Deliverable D 2.3

WP2 Report on experimentation, analysis, and discussion of results

Project acronym:	RAILS
Starting date:	01/12/2019
Duration (in months):	43
Call (part) identifier:	H2020-S2R-OC-IPX-01-2019
Grant agreement no:	881782
Due date of deliverable:	Month 37
Actual submission date:	February 28st 2023
Responsible/Author:	Stefania Santini (CINI)
Dissemination level:	Public
Status:	Issued

Reviewed: yes

Document history		
Revision	*Date*	*Description*
1	30th January	First issue for internal and AB review
2	28th February	Second issue for EU review
3	25th May	Third issue after EU review

Report contributors		
Name	Beneficiary Short Name	Details of contribution
Stefania Santini	CINI	WP2 Leader and Contributor
Lorenzo De Donato	CINI	Contributor
Francesco Flammini	LNU	Internal Review
Rob Goverde	TUDELFT	Internal Review

Advisory Board Reviewers	
Name	*Company or Institution*
Danijela Ristic-Durrant	University of Bremen, Germany, and S2R SMART2 project
Milan Banić	University of Niš, Republic of Serbia, and S2R SMART2 project

Funding

This project has received funding from the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement n. 881782 Rails. The JU receives support from the European Union's Horizon 2020 research and innovation program and the Shift2Rail JU members other than the Union.

Disclaimer

Contents

Executive Summary

This deliverable proposes innovative AI approaches to address the problems and challenges that emerged from the methodological proofs-of-concept (PoCs) reported in Deliverable D2.2, in order to investigate the adoption of learning techniques and other AI methods for enhanced rail safety and automation. On the basis of the objectives, research questions, and AI techniques identified in the previous deliverable for two pilot case studies, experimental proofs-of-concept are provided to explore the technical feasibility of specific railway functionalities through the use of AI approaches transferred from other transportation sectors.
To this aim, the document addresses the following themes for each case study: $i)$ a technical description of the proposed innovative methods, highlighting the problem statement to solve and the learning techniques which are going to be leveraged; $ii)$ the definition of the training phase for the learning approaches, which could be conducted exploiting selected datasets, as well as through the use of ad-hoc simulation platforms; $iii)$ the description of the validation procedure to show the effectiveness of the proposed strategies in concrete operational scenarios; $iv)$ a preliminary discussion of the results to highlight possible benefits and drawbacks of the innovative approaches; this phase can possibly include a comparison analysis with traditional models to be evaluated through some of the key performance indicators identified in the previous deliverable.
A description of the background for each case study is addressed in Section 1; the main objectives of the deliverable are detailed in Section 2, while the content of the two experimental proofs-of-concept is reported in Section 3.

Public Benchmark from WP2. Most of the data produced to develop the "Railway Obstacle Detection and Collision Avoidance" PoC (discussed in Chapter 4) are publicly available on Zenodo[1]. These data are a product of the research activity conducted in RAILS and created for scientific purposes only to study the potentials of Deep Learning (DL) approaches when used to analyse Video Data in order to detect possible obstacles on rail tracks and thus avoid collisions. The RAILS Consortium and the authors **do not assume** any responsibility for the use that other researchers or users will make of these data.

[1] https://zenodo.org/record/7924875

Abbreviations and acronyms

Abbreviations / Acronyms	Description
A-NAD	Augmented Non-Anomaly Dataset
ADM	Anomaly Detection Module
AE	Autoencoder
ATD	Autonomous Train Driving
AF	Actication Function
ATO	Automatic Train Operation
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DNN	Deep Neural Network
DPG	Deep Policy Gradient
DQN	Deep Quality Network
DRL	Deep Reinforcement Learning
EC	Energy Consumption
FPS	Frames Per Second
FTV	Free Track Video
GPU	Graphics Processing Unit
HST	High Speed Train
ILS	Initial Labelled Set
IoU	Intersection over Union
KPI	Key Performance Indicator
LPF	Leader-Predecessor-Follower
ML	Machine Leraning
MPC	Model Predictive Control
NAD	Non-Anomaly Dataset
NN	Neural Network
ODM	Object Detection Module
PxACC	Pixel Accuracy
R-CNN	Region-Based CNN
RBC	Radio Block Center
RBD	Relative braking Distance
RDM	Rails Detection Module
SS	Semantic Segmentation
SSIM	Structural Similarity Index Measure
T2I	Train to Infrastructure
T2T	Train to Train
TI	Tracking Index
VBODS	Vision-Based Obstacle Detection System
VC	Virtual Coupling
VCTS	Virtually Coupled Train Set
VQ-VAE	Vector Quantised-Variational AE

WCV	With Car Video

1. Background

The present document constitutes Deliverable D2.3 “WP2 Report on experimentation, analysis, and discussion of results” of the Shift2Rail JU project “Roadmaps for AI integration in the Rail Sector” (RAILS). The project is in the framework of Shift2Rail’s Innovation Programme IPX. As such, RAILS does not focus on a specific domain, nor does it directly contribute to specific Technical Demonstrators but contributes to Disruptive Innovation and Exploratory Research in the field of Artificial Intelligence within the Shift2Rail Innovation Programme. The successor of the Shift2Rail Joint Undertaking is currently the Europe’s Rail Joint Undertaking (EU-Rail) established by Council Regulation (EU) 2021/2085 of 19 November 2021.
The RAILS Workpackage WP2 investigates the adoption of learning techniques and other AI methods for enhanced rail safety and automation. The present deliverable is consequent to the results reported in Deliverable D2.2, in which methodological proofs-of-concept have been provided for two selected case studies: “Obstacle Detection for Collision Avoidance”, and “Cooperative Driving for Virtual Coupling of Autonomous Trains”. The first one aims to effectively locate, identify, and detect any kind of obstacles on the railway track, by leveraging lightweight equipment, such as a single camera mounted in front of the train. As to the second case study, the main goal is to virtually couple two or more trains in a single convoy through a Train-to-Train communication network to reduce the headway between them, thus enhancing lane capacity. Some research questions have arisen, which try to investigate if some of the AI approaches already assessed in other transportation sectors could be actually transferred to the railway field for the specific domains. After an examination of different AI methods, some learning approaches have been assumed to be leveraged, namely, computer vision and unsupervised learning for the first case study, and reinforcement learning for the second one, and a description of the corresponding methodologies have been provided. Furthermore, specific key performance indicators have been identified to evaluate the effectiveness of the proposed approaches, some of them being transferred from the automotive field. Finally, the expected results and possible criticalities of the proposed methodologies have arisen from this preliminary analysis.
The following step is therefore to support the theoretical proofs-of-concept proposed in Deliverable D2.2 with experimental results. This could answer to the research questions and the expected results that emerged in the previous deliverable, and could represent a further step towards the definition of a benchmark for future research inspiration.

2. Objective

This document, in line with the previous deliverables, deals with the following objectives of the RAILS project:

- *Objective 4: Development of methodological and experimental proofs-of-concept*;
- *Objective 5: Development of Benchmarks, Models and Simulations*.

In particular, in the previous deliverables, two pilot case studies have been selected, namely, “Obstacle Detection for Collision Avoidance”, and “Cooperative Driving for Virtual Coupling of Autonomous Trains”; for each of them, methodological proofs-of-concept have been addressed to study the feasibility of AI methods in the railway field, and some learning approaches have been identified as a potential solution to develop their respective railway functionalities.
On the basis of the considerations made in Deliverable D2.2, the main goal of this deliverable is to investigate innovative AI models for the considered case studies to be evaluated via test and validation activities, in order to understand how and if the AI approaches identified during the previous tasks can support and enhance rail safety and automation.
Indeed, the ultimate goal of the proofs-of-concept is not to propose full solutions to a specific problem but to derive lessons learned and recommendations for future research that will be collected in the next Deliverable D2.4. To this aim, the present document focuses on the following objectives:

- the definition of detailed AI models based on the selected learning approaches;
- the description of the learning process of the proposed methods through a training phase, which can be carried out exploiting specific datasets, as well as through a simulation platform developed ad-hoc for the purpose;
- the validation of the proposed models to show their effectiveness in simulated operational scenarios;
- a preliminary analysis of the results, to highlight the possible benefits and drawbacks of the proposed techniques.

Hence, this study is meant to be a step towards the acquisition of the necessary knowledge to understand the potentiality of AI in railways, and to drive the rail sector towards a vision of safe automation and intelligent train control. In this direction, the main object of the next deliverable will be an in-depth analysis to identify gaps and opportunities, weaknesses and strengths that emerged from each case study, with the final aim of defining technology roadmaps towards the effective adoption of AI in the rail sector.

3. Introduction

Deliverable D2.3 reports the validation activities of the solutions and approaches described and deeply analysed in Deliverable D2.2. It focuses on the analyses and simulations conducted applying the selected AI techniques to the pilot case studies identified in Deliverable D2.1 in concrete operational scenarios. Hence, this deliverable provides meaningful insights and information on the validity of the research results and the feasibility of the approaches in real settings. The heart of this document consists of two main chapters, addressing the aforementioned issues for the two selected pilot case studies: Chapter 4 is devoted to "Obstacle Detection for Collision Avoidance" and Chapter 5 deals with "Cooperative Driving for Virtual Coupling of Autonomous Trains".
The two chapters share the same structure: a brief introduction constitutes Sections 4.1 and 5.1; a detailed description of the proposed AI models is provided in Sections 4.2 and 5.2; the selection of specific datasets as well as the development of an ad-hoc simulation platform for training and validation purposes is described in Sections 4.3 and 5.3; the training phase together with the validation of the proposed approach in concrete operational scenarios are deeply analysed in Sections 4.4 and 5.4; the results of the validation phase are shown in Sections 4.5 and 5.5; finally, a preliminary discussion of the potential advantages of the proposed approaches is addressed in Sections 4.6 and 5.6.
A critical examination of the work and of the results obtained in this Deliverable, also against the current state-of-the-art in railways, will be the object of the next Deliverable D2.4 ("Report on identification of future innovation needs and recommendations for improvements"). Specifically, the latter will report lessons learned, weaknesses and strengths shown by each exploited technology, technical and implementation recommendations, unaddressed issues, and innovation needs, with the aim of identifying technology roadmaps for AI integration in the rail sector.

4. Railway Obstacle Detection and Collision Avoidance

4.1. Introduction

In the context of Autonomous Train Driving (ATD), *environmental awareness* is one of the fundamental requirements to safely take autonomous driving decisions, especially in open railway environments (i.e., railways that are not completely isolated from external factors) [1]. In order to move towards full autonomy, trains should be equipped with adequate sensors and systems that allow them to achieve **full situation awareness** in relation to the health status of the internal components of the vehicle and external threats or signals. We propose a generic framework for AI ATD systems in Fig. 4.1.

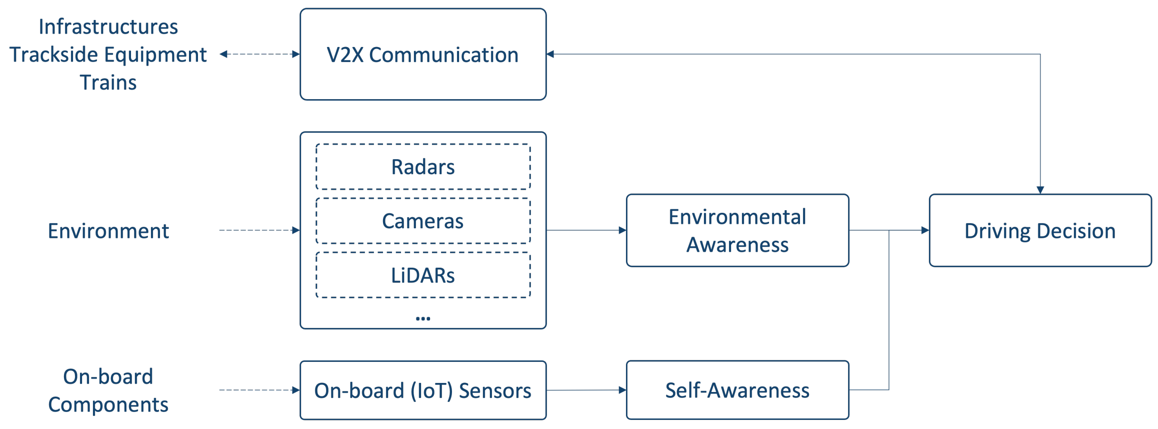


Fig. 4.1. A Generic Framework for AI ATD Systems.

In the following of this chapter, we mainly focus on *Environmental Awareness*, and specifically on Vision-Based Obstacle Detection (sub-)System (VBODS) enabled by AI approaches.

To detect obstacles in safety-critical scenarios, obstacle detection systems typically leverage data coming from multiple sensors including, among others, LiDARs, radars, and cameras (as shown in Fig.4.1). Furthermore, obstacle recognition may also involve different systems deployed at different "levels" (e.g., on-board, trackside, and airborne sub-systems as discussed within the SMART [2] and SMART2 [3] projects). Nevertheless, such complex systems, despite being (potentially) extremely effective, introduce several costs given the massive sensorization required for their implementation.

In these contexts, cameras are mainly used to classify the type of the obstacles[1] while their identification within the environment (i.e., their localisation) is demanded to the other sensors. Indeed, in the literature, different solutions have already been proposed for vision-based obstacle detection on rail tracks, involving both AI and non-AI approaches [4]; however, to the best of our knowledge, the developed AI systems mainly rely on *Supervised approaches*. Briefly, these kinds of AI models (to be specific, Deep Learning - DL - models) are trained to detect a set of elements (obstacles, in our case) that are specified in advance. For example, a Deep Neural Network (DNN) can be trained to detect some pre-specified

[1]Cameras have also been used for object distance estimation in SMART [2] and SMART2 [3]

obstacles such as cars, pedestrians, some species of animals, and the like. The main problem with these approaches is that if an obstacle that has not been taken into account when training the DNN has to be detected, the DNN will most likely fail.

The research activities conducted to develop the proof-of-concept described in this chapter, aimed at understanding to what extent it would be possible to overcome this *coverage* issue and leverage one of the cheapest sensors, i.e., **a RGB camera**, in order to discuss opportunities and shortcomings. Hence, we investigated:

- Unsupervised DL approaches that would potentially allow VBODs to detect any obstacle (and not just those specified *a priori*);
- Strengths and limits of implementing a *cheaper (or supportive) alternative*, compared to the systems introduced above, *which exploits AI, artificial vision, and data coming from a single camera mounted in front of the train.*

For the sake of readability, in the following, we refer to these terms:

- Obstacles that are known a-priori will be referred to as **objects**, while the action of detecting objects will be referred to as **Object Detection**. Theoretically, it would be possible to pre-specify any kind of obstacle, build a labelled dataset containing all these objects, train a DNN in supervised mode, and build an obstacle detection system based on such DNN. However, this is too far from being physically feasible given the extremely large amount of data and time needed to build a general-purpose object detector. Hence, the assumption in this study is that it would be feasible to pre-specify just a subset of objects
- Given the assumption above, we will refer to obstacles that are not known, i.e., that are difficult to properly specify and identify a-priori, as **anomalies**; consequently, the action of detecting anomalies will be referred to as **Anomaly Detection**.
- Finally, with the term **obstacle**, we will refer to both objects and anomalies.

To formalise the objectives to achieve and understand the potential effectiveness and the theoretical applicability of the approach, in Deliverable D2.2 [5] some Research Questions (RQs) and Key Performance Indicators (KPIs) were defined. They are reported in Table 4.1 and Table 4.2, respectively.

Table 4.1: Research Questions.

RQ	Description
RQ1	How can we detect known or unknown obstacles, such as rocks, vehicles, trees and people, in front of the train by using train front cameras and artificial vision?
RQ2	Can obstacle detection solutions based on artificial vision be transferred and adapted from other sectors (e.g., automotive, avionics, robotics, etc.) to railways?
RQ3	Can we demonstrate possible answers to RQ1 and RQ2 through a simple proof-of-concept demonstrator in order to inspire future developments and a technology roadmap?

To conclude, the case study under analysis is an extremely interesting test-bench given its relevance in the railway domain, however, we firmly think that *lessons learnt and approaches developed by facing this case study would be valuable also in other contexts* (e.g., obstacle detection at level crossings).

Table 4.2: KPIs for the evaluation of AI-based obstacle detection systems.

KPI	Description
KPI1	**Detection Distance**. Indicates the ability of the system to detect distant obstacles in order to mitigate as much as possible the effects of any impact. Measured in meters.
KPI2	**Obstacle Coverage**. Indicates the coverage of the system in terms of the types of obstacle it can detect. Measured in number of objects' classes (limited in case of supervised approaches).
KPI3	**Computation Time**. Indicates the time required by the system to detect possible obstacles on the tracks. Measured in milliseconds.

4.2. Model Description

As from the comprehensive analysis performed in [4], VBODSs typically involve two phases: first, they detect rail tracks, then, they detect and classify obstacles on or near rail tracks. The main problem is that, as introduced in Section 4.1, most DL systems are trained in a supervised manner. Architecture families such as YOLO or region-based CNNs (R-CNNs) - commonly known as object detectors - have shown incredible performance when it comes to detecting objects; additionally, they have already been widely investigated for maintenance/inspection applications in the rail sector [6, 7]. However, in the training phase, they should be fed with labelled data (objects bounding boxes and labels). This means that if an obstacle, that has not been elaborated in the training phase, appears on the track, it will most likely not be detected. This is an extremely sensitive issue; it would be nearly impossible to take into account (a-priori) every object that could possibly occupy the rail tracks. Common objects could be vehicles, pedestrians, animals, etc; however, also these “classes” are composed of different sub-classes (e.g., cars, motorbikes, and trucks for the vehicle class). Therefore, building a comprehensive dataset that incorporates all the possible classes and sub-classes is extremely time-consuming and, most likely, physically unfeasible as hundreds of kinds of objects should be considered.

To partially overcome this issue, it would be possible to leverage systems oriented at detecting anomalies based on unsupervised DL approaches. Visionary, an anomaly detection system will learn characteristics related to “free” tracks and, in case an uncommon scenario is presented in input, it will produce an anomaly map indicating where the anomaly is located (however, it will not specify the class of the anomaly). Worth mentioning, an anomaly detection system could improve the robustness and the coverage of the whole obstacle detection system as it could:

- to support the decision taken by an object detector by confirming its output (i.e., it will highlight an anomaly in the same position as an object detected by the object detector);
- to act as a *complementary* system as it would be capable of detecting unknown anomalies that the object detector would not be able to detect.

Taking into account this analysis, the idea is to extend the typical VBODS architecture as shown in Fig. 4.2. First, the **Rails Detection Module (RDM)** extracts the rails from the input frames; then, in parallel, the **Object Detection Module (ODM)** and the **Anomaly Detection Modules (ADM)** respectively detect objects and anomalies within the frame in output to the

RDM; hence, the **Obstacle Detection Module** merges the results from the ODM and the ADM to obtain one prediction for each obstacle; lastly, the **Distance Estimation Module** computes the distance(s) of the obstacle(s) from the train. These models are described below.

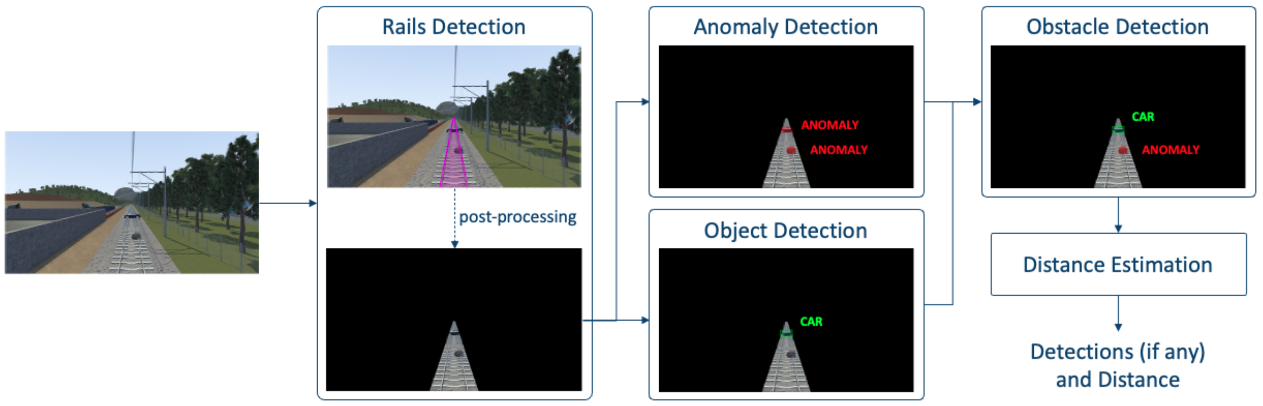


Fig. 4.2. An Architecture for Vision-Based Obstacle Detection.

4.2.1. Rails Detection Module (RDM)

As mentioned, this module is oriented at extracting rail tracks from video frames. Different solutions have already been proposed in the literature to address this specific task; some of them rely on traditional Computer Vision approaches (i.e., not based on AI, such as geometric approaches), however, their performance appears to be extremely dependent on light and weather variations, hence, they do not seem to be as efficient as DL solutions [4].

Among the works based on DL (and RGB cameras), reference [8] implements a ResNet50 DNN combined with fully convolutional layers to segment the rail tracks, while reference [9] proposes a solution based on SegNet [10]. Despite the fact that, in both cases, tests were performed on images collected from real scenarios, in our understanding, these studies seem not to take into account some *task-specific* characteristics (discussed in Section 4.3) when building the dataset. However, the results they achieve indicate that DL Semantic Segmentation (SS) approaches are suitable for rails detection.

Based on these results, we adopted a SS approach to implement the RDM by leveraging U-Net (originally conceived for medical image analysis [11]) instead of the aforementioned DNNs given the following reasons:

1. The solution proposed in [8], despite being quite effective, introduces some complexities in terms of implementability and network structure. Hence, inspired by [9], we decided to adopt a simpler DNN for SS and then, in case, apply some post-processing to understand whether a simple approach could still reach good results.
2. SegNet and U-Net share a similar basic encoder-decoder architecture. They involve some convolutional layers (encoder) to extract relevant features from the input image and some deconvolutional layers (decoder) to generate the output (e.g., a mask/label) based on the features extracted by the encoder. To improve results, both SegNet and U-Net introduce some connections between convolutional and deconvolutional layers; the concept is to pass to deconvolutional layers some additional information in order to allow them to produce more suitable segmentation masks. Without going deeper into the details, the main difference between SegNet and U-Net lies in the “typology” of the

passed information. In U-Net, deconvolutional layers receive the entire feature maps produced by the corresponding convolutional layers; in SegNet, deconvolutional layers receive an encoded version of the feature maps[2]. Given that, SegNet requires lower memory storage but it pays this advantage in accuracy [10]. Hence, we decided to opt for U-Net which, despite being more memory expensive than SegNet, has proved to be (a bit) more accurate.

As better discussed in the following sections, we collected a dataset related to *a single railway line* from a simulated environment we implemented by leveraging Mathworks' RoadRunner 3D Editor[3]. Hence, we implemented a SS approach based on U-Net (adapting a PyTorch implementation available on GitHub[4]), to segment the rail tracks. Lastly, post-processing is applied to "mask" the input frames and obtain images depicting only the rails without the background. These frames will be then processed, in parallel, by the ODM and the ADM.

4.2.2. Object Detection Module (ODM)

The *ODM* will implement a DNN trained in supervised mode, hence, it will be able to detect only the objects it has seen during the training phase.

To implement this module it would be possible to leverage, adapt, and/or improve the disparate supervised solutions for object detection that have been proposed in the literature [12, 13]; notably, some of them, e.g., detectors belonging to the Region-based CNNs (R-CNN) and YOLO families, have already been tested for object detection on rail tracks [4,14]. Important to mention, this is not a specific railway task, objects that can be detected on rail tracks (e.g., pedestrians, vehicles, animals, etc.) are actually common entities that can be found in different contexts. Hence, to train (more specifically, pre-train) object detectors, it would be possible to leverage multiple datasets available online like KITTI[5], MS COCO[6], and Pascal VOC[7].

Given the large attention that object detection has received during the last few years and the available results, in this document we do not pay much attention to the ODM module but mainly focus on the development of the anomaly detection module.

4.2.3. Anomaly Detection Module (ADM)

As just underlined, in the literature, there is plenty of studies discussing the realization of Object Detection architectures oriented at detecting and classifying objects. However, provided that it is nearly impossible to catalog all the possible obstacles, an anomaly detection approach that could highlight any kind of obstacles on the tracks (*based on data coming from a single camera*) may be useful to increment the robustness and the coverage of VBODSs. To that aim, the *ADM* will implement a DNN trained in Unsupervised mode which would potentially be capable of detecting any kind of anomaly within video frames.

Many studies are available that address anomaly detection within images. Tailored to the rail sector, reference [15] proposes an approach based on Autoencoders (AEs) to identify frames containing anomalies. As far as we understood, the model produces in output an anomaly

[2]Refer to [10] for further details.
[3]https://www.mathworks.com/products/roadrunner.html
[4]https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/image_segmentation/semantic_segmentation_unet
[5]https://www.cvlibs.net/datasets/kitti/index.php
[6]https://cocodataset.org/#home
[7]http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html

score indicating whether the frame contains an anomaly or not but it does not locate the anomaly within the frame itself. In addition, the score seems to be computed by comparing two reconstructions: one related to a rail image without obstacles and one related to the same image but with obstacles. Hence, provided that the identification of the anomaly within the frame can be done by comparing (at the pixel level) the two reconstructions, and despite achieving very promising results in identifying anomalous frames, conceptually, in our understanding, this approach expects that an obstacle-free image is available with which to compare the potentially anomalous image. In our opinion, this may introduce some complexities at run-time given the fact that another mechanism should be implemented to recognize the exact obstacle-free frame with which to compare the frame collected in real-time.

On the other hand, by looking at the automotive field, the authors in [16], inspired by [17], implemented a multi-step process to detect anomalies in road scenes. Briefly, they first segment the image in input through a SS approach based on AEs to produce a segmentation mask; then, they apply another DNN to *synthesize* the image (i.e., generate a reconstructed image) starting from the segmentation mask obtained at the previous step; lastly, they compare the original image with its synthesized version to obtain an anomaly map. These approaches (both [16] and [17]) are more in-line with the idea we would like to implement, however, the main problem is that, in our understanding, DNNs oriented at image synthesis (e.g., based on Generative Adversarial Networks [18] or ad-hoc DNNs [19]) typically take in input “complex” segmentation masks identifying multiple objects like trees, cars, signs, pedestrians, road, sky, and so on; hence, they learn how to reconstruct these parts of the images. In our case, after the RDM (oriented at producing a mask identifying the rail track as deeply discussed in Sections 4.3 and 4.4.1), images only contain a rail track and a black background.

Therefore, we tried to find simpler and more straightforward solutions that could allow us to produce a reconstructed image starting from the original image itself rather than its segmentation mask. Among them, with the aim of detecting anomalies within generic (mostly manufacturing-oriented) images, reference [20] discusses an AE combined with an SSIM-based loss function (SSIM-AE) which helped to improve detection performances, while reference [21] developed an architecture based on a Vector Quantised-Variational AE (VQ-VAE). If analyzed from a high-level perspective, both these studies applied the following reasoning: the DNN is trained on defect-free images only; then when at inference time a defective image is presented in input, the model reconstructs the image and the anomaly map is computed by comparing the reconstructed image with the original one. As better discussed in Section 4.4.2, inspired by both these works, we implemented the ADM by adopting a VQ-VAE with a loss function based on the SSIM; we were able to get some promising results (especially when it comes to images reconstruction), however, there is still room for improvements in the context of anomaly detection.

4.2.4. Obstacle Detection and Distance Estimation Modules

The Obstacle Detection module will simply merge the detection performed by the ODM and the ADM. The ADM should be able to detect also the obstacles detected by the ODM, therefore, assuming it is completely reliable, the results from the ADM could be used by this module to check the detection performed by the ODM. Then, for the obstacles which have been detected by both the ODM and ADM, the Obstacle Detection module will take into account the detection produced by the ODM (as it also identifies the class of the obstacle); otherwise, it will consider the detection produced by the ADM. Ideally, this module will produce in output a map that i) identifies the anomalous regions and ii) identifies and classifies the objects. Most likely, this module would not involve any AI functionality.

To conclude, the Distance Estimation Module is in charge of estimating the distance of the obstacle from the train. This specific task has been recently analyzed within the SMART project (DisNet [22]), however, despite being innovative, it seems to be class-dependent, i.e., to estimate the distance it takes into account the dimensions of a specific object in different circumstances. This means that it would be possible to estimate the distance of objects in output to the ODM. Differently, for what concerns anomalies, a suitable approach could be stereo triangulation [23], in case multiple cameras are involved.

4.3. Dataset Generation

One of the most relevant issues that we faced when addressing this case study is related to data availability. Indeed, to the best of our knowledge, besides some datasets like RailSem19[8], no suitable dataset is available online; where by **suitable** we mean datasets "taking into account some domain-specific and task-specific characteristics" and/or "depicting obstacles on rail tracks". In our view, domain-specific and task-specific characteristics are of extreme importance when developing AI applications in complex and safety-related contexts.

