Railway Obstacle Detection Using Unsupervised Learning: An Exploratory Study

Boussik Amine, Wael Ben Messaoud, Smail Niar, Abdelmalik Taleb-Ahmed

Railway Obstacle Detection Using Unsupervised Learning: An Exploratory Study

Amine Boussik[1], Wael Ben-Messaoud[1], Smail Niar[2], Abdelmalik Taleb-Ahmed[2]

***Abstract*— Autonomous Driving (AD) systems are heavily reliant on supervised models. In these approaches, a model is trained to detect only a predefined number of obstacles. However, for applications like railway obstacle detection, the training dataset is limited and not all possible obstacle classes are known beforehand. For such safety-critical applications, this situation is problematic and could limit the performance of obstacle detection in autonomous trains. In this paper, we propose an exploratory study using unsupervised models based on a large set of generated convolutional autoencoder models to detect obstacles on railway's track level. The study was conducted based on three components: loss functions, activations and optimizers. Existing works rely on fixing thresholds to judge the performance of the model. We propose instead a methodology based on Multi-Criteria Decision Making (MCDM) to evaluate the performance of all models. Furthermore, we introduce the notion of gap-score to evaluate each model by calculating the average difference between the reconstruction score on images with and without obstacles. The aim is to find models maximizing the average of gap-scores and rank them according to their performances. Experimental results show that the evaluated models can provide up to 68% average gap-score.**

I. INTRODUCTION

Obstacle detection is still one of the most challenging and critical tasks in computer vision. From autonomous cars [3] [2], to autonomous trains [23] [22] [24], obstacle detection plays an important role to enhance the perception of the autonomous system. This task is of utmost priority, especially for environment monitoring in autonomous trains. Thanks to the ever growing deep learning methods and computing power of embedded systems, the monitoring of the environment in such systems have become reasonably easy to handle.

In this paper, we exploit deep learning approaches in order to detect obstacles for railway applications. We focus only on the track level using mounted RGB cameras on the train. Due to the scarcity of works in terms of railway obstacle detection using deep learning methods and observing that the majority of works are leaning towards the use of supervised methods such as object detectors [4], [6] [5] or segmentation models [7], we aim to explore the use of unsupervised learning to solve this problem, especially the use of convolutional autoencoders.

This choice is driven by three factors: First, as aforementioned, a scarcity of works exploiting unsupervised methods in railway obstacle detection is noticeable with the only exception of the recently published [8]. Second, the use of supervised methods requires previous knowledge about the obstacles, hence enumerating all possible obstacles that we may encounter and labeling them to create classes which is nearly impossible in a real-life scenario. Third, to the best of the authors knowledge, railway-oriented datasets describing real-world obstacles are few to non-existent, but on the other hand, normal data without obstacles are available.

We propose in this paper an exploratory study using convolutional autoencoder to discriminate, at frame level, between normal and anomalous images using reconstruction score. In our study, we mainly focus on the following three components: Activation functions, loss functions and optimizers in order to find the best model that is able to detect anomalies on railway's track level.

In our work, we generate 240 models following the combination of 5 loss functions, 8 activation functions and 6 configuration of 3 different optimizers. We used the architecture depicted in 1 to measure their influence on improving the results and also to explore which best configuration gives the best results. We also propose a method to rank all the models from best to worst by introducing the notion of *gap-score*. Gap-score is a percentage-based metric based on the difference between reconstructions of images without any obstacles and reconstructions of the same images with obstacles.

The training process was conducted using a subset of RailSem19 [11] consisting of Regions of Interest (RoI) of track levels. In order to rank all the models on gap-scores, we use a testing dataset consisting of normal images and the same images with obstacles. Lastly the evaluation process of the 240 generated models was conducted using the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [12] an MCDM algorithm. The used criteria are the average gap-score on all normal images and the number of positive gap-scores for each normal image. By doing so, we avoid relying on arbitrary thresholds to judge an input whether it is anomalous or not.

The best resulting model was tested on a real world scenario describing a real obstacle with convincing results where the gap-score is considerably high on frames where the obstacle is present and considerably low on normal ones as described in section VI.

We summarize our contributions as follows:

- First, we propose an exploratory study using 240 convolutional autoencoders to detect on frame level railway obstacles. This large set of convolutional autoencoders

[1] Railenium Institute of Technological Research (IRT). amine.boussik@railenium.eu, wael.benmessaoud@railenium.eu

[2] Univ. Polytechnique Hauts-de-France and Railenium Institute of Technological Research (IRT), France smail.niar@uphf.fr, abdelmalik.taleb-ahmed@uphf.fr

has been built by combining the most frequently used activation functions, loss functions and optimizers.
- Second, we propose a methodology for ranking all the generated models using multi-criteria decision making from best to worst. The ranking is based on the gap-score which is the average of all gap-scores for different images. This comparison can help the designer to choose the most efficient model in obstacle detection.
- Third, we used RailSem19 [11] to extract RoIs to train the convolutional autoencoder instead of training on the whole image. This helps the model to concentrate only on features present on track level and avoid learning varieties on the whole image. By this way, we built a large input-data for training, testing and comparing the 240 convolutional autoencoders.

