Università degli Studi di Salerno

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA
E MATEMATICA APPLICATA

Corso di Laurea in Ingegneria Informatica

Detecting obstacles in railway environments in real time

Supervisor:
Prof. Alessia Saggese

Candidate:
Camilla Spingola
Mat. 0622701698

Supervisor:
Prof. Antonio Greco

ANNO ACCADEMICO 2022/2023

A tutti i sacrifici
fatti da chiunque e in qualsiasi parte del mondo
per creare innovazioni
che migliorino la vita dell'essere vivente.

Abstract

Description of the problem faced

Railways networks for centuries have played a fundamental role in global transportation, facilitating the efficient movements of goods and passengers over long distances, and they remain the backbone of global commerce, transporting goods that fuel economies and providing efficient transportation options for millions of passengers daily. So, an increase in the security of these systems is necessary and this is where artificial vision systems come into play as a transformative technological solution. In fact, artificial vision systems, powered by cutting-edge computer vision and machine learning algorithms, have the potential to revolutionize railway safety, in terms of supporting human drivers to mitigate the intrinsic complexities of this transportation system, due to the excessive speeds reached even in proximity to urban centers, and replacing traditional methods of track inspection, which often rely on manual labor and periodic inspections, leaving room for human error and potentially catastrophic consequences if obstacles go undetected. To ensure the safety and the reliability of these systems the main challenge lies in the detection of obstacles that may be on the tracks and which can pose substantial threats to the safe operation of the trains, like humans, other vehicles, animals and also rocks. So within this thesis it was decided to focus on the development of an artificial vision system for the detection, among all possible obstacles, of rocks.

Framing of the paper in the contemporary technical scenario

The systems oriented towards the detection of obstacles on the tracks consist of a first phase focused on identifying the Region of Interest (RoI) and a second phase focused on detecting objects and assessing whether they are dangerous ob-

stacles in relation to their proximity to the region of interest. Although approaches with traditional techniques, known a priori information and geometric features have been experimented, the most promising methods have been those based on deep neural networks. For the first task, semantic segmentation networks are employed, and one of the main publicly available datasets designed for this purpose in railway scenarios is RailSem19. For the second task, object detectors are generally used, but not with the same sensitivity for all obstacles, due to the absence of large publicly available datasets for some object categories, like rocks or fallen trees. This represents a significant deficiency since rocks, whether dislodged from natural formations (adverse weather conditions or geological instability) or intentionally placed by external forces, whether small pebbles or larger boulders, when encountered on the tracks, could damage components located in the lower part of the vehicle, or more seriously, they could derail trains, causing catastrophic accidents, service disruptions, and potential derailments. Addressing these dangers is not merely a matter of operational efficiency or of the preservation of invaluable transport infrastructure, it is a matter of life and death.

Personal contribution of the candidate to the solution of the problem described

The objective of this thesis work is to develop an artificial vision system capable of detecting the presence of rocks on railway tracks using a camera mounted on board a vehicle, which is considered a more functional setup for various reasons than a fixed camera arrangement. An operational framework of this kind necessarily requires that the system is resistant to meteorological, lighting and background variations, to noisy acquisitions due to the train's movement speed and that is capable of recognizing distant obstacles. Due to the absence of available data, it has been proceeded with the generation of a synthetic dataset using background images from RailSem19 and FRSign datasets. This approach ensures the creation of two datasets, one for validation-training and one for testing, that are completely uncorrelated. A version of BiSeNet V2 was employed for the segmentation of the region of interest, trained on the RailSem19 dataset, focusing solely on classes related to the structural components of the railway tracks. The segmentation mask obtained is

divided into horizontal slices with heights set to maintain a perspective order. The connected components algorithm is applied to each slice, and a bounding box is extracted for each connected component. Based on the coordinates of these bounding boxes, patches are extracted from the original image, where non-segmented pixels are set to black. These patches were subjected to the developed binary classifier whose architecture was constructed by adding a classification head to the ResNet50, pretrained with ImageNet weights. To enhance dataset variability during training, some data augmentation procedures were also included.

Description of the application/experimental content of the paper

The approach used for the datasets generation contributes to technological innovation by incorporating the use of Generative Fill available in Photoshop, allowing the modification of parts of the image through the use of a GAN. This results in generated images where the source and target domains are well-integrated. Simultaneously the methodological innovation of this work lies in the proposed solution, which replaces traditional object detectors with a simple processing module and a binary classifier. This results in a very streamlined system. Additionally, training the classifier on patches rather than full images has yielded several benefits, such as increased robustness to obstructions and good generalization capabilities to other obstacles. After various comparisons regarding the added classification head, training hyperparameters, and backbones (ResNet50, ResNet101, and MobileNetV2), the best obtained model, and thus the proposed one, achieves an F1-Score of 0.9546 on the validation set, 0.8745 on the generated test set, and 0.7636 on the real test set. The lower performance on the real test set is actually justifiable due to the difficulty in obtaining images of this kind and the consequent diversity they present, even in comparison to the operational framework described at the beginning. The results obtained on generated test set are satisfactory considering the lack of correlation between sets, and in general the realized system, with minimal efforts, paves the way for possible improvements.

Contents

Chapter 1

Introduction

In the realm of modern transportation, the quest for autonomy has been an incessant force propelling innovation. As we stand on the precipice of the fourth industrial revolution, marked by unprecedented technological advancements, the vision of fully autonomous vehicles has transitioned from science fiction to an imminent reality. While the spotlight often shines brightly on autonomous driving in the context of roads and highways, another vital dimension of this revolution silently gathers momentum: autonomous driving on railways.

Most likely, railways networks, more than other driving context, are in need of this revolution. In fact, for centuries, they have played a fundamental role in global transportation, facilitating the efficient movements of goods and passengers over long distances, and they remain the backbone of global commerce, transporting goods that fuel economies and providing efficient transportation options for millions of passengers daily. Similarly more than roads and highways, they would require artificial intelligence systems aimed, at the very least, at supporting human drivers in order to mitigate the intrinsic complexities of this transportation system, due to the excessive speeds reached even in proximity to urban centers, for which a decisive safety enhancement would be necessary.

To ensure the safety and the reliability of these systems the main challenge lies in the detection of obstacles that may be on the tracks and which can pose substantial threats to the safe operation of the trains, like humans, other vehicles, animals and also rocks.

Traditional methods of track inspection often rely on manual labor and periodic inspections, leaving room for human error and potentially catastrophic consequences

if obstacles go undetected. This is where artificial vision systems come into play as a transformative technological solution. Artificial vision systems, powered by cutting-edge computer vision and machine learning algorithms, have the potential to revolutionize railway safety. By providing the capability to autonomously detect and respond to obstacles on the tracks, these systems can significantly reduce the risk of accidents and costly delays.

The future goal of integrating these safety systems onto high-speed trains has led to an increasing interest among railway transport companies in the development of video analysis algorithms capable of detecting obstacles on the tracks. So a form of awareness of this need is beginning to develop, albeit not equally for all obstacles. Indeed, greater attention has been placed on human obstacles or various vehicles for which publicly available datasets are also accessible; the same does not hold true for obstacles like rocks, whose negative effects seem to be underestimated. In reality, however, in the context of railway safety, the perilous presence of rocks and the ever-looming threat of landslides represent formidable adversaries. Rocks, whether dislodged from natural formations or intentionally placed by external forces, whether small pebbles or larger boulders, when encountered on the tracks, could damage components located in the lower part of the vehicle, or more seriously, they could derail trains, causing catastrophic accidents and placing lives of passengers and railway personnel at risk. Moreover, landslides, often triggered by adverse weather conditions or geological instability, can lead to track blockages, service disruptions, and potential derailments, inflicting significant economic and logistical burdens on railway operations. Addressing these dangers is not merely a matter of operational efficiency or of the preservation of invaluable transport infrastructure, it is a matter of life and death.

This is the reason why, within this thesis, it was decided to focus on the development of an artificial vision system for the detection, among all possible obstacles, of rocks, recognizing their danger and identifying them as a less addressed topic in the current state of the art.

In order to achieve this goal, it is obviously necessary to be able to identify relevant images, process them to promptly detect anomalies if they are present in areas of interest, and in this way, prevent railway disasters.

While an approach involving the deployment of fixed cameras along railway tracks is feasible, this thesis work intends to fit within a setup that involves a camera mounted on the front of the vehicle to detect obstacles on the tracks. So, following the identification of areas of interest in the image, relevant features are extracted.

Specifically, the use of fixed cameras along railway tracks certainly has the drawback of requiring an increasing number of installations in proportion to the number of areas to be monitored, resulting in a significant hardware investment as well as the need for synchronization between different video streams. Furthermore, such installations would still need to be deployed in remote areas, far from populated centers or railway stations, where electrification and maintenance could become increasingly complex. Conversely, some of these issues are resolved through the use of cameras mounted on board vehicles. However, this approach collects highly diverse images with complex backgrounds and a substantial amount of noise due to the vehicle's travel speed. Therefore, it exposes us to the need for specific hardware prerequisites for the cameras, as well as the necessity to employ non-traditional video analysis methods. These methods should possess strong generalization capabilities, excellent accuracy, and real-time performance - all of which this experimental work aims to satisfy.

So, in the subsequent chapters, we will delve into the theoretical foundations of computer vision and machine learning, exploring the technologies that underpin our artificial vision system. We will outline a description of state-of-the-art methods that have worked towards the same goal and highlight the need for this study. We will also detail the design, development, and testing of our system, with a particular emphasis on its ability to accurately identify rocks. Moreover, we will assess the real-world applicability of our solution within the context of railway operations, aiming to demonstrate its potential to enhance safety and efficiency.

In essence, this thesis represents a step towards harnessing the power of autonomous driving in the railway sector. By addressing the critical issue of rock detection, we contribute to the broader goal of creating a safer, more reliable, and sustainable railway system for the future.

Chapter 2

State of the art

In the domain of autonomous transportation, ensuring safe and efficient navigation through dynamic environments is paramount. One critical aspect of this challenge lies in the accurate detection and estimation of obstacles within a vehicle's immediate surroundings. While the use of on-board sensors for obstacle detection in road vehicles has seen extensive research and development, the application of similar technologies to railway systems has garnered comparatively less attention. So this section has the aim to present a comprehensive review of the current state of art in vision-based on-board obstacle detection, focusing on railway applications, in order to understand the applicability and the adaptability of vision-based technologies also to railway-specific challenges. This section will focus on systems that use onboard vehicle cameras as, among all the possible on-board sensors, cameras are the ones that most precisely capture accurate data at high resolution. Like human eyes, cameras capture the resolution, minutiae and vividness of a scene with such detail that no other sensors such as radar, ultrasonic and lasers can match. Because of this cameras are in practice essential sensors in multi-sensors OD systems as only vision sensors can provide explicit rich visual information about the scene in front of the train. In addition, the fact that cameras are currently less expensive than other typical sensors for OD solutions has influenced the significant amount of research and development of vision-based on-board obstacle detection systems. Advances in computational platforms and vision algorithms have enabled the use of low-cost cameras to capture images in real time and perceive the surroundings of a rail vehicle, leading to a number of purely vision-based on-board OD systems being developed. We delve into the categorization of methods devoted to the individuation

of the Region of Interest (RoI) and to the detection of obstacles in this section, distinguishing between traditional Computer Vision approaches and those rooted in Artificial Intelligence. The objective of this exploration is that of shed light on existing methodologies, their strengths and their limitations, ultimately paving the way for the development of more robust, adaptive and efficient obstacle detection systems tailored to the unique challenges posed by railways, which is what this thesis work is intended to be.