Typically, when it comes to developing ML approaches, it is good practice to evaluate their generalisation performance, which means estimating how the model would perform when dealing with situations it has not seen before. Usually, this is done by training the ML model with a training set and a validation set, and then testing the generalisation performance on data (the test set) it has not seen during the training phase. The motivations are related to the fact that it is not always possible to include in the training set all the possible scenarios the ML model would analyse once deployed due to the extreme aleatory of some environments. However, in railway and for this specific case study, it would be possible to reduce such an aleatory by building different datasets for each railway line (or even railway line sections). To better understand this concept, let's consider two railway lines: Line A does not include any structural "complexity" (it is composed of a single rail track), while Line B includes two lanes (i.e., two rail tracks) and a turnout. Assuming we want to build a DL model to detect rail tracks, is it convenient from a safety perspective to generate a unique dataset that encompasses frames related to both railway lines and build[9] a unique DL model capable of working on both these lines? Probably, it does not exist a 100% correct answer to this question, however, we think that generating different datasets (one for each railway line) could help to reduce the structural aleatory the DL model should face. The same reasoning could be applied at the AI system level; as depicted in Fig. 4.3, the reference architecture (and possibly also the specific DNNs used to implement it) would always be the same, but the systems (i.e., the combination of architecture and dataset) would be built ad-hoc for each railway line. Then, at the very end, the aleatory is "limited" to weather and light conditions and possible intrusions.

Notably, this aspect also leads to another advantage. If we focus on a specific railway line and collect multiple videos related to that line under different light and weather conditions, it would be highly probable that, at run-time, the AI system would analyse a scenario that is

[8] https://wilddash.cc/railsem19
[9] "Build" here means train the DL model on a specific dataset.

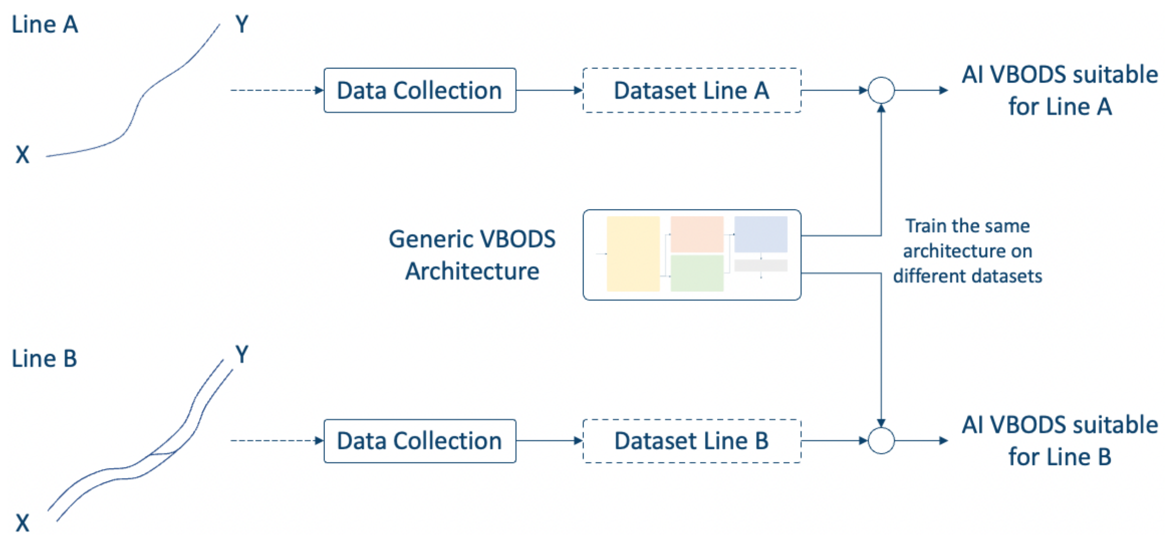

Line A
Y
X
Data Collection
Dataset Line A
AI VBODS suitable for Line A
Generic VBODS Architecture
Train the same architecture on different datasets
Line B
Y
X
Data Collection
Dataset Line B
AI VBODS suitable for Line B

Fig. 4.3. Building Different DL Models for Different Railway Lines

very similar (but, of course, not identical) to something it has already seen during the training phase. Worth highlighting, the concept expressed above is a peculiarity of the railway domain as the path the train will travel is pre-determined and constrained by the rail tracks. In automotive, for example, this concept can hardly be applied as a generic vehicle is "free" to travel any kind of road and it would be nearly impossible to build different DL models for all the possible paths a vehicle can travel.

Basing on this idea, we leveraged the MathWorks' 3D editor RoadRunner[10], conceived for the creation of road scenes, to build a railway scenario depicting a specific railway line from which to capture videos to train and test AI algorithms. Fig. 4.4 shows the scenario we built and some of the frames we extracted by simulating a single camera mounted in front of a train.

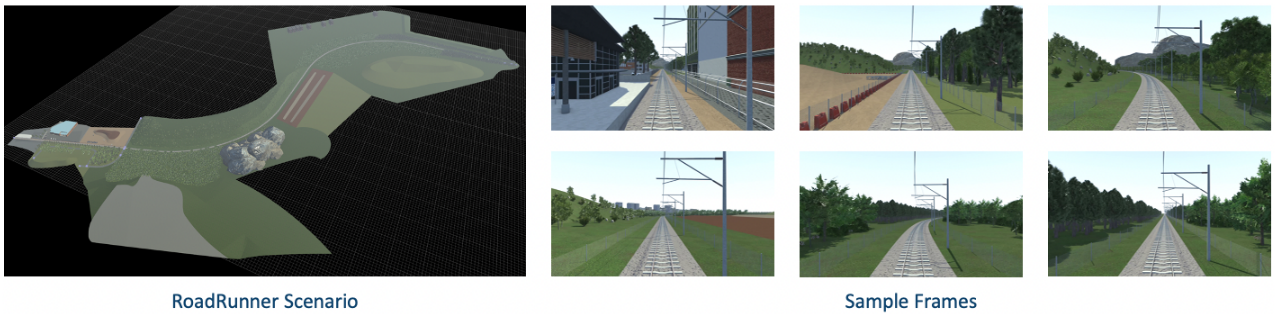


Fig. 4.4. Railway Scenario build in RoadRunner and related Frames

4.3.1. Data Preparation

The modules we implemented, i.e., the RDM and the ADM, *are sequential in our architecture* (see Fig. 4.2). Therefore, to train the DNN composing the ADM, we had to first process the frames extracted from the simulated scenario through the RDM.

In the following of this section, we will describe how the frames extracted from the videos collected through the simulated scenario were labelled to train the U-Net composing the RDM;

[10]https://www.mathworks.com/products/roadrunner.html

then, the particular realisation of the datasets, i.e., their specific decomposition in training, validation, and test sets (together with the application of data augmentation algorithms), will be discussed in detail for each module in Sections 4.4.1.1 (for the RDM) and 4.4.2.1 (for the ADM).

4.3.2. Data Labelling for the Rails Detection Module

Once frames have been extracted from the simulated scenario, we needed to label them. The main issue is that manual labelling is extremely time-consuming, therefore, we implemented a **semi-automatic labelling** approach based on **Transfer Learning** to label the majority of the frames as depicted in Fig. 4.5 and better discussed in the following.

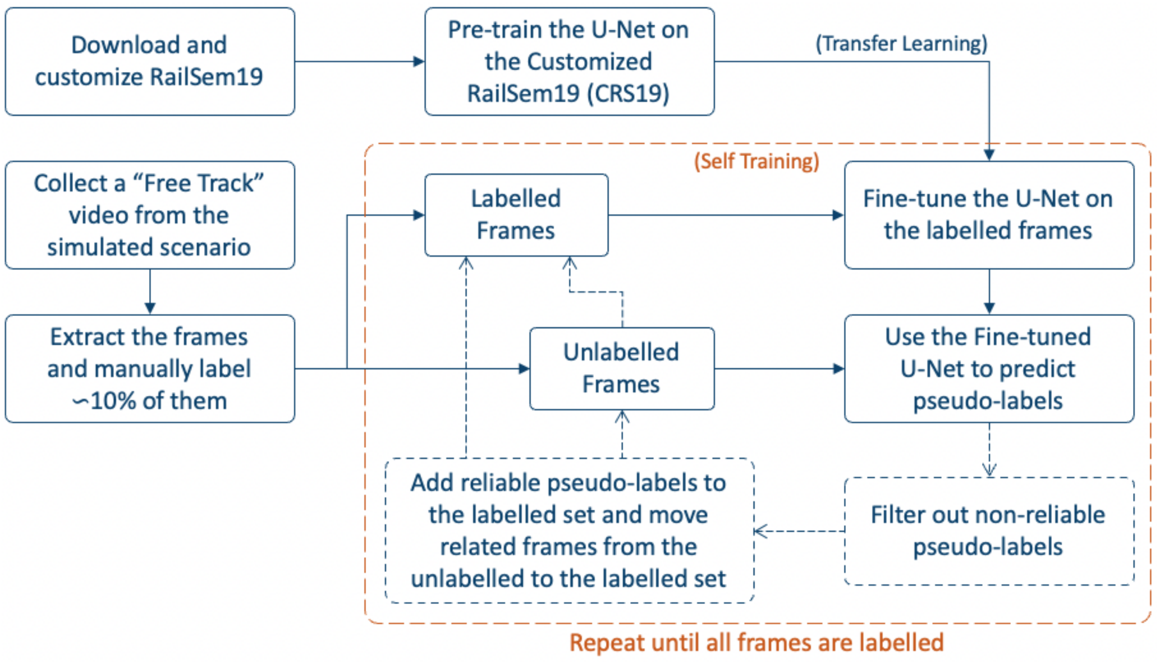

Download and customize RailSem19
Pre-train the U-Net on the Customized RailSem19 (CRS19)
(Transfer Learning)
Collect a "Free Track" video from the simulated scenario
(Self Training)
Labelled Frames
Fine-tune the U-Net on the labelled frames
Extract the frames and manually label ~10% of them
Unlabelled Frames
Use the Fine-tuned U-Net to predict pseudo-labels
Add reliable pseudo-labels to the labelled set and move related frames from the unlabelled to the labelled set
Filter out non-reliable pseudo-labels
Repeat until all frames are labelled

Fig. 4.5. Semi-automatic Labelling Process

4.3.2.1. Customisation of RailSem19

First, we leveraged the RailSem19 dataset to pre-train the U-Net introduced in Section 4.2.1. RailSem19 is composed of about 8500 images properly labelled for semantic segmentation. However, these images are associated with multiple labels (e.g., road, pole, traffic, tram track, and so on) and, in this case, we are only interested in the "rail track" label. Hence, we customised the RailSem19 by adapting some Python scripts available on GitHub[11] to filter out all the superfluous labels and all the images that did not contain any rail tracks. Then, we processed the obtained files through LabelMe[12] and other scripts[13] to obtain masks that can be used to train the U-Net. Worth emphasising, when dealing with semantic segmentation, a **"mask" is the label associated with a specific input image**; the "mask" itself is an image where, assuming a simple case with only one object to segment, a pixel assumes the value 1 if it is related to the object, or 0 if it is related to the background (i.e., all the objects that are

[11]https://github.com/xmba15/rail_marking/tree/master/scripts
[12]https://github.com/wkentaro/labelme
[13]https://github.com/wkentaro/labelme/tree/main/examples/bbox_detection#convert-to-voc-format-dataset

not of interest). In our case, masks' pixels are equal to 1 if they are related to the rail tracks, otherwise, their value is 0.

Eventually, we obtained a Customised RailSem19 (CRS19) dataset composed of 5519 images (and related masks) which we split into Training (3863 images; $\sim 70\%$), Validation (828 images, $\sim 15\%$), and Test (828 images; $\sim 15\%$) sets.

4.3.2.2. Extracting Frames from the Simulated Scenario

At the same time, we collected a "Free Track" Video (FTV) from the simulated scenario composed of 4143 frames (enumerated from 0 to 4142). For this proof-of-concept, we used only the first 2000 even frames (i.e., those with an even id from 0 to 3998). Of these 2000 frames, we manually labelled only 10% (200 frames) through LabelMe by taking into account a step of 10; i.e., we labelled the 10-th, the 20-th, the 30-th, and so on. These compose the "Initial Labelled Set" (ILS) of frames. Lastly, we split the ILS into Training and Validation sets with a 90-10% ratio (180 frames as training samples and 20 frames as validation samples). Also in this case, we used a specific sampling step instead of a random sampling; so, of the 200 labelled frames, the 10-th, the 20-th, and so on, belong to the Validation set, while the others to the Training set.

This schematic subdivision is necessary to give the DNN the possibility to learn the characteristics of the whole railway line and also to test the segmentation performances of the network along the entire railway line.

4.3.2.3. Pre-training U-Net on the Customised RailSem19

The CRS19 dataset has been used to pre-train the U-Net. The specific implementation values will be recalled afterwards in this document (Section 4.4.1), however, it is important to emphasise that for this pre-training phase we adopted the Adam optimizer with a learning rate equal to 10^{-4}, the Binary Cross-Entropy With Logits Loss (BCEWithLogitLoss[14]) as the loss function, batch size equals to 8; in addition, we set the maximum number of epochs to 100 but we also implemented an EarlyStopping criterion to stop the training if the loss evaluated on the Validation set did not improve for 10 consecutive epochs (this number of epochs - the 10 in this case - is named "patience"). Hence, the training stopped at the 48th epoch and the lowest loss on the Validation set was reached at the 38th epoch (best epoch). Note that images were scaled from 1920x1080 to 480x270 to be processed by U-Net because of GPU RAM and time constraints.

Fig. 4.6 shows the performance of the U-Net during the training phase in terms of loss (BCEWithLogitLoss) and three metrics namely Pixel Accuracy (PxAcc), Intersection over Union (IoU), and Dice Score. These metrics simply compare the target mask (the label) with the predicted mask (the output of U-Net); notably, the last two metrics have been computed by taking into account only the pixels related to the rail tracks. Table 4.3 shows the values indicated above related to the best epoch (38th) evaluated also on the Test set.

For the sake of knowledge, the PxAcc is not fully representative of the problem given the fact that it takes into account all the predicted pixels (the majority of which are background pixels), hence, its value is biased from the "background class", while we are interested in how the DNN is capable of predicting the "rail track class". Also, the Dice Score is computed as *two times the IoU divided by the total number of pixels contained in the two images that are being compared*, hence, also the IoU is quite superfluous as it is already considered

[14] https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

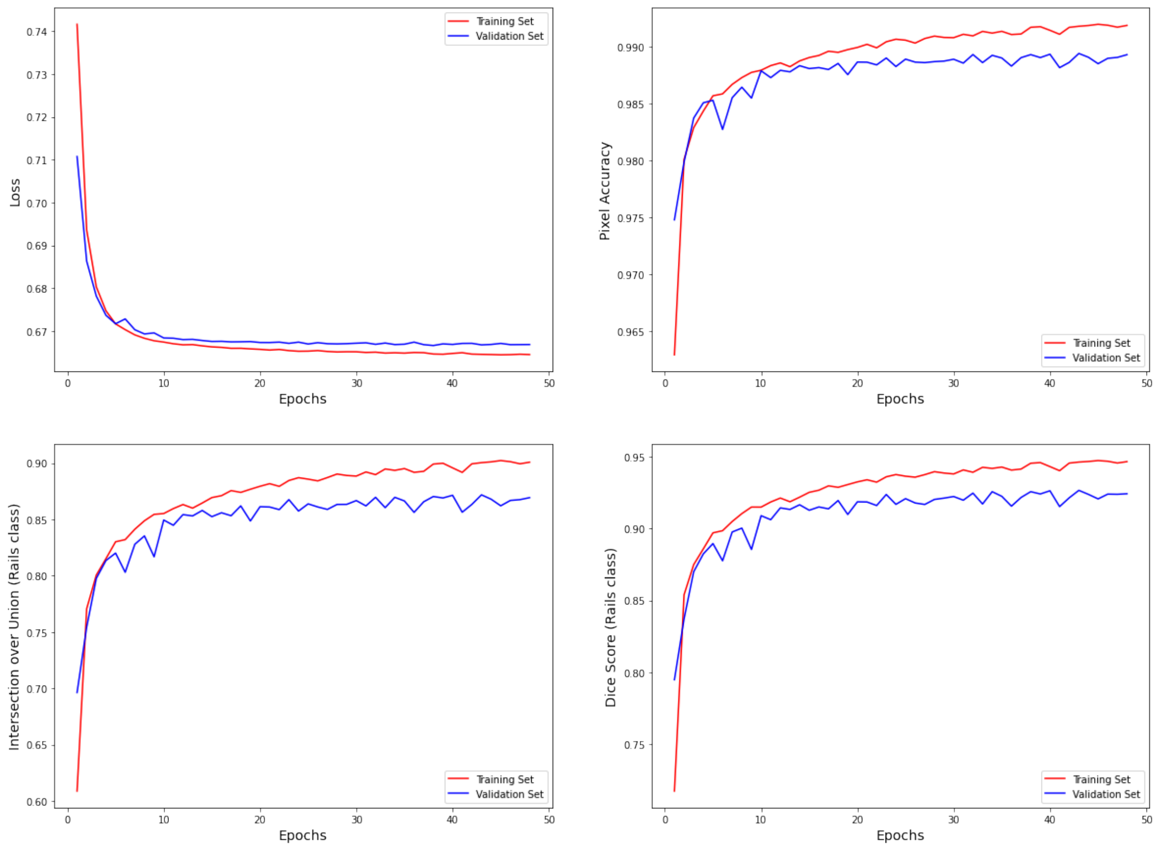


Fig. 4.6. Pre-training Losses and Metrics.

Table 4.3: Performance of U-Net at the best epoch

Set	***Loss***	***PxAcc***	***IoU***	***Dice***
Training	0.664641	0.9917	0.8992	0.9455
Validation	0.666619	0.9893	0.8703	0.9257
Test	0.666413	0.9885	0.8612	0.9183

in the Dice Score. Hence, in the following of this chapter, we will take into account only the Dice Score to evaluate the "accuracy" of U-Net. Important to mention, we modified the computation of the Dice Score by taking into account only the white pixels (i.e., those that are related to the rail tracks in the label and should be related to the rail track in the predicted mask):

$$Dice\ Score = \frac{2 \times Number\ of\ overlapping\ white\ pixels}{White\ pixels\ in\ the\ target\ mask + White\ pixels\ in\ the\ predicted\ mask}$$

where the "Number of overlapping white pixels" is simply the IoU computed by taking into account only the pixels related to the rail tracks (i.e., the white pixels).

4.3.2.4. Self-Training for Semi-Automatic Labelling

The main goal is to automatically label the 1800 frames extracted from the FT video discussed in Section 4.3.2.2 by leveraging the 200 manually labelled frames composing the ILS and the weights related to the best epoch (38th) obtained during the pre-training phase (to which we will simply refer as "pre-trained weights"). To that aim, **we implemented a Self-Training approach based on Transfer Learning**.

In the literature, there are different Self-Training methods [24–27]. In general, the base concept is to build a "teacher" which takes in input some labelled "real" data and produces some *pseudo-labels* for the unlabelled data; then, a "student" model is trained by combining data with hand-crafted labels and pseudo-labels.

Typically, by taking into account trivial tasks as the classification one, the trainer produces a pseudo-label with a specific confidence (e.g., the image contains a cat with a confidence of 99%). If such confidence is greater than a given threshold established a-priori, the label is assumed to be "potentially correct" and can be used to train the student. The main problem, in this case, is that we do not have a label for each image, instead, we have a label for each pixel (0 = background, 1 = rail tracks). Hence, to find a suitable solution for Self-Training, we formalised three main questions:

- How can we understand which is the best epoch to stop the training of the teacher to produce pseudo-labels?
- How can we understand if the teacher has correctly classified a specific pixel?
- How can we understand if the mask (pseudo-label) produced by the teacher is reliable/correct?

The first question is quite trivial; we adopted the same EarlyStopping criterion used during the pre-training (Section 4.3.2.3) but we set a patience of 20.

As for the two remaining questions, we adopted the following approach. For each pixel, the network produces a "confidence" score between 0 and 1. *The closer the score is to 0, the more confident the network is that it is a background pixel; conversely, the closer the score is to 1, the more confident the network is that it is a pixel related to the rails class*. Therefore, we set two thresholds: **bg_threshold = 0.05** and **rails_threshold = 0.95**. In particular: *if the prediction related to the i-th pixel is less than bg_threshold (0.05), we are quite confident that the pixel belongs to the background; in the same way, if the prediction is greater than rails_threshold (0.95), we are quite confident that it belongs to the rail track*. Then, to make the automatic labelling more robust, we needed a way to evaluate the correctness of the pseudo-label by considering the "uncertain" pixels (i.e., those for which bg_threshold $<$ prediction $<$ rails_threshold).

The solution we adopted is to run the model on both the original image and a horizontal-flipped image as depicted in Fig. 4.7. The prediction is made on both the original and the flipped image; then, the prediction of the flipped image is flipped again to bring it back to its original position; conceptually, we have two predictions for each pixel. Hence, the positional mean between the values of the two predicted masks is computed. Thus, the aforementioned thresholds, i.e., bg_threshold and rails_threshold, are applied. At this point, we set another threshold to evaluate the reliability of the generated pseudo-label based on the number of uncertain pixels. So, if the number of uncertain pixels is less than 0.001% (**trust_threshold**), the label is considered reliable and is added (together with the related frame) to the "labelled set".

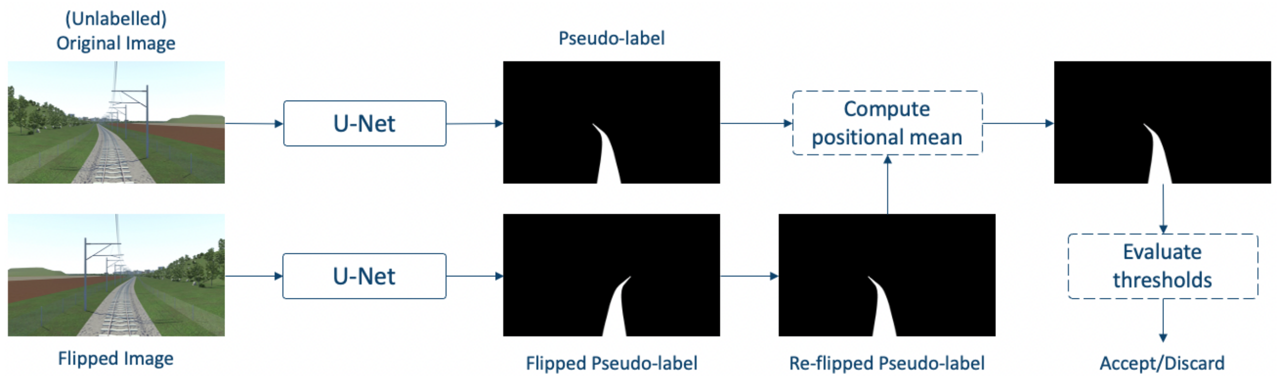


Fig. 4.7. Evaluation of Generated Pseudo-labels

The Self-Training process can go through several steps, depending on how many steps are required to label all the unlabelled data. At each phase: i) the U-Net (pre-trained on CRS19) is fine-tuned by using the Training and Validation sets of the “labelled set”, which is first augmented by flipping all the images and masks; ii) once the fine-tuning is ended given the EarlyStopping criterion, all the frames contained in the “unlabelled set” are processed by the fine-tuned U-Net as shown in Fig. 4.7; iii) all the reliable pseudo-labels and related frames are added to the labelled set by randomly splitting them between the training and the validation sets keeping the 90-10% ratio. In our case, the Self-Training process terminated after four steps as summarised in Table 4.4. The dataset used in Step 1 is the ILS defined above; also, the “$\rightarrow$” indicates the augmentation process. Notably, after the fourth step, there is no need to split the new samples into Training and Validation since the Self-Training has ended.

Table 4.4: Self-Training Steps

Step	**Fine-Tuning**					**Automatic Labelling**			
	Dataset		**Results on Validation Set**						
	Training	*Validaiton*	*Lowest Loss*	*Epoch*	*Dice Score*	*New Labelled Frames*	*Added to Training*	*Added to Validation*	*Frames to Label*
1	180 $\rightarrow$ 360	20 $\rightarrow$ 40	0.682783	29	0.9648	248 / 1800	224	24	1552
2	404 $\rightarrow$ 808	44 $\rightarrow$ 88	0.682663	44	0.9838	1521 / 1552	1369	152	31
3	1773 $\rightarrow$ 3546	196 $\rightarrow$ 392	0.682519	53	0.9961	10 / 31	9	1	21
4	1782 $\rightarrow$ 3564	197 $\rightarrow$ 394	0.682493	95	0.9996	21 / 21	/	/	0

Downstream this process we have labelled all the 2000 frames extracted from the FTV.

Is Transfer Learning needed? Before continuing with the implementation of the models, it is important to underline the contribution of Transfer Learning. To this aim, Fig.4.8 compares the performance of U-Net in case of fine-tuning (i.e., loading the pre-training weights) and training from scratch (i.e., without loading them) by taking into account the augmented ILS; performances are evaluated on the Validation set.

Considering the chart on the left hand, when training the network from scratch, the training completes all the 100 epochs and the EarlyStopping criterion is not triggered; the lowest loss on the Validation set is registered at the 100th epoch. This practically means that the training should continue further until a sub-optimal solution is reached. Differently, when fine-tuning the network starting from the pre-trained weights, the EarlyStopping is triggered

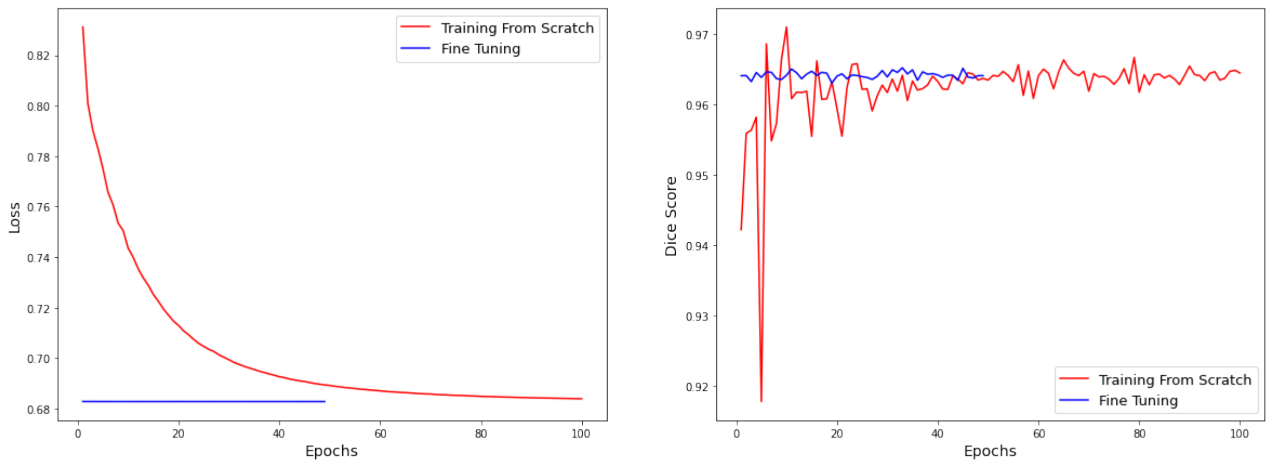


Fig. 4.8. Comparison of U-Net Performance on ILS Validation Set

and the lowest loss on the Validation set is registered at the 29th epoch. As for the Dice Score (the chart on the right hand), it is possible to note that in case of fine-tuning the Dice Score is centred around the value 0.964, while, in the other case, the performances are characterised by consistent oscillations.

The scales between the training from scratch and the fine-tuning are quite different, therefore, for the sake of completeness, we report in Fig. 4.9 the performance of U-Net in case of fine-tuning. It is possible to note some oscillations but, this time, they are limited to very narrow intervals.

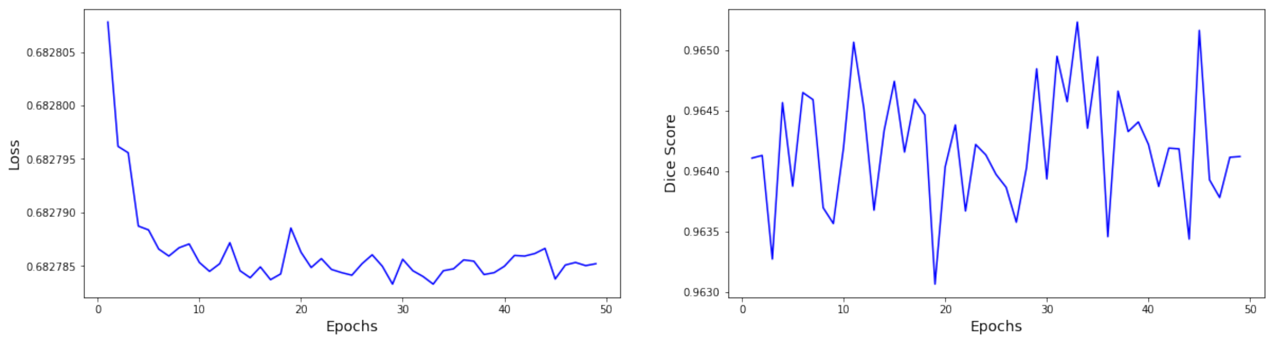


Fig. 4.9. U-Net Performance on ILS Validation Set in Case of Fine-Tuning

To conclude, Table 4.5 compares the performance of U-Net by taking into account the best epochs of the two training.

Table 4.5: Comparison of U-Net Performance on ILS Validation Set (best epochs)

Training Mode	**Lowest Loss**	**Epoch**	**Dice Score**
From Scratch	0.6839	100	0.9644
Fine-tuning	0.6827	29	0.9648

To answer the question object of this paragraph, the evidence above shows that Transfer Learning can contribute to makeing the network more stable (in terms of oscillations in the

Dice Score trend) and achieve more or less the same sub-optimal solution but within a reduced number of epochs reducing, de facto, the time needed to train the DNN.