The remainder of this paper is organized as follows. Section II gives an overview of the state of the art in the domains of unsupervised models and Railway obstacle detection applications. In section III, we present our methodology for obstacle detection based on convolutional autoencoders. Section IV is devoted to the evaluation methodology. Experimental results are presented in Section V. Finally, the conclusion and future work will be given in section VI.

II. Related Works

Existing works in the domain of this paper can be divided in two parts:

A. Unsupervised models for anomaly detection

In the literature, there is an important number of works that uses unsupervised models for anomaly detection. Deterministic models, such as [1], propose an autoencoder for anomaly detection using non linear data. The authors in [13] use a convolutional autoencoder to detect anomalies on image logos of mobiles. They identify the input image as negative when it exceeds a predefined threshold. The authors in [14] exploit the use of convolutional autoencoders to detect defects in concrete. Their work relies on thresholding on pixel level where the mean value of the anomalous class is supposed to be as high as possible.

In addition to deterministic models, generative adversarial networks (GANs) have also been used in image anomaly detection. The authors in [15] propose Ano-GAN, one of the first works that exploits a GAN [16] in anomaly detection. The authors propose an iterative method to map input images to their corresponding noise in order to create their corresponding generated image and compare it with the original input. While the work in [17] is not being a method to detect anomalies using generative models, it represents a stepping stone for a lot of works using generative models in anomaly detection such as [19]. It uses a Bidirectional-GAN [17] to learn the mapping from the image to its corresponding latent representation automatically without the need of an iterative step. The authors in [18] propose GANomaly. Their work proposes enhancements in contrast to previous works in terms of the architecture and the used loss function. They use an autoencoder followed by an encoder in the generator and a novel loss function consisting of three components: adversarial loss, contextual loss and encoder loss.

B. Railway obstacle detection applications

In the context of using deep learning methods to detect railway obstacles, the majority of projects use image processing approaches and rarely rely on deep learning methods. In [20] a study on the feasibility of detecting railway obstacles at crossing level is presented. Their proposed method focuses on a charge-coupled device (CCD) greyscale camera and image processing to implement the detection method. The validation step is carried over by using a miniature crossing. The authors in [21] list a set of existing solutions and methods in the field of obstacle detection on railway's crossings. Among those methods are image processing based methods and sensor-based such as optical beams and 3D-laser radars. The work in [22] proposes a method based on background subtraction. Their method computes frame-by-frame correspondences between the current and the reference image sequences. Obstacles in their method are detected by applying image subtraction to corresponding frames. In [23], the authors apply hough Transform to detect possible obstacles using frontal mounted cameras.

As aforementioned, a scarcity in terms of the application of machine learning models in railway obstacle detection is considerably discernible. Nevertheless, some works tried to exploit supervised methods in order to detect obstacles. The authors of [24] exploit Fast Region-based Convolutional Network (Fast R-CNN) [5] on a predefined set of obstacle labels such as animals, persons, trains. While not proposing a specific detection method for obstacles, [25] proposes a new architecture to detect only rail tracks with high accuracy.

Moreover the specific use of unsupervised learning methods is still unexploited for railway obstacle detection. The closest work to the work we present here is a recently published in [8]. In this paper, the author propose a framework to detect obstacles in night-time using a convolutional autoencoder by producing absolute and gradient differences of the reconstructed image. The framework is consisting of a convolutional autoencoder, a CNN to predict on the frame level whether the image is anomalous or not. The CNN is also used to extract the corresponding heatmap to locate the anomaly. They use a pre-trained CNN on their own dataset entitled vesuvio [8] to predict the classes and evaluate the localisation of the obstacles. An initial version of the same work by the same authors can be found in [9]

III. Obstacle Detection Using Convolutional Autoencoder

We use a convolutional autoencoder where only normal unlabeled data is required to extract knowledge without any labels. In terms of anomaly detection, the model is trained on healthy inputs only. Given an input x, the model tries to encode it by compressing it to extract its latent data then decode it by comparing it to the original input generating by this its reconstruction x'. The autoencoder is divided in three components:

Encoder: The encoder part maps an input sample x to the bottleneck layer z.

Bottleneck Layer: The bottleneck z layer stores the low-dimensional latent representations for every sample x.

Decoder: The decoder part maps back the data from z to generate a reconstruction x' of the input x.

We use backpropagation algorithm to minimize the difference between the input data x and its reconstruction x' jointly for both the encoder and the decoder.

$$\arg\min_{\omega,\theta} \mathcal{L}(x, x') \tag{1}$$

Where ω and θ are respectively the parameters to optimize for the encoder and the decoder. $\mathcal{L}$ is the used loss function.

A. Loss functions

In this subsection, we present the different loss functions we consider in our comparison: Mean squared error (MSE) [26] is a pixel-wise loss function measuring the square difference between the ground truth and predicted labels. Mean absolute error (MAE) [26] is a pixel-wise loss function measuring the absolute difference between the ground truth and the predicted labels. Peak signal-to-noise ratio (PSNR) [27] is an image quality assessment metric used to compare the quality of two images. Often, the ground truth is referred to as signal and the prediction is referred to as noise. Structural similarity (SSIM) [28], [27] is a patch-wise quality metric used to measure the similarity between two images. Instead of using traditional error summation methods, this metric is designed by modeling any image distortion as a combination of three factors that are structure, luminance and contrast. Multi-scale Structural similarity (MS-SSIM) [28] is a variant of SSIM. MS-SSIM takes into consideration many levels of resolution and distortion and can be more robust with regard to variations in viewing conditions.