2.1 Rail Track detection system

The main goal of OD in railways is the detection of objects (possible obstacles) on and near the rail tracks, but to do this it is necessary to include in the system a rail tracks detection for the purpose of identifying the image Region of Interest (ROI), within which is necessary to perform the search for the objects that in this case would be considered hazardous.

2.1.1 Traditional methods

Traditional obstacle detection algorithms are mainly designed to find the rails up until an obstacle. In fact, some of the methods presented in the literature contains only rail tracks detection and do not explicitly consider obstacle detection at all, defining the obstacle free region from the train to the point where the rail track is no longer detected, that could happen also if rails are outside the field of view. In general these methods are based on the assumption that to define a collision free region is sufficient to find a non-interrupted rail track region in front of the train. Some of this type of methods are geometry based [1] [2] and so extends the traditional line detection methods using the geometric characteristics of the specific rail tracks, like the a priori known distance between the rails for each country, or the constancy of the gap between rails at the initialization stage and the fact that in the bird's-eye view the rails remain parallel while in the projectively distorted image the rails tracks intersect in a vanishing point. The bird's-eye view enables to have parallel rail tracks, in a less distorted view, and also a regular spacing of sleepers between parallel rails, so for this reason different methods uses the projective transformation of the camera image into an image where the rails are parallel, and tracks and evaluate the two rail curves using small parallel segments.

Some other methods are based on the extraction of specific rail track characteristics through the individuation of sharp edges corresponding to the point of the image with large gradient. For example, the authors in [3], that is a method among that of this type with very good results, used a method for the features extraction known as Histogram of Oriented Gradients (HOG), in which are computed HOG features, the integral image is generated and the railway tracks are

extracted using a region-growing algorithm.

In the literature, there are also examples of traditional approaches that have employed dynamic programming. For instance, in [4] to extract the train trajectory and the area of rail tracks in front of the train using gradient-based analysis, they first extract a series of information from the image such as the vanishing point and the distance between the tracks, followed by a Hough transformation. Dynamic programming is used to compute the minimum-cost path for extracting the railway track area, both the right and left rails, without the need for any calibration process.

Of course, the approaches previously mentioned have sometimes been combined as well. For instance, in [5] the Hough transformation was applied to the bird's-eye view image obtained from the original image through Inverse Projective Mapping (IPM) and subsequently segmented to detect line segments, this detection is then further refined by imposing geometric constraints derived from prior knowledge. A continuation of this approach is represented by the method [6] in which is introduced a method to compare the extracted edges with candidate parametric rail models obtained through geometric constraints, using a similarity measure.

In the method presented in [7] an additional filtering of the IPM image was proposed using a Difference of Gaussian (DoG) smoothing filter to eliminate any blurring that might result from camera focus errors and motion. The resulting image is then segmented to obtain a binary image, which is subsequently divided into ten equally sized vertical slices to break up longer lines and facilitate the extraction of parallel line segments.

In [8] a frequency domain method is proposed, which becomes sufficiently insensitive to fine-tuning of processing parameters. This method involves dividing each frame into four sub-regions: near, far, remote, and horizon, each of which is then filtered by two-dimensional Gabor wavelets at various scales. These Gabor wavelets are tuned to specific frequencies based on the different sizes and orientations of the railway tracks in each sub-region, in order to highlight the railway edges and filter out noise. The horizon sub-region can be ignored as it contains no information about the railway tracks, while all the others share the characteristic of decreasing rail thickness as directional features increase.

In [9], with respect to the other studies, based on presumed knowledge and assumption about the geometry or railway tracks, is presented a statistical approach, in which are collected many video recordings to quantify and bound variation, introducing a method to predict the railway tracks using a polynomial approximation followed by multi-layer perceptron networks.

It is clear that traditional methods for rail track extraction mentioned above may deteriorate due to different lighting conditions, weather conditions and complex backgrounds. So these methods could have good performance only in a fixed scene but may deteriorate as the scene changes, just as it happens with an on-board camera mounted on a moving train that travels through constantly changing scenarios, and constantly subjected to vibrations that can cause blurring.

2.1.2 AI-Based methods

Due to the undeniable limitations of traditional approaches and in light of the growing advancements in neural networks, there has been increasing interest in the application of artificial intelligence techniques in recent years. The goal is to achieve feature extraction no longer through "hand-crafted" methods but by employing Convolutional Neural Networks (CNN). This interest, primarily rooted in the automotive domain, has also extended to the railway sector. AI-based approaches have effectively transformed the task of rail track detection from one of edge or line detection, as in traditional methods, into an image segmentation problem. Therefore, in this context, the methods that have achieved the best results make use of widely adopted and more efficient segmentation networks.

For example, within the method mentioned in [10] a segmentation network inspired by SegNet [11] is employed. This architecture consists of an encoder for feature extraction from the railway area in the image and a decoder that upscales the feature maps from the encoder to align with the input image resolution and performs classification of the area of interest in the image using a Softmax layer. This method propose an extraction of the railway area and a subsequent refinement, maximizing the outline of the extracted railway area using a polygon fitting method. This architecture achieves satisfactory results in terms of precision in extraction and efficiency.

In the method described in [12] a deep learning algorithm for segmentation is presented, consisting of a feature extraction network and a segmentation network. The former employs a pyramid structure to obtain a feature vector, while the latter is also a convolutional network responsible for generating the rail track segmentation map. The backbone of this architecture, which is ResNet50 [13] was initialized with pre-trained weights from the publicly available ImageNet dataset [14].

Always with the same goal of semantic segmentation for rail track detection in the [15] method, a new encoder-decoder architecture called DFA-UNet has been proposed. This architecture consists of a GAMP module used to extract low-level features and reduce the number of parameters, as well as a DFA module for feature map fusion at different scales, achieving excellent comparative performance.

Another proposed method is a full-resolution residual network, FRNN, [16] that is a benchmark method for fully segmenting Railway scenes from an ego-perspective view. This model is first trained on the Cityscape dataset and then refined on the Railsem19 dataset. This approach achieves a good level of segmentation on the different classes in the rail and streetcar configurations, and in particular on the rail-specific labels, with 71.5% mIoU on elevated railroads, 45% mIoU on embedded railroads, 81.9% mIoU on railroads, and 40.1% mIoU on streetcar tracks.

In [17] a comparison is presented between the UNET [18] and FRNN [16] networks for semantic segmentation. Both methods were trained, validated, and tested on different combinations of data, including images from the RailSem19 [19] and RailSet [17] datasets, a dataset presented by the authors themselves. In both models, parameters such as input size, batch size, optimizer, and learning rate were the same. Likewise, the Jaccard index was used as the metric for both. For each segmentation model, the performance's validation and the test has been done on both datasets. The test set was composed of 1419 images from both datasets in a proportion of 10% each. For the training set, depending on the experience, it varies at proportion of 80% of the involved data. All these sets was selected equitably by taking into account various level of complexity presented on the number of rails, climatic conditions and visibility. To both datasets an augmentation has been applied using as transformation a mirroring with an horizontal shift and a zooming

by warping the perspective of the region that contains all the rails to the full size image. Also were have been added 1290 crops of objects semantically similar to the rails, such as poles and wires, to the empty images where the rails are not visible due to lighting, such as in tunnels. By doing this, the goal is to obtain an improvement in the discrimination ability of the models, thereby reducing false positives. Also to avoid having biased segmentation models due to the presence of a strong intrinsic imbalance in the dataset,for each batch of images and masks have been generated corresponding weights in order to penalize misclassifications. Both architecture have been tested in different scenarios in terms of data used, classes considered and the usage or not of augmentation procedure.

Another semantic segmentation architecture used in railway scenario is a version of Bilateral Segmentation Network (BiSeNetV2) [20] trained on a modified version of the RailSem19 Dataset [19], so with only three classes "rail-raised", "rail-track" and enquotebackground. This architecture involves: a Detail Branch, with wide channels and shallow layers to capture low-level details and generate high-resolution feature representation; a Semantic Branch, with narrow channels and deep layers to obtain high-level semantic context. The Semantic Branch is lightweight due to reducing the channel capacity and a fast-downsampling strategy. Furthermore, it was design a Guided Aggregation Layer to enhance mutual connections and fuse both types of feature representation. Besides, a booster training strategy is designed to improve the segmentation performance without any extra inference cost. Extensive quantitative and qualitative evaluations demonstrate that the proposed architecture performs favourably against a few state-of-the-art real-time semantic segmentation approaches.

It is clear that CNN-based methods are more capable of automatically learning rail features with respect to traditional methods.

2.1.3 Principles of Semantic Segmentation Networks

Semantic Segmentation is the process of dividing an image into multiple segments. In this process, every pixel in the image is associated with an object type, so it helps to understand the content of the image. It has many applications such as image compression, scene understanding or object location. There are two major

types of image segmentation: semantic segmentation and instance segmentation. The difference between these two types of segmentation is that in the first all objects of the same type are marked using one class label while in the second one similar objects get their own separate labels. Over time, many algorithms have been developed for image segmentation but with the advent of deep learning in computer vision, many deep learning models for image segmentation have also emerged. In the following it is reported a comprehensive review of the major of image segmentation approaches using deep learning techniques.

Fully Convolutional networks

A Fully Convolutional Network (FCN) consists of only convolutional layers where features are extracted by convolving a filter of weights. It takes any image of arbitrary size and produces a segmentation map of the same size. It uses skip connections which allows feature maps from final layers to be up-sampled and fused with features maps of earlier layers. This helps the model to produce a very accurate and detailed segmentation by combining the semantic information from the deep and coarse layers with the appearance information from the shallow and fine layers. Few notable applications of FCNs are iris segmentation and brain tumor segmentation.

