Railway Track Fault Detection
Project Work

Tamás DEMUS
XP4B9D

Spring Semester 2022/2023

Abstract

Following project is a continuation of the Mathematic Modeling Practice subject. During the first semester a set of convolutional networks were built to compare on the task of classifying different railway track faults. The project was followed up by contacting MÁV Central Rail And Track Inspection Ltd. who has provided sample dataset for further research and study purposes. This dataset is limited in terms of track failures however it provides the opportunity to apply anomaly detection models. The sample contains video footage of a short section of a single track of approx. 3 minutes, with a few seconds of rail sections covered with grass and/or containing double tracks. The latter two is considered as outlier from the dataset. A set of autoencoder models built to detect these outliers in the sample. The autoencoder is based on the convolutional models of VGG19, ResNet50 and EfficientNetV2L. Different anomaly detection methods were applied, an approach based on the calculation of a loss measure between the input and the output of the autoencoder and IsolationForest algorithm applied on the feature space of the inputs generated by the encoder part.

Acknowledgement

I would like to express my deepest gratitue 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 Artifical 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 footages 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 obsessesion.

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

Contents

1 Introduction

1.1 Previous work

Current research started as part of the studies on the Mathematics Expert in Data Analytics and Machine Learning postgraduate specialization program [1] held by the AI Research Group of Eötvös Loránd University [2]. During the one year program a thesis work has to be created that is often preceeded 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 problem.

1.2 Available results

The work of the first semester can be found at [3]. Following is a short summary of the results obtained.

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

The dataset was taken from the Kaggle webpage [4] consisting of a limited number of images with defective and non-defective railway tracks. The images combine a high variety of failures with very limited examples leading to a very specific dataset where the provided validation and test split is not representing the initial set of images.

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.

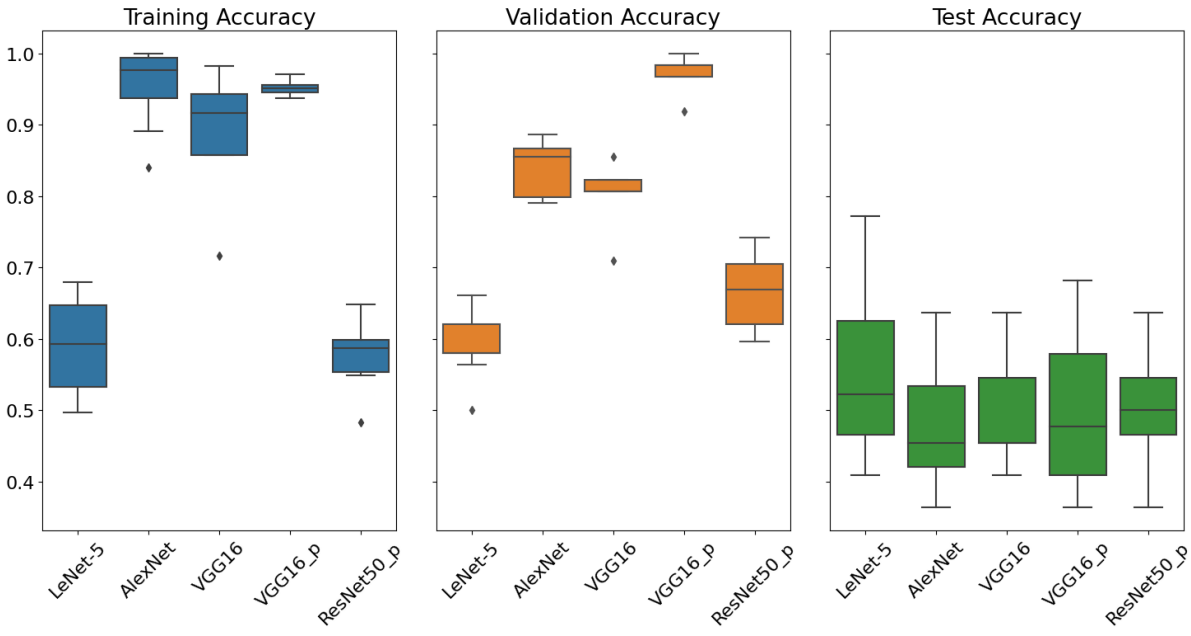

Figure 1: Results of first classification approach on the Kaggle dataset

Most of the models were 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 achieveing 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.