B. Activation functions

For the second component of our exploratory study, we explore the influence of the used activation functions. We compare 8 different activation functions : Rectified linear unit (relu) [33], Exponotial linear unit (elu) [34], Scaled Exponential Linear Unit (selu) [34], Mish [36], Swish [37], Hyperbolic tangent (tanh) [32], Mila [38] and Sharkfin [39].

C. Used Optimizers

For the last component of our exploratory study, we explore the importance of the used optimizers and their impact on the model's end results. In this study we compare three different optimizers: Adam [29], RAdam [30] and Novograd [31]. Further information regarding the used configurations is described in section V-A

IV. Evaluation Methodology

A. The Proposed Methodology

Most works which use convolutional autoencoders to detect anomalies are relying on its performance and the used training loss in order to set a fixed threshold to categorize the input as being anomalous or not. If the score generated by comparing the input and its reconstruction exceeds the threshold, it is flagged as anomalous. On the contrary, if the input does not exceed this threshold, it is categorized as a normal input. We assume that the key idea of image anomaly detection using a score-based unsupervised model is to discriminate well between normal and anomalous inputs by giving a minimal score on normal inputs and a maximal score on anomalous input. In existing works, every input is compared to an arbitrary fixed threshold without further assessment on the performance of the model on images with and without anomalies. We instead propose a method to rank and evaluate the models using gap-scores. These gap-scores correspond to the percentage gaps between the 2 reconstructions: the one obtained from the normal image without obstacle and the one obtained from the image with the obstacle. In other words, the gap-score is a percentage metric measuring how much the normal and anomalous images are different from each other. We then generate for each model two criteria described in section IV-B and maximize them using TOPSIS [12] as described in section IV-C.

B. Respected Criteria

We denote by *background image* the normal image without obstacles. We also denote by *inlaid image* the same normal image with inlaid obstacles. Let $\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_n\}$ be the dataset of background images containing N images and $\hat{\mathcal{X}} = \{\hat{x}_{1,1}, \hat{x}_{1,2}, \hat{x}_{2,1}, \hat{x}_{2,2}, \ldots, \hat{x}_{n,m}\}$ the dataset of inlaid images containing M images with $M \geq N$. Each background image x_i can have at least one or many inlaid counterparts $x_{i,1}$, $x_{i,2}$ ... $x_{i,m}$. A good model is a model that maximizes the following two criteria:

- For every background image and its generated inlaid counterparts, the model should maximize the average of the the gap-score between every inlaid image and its original background:

$$s_i = d(x_i, x'_i),\ \forall x_i \in X \tag{2}$$

$$s_{i,j} = d(\hat{x_{i,j}}, \hat{x'_{i,j}}),\ \forall x_i \in X, \forall x_j \in \hat{X} \tag{3}$$

$$C_1 = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s_{i,j} - s_i}{s_i} \tag{4}$$

  Where x_i and x'_i are respectively a normal image and its reconstruction by a given convolutional autoencoder model using a unique metric d. $\hat{x_{i,j}}$ and $\hat{x'_{i,j}}$ are respectively an inlaid counterpart of image x_i and its reconstruction by a given convolutional autoencoder model using a unique metric d. Here d is MSE [26]. s_i and $s_{i,j}$ are respectively the MSE scores for both a background image and its inlaid counterpart. We denote by C_1 the average gap-score criterion.
- For every background, a model should maximize the number of positive gap-scores generated for every background:

$$C_i = \begin{cases} 1, & \text{if } \sum_{j=1}^{m} \frac{s_{i,j} - s_i}{s_i} \times 100 > 0, \forall\, x_i \in X \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$C_2 = \sum_{i=1}^{n} C_i \tag{6}$$

We denote by C_2 the positive scores criterion.

C. Multi-criteria decision making for model evaluation

In order to evaluate the models generated from our exploratory study, we use MCDM methods, especially TOPSIS [12]. The evaluation matrix $(x_{ij})_{m\times n}$ is composed of n criteria and m alternatives. In our case, the criteria are the ones described in (section criteria), and the alternatives are the generated models. We fix a maximization strategy to know which model maximizes both criteria following the attributed weights for every criteria. Finally we calculate the distance d between each alternative, the ideal and anti-ideal solution extracted from the evaluation matrix to attribute a score for every alternative in order to rank them according to their distance to the ideal solution.