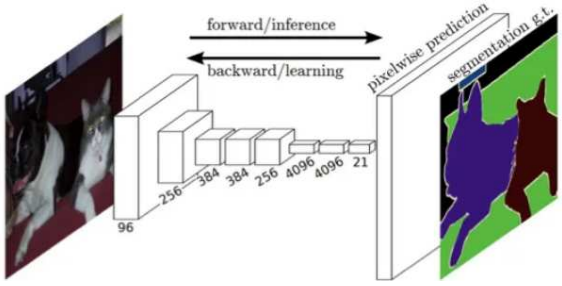

Figure 2.1: A fully convolutional image segmentation network [21]

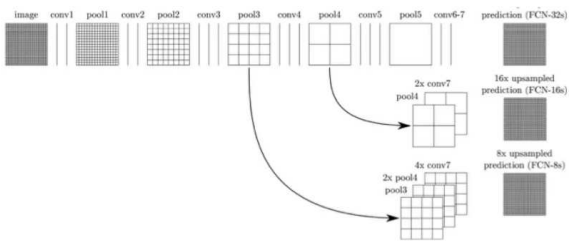

Figure 2.2: Skip connections combining coarse, high-level information with fine, low-level information. [21]

Convolutional Models with Graphical Models

Because of the fact that CNN are characterized by a poor localization property, responses at the final layers of CNNs are insufficiently localized to produce accurate object segmentation. So a solution is the combination of the final CNN layer with a fully connected Conditional Random Field (CRF), in order to achieve higher accuracy.

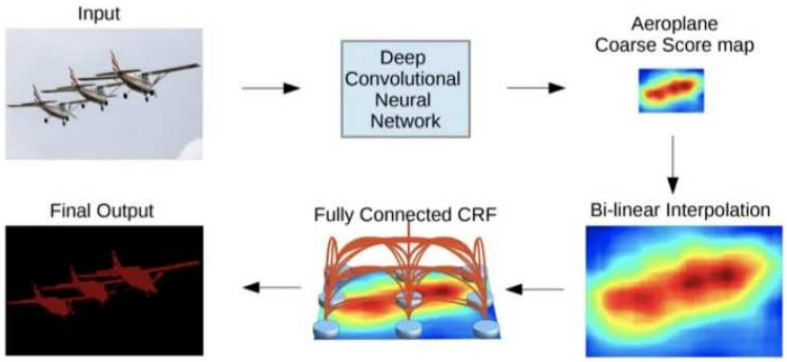

Figure 2.3: CNN+CRF model [22]

Encoder-Decoder Based Models

One of the most popular architecture for image segmentation consists of an encoder and a decoder. The encoder extracts features from the image through filters. The decoder is responsible for generating the final output which is usually a segmentation mask containing the outline of the object, in particular it generates a map of pixel-wise class probabilities based on the input feature vector. These types of models can be grouped into two categories: for general segmentation and for medical and biomedical image segmentation. The most popular methods in the first category are SegNet [11] and HRNet, while the most popular in the second category are U-Net [18] and V-Net[23]. U-Net has an architecture composed of two parts: the left part (the contracting path) and the right part (the expansive path), the purpose of the contracting path is to capture context while the role of the expansive path is to aid in precise localization. The contracting path is made up of two three-by-three convolutions, the convolutions are followed by a rectified linear unit and a two-by-two max-pooling computation for downsampling.

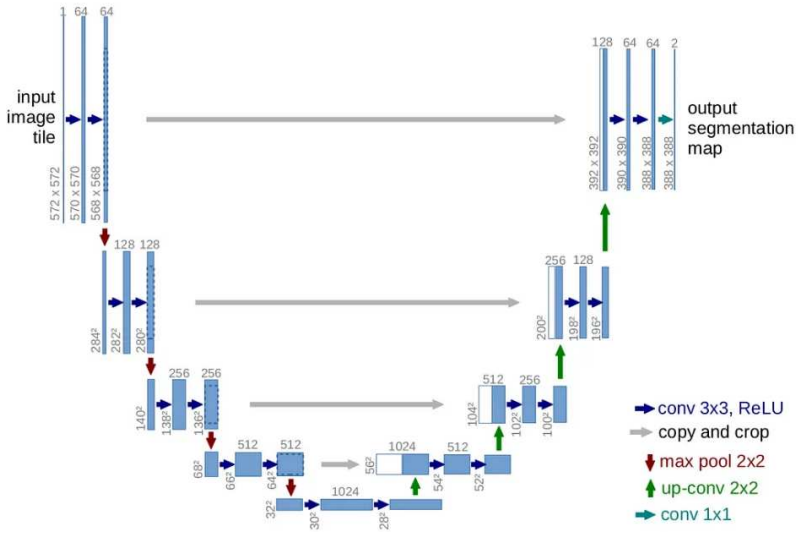

Figure 2.4: UNet architecture image segmentation [18]

In the reported figure each blue box corresponds to a multi-channel feature map. The number of channels is denoted on the top of the box, while the x-y-size is provided at the lower left edge of the box. White boxes represents copied feature maps while arrows denote different operations.

V-Net is instead a network oriented to 3D medical image segmentation and used a new objective function for the training of the model which is based on Dice coefficient.

Multi-Scale and Pyramid Network Based Models

Feature Pyramid Network (FPN) [24] is the most popular model in this category. Initially it was developed for object detection but later on was used for image segmentation as well. It constructs pyramid of features and uses a bottom-up pathway, a top-down pathway and lateral connections to merge low and high resolution features. It then uses a 3×3 convolution on concatenated feature maps to produce the output of each stage. Finally, each stage of the top-down pathway generates a prediction to detect an object. For image segmentation, the authors use two multi-layer perceptrons (MLPs) to generate the masks.

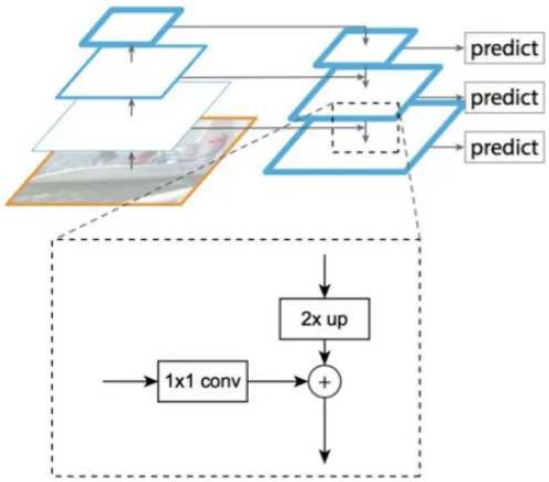

Figure 2.5: A building block illustrating the lateral connection and the top-down pathway, merged by addition [24]

DeepLab family models

In this types of models an additional parameter is added to convolutional layers, known as dilatation rate which defines a spacing between the weights of the kernel. These architetcures are very popular for real-time segmentation. To this family of architectures belong DeepLab v1, DeepLab v2 and DeepLab v3, which represent the state of art for image segmentation in this type of approaches.

The key features of these architectures and in particular of DeepLab v2 are the usage of dilated convolution to address the decreasing resolution in the network (caused by max-pooling and striding), the usage of Atrous Spatial Pyramid Pooling (ASPP), which probes an incoming convolutional feature layer with filters at multiple sampling rates, thus capturing objects as well as image context at multiple scales to robustly segment objects at multiple scales, an improvement in the localization of object boundaries by combining methods from deep CNNs and probabilistic graphical models.

Recurrent Neural Network Based Models

RNNs have also proved useful in image segmentation, in fact they potentially improve the estimation of the segmentation map by modeling the short and long term dependencies among pixels. ReSeg was the first RNN-based model used for image segmentation. It was developed from ReNet which was used for image

classification. ReSeg model uses ReNet layers which are stacked on top of the pre-trained VGG-16 convolutional layers that extract generic local features to perform image segmentation. In order to recover the original image resolution in the final predictions ReNet layers are then followed by up-sampling layers. It uses Gated Recurrent Units (GRUs) as they provide a good balance between memory usage and computational power.

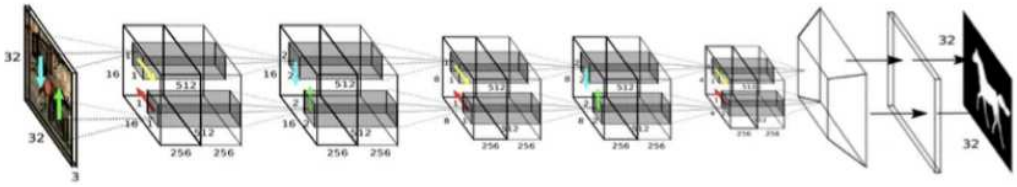

Figure 2.6: ReSeg Model. [25]

Later on, another semantic segmentation model emerged based on Graph LSTM (Graph Long Short-Term Memory)network which was a generalization of LSTM from sequential or multidimensional data to general graph-structured data. In this they take each arbitrary-shaped superpixel as a semantically consistent node, and adaptively construct an undirected graph for the image, where the spatial relations of the superpixels are naturally used as edges.

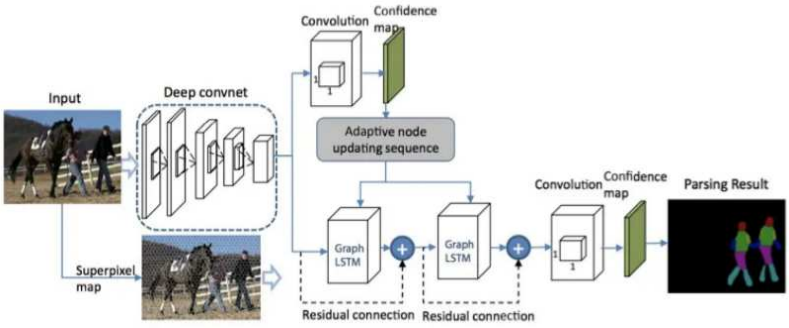

Figure 2.7: The Graph LSTM model for semantic segmentation. [26]

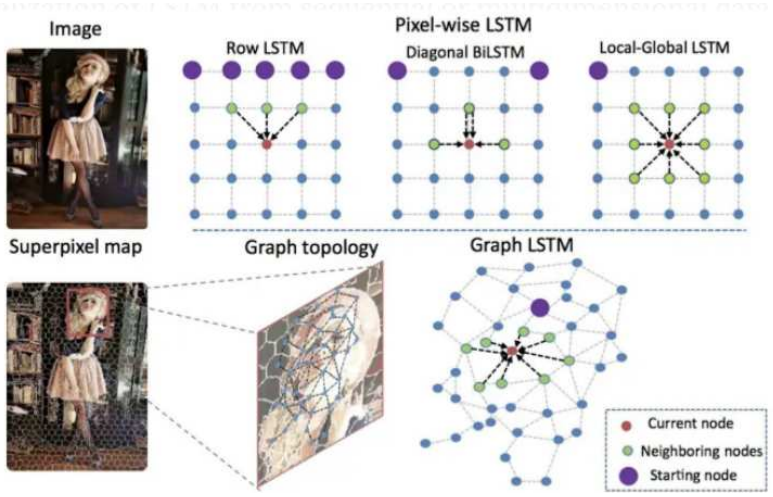

Figure 2.8: Comparison between the graph-LSTM model and traditional pixel-wise RNN models. [26]

Attention-Based Models

The attention mechanism outperforms average and max pooling, and it enables the model to assess the importance of features at different positions and scales. Unlike CNN models, where convolutional classifiers are trained to learn the representative semantic features of labeled objects, the Reverse Attention Network (RAN) architecture trains the model to capture the features that are not associated with a target class. The RAN is a three-branch network that performs the direct, and reverse-attention learning processes simultaneously.