4.4. Training and Validation

This section discusses the training and validation phases of the DNNs that have been chosen to implement the modules of the VBODS in Fig. 4.2.

4.4.1. Rails Detection Module

As indicated in Section 4.2.1, the RDM aims at detecting rail tracks within video frames. To implement this module, we adopted the same U-Net discussed in the previous section. However, before proceeding with the technical implementation of the network, it is important to underline how data were pre-processed.

4.4.1.1. Data Preparation

Starting from the 2000 labelled frames (which are related to free tracks, i.e., do not contain any obstacle) obtained downstream the Self-Training process described in Section 4.3.2, we applied some **Data Augmentation** by leveraging the Automold Python library[15]. Important to mention, the Automold library offers different transformations from which we selected only those summarised in Table 4.6. Notably, most of these transformations introduce some randomness, for example: the “Rain” produces drops with random intensity and angulation; the “Bright” increases the brightness of the pixels according to a random factor; and so on.

Table 4.6: Types of Data Augmentation Transformations

Transformation	Acronym	Description
Bright	BR	Randomly increase the brightness of the image
Dark	DA	Randomly decrease the brightness of the pixel
Rain	RN	Simulate rain by randomly adding drops with different intensities and angulation
Shadow	SH	Simulate shadows by adding random semitransparent masks on the image
Sun Flare	SF	Simulate sun flares by adding concentric and random circles in the image

The frames we extracted from the simulated scenario have a resolution of 2080x1170 pixels, therefore, they are quite heavy in terms of memory. Given some constraints related to the storage and GPU RAM of the machine we used to train U-Net, we had to reduce the number of frames. Hence, we selected only 1600 frames among the 2000 (distributed to maintain information related to the whole railway line) that we used as “Original (OR) frames” to perform the augmentation. Then, for each frame, we applied all the transformations indicated in Table 4.6. Fig. 4.10 shows the effects of the transformations in relation to one of the original frames; worth underlining, the mask related to the OR frame (“Common” Mask in the figure) has been used for all the transformed frames retrieved from it.

Then, we subdivided the Augmented Dataset into Training, Validation, and Test sets as shown in Table 4.7.

Practically, we got 1600 frames for each transformation plus the 1600 OR frames. Notably, the Training-Validation-Test splitting ratio (80-10-10%) was kept both at the dataset level and

[15] https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library

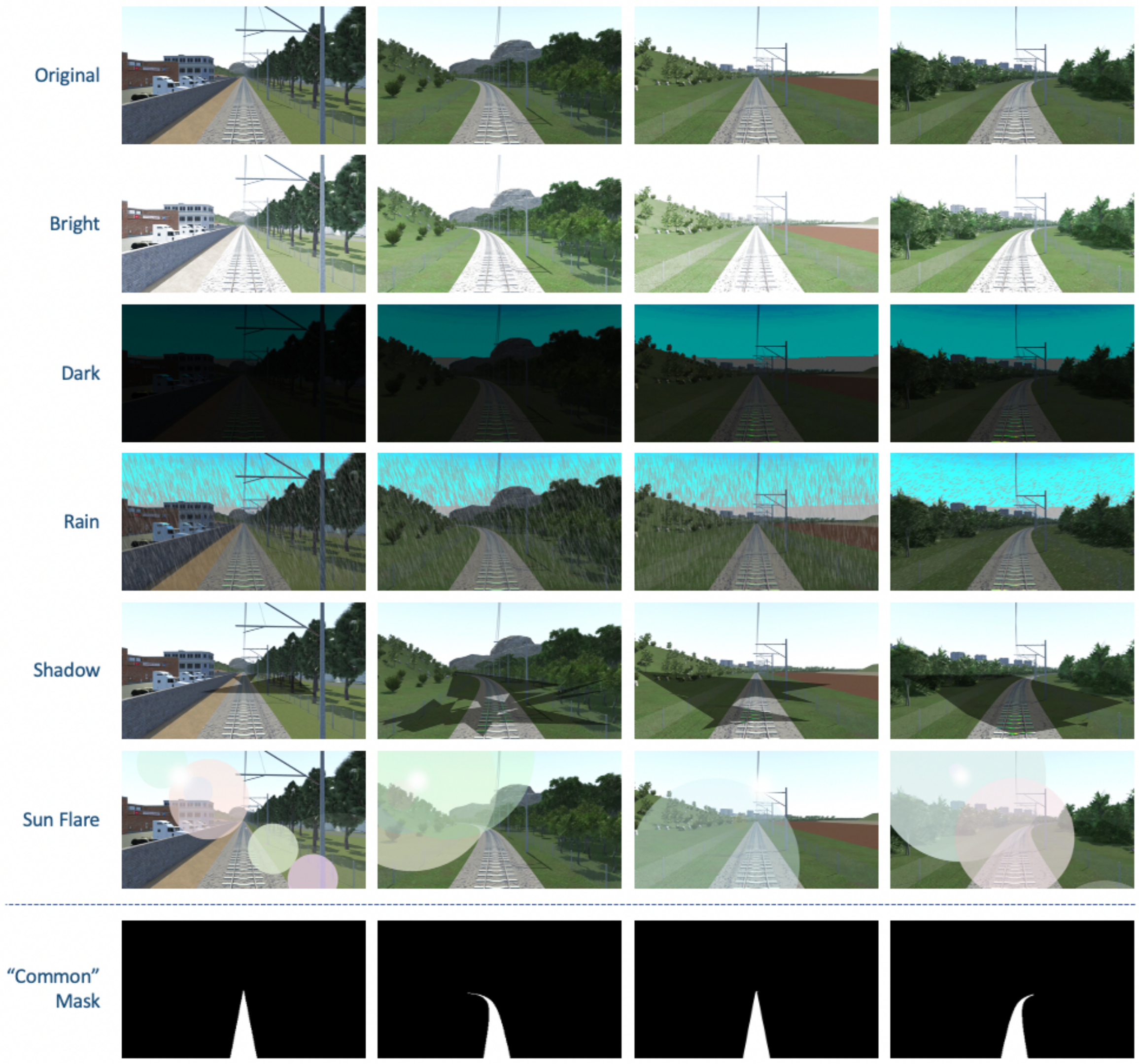


Fig. 4.10. Examples of Transformations

Table 4.7: Augmented Dataset Subdivision into Training, Validation, and Test Sets

	Transformation						
Set	***OR***	***BR***	***DA***	***RN***	***SH***	***SF***	**Total**
Training	1280	1280	1280	1280	1280	1280	7680 (80%)
Validation	160	160	160	160	160	160	960 (10%)
Test	160	160	160	160	160	160	960 (10%)
Total	1600	1600	1600	1600	1600	1600	9600

at the transformation level. Furthermore, this splitting took also into account the chronological order of the frames. As an example, considering the OR frames: the first four frames were added to the training set; then, the fifth was added to the Validation set; hence, again, the subsequent four frames were added to the Training Set; and, lastly, the tenth frame was added to the Test Set. This repeats cyclically and the starting point of this "cyclical split" randomly changes from transformation to transformation. Fig. 4.11 visually shows this process.

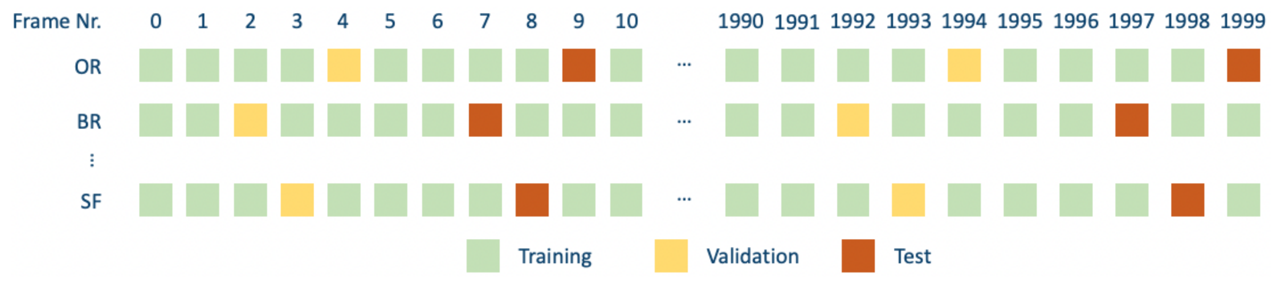


Fig. 4.11. Splitting Frames into Training, Validation, and Test Sets

The motivation behind this split is that by randomly splitting the samples, in the Validation or Test sets, we could have had a non-balanced distribution of the frames in relation to the railway line; e.g., we could have had 600 frames related to the first half of the line and 360 frames related to the second half. Instead, by "controlling" the split, we can evaluate how the network performs along the whole railway line.

4.4.1.2. Training, Validating, and Testing U-Net

To implement the RDM, we trained the U-Net by taking into account the Training and Validation sets just presented and adopted the following procedure. First, we resized the images from 2080x1170 to 480x270, then, we loaded the pre-trained weights obtained as described in Section 4.3.2.3; notably, 480x270 is also the dimension of the mask U-Net produces in output. Hence, we set the network's parameters and hyperparameters as reported herein:

- Adam Optimizer with Learning Rate equals to 1e-4;
- BCEWithLogitsLoss as the loss function;
- Batch size equal to 16;
- Maximum number of epochs equal to 100 (for the feasibility of the training - each training epoch takes about 12 minutes to be completed);
- Early Stopping criterion with patience equal to 10 (i.e., if the loss computed on the validation set does not decrease for 10 epochs the training stops).

The U-Net performances during the training phase in terms of loss and Dice Score are reported in Fig. 4.12. The lowest loss on the Validation set was reached at the 72nd epoch (best epoch), hence, given the Early Stopping criterion, the training stopped at the 82nd epoch. At the best epoch we had, as for the Training set, a loss of 0.682516 and a Dice Score of 0.9954, while, concerning the Validation set, the loss was equal to 0.682545 and the Dice Score was equal to 0.9907.

Hence, we saved the weights obtained at the 72nd epoch and proceeded with the following tests.

Testing U-Net on the Test Set. First, we tested the performance of U-Net on the Test set defined in the section above. Fig. 4.13 reports the average Dice Score and loss the network achieves on the Test set as a whole (All) and for each sub-class (Original, Bright, Dark, and so on). Notably, there is no transformation on which the U-Net performs extremely better or extremely worse compared to the others; the "Rain" transformation may appear a bit "problematic", however, by looking at the scale of the y axes, the Dice Score is always above the 0.9988 threshold. For the sake of completeness, Fig. 4.14 reports the masks

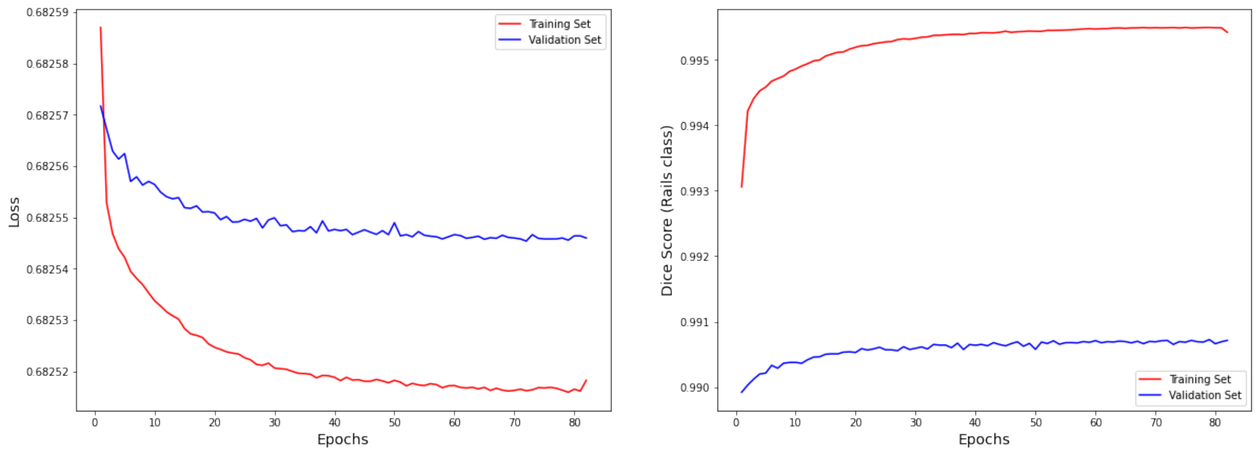


Fig. 4.12. Training U-Net on the Augmented Dataset

generated by U-Net for some frames of the Test set.

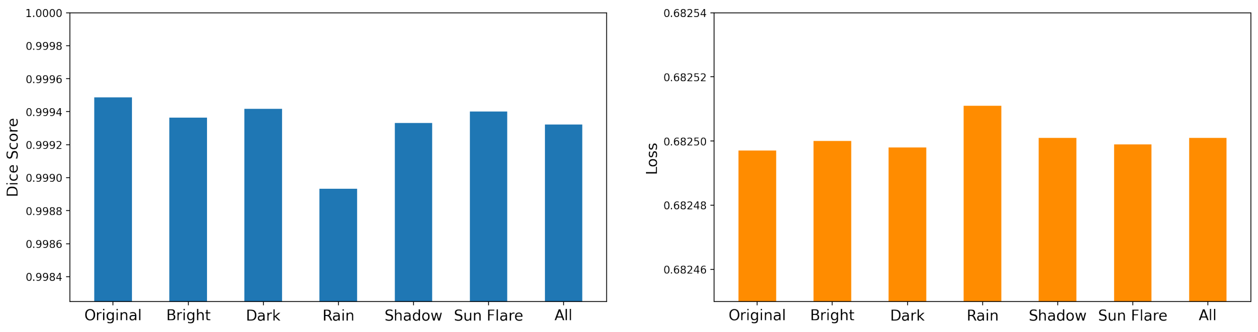


Fig. 4.13. Average Dice Score and Loss of U-Net on the Test Set

Given these results, we can conclude that U-Net is quite suitable for such a task (in this scenario, with this specific dataset) and, in case no obstacles are present or rail tracks, it does not require any post-processing to adjust the predicted mask (as, instead, happened for SegNet in [9]).

Testing U-Net on Frames with Obstacles. The dataset used so far contains only “Free Track” samples, i.e., there are no obstacles within the frames. Hence, to deeper test the U-Net, we put a vehicle on the rail tracks in the simulated scenario and collected a new video we named “With Car” Video (WCV). Then, we processed this video with the U-Net. By performing this test, we noticed that the obstacle introduces some noise in the predicted mask, especially when it is close to the train (see Fig. 4.15). *Worth underlining, the fact that U-Net encounters these difficulties may also be an indicator of the fact that there may be an obstacle on the rail tracks.* However, it would be better not to rely (only) on this sort of miscalculation to detect obstacles as it could not be robust in time. Therefore, we needed to apply some post-processing to avoid this issue as described in the following.

4.4.1.3. U-Net Post-Processing

First of all, it is necessary to emphasise an important aspect of the RDM. The U-Net implemented above is capable of detecting only the portion of the image which is exactly between

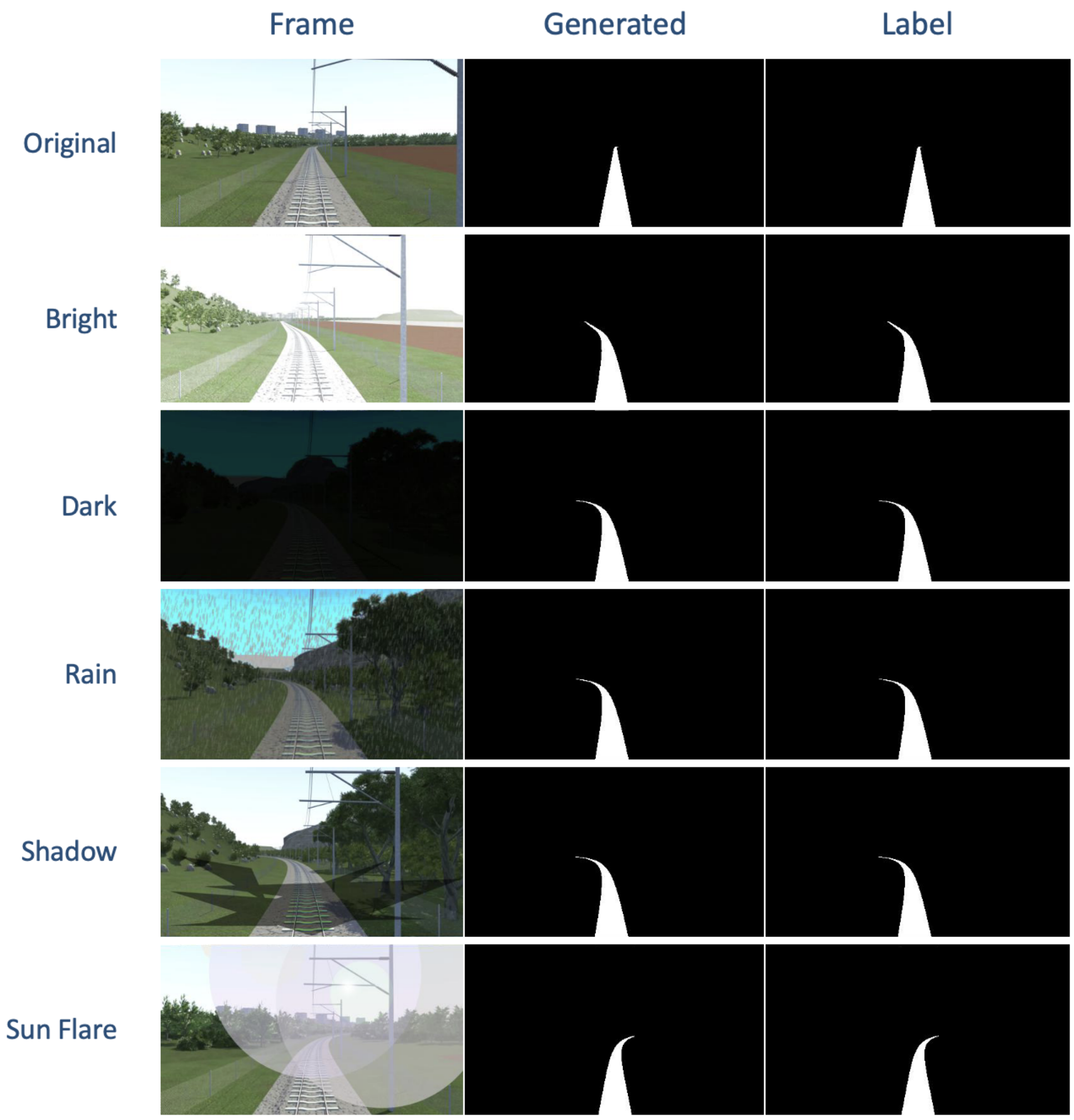


Fig. 4.14. Masks Generated by U-Net for Test Frames

the two rails. However, there may be cases in which obstacles are not positioned exactly on the rail tracks but are close enough to them that they can be hit by the shape of the train. Hence, in addition to the noise reduction, with this post-processing, we also aim at enlarging the predicted mask to let it include the sides of the rail tracks (see Fig. 4.16).

Basing on what discussed so far, the steps of the post-processing are discussed in the following.

Step 1: Filter out Macro Noise from the Predicted Mask. This is done by taking into account the domain-specific characteristics of the task we are performing. Basically, we built a network by considering frames coming from the same railway line and also the WCV comes from the same railway line. The concept is that it is possible to identify a region of the frame where the rail tracks will be located (rails region). To identify this region, we merged

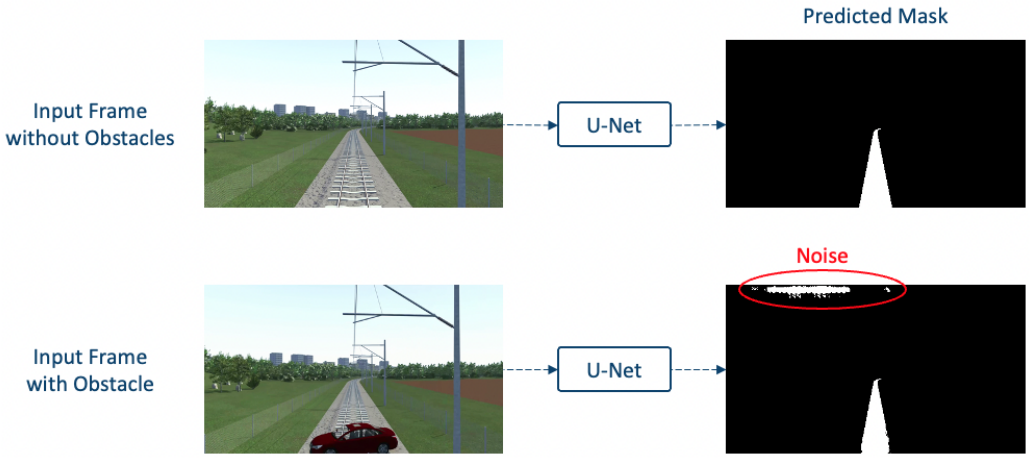
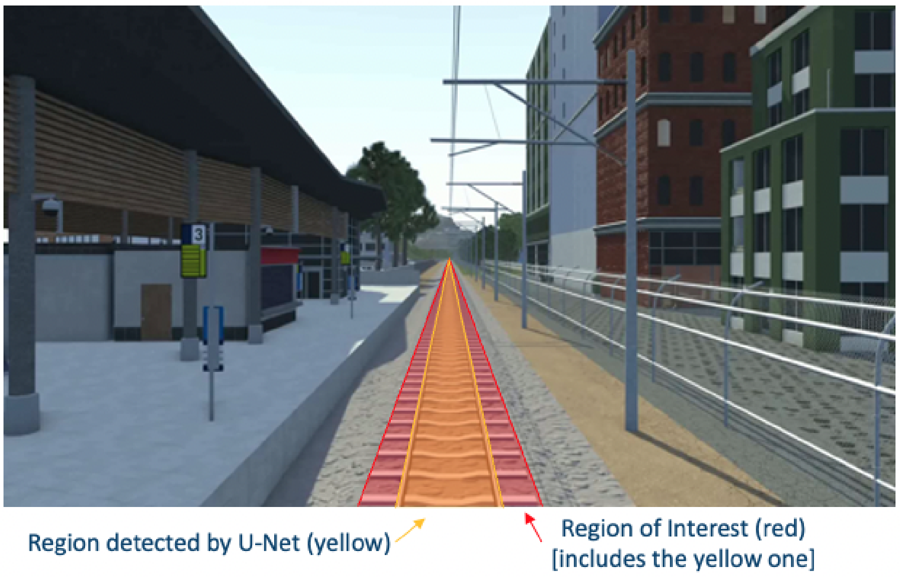


Fig. 4.15. Noise in the Predicted Mask Introduced by the Obstacle



Region detected by U-Net (yellow)
Region of Interest (red) [includes the yellow one]

Fig. 4.16. Enlarging the Mask produced by U-Net

all the masks produced by U-Net on the WCV (by simply performing a bit-wise and between them). Then, we computed a "cleaning mask" that we used at run-time to filter out the macro noise by simply performing a bit-wise and between the prediction and the cleaning mask. Fig. 4.17 graphically shows what is performed within this step.

Step 2: Filter out any other Micro Noise. In order to prevent any other kind of noise, once the mask has been cleaned, we also applied some functions of the *OpenCV* library[16] to extract the *polygons* related to the white-pixel clusters contained in the mask. Trivially, we assume that the cluster related to the rails would be the larger cluster in the mask (after Step 1), hence, all the other clusters can be eliminated. This is done by considering the polygon related to the larger cluster and re-drawing the mask by filling this polygon. The process is summarised in Fig. 4.18.

[16] https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

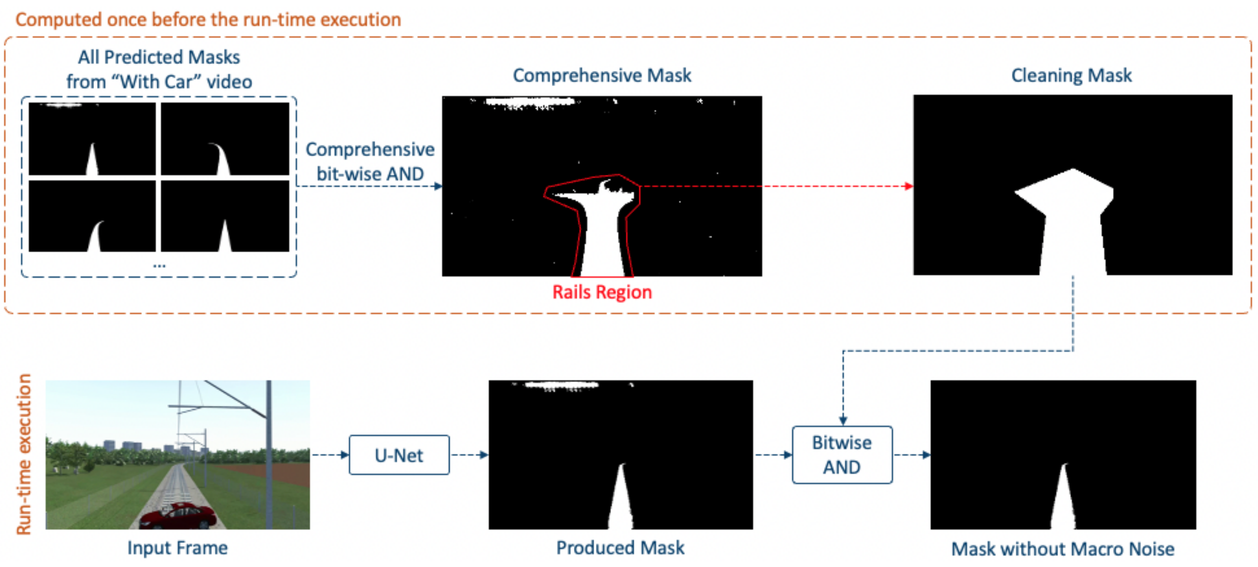


Fig. 4.17. Filtering out Macro Noise from Predicted Masks

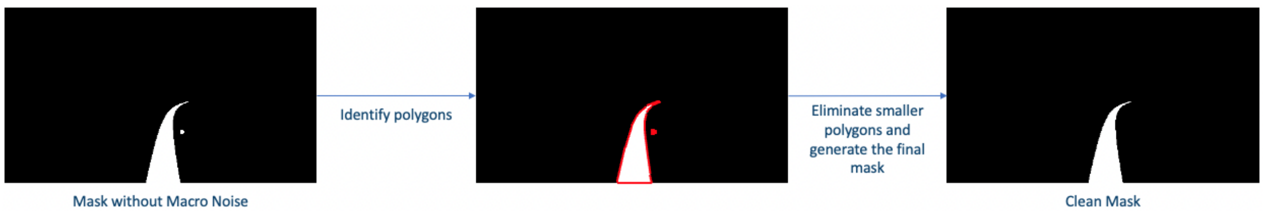


Fig. 4.18. Filtering out Micro Noise

Step 3: Enlarging the Rail Track Mask. The enlargement of the mask must be done by considering the perspective. Hence, we considered the vertexes of the polygon extracted at the previous step and selected the vertex representative of the "farthest" point (in perspective) of the rail tracks. This point is used as the pivot for two or four rotations (1 or 2 left rotations and 1 or 2 right rotations depending on the accentuation of the curve) which, merged with the original mask, produce what we named "enlarged mask". Fig. 4.19 shows what is performed within this step.

To conclude, by applying U-Net and the post-processing algorithm to any frame collected through the simulated scenario, we should be able to extract the region related to the rail tracks. Fig. 4.20 summarises the rails detection process performed by the RDM by applying U-Net and the post-processing algorithm.