We summarize the steps used in TOPSIS [12] as the following:

- Let's consider $T = (x_{ij})_{m\times n}$ our evaluation matrix consisting of $m = 240$ and $n = 2$ where m is the number of generated models and n the number of criteria. We normalize the matrix as follows:
$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}}, i = 1, ..., m; j = 1, ..., n. \tag{7}$$
- Multiply the columns of the normalized decision matrix by the associated weights to obtain the weighted decision matrix:
$$v_{ij} = w_j . r_{ij}, i = 1, ..., m; j = 1, ..., n \tag{8}$$
Knowing that we have two criteria, we chose two weights, w_1 and w_2 where $w_1 + w_2$ = 1.
- Determine the ideal and anti-ideal solutions. The ideal solution, denoted as A^+, and the anti-ideal solution, denoted as A^-, are defined as follows (reformulate):
$$A^+ = \{v_1^+, v_2^+, .., v_n^+\} \tag{9}$$
$$A^+ = \{(\max_i v_{ij} | j \in K_p)\}\{\min_i v_{ij} | j \in K_p)\} \tag{10}$$
$$A^- = \{v_1^-, v_2^-, .., v_n^-\} \tag{11}$$
$$A^- = \{(\min_i v_{ij} | j \in K_n)\}\{\max_i v_{ij} | j \in K_n)\} \tag{12}$$
Where K_p and K_n are respectively the set of criteria having a positive impact, and the set of criteria having a negative impact.
- Next, we calculate the L^2 distance between the target alternative i and the ideal and anti-ideal solution respectively:
$$S_i^+ = \sqrt{\sum_{j=1}^{n} (v_j^+ - v_{ij})^2}, i = 1, ..., m; j = 1, ..., n. \tag{13}$$
$$S_i^- = \sqrt{\sum_{j=1}^{n} (v_j^- - v_{ij})^2}, i = 1, ..., m; j = 1, ..., n. \tag{14}$$
- Lastly, we calculate the similarity to the ideal solution:
$$S_i^* = \frac{S_i^-}{S_i^- + S_i^+} \tag{15}$$

V. Experiments

A. Architecture and Hyperparameters

In terms of the convolutional autoencoder's architecture, we set the number of layers, depth, hyperparameters, upsampling, maxpooling, transposed convolutions, strided convolutions and the bottleneck size as shown in 1. The encoder and decoder parts consist of 6 convolutional layers respectively to downsample and upsample the image. For each layer in the encoder, we use strided convolutions to reduce the size of the image by 2 for each step. We observed that avoiding maxpooling helped generating good reconstructions. with each layer we set the filters from 32 to 128 with kernels of size 3x3 except for the bottleneck layers that has 48 filters. We observed that below this value, the performance of the model becomes weaker, and above this value, the model tries to easily reconstruct unusual segments of an anomalous image. For each layer of the decoder part, we use Upsampling instead of transposed convolution and filters ranging from 128 to 32 with kernels of size 3x3 (figure architecture). The aim of this part is to get the adequate architecture depending on the used training dataset to initiate our exploratory study. Using the same architecture, we generate each time a model following a combination of the aforementioned components of our exploratory study. In terms of the activation functions' hyperparamters, MSE [26], MAE [26] and PSNR [27] are without parameters. Only SSIM [28] and MS-SIM [28] are concerned. We set their parameters to the default parameters recommended in their original papers. Every activation function is paired with uniform glorot initialization [40] except for selu [35], which is paired with normal lecun initialization [35]. Regarding the used optimizers, we differentiate different kinds of configurations for each optimizer used. For Adam [29] we use only one configuration with a learning rate = 10^{-3}. For RAdam [30] we set two different configurations, a raw configuration using the following parameters, learning rate = 10^{-3}, steps = 0, warmup = 10^{-1} and minimum learning rate set to 0. The other configuration is the parameterized configuration with learning rate = 10^{-3}, steps = 10^5, warmup = 10^{-1} and

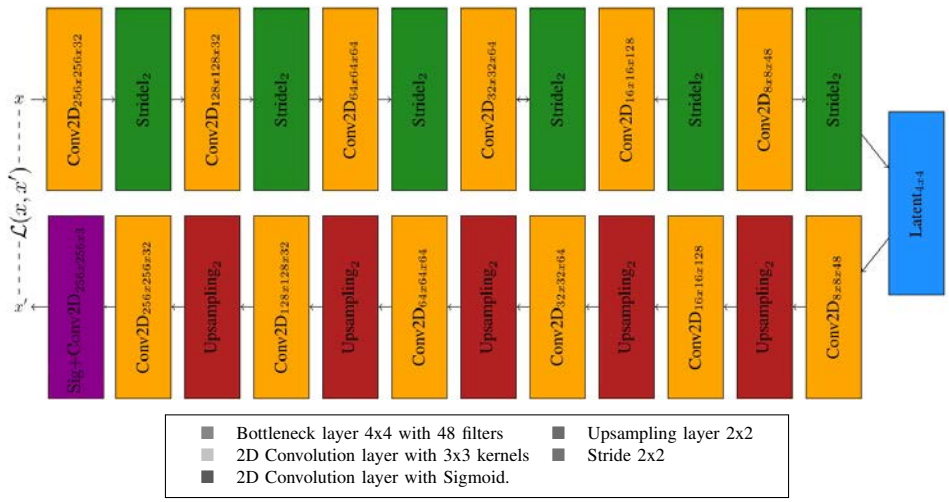

Fig. 1. Convolutional autoencoder architecture, all layers use 3x3 kernels, the latent layer represents the bottleneck layer of the model. The input of the encoder is a normalized image of size 256x256 of track level regions of interest. The last layer of the decoder represents the output of the whole model and uses sigmoid [42] as activation to output pixel values ranging between 0 and 1.

minimum learning rate set to 10^{-5}. For Novograd [31] we distinguish three kind of configurations with the following parameters, a raw configuration with learning rate = 10^{-3}, decay = 0 and without using gradient averaging. A second parameterized configuration with learning rate = 10^{-3}, decay = 10^{-1} with gradient averaging and the last configuration for Novograd [31] is an averaged parameterized configuration with learning rate = 10^{-3}, decay = 10^{-3} with gradient averaging.