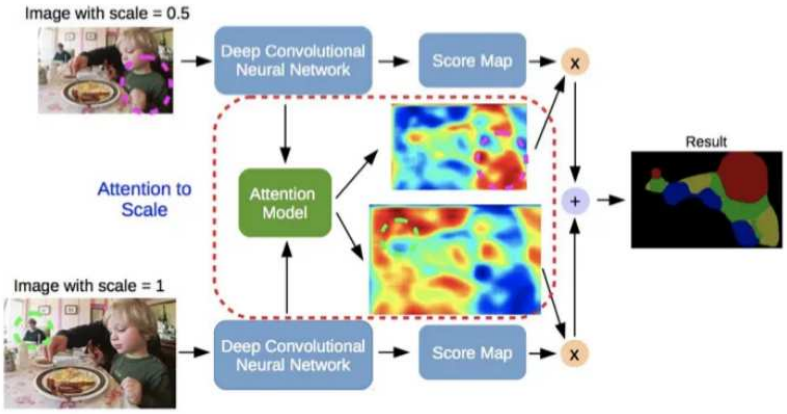

Figure 2.9: Attention-based semantic segmentation model. The attention model learns to assign different weights to objects of different scales; e.g., the model assigns large weights on the small person (green dashed circle) for features from scale 1.0, and large weights on the large child (magenta dashed circle) for features from scale 0.5. [27]

CNN Models With Active Contour Models

The FCNs along with Active Contour Models (ACMs) have recently gained interest and it is an ongoing research. One of its approach involves formulating new loss functions inspired by various ACM principles where as other approach utilizes ACM merely as a post-processor of the output of an FCN and several efforts attempted modest co-learning by pre-training the FCN. One of its example is ACM post-processor for the task of semantic segmentation of natural images where level-set ACMs are implemented as RNNs.

2.2 Dataset for semantic segmentation in the railway scenario

It is undeniable the attention being given to autonomous driving in general, but while public datasets for road scene segmentation have been increasing over the years, in the railway domain, data is still very scarce and needs to be collected, annotated, and made open-source. In particular, due to the lack of these data of railway anomalies, a large majority of anomaly detection literature rely on ultrasonically collected data by attaching sensors to the train. Although introducing these sensors to trains does not require major changes to existing systems and gives very good results, their accuracy is greatly affected by weather and distance. It should also be pointed out that these special sensor devices such as stereo cameras and LIDAR are extremely expensive and their data processing is power consuming, so the need for image datasets is evident. Within this section, state-of-the-art image datasets designed to fulfill the semantic segmentation task for roads will be reviewed and described.

2.2.1 BH-rail

This dataset has been used for the training and the testing of the method proposed in [10]. This dataset was annotated according to CityScape dataset standards, so was used a pixel level labeling for the rail region and the non-rail region. It was recorded on the Beijing Metro Yanfang line and the Shanghai metro line 6 with an on-board RGB camera, and contains images with resolution of 1280*720. The BH-rail dataset has 5617 annotated images(4494, and 1123 images, respectively) and covers representative real scenarios including tunnel and elevated environments. Besides, to verify the robustness of our method in different light environments, the dataset contains images at different times, images in the daytime scene and images in the night scene.

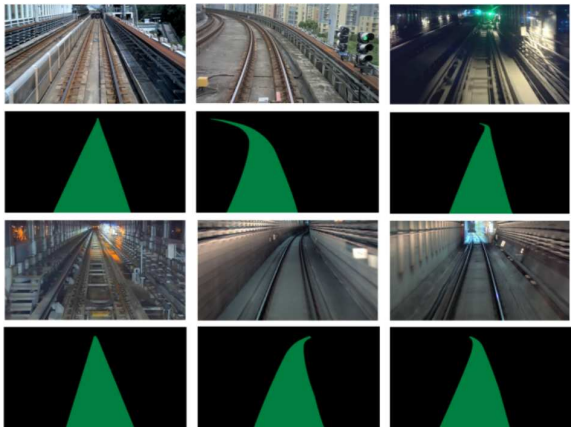

Figure 2.10: Examples of images and annotation of BH-Rail Dataset.

2.2.2 Railroad Segmentation Dataset (RSDS)

The Railroad Segmentation Dataset (RSDS) was created to train and to test the RailNet network [12]. This dataset includes 3000 images, 2500 are used for training, 200 for validation and 300 for testing. All of the images came from a real-world train's driving environment. In the ground truth labelling of the railway datasets, the railway was defined as all pixels between two rail tracks, all the collected images have a size of 1920x1080. Also, due to the small size of the dataset, during training was applied a data augmentation algorithm, even if no details of this were given.

Figure 2.11: Examples of images and annotation of RSDS Dataset.

2.2.3 RailSet

The RailSet Dataset is introduced in [28] as a new innovative dataset relevant fot railway anomaly detection, so it is an extension of the Railsem19 dataset to more complex scenes, richer annotations in terms of semantic segmentation of

the tracks, higher diversity in weather and a specialization for high-speed lines environment. Other variations are the diversity of rail visibility images, long distance tracks, complex situations such as curves, and finally noisy images due to flow movement. This adds more situations and complexity to Railsem19. It consists of 6600 high-quality manually annotated images containing normal situations and 1100 images of railway defects such as hole anomaly and rails discontinuity, which pose serious threats to railway safety. Due to the lack of anomaly samples in public images and difficulties to create anomalies in the railway environment,the images of abnormal scenes have been generated artificially, using a deep learning algorithm named StyleMapGAN [29], so using a GAN for the image generation. This type of architectures are based on the idea of estimating the distribution of input data and then generating similar one. This is accomplished by training both a generator that learns how to generate realistic-looking images and a discriminator that distinguishes between fake and real images. The mentioned dataset is created starting from 23 selected videos of the train driver's perspective, which cover over 34 hours of diverse train traffic from different countries and under varying setups: weather conditions, camera models, mounting positions and lightning conditions. To avoid redundancy and overlapping scenes, per video are chosen an average of 280 frames, with each frame relatively separate from the previous one. It has also been fitted a SSIM metric to measure the structure difference between the selected frames and filter out repetitive scenes. Have been also included empty scenes where the rails are invisible, such as in tunnels with no lighting, and snowy scenes where the rails are completely covered, in order to evaluate the consistency of existing models over more variable situations. RailSem19's labeling policy, respect to that of this dataset, contains more rail-specific labels, and in fact in this dataset only two classes have been annotated: rail and rail-track.

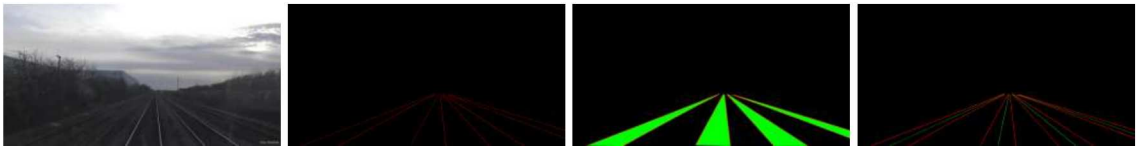

Figure 2.12: RailSet-Seg different annotations masks

Then the work was mainly oriented to the generation of a dataset containing

two types of anomalies: hole under the tracks and rail discontinuities. Considering the fact that there were no real anomaly images to train the GAN network with, it was implemented a copy-paste algorithm that takes as input a railway image as well as its rails segmentation results, then selects randomly a region of interest in between a track on which it pastes the anomaly and finally returns a composite image that was then used to train the GAN network. In particular to use the copy-paste procedure 100 RGB hole images were collected, and they were pasted to railway images from RailSet-Seg dataset and from Railsem19, the 15100 obtained images were used to train the two GAN networks, each of one oriented to a specific anomaly, in order to generate images as realistically as possible.

Figure 2.13: RailSet-Ano data processing, copy paste procedure

Figure 2.14: Random generation results - hole anomalies

Figure 2.15: Image reconstruction results - rail discontinuity anomalies

2.2.4 RailSem19

Railsem19 [19] is the first public dataset for semantic scene understanding for trains and trams. This dataset consists of 8500 short sequences annotated from the ego perspective of trains, including over 1000 examples with railway crossings

and 1200 tram scenes. As the first image dataset targeting the railway sector, a new labeling policy has been designed from scratch and in fact it focuses on rail-specific labels not covered by any other datasets. The annotations made consist of manual annotations in the form of geometric shapes and also dense semantic labeling at the pixel level, which arises from the combination of geometric shapes and annotations generated by existing specialized semantic segmentation networks for the road domain. The base sequences used within this dataset are sequences collected from the perspective of the vehicle, sourced from a large online community of train enthusiasts who upload videos from the driver's cabin. The selected sequences amount to approximately 350 hours of railway and tram traffic from 38 different countries, collected across all four seasons and various weather conditions. The dataset's variety is further expanded by the fact that the images were collected using a wide range of camera models, mounting positions, and lighting conditions.

Figure 2.16: Examples from RailSem19: front views from train and tram rides in varying seasons, lightings, and environments.

In the image selection process, the aim was to include only frames that avoided redundancy. Below, it is possible to see the labels considered within the dataset and their corresponding quantities.

label	**buffer-stop**	**crossing**	**guard-rail**	**train-car**	**platform**	**rail**	**switch-ind.**	**switch-left**	**switch-right**	**switch-unknown**	**switch-static**
count	218	2885	4873	1951	2482	58322	2007	1965	2083	2491	2519
label	**track-sign-front**	**track-signal-back**	**track-signal-front**		*person-group*	*car*	*fence*	*person*	*pole*	*rail-occluder*	*truck*
count	7404	3448	5768		62	172	291	234	10875	3920	11

Figure 2.17: Number of annotations per class for Railsem19; cursive labels are rail occluders.

Since the release of this dataset, several experiments have been conducted related to typical computer vision scene understanding tasks: image classification and dense image segmentation. The initial results obtained from these experiments clearly demonstrate the benefits of using the new dataset for tasks related to the railway domain.

2.2.5 Others

There are also some datasets that are road-oriented such as Cityscapes [30], Mapillary vistas [31], COCO-stuff [32] and KITTI [33], which may contain some railway scenes, that are a combination of interior views, road views and pedestrian views, but driver's perspective scenes are almost non-existent in them and also it is clear that none of these datasets provide sufficient quantity and diversity of data to train deep learning models for the task we are talking about.

2.3 Methods for detecting obstacles on railway tracks

Once the region of interest is identified, in order to diagnose potential risk cases in the railway sector, it is necessary to detect any obstacles present in the region of interest, whether on the tracks or in adjacent areas. The majority of scientific works aimed at this objective typically address the resolution of only one of the two tasks. In this section, we will proceed with a description of methodologies focused on the detection of objects in railway scenarios, both using traditional methods and those based on deep learning techniques.

2.3.1 Traditional methods

Similarly to traditional methods aimed at detecting the region of interest, for this task as well, an approach is followed for the extraction of hand-crafted features in order to identify potential objects, inspecting only the ROI. For example, in [34], once the railway tracks are extracted, various methods are applied for object detection. Specifically, for moving objects, an optical flow is applied, while for stationary objects, the Sobel edge detection method is used, followed by a morphological process on the image with the detected edges.

In particular, in [35], following the identification of railway tracks through the use of the Hough transform, obstacle detection is carried out using the Canny algorithm. This involves filtering out excessively small or disconnected detected edges, and subsequently, a systematic search method is implemented using the tracks as a guide. An extension of this work is represented by [36], where the systematic search is conducted from bottom to top. This approach proves to be effective in the case of stationary objects in front of or near the train. In contrast, within this work, for the detection of dynamic objects, the concept of optical flow between frames is employed. This involves tracking the trajectory of objects of interest to assess whether they might collide with the train in the future. Approaches with windows from top to bottom, after identifying the region of the tracks, have also been used. However, the authors of these works have highlighted the limitations of the proposed

methods as they have not been tested on real images, which are seldom available, instead, the tests were conducted on digitally modified images with the addition of obstacles.

Another type of possible approach is background subtraction, as used, for example, in work [37]. This approach involves using certain images as a baseline, comparing them with the current images acquired by the on-board vehicle camera. A frame-by-frame correspondence is then calculated between the current image and the previous ones captured at the most similar point to find the frame to use as a reference. Assuming that there are no obstacles in the frame chosen as a reference, a "difference image" is created by subtracting the current frame from the corresponding reference frame. This should have values significantly different from zero only in pixels containing obstacles. A very similar approach is presented in method [38], with the only difference being that the images used as references are frames captured by a camera on board another train that has traveled the same road before.

Another possible approach is represented by [39], where computer vision and Principal Component Analysis (PCA) are employed. Specifically, images of the static context are acquired, and from these, the transformation matrix used by PCA is then calculated. By means of this transformation matrix, the consecutive camera images are projected into the transformation space and recovered later using inverse transformation. Motion detection is accomplished by calculation of the Euclidean distances between the original and recovered images. Image regions whose Euclidean distances are larger than a pre-defined threshold value are considered to belong to detected dynamic objects.

Most of the works reported up to this point are explanatory of the possible traditional approaches explored. However, there are indeed works of this kind that have been characterized not only by obstacle detection but also by the identification of the class, at least pedestrians and vehicles.

2.3.2 AI-Based methods

Published AI-based methods for obstacle detection in railways have mainly used DL-based networks for object detection already proven in other applications.

As presented in the following, for the purpose of object detection in images of railway scenes, these established DL-based networks are used in their original form, as trained with large publicly available object detection datasets, or they are re-trained with custom made railway datasets.

Some of the methods used in the literature involve the use of the Faster Region based Convolutional Neural Network (Faster R-CNN, [40]). For example, [41], this architecture is employed using a dataset of 20,000 images of railway scenarios available on the internet. In these images, three classes were identified: trains, people, and animals. The same architecture was also used in method [42], which is a hybrid method where the extraction of the region of interest was performed using a traditional approach, specifically through the Canny edge detector and the Hough transform. For obstacle detection, after converting the acquired thermal images to the HSV format to remove noise, a Faster R-CNN was employed. The architecture used consists of two modules: a first deep convolutional network to obtain region proposals and a second module that serves as the detector for object detection.

Continuing, another method that employed a modified version of the same architecture is [43], using a dataset that covers more scenes and adds diverse obstacle information, for a total of obstalce's type equal to 9. Also the ability of the method is tested based on various scene conditions such as different regions and lighting, proving a better balance between speed and accuracy compared with other models including also YOLOv4.

Another type of architecture used is represented by the Feature Fusion Refine Neural Network (FR-Net), based on a CNN and composed of three connected modules: the "depth-wise-pointwise" convolution module, which improves real-time detection; the "coarse detection module" which approximates the locations and sizes of prior anchors to provide better initialization for the subsequent module and also reduces the search space for classification; and the "object detection module" that provides more accurate object locations and predicts the class labels for the prior anchors. The model developed was evaluated on the Railway Object Dataset, developed by the authors of the work [44], which includes seven different types of objects typically present in railway environments: bullet train, railway straight, railway left, railway right, pedestrian, helmet, and spanner.

An improvement to the FR-Net method, and that used the same dataset, is proposed in [45] and is called Differential Feature Fusion CNN (DFF-Net). DFF-Net is an end-to-end object detection network with a fully convolutional network architecture. DFF-Net includes two modules: the prior object detection module, which produces initial anchor boxes; and the object-detection module, which applies a differential feature fusion sub-module onto the initial anchor boxes to enrich the semantic information for object detection, enhancing the detection performance, particularly for small objects.

The same dataset was also used in [46], for a method named Feature-Enhanced Single-Shot Detector (FE-SSD) which exploits the capabilities of a prior module detection mechanism named RON–Reverse connection with Objectness prior Networks [47], and Faster Better Network (FB-Net [48]); it does this by fusing the adjacent feature maps to learn deeper semantic information and by using a feature transfer block. The evaluation results showed that the proposed method provided a good balance between accuracy and real-time performance for object detection and significantly improved the small-object detection performance with respect to several other considered methods derived from same original DL-method.

A multi-stage obstacle detection method is also proposed in [49]. The method has two steps: feature map creation and feature fusion. In the first step, the input image is converted into multi-scale feature maps using a Residual Neural Network (RNN). Multi-scale features maps improve the identification of objects of different sizes at different distances. Specifically, low-level features lack semantic information but provide precise object location information, while high-level features are rich in semantic information but provide only approximate location information. Once the multi-scale features maps are created, a series of convolution layers are added to extract features, and the network calculates a confidence score and bounding boxes for possible obstacles. For training and testing the network, the authors made a custom dataset which contained large-scale urban rail transit video frames taken by an on-board HD camera. The dataset was gathered from three metro railways in China with camera image resolution of 1280×720 pixels, and has 6384 images with 5107 images for training and 1277 for testing. The dataset covers different lighting and weather conditions and its images have three classes of manually annotated

objects: train, luggage, and humans.

Certainly, among the utilized approaches, it's important to include those employing networks belonging to the YOLO family, whose main advantage is speed, making them particularly suitable for real-time applications such as onboard obstacle detection on railways. Among the methods that have employed YOLO, there are methods [50] [51] which, however, retrained the original YOLO model on the custom long-range railway dataset developed as part of the SMART project [51]. In total, 998 images captured with SMART RGB cameras were used, with 2238 labeled objects of the classes human, car, bicycle, and large animal. These images were recorded by a multicamera system mounted on a Serbia Cargo locomotive during operational runs in various lighting conditions along a 120 km section of the pan-European 'Corridor X' route.

Another proposed method that used a network of YOLO family is [52] in which is presented a framework focused on the detection of external objects, proposing a two-phase structured detection framework. The first part of the work is carried out by a "Lightweight Railway Image Classification Network" which classifies each input image as normal or containing intrusions. This is achieved through a series of enhancements, including an attention mechanism and an inverted residual unit. The second part of the work, applied only to images deemed to contain an intruder, is oriented towards detecting the class and position of objects. Specifically, this method is implemented using YOLOv3. Within this work, it is evident that the performance of the proposed classifier in the first phase is nearly comparable to networks such as ResNet-50 or MobileNet (F1 score of 96.85) but with a higher FPS value than both models. The YOLOv3 model, applied to 487 test images, achieves a mean Average Precision (mAP) of 85.89% with an FPS value of 32.6.

An additional approach that is completely different from the previous ones is [53], it proposed a three-stage obstacle detection framework, composed by a coarse region proposal (CRP) module and a lightweight railway obstacle detection network (RODNet) and a postprocessing stage. The objective of the CRP module is to produce a small set of objectness regions. Then, the two-scale parallel processing method supplemented by the selection and merging strategy is developed to efficiently and effectively generate candidate regions. In this way, some background

areas can be filtered out. In the RODNet module, a lightweight detector is constructed to reduce the parameters and to accelerate the inference. RODNet simultaneously processes the original railway image and candidate cropped regions to output predicted bounding boxes information of different obstacles. Finally, at the postprocessing stage, a processing pipeline integrated with fusion, removal, and suppression method is developed to deal with redundant boxes on the boundary of each subregion before the nonmaximum suppression (NMS) is applied. To train and validate the efficiency of model, high-resolution images in various natural railway scenes have been captured, at different times of the day and simulating the crossing and staying behavior of a single or multiple people and also cars on the tracks, and covering sunny weather and nightfall. These images were obtained by sampling every 5-10s the videos and removing similar images. Due to the fact that the images were not enough and to prevent overftting, the training of the model was done also using existing public datasets, like MSCOCO and VOC2007&2012. The performance obtained by this architecture (mAP equal to 80.3%) are comparable to that of YOLOv4.

The majority of the methods we analyzed were characterized by having an onboard vehicle-mounted camera. However, one method that involves the use of a fixed camera is [54], in which the installation is positioned at a certain height to have a wider field of view and avoid collisions. Within this method, objects are continuously monitored, assigned an ID, and if the time they remain in the vicinity exceeds 30 seconds, an alarm is generated at both neighboring stations. These stations should communicate to the train the need to brake. The proposed method employs a Single Shot Detector (SSD), which is much less expensive than You Only Look Once (YOLO) models. The proposed method was trained on the MSCOCO dataset.

Some other approaches like that proposed in [55] involve the combination of a deep network with information from other sensors. Specifically, within this work, a voting mechanism is used that combines radar data with images from a camera. These images undergo analysis by a very simple convolutional neural network whose operating weights were obtained through training on an out-of-context dataset, namely ImageNet [14].

2.4 Real Datasets for obstacle detection in the railway scenario

Methods for obstacle detection based on artificial intelligence have unquestionably shown better results compared to traditional ones. However, they come with the requirement for a large amount of data for training to create a system with good generalization capabilities. Unfortunately, in the railway domain, obstacle datasets on tracks are very limited, and many are not even publicly available. In this section, we will proceed to describe some of the main datasets of obstacle images on tracks, in addition to those already mentioned. Specifically, many of the aforementioned datasets, initially designed for semantic segmentation purposes, are also oriented towards the detection of various types of obstacles.

2.4.1 Dataset used in Railway obstacle detection algorithm using neural network

The dataset used in the works [41] and [42] and is composed by 20000 pictures of outdoor tracks obtained from Internet. The objects in these pictures were divided into three categories: train, people and animal. The entire dataset was divided in a training set composed by 15000 images and a test set composed by 5000 images.

Figure 2.18: Some examples of the images contained in the Dataset.

2.4.2 Railway Object Dataset

In works [45], [44] and [46] was used a dataset created by the authors of [44]. The equipment to collect images was placed in front of a train in order to capture real-world railway traffic videos. All the collected samples have been annotate accordingly to human visual habits. In particular the annotations made include Bullet Train, Pedestrian, Railway Straight, Railway Left, Railway Right, Helmet, and Spanner. To ensure diversity in the data, all the videos have bee acquired under different lighting and weather conditions, subsequently the sampling was done every five images.

Figure 2.19: Examples of images and annotations contained in the Dataset.

At the end they were collected 7342 sample images with a resolution of 640 × 512 pixels. 70% of these images were shuffled for training and validation, and the remainder of the images were used for testing.

Category	Bullet Train	Pedestrian	Railway Straight	Railway Right	Railway Left	Helmet	Spanner
Occurrences (number)	3671	9371	3863	652	1804	3089	761
Mean size (pixels)	248 × 248	171 × 171	535 × 535	259 × 259	237 × 237	34 × 34	35 × 35
Mean area ratio	3.75%	2.23%	18.9%	4.38%	3.61%	0.31%	0.32%

Figure 2.20: Occurrences, mean size and mean area ratio of each category in railway traffic dataset.

2.4.3 Dataset used in Real-time Obstacle Detection Over Rails Using Deep Convolutional Neural Network

In order to verify the accuracy and real-time performance of the algorithm proposed in [49], it was made a dataset which contains a large-scale urban rail transit video frames taken by on-board HD camera. The dataset was gathered from Hongkong MTR Tsuen Wan Line, Beijing Metro Yanfang line and Shanghai metro line 6 with resolution of 1280*720 pixels. A total of 6384 images constituted the dateset and are divided into two groups: 5107 images for training and 1277. To enhance the robustness of the proposed method in different illumination and weather, this dataset contains images in daytime and night, sunny and rainy days. The coordinates of the obstacles appeared in the images were manually annotated.

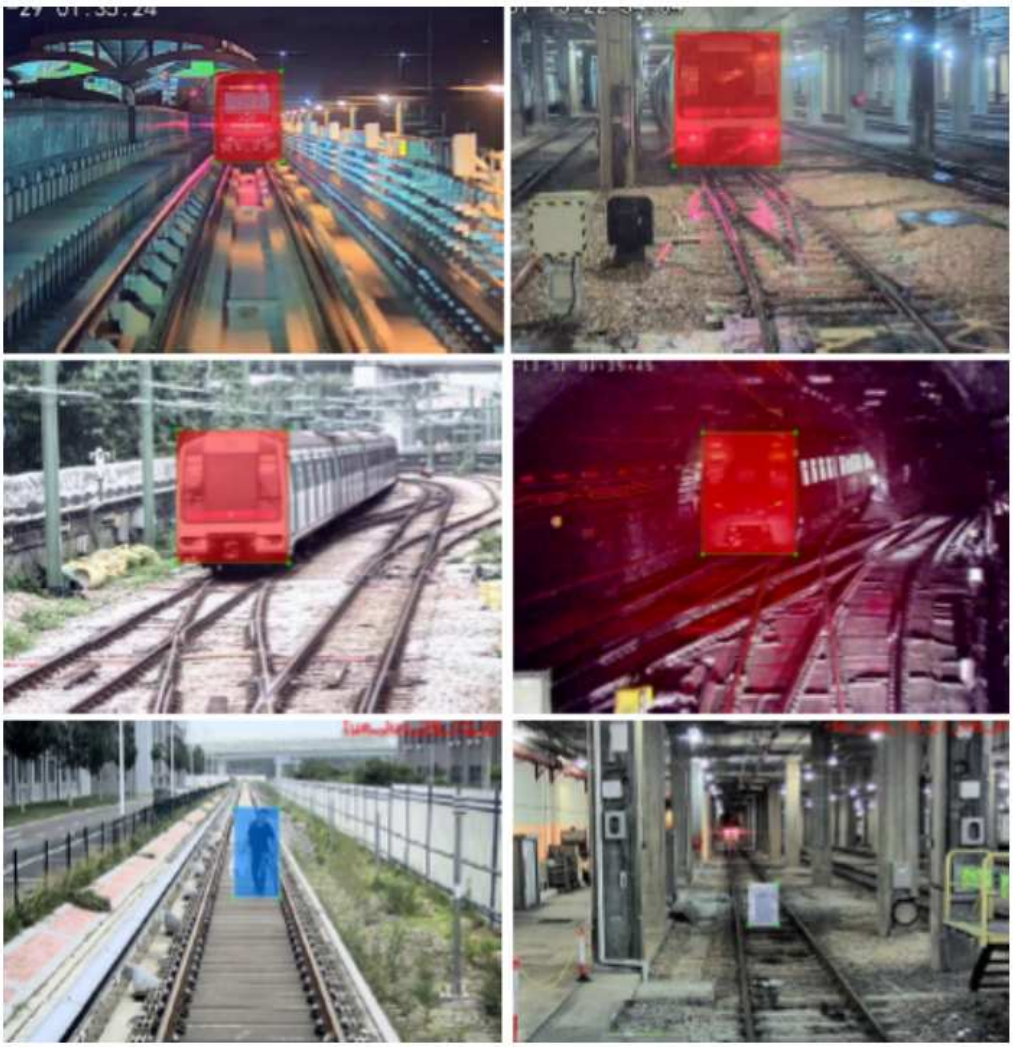

Figure 2.21: Examples of images and annotations in the Dataset.

2.4.4 Dataset developed in Artificial intelligence for obstacle detection in railways: Project smart and beyond

In the context of scientific work [51] it has been realized a dataset composed of the images extracted from about 8 h. These images were recorded by a multi-camera system mounted on a Serbia Cargo locomotive, during operational runs in different illumination conditions along a 120 km long Serbian section of the pan-European 'Corridor X' route. In these images are present both static and moving obstacles including humans, vehicles, bicycles and animals, for a total of 998 images with 2238 labelled objects. Each object in the dataset is described by a number of parameteres including the information about the class of the object, the 2D bounding box and some others that refers also to the position of the object in terms of distance from the camera, because of the fact that this dataset was also used for distance estimation task.

Figure 2.22: Examples from SMART railway dataset. Different object classes on the/near the rail tracks (humans, different vehicles, animals).

2.4.5 Dataset Used in Foreign Object Detection in Railway Images Based on an Efficient Two-Stage Convolutional Neural Network

In order to evaluated the performance of the proposed method in [52] the authors themselves have built a foreign objects dataset including 3145 railway images. It consists of 1523 normal railway images and 1622 intruded railway images. The size of the images is 720x1280 pixels. The normal railway images were taken from multiple views of a railway scenery that is located at the campus of Guangzhou Railway Polytechnic, Guangzhou, China. To generate the intruded railway images, we placed several foreign objects on the railway and took photographs from multiple views. There are three types of foreign objects in 1622 intruded railway images, including 594 bottles, 712 tires, and 316 umbrellas.

Figure 2.23: Examples of Dataset images with and without intruding object.

These objects were selected to simulate possible foreign objects of different sizes intruding on the railway, with manually annotated bounding boxes. Among 3145 images, we selected 1885 images to train the image classification network. The remaining 1260 images were used to evaluate their performance in the experiments. In particular, the training set consists of 912 normal images and 973 intruded images, and the test set consists of 611 normal images and 649 intruded images. Then, 1622 intruded images have been used to evaluate the second-stage foreign object detection. The remaining 1523 normal images were not used in this stage because there is no object to detect in these images. In particular, 1021, 114, and 487 intruded images were split into training, validation, and testing images, respectively. There are 176 bottles, 204 tires, and 107 umbrellas in 487 test intruded images.

2.5 Generated Datasets for obstacle detection in the railway scenario

In the following section, it will be proposed some methods addressing the problem of obstacle detection on railway tracks which used a dataset generated from the author themselves for the training and/or evaluation. Regarding generated and synthetic datasets, it is clear that they are certainly exposed to a very high risk due to the fact that the target domain and the source domain (the domain of the object added synthetically) are different. This could result in the object detection network's performance not reaching a sufficiently high reliability. It is clear, however, that in situations where the alternative would be the total absence of such systems due to the non-existence of real images, or the compromise of systems trained on the same objects but in different contexts, synthetically generated images can be considered a valid alternative, especially because experiments on high-speed railways in operation are not allowed.

2.5.1 Dataset proposed in Obstacle Detection over Rails Using Hough Transform

Due to the limited availability of real world video footage in which obstacles appear in front of trains, the method presented in [35] artificially added obstacles to route recordings published on Internet. Before identifying potential obstacles, the system focuses on detecting the tracks through the Hough transform. The system was tested using these modified videos, and a particular modified video of the route between Landskrona and Ramlösa in Sweden was used for evaluating the system performance. This video was recorded from the driver perspective with favorable weather conditions, at day, without turnouts and with splendid visibility. Using 7 minutes of recording have been added a total of 20 digital obstacles of different nature, shape and obstruction trajectory. The system successfully identified 19 of the 20 obstacles, but still has a high occurrence of false positives. Most false detections were due to the presence of objects near the road but actually do not represent real risk for the train travel, these includes signs, pass levels, bridges over

the rail, some platforms, etc. The real time performance of the system reached rates around 60 fps using image resolutions of 640x480 pixels. The metrics used for evaluating the proposed method refer to the false positive rate, false negative rate, true positives, and true negatives. So the performances obtained, considering a number of FP=12 and a number of TP=19, are a precision of 0.61 and a sensibilidad of 0.95. Although the results obtained may be considered quite good, they still leave room for various challenges related to the robustness of the method under different weather conditions and its ability to recognize objects typical of railway scenarios, such as lights or signals, as benign objects rather than obstacles. The addition of obstacles has been made using the video editor for GNU/Linux Kdenlive. All the system implementation was accomplished using the C++ libraries of OpenCv.

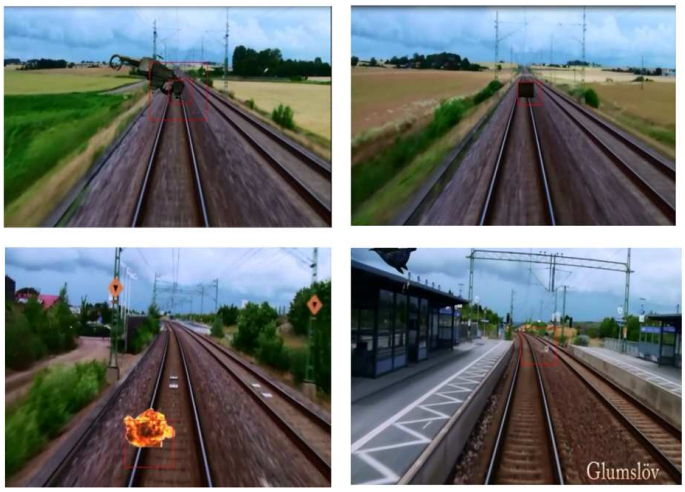

Figure 2.24: Examples of generated images and corresponding annotations made by the system. The first three are correct while the last one is a false positive.

2.5.2 Dataset proposed in High-Speed Railway Intruding Object Image Generating with Generative Adversarial Networks

In this work it has been proposed a a novel method to generate railway intruding object images based on an improved conditional deep convolutional generative adversarial network (C-DCGAN, an improved conditional DCGAN that is a deep convolutional GAN). It consists of a generator and multi-scale discriminators. Loss function is also improved so as to generate samples with a

high quality and authenticity. The generator is extracted in order to generate foreign object images from input semantic labels. So the generated objects are synthesize in the railway scene but, to make these objects more similar to real ones, in this work is proposed an algorithm for scale estimation based on the gauge constant, applicable at various positions in a railway scene. The experimental results on the railway intruding object dataset show that the proposed C-DCGAN model outperforms several state-of-the-art methods and achieves a higher quality (the pixel-wise accuracy, mean intersection-over-union (mIoU), and mean average precision (mAP) are 80.46%, 0.65, and 0.69, respectively) and diversity (the Fréchet-Inception Distance (FID) score is 26.87) of generated samples. The mIoU of the real-generated pedestrian pairs reaches 0.85, and indicates a higher scale of accuracy for the generated intruding objects in the railway scene. The conditional deep convolutional GAN (C-DCGAN) model is first trained with image pairs of semantic labels and real images. For application, the trained generator is used to translate the semantic labels to various real images. The semantic image is also used to segment the object's contours in the railway scene. After the object scale size is calculated at the position, the generated intruding object is synthesized to the railway scene.

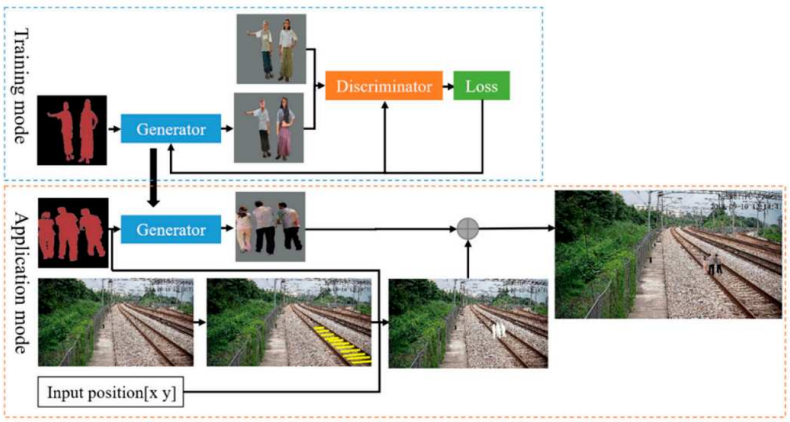

Figure 2.25: overview of the proposed algorithm.

Potential intruding objects on railways mainly include pedestrians and large livestock (sheep, horses and cows). So it was first built a dataset of railway intruding object images derived from the public datasets LIP [56] and MS-COCO. Then some preprocessing procedures have been made, such as resizing, background

segmentation, and reassigning values to the labels, obtaining 11,615 semantic and real-images pairs, whose 80% was used for training set and the remaining 20% for validation sets. The railway scene dataset was constructed based on surveillance videos along the high-speed rail lines, and so the images contain different scenes, under different weather conditions.

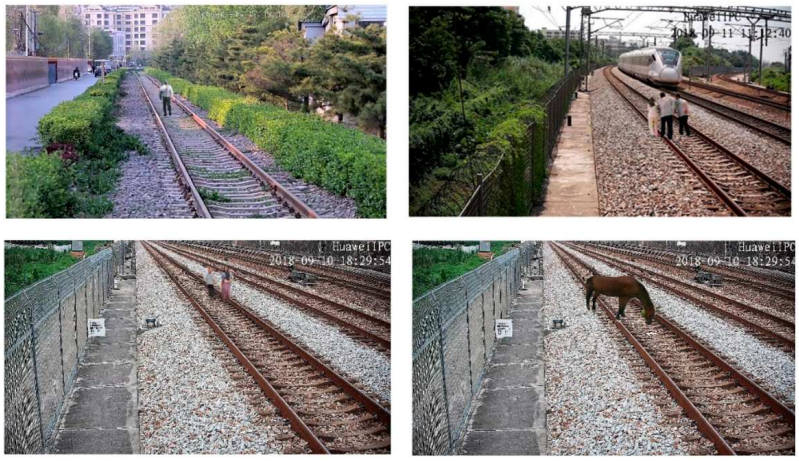

Figure 2.26: Samples of generated railway objects intruding on the image's dataset.

In order to evaluate the authenticity of the generated images under a global coarse scale and local fine scale, the datasets, generated and real, were input into YoloV3 with different sizes, and a small performance gap between the two datasets has been demonstrated in the case of people.

2.5.3 Dataset proposed in Automatic Detection of Obstacles in Railway Tracks Using Monocular Camera

In this work [57] attention was focused on identifying obstacles on the tracks and obstacles located in the surrounding areas of the track itself, still representing an obstruction. Due to the absence of a specialized database and specific images typical of these critical situations, video editing software was used to simulate this type of imagery, adding them to existing real images. In particular to test the method proposed in this work a set of 60 images was used on which the obtained performances consists of an accuracy of 75%, a specificity of 60% and a sensitivity of 90%.

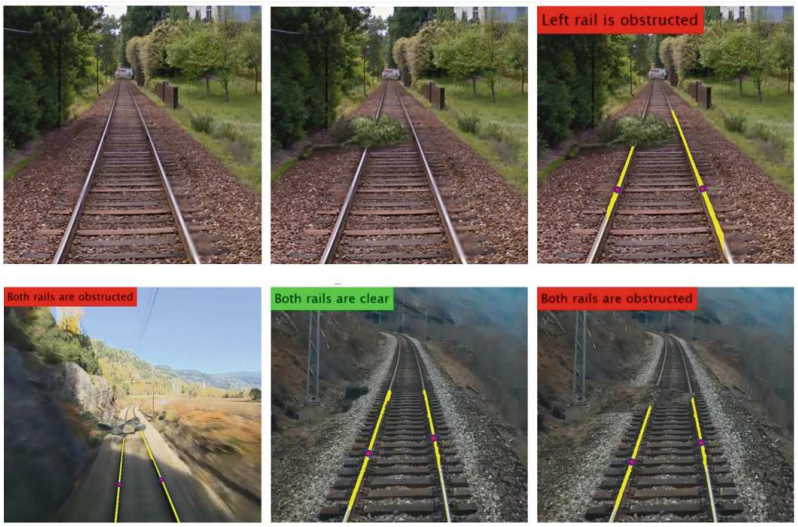

Figure 2.27: Some examples of the generated images and corresponding system's output.

2.5.4 Dataset proposed in Automatic Detection of Railway Track Obstacles using Monocular Camera

One of the main contributions of the thesis [57] is the creation and the collection of a novel dataset of railway track images with and without obstacles. The images having no obstacles have been assembled from two sources, the first source is a compilation of several images obtained fro Google TrainView, the second source comes from a collection of YouTube videos where the frames were collected. The images containing obstacles, instead, were artificially generated, because there was no large dataset of railway tracks images with obstacles. For the generation of images two approaches have been followed, the first one used the algorithm already developed in another work, in which the obstacles instances were automatically generated by placing the obstacle images (a .png of trees and rocks) in the original image, in the second method, instead, the obstacles instances were manually generated using image manipulation software (GIMP), by outlining specific contours in the target image, that could represent obstacles in a real scenario. The obstacle instances were generated by placing the contours in such a manner to simulate rock slides or landslides. In this way it was obtained a dataset containing a total of 2319 images, 1572 of them without obstacle and 747 with obstacle.

In order to have two classification classes with an equal number of images, the

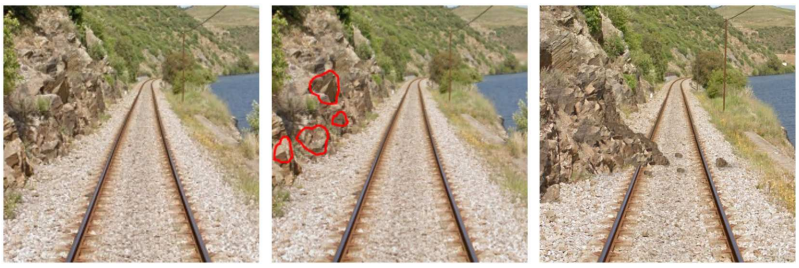

Figure 2.28: Process of obtaining an obstacle instance with GIMP.

Obstacle class images are subject to data augmentation techniques to generate more cases in the reference method. The data augmentation procedure included a vertical flip, some modifications in the contrast and another vertical flip. In the described method From the total 1572 No Obstacle images, a proportion of 70%-15%-15% is selected for the training,validation and test sets, corresponding to an amount of 1100, 236 and 236 No Obstacle images, respectively. From the 747 distinct Obstacle images, 236 are selected for the test set. To avoid having images in the validation set generated by data augmentation techniques from training examples, from the remaining 511, 236 are randomly selected as a validation set. The remaining 275 are subject to data augmentation techniques, accounting for a total of 1100 Obstacle images to be used as training set.

2.6 Datasets in railway scenarios for other purposes

From the analysis we are conducting, it is clear that among the main objectives for which the scientific community is approaching the railway scenario is the realization of a system that can be used to support, if not even replace, the human driver, thus achieving the long-desired autonomous driving. Naturally, for this goal, the considerations made about the need for a system capable of detecting tracks and any obstacles on them certainly remain valid. Equally important is having a system capable of identifying other characteristic components of the railway scenario essential for correct guidance, such as signals. Furthermore, with the perspective of creating synthetic datasets, leveraging such images can ensure the integration of the generated object into a real background. Therefore, in the following part of this section, it is reported an example of these datasets, but it is necessary to specify that, as it is possible to see in [58] there are different types of dataset of images in railway scenarios for different purposes.

2.6.1 FRSign

FRSign [59] is a large-scale and accurate dataset for vision-based railway traffic light detection and recognition. The recordings were made on selected running trains in France and benefited from carefully hand-labeled annotations. An illustrative dataset which corresponds to ten percent of the acquired data is published in open source with the paper. It contains more than 100000 images illustrating six types of French railway traffic lights and their possible color combinations, together with the relevant information regarding their acquisition such as date, time, sensor parameters, and bounding boxes. Unlike the automotive sector, railway lights specifications are not unified, and each country designs and uses its own signals. This dataset was acquired in France, in fact its name stands for

French Railway Signalling and contains multitudes of samples of the French railway traffic lights. The train used for our work is equipped with multiple sensors that detect and capture the surrounding environment. For instance, it is equipped

Figure 2.29: Examples of images and corresponding annotations of FRSign dataset.

with a LiDAR fixed on top of the train's engine. Most importantly, it is equipped with two cameras, located in the driver's cabin, that capture the railway in front of the train. Therefore, these cameras film the railway from the train driver's point of view. Annotations are implemented in the form of objects called "bounding boxes", and also in this work is defined the terminology and the specifications used for French railway infrastructure such as panel types and their combinations. So in conclusion, FRSign is composed of 393 video sequences,which make up 105352 individual images.

It is important to notice that the acquisition of such images was not trivial, because it faced multiple challenges on several levels like having the necessary authorizations, configuring the hardware and understanding the panels. Some possible ways to complete and to improve this dataset, individuated from the authors themselves, are the acquisition of images in multiple weather conditions, the achievement of a balanced distribution of different panels, and an acquisition of images that covers the entire country including other settings and environment.

2.7 Methods for detecting rocks on railway tracks

The method proposed in [57], is composed of different steps, like a preprocessing of the image, the rail segmentation and then the obstacle detection. In this sequence of operations the image histogram is equalized to enhance the contrast of the image in order to have a better performance in edge detection, then is applied a Gaussian filter in order to attenuating the response of high frequency and undesired features produced by the edge detection algorithm. Finally is extracted a normalized region of interest, discarding unnecessary sections. Once the edges are detected, it becomes necessary to select which of these belong to the tracks. To achieve this, the Hough transform is employed to identify straight lines that correspond to the edges of the tracks, and specifying a number of desired Hough Peaks equal to two. Certainly, an approach of this kind works very well for mild curvatures, however, it proves inadequate for pronounced curves. That's why, in this work, has been developed and employed the Kano's method, based on mathematical morphology operations and BLOB analysis. This method requires the removal of smaller components. Subsequently, starting from individual edges, through a closing operation with a small structural element, objects are obtained. In a binary image, objects that do not meet certain orientation prerequisites with the rails are discarded. Then, a closing operation with a larger structural element is performed to reconstruct the geometry of the rails. Having done this, to detect the possible presence of obstructions, the number of blobs on the sides is evaluated. If this number is different from 1, it indicates the presence of an obstruction, taking into account additional factors. In the absence of obstacles on the tracks, the analysis proceeds to the surrounding regions.

Instead in the thesis [60] have been compared a computer vision and a deep learning approaches for the automatic detection of obstacles in railway tracks, dimostrating many advantages of deep learning method respect to approaches based on the detection of straight lines [57]. The approach based on computer vision consists of a preprocessing phase, composed of the histogram equalization and the application of a Gaussian filter, and the extraction of the ROI. Subsequently there is a phase of segmentation in which are tested Sobel operator and Canny edge detector.

The last phase is that of Obstacle Detection, which has the aim of identifying both direct and indirect obstacles, using mainly geometric considerations. In the context of a deep learning-based approach, within the proposed work, many combinations of hyperparameters were tested by modifying the number of trainable parameters of GoogLeNet. The performance observed between the two methods highlights a significantly superior capability of the deep learning-based method.

		Ground truth	
		Positive	Negative
Pred.	Positive	225	106
	Negative	11	130

		Ground truth	
		Positive	Negative
Pred.	Positive	229	13
	Negative	7	223

Figure 2.30: Performances of traditional and deep learning methods respectively.

Naturally, both methods are still affected by issues such as shadows causing interruptions in the tracks, albeit to varying degrees. Certainly, the benefit of employing a deep learning method could be further realized through the use of a larger dataset, which remains one of the challenges identified at the conclusion of the described work.

The method proposed in [61] is oriented to the detection of rocks, animals and vehicles and is a multi-block SSD method integrating object detection and classification based on deep learning, having as base netwirk a VGG16. SSD has better detection performance than other state-ofthe-art algorithms. SSD has two advantages, namely, realtime processing and high accuracy. However, few problems are observed in SSD application to railway scenes, for example, small objects are difficult to detect, whereas model training requires numerous labeled data. To improve the detection accuracy of small objects, the original image is sliced into several overlapped sub-images. Then, these sub-images are fed into the SSD network. The segmentation of sub-images can enhance the local contextual information and prevent the objects from being truncated. This study aims to improve object detection accuracy when only limited samples are available. A two-stage training strategy leveraging to transfer learning is adopted to solve this issue. The deep learning model is preliminarily trained using auxiliary labeled data and then refined using a few samples of railway scene. To train the architecture, considering the difficulty to obtain a large number of images in railway scenario, due to security reasons, it was first used a dataset composed of 5000 images collected from normal

scenarios in order to do a pre-train of the SSD-network and then a second dataset of 2000 images collected from railway scenarios is used to fine-tune the network. Finally the test dataset was a collection of 1500 samples of the railway scene.

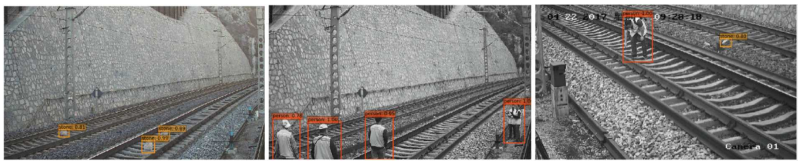

Figure 2.31: Detection results of multi-block SSD method.

On this test set, a performance analysis was conducted, as reported below, demonstrating an improvement of this architecture compared to the conventional SSD, especially in the detection of stones.

Class	Precision (%)	Recall (%)	F1 (%)
Person	98.3	98.3	98.3
Stones	98.9	92.1	95.4
Train	99.0	93.2	96.0

Figure 2.32: Multi-block SSD detection results.

2.8 Research Gaps and Future Research Directions

Through the analysis of the methods currently available in the state of the art, it is evident that the early works on obstacle detection in railways have followed methodologies previously developed in the field of obstacle detection in the automotive sector. Clearly, an approach of this kind can only have limited effectiveness, as methods aimed at operating on the railway need to address specific challenges of the railway environment. These challenges include the fixed path of the train on the tracks, whose appearance changes significantly in different lighting and weather conditions, as well as the large braking distance that a train requires at high speeds, necessitating long-range obstacle detection.

It is clear that even in this inheritance approach from the automotive world, approaches based on traditional techniques have more visible issues compared to those based on artificial intelligence. The latter, by employing features extracted from convolutional layers following network training, enjoy significantly higher generalization capabilities. This makes knowledge transfer between different domains still considered effective. However, it is essential not to underestimate the fact that, although these methods are based on artificial intelligence, they have not been explicitly designed to meet specific railway requirements, such as the accurate and timely detection of objects on or near the tracks.

Another noteworthy problem arises from the difficulty in evaluating the applicability of already published approaches to the railway domain due to the absence of specific datasets for railways. Currently available datasets for the railway domain are, as evident from the analysis just conducted, mostly unavailable and incapable of covering relevant use cases. They lack standardized scenarios, making it unthinkable to compare various methods. It is clear that with the growing interest in the railway scenario, real datasets will become more available. However, these datasets will likely cover common railway objects such as cars and pedestrians. Meanwhile, datasets depicting rarer objects like rocks or fallen trees, which still pose significant threats to the safe operation of trains, will likely remain scarce, hence, currently, the detection of rocks or fallen trees is problematic, if not impossible,

precisely due to the underrepresentation of this scenario in the training dataset.

Chapter 3
Original contribution to problem's solution

This chapter aims to illustrate, starting from the analysis of the problem to be addressed, from the limitations that will be identified, and from the analysis of the state of the art of the systems proposed for this purpose, a solution proposal that is innovative in both implementation and results achieved, compared to the methods that alreasy exist.

3.1 Description of the problem

This thesis work, as mentioned earlier, represents a single piece in a much broader framework for which the scientific community has been showing sensitivity in recent years, namely autonomous driving in the railway scenario. It is important to emphasize that a pronounced sensitivity towards the railway scenario began to manifest itself with a visible delay compared to what happened in the automotive world, and this highlights the underlying motivation of this thesis, which aims to address some identified gaps. The revolution of autonomous driving in the railway domain is indeed necessary for any type of vehicle traveling on the tracks, from those transporting goods to those carrying passengers, not overlooking inspection vehicles. Traditional methods of track inspection often rely on manual labor and periodic inspections, leaving room for human error and potentially catastrophic consequences if obstacles go undetected. This is where artificial vision systems come into play as a transformative technological solution. The whole becomes, of

course, even more challenging when considering the future widespread adoption of high-speed rail.

Certainly, the introduction of such a system is oriented towards reducing the risk of accidents. To achieve this, however, it becomes necessary to have an excellent obstacle detection system on the tracks, which could include people, other vehicles, animals, rocks, and fallen trees. The relatively recent sensitivity towards this issue has certainly resulted in what is now the main problem for the development of such systems, namely the lack of data. Specifically, for some types of obstacles, there are some datasets, mostly not freely available, that do not cover a wide range of scenarios. For other categories of obstacles, such as logs or boulders, there are no publicly available datasets at all. The level of progress achieved in these areas relies on methods proposed by authors who have created datasets, mostly very small ones, not publicly released, making any attempt to compare different solutions impossible.

The lack of interest in types of obstructions such as rocks is completely inexplicable, considering the fact that, regardless of the dimension of the rock and regardless of whether the rocks were dislodged from natural formations, like adverse weather conditions or geological instability, or intentionally placed by external forces, when encountered on the tracks, could damage components located in the lower part of the vehicle, or more seriously, they could derail trains, causing catastrophic accidents and placing lives of passengers and railway personnel at risk. Addressing these dangers is not merely a matter of operational efficiency or of the preservation of invaluable transport infrastructure, it is a matter of life and death.

Therefore, having identified a gap concerning this type of obstacle, the system proposed within this thesis develops precisely in that direction. Specifically, the aim of this work is to develop a computer vision system for the detection of boulders. This system will be capable of capturing images from the surrounding environment, analyzing them using the methodology defined below to identify the presence or absence of boulders on the tracks, and potentially triggering an alert to prevent railway disasters.

The envisioned scenario for the proposed solution is that of a camera mounted on board a vehicle. It is clear, therefore, that the proposed system must be resistant to the continuous changes in the environment that a moving train may

encounter, as well as the blurring of acquired images due to the high speed of the train. Additionally, it must benefit from real-time performance, crucial as the train approaches the obstacle. Among the prerequisites for such a system, there must be an ability to detect obstacles even at considerable distances due to the extended braking distance required by the vehicle. Furthermore, the system should exhibit a high capacity for generalization to scenarios with significant variations in terms of lighting and weather conditions.

For these crucial requirements of generalization, it is believed that it is necessary to create a system entirely designed for the specific application domain, rather than inheriting one from other scenarios and attempting knowledge transfer. Since publicly available datasets for this purpose are absent, in order to train, validate, and test the architecture to be developed, the described work also includes the creation of a dataset depicting boulders on tracