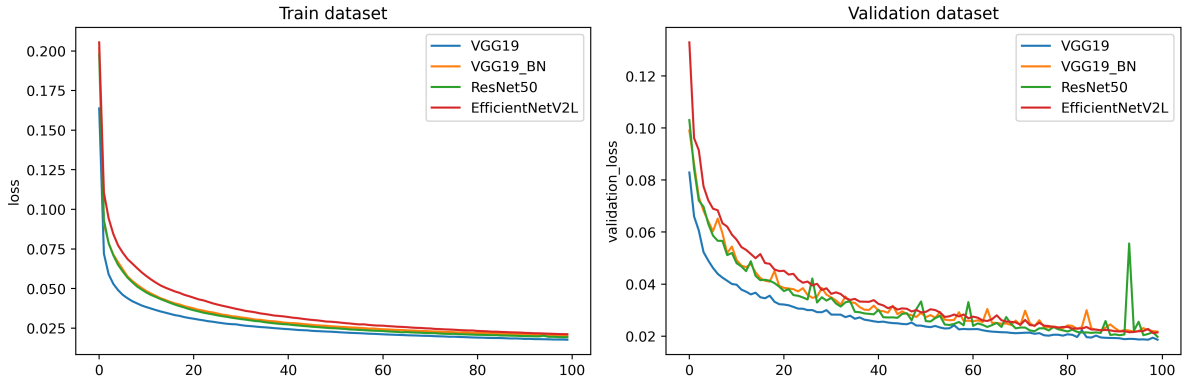


Figure 25: Learning curve comparison of different encoders

The learning curves on the *train* dataset are quite comparable, the main differences are the starting and ending value and the gradient over the epochs. As the starting loss heavily depend on the initialization of the starting weights, an option would be to set these values based on a profound approach in the future. The resulting value is quite close for each model, reaching a value around $0.02 - 0.025$. Similar behavior is observed on the *validation* dataset, the noticeable difference is the gradient of the curve, which in this case shows a less smooth evolution.

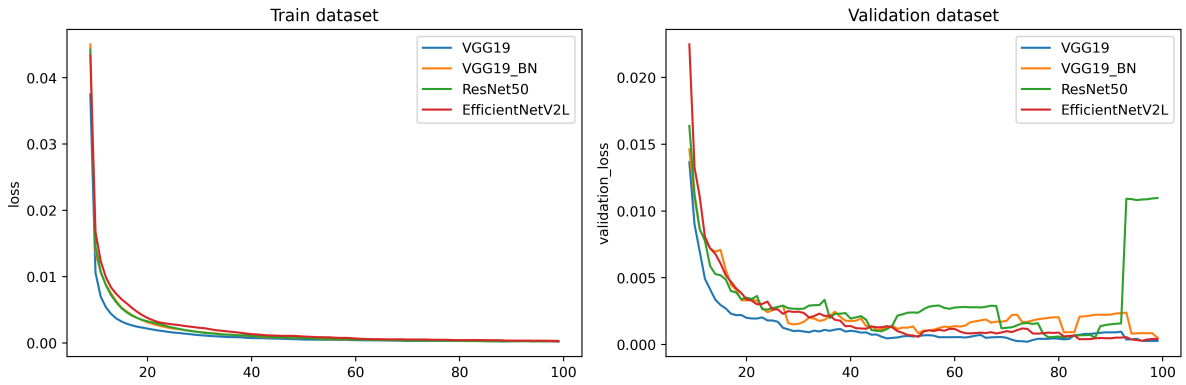


Figure 26: Standard deviation of learning curves

The standard deviation of the learning curves also indicate key characteristics. In case of the *train* dataset the variation over the epochs drop rapidly and converges towards zero. The question can be risen whether further training would improve the model as most of the autoencoded images are incomplete. It might happen that the models are not complex enough to learn all the characteristics of the images, on the other hand it could also happen that the changes of the weights, due to their high number, average out the changes, so the optimizer can further tune the model, but this can not be seen on the loss values clearly. This raises the importance of introducing further monitoring measures or different loss definitions.

The curves of the *validation* dataset confirm our observations that there is a clear difference of the variance. The most stable is the VGG19 model, whilst the largest range is obtained by the ResNet50 model. With adaptation of the learning rate this could be further optimized.

6.2 Planar visualization

The PCA combined with t-SNE resulted in a visualization that is able to trace the changing of the images as passing through the model. Normally all input phase representation should be the same (as we are visualizing the same unprocessed data), however due to the stochastic nature of the t-SNE there are slight deviation among the different representations provided.

Independently of this similar structure is grabbed on the input data forming three major clusters of the *normal* images and the outliers are embedded in all cases at the same location. Afterwards encoding the same behavior is observed with some differences in the details. First the outliers started to move out and separate out from the main clusters at the top right and bottom left part. At the bottom left part the separation was quite successful, on the top right part the models behaved differently, the more complex models resulted in better performance. The VGG19 retained these top right outliers still in the main cluster, whilst the EfficientNetV2L has remarkably opened up the cluster and moved these images almost to the outside of the *normal* cluster. Even the three major *normal* clusters were separated better, and further internal structure is suggested.

The matching of the number of filters via the additional convolutional layers did not introduce any significant change to the planar visualization. However, use of these layers are not mandatory and shall be investigated at a later step.

The decoding capability of the models are very impressive, the reduced dimensions are reverted almost to the original representation that was seen during input phase. This indicates that the Autoencoder model realizes its main function.

One further exploration was done using this visualization, that is, investigating whether the sequence of the images can be captured on this feature plane. This is shown on Figure 27, that colors the markers in the sequence as it can be seen in the video footage. The visualization in this case was done using the EfficientNetV2L model.

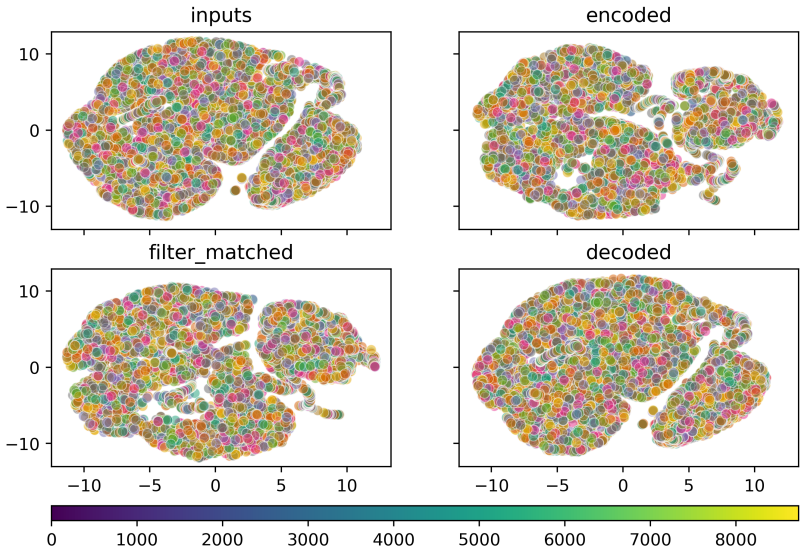


Figure 27: PCA / t-SNE representation of the sequence of the video images

6.3 Classification metrics

The confusion matrices are summarized in Table 3. This already shows that the performance of the loss-based anomaly detector is very stable, but as stated before this happens due to the selection of the threshold value. The Isolation Forest results highly depend on the structure of the model and resulting feature vectors. The calculated classification metrics are shown in Table 4, where LB refers to loss-based and IF refers to Isolation Forest method.

Encoder type	True values	Loss-based		Isolation Forest	
VGG19	Positive	31	62	45	48
	Negative	8640	0	8578	62
VGG19 BN	Positive	30	63	41	52
	Negative	8640	0	8626	14
ResNet50	Positive	30	63	37	56
	Negative	8640	0	8577	63
EfficientNetV2L	Positive	30	63	37	56
	Negative	8640	0	8510	130
		False	True	False	True
		Predictions		Predictions	

Table 3: Confusion matrices of different encoders

The most important metrics in the actual use case is the *Recall*, as it measures the proportion of outliers found by the algorithm. In this metric the loss-based approaches perform better than the Isolation Forest methods. The *Specificity* measures the proportion of negative hits, that is the *normal* rail in this case. As expected, due to the heavily imbalanced dataset, this is always close to 100%. The *Precision* measures if there is a positive prediction what is the probability that it really is. Due to the threshold setting this is always maximal in case of the loss-based approaches. The two general metrics *F1 score* and *Balanced accuracy* indicates the overall performance of the models. The first metric do not consider true negative values, therefore in our case it is more sensitive to the performance shown on the positive predictions and true positive. The latter one balances between true positives and true negatives. To judge the overall performance the *F1 score* and the *Recall* shall be considered simultaneously. Not finding a true positive might increase transportation risks arising from bad track condition, however providing too much false positives increases the cost of the maintenance (at least control monitoring) that needs to be done, therefore due to economic reasons shall be reduced.

Based on the selected metrics the best performance is reached by the loss-based approaches, however still not reaching an acceptable rate of *Recall*.

	VGG19		VGG19 BN		ResNet50		EfficientNetV2L	
Metrics	LB	IF	LB	IF	LB	IF	LB	IF
Positives	93	93	93	93	93	93	93	93
Negatives	8640	8640	8640	8640	8640	8640	8640	8640
True positives	62	48	63	52	63	56	63	56
True negatives	8640	8578	8640	8626	8640	8577	8640	8510
False positives	0	62	0	14	0	63	0	130
False negatives	31	45	30	41	30	37	30	37
Recall (Sens.)	66.67%	51.61%	67.74%	55.91%	67.74%	60.22%	67.74%	60.22%
Specificity	100.00%	99.28%	100.00%	99.84%	100.00%	99.27%	100.00%	98.50%
Precision	100.00%	43.64%	100.00%	78.79%	100.00%	47.06%	100.00%	30.11%
F1 score	80.00%	47.29%	80.77%	65.41%	80.77%	52.83%	80.77%	40.14%
Balanced acc.	83.33%	75.45%	83.87%	77.88%	83.87%	79.74%	83.87%	79.36%

Table 4: Classification metrics of different encoders

These results indicate that the fine-tuning of the loss threshold might improve the performance of the anomaly detector models. A simulation of the threshold value is shown on Figure 28. The threshold is defined as the multiplication of the standard deviation of the losses of the images. The limit value is the loss mean and the multiplied standard deviation added. Above this value the image is considered as outlier. The optimum in terms of *F1 score* is between 2.07 and 2.24 times the standard deviation added to the mean of losses. Such optimum gives a balance between precision and recall, indicating how well the anomalies found, quantifying how much is found considering the false negative hits. Further optimization possibilities, more decent modeling, more refined loss calculation is possible to enhance the performance of the models.

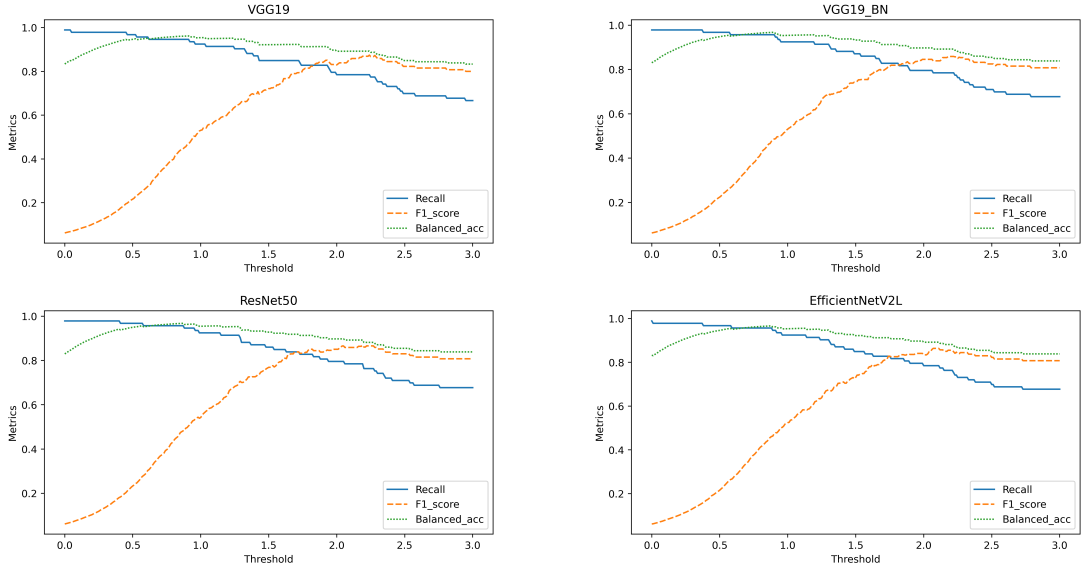


Figure 28: Simulation of loss-based thresholds

6.4 Detected anomalies

To understand the learning capability of the Autoencoder, together with the anomaly detection performance a visual analysis of the predicted or misclassified outliers is done. The threshold for the loss-based approach was set to 2.15 for all models. A comparison of identified outliers is shown on Figure 29.

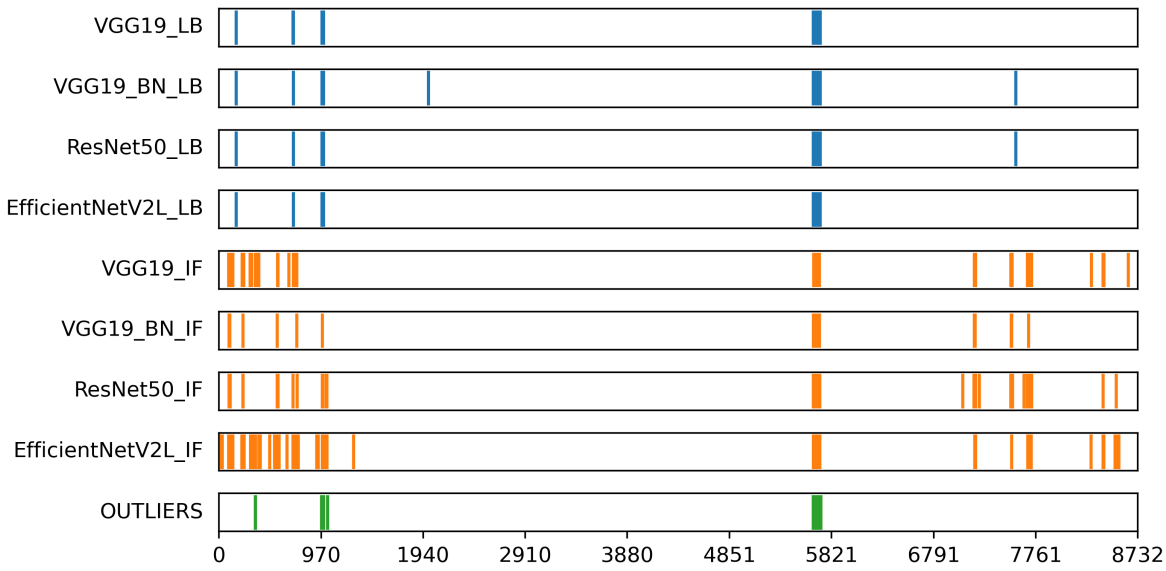
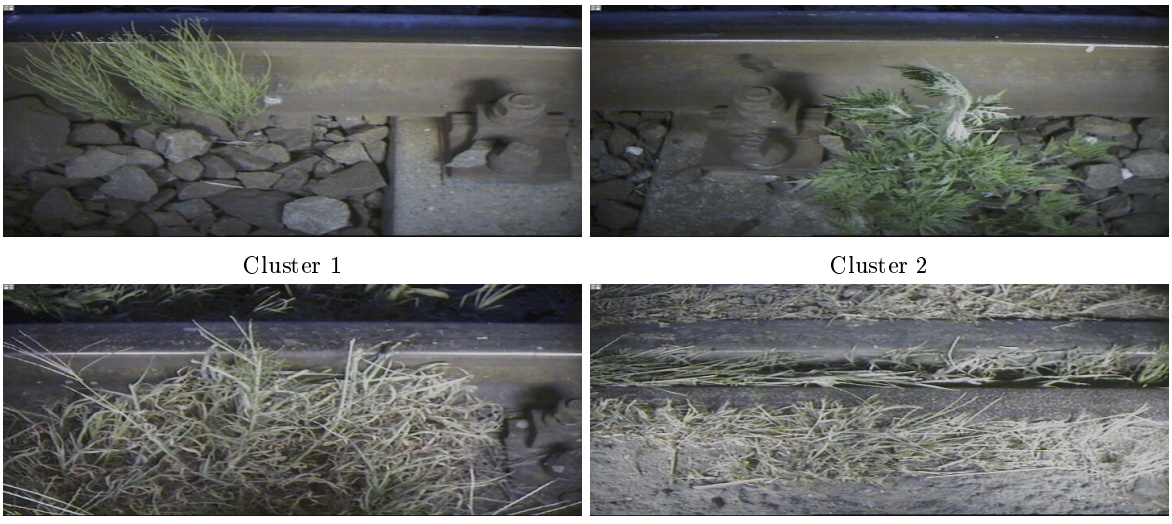


Figure 29: Comparison of identified outliers of different models

The loss-based approach concentrates clearly on the clusters on the three main clusters of the outliers, examples are shown on Figure 30. From a practical perspective, this already narrows down the search that has to be done on a full video footage. The Isolation Forest delivers also a good marking, almost all captures the three main clusters, except the *VGG19*, that misses the second cluster. On the other hand, this approach delivers much higher rate of false positives.

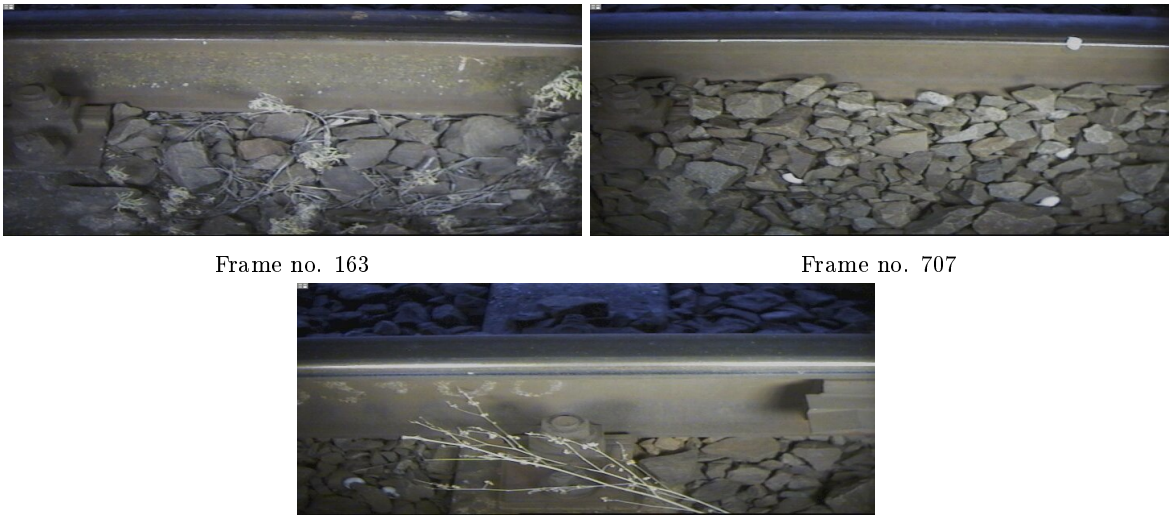


Cluster 1 Cluster 2

Cluster 3 Cluster 3

Figure 30: Examples of the three outlier clusters

From Figure 29 it can be seen that there are three false positive indications resulting from almost all models at frame number 163, 707 and 7573. These are presented on Figure 31. The samples indicate the presence of some grass or significant amount of ballast rocks covering even the sleepers. Deeper investigations on the reproduced output images reveal that the Autoencoder is very sensitive to the presence of edges (please refer for example to Figure 21). Such edges are resulting mostly from grass, therefore a small amount might be enough to increase the probability of an outlier indication. On the other hand, the lack of sleeper is removing some features from the decoded images, providing a more homogeneous image, that might be misinterpreted by the anomaly detector.



Frame no. 163 Frame no. 707

Frame no. 7573

Figure 31: Remarkable false positives

7 Conclusion

In current study an application of an Autoencoder as anomaly detector is presented. The model is trained and evaluated of real life sample data provided by MÁV CRTI Ltd. Four different models were built depending on the applied Encoder. The anomaly detection is performed based on the loss values defined between input and output images and based on the Isolation Forest algorithm applied to the bottleneck vectors. Performance evaluation covered visual check of the outliers, evaluation of classification metrics, that was granted via manual annotation of the sample data. The performance of the model is visualized via application of PCA and t-SNE algorithms that provided an insight view how features are translated through the models.

The results highlighted that such models can be very effective on finding outliers in a given dataset, however the following remarks have to be noted for the sake of completeness:

- The sample dataset is limited in terms of variation, the video footage shows mostly the same side view of a *normal* rail with two type of outliers: *grass covered* and presence of *double rails* which are very seldom found.
- The outliers differ from the normal images significantly and are concentrated to three main clusters in the dataset.
- There was no preprocessing of the images except resizing and normalization.
- The training of the neural networks is limited, there was no hyperparameter optimization done.
- The anomaly detection methods were also not optimized for the dataset.

Even though of these limitations the results of the models are remarkable, especially the loss-based method provided good results: an indication of the three main outlier clusters. In current use case it is vital to identify as many outliers as possible to ensure railway operation safety. On the other hand it is economically reasonable to limit the false positives to a minimum to ensure efficient review of the results. However, in real life application the task is not to identify grass near the rails or double rail construction, much finer outliers need to be detected: surface issues of the rail running surface or missing parts of the rail fastening. This highlights the possible limitations of the current model. On the other hand this draws the attention to additional use cases coupled to this challenge. The model might be able to classify the images to main rail construction groups, for example separate the images to normal rails, turnouts, bridges, ballast covered track, etc. Later on a more refined model can be used on these groups for anomaly detection.

The hope is not yet lost due to the many options present to improve our model. A shortlist of these improvements is listed below:

- Apply image preprocessing, like CLAHE histogram equalization to boost the learning capability of the neural networks.
- Optimize parameters of the neural network to fit to given problem and dataset.
- Besides classification, segment the images and detect major elements of the track, such as rails, fastening elements to limit the anomaly detection to major parts.
- Introduce further anomaly detector algorithms, for example One Class SVM or apply deep learning methods to obtain better kernels.

- Loss calculation can be refined to match better to the input and output images.

Before venturing to further improvements the opportunity is given to understand deeper how the model works. We have seen characteristic changes on the PCA / t-SNE representation. This representation can be further tuned and extended with different approaches to visualize behavior of the dataset de- and recomposition capability of the model. Furthermore, analysis of the latent space is not yet exhausted, introduction of pairwise distance calculation might reveal more information on the abstract representation of the images. And last but not least, as further video footage is available we can obtain additional information from real life videos that might deliver further information and guidance how the problem could be tackled by machine learning approach.

List of Figures

1 Results of first classification approach on the Kaggle dataset . . . 6
2 Railway inspection vehicles of MÁV CRTI Ltd. . . . 7
3 General structure of Autoencoders [7] . . . 7
4 Sample images from the dataset . . . 9
5 Learning curve of the VGG19 Encoder . . . 12
6 Predicted images in case of VGG19 Encoder . . . 12
7 PCA / t-SNE visualization of the VGG19 Encoder . . . 13
8 Loss values of the dataset with VGG19 encoding . . . 14
9 Confusion Matrices of the VGG19 Encoder . . . 14
10 Learning curve of the VGG19 (BN) Encoder . . . 15
11 Predicted images in case of VGG19 (BN) Encoder . . . 15
12 PCA / t-SNE visualization of the VGG19 (BN) Encoder . . . 16
13 Loss values of the dataset with VGG19 (BN) encoding . . . 16
14 Confusion Matrices of the VGG19 (BN) Encoder . . . 17
15 Learning curve of the ResNet50 Encoder . . . 17
16 Predicted images in case of ResNet50 Encoder . . . 18
17 PCA / t-SNE visualization of the ResNet50 Encoder . . . 18
18 Loss values of the dataset with ResNet50 encoding . . . 19
19 Confusion Matrices of the ResNet50 Encoder . . . 19
20 Learning curve of the EfficientNetV2L Encoder . . . 20
21 Predicted images in case of EfficientNetV2L Encoder . . . 20
22 PCA / t-SNE visualization of the EfficientNetV2L Encoder . . . 21
23 Loss values of the dataset with EfficientNetV2L encoding . . . 21
24 Confusion Matrices of the EfficientNetV2L Encoder . . . 22
25 Learning curve comparison of different encoders . . . 22
26 Standard deviation of learning curves . . . 23
27 PCA / t-SNE representation of the sequence of the video images . . . 24
28 Simulation of loss-based thresholds . . . 25
29 Comparison of identified outliers of different models . . . 26
30 Examples of the three outlier clusters . . . 26
31 Remarkable false positives . . . 27

List of Tables

1 Dataset obtained from sample video 9
2 Latent space shape of different Encoders 10
3 Confusion matrices of different encoders 24
4 Classification metrics of different encoders 25

References

[1] Tamás Demus. *Mathemathical Modeling Practice - Railway track defect detection*. original-date: 2022-12-02T20:47:30Z. Dec. 2022. URL: https://github.com/demustamas/Mathemathical-Modeling-Practice (visited on 05/23/2023).

[2] *AI&ML Training – AI Research Group*. en-US. URL: https://ai.elte.hu/training/ (visited on 05/23/2023).

[3] *AI Research Group – Artificial Intelligence & Data Science*. en-US. URL: https://ai.elte.hu/ (visited on 05/23/2023).

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

[5] *MÁV Központi Felépítményvizsgáló Kft.* URL: http://www.mavkfv.hu/index.php?lngchg=en&f= (visited on 05/23/2023).

[6] *The MÁV Central Rail and Track Inspection Ltd. 25 years*. 2021. URL: http://www.mavkfv.hu/index.php?f=kfv25.

[7] Ritwek Khosla. *Auto-Encoders for Computer Vision: An Endless world of Possibilities*. en. Jan. 2021. URL: https://www.analyticsvidhya.com/blog/2021/01/auto-encoders-for-computer-vision-an-endless-world-of-possibilities/ (visited on 05/24/2023).

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[9] *Autoencoder*. en. Page Version ID: 1153796242. May 2023. URL: https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=1153796242 (visited on 05/24/2023).

[10] *Anomaly detection*. en. Page Version ID: 1153438169. May 2023. URL: https://en.wikipedia.org/w/index.php?title=Anomaly_detection&oldid=1153438169 (visited on 05/26/2023).

[11] *2.7. Novelty and Outlier Detection*. en. URL: https://scikit-learn/stable/modules/outlier_detection.html (visited on 06/03/2023).

[12] *2.3. Clustering*. en. URL: https://scikit-learn/stable/modules/clustering.html (visited on 06/03/2023).

[13] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Tech. rep. arXiv:1409.1556 [cs] type: article. arXiv, Apr. 2015. URL: http://arxiv.org/abs/1409.1556 (visited on 05/26/2023).

[14] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Tech. rep. arXiv:1512.03385 [cs] type: article. arXiv, Dec. 2015. DOI: 10.48550/arXiv.1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 05/26/2023).

[15] Mingxing Tan and Quoc V. Le. *EfficientNetV2: Smaller Models and Faster Training*. Tech. rep. arXiv:2104.00298 [cs] type: article. arXiv, June 2021. DOI: 10.48550/arXiv.2104.00298. URL: http://arxiv.org/abs/2104.00298 (visited on 05/26/2023).

[16] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. Pisa, Italy: IEEE, Dec. 2008, pp. 413–422. ISBN: 9780769535029. DOI: 10.1109/ICDM.2008.17. URL: http://ieeexplore.ieee.org/document/4781136/ (visited on 06/03/2023).