Each generated model uses the same architecture and configurations described above and is trained using the dataset described in section V-B for 300 epoches. Figure 1 shows the final architecture of each generated model.

B. Training Dataset

The used dataset in training the convolutional autoencoders is a subset of RailSem19 [11] where rail tracks are extracted as regions of interest. The original dataset is processed beforehand where we omit unexploitable and tramway scene images leaving only the railway images. Then, we generate the corresponding masks for each region of interest by connecting the polygons in the corresponding json file for each image scene instead of working with the corresponding predefined masks of each image. By doing so, we can individually extract each rail track present on the scene. By applying the extracted mask for each scene, we generate a dataset consisting only of regions of interest. Each region is then cropped and resized to a resolution of 256x256 pixels. Figure 2 shows a preview of the extracted dataset. The resulting dataset consists of 1353 images of track level regions of interest and split into 80% for training and 20% for validation.

C. Test Dataset

In order to calculate the gap-score and due to the lack of a railway-driven dataset describing real world obstacles over rails, we generate our test dataset using Gaussian-Poisson

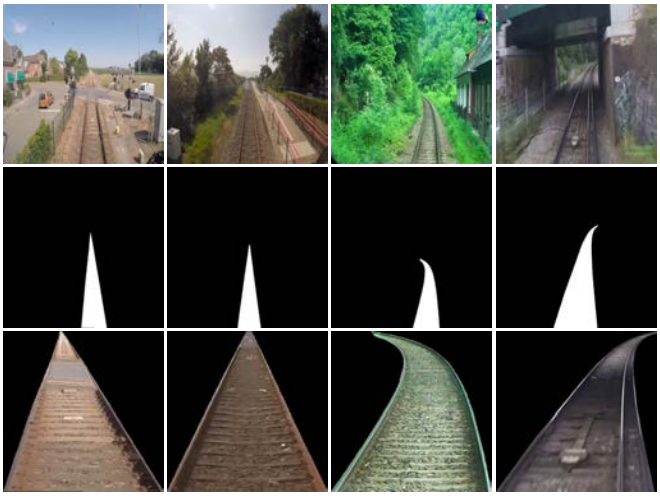

Fig. 2. Preview of our dataset of RoIs using RailSem19 [11].

Generative Adversarial Network (GP-GAN) [10] which is a generative model capable of blending images with high-resolution and generating realistic blends to inlay obstacles synthetically on RailSem19 [11] images. The dataset consists of 19 background images where no obstacle is inlaid, this represent our reference in order to calculate the gap-score. Then for each background, we inlay on it obstacles by means of GP-GAN [10] in order to create inlaid counterparts. Each background have an average of 10 inlaid counterparts. The resulting dataset consists of 19 regions of interest for the background and 181 regions of interests for the inlaid counterparts. The regions of interests are extracted following the same fashion described in V-B. Figure 3 shows a preview of the dataset with some reconstructions results.

VI. Results

In this section, we show the results of our exploratory study on background images and their inlaid counterpart.

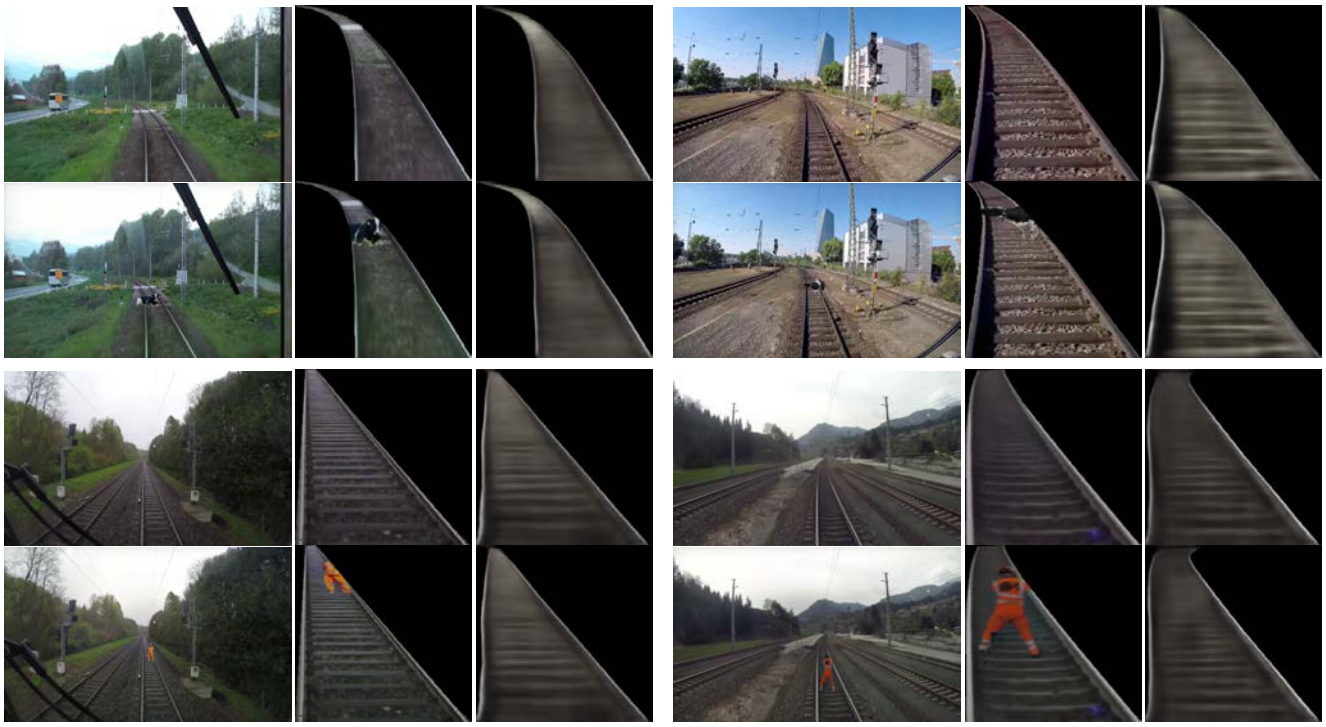

Fig. 3. Preview of the test dataset showing four different scenarios. The first and third rows show four examples of background images without any obstacles followed by their region of interest and the reconstruction of the same region by a picked model. The second and the last rows show the inlaid counterparts of the previously shown backgrounds, each with their respective regions of interest and reconstructions by the same model.

Then we show the results of every model in relation to the corresponding calculated gap-scores and their ranking using TOPSIS.

For the sake of simplifying the results of every model generated in our study, we take only the 10-best generated models from the 240 models. Furthermore, to simplify showing the results, a naming convention is used for each model to combine it with the corresponding components as follows: optimizers_loss_activation. For optimizers, as described in as described in section V-A, we distinguish the following naming conventions:

- radam_[par,raw]: Here 'par' label signifies custom RAdam optimizer with the parametrized configuration and 'raw' signifies the raw configuration.
- novograd_[par,raw]_[avg]: Here 'par' label signifies Novograd optimizer with the parameterized configuration, 'raw' signifies the raw configuration and 'avg' signifies the use of gradient averaging.

A. Models Evaluation and Ranking

This section shows the final results generated from the two previously shown results. It represents as well the output of TOPSIS [12] method to rank all models according to the respected criteria. 'Average gap-scores' column represents the first criterion shown in section IV-B where calculate the average of all gap-scores for each background, 'Positive scores' represents the second criterion shown in section IV-B. By applying TOPSIS [12] on both criteria we get the third column 'Rank' in order to rank the top models from best to worst.

For the first table I, we observe that the majority of models have a combination of RAdam [29] as an optimizer and psnr [27] as a loss function with the exception of the best model that has Adam [29] as an optimizer and two other models having MSE [26] as a loss function. With regard to Gap-scores, all models have scores ranging from 52.97% to 68.17%. In terms of average positive scores of the test dataset, none of the 240 generated models reaches a perfect score of 19 positive backgrounds with the maximum reached is 18 of 19. The corresponding weights of TOPSIS [12] method are of equal importance for both criteria i.e $w_1 = 0.5$, $w_2 = 0.5$. With that, the method tries to find the best models that respect an equal trade-off between maximized gap-scores and positive scores respectively. Finally we observed a variety of activation functions for every model such as mila [38], relu [33], mish [36], elu [34], selu [35] and tanh [32].

However, for the second table II, by giving much importance to positive scores, i.e $w_1 = 0.1$, $w_2 = 0.9$, we observe the emergence of new models having different combinations such as the pairing of RAdam [29] with MS-SSIM [28] and swish [37], the pairing of RAdam [29] with SSIM [28] and as well as the pairing of Novograd [31] and MSE [26]. Those same models are the only ones having the most of positive score with the exception of the model using SSIM [28] by reaching 18 out of 19 positive scores on all backgrounds, but at the cost of average gap-score ranging from 36.93% to 44.62%. We observe the same tendency concerning the top model which stays the same even when changing the weights for every criterion.