Notably, *the last step can be bypassed* if the DNN is trained on frames whose label masks already include the sides of the rail tracks; we decided not to adopt this strategy for a simple reason. In our simulated scenario, the ballast under the rail tracks is visible and separated from the rest of the environment by means of sharp and well-defined lines; hence, it would have been trivial to train the U-Net to detect the whole ballast (including the tracks) as this region has graphics characteristics (which DNNs can easily learn) that are almost completely different from the rest of the environment. However, in real environments, this is not always true; there may be cases in which only the rails (not even the sleepers) are "clearly" visible. In these scenarios, it would still be possible to build a DNN to detect the whole region of

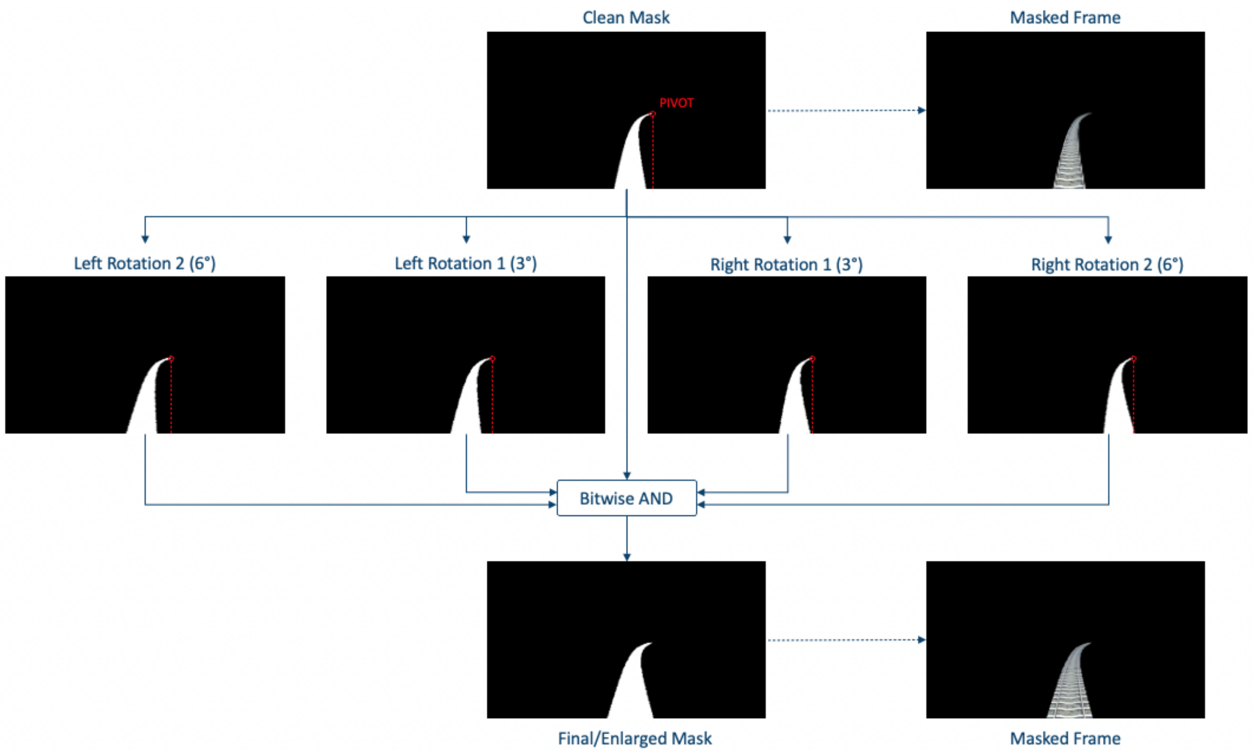


Fig. 4.19. Enlarging the Rail Track Mask

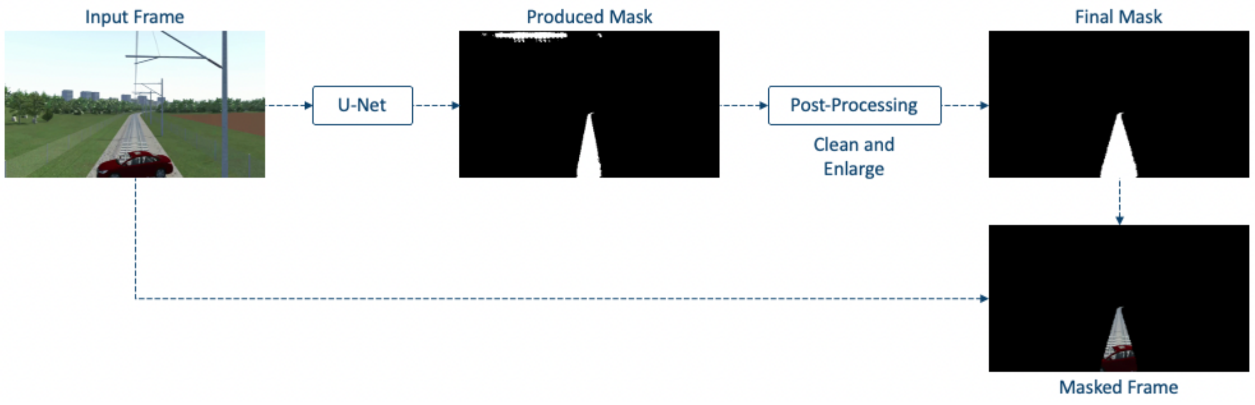


Fig. 4.20. Extracting the Rail Track by Applying U-Net and the Post-Processing Algorithm

interest (rail tracks + sides), but, in case it would not achieve good results, we have shown an alternative to first detect the rails (task that could be easier) and then enlarge the mask.

4.4.2. Anomaly Detection Module

To implement this module, we adopted an unsupervised approach. The main idea was to train a DNN on non-anomaly data to allow it to the learn characteristics of “normal” (non-anomalous) scenarios. Then, when an anomalous image (i.e., with an obstacle) is presented in input, the anomaly will be (theoretically) poorly reconstructed and thus highlighted. As introduced in Section 4.2.3, we implemented this module by taking inspiration from the studies conducted in [20] and [21].

4.4.2.1. Data Preparation

The non-anomaly dataset we used to train and validate the DNN was obtained by processing the FTV through the RDM module as discussed above. In addition, given some properties of the DNNs we used (presented below), we cropped the frames from 480x270 to 256x256. Hence, all the images in this dataset, which we will refer to as “Non-Anomaly Dataset (NAD)”, are characterised by a black background as depicted in the bottom right frame in Fig. 4.20 and have a dimension of 256x256. In addition, we applied the same data augmentation techniques reported in Table 4.6 to obtain the Augmented NAD (A-NAD). Tests were performed by considering both the NAD and A-NAD. In both cases, we split the dataset into Training (80%), Validation (10%), and Test (10%) sets as reported in Table 4.8. Notably, the A-NAD considers only 1500 of the original frames (i.e., those contained in NAD) because of some time and GPU RAM constraints. Basically, the larger the dataset the more time the network will take to be trained, especially if it would not be possible to adopt “larger” batch sizes as discussed in Section 4.4.2.3.

Table 4.8: Non-Anomaly Datasets for Unsupervised Anomaly Detection

Set	**NAD**	**A-NAD**						
		OR	**BR**	**DA**	**RA**	**SH**	**SF**	**Total**
Training	3315 (80%)	1200	1200	1200	1200	1200	1200	7200 (80%)
Validation	414 (10%)	150	150	150	150	150	150	900 (10%)
Test	414 (10%)	150	150	150	150	150	150	900 (10%)
Total	4143	1500	1500	1500	1500	1500	1500	9000

4.4.2.2. Proposed Anomaly Detection Framework

As mentioned, we leveraged the studies conducted within [20] and [21] to build the DNN for anomaly detection.

First, we tried to adapt the network proposed in [20] (named SSIM-AE) by leveraging the implementation the authors shared on GitHub[17]. The main problem we encountered in using this SSIM-AE is that, during the training, the architecture reached a sort of plateau; practically, given that the rails occupy only a small portion of the image, if the DNN predicts a full-black image, the loss is quite low and remains constant over time. Therefore, inspired by the work proposed in [21], we decided to study the behaviour of the DNN they adopted (a VQ-VAE [28]). As shown in the following, we were able to get some promising results (especially when it comes to image reconstruction), however, there is still room for improvements in the context of anomaly detection.

By fusing the concepts expressed in [20] and [21], we implemented a VQ-VAE network with a SSIM-based loss function (SSIM-VQ-VAE) exploiting some PyTorch implementations available on GitHub[18,19] as starting points.

Before diving into the implementation details, it is important to highlight the main difference between VQ-VAEs and traditional AEs, as it seems to be the key factor that makes VQ-VAEs

[17]https://github.com/plutoyuxie/AutoEncoder-SSIM-for-unsupervised-anomaly-detection-

[18]https://github.com/nadavbh12/VQ-VAE

[19]https://github.com/Po-Hsun-Su/pytorch-ssim

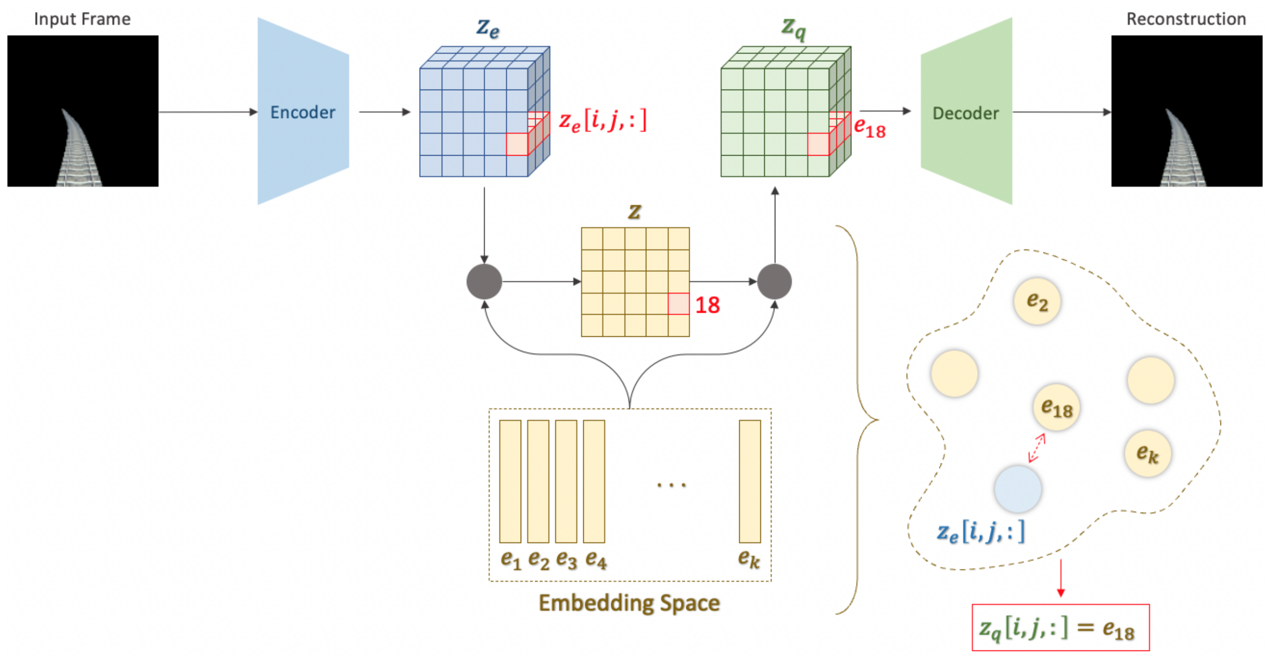


Fig. 4.21. Hihg-Level Architecture and Functioning of VQ-VAE, excerpted from [28].

suitable for this task, while AEs do not perform properly. As shown in Fig. 4.21, the output of the Encoder (Z_e in the figure), instead of being processed directly by the Decoder (as it would happen in traditional AEs), is first processed by an embedding algorithm which related *Embedding Space* is automatically learnt during the training. To be more specific:

1. The Encoder processes an image and produces in output a block of features named Z_e.
2. Then, this block of features is processed by an embedding algorithm (integrating the concept of nearest neighbour look-up [28]). Practically, the Embedding Space is composed of a finite number (K, established a-priori) of "centroids" (e_i in the figure) which are updated during the training. Notably, each centroid (or array in the embedding space) is associated with an ID ranging from 1 to K.

   Hence, Z_e is processed by the embedding algorithm that substitutes the Z_e arrays with the related nearest centroids to produce Z_q. Important to note, the embedding algorithm works on "depth arrays". Considering Z_e as a 3D matrix with dimensions W, H and D (where D is actually the number of feature maps in output to the last layer of the encoder which, in our case, is equal to 128), the algorithm substitutes each $Z_e[i, j, :]$ (where i goes from 0 to W-1, and j goes from 0 to H-1) with the related nearest centroid (e_{18} in the figure).
3. Lastly, Z_q is processed by the Decoder which will produce the output (i.e., the reconstruction of the image given in input to the whole network).

Worth highlighting, the embedding algorithm also produces an *index map* (*Z*, in Fig. 4.21). This map contains, in each position (i, j), the ID of the centroid that $Z_e[i, j, :]$ should be substituted with. Hence, for each image, the algorithm produces a single index map which could be leveraged for anomaly detection since, by modifying a value in this map, we can practically modify the image that the Decoder will produce in output; reference [21] adopts this strategy. Conversely, as better discussed below, we only used the VQ-VAE to reconstruct the image in input, and the anomaly was detected through the SSIM. The "discretization" introduced by the Embedding Space seems to allow for better results with respect to traditional AEs. Practically, only a few vectors (e_i) are used to represent the black background in the latent space. All the remaining, instead, are used to reproduce the rail tracks.

The original implementation of VQ-VAE [28] uses a composed loss function where one of the components, without going deeper with the details, leverages the Mean Squared Error (MSE) computed between the pixels of the image in input and the pixels of the reconstructed image. We substituted this MSE-based component with a SSIM-based one for two main reasons: i) instead of simply computing the difference between pixels, this metric takes into account different characteristics of the images (namely luminescence, contrast, and structure [29]) which allows for a more meaningful comparison; ii) the anomaly map is computed directly within the SSIM-VQ-VAE. The second reason comes from the fact that the Mean SSIM score (MSSIM, which we used in the loss function) is obtained by comparing the two images region by region (through convolutions) and averaging these comparisons. These convolutions produce a score for each pair of pixels of the two images under analysis, hence, the anomaly map is nothing more that the visual representation of the results of these convolutions. For the sake of knowledge, the MSSIM score is a value in [0, 1], where 1 means that the two images are practically identical, while 0 means that the two images are completely different; hence, the SSIM-based component of SSIM-VQ-VAE's loss function is computed as $1 - MSSIM$.

4.4.2.3. Training, Validating, and Testing SSIM-VQ-VAE

To train the SSIM-VQ-VAE, we set the following parameters and hyperparameters:

- Embedding Space with 512 centroids.
- Loss function based on the SSIM. The computation of the SSIM requires setting a filter to weight the computations; we adopted an 11x11 circular-symmetric Gaussian filter (as in [29]).
- Adam Optimized with starting Learning Rate equal to 1e-4.
- Batch size equal to 6. The network is quite "heavy" in terms of memory occupation; this is also due to the fact that, in addition to the network's weights, also the embedding space must be saved in memory. This is also the reason why we adopted a reduced set of frames from the NAD to build the A-NAD (as discussed in Section 4.4.2.1); given that the batch size could not be greater than 6, increasing the dataset means substantially increasing the training time.
- Maximum number of epochs set to 100.
- Early Stopping criterion with patience equal to 20.

We trained the network on both the NAD and A-NAD to study the effect of data augmentation. Figures 4.22 and 4.23 report the performance of the network trained on NAD and A-NAD separately in terms of losses and SSIM scores. In this case, the SSIM value, besides being used in the loss function, is also used to evaluate the reconstruction capability of the DNN since the higher the SSIM score, the better the network reconstructs the images. Notably, as specified in Table 4.9 (discussed later), in case of training on NAD, the lowest loss on the Validation set was reached at the 45th epoch (hence, the training stopped at the 65th epoch); differently, in case of training on A-NAD, the lowest loss on the Validation set was achieved at the 52nd epoch and the training stopped after 72 epochs. To make trends comparable, Figures 4.22 and 4.23 report the values of the loss and the SSIM score up to the 65th epoch in both cases. Actually, because of some accentuated oscillations at the beginning of the training on the A-NAD, the third chart in both the figures reports the trends from the 10th to the 65th epoch (for the sake of readability).

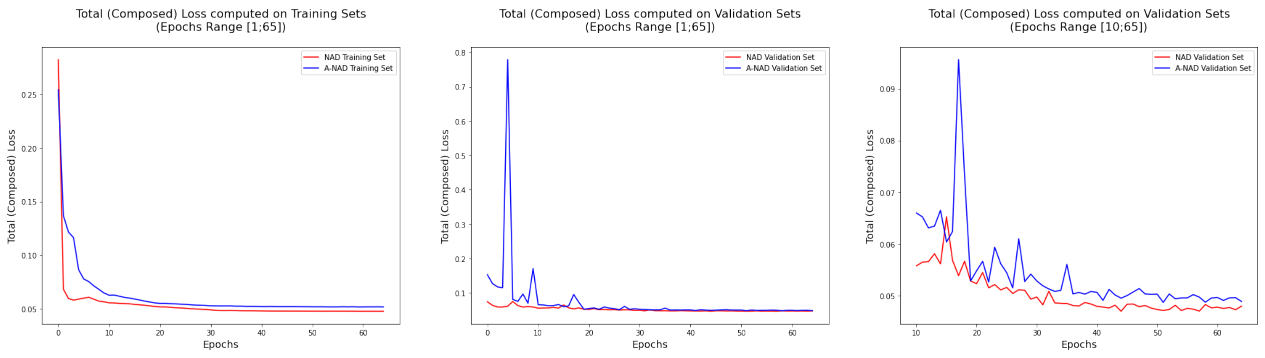


Fig. 4.22. Training SSIM-VQ-VAE on NAD and A-NAD: Loss Trends

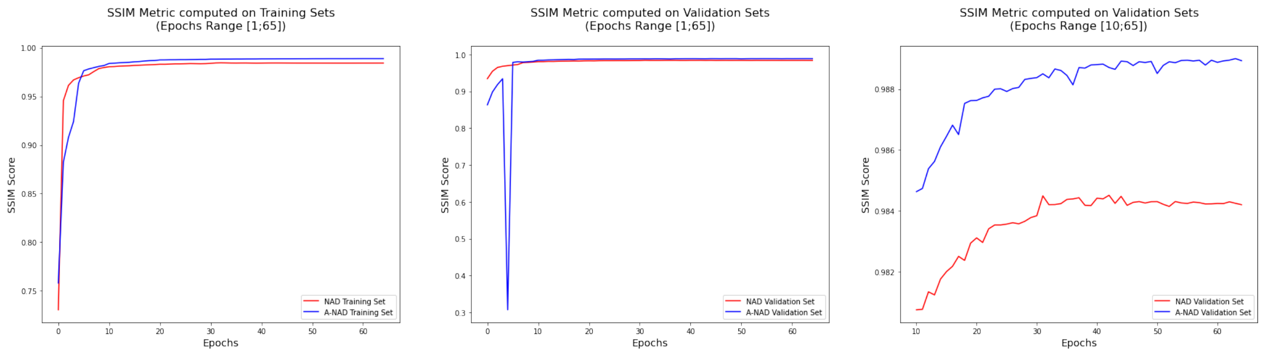


Fig. 4.23. Training SSIM-VQ-VAE on NAD and A-NAD: SSIM Scores

Besides some oscillations in the first phases of the training, the performance of the network does not differ that much. It is also important to underline that the two datasets are different, therefore, the comparisons of the performances, at this stage, are not that meaningful; hence, we performed further tests as discussed below.

Testing SSIM-VQ-VAE on NAD and A-NAD Test Sets. Table 4.9 reports the performance of our network in relation to the best training epoch (i.e., those related to the lowest loss on the Validation set). For the network trained on A-NAD, we also make some tests to check how the network performs on the different augmentation classes (transformations in Table 4.6) separately; Fig. 4.24 reports the results in terms of loss (the lower the better), and SSIM score (the higher the better). Then, for the sake of completeness, Fig. 4.25 presents some visual results of the reconstruction capabilities of the network.

To conclude, we evaluated the performance of SSIM-VQ-VAE trained on the NAD by considering the A-NAD Test set, which contains augmented data. Fig. 4.26 reports the results of these tests in terms of losses and SSIM scores; as expected, given that the NAD does not contain data different from the original ones, the network trained on this dataset is not able to properly reconstruct most of the augmented frames. Furthermore, we can also visually

Table 4.9: Performances of SSIM-VQ-VAE at the Best Epoch(s)

	Performances on NAD	***Performances on A-NAD***
Best Training Epoch	45	52
Loss on Validation Set	0,047049	0,048762
SSIM Score on Validation Set	0,984478	0,988777
Loss on Test Set	0,046928	0,049173
SSIM Score on Test Set	0,984483	0,988714

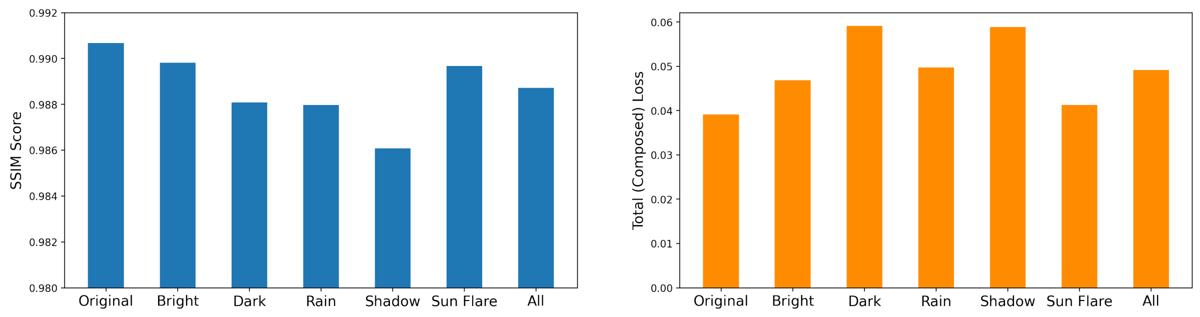


Fig. 4.24. Transformation-by-Transformation Performance of SSIM-VQ-VAE when Trained on A-NAD

note from Fig. 4.25 that the SSIM-VQ-VAE trained on NAD seems not to be efficient with original data too; indeed, all the reconstructions are characterised by that red patina which is absent in the reconstructions made by the network if trained on A-NAD. Given this evidence, we can affirm that *augmentation allows the networks to behave better in all the considered light and weather conditions (including the Original one).* Hence, in the following analyses, we will consider only the SSIM-VQ-VAE trained on A-NAD.

Testing SSIM-VQ-VAE on Frames with Obstacles We evaluated the performances of SSIM-VQ-VAE (trained on A-NAD) on the frames extracted from the WCV (as we did for U-Net in Section 4.4.1.2). Fig. 4.27 reports some examples of anomaly maps generated through the SSIM-VQ-VAE; notably, we applied a simple post-processing step based on the K-Means clustering algorithm[20] to cluster similar pixels and thus try to reduce the "noise" in the anomaly maps (for the K-Means, we used K=7); hence, we computed a heat-map to further highlight the anomalies.

However, it is important to underline that the implemented SSIM-VQ-VAE, besides the fact that the anomaly is not perfectly outlined, suffers from some other problems (e.g., false positives) which are discussed in the following section.

[20]https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html

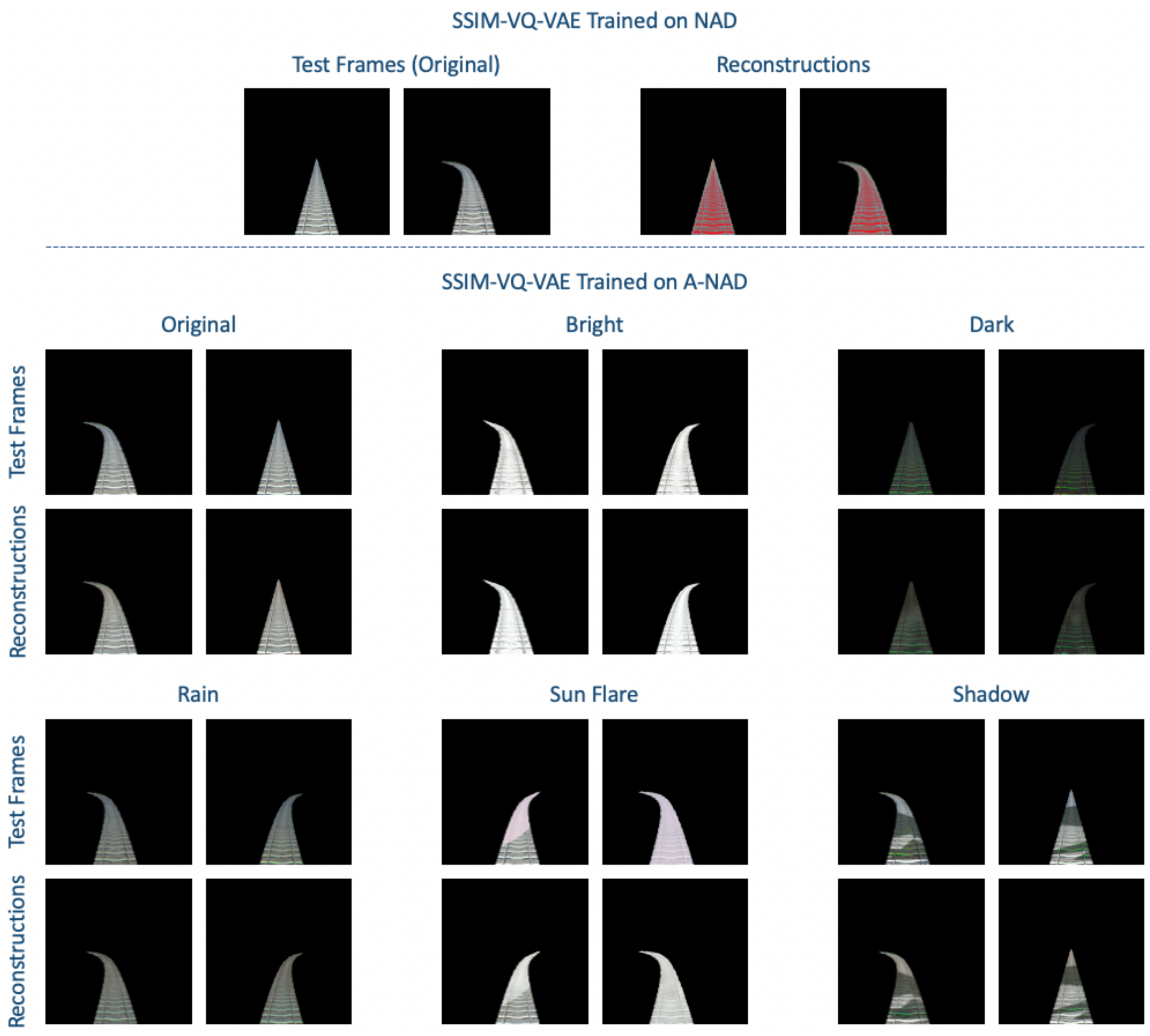


Fig. 4.25. SSIM-VQ-VAE Reconstructions

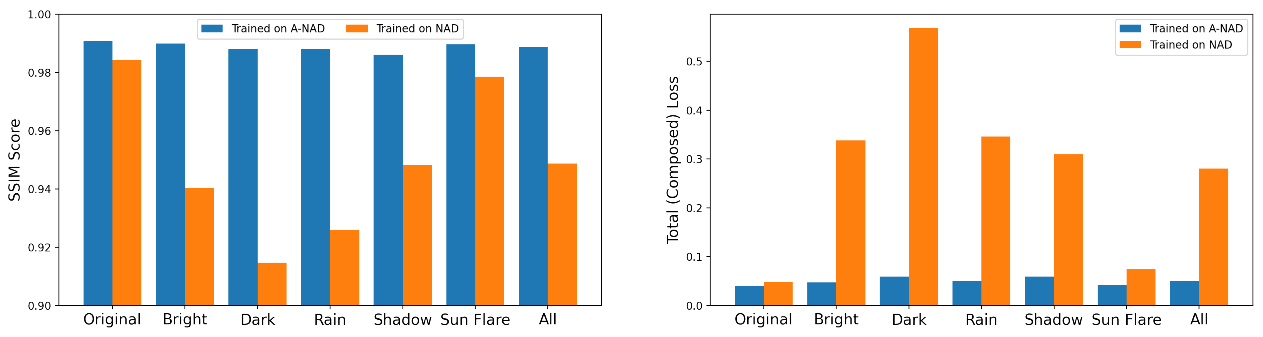


Fig. 4.26. Performances of SSIM-VQ-VAE trained on NAD (orange) and on A-NAD (blue) evaluated on A-NAD's Test Set

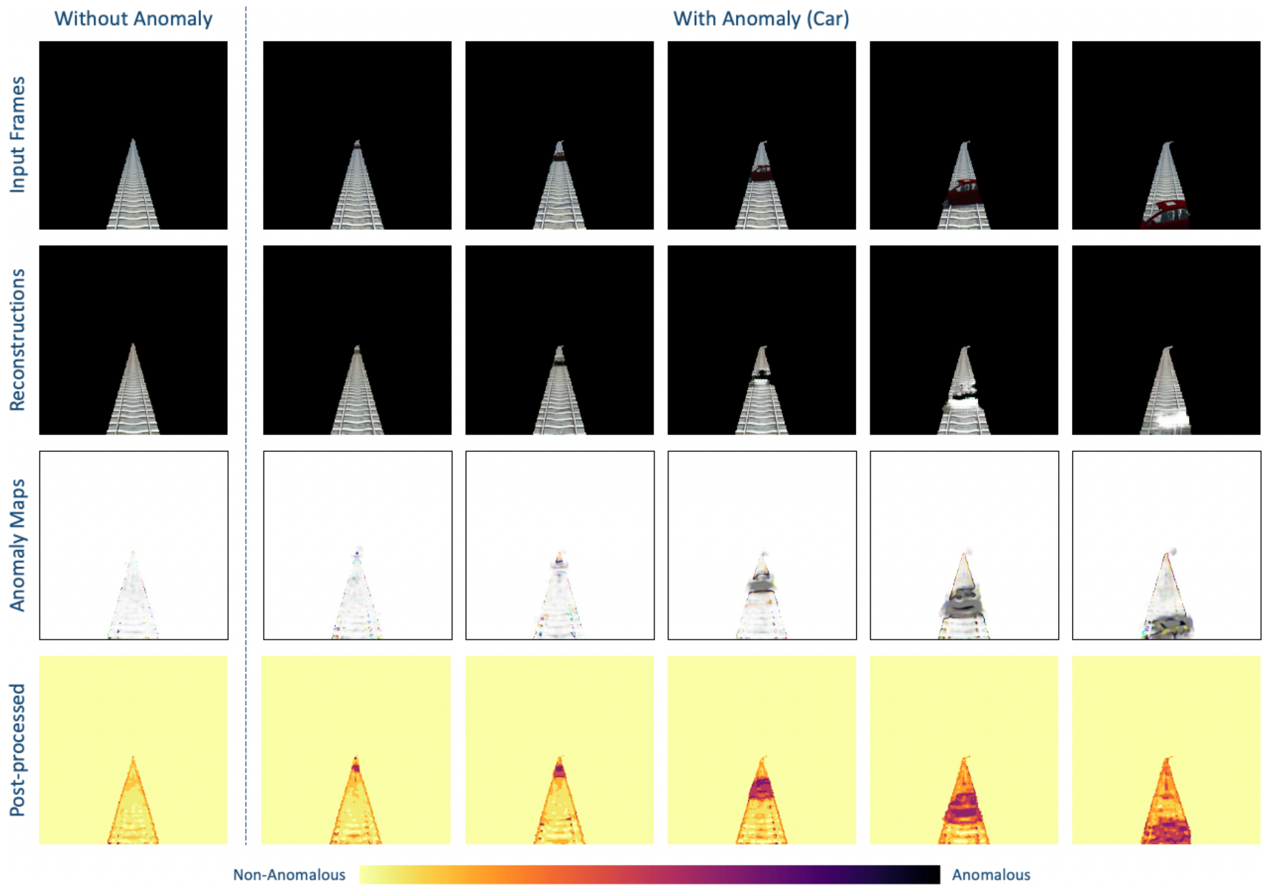


Fig. 4.27. Examples of Anomaly Maps Generated through the SSIM-VQ-VAE

4.5. Evaluation of Results

In the above sections, we discussed the implementation of the Rails Detection Module based on U-Net (Section 4.4.1) and of the Anomaly Detection Module based on SSIM-VQ-VAE (Section 4.4.2). Fig. 4.28 shows a sub-pipeline, extracted from the Obstacle Detection Architecture in Fig. 4.2, which takes into account only the Rails and Anomaly Detection Modules discussed in this chapter together with their post-processing steps.