1.3 MÁV CRTI Ltd.

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 classifier model.

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 ralway 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 measurment equipment and inspection vehicles, Sstarting 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 reflect the importance of such approach, as today visual inspection demands heavy efforts let that be the work done by the maintenance personnal (intrinsically walking along the tracks) or the monitoring of the video footages 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.

1.4 Modeling 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. 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?

1.5 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 7.

2 The dataset

The video system of the SDS vehicle records both rails from two angles resulting in four video footages 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. Some examples of the images extracted from the video is shown on Figure 3.

A single footage sample video of approximately 3 minutes was provided 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 of showing a double rail section.

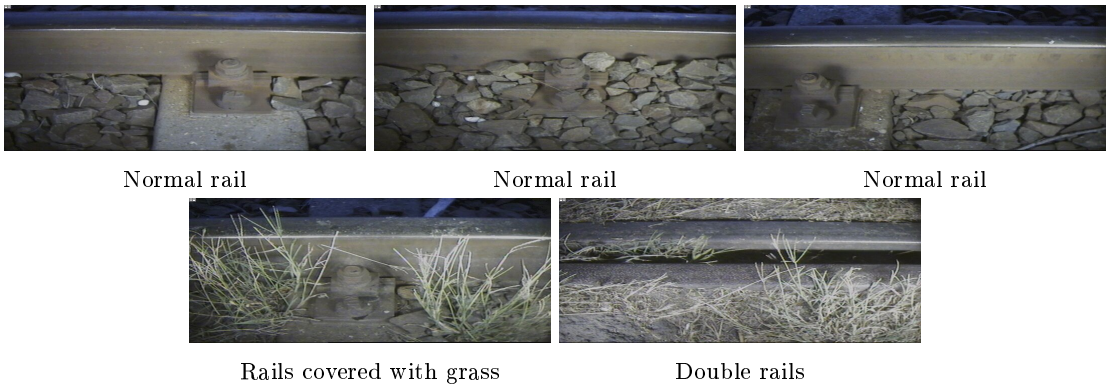

Figure 3: Sample images from the dataset

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 performance evaluation possibility.

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

There is no slurring observed on the images that is remarkable considering that the video is taken with a vehicle speed up to 80 to 100 kilometer per hour. However a slight fisheye distortion can be seen that is noticeable mostly in the rail itself, as it is not tend to follow a stright line, a small bending effect is given. Also the lighting of the pictures result in a brighter spot on 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.

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 (please see Section 3).

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, evaluating or tuning the models.

3 Description of the model

3.1 Autoencoders

Autoencoders are widely used in Deep Learning applications with the main intention the describe a set of data, mostly targeting dimensionality reduction of creating a general represenation of the dataset. This allows the application of such models for anomaly detection, that can be translated as outlier detection problem in this generated representation, called feature vector or latent space. Such approach does not require annotated dataset, therefore belongs to the class of unsupervised learning algorithms.

The general structure of Autoencoders are shown in Figure 4.

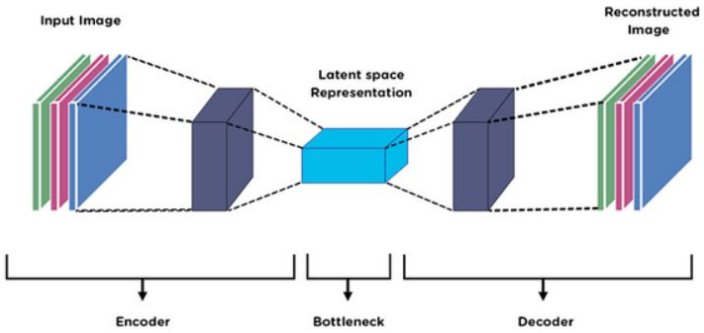

Figure 4: General structure of Autoencoders [7]

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 represenation 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.

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

3.2 Type of Encoders

During the research four type of Encoders used as listed below.