B. Testing on a real world scenario

We test our best evaluated model with the configuration 'adam_psnr_mila' on a video [41] describing a rare event of a real-life scenario of railway obstacle. The video shows a scenario where a horse tries to trespasses the railway level along with the driver of train. We extract from the video two segments that highlights both the obstacle and the driver on track level. We then extract the corresponding regions of interest of each frame for each segment to prepare them for the model. For the sake of showing the results as gap-scores on each frame, we calculate the score for only one image on the beginning of the video that shows no sign of obstacle in order to calculate the gap-score for other frames containing obstacles. Figure 4 and 5 are diagrams showing the results of the best evaluated model (first in rank) using gap-scores for the first and second segment respectively. Each diagram shows the gap-score of each frame in percentage along the Y-axis, and the frame number in segment along X-axis. Both results can be found at the following video links in [43], [44]

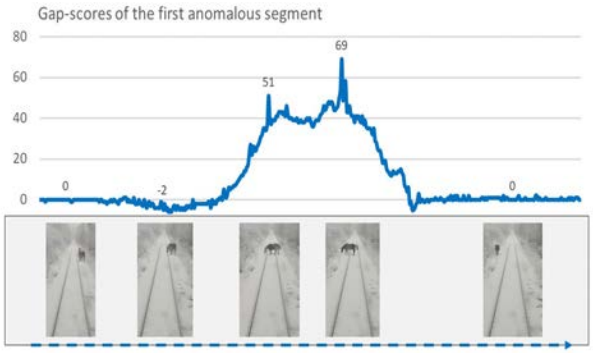

Fig. 4. Results of the gap-score on the first segment. The gap-score stays relatively constant for frames at the beginning and the end and rises with the obstacle reaching peak values up to 69%

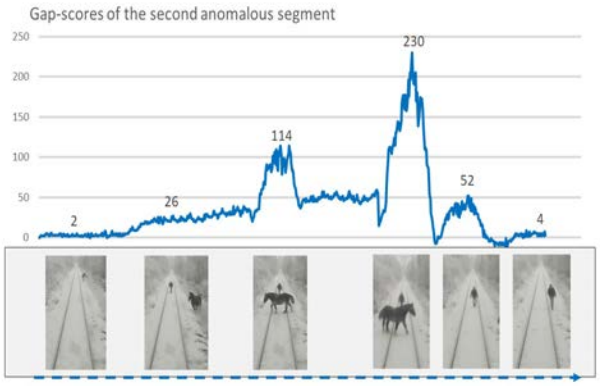

Fig. 5. Results of the gap-score on the second segment. The gap-score stays relatively constant for frames at the beginning and the end and rises with the obstacle reaching peak values of 114%, 230% and 52% respectively.

Models	Average gap-score	Positive scores	Rank
adam_psnr_mila	68.17%	17	1
radam_par_psnr_mila	66.62%	16	2
radam_par_mse_relu	62.88%	15	3
radam_par_psnr_mish	61.14 %	15	4
radam_raw_psnr_mila	61.14%	15	5
radam_raw_psnr_elu	57.42 %	14	6
radam_raw_psnr_selu	55.00%	16	7
radam_par_mse_tanh	53.62 %	17	8
radam_raw_mse_relu	53.12 %	15	9
radam_par_psnr_elu	52.97 %	14	10

TABLE I: Results of evaluation based on the generated models (top 10) with equal importance to both criteria with the following weights, w_1 = 0.5 for gap-scores and w_2 = 0.5 for positive scores.

Models	Average gap-scores	Positive scores	Rank
adam_psnr_mila	68.17%	17	1
radam_par_mse_tanh	53.62%	17	2
radam_par_msssim_relu	44.62%	18	3
radam_raw_psnr_mish	49.78%	17	4
radam_raw_ssim_relu	45.21%	17	5
radam_par_psnr_mila	66.62%	16	6
radam_raw_mse_mila	42.78%	17	7
novograd_par_avg_mse_tanh	36.93%	18	8
radam_par_msssim_swish	41.67%	17	9
adam_mse_tanh	40.72%	17	10

TABLE II: Results of evaluation based on the generated models (top 10) with more importance given to the number of positives scores for each background with the following weights, w_1 = 0.1 for gap-scores and w_2 = 0.9 for positive scores.

VII. Conclusion

In this paper, we have explored the use of convolutional autoencoders for railway obstacle detection by generating 240 models with different components. We introduced gap-score which is a percent-based metric dedicated to evaluate the performances of the generated models. We also used MCDM algorithms to rank them. The top 10 models showed performances reaching an average gap-score up to 68% on the test set. The best model was tested on a rare event of an obstacle over rails, the model shows high values when encountering an obstacle and low values on normal frames. The remaining models have also proved their efficiency on the same real scenario. The ranking of the models stayed the same on both synthetic and real cases.

The next step will be the development of a segmentation model to locate in real time the RoIs as well as adding more decision criteria tailored specifically for railways.

ACKNOWLEDGMENT

This research work contributes to the french collaborative project TFA (autonomous freight train). It was carried out in the framework of IRT Railenium, Valenciennes, France, and therefore was granted public funds within the scope of the French Program "Investissements d'Avenir".

REFERENCES

[1] Sakurada, Mayu, and Takehisa Yairi. "Anomaly detection using autoencoders with nonlinear dimensionality reduction." Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. 2014.

[2] Prabhakar, Gowdham, et al. "Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving." 2017 IEEE region 10 symposium (TENSYMP). IEEE, 2017.

[3] Garnett, Noa, et al. "Real-time category-based and general obstacle detection for autonomous driving." Proceedings of the IEEE International Conference on Computer Vision Workshops. 2017.

[4] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).

[5] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." arXiv preprint arXiv:1506.01497 (2015).

[6] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[7] He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

[8] Gasparini, Riccardo, et al. "Anomaly Detection, Localization and Classification for Railway Inspection." 25th International Conference of Pattern Recognition. 2020.

[9] Gasparini, R., Pini, S., Borghi, G., Scaglione, G., Calderara, S., Fedeli, E. and Cucchiara, R., 2020, September. Anomaly detection for vision-based railway inspection. In European Dependable Computing Conference (pp. 56-67). Springer, Cham.

[10] Wu, Huikai, et al. "Gp-gan: Towards realistic high-resolution image blending." Proceedings of the 27th ACM international conference on multimedia. 2019.

[11] Zendel, Oliver, et al. "RailSem19: A dataset for semantic rail scene understanding." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2019.

[12] Lai, Young-Jou, Ting-Yun Liu, and Ching-Lai Hwang. "Topsis for MODM." European journal of operational research 76.3 (1994): 486-500.

[13] Ke, Muyuan, Chunyi Lin, and Qinghua Huang. "Anomaly detection of Logo images in the mobile phone using convolutional autoencoder." 2017 4th International Conference on Systems and Informatics (ICSAI). IEEE, 2017.

[14] Chow, Jun Kang, et al. "Anomaly detection of defects on concrete structures with the convolutional autoencoder." Advanced Engineering Informatics 45 (2020): 101105.

[15] Schlegl, Thomas, et al. "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery." International conference on information processing in medical imaging. Springer, Cham, 2017.

[16] Goodfellow, Ian J., et al. "Generative adversarial networks." arXiv preprint arXiv:1406.2661 (2014).

[17] Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning." arXiv preprint arXiv:1605.09782 (2016).

[18] Akcay, Samet, Amir Atapour-Abarghouei, and Toby P. Breckon. "Ganomaly: Semi-supervised anomaly detection via adversarial training." Asian conference on computer vision. Springer, Cham, 2018.

[19] Zenati, Houssam, et al. "Efficient gan-based anomaly detection." arXiv preprint arXiv:1802.06222 (2018).

[20] Pu, Yong-Ren, Li-Wei Chen, and Su-Hsing Lee. "Study of moving obstacle detection at railway crossing by machine vision." Y.-R. Pu. Informational Technology Journal 13.16 (2014): 2611-2618.

[21] Pavlović, Milan, Nenad T. Pavlović, and Vukašin Pavlović. "Methods for detection of obstacles on the railway level crossing." Proc. 17th Scientific-Expert Conference on Railways RAILCON '16. 2016.

[22] Mukojima, Hiroki, et al. "Moving camera background-subtraction for obstacle detection on railway tracks." 2016 IEEE international conference on image processing (ICIP). IEEE, 2016.

[23] Rodriguez, LA Fonseca, Jonny Alexander Uribe, and JF Vargas Bonilla. "Obstacle detection over rails using hough transform." 2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA). IEEE, 2012.

[24] Yu, Mingyang, Peng Yang, and Sen Wei. "Railway obstacle detection algorithm using neural network." AIP Conference Proceedings. Vol. 1967. No. 1. AIP Publishing LLC, 2018.

[25] Wang, Zhangyu, et al. "Efficient rail area detection using convolutional neural network." IEEE Access 6 (2018): 77656-77664.

[26] Botchkarev, Alexei. "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology." arXiv preprint arXiv:1809.03006 (2018).

[27] Hore, Alain, and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM." 2010 20th international conference on pattern recognition. IEEE, 2010.

[28] Wang, Zhou, et al. "Image quality assessment: from error visibility to structural similarity." IEEE transactions on image processing 13.4 (2004): 600-612.

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

[30] Liu, Liyuan, et al. "On the variance of the adaptive learning rate and beyond." arXiv preprint arXiv:1908.03265 (2019).

[31] Ginsburg, Boris, et al. "Training Deep Networks with Stochastic Gradient Normalized by Layerwise Adaptive Second Moments." (2019).

[32] Manessi, Franco, and Alessandro Rozza. "Learning combinations of activation functions." 2018 24th International Conference on Pattern Recognition (ICPR). IEEE, 2018.

[33] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Icml. 2010.

[34] Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." arXiv preprint arXiv:1511.07289 (2015).

[35] Klambauer, Günter, et al. "Self-normalizing neural networks." arXiv preprint arXiv:1706.02515 (2017).

[36] Misra, Diganta. "Mish: A self regularized non-monotonic neural activation function." arXiv preprint arXiv:1908.08681 4 (2019).

[37] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).

[38] Misra, Diganta, Mila: Controlling Minima Concavity in Activation Function, https://github.com/digantamisra98/Mila

[39] Misra, Diganta, SharkFin: A Modified Version of ReLU, https://github.com/digantamisra98/SharkFin

[40] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.

[41] Liono, pointcom, AB-VIEW Un cheval sur la voie sncf en Ariège, https://www.youtube.com/watch?v=N8DYcNRkauY

[42] Nwankpa, Chigozie, et al. "Activation functions: Comparison of trends in practice and research for deep learning." arXiv preprint arXiv:1811.03378 (2018).

[43] Link to the first segment's results: `https://youtu.be/kXr2otmpkM8` .

[44] Link to the second segment's results: `https://youtu.be/fv7p5uPyLds`