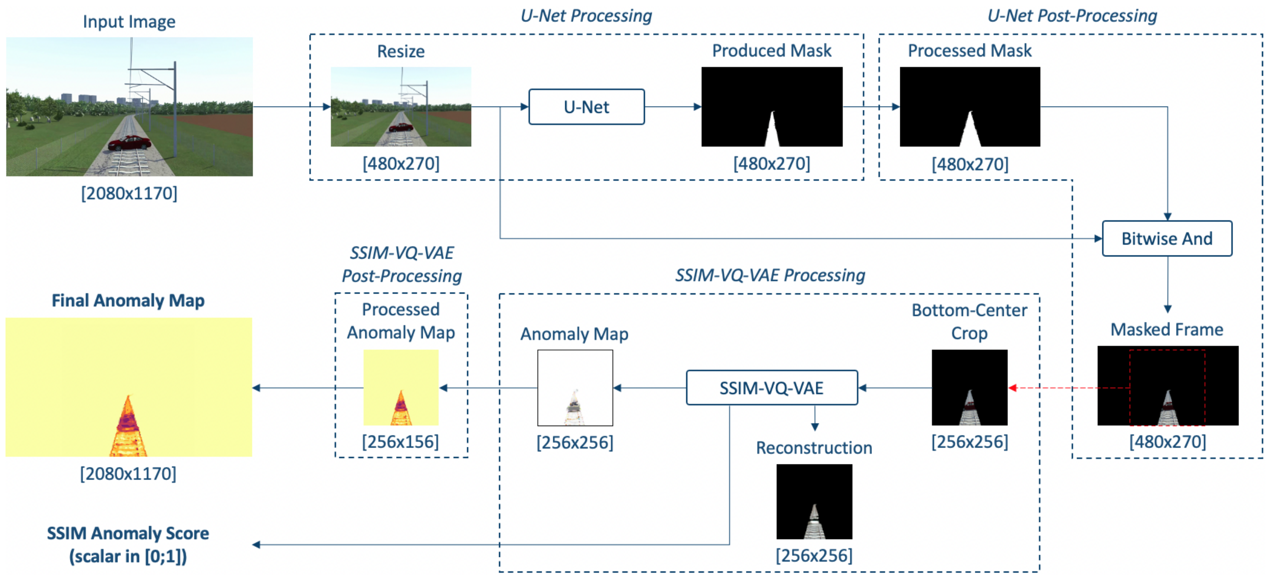


Fig. 4.28. Rails and Anomaly Detection Pipeline

We evaluated the theoretical applicability of such a pipeline from the perspective of the KPIs defined in Table 4.2.

As for the **Computational Time (KPI3)**, by evaluating the processing times on the 1663 frames extracted from the "With Car" Video (WCV), on average, the whole pipeline takes about 476 milliseconds (ms). The various processes contribute as follows:

- U-Net Processing: ∼20.6 ms (∼48.5 frames per second - FPS)
- U-Net Post-Processing: ∼363 ms (∼2.7 FPS)
- SSIM-VQ-VAE Processing: ∼38.9 ms (∼25.7 FPS)
- SSIM-VQ-VAE Post-Processing: ∼53.8 ms (∼18.6 FPS)

Notably, the process that takes more time, and actually increments the computation time by a lot, is the post-processing we implemented to adjust the mask produced by U-Net. Hence, further optimisation would be required (at least) in this direction to make the pipeline suitable for real-time predictions. In addition to that, this approach has other **limits** that should be addressed before considering it viable.

The 1663 frames extracted from the WCV contain the following peculiarities (also depicted in Fig. 4.29):

- In frames from 0 to 37, there is a tree on the left which shadow is projected on part of the rail tracks.
- A human operator would be (probably) capable of seeing the car starting from frame 631 (about 424 meters in advance), even though is quite imperceptible. Also, we could practically say that the train collides with the car in frame 949.

- In almost all the frames, the shadows of the catenaries on the right hand of the rails are projected on the rail tracks.

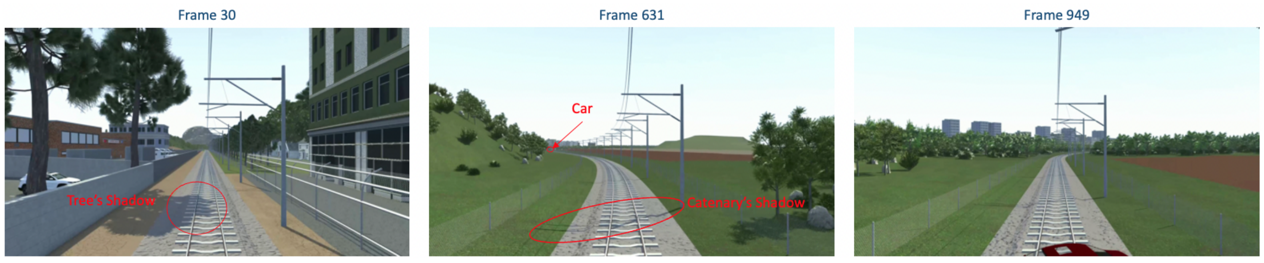


Fig. 4.29. Peculiarities of WCV Frames

While catenaries' shadows do not seem to introduce complications, the proposed pipeline has some limits when it comes to the other two features.

Starting from U-Net (RDM), despite it could be possible to "see" the car starting from frame 631, the mask U-Net produces for that frame (and some subsequent frames) does not include the part of the track occupied by the car affecting the detectability of the obstacles and, consequently, the potential performance of the pipeline according to **KPI1 (Detection Distance)**. Indeed, masked frames (see the bottom right picture in Fig. 4.28) would start "containing" the car approximately from frame 791 (about 180 meters before the collision), hence, for 151 frames (and roughly 240 meters), the car could not be detected. The two main possible causes and directions for improvements are listed herein:

- The labelling process should be performed more meticulously and the self-labelling process should be improved and controlled/tested deeply.
- The down-scaling of the images, from 2080x1170 to 480x270, may introduce some information loss. Practically, in frame 631 (as well as other subsequent frames), the portion occupied by the rails "on the horizon" is quite thin. By down-scaling the images, and obviously the related masks (labels), the segmentation may get lost. A possible solution would be not to down-scale the images too much, but this will increment the GPU memory required by U-Net to be trained and to process the images. Besides that, also the architecture of U-Net may become more complex (more nodes and/or layers), which would increase the memory space needed to "store" the weights of the U-Net itself.

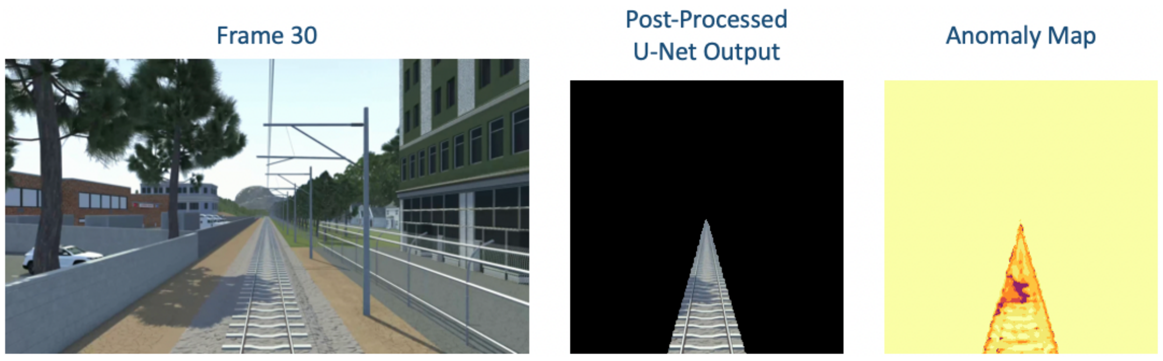


Fig. 4.30. False Positive Generated because of the Shadow of a Tree

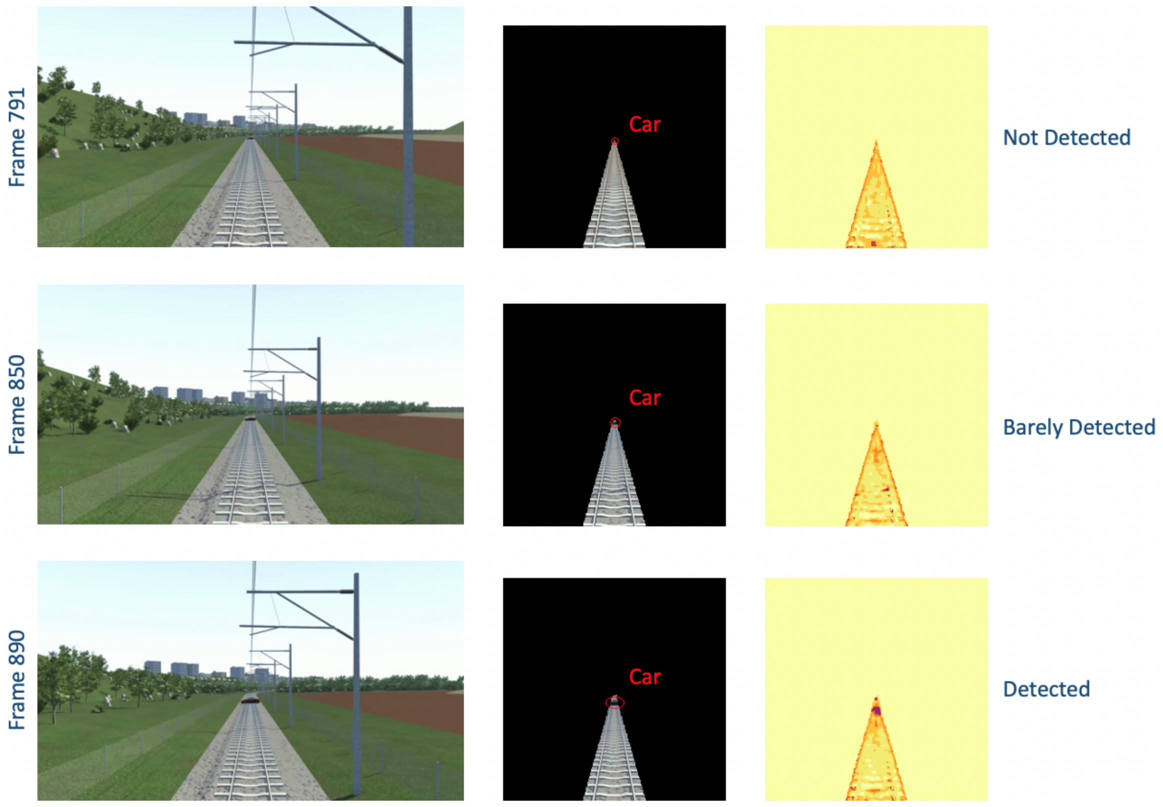


Fig. 4.31. Missed Detection When the Anomaly is too Distant from the Train

As for the SSIM-VQ-VAE (ADM), limitations are more related to the reconstruction capability of the network and resulting false positives/negatives. Fig. 4.30 shows the false positive related to the detection of the shadow of the tree in frame 30. In addition, Fig: 4.31 shows the anomaly maps for frames 791 (about 180 meters from the obstacle), frame 850 (about 115 meters), and frame 890 (about 67 meters). Considering frame 791, the car is in the masked frame but is not highlighted in the anomaly map. This may be due to the fact that the network properly reconstructs also the anomaly, hence, when comparing the reconstruction with the original masked frame, the anomaly is not highlighted; it is like the network is somehow over-performing. These aspects clearly affect the performance of the network from the perspective of **KP1** and **KP2 (Obstacle Coverage)**. Indeed, if on the one hand, the coverage of the whole VBOBS (KPI2) can be potentially increased by introducing an Anomaly Detection approach based on unsupervised learning, on the other hand, the false positives that such approach could introduce may affect the *real* coverage of the system (i.e., the system detects as obstacles items that should not be detected). As for KPI1, the obstacle under analysis is *properly* detected at about 70 meters, which may affect the implementability of the approach in real scenarios (further considerations are given in the following section).

Some of the possible actions that may be taken to deeply investigate the suitability of the SSIM-VQ-VAE against the issues reported above are:

- Increase the dataset including more samples. This would potentially allow the network to better learn the characteristics related to false anomalies (e.g., shadows, sun flares, etc.) and produce more suitable reconstructions.
- Increase the dimensions of the latent space. In this analysis, we set 512 centroids.

As discussed in Section 4.4.2.2, the features extracted from the input images by the encoder are substituted, at a given point, with the vectors (centroids) contained in the latent space. This substitution represents a sort of architectural limit as the reconstructed image could not be perfectly identical to the input image. Probably, by increasing the dimensions of latent space, we could have better reconstructions that would allow for better identification of the anomalies.

- Slightly modify the dataset. After the resizing and the cropping, the 256x256 images we used to train and test the network are characterised, for a very large percentage, by a black background. Tests may be performed by varying the dimensions of the images (i.e., larger than 256x256) and/or by cropping them differently (to let the rails occupy more space in the image).

  Alternatively, it would be possible to process directly the original frames (without preprocessing them through U-Net); nevertheless, at that point: i) the architecture proposed in 4.2 should be changed as the RDM and the ADM would work in parallel; and ii) it would be a quite different task which could introduce other complications since the images that should be reconstructed would be more complex (in terms of number and variety of the items within the frames that should be properly reconstructed).

To conclude, for the sake of knowledge, the SSIM-VQ-VAE, besides the Reconstruction and the anomaly map, produces a SSIM (Anomaly) Score which is a numerical representation (scalar) of the anomaly map. Practically, the higher the score the more different the two images, hence, the higher the probability that there is an anomaly. Fig. 4.32 plots the SSIM Scores for each of the 1663 frames of the WCV. The blue line represents the actual scores, i.e., those produced by the SSIM-VQ-VAE; the dashed red line represents how these scores should ideally look like, if there is no anomaly the SSIM Score should be 1 (the two images are identical), otherwise, it should be lower depending on the portion of the image occupied by the anomaly; lastly, the dashed green line represents the ideal trend but takes into account some reconstruction or processing (theoretically approximated) errors that could not be completely eliminated from DNNs in general (an example, for this DNN, is the error introduced by the latent space as discussed above). Notably, there is a drop around frame 37 (false positives related to the tree's shadow); also, the anomalies should be detected, hence, the trend should start dropping, at least from frame 791, however, the chart shows a sudden drop just a few frames before the potential collision.

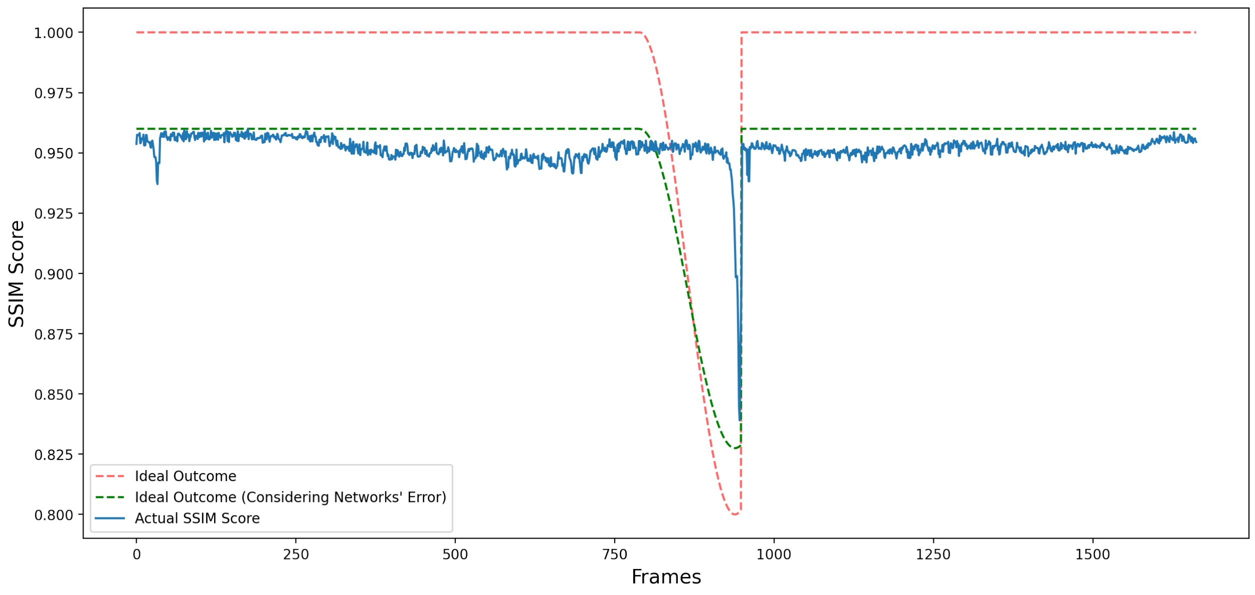


Fig. 4.32. SSIM Score Generated by the SSIM-VQ-VAE

4.6. Discussion of Results

As mentioned in Section 4.1, obstacle detection systems typically involve different sensors (including LiDARs, radars, and cameras); in this context, cameras are used to identify the type of the obstacle while their detection is primarily demanded to the other sensors. Also, when it comes to VBODSs specifically, most of them rely on supervised DL approaches which fail in detecting anomalies that have not been considered during the training phase. With this proof-of-concept we aimed at understanding whether it would have been possible to detect any kind of obstacle (both known and unknown a-priori) while leveraging a cost-effective sensor (i.e., a camera). Specifically, we investigated the possibility of using Unsupervised DL models to detect obstacles that are not known a-priori while leveraging data collected through a simulated environment we developed by exploiting RoadRunner (see Section 4.3).

Given the large attention the (Vision-Based) Object Detection topic has received during the last years within the literature, in this document, we mainly focused on the implementation of the Anomaly Detection Module to study the effectiveness of Unsupervised approaches in detecting anomalies and answer to the RQs reported in Table 4.1. In doing so, we also implemented the Rails Detection Module to practically process and mask video frames to produce data to train and test the SSIM-VQ-VAE (the core component of the ADM).

4.6.1. Evaluation of KPIs and Limits of the Approach

In Section 4.5, we have deeply analysed the performance of the implemented approaches with respect to KPIs reported in Table 4.2. In this section, we summarise the obtained results and highlight their relevance in connection with the practical applicability of the proposed pipeline (see Fig. 4.28), at the current stage of development, in real scenarios. Table 4.10 summarises the results we obtained in relation to the considered KPIs.

Our tests show that the anomaly detection approach can effectively support the identification of obstacles not specified in advance (KPI2). As for the usage of just one RGB camera on

Table 4.10: Summary of Results against KPIs.

KPI	Results
KPI1: Detection Distance	Basing on our tests, anomalies can be detected within a range of about 70 meters.
KPI2: Obstacle Coverage	Object Detectors are limited to a specific number (defined a-priori) of object classes. In our view, this number can be increased by adopting Unsupervised Anomaly Detection approaches since they do not require to be trained with pre-defined sets of obstacles, rather, they focus on highlighting any kind of difference from the nominal condition (i.e., free tracks).
KPI3: Computation Time	At the current stage of development, the implemented pipeline elaborates a single video frame in about 476 ms.

board the train, the pipeline we implemented allows detecting the obstacles within a range of about 70 meters (KPI1). Of course, this specific result strongly limits the applicability of just one camera when the stopping distance, on average, is greater than 70 meters; however, we identified a list of actions that can be taken to potentially improve the current performances and increase the detection distance (see Section 4.5). In addition to that, as for the computation time (KPI3), the pipeline takes about 476 ms to process a single video frame. Worth noting, the computation time should be considered in relation with the FPS of the camera; for example, a 30 FPS camera records a frame each 33.3 ms, therefore, if the system processes a frame within that time, it could be considered viable for real-time. Clearly, to higher FPS correspond lower computation time constraints (e.g., 60 FPS camera $\rightarrow$ max 16.6 ms per frame). However, the pipeline we implemented largely exceeds these time constraints; to be specific, the processes that contribute most to the computation time are the two post-processing algorithms, rather than the two Deep Learning approaches (see Section 4.5). Therefore, further optimizations are required before considering the system viable for some real-time applications, especially in high-speed railways.

However, it is important to mention that there are some scenarios in which the experimented VBODS approach, once properly tested and further optimised, can potentially bring benefits and help to improve safety or automation in a cost-effective manner. For example, the one-camera solution could be useful to assist the driver, or to implement autonomous behaviours in secondary railway lines, e.g., regional or urban (intra-city) lines, where trains run at reduced speed. Actually, it could also be used in main lines as low-speed complement and/or to assist the driver during procedures such as the “Track Ahead Free”, where the driver is asked for confirmation that no obstacles are on the same track circuit the train is occupying during the ERTMS/ETCS “Start of Mission” scenario [1, 30], or to integrate a “return to home” functionality; for example, in case of malfunction of driving systems that would not allow trains to safely operate at their maximum efficiency, a one-camera system could support the train to autonomously and safely reach the nearest station by proceeding at reduced speed.

4.6.2. Main Findings and Future Improvements

Beyond autonomous train driving applications, as also mentioned in Section 4, our analyses were also oriented at understanding whether the approaches we investigated could be useful to detect obstacles in other contexts and, based on the obtained results, it is not so visionary to assume that such DNNs represent suitable starting points for more static applications, e.g., obstacle detection within the Level Crossings area.

In summary, this proof-of-concept allowed us to investigate possible DL approaches and techniques that could be exploited to deal with obstacle detection starting from what has already been proposed within the rail and other sectors (**RQ3**). The main key findings that can be deducted from our analyses are reported herein:

- Editors like RoadRunner could help to simulate driving scenarios and safely collect a potentially infinite amount of data to initially train and validate AI models. In combination, Data Augmentation techniques allow simulating weather and light condition to have a more comprehensive dataset encompassing as much environment aleatory as possible. It is also important to mention that scenario developed in RoadRunner can be integrated within other 3D editors, like Unity or Unreal Engine, which would allow for further customisation (e.g., directly implement different light and weather conditions within the editor itself) of the environment making it more similar to real railway scenarios.
- Transfer Learning could help to obtain more stable results and reduce the training time of DNNs (see Section 4.3.2.4).
- Self-Training (that we combined with Transfer Learning, Section 4.3.2.4) seems to be a promising direction towards which investigating further when it comes to semi-automatic labelling. Possibly, in order to improve the performances, the starting dataset could be larger (e.g., 20-30% of the dataset instead of 10% as in our case) and the down-scaling of the images can be reduced (e.g., from 2080x1170 to 950x540 instead 480x270) not to lose too much information. In the second case, it is important to note that to larger images could correspond larger GPU memory requirements.
- Autoencoders oriented at reconstructing the image in input can be adopted, if properly adapted, to detect anomalies within images (**RQ1**). In this case, the suggestion would be to apply a pre-processing step to eliminate as much as possible the disturbing elements that would not be useful for the task in order to allow the network to “focus” only on the relevant aspects. For example, we masked the frames instead of using them as they were collected; however, it is important to underline that, in our case, given that the images were characterised by a black background, traditional Autoencoders did not work properly according to the tests we performed.

In addition to that, there is another important aspect to discuss. Our tests were performed by considering one of the simplest possible scenarios, i.e., a railway line with a single track. However, we think that the approach can be scaled also to railway lines with multiple tracks. In Section 4.3.2.1, we have discussed how RailSem19 was customised; however, there is another thing that should be underlined. Basically, the scripts we adapted to extract rail labels from RailSem19 propose two approaches: i) extract all the rail tracks contained in each frame; or ii) extract only the rail track that the train is travelling (ego rail). Although, in this proof-of-concept, we analysed the simplest case, we actually adopted the first approach so that the pre-training on the customised RailSem19 (Section 4.3.2.3) could be also exploited in the future to deal with more complex scenarios involving multiple rail tracks. Therefore, U-Net performances reported in Section 4.3.2.3 (Table 4.3 to be specific) are related to the capability of the network of detecting multiple rail tracks within the same video frame. The results we obtained suggest that is not so visionary to assume that, in case the railway line to be monitored would involve multiple rail tracks, the Rails Detection Module would be still capable of performing properly. Then, to perform the object and anomaly detection on the rail track the train is traveling (ego-perspective track), it would we be possible to extract the ego-perspective track by filtering out the other tracks through an algorithm that would be similar to that discussed in Section 4.4.1.3 (Phase 2); hence, it would be possible to proceed with the ODM and ADM as presented in the above sections. Alternatively, it would be also possible to fine-tune U-Net, by feeding it with labels highlighting only the ego-perspective track and leaving the other tracks as part of the background, so that it would be capable of directly extracting the ego-perspective track. Basing on other tests we performed, this approach seems to be viable and seems not to introduce additional complications (especially if combined with the filtering algorithm), however, further investigations would be required to validate its effectiveness.

To conclude, the results obtained within this proof-of-concept indicate that the approaches we have investigated have good potential in the context of autonomous driving and beyond; however, as also mentioned above, there is still room for some improvements and further actions and tests can be taken (and are recommended) to assess their viability for some specific applications in real-world scenarios. With the aim of facilitating future research, we report in Table 4.11 the category of possible approaches that, in our view, could represent suitable starting points and/or alternatives to improve or support the modules analysed in this chapter and implement those we did not address within this manuscript.

Table 4.11: AI Approaches for Obstacle Detection

Module	Implemented Approach	Suitable Alternatives
Rails Detection	SS Approach based on U-Net [11] (Section 4.4.1)	Other SS Approaches (e.g., [8, 9]).
Object Detection	/	Potentially any Object Detector [4, 12, 13] (e.g., YOLO [14]).
Anomaly Detection	Unsupervised Approach based on VQ-VAE [28] (Section 4.4.2)	DL Approaches for Image Reconstruction (e.g., based on AEs [20, 21]) or Image Synthesis (e.g., based on GANs [18] or ad-hoc DNNs [19]). Alternatively, DL combined with Stereo Vision (e.g., [31, 32]).
Distance Estimation	/	DDNs in case of Objects Distance Estimation (e.g., [22]), or Triangulation in case of Anomalies Distance Estimation and Stereo Vision (e.g, [23]).

The table summarises the literature analysis we have discussed in Section 4.2, where we also reported the reasoning we applied to choice the specific DL approaches we investigated. For the sake of simplicity, we briefly recall what we have discussed in that section in relation to the approaches we implemented and some suitable alternatives:

- For the Rails Detection Module, approaches based on DL could be more suitable than traditional Computer Vision approaches as they can better adapt to changes in the environment (e.g., different light and weather conditions) [4]. Particularly, SS approaches (mainly based on AEs) represent suitable solutions to extract the rail tracks from input frames as demonstrated in this document but also in other previous work carried out within the rail sector [8, 9]. Alternatively, lane detection approaches (as those reviewed in [33]) can be borrowed from automotive and potentially adapted for rail tracks detection (**RQ2**); also in this case, even though they have also been implemented classification and object detection approaches, SS ones seem to be the most adopted.

- Object Detection is one of the most addressed DL-based Computer Vision tasks within the literature. Potentially any kind of the so-called object detectors can be adopted (and adapted) for this task, as those reviewed in [12,13] (**RQ2**). Notably, some of them, e.g., detectors belonging to the Region-based CNNs (R-CNN) and YOLO families, have already been tested for object detection on rail tracks [4,14].
- As for the Anomaly Detection Module, starting from works that have been proposed to cope with anomaly detection within images depicting, from a high-level perspective, manufacturing/industrial items [20,21] (**RQ2**), we implemented an approach based on VQ-VAEs (a particular kind of AEs) to directly reconstruct the image in input and then produce the anomaly map by comparing the original and the processed images. However, image synthesis approaches, e.g., based on Generative Adversarial Networks (GANs) [18] or DNNs built ad-hoc [19], seem to be suitable alternatives to reconstruct the images in input. In addition, in this document, we focused on a single camera for the motivations expressed in Section 3, but it is also important to mention that the combination of DL and stereo vision (i.e., multiple cameras) [31,32] could represent a suitable alternative to implement vision-based anomaly detection.
- Lastly, for the Distance Estimation Module, we think that the DNN developed within the SMART project (DisNet [22]) could be a suitable starting point to detect the distance of the objects from the camera. On the other hand, in case of anomalies, stereo triangulation [23] could be a solution in case multiple cameras are adopted.

5. Cooperative Driving for Virtual Coupling of Autonomous Trains

5.1. Introduction

In RAILS Deliverable D2.2 [5], the cooperative driving for Virtual Coupling (VC) of autonomous trains case study has been addressed to investigate the potential adoption of AI techniques for enhanced rail safety and automation. Specifically, after a deep analysis of the state of the art, and considering the VC results in the automotive field, a methodological proof-of-concept has been provided, in which the following research questions have arisen:

- **RQ1** Can we transfer to railway VC some of the methods already exploited for vehicle platooning?
- **RQ2** Can AI approaches be leveraged to ensure the effectiveness of the tactical layer functionalities both in nominal and uncertain environments?
- **RQ3** Can we address possible answers to RQ1 and RQ2 through a proof of concept in order to inspire future developments and a technology roadmap?

In an attempt to give an answer to the above questions, AI approaches have been evaluated to explore the technical feasibility of railway VC. Among them, a Deep Reinforcement Learning (DRL) approach based on the Deep Deterministic Policy Gradient (DDPG) strategy has been identified as one of the potential solutions to implement railway VC from a tactical layer perspective. Expected results and possible criticalities related to the exploitation of this approach have been stressed.

In view of this methodological analysis, in what follows we propose the consequent experimental part of the aforementioned proof-of-concept. The latter accounts for the definition of an effective and robust control strategy for railway VC, able to simultaneously cope with uncertain nonlinear heterogeneous train convoys, uncertain nonlinearities and unexpected/unpredictable external factors. Specifically, we design a novel DDPG controller to coordinate and manage the Virtually Coupled Train Set (VCTS) such that this latter can autonomously adapt its behaviour to the encountered driving scenarios. The effectiveness of the proposed approach is evaluated via a numerical analysis, carried out via an *ad-hoc* implemented simulation platform. Namely, after verifying the efficiency of the training process in ensuring the fulfillment of the VC control objectives, extensive non-trivial simulations involving cooperative manoeuvres in concrete operational scenarios are performed for the validation phase. Results confirm how the proposed model-free DDPG approach guarantees the VC for nonlinear heterogeneous train convoys, despite the co-presence of uncertainties and unknown external factors.

Finally, the advantages and the benefits of the proposed DRL controller are disclosed via a comparison analysis against an optimal model-based strategy. The comparison is carried out exploiting some of the Key Performances Indicators (KPIs) provided in the previous Deliverable (see Table 5.1) as evaluation criteria to assess the feasibility of VC in railways. The comparison analysis shows that the proposed control algorithm can provide meaningful advantages in terms of lane capacity, robustness to parameters uncertainties, flexibility to the encountered driving scenarios, and energy saving.

Table 5.1: KPIs to evaluate the effectiveness of VCTS paradigm [5]

KPI	Description
KPI1	**Time gap**, i.e., the time interval between two consecutive consists
KPI2	**Time to collision**, which is the remaining time before a rear-end accident, assuming unchanged speeds of both consists
KPI3	**Trip time**, that is, the time needed to accomplish a given mission
KPI4	**Tracking error**, i.e., the error between the desired reference behaviour and the actual one
KPI5	**Energy consumption**, defined as in [34], to evaluate the energy efficiency

5.2. Model Description

A VCTS is a convoy of N heterogeneous autonomous consists able to share their state information, i.e., the absolute position and speed, with the other communicating consists via a Train to Train (T2T) wireless communication network. The reference behaviour for the VCTS

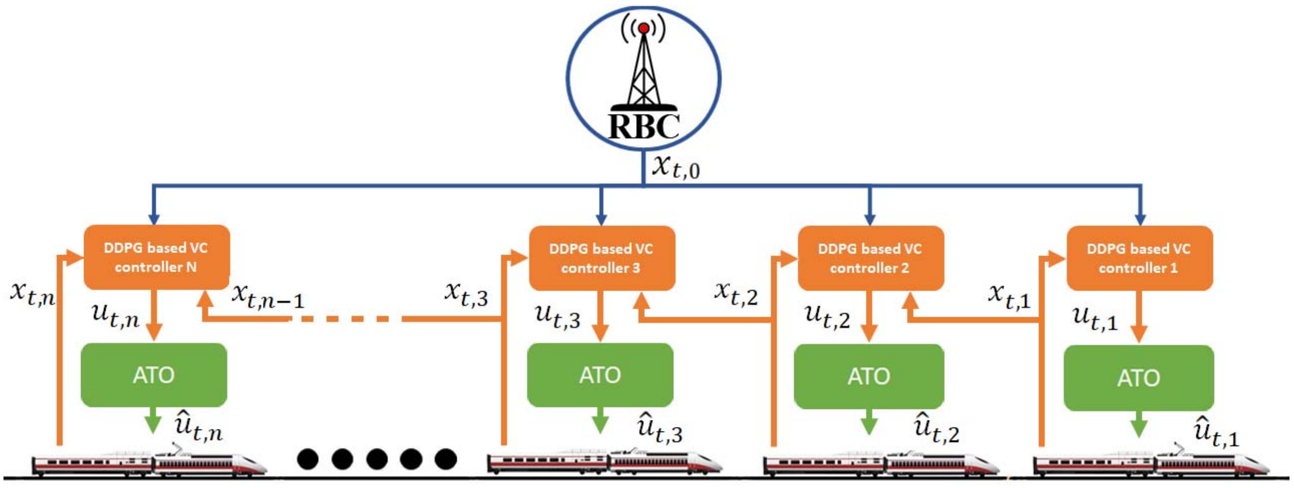


Fig. 5.1. VCTS of N autonomous consists receiving information from RBC through the T2I communication network (blue lines), and sharing information among each other through the T2T communication network (orange lines)

is imposed through a Train to Infrastructure (T2I) communication network by the Radio Block Center (RBC), which acts as a virtual leader and is indexed with 0 (see Fig. 5.1 for a conceptual view of a VCTS). The communication architecture proposed in Fig. 5.1 reflects the well assessed Leader-Predecessor-Follower (LPF) communication topology deeply exploited in the automotive field (see for instance [35]). We highlight that future research will be devoted to the investigation of the latency sources in the communication topology.
The aim is to design a decentralised DDPG control algorithm able to fulfill the VCTS tactical layer functionalities described in [36]. Namely, the tactical layer coordinates the actual platoon movements and manoeuvres from the instance a joining request is received until the platoon is dissolved, according to the composition, ordering and coupling/decoupling instruction imposed by the strategic layer. The latter is herein emulated through a basic logic according to the VC operational states and transitions proposed in [37, 38].

In view of this, the control algorithm has to ensure that: $a)$ each consist i $(i = 1, \cdots, N)$ travels at the desired speed imposed by the virtual leader; $b)$ it maintains a secure inter-train distance (w.r.t. the preceding vehicle $i-1$), which explicitly takes into account the relative braking curve. To this aim, the proposed DRL solution computes the desired driving action to be imposed to the operational layer, that is, the Automatic Train Operation (ATO) system embedded in each consist i, as shown in Fig. 5.1.
We emphasize that the effectiveness of the VCTS paradigm is especially crucial for High-Speed Trains (HSTs), where, due to the high-speed operating ranges, uncertain factors have a stronger impact on the VC performance.

5.2.1. Consists Dynamics

To emulate the consists dynamics, the behaviour of each train, identified with its related DDPG agent i $(\forall i = 1, \cdots, N)$, is described by its nonlinear longitudinal dynamics, which accounts for the driving/braking propulsion system, the aerodynamic drag and the effects of unknown external disturbances, as in [39]:

$$\dot{p}_i(t) = \ v_i(t) \tag{5.1}$$

$$\dot{v}_i(t) = \frac{1}{M_i(t)}\big[u_i(t) - R_i(v_i(t))\big] + \omega_i(t) \tag{5.2}$$

where $p_i(t)\ \ [m]$, $v_i(t)\ \ [m/s]$ are the position and speed of the i-th train, respectively; $R_i(v_i(t)) = C_{1,i}(t) + C_{2,i}(t)\,|v_i(t)| + C_{3,i}(t)\ v_i(t)^2$ is the propulsion resistance expressed via the Davis formula [40], with $C_{1,i}(t)\ [kg\ m/s^2]$, $C_{2,i}(t)[kg/s]$ and $C_{3,i}(t)\ [kg/m]$ representing the Davis parameters of the aerodynamic drag; $M_i(t)\ [kg]$ is the i-th train mass; $\omega_i(t)\ [m/s^2]$ represents the unknown nonlinearities due to the friction force, external disturbances, such as wind force, the curvature and slope forces imposed by the railway track [41]; $u_i(t)\ [N]$ is the driving or the braking force, evaluated by the DDPG agent by exploiting local measurements, i.e., $x_i(t) = [p_i(t)\ v_i(t)]^\top$, and the networked information shared by the communicating consist $i-1$ and the leader, i.e., $x_{i-1}(t) = [p_{i-1}(t)\ v_{i-1}(t)]^\top$ and $x_0(t) = [p_0(t)\ v_0(t)]^\top$.
Note that the value of the i-th train parameters (i.e., $M_i(t)$, $C_{1,i}(t)$, $C_{2,i}(t)$, $C_{3,i}(t)$) strongly depends on the different driving conditions, the changing of the operating speed and operational scenarios. Here, to take into account this variability, we assume all trains parameters to be uncertain and time-varying. In detail, we model the parameters as:

$$M_i(t) = \bar{M}_i + \delta M_i(t)\ C_{1,i}(t) = \bar{C}_{1,i} + \delta C_{1,i}(t)\ C_{2,i}(t) = \bar{C}_{2,i} + \delta C_{2,i}(t)\ C_{3,i}(t) = \bar{C}_{3,i} + \delta C_{3,i}(t) \tag{5.3}$$

where $\bar{(\cdot)}$ indicates the nominal values of the corresponding parameters assumed to be known (for instance, $\bar{M}_i$ can include the rotating mass factor), while $\delta\bar{(\cdot)}(t)$ represents their related uncertainties.
Conversely, the reference behaviour imposed by the virtual leader is described by the following non-autonomous system:

$$\dot{p}_0(t) = v_0(t) \tag{5.4}$$

$$\dot{v}_0(t) = u_0(t) \tag{5.5}$$

being $p_0(t)\ [m]$, $v_0(t)\ [m/s]$ and $u_0(t)\ [m/s^2]$ the absolute position, speed and acceleration of the virtual leader, respectively.

5.2.2. Safety Relative Braking Distance

The VCTS paradigm *breaks the walls* from absolute to relative braking distance (RBD) among the consists, in order to increase line capacity. Thus, an accurate definition of the safe inter-train distance $d_{s,i}(t)$ is provided for each train i ($\forall i = 1, \cdots, N$). According to [42], it takes into account the most favourable braking condition for the preceding consist $i-1$ and the guaranteed emergency brake rate for the consist i, i.e., the most unfavourable braking one. In addition, $d_{s,i}(t)$ explicitly considers the braking reaction delay, the time required for its computation, the maximum and minimum acceleration rate, as well as uncertainties in train location information.
Therefore, the safe inter-train distance is given by:

$$d_{s,i}(t) = d_{fmax,i}(t) - d_{lmin,i-1}(t) + s_m + 2\, e_{s,i}(t) \tag{5.6}$$

where, $d_{fmax,i}$ is the most unfavourable braking distance of consist i; $d_{lmin,i-1}$ is the most favourable braking distance of consist $i-1$; s_m is the safety margin imposed between adjacent consists; $e_s(t)$ is the trains localization error. More specifically, $d_{fmax,i}$ and $d_{lmin,i-1}$ are computed as follow:

$$\begin{aligned} d_{fmax,i}(t) = & \Big(v_i(t) + v_{es,i}(t)\Big)\,\Big(t_{b,i} + t_{c,i} + t_{re,i}\Big) + \\ & + \frac{1}{2}\, a_{d,i}\, t_{re,i}^2 + \Big(v_i(t) + v_{es,i}(t) + a_{d,i}\, t_{re,i}\Big)\, t_{e,i} + \frac{\Big(v_i(t) + v_{es,i}(t) + a_{d,i}\, t_{re,i}\Big)^2}{2\, a_{bs,i}} \\ & + \left(\frac{1}{2\, a_{b,i}} - \frac{1}{2\, a_{bs,i}}\right)\Big(v_i(t) + v_{es,i}(t) + a_{d,i}\, t_{d,i} - a_{bs,i}\, t_{bu,i}\Big)^2, \end{aligned} \tag{5.7}$$

$$d_{lim,i-1}(t) = \frac{\Big(v_{i-1}(t) - v_{es,i-1}(t)\Big)^2}{2\, a_{bs,i-1}} + \left(\frac{1}{2\, a_{b,i-1}} - \frac{1}{2\, a_{bs,i-1}}\right)\Big(v_{i-1}(t) - v_{es,i-1}(t) - a_{bs,i-1}\, t_{bu,i-1}\Big)^2, \tag{5.8}$$

where $v_{es,i}(t)$ and $v_{es,i-1}(t)$ are the speed measurement error for consist i and $i-1$, respectively; $v_i(t)$ and $v_{i-1}(t)$ are the speed of consist i and $i-1$, respectively; $t_{i,b}$ is the sensing delay for consist i; $t_{i,c}$ is the response time to the braking command for consist i; $t_{i,re}$ is the i-th train traction cut-off time; $t_{i,e}$ is the emergency brake establishment time for consist i; $t_{i,bu}$ and $t_{i-1,bu}$ are the emergency brake establishment extra time for consist i and $i-1$, respectively; $a_{i,d}$ is the maximum traction acceleration for consist i; $a_{i,b}$ and $a_{i-1,b}$ are the maximum braking acceleration for consist i and its preceding, respectively; $a_{i,bs}$ and $a_{i-1,bs}$ are the normal braking acceleration of consist i and its preceding, respectively. Hence, via the characterization of the train brake performance in terms of the equipment response time, cut-off traction time, brake establishment time, and all the factors influencing reaction delays, the safe inter-train distance can be expressed as a function of the preceding train speed.

Remark 1. *Without loss of generality, the speed measurement error $v_{i,es}(t)$ is modelled as a random variable uniformly distributed within the range $[0; 1]$ $[km/h]$. Moreover, the train localization error $e_s(t)$ is again modelled as a random variable, but uniformly distributed within the range $[-0.5; 0.5]$ $[m]$ [41, 43].*

5.2.3. Problem Statement

Consider a VCTS of N heterogeneous autonomous HSTs, sharing state information via T2T, plus a virtual leader imposing the reference behaviour for the whole platoon. Find a robust DDPG control strategy $u_i(t)$ driving the motion of each consist i ($\forall i = 1, \cdots, N$) in tracking the reference behaviour $x_0(t)$ while maintaining the safe inter-train distance $d_{s,i}(t)$ w.r.t. its predecessor $i-1$, i.e.:

$$\lim_{t\to+\infty} \|v_0(t) - v_i(t)\| = 0, \tag{5.9}$$

$$\lim_{t\to+\infty} \|p_{i-1}(t) - p_i(t) - d_{s,i}(t)\| = 0, \tag{5.10}$$

despite the presence of:

- uncertainties in train dynamic parameters, i.e., $M_i(t)$, $C_{1,i}(t)$, $C_{2,i}(t)$, $C_{3,i}(t)$;
- heterogeneity in trains dynamics, as they can be characterized by different operational performances (e.g., different braking capabilities, different speed categories);
- uncertainties in track conditions (e.g., adhesion factors, gradients profile), external disturbances (e.g., wind speed) and unknown exogenous forces due to the curvature and the slope, i.e., $\omega_i(t)$;
- uncertainties in reaction delay when performing braking manoeuvres, which influence the inter-train distance $d_{s,i}(t)$;
- uncertainties in train location information, indicated with $e_s(t)$;
- on-board speed error measurements for each train i, indicated with $v_{i,es}(t)$.

5.2.4. DDPG Virtual Coupling Controller

The DRL theory is exploited for the development of a control strategy able to solve the VC problem stated in Section 5.2.3. In the DRL framework, an agent interacts with an unknown environment which is commonly modelled as a Markov Decision Process (MDP), i.e., an efficient mathematical tool for uncertainty modelling in the surrounding current state, future evolution and interactive behaviour with the agent. The maximization of an expected cumulative reward function represents the interaction process between the agent and the environment, with the aim of optimizing the agent behaviour via a *learning-by-doing* process. Namely, the environment reactions are iteratively evaluated for determining the next optimal action for the agent to perform. Specifically, on the basis of the qualitative analysis conducted in RAILS Deliverable 2.2 [5], an actor-critic method, known as DDPG, originally proposed in [44], is exploited. It is a model-free, off-policy and actor-critic approach that extends the fast learning convergence properties of the Deep Quality Network (DQN) to the continuous space while taking advantage from the Deep Policy Gradient (DPG) method [45].
In the VC context, DDPG agent $i, \ \forall i = 1, 2, \cdots, N$, computes the control action $u_i(t)$ to be imposed according to the environment reactions. Each agent is composed of 4 fully-connected Deep Neural Networks (DNN), that is, actor, actor target, critic and critic target, as shown in Fig. 5.2. Here, the actors compute the action a_t to be performed by each consist i, while the critics evaluate, at each time state, the quality of the chosen action, according to the *learning-by-doing* process.
For each agent i, the observation vector $s_t \in \mathbb{R}^6$ is defined as the set of the i-th measured on-board train state information, i.e., $x_i(t) = [p_i(t), \ v_i(t)]^T \in \mathbb{R}^2$, and the networked information coming from the active T2T communication links, i.e., $x_{i-1}(t) = [p_{i-1}(t), \ v_{i-1}(t)]^T \in \mathbb{R}^2$ and $x_0(t) = [p_0(t), \ v_0(t)]^T \in \mathbb{R}^2$. The action space $a_t \in \mathbb{R}$ represents the set of the possible

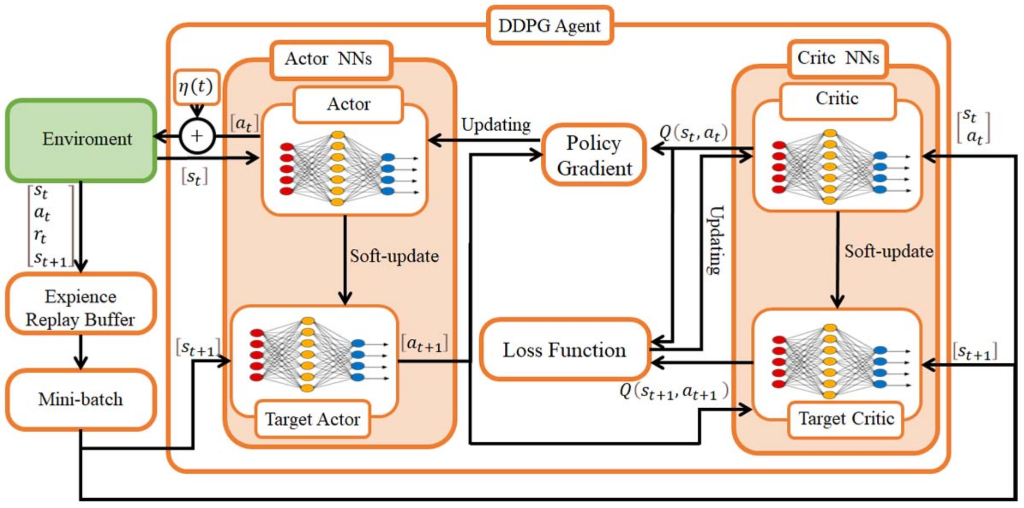


Fig. 5.2. DDPG control architecture for the computation of the control action $u_i(t)$, $\forall i$.

values assumed by the control action $u_i(t) \in \mathbb{R}$. Hence, at each time instant, the i-th agent, interacting with the environment and based on the past and present observations state s_t, computes and performs the chosen action a_t, which influences the next state transition of the system, i.e., from s_t to s_{t+1}; then, the agent receives the new state s_{t+1} and the scalar reward $r_t = f(s_t, a_t)$.
As discussed in RAILS Deliverable 2.2 [5], the design of the reward function is a crucial issue in the definition of the DDPG control strategy, as it has to be properly defined in order to fulfill the main VCTS objectives, namely, increasing line capacity while mantaining a safe inter-distance among trains, according to the RBD paradigm. Therefore, a proper safe inter-train distance has been defined in Section 5.2.2, and is exploited for the evaluation of the reward function, which is defined as follows for each agent i:

$$r_t(s_t, a_t) = -w_1 \left\| v_i(t) - v_{i-1}(t) \right\|_2 - w_1 \left\| v_i(t) - v_0(t) \right\|_2 - w_2 \left\| p_i(t) - p_{i-1}(t) - d_{s,i}(t) \right\|_2 - \alpha(t)\ P_s, \tag{5.11}$$

where w_1, w_2 are positive weighting factors; P_s defines a penalty factor weighted by the Boolean variable $\alpha(t)$, which is set to 1 when the relative distance between consist i and the train ahead $i-1$ exceeds the security distance, i.e. $p_i(t) - p_{i-1}(t) < d_{s,i}(t)$; 0 otherwise. In this way, the i-th agent is rewarded if consist i reaches the speed of consist $i-1$ and 0 without exceeding the relative security distance $d_{s,i}(t)$, while it is penalized when dangerous actions are imposed on the i-th train motion. Note that, the quadratic terms in (5.11) can bound the overshoot and the undershoot of the train motion.
Based on the values of the reward function, since the space s_t is fully-observed, the i-th agent computes the following discounted cumulative reward:

$$R_t = \sum_{k=T}^{t} \gamma^{(k-t)}\ r_t(s_k, a_k), \tag{5.12}$$

being T the first time-step already stored into the experience reply buffer at the time step t. Note that the expected return depends both on the observation state and the undertaken action a_t.
The control input $u_i(t)$ is, hence, selected by the actor-network via iterative steps. These

also involve the critic, the target networks and the predictions of the correct behaviour to be imposed with the aim of maximizing the cumulative reward function (5.12), i.e.:

$$u_i(t) = a_t = \mu(s_t|\theta_\mu) + \eta_t, \tag{5.13}$$

where $\mu(s_t)$ is the actor-state policy mapping sequence, with actor-network weights θ_μ; η_t represents the exploration noise modelled as, e.g., *White Noise* or *Ornstein-Uhlenbeck* process and is introduced only for the DDPG training purpose. The chosen action $u_i(t)$, depending on the $\varepsilon - greedy$ method [44], is computed through a Q-value function estimation, which in the optimal case is defined as

$$Q^\star(s_t, a_t) = \max_\mu \mathbb{E}\Big[R_t \mid s_t, a_t, \mu\Big]. \tag{5.14}$$

This estimation, carried out by the critic-networks, is exploited to update θ_μ at each time step, according to the following gradient-descent equation:

$$\theta_\mu \leftarrow \alpha \Big[\nabla_\mu \, Q(s_t, \mu_\theta(s_t|\theta_\mu)) \, \nabla_{\theta_\mu} \, \mu(s_t|\theta_\mu)\Big], \tag{5.15}$$

where $Q(s_t, \mu(s_t|\theta_\mu))$ is the action-state Q-function, i.e., the critic-network policy mapping sequence, with weights θ_Q. Similarly, θ_Q are updated such that the following Mean-Squared Bellman Error function [46] is minimised:

$$L(\theta_Q) \;\; = \mathbb{E}_{\mu(s_t),s_t}\Big[\Big(Q(s_t, \mu(s_t|\theta_\mu)|\theta_Q) - (r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}|\phi_\mu)|\phi_Q))\Big)^2\Big], \tag{5.16}$$

being s_{t+1} the prediction of future observations, while $\mu(s_{t+1}|\phi_\mu)$ is the related future action to be undertaken. Hence, the next Q-value, namely $Q(s_{t+1}, \mu(s_{t+1}|\phi_\mu)|\phi_Q)$, is evaluated via the exploitation of both the target actor and critic-networks. The weights of the target critic and target actor, i.e. ϕ_μ and ϕ_Q, are then computed according to the soft update fashion [47] as:

$$\phi_\mu \leftarrow \tau \, \phi_\mu + (1-\tau)\theta_\mu, \tag{5.17}$$

$$\phi_Q \leftarrow \tau \, \phi_Q + (1-\tau)\theta_Q, \tag{5.18}$$

with $\tau \ll 1$ being the updating probability. In this way, the target network values are constrained to change slowly, hence improving the convergence stability property. Algorithm (1) summarizes the design stage of the DRL VC robust control $u_i(t)$.

5.2.4.1. Neural Network Structures

The Neural Networks (NNs) involved in the DDPG control strategy are exploited to model the actor, the critic and their respectively target NNs. Their structure are defined in terms of Activation Functions (AFs), numbers of layers χ and neurons $\lambda_{z,r}$, $\forall \; z = 1, \cdots, \chi$ with $r = 1, \cdots, \Lambda_z$, being Λ_z the number of neurons for the z-th layer. The structural parameters χ and Λ_z are selected considering a trade-off between performance and limited computational burden required at the on-board train controller device. The related target NNs are characterised by the same structure as the main ones.

The architecture of the critic NN, depicted in Fig. 5.3(a), is composed of $\chi = 9$ layers, characterized by two different input paths and one output path, namely: $i)$ the *state-path* takes in

Algorithm 1: DDPG algorithm

```
Initialization
Randomly initialize critic-network $Q(s_t, a_t|\theta_Q)$ and actor-network $\mu(s_t|\theta_\mu)$ with weights $\theta_Q$
  and $\theta_\mu$.
Randomly initialize target networks $Q_{targ}(s_t, a_t|\phi_Q)$ and $\mu_{targ}(s_t|\phi_\mu)$ with weights $\phi_Q$ and
  $\phi_\mu$.
Define the DNNs Hyper-Parameters.
Computation
for $episode = 1$ to $N_{episode}$ do
    $s_0 \leftarrow$ Reset Environment
    while $t_k = 1$ to $T_{steps}$ or $!done$ do
        if random value $\leq \epsilon$ then
            $a_t \leftarrow$ random action
        else
            $a_t = \mu(s_t|\theta_\mu) + \eta_t$
        end
        $s_{t+1}, r_t \leftarrow$ StepEnvironment($a_t$)
        Store tuple $(s_t, a_t, r_t, s_{t+1})$
        Sample mini-batch $M_{batch}$ from replay memory
        Compute targets
        if doTraining then
            Update actor NN according to (5.15) and critic NN by gradient descent of
              (5.16)
            Update target networks with soft-update (5.17), (5.18)
        end
    end
end
```

input the 6 observations, i.e. s_t, for the Q-value estimation and it is composed of 1 normalized Input layer and 4 fully-connected Dense layers, i.e., $2, 3, 4, 5$; ii) the *action-path* maps the relation between the chosen action a_t and the Q-value, and is composed of 1 normalized Input layer and 1 Dense layer; iii) the *output path* is composed of 2 layers, where the former, i.e., the 8^{th} Connection layer, aims at connecting the two input paths merging their information to estimate the Q-value. The number of neurons Λ_z for the two Normalized Input layers, related to the *state-path* and *action-path*, are equal to the number of the respective inputs, i.e. $\Lambda_1 = 6$ and $\Lambda_6 = 1$, respectively. The number of neurons related to 9^{th} Output layer, instead, is equal to the number of output, i.e. $\Lambda_9 = 1$. Moreover, the number of neurons for Dense layers is equal to:

$$\Lambda_j = 2^{8-j} \quad \forall j = 1, \cdots, \mathcal{D}, \tag{5.19}$$

where $\mathcal{D} = \{2, 3, 4, 5, 6, 7, 8\}$ is the subset indexes of layers χ associated to Dense layers. Therefore, while the first Dense layer, i.e., 2^{nd} Dense layer, is characterized by $\Lambda_2 = 256$, the neurons number for the others $j \in \mathcal{D}$ decreases as a power function. Finally, as AFs, we select the Rectified Linear Units (RELU) one for the first $\chi - 1$ layers and the linear one for the layer $\chi = 9$. Critics NN features are summarised in Table 5.2(a).
The actor NN, reported in Fig. 5.3(b), is composed of 1 Normalized Input layer, 4 Dense lay-

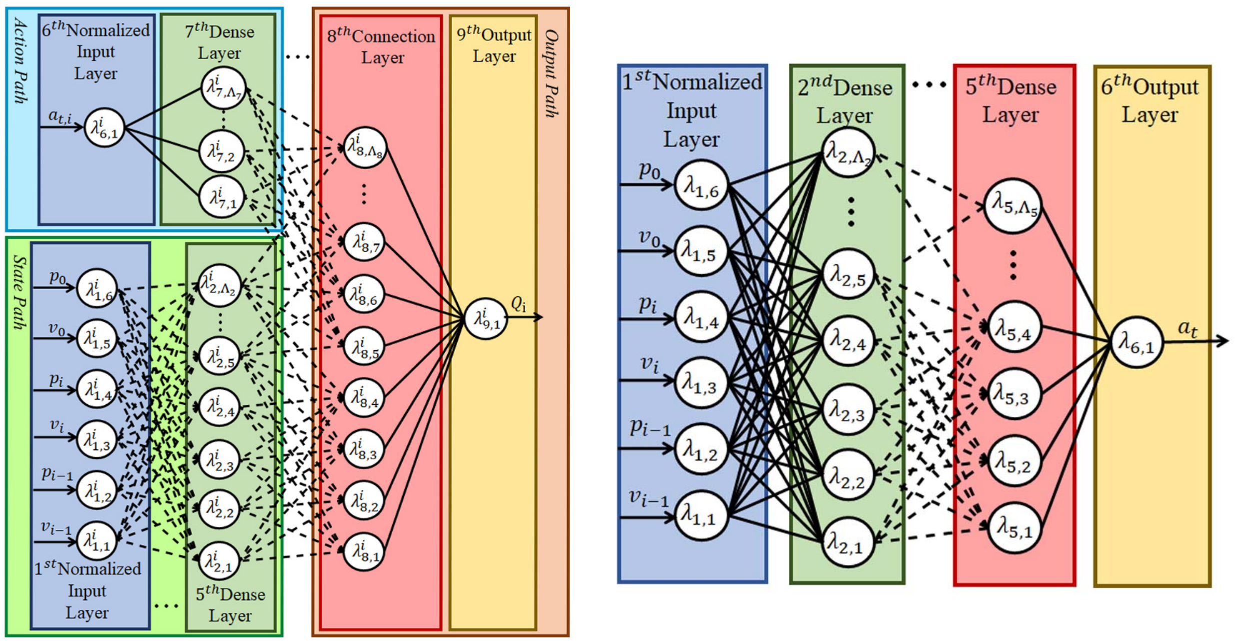

(a) (b)

Fig. 5.3. DDPG Neural Networks structure: (a) Critic Neural Networks; (b) Actor Neural Networks.

ers and one single Output layer, for a total of $\chi = 6$ layers. The normalized Input layer takes as input the observation vector s_t and, hence, accounted for 6 neurons. The first Dense Layer is composed of 256 neurons, while the remaining dense layers present a number of neurons according to (5.19). Moreover, the Output Layer includes only one neuron. Finally, as AFs, we select the Rectified Linear Units (RELU) one for the first $\chi - 1$ layers and a hyperbolic tangent (tanh) for the Output layer to let the actor NN compute a_t while constraining it in the range $[-1, 1]$. Actors NN features are summarised in Table 5.2(b).

5.2.4.2. Exploration Strategy

The exploration strategy is crucial for the design of the actor/critic NNs since it guarantees the optimization of the reward function during the agent training phase. Namely, it allows the agent to explore the unknown surrounding environment. To this aim, the approach first considers the introduction of the exploration noise, here modelled as an Ornstein-Uhlenbeck process [48], i.e.:

$$\dot{\eta}(t) = -\beta\, \eta(t) + \delta\, \mathcal{N}(t) \tag{5.20}$$

where β and δ are positive parameters, while $\mathcal{N}(t)$ is a White Noise process with mean equal to zero and standard deviation equal to 0.10. Note that the exploration noise acts at each time-step.

Then, we leverage the ε-*greedy* method to improve the agent training performance. Specifically, at each time-step, it generates a random value ρ_t and compares this latter with the threshold evaluated for the current training episode, i.e., $\varepsilon_{episode}$. If ρ_t is within the exploration area ($\rho_t \leq \varepsilon_{episode}$), the agent chooses at time-step t a random action a_t. Conversely, if ρ_t is inside the exploitation area ($\rho_t > \varepsilon_{episode}$), the agent chooses the action as in (5.13). The dynamics of $\varepsilon_{episode}$ along all the appraised episodes number $N_{episode}$ are described with the

Table 5.2: DDPG Architecture

(a) Critic Neural Network Structure

State-Path		
z^{th}-Layers	Λ_z	Activation function
1^{st}-Normalized Input layer	6	RELU
2^{nd}-Dense layer dims	256	RELU
3^{th}-Dense layer dims	128	RELU
4^{th}-Dense layer dims	64	RELU
5^{th}-Dense layer dims	32	RELU
Action-Path		
z^{th}-Layers	Λ_z	Activation function
6^{th}-Normalized Input layer	1	RELU
7^{th}-Dense layer dims	256	RELU
Output-Path		
z^{th}-Layers	Λ_z	Activation function
8^{th}-Connection Layer	16	RELU
9^{th}-Output Layer	1	Linear

(b) Actor Neural Network Structure

z^{th}-Layers	Λ_z	Activation function
1^{st}-Normalized Input layer	6	RELU
2^{nd}-Dense layer dims	256	RELU
3^{th}-Dense layer dims	128	RELU
4^{th}-Dense layer dims	64	RELU
5^{th}-Dense layer dims	32	RELU
6^{th}-Output layer dims	1	$tanh$

exponential ε-*greedy* method, i.e.:

$$\varepsilon_{episode+1} = \varepsilon_{episode}\ e^{\varepsilon_d} + \varepsilon_f, \tag{5.21}$$

where ε_f is the minimum exploration value for $episode \to N_{episode}$; ε_d is the decay rate of the ε-*greedy* law. The initial value for $\varepsilon_{episode}$ is set to $\varepsilon_0 = 1$.

5.3. Dataset Generation

Another critical issue highlighted in RAILS Deliverable 2.2 [5], strictly related to the training and validation phases of the DDPG controller, is represented by the need of advanced simulation platforms [49], since any dataset would be inadequate to work in high-dimensional and continuous action spaces. Therefore, we propose an *ad-hoc* VCTS simulation platform for the DDPG control design and validation purposes.
The VCTS simulator is implemented by leveraging Python v3.9 and Conda interpreter, as well as the combination of some open-source libraries, i.e. Keras, Tensorflow and OpenAiGym.

This latter has been suitably adapted to emulate the functioning of the tactical layer enabling trains VC, in addition to a basic logic for the emulation of the strategic layer. We remind that, according to [36], the latter is in charge of defining the platoons, their composition and ordering, based on compatibility, destinations and schedules. It also defines when and where the cooperative manoeuvres, such as coupling and decoupling, should occur.
The hardware requirements for the proposed simulation platform are a Nvidia-GPU to train the DDPG agent and a generic CPU for the validation phase. Note that this kind of virtual simulation is a cost-effective alternative for VCTS testing, since the Python language along with the exploited libraries are open-source tools.
The key components of the proposed simulation platform are reported in Fig. 5.4 and are detailed in the following:

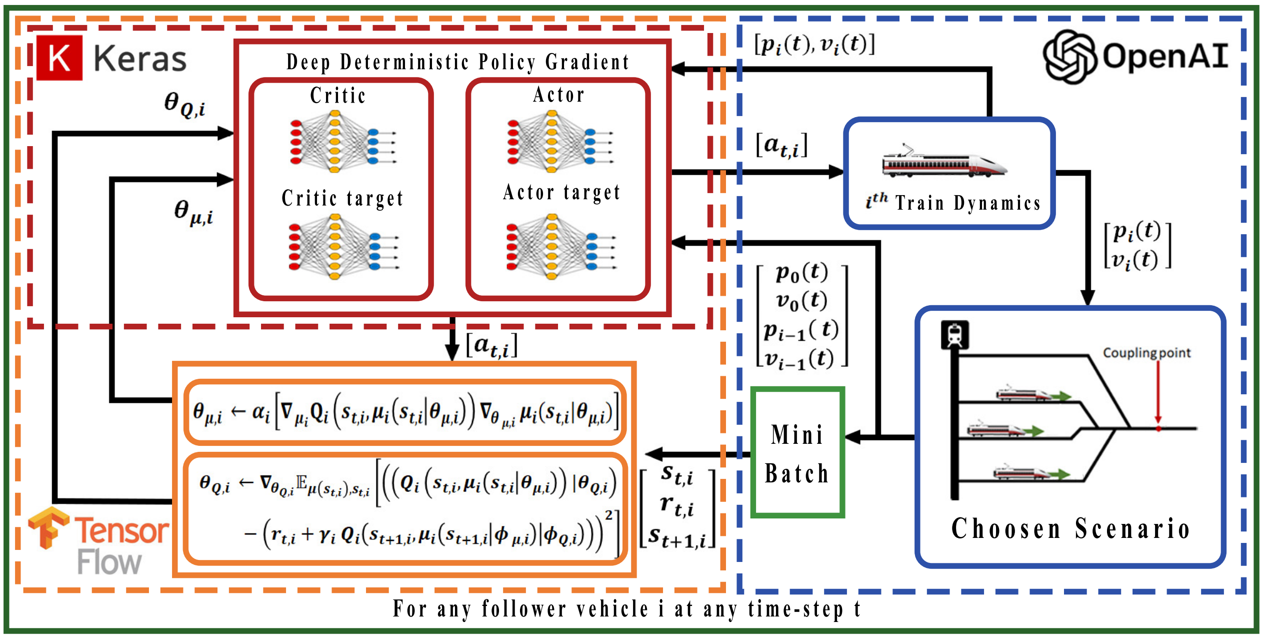

Keras
$\theta_{Q,i}$
$\theta_{\mu,i}$
Deep Deterministic Policy Gradient
Critic
Critic target
Actor
Actor target
$[a_{t,i}]$
$\theta_{\mu,i} \leftarrow \alpha_i [\nabla_{\mu_i} Q_i(s_{t,i}, \mu_i(s_{t,i}|\theta_{\mu,i})) \nabla_{\theta_{\mu,i}} \mu_i(s_{t,i}|\theta_{\mu,i})]$
$\theta_{Q,i} \leftarrow \nabla_{\theta_{Q,i}} \mathbb{E}_{\mu(s_{t,i}), s_{t,i}} [((Q_i(s_{t,i}, \mu_i(s_{t,i}|\theta_{\mu,i}))|\theta_{Q,i}) - (r_{t,i} + \gamma_i Q_i(s_{t+1,i}, \mu_i(s_{t+1,i}|\phi_{\mu,i})|\phi_{Q,i})))^2]$
TensorFlow
$[p_i(t), v_i(t)]$
OpenAI
$[a_{t,i}]$
i^{th} Train Dynamics
$\begin{bmatrix} p_i(t) \\ v_i(t) \end{bmatrix}$
$\begin{bmatrix} p_0(t) \\ v_0(t) \\ p_{i-1}(t) \\ v_{i-1}(t) \end{bmatrix}$
$\begin{bmatrix} s_{t,i} \\ r_{t,i} \\ s_{t+1,i} \end{bmatrix}$
Mini Batch
Coupling point
Choosen Scenario
For any follower vehicle i at any time-step t

Fig. 5.4. Overview of the proposed simulation platform

- **Keras v2.8.0** is used for the definition and the building up of the DNN related to the actor, critic and the corresponding target networks;

- **Tensorflow v2.8.0** allows the computation of the Gradient Descent formula in (5.15) and the Mean-Squared Bellman error in (5.16) via the *Adam* optimizer;

- **OpenAiGym v0.23** is properly enlarged with the libraries related to train dynamics, communication network and strategic control layer logic, in order to simulate the VCTS paradigm environment.

Specifically, Keras and Tensorflow implement the designed DDPG tactical layer reported in Section 5.2.4. The OpenAiGym, instead, emulates: i) the heterogeneous nonlinear dynamic model of the trains as in (5.1)-(5.2) together with the time-varying parameters uncertainties arising from the different operating driving conditions and train heterogeneity [50, 51]; ii) the T2T communication networks among the trains as well as the T2I communications, both for trains-RBC which enable information sharing with the uncertainties affecting the train location information, and for the on-board sensors measurements errors (see Remark 1); iii) the strategic layer logic responsible for the coupling/decoupling conditions and commands [37]; iv) the railway environment, the induced uncertainties and the nonlinearities effects considering the different values assumed by the adhesion factor for different weather conditions [52], several curvature radii and gradient profiles according to the European railway lines [1], different windy conditions [39, 53, 54].

5.4. Training and Validation

The simulation platform designed in Section 5.3 has a twofold task: 1) the training of the designed DDPG agent; 2) the validation of the proposed VCTS tactical layer controller. Thus, based on the specific usage, selected via the main file, only some libraries of the platform are invoked. In the following, both the training and validation phases are detailed.

5.4.1. DDPG Agent Training

The training phase requires the initialization of the hyper-parameters setup and the random generation of the model parameters, of the driving scenarios (i.e. initial conditions of the preceding and the follower train), as well as the unknown surrounding railway nonlinearities. Specifically, the train dynamical control-oriented model in (5.1) is discretized via a sampling time of $T_c = 0.1[s]$ and integrated with a Trapezoidal-Euler algorithm. Moreover, the selected hyper-parameters are reported in Table 5.3, while the training horizon is set to $N_{episode} = 5000$ with an iteration step, for each episode, of $N_{steps} = 10000$, corresponding to a simulation time of $T_s = 1000\ [s]$. Note that, N_{steps} could be shorter if some special conditions, indicated via the flag $done$ in Algorithm (1) occur, i.e., a possible collision among trains or a negative value for trains speed.
As to the training scenario, for each train i we consider: i) the preceding $i-1$ and the virtual leader move according to time-varying speed profile with values raging into $[20; 83]\ [m/s]$; ii) the initial value for the inter-train distance between train i and $i-1$, i.e. $p_i(0) - p_{i-1}(0)$, varies every 25 episodes into the range $[250; 1000]\ [m]$; iii) the parameters of the i-th train randomly vary every 50 episodes within the range $[0; 20]\%$ of the nominal values; iv) the acceleration component of the external force due to the unknown railway nonlinearities $\omega_i(t)$ is assumed to be constant over an episode and varying into the range $[-0.9,\ 0.9]\ [m/s^2]$ every 10 episodes. Furthermore, the reward function parameters in (5.11) are selected as: $w_1 = 0.8$, $w_2 = 0.6$ and $P_s = 1000$.

The training process is performed via Intel® Core^{TM} i7-11900K 3.2 GHz, 64 GB 4.400 MHz DDR5 of RAM memory, GPU NVIDIA ® GeForce RTX^{TM} 3080 (with 16 GB GDDR6 of memory) and Windows 11 system by exploiting the designed VCTS platform detailed in Section 5.3. More in detail, Tensorflow computes the new weights of the DNNs, while Keras receives them and, based on the last observation vector, computes a new action to be imposed to the train i. Accordingly, this action is shared with OpenAiGym, which updates the kinematic train information for the next iterative step.
The effectiveness of the training results are shown in Fig. 5.5, where the trend of the reward function goes towards zero over the epochs, hence confirming that the DDPG agent has *learned* the correct behaviour to be imposed to train i *by doing*.

[1] https://unece.org/DAM/trans/main/ter/terdocs/TER_High-Speed_Master_Plan_Study.pdf

Table 5.3: DDPG hyper-parameters setup

Parameter	Value	Parameter	Value
Replay buffer size	100.000	Batch size	64
Delay steps for training	1024	Update step	64
Discount factor γ	0.99	Smooth factor τ	0.001
Epsilon starting value ϵ	0.99	Epsilon final value ϵ_f	0.05
Critic learning-rate	0.001	Actor learning-rate	0.0002
Ornstein Uhlenbeck mean	0.5	Ornstein Uhlenbeck divergence	0.6

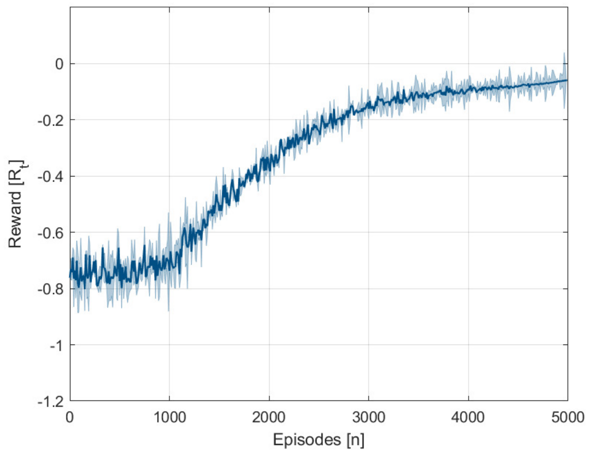


Fig. 5.5. Performance results about the training phase of the DDPG agent. Trend of the reward function over the 5000 epochs: the blue line are the average normalized episode reward while the shaded blue area is the variance of the reward function over the 1000 simulation-steps.

5.4.2. Validation of the VCTS Controller

The validation phase is carried out to show the effectiveness of the proposed DDPG tactical layer controller in guaranteeing the VC for HST convoys. To do this, concrete operational scenarios involving a combination of manoeuvres, as suggested by Shift2Rail [55] and [38, 56], are considered. They are defined in terms of: number of trains composing the VCTS; model and braking characteristics of the consists, i.e., the train parameters; initialization of the trains positions and speeds; the strategic logic commanding the specific cooperative manoeuvres to be performed; the selection of the travelling route and of the weather/windy conditions.
Specifically, in each scenario we consider a VCTS composed of 4 HSTs plus the RBC acting as a virtual leader. Each train i $(i = 1, \cdots, 4)$ communicates with the preceding vehicle $i-1$ via the T2T communication network and with the RBC via the T2I paradigm, as depicted in Fig. 5.1.
The uncertain parameters for each train i are selected according to [57], with nominal value reported in Table 5.4 along with their respective uncertain variation ranges.
In order to evaluate the flexibility/robustness of the proposed controller w.r.t. these unknown uncertain parameters, as well as all the possible combinations of them, the Monte Carlo method is exploited. This allows to achieve a broad test coverage by varying a large number

Table 5.4: Dynamics train parameters for the validation analysis

Parameter	Value				Ranges
$\bar{M}_i\ [t]$	$\bar{M}_1 = 475$	$\bar{M}_2 = 489$	$\bar{M}_3 = 496$	$\bar{M}_4 = 502$	$\pm 30\%$
$\bar{C}_{1,i}\ [kg\ m/s^2]$	$\bar{C}_{3,1} = 0.8420$	$\bar{C}_{3,2} = 0.7314$	$\bar{C}_{3,3} = 0.6852$	$\bar{C}_{3,4} = 0.7862$	$\pm 20\%$
$\bar{C}_{2,i}\ [kg/s]$	$\bar{C}_{2,1} = 0.00681$	$\bar{C}_{2,2} = 0.00613$	$\bar{C}_{2,3} = 0.00708$	$\bar{C}_{2,4} = 0.00662$	$\pm 20\%$
$\bar{C}_{3,i}\ [kg/m]$	$\bar{C}_{1,1} = 000209$	$\bar{C}_{1,2} = 000105$	$\bar{C}_{1,3} = 000149$	$\bar{C}_{1,4} = 000187$	$\pm 20\%$

of parameters values in a wide range, which is commonly not possible, easy, or cost-efficient in test field experiments (e.g., due to the limited number of vehicles usually involved in the experiments and with specific characteristics and mechanical features).
The train unknown dynamic nonlinearities $\omega_i(t)$, given the friction force and the external disturbances, are modelled as [41]:

$$\omega_i(t) = a_g(t) + a_R(t) + a_W(t), \tag{5.22}$$

with

$$a_g(t) = -\mu(t)g \times slope(t), \tag{5.23a}$$
$$a_R(t) = -\frac{6 \cdot 10^6}{R(t)}, \tag{5.23b}$$
$$a_W(t) \in \mathcal{L}_2. \tag{5.23c}$$

$a_g(t)\ [m/s^2]$ in (5.23a) is the acceleration component of the friction force, where $\mu(t) = 0.25$ is the railway line adhesion coefficient in dry conditions, $g\ [m/s^2]$ is the gravity acceleration while $slope(t)\ [mm/m]$ is the gradient profile. $a_R(t)\ [m/s^2]$ in (5.23b) is the acceleration curving component where $R(t)\ [m]$ is the curvature radius. $a_W(t)\ [m/s^2]$ is the resistance wind acceleration component. The value of $slope(t)$ and $R(t)$ are based on the Italian railway lines characteristics [2]. The safe inter-train distance between consecutive vehicles is modelled as in (5.6) with braking capability parameters, $\forall i$ as in Table 5.5. Without loss of generality, we assume for each train the same power-train system, i.e., the same braking performance.
As already pointed out, the validation process is performed exploiting the designed VCTS platform detailed in Section 5.3. Precisely, the data-flow enabling the validation phase involves just *Keras* and *OpenAiGym*, since the updating process of the DNNs via Tensorflow is not required. Thus, *OpenAiGym* emulates the selected driving scenario, while *Keras* computes the optimal action to be performed by the trained model.

[2]https://www.gazzettaufficiale.it/atto/serie_generale/caricaArticolo?art.progressivo=0&art.idArticolo=17&art.versione=1&art.codiceRedazionale=089G0159&art.dataPubblicazioneGazzetta=1989-04-24&art.idGruppo=0&art.idSottoArticolo1=10&art.idSottoArticolo=1&art.flagTipoArticolo=2

Table 5.5: Train braking performance

Parameters	Value	Parameters	Value	Parameters	Value
$v_{max,i}$ $[m/s]$	85	$t_{b,i}$ $[s]$	0.3	$t_{bu,i}$ $[s]$	1
$a_{b,i}$ $[m/s^2]$	0.9	$t_{c,i}$ $[s]$	0.2	$\pm e_{s,i}$ $[m]$	± 0.5
$a_{d,i}$ $[m/s^2]$	1	$a_{re,i}$ $[m/s^2]$	0.15	s_m $[m]$	15
$a_{bs,i}$ $[m/s^2]$	0.6	$t_{e,i}$ $[s]$	1	$v_{es,i}$ $[m/s]$	± 3.6

5.5. Evaluation of Results

In what follows, we consider three specific operational scenarios, defined as in Section 5.4.2, which involve the following cooperative manoeuvres: i) VCTS Forming; ii) VCTS Splitting; iii) Leader-Tracking. Each of them is constructed by combining the basic coupling/decoupling for the VCTS (properly described in the technical report [55]), with the aim of emulating joining and leaving operations. It is worth noting that, with the aim of assessing the correct functioning of every single DDPG controller i, the coupling/decoupling operation i)/ii) is emulated such that each train i joins/leaves the forming/formed convoy at a different time instant. The strategic layer is emulated via a set of coupling/decoupling commands, along with the definition of the number of trains belonging to the VCTS and their position indexes within the platoon, which are defined based on the logic rules as in [38].
The virtual testing results in each operational scenario are discussed in the following.

5.5.1. Operational Scenario 1: VCTS Forming

In this driving scenario, each consist travels on different secondary railway lanes with a proper cruising speed until reaching the relative switching point merging these lanes towards the primary railway one, where the 4 HSTs have to move in a VCTS.
We assume that the consists start from different initial positions on their own railway lanes, i.e., $p_1(0) = 1800$ $[m]$, $p_2(0) = 1200$ $[m]$, $p_3(0) = 600$ $[m]$, $p_4(0) = 0$ $[m]$, with a different cruising speeds, i.e., $v_1(0) = 83.8$ $[m/s]$, $v_2(0) = 85$ $[m/s]$, $v_3(0) = 86.3$ $[m/s]$, $v_4(0) = 87.6$ $[m/s]$, until reaching the primary lane, where the VC can be enabled. The RBC is located on the primary lane and imposes the reference behaviour according to (5.4)-(5.5) with the following initial conditions: $p_0(0) = 2400$ $[m]$ and $v_0(0) = 75.0$ $[m/s]$. Note that, under the management of the signalling system, the switching towards the primary railway lane occurs without collisions [38]. At this lane, each consist moves at its cruising until the coupling condition w.r.t. the preceding train is verified. Now, the emulated strategic layer sends the coupling command to the i-th train at the instant time $\tilde{t}_{coupling,i}$ $[s]$ and the virtual leader imposes the reference speed of 75.0 $[m/s]$ to the i-th train.
Virtual simulation results are reported in Fig. 5.6. Based on the trains initial conditions, the coupling command occurs for each train i at $\tilde{t}_{coupling,1} \approx 200$ $[s]$, $\tilde{t}_{coupling,2} \approx 350$ $[s]$, $\tilde{t}_{coupling,3} \approx 450$ $[s]$, $\tilde{t}_{coupling,4} \approx 550$ $[s]$, respectively. Accordingly, Fig. 5.6(a)-(b) disclose the effectiveness and the robustness of the proposed DDPG control in guaranteeing the VC of each train i w.r.t. its predecessor at the time instant $\tilde{t}_{coupling,i}$ while maintaining a safe inter-train distance $e_{p,i}(t)$ (see Fig. 5.6(c))- with bounded speed error $e_{v,i}(t)$ (see Fig. 5.6(d))- despite presence of unknown uncertainties and nonlinearities. Fig. 5.6(a)-(b)-(c)-(d)-(e) are obtained by leveraging the Monte Carlo method for the unknown time-varying train

parameters with nominal values and range variations reported in Table 5.4. As it is possible to observe in Fig. 5.6(e), for each time-varying parameters combination, the DDPG provides a suitable trend for the control effort to be imposed on the i-th HST in order to guarantee the manoeuvre task accomplishment according to the VCTS control objectives (5.9)-(5.10) (as also highlighted in Fig. 5.6(a)-(b)-(c)-(d)), hence confirming the ability of the proposed control technique in counteracting the unknown factors. To better show the feasibility of the imposed control action, we report in Fig. 5.6(f) the traction force trend for the nominal value of train parameters as listed in Table 5.4.

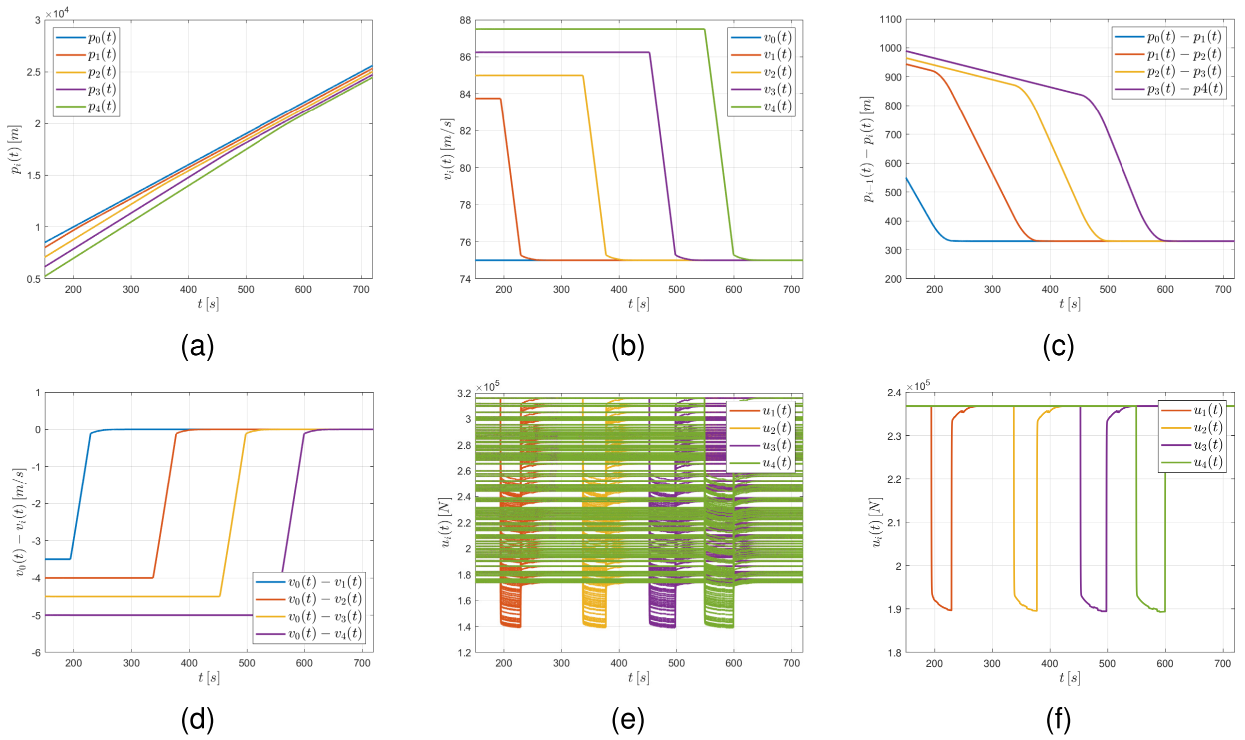


Fig. 5.6. Performance of the DDPG tactical layer robust control. VCTS Forming operational scenario. Time history of: (a) train position $p_i\ (i = 1, \cdots, 4)$ plus virtual leader position $p_0(t)$; (b) train speed $v_i(t)\ (i = 1, \cdots, 4)$ plus virtual leader speed $v_0(t)$; (c) position errors computed as $p_{i-1}(t) - p_i(t)\ (i = 1, \cdots, 4)$; (d) speed errors computed as $v_0(t) - v_i(t)$ $(i = 1, \cdots, 4)$; (e) various train traction input force curves $u_i(t)\ (i = 1, \cdots, 4)$; (f) train traction input force $u_i(t)\ (i = 1, \cdots, 4)$ in the nominal parameters case, as listed in Table 5.4.

5.5.2. Operational Scenario 2: VCTS Splitting

Here, the HSTs travel in VC formation on a primary railway lane at the cruising speed of $83\ [m/s]$ (and with safe inter-train distance $d_{s,i}(t)$) until reaching several diverging junction points. Note that, in this case, for each decoupling train, the safety distance explicitly takes into account the absolute braking distance with respect to the related diverging junction point via $d_{lim,i-1}$ in (5.8), computed according to the imposed track constraints. The consists split the primary railway section into 4 secondary lanes, where the 4 HSTs move according to the proper journey profile. Specifically, the manoeuvre is composed of a combination of 4 sequential decoupling operations, starting from the vehicle in tail according to the strategic layer logic. Once the leaving request is accepted, the leave command is sent to the i-th train at time instant $\tilde{t}_{decoupling,i}\ [s]$.