- VGG19
- VGG19 with batch normalization
- ResNet50
- EfficientNetV2L

The VGG19 is one of the first deep convolutional networks introduced in [10]. In the second model batch normalization was added after each convolutional layer. The ResNet50 [11] overcomes the issue of the wanishing gradient by adding the skip connections. EfficientNetV2 [12] 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

3.3 Applied Decoder

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 is only the number of filters, the shape is adjusted via addional convolutional layers:

- 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.

3.4 Basis of anomaly detection

The main question of anomaly detection is how far can we distinguish between *normal* and *abnormal* data vector, that is in this case an image or a feature representation. There are several approaches (a short list can be found at [13]), among those we applied the replication neural network, more in particular an Autoencoder approach. This brings us to two main approaches that we followed.

The first one is to use the decoding capability of the network and define the outliers of the dataset by the calculated loss. The assumption behind is that during training the model can learn and generalize to the input images that are *normal*, but can not learn the few outliers, therefore the calculated loss shall be higher in latter case as the *abnormal* part of the image can not be recreated so well. The definition of the loss (or distance between original and replicated image) plays an important role in the detection.

In current approach the loss function is set as the pixelwise mean squared error of the input and the 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.

The second approach that we utilize the representation of the images and trace back the outlier detection to the latent space. The represenation created by the encoder shall bear all the key characteristics of the images, if an image contains a certain defect (and thus becomes an outlier), then it's representation shall be an outlier as well. This method allows to introduce several tools for detecting the outlier vector, such as Isolation Forest or any basic clustering methods (DBSCAN, OPTICS, etc.).

Our research focused on the Isolation Forest, that was applied on the latent space with default settings.

3.5 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 pixelwise mean squared loss between the original and replicated images. Adam optimizer is applied with a learning rate of $5e-6$. The batch size for training is set to 8.

The dataset is splitted 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).

3.6 Performance evaluation

The first approach on the performance evaluation is the general classification metrics based on the confusion matrices of each approach. Furthermore a visualisation 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. This allow a visualisation how outliers behave over the whole model.

4 Software implementation

The project is written in Python language partially in python sripts 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 modelfiles and the original dataset is not uploaded to GitHub. The basic folder and file structure is as follows:

1. ./build - Build directory of LaTeX, contains project documentation
2. ./data - Contains the dataset (not available on GitHub)
3. ./results - Contains resulting models, images, tables
4. ./tex_content - LaTeXdocumentation files
5. ./tex_images - LaTeXimages for documentation
6. ./tex_refs - Style files, bibliography
7. ./toolkit - Contains main scripts
8. ./toolkit/classes.py - Class definitions for loading the dataset
9. ./toolkit/pytorch_tools.py - Class definitions for Pytorch
10. ./project_work.ipynb - Jupyter notebook project file
11. ./project_work.tex - Main LaTeXdocumentation file
12. ./project_work_pres.tex - Final presentation of the project
13. ./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 infratsructure. 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 curve 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. The validation curve is slightly above the train losses, there is no sign of overfitting or abnormal training of the model.

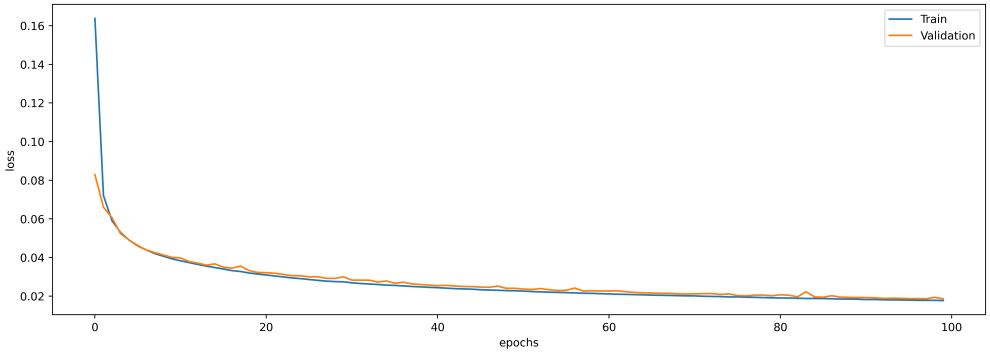

Figure 5: Learning curve of the VGG19 Encoder

After training some example images are predicted (encoded and decoded) to visualise 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. The details that are close to the center of the image or are brighter parts of the image are more likely to be captured.

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 (less transparent) the marker is. The outlier images form a certain cluster already among the input images, that is, at least to some extent, retained after encoding. This cluster are separated partially after encoding, there is a clear cluster made of the outliers, but this cluster does not contain all members, some are still embedded among the *normal* images, however slightly moved to the boundary of it. It can be also seen that after encoding 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 readjust this clustering back to it's origin, basically proving that the main idea of the Autoencoders is represented in the model.

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 indicate 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 it's neighbouring images. Similarly, but in the opposite direction, at some *normal* pictures the loss value is significantly higher then the preceeding or following images.

6 Discussion

7 Conclusion

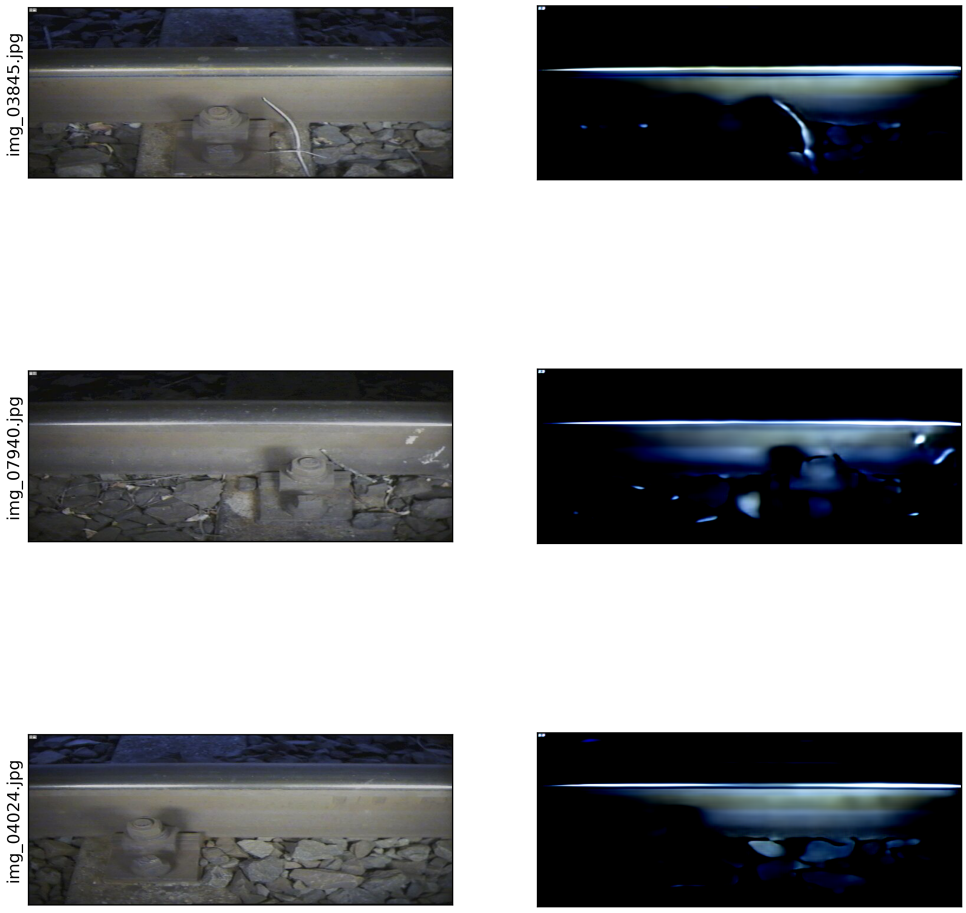

Figure 6: Predicted images in case of VGG19 Encoder

List of Figures

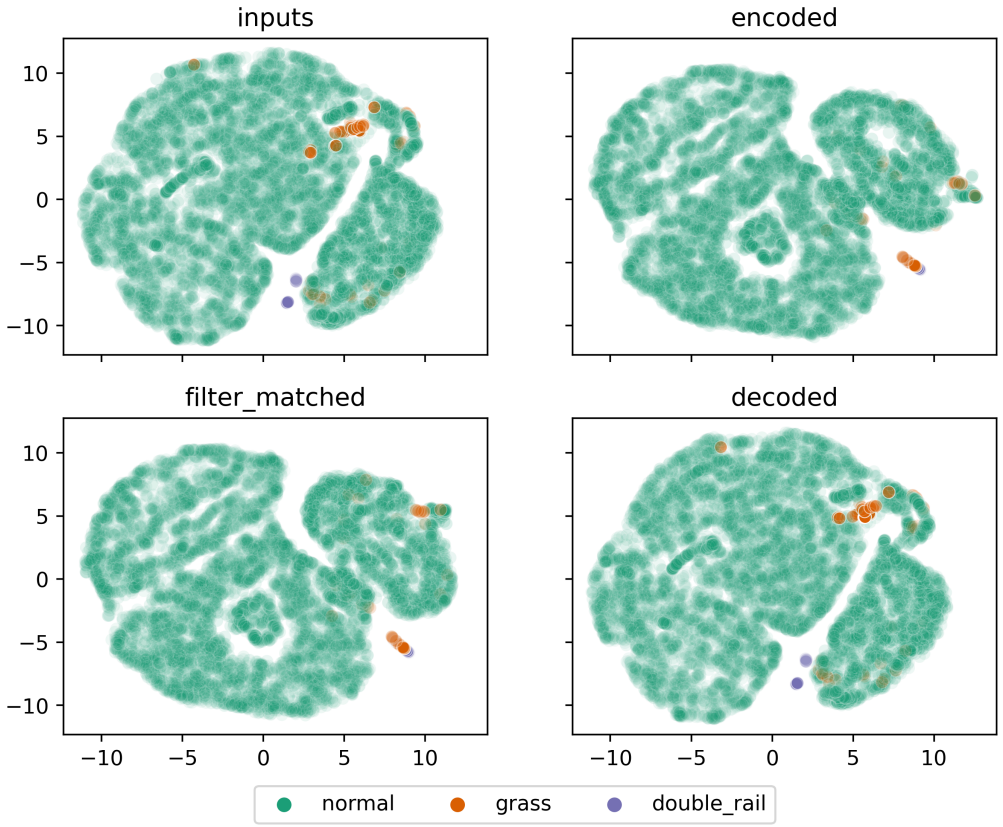

Figure 7: PCA / t-SNE visualization of the VGG19 Encoder

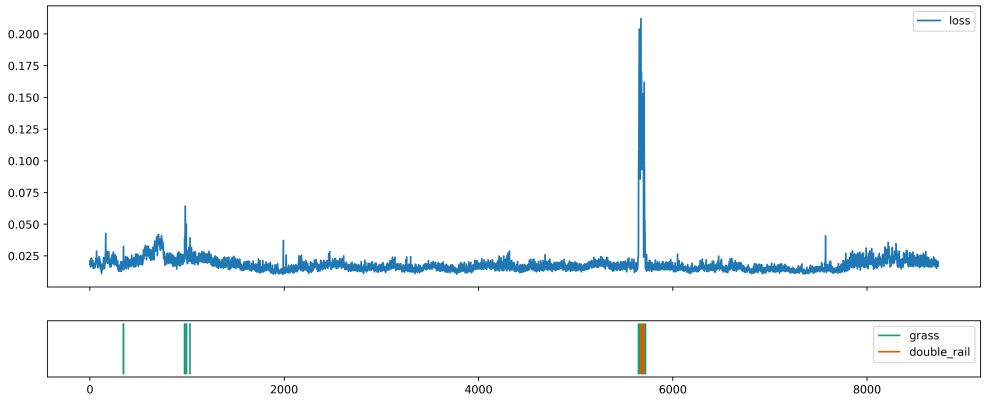

Figure 8: Loss values of the dataset with VGG19 encoding

List of Tables

References

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

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

[3] 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).

[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] 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).

[11] 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).

[12] 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).

[13] *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).