Virtual simulation results are reported in Fig. 5.7, where it is possible appreciating the effectiveness of the DDPG control in performing the split manoeuvre in uncertain and unknown conditions. Indeed, when the first decoupling command is sent at $\tilde{t}_{decoupling,4} = 500\ [s]$, train 4 leaves the convoy, switches on its own secondary lane and decreases its speed to $73\ [m/s]$ (see Fig. 5.7(b)). Likewise, train 3 leaves the formation at time $\tilde{t}_{decoupling,3} = 600\ [s]$ and approaches the new cruising speed of $76\ [m/s]$, then followed by the 2-nd and 1-st train which exit the convoy (at $\tilde{t}_{decoupling,2} = 700\ [s]$ and $\tilde{t}_{decoupling,1} = 800\ [s]$) and travel at their new constant speeds of $v_2 = 80\ [m/s]$ and $v_1 = 83.8\ [m/s]$, respectively (see Fig. 5.7(b)). In Fig. 5.7 (a), the reported train positions reflect the aforementioned events. The ability of DDPG control in performing the manoeuvre is further shown in Fig. 5.7 (c)-(d), where the time-histories of the relative position of each train w.r.t. its predecessor, as well as speed errors w.r.t. the leader, are provided. Since the robustness of the DDPG control strategy is evaluated via Monte Carlo method, we report all the suitable trends for the traction force in each uncertain case in Fig. 5.7 (e), while Fig. 5.7 (f) reports a single realization of the control inputs, obtained according to the nominal model parameters listed in Table 5.4.

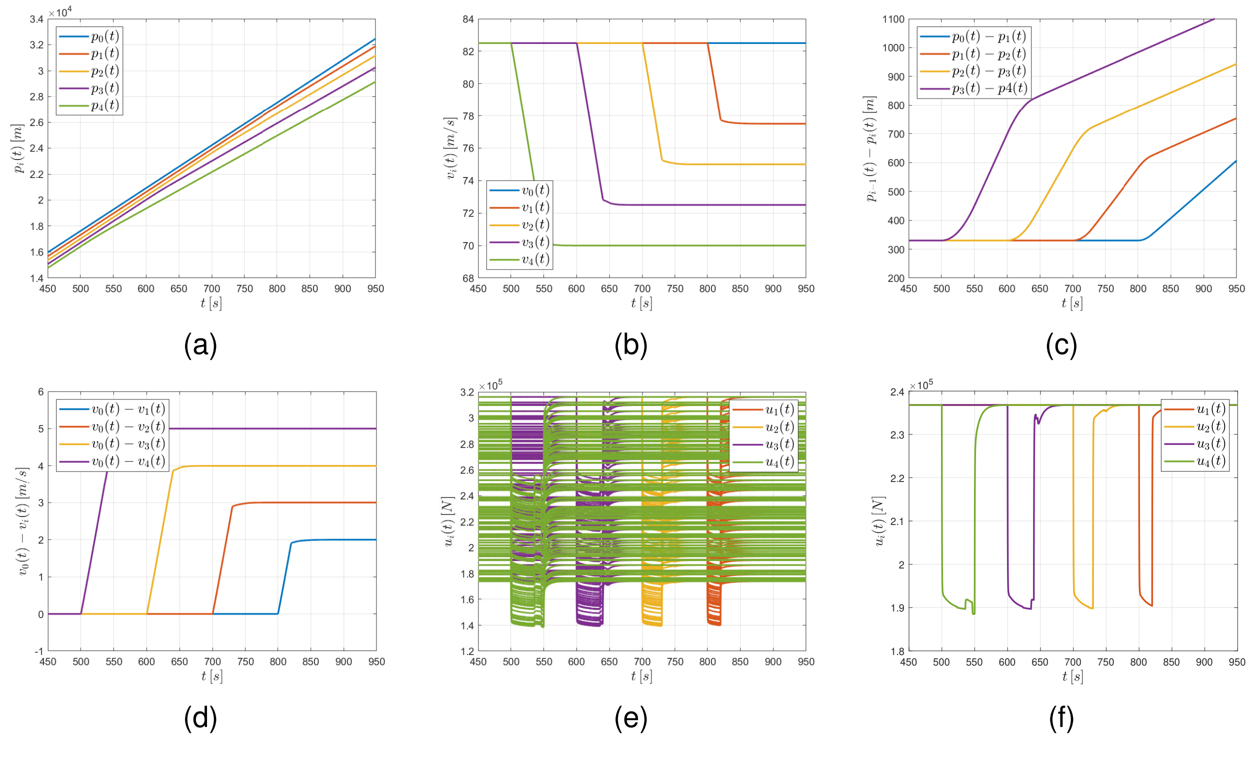


Fig. 5.7. Performance of the DDPG-based tactical layer robust control. VCTS splitting operational scenario. Time history of: (a) train position $p_i\ (i = 1, \cdots, 4)$ plus virtual leader position $p_0(t)$; (b) train speed $v_i(t)\ (i = 1, \cdots, 4)$ plus virtual leader speed $v_0(t)$; (c) position errors computed as $p_{i-1}(t) - p_i(t)\ (i = 1, \cdots, 4)$; (d) speed errors computed as $v_0(t) - v_i(t)$ $(i = 1, \cdots, 4)$; (e) various train traction input force curves $u_i(t)\ (i = 1, \cdots, 4)$; (f) train traction input force $u_i(t)\ (i = 1, \cdots, 4)$ in the nominal parameters case, as listed in Table 5.4.

5.5.3. Operational Scenario 3: Leader Tracking

In this operational scenario, we consider that the 4 HSTs travel in formation on the primary lane, with a constant speed of $85\ [m/s]$ while maintaining a safe inter-train distance (whose initial condition is of $d_{s,i}(t) = 339\ [m]$), when the leader imposes a trapezoidal speed manoeuvre, consisting of a succession of acceleration and deceleration phases. Specifically, at the time instant $t = 400\ [s]$, the leader accelerates until reaching the new constant speed of $88\ [m/s]$; then, at time instant $t = 850\ [s]$, a braking manoeuvre is performed until achieving again the initial constant speed of $85\ [m/s]$. The effectiveness of the proposed DDPG control strategy in ensuring the leader tracking with a safe inter-train distance $d_{s,i}(t)$ (see (5.6)) is shown in Fig. 5.8: each train tracks the trapezoidal speed profile $v_0(t)$ (see Fig. 5.8(b)), while maintaining the safe inter-train distance (see Fig. 5.8(a)-(c)) with bounded and very small speed tracking errors (see Fig. 5.8(d)). The suitable trends computed by the DDPG controller via the Monte Carlo method for counteracting the different uncertain cases are disclosed in Fig. 5.8 (e), while Fig. 5.8 (f) depicts the realization of the control inputs when considering trains nominal parameters as defined in Table 5.4.

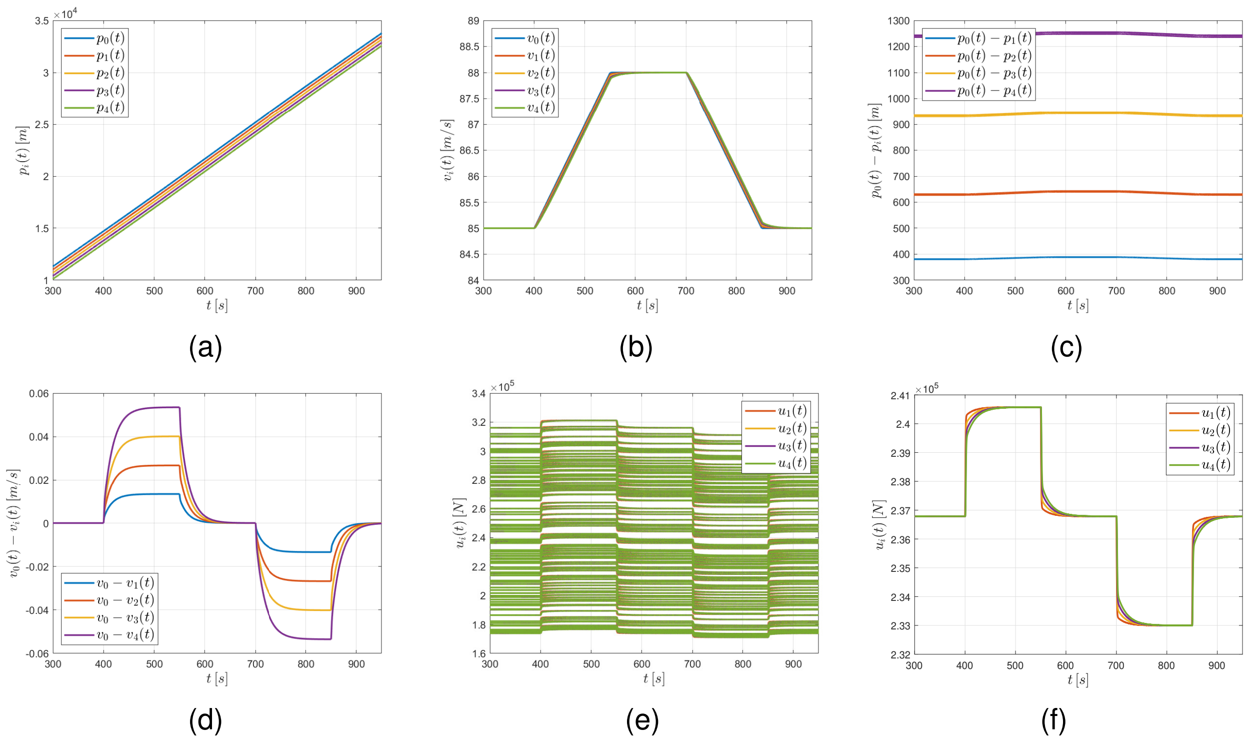


Fig. 5.8. Performance of the DDPG tactical layer robust control. Leader Tracking operational scenario. Time history of: (a) train position $p_i(t)$ $(i = 1, \cdots, 4)$ plus virtual leader position $p_0(t)$; (b) train speed $v_i(t)$ $(i = 1, \cdots, 4)$ plus virtual leader speed $v_0(t)$; (c) position errors computed as $p_0(t) - p_i(t)$ $(i = 1, \cdots, 4)$; (d) speed errors computed as $v_0(t) - v_i(t)$ $(i = 1, \cdots, 4)$; (e) various train traction input force curves $u_i(t)$ $(i = 1, \cdots, 4)$; (f) train traction input force $u_i(t)$ $(i = 1, \cdots, 4)$ in the nominal parameters case, as listed in Table 5.4.

5.6. Discussion of Results

In the previous sections, the effectiveness of the proposed model-free DRL control strategy to guarantee the achievement of the VC requirements and the fulfillment of the manoeuvres tasks in different operational scenarios, as well as the robustness w.r.t. unknown factors, has been shown through the validation phase. However, a question arises: what are the benefits of using a DRL method with respect to traditional model-based approaches, which are already assessed for the implementation of vehicle VC in the automotive field?

To answer the question, we carry out a comparison analysis between the performance of the DDPG controller and the one achievable via the optimized Model Predictive Control (MPC) strategy recently proposed in [57] considering the Leader Tracking scenario detailed in Section 5.5.3, in which gradients and slopes are disregarded. The comparison between the two approaches is carried out by leveraging the simulation platform deeply analyzed in Section 5.3, and is assessed via two suitable KPIs suggested in RAILS Deliverable 2.2 [5] (see Table 5.1). In particular, we exploit the *Tracking Index* (TI) [58], which is based on the tracking error (**KPI4**), and the *Energy Consumption* (EC) [34], corresponding to **KPI5**. We highlight that other KPIs as well as other operational scenarios can be considered for the comparison, since the selected ones are meant to be a benchmark for future research.
The TI$_i$ weights the tracking error for each vehicle i ($\forall i = 1, \cdots, 4$) as:

$$TI_i = \frac{1}{T_i} \int_0^{T_i} \left| SDE \cdot e_{p,i-1,i}(t) + SVE \cdot e_{v,0,i}(t) \right| dt \qquad (5.24)$$

where $T_i = 450\ [s]$ is the traveling time to perform the trapezoidal manoeuvre; SDE and SVE are positive weights that represent the sensitivity to the distance and the velocity error, which are set, according to [58], as $SDE = 1$ and $SVE = 10$; $e_{p,i-1,i}(t) = p_{i-1}(t) - p_i(t) - d_{s,i}(t)\ [m]$ and $e_{v,0,i}(t) = v_0(t) - v_i(t)\ [m/s]$ are the position and the speed error, respectively.
Instead, the EC$_i$ index for each consist i within the VCTS is computed as:

$$EC_i = \frac{1}{d_{pi}} \int_0^{T_i} v_i(t) \cdot u_i(t)\, dt, \qquad (5.25)$$

being $d_{pi} \approx 17\ [km]$ the distance travelled by train i during the manoeuvre, while $u_i\ [N]$ is the friction force required at the vehicle for travelling at the speed $v_i(t)\ [m/s^2]$.
Comparative results in Fig. 5.9(a) confirm the good tracking performance achievable via the proposed DDPG approach. Namely, the following variations in mean and standard deviation of the proposed approach and the MPC, respectively, can be appreciated: -70.32% and -65.22% for train 1; 0.65% and -3.84% for train 2; -17.72% and -36.68% for train 3; -16.43% and -14.29% for train 4. The TI results of each train computed for the two considered control strategies are summarised in Table 5.6. These results clearly indicate that the developed model-free controller better guarantees the reference tracking performance in the mean. Furthermore, the standard deviation reduction in the output distribution proves that the proposed approach is less sensitive to parameter uncertainties, differently from the model-based one which suffers from this issue.

The improvement in tracking performance, as well as the robustness property of the approach, also involves the train consumption; in this regard, comparative results w.r.t. MPC clearly disclose the efficiency of the DDPG tactical layer in energy saving, as depicted in Fig. 5.9(b). Specifically, the following variations in mean and standard deviation between DDPG and MPC, respectively, are obtained: -12.16% and -8.48% for train 1; -12.45% and -10.15% for train 2; -11.98% and -4.17% for train 3; -12.21% and -5.28% for train 4. The EC results of each train computed for the two considered control strategies are summarised in Table 5.7. It is worth highlighting that this energy saving is exactly due to the ability of the DDPG controller in counteracting different uncertainties factors and in guaranteeing the proper leader tracking in different driving conditions.

This experimental proof-of-concept has been proposed to give an answer to the research

Tracking Index	Control Strategy	Train 1	Train 2	Train 3	Train 4
Mean	DDPG	0.1662	0.5608	0.4560	0.4607
	MPC	0.5601	0.5571	0.5542	0.5513
	Variation (%)	-70.32	0.65	-17.72	-16.43
Standard Deviation	DDPG	0.1186	0.3269	0.2145	0.2892
	MPC	0.3411	0.3399	0.3387	0.3375
	Variation (%)	-65.22	-3.84	-36.68	-14.29

Table 5.6: Tracking Index mean and standard deviation of each train i $(i = 1, \dots, 4)$ for DDPG and MPC control strategies, including variations between DDPG and MPC results, respectively.

Energy Consumption	Control Strategy	Train 1	Train 2	Train 3	Train 4
Mean	DDPG	0.0776	0.0769	0.0769	0.0763
	MPC	0.0883	0.0878	0.0873	0.0869
	Variation (%)	-12.16	-12.45	-11.98	-12.21
Standard Deviation	DDPG	0.0073	0.0067	0.0067	0.0062
	MPC	0.0079	0.0074	0.0070	0.0065
	Variation (%)	-8.48	-10.15	-4.17	-5.28

Table 5.7: Energy Consumption $[kWh/km]$ mean and standard deviation of each train i $(i = 1, \dots, 4)$ for DDPG and MPC control strategies, including variations between DDPG and MPC results, respectively.

questions (see Section 5.1) arisen from the methodological analysis proposed in Deliverable D2.2 [5]. Results confirmed that the proposed approach, which leverages AI tools transferred from vehicle platooning (**RQ1**), could represent a valuable solution for the development of the tactical layer functionalities in the field of railway VC. The validation phase, indeed, showed the effectiveness of the approach in guaranteeing the tracking of the virtual leader reference speed and the maintenance of safe inter-train distances in different operational scenarios, despite the presence of uncertainties, nonlinearities, and disturbances due to train dynamics and unknown railway environmental effects (**RQ2**). Furthermore, from the evidences of the comparison analysis, it emerged that the proposed DRL control strategy could also provide several advantages w.r.t. the considered model-based method, such as:

- **capacity**: it could allow better reference tracking performance on average, thus enhancing lane capacity, which represents one of the main objectives of the VC paradigm;
- **flexibility**: differently from model-based approaches, which require a detailed modelling of the train dynamics and the surrounding environment, the DDPG controller is a model-free approach able to adapt the trains behaviour to the encountered driving scenarios, without any prior knowledge of the train dynamics and the travelling environment;
- **robustness**: the proposed solution is robust to parameters uncertainties compared to MPC, as it can better address trains heterogeneity, unknown and time-varying train dynamics, uncertainties and nonlinearities due to unknown track conditions and external

disturbances;

- **energy saving**: the proposed DDPG tactical layer could substantially reduce energy consumption, thus supporting energy saving.

In conclusion, this proof-of-concept allowed to investigate the possible adoption of AI techniques transferred from other transportation sectors to railway VC; its promising preliminary results can be considered a benchmark to inspire future developments towards the definition of a technology roadmap (**RQ3**).

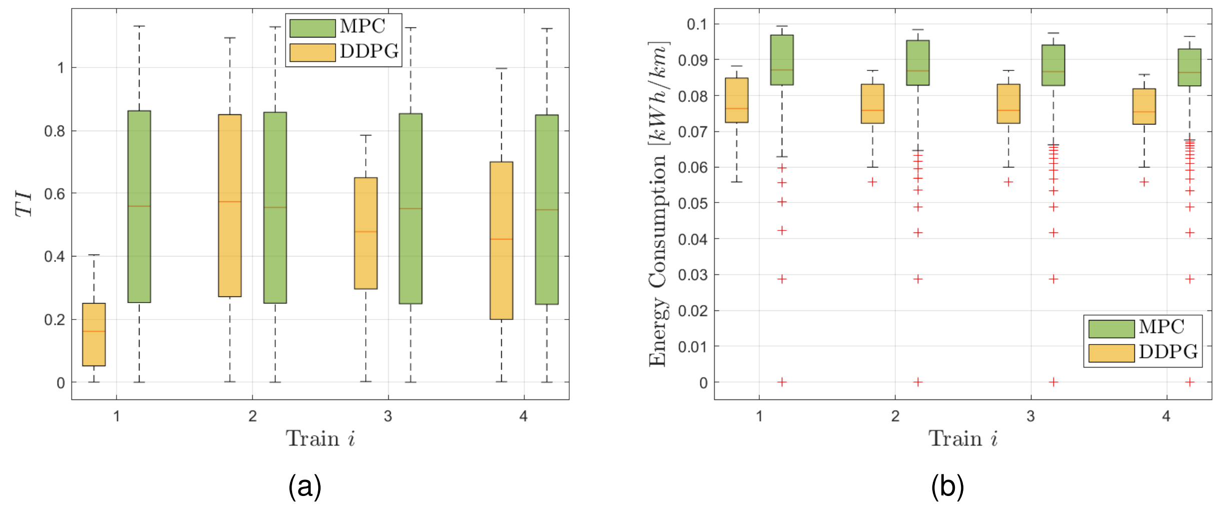


Fig. 5.9. Comparison analysis w.r.t. the MPC control strategy proposed in [57]. Leader Tracking operational scenario (see Section 5.5.3). (a): Tracking Index TI_i $(i = 1, \cdots, 4)$; (b): Energy Consumption EC_i $(i = 1, \cdots, 4)$.

6. Conclusions

In this deliverable, the potential of AI solutions towards a vision of safe automation and intelligent train control in the railway sector has been investigated through experimental proofs-of-concept. These last represents the continuation of the methodological analyses provided in the previous deliverable for two selected case studies, namely, “Obstacle Detection for Collision Avoidance”, and “Cooperative Driving for Virtual Coupling of Autonomous Trains”. In accordance with the objectives, techniques, and research questions identified in the previous deliverable, for each case study, an innovative approach which exploits AI techniques transferred from other transportation sectors has been provided. The effectiveness of the proposed strategies has been evaluated via experimental validation in concrete operational scenarios. Results showed that AI can represent a valuable solution for enhancing rail safety and automation. These proofs-of-concept are meant to be a first step to inspire future developments and a technology roadmap. A detailed analysis of the results from each case study to identify opportunities, gaps, strengths, and weaknesses will be indeed the main object of the next deliverable.

Bibliography

[1] F. Flammini, L. De Donato, A. Fantechi, and V. Vittorini, “A vision of intelligent train control,” *Preprint submitted to International Conference on Reliability, Safety, and Security of Railway Systems (RSSRail 2022), Lecture Notes in Computer Science, Springer*, 2022.

[2] SMART, “Smart Automation of Rail Transport,” 2016-2019, http://smart.masfak.ni.ac.rs.

[3] SMART2, “Advanced integrated obstacle and track intrusion detection system for smart automation of rail transport,” 2019-2022, https://smart2rail-project.net.

[4] D. Ristić-Durrant, M. Franke, and K. Michels, “A review of vision-based on-board obstacle detection and distance estimation in railways,” *Sensors*, vol. 21, no. 10, p. 3452, 2021.

[5] RAILS, “Deliverable D2.2 – WP2 Report on AI approaches and models,” 2022. [Online]. Available: https://rails-project.eu/downloads/deliverables

[6] L. De Donato, F. Flammini, S. Marrone, C. Mazzariello, R. Nardone, C. Sansone, and V. Vittorini, “A survey on audio-video based defect detection through deep learning in railway maintenance,” *IEEE Access*, pp. 1–1, 2022.

[7] RAILS, “Deliverable D1.2: Summary of existing relevant projects and state-of-the-art of AI applications in railways,” 2021, DOI: 10.13140/RG.2.2.11353.03686. [Online]. Available: https://rails-project.eu/downloads/deliverables

[8] Y. Wang, L. Wang, Y. H. Hu, and J. Qiu, “Railnet: a segmentation network for railroad detection,” *IEEE Access*, vol. 7, pp. 143 772–143 779, 2019.

[9] Z. Wang, X. Wu, G. Yu, and M. Li, “Efficient rail area detection using convolutional neural network,” *IEEE Access*, vol. 6, pp. 77 656–77 664, 2018.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” 2015.

[11] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[12] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[13] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” 2019.

[14] D. Ristić-Durrant, M. A. Haseeb, M. Franke, M. Banić, M. Simonović, and D. Stamenković, “Artificial intelligence for obstacle detection in railways: project smart and beyond,” in *European Dependable Computing Conference*. Springer, 2020, pp. 44–55.

[15] A. Boussik, W. Ben-Messaoud, S. Niar, and A. Taleb-Ahmed, “Railway obstacle detection using unsupervised learning: An exploratory study,” in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 660–667.

[16] T. Ohgushi, K. Horiguchi, and M. Yamanaka, “Road obstacle detection method based

on an autoencoder with semantic segmentation," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[17] K. Lis, K. Nakka, P. Fua, and M. Salzmann, "Detecting the unexpected via image resynthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2152–2161.

[18] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.

[19] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1511–1520.

[20] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," in *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019.

[21] L. Wang, D. Zhang, J. Guo, and Y. Han, "Image anomaly detection using normal data only by latent space resampling," *Applied Sciences*, vol. 10, no. 23, 2020.

[22] D. Ristić-Durrant, M. Franke, K. Michels, V. Nikolić, M. Banić, and M. Simonović, "Deep learning-based obstacle detection and distance estimation using object bounding box," *Facta Universitatis. Series: Automatic Control and Robotics*, vol. 20, no. 2, pp. 075–085, 2021.

[23] SMART, "Deliverable D3.1: Report on algorithms for 2D image processing," 2018. [Online]. Available: https://projects.shift2rail.org/s2r_ip5_n.aspx?p=SMART

[24] Y. Zou, Z. Yu, B. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 289–305.

[25] G. Pastore, F. Cermelli, Y. Xian, M. Mancini, Z. Akata, and B. Caputo, "A closer look at self-training for zero-label semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2693–2702.

[26] F. Pan, I. Shin, F. Rameau, S. Lee, and I. S. Kweon, "Unsupervised intra-domain adaptation for semantic segmentation through self-supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3764–3773.

[27] Y. Zhu, Z. Zhang, C. Wu, Z. Zhang, T. He, H. Zhang, R. Manmatha, M. Li, and A. Smola, "Improving semantic segmentation via self-training," *arXiv preprint arXiv:2004.14960*, 2020.

[28] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[29] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[30] European Railway Agency, "ERTMS - System Requirements Specification - UNISIG

SUBSET-026," 2014. [Online]. Available: https://www.era.europa.eu/content/set-specifications-3-etcs-b3-r2-gsm-r-b1_en

[31] N. Bernini, M. Bertozzi, L. Castangia, M. Patander, and M. Sabbatelli, "Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 873–878.

[32] A. Dairi, F. Harrou, M. Senouci, and Y. Sun, "Unsupervised obstacle detection in driving environments using deep-learning-based stereovision," *Robotics and Autonomous Systems*, vol. 100, pp. 287–301, 2018.

[33] J. Tang, S. Li, and P. Liu, "A review of lane detection methods based on deep learning," *Pattern Recognition*, vol. 111, p. 107623, 2021.

[34] N. Zhao, C. Roberts, S. Hillmansen, Z. Tian, P. Weston, and L. Chen, "An integrated metro operation optimization to minimize energy consumption," *Transportation Research Part C: Emerging Technologies*, vol. 75, pp. 168–182, 2017.

[35] G. Fiengo, D. G. Lui, A. Petrillo, S. Santini, and M. Tufo, "Distributed robust pid control for leader tracking in uncertain connected ground vehicles with v2v communication delay," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 1153–1165, 2019.

[36] "X2Rail-3 Deliverable 6.1 - Virtual Train Coupling System Concept and Application Conditions," 2020. [Online]. Available: https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-3

[37] E. Quaglietta, M. Wang, and R. M. Goverde, "A multi-state train-following model for the analysis of virtual coupling railway operations," *Journal of Rail Transport Planning & Management*, vol. 15, p. 100195, 2020.

[38] E. Quaglietta, P. Spartalis, M. Wang, R. M. Goverde, and P. van Koningsbruggen, "Modelling and analysis of virtual coupling with dynamic safety margin considering risk factors in railway operations," *Journal of Rail Transport Planning & Management*, vol. 22, p. 100313, 2022.

[39] J. Park, B.-H. Lee, and Y. Eun, "Virtual coupling of railway vehicles: Gap reference for merge and separation, robust control, and position measurement," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[40] S. Iwnicki, *Handbook of railway vehicle dynamics*. CRC press, 2006.

[41] J. Felez, Y. Kim, and F. Borrelli, "A model predictive control approach for virtual coupling in railways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2728–2739, 2019.

[42] Y. Yang and F. Yan, "Research on train dynamic coupling strategy based on distributed model predictive control," in *Journal of Physics: Conference Series*, vol. 2183, no. 1. IOP Publishing, 2022, p. 012029.

[43] M. Schenker, R. Parise, and J. Goikoetxea, "Concept and performance analysis of virtual coupling for railway vehicles," in *Proceedings of the 3rd SmartRaCon Scientific Seminar*, vol. 38. Deutsches Zentrum für Luft-und Raumfahrt eV Institut für Verkehrssystemtechnik, 2021, pp. 81–91.

[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[45] A. Candeli, G. De Tommasi, D. G. Lui, A. Mele, S. Santini, and G. Tartaglione, "A deep deterministic policy gradient learning approach to missile autopilot design," *IEEE Access*, 2022.

[46] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[47] T. Kobayashi and W. E. L. Ilboudo, "T-soft update of target network for deep reinforcement learning," *Neural Networks*, vol. 136, pp. 63–71, 2021.

[48] D. T. Gillespie, "Exact numerical simulation of the ornstein-uhlenbeck process and its integral," *Physical review E*, vol. 54, no. 2, p. 2084, 1996.

[49] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.

[50] H. Tang, Q. Wang, and X. Feng, "Robust stochastic control for high-speed trains with nonlinearity, parametric uncertainty, and multiple time-varying delays," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1027–1037, 2017.

[51] H. Tang, X. Ge, Q. Liu, and Q. Wang, "Robust $H\infty$ control of high-speed trains with parameter uncertainties and unpredictable time-varying delays," in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 10 173–10 178.

[52] S. Yi, *Principles of railway location and design*. Academic Press, 2017.

[53] D. Cabecinhas, R. Cunha, and C. Silvestre, "A nonlinear quadrotor trajectory tracking controller with disturbance rejection," *Control Engineering Practice*, vol. 26, pp. 1–10, 2014.

[54] C. Baker, "The simulation of unsteady aerodynamic cross wind forces on trains," *Journal of wind engineering and industrial aerodynamics*, vol. 98, no. 2, pp. 88–99, 2010.

[55] Shift2Rail, "Deliverable D4.1: Reference Scenario," 2016. [Online]. Available: https://projects.shift2rail.org/download.aspx?id=b13c7f54-eac0-4e36-8f3a-e61d6b76c489

[56] J. Aoun, E. Quaglietta, R. M. Goverde, M. Scheidt, M. Blumenfeld, A. Jack, and B. Redfern, "A hybrid delphi-ahp multi-criteria analysis of moving block and virtual coupling railway signalling," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103250, 2021.

[57] S. Su, J. She, K. Li, X. Wang, and Y. Zhou, "A nonlinear safety equilibrium spacing based model predictive control for virtually coupled train set over gradient terrains," *IEEE Transactions on Transportation Electrification*, 2021.

[58] Y. Wu, S. E. Li, J. Cortés, and K. Poolla, "Distributed sliding mode control for nonlinear heterogeneous platoon systems with positive definite topologies," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1272–1283, 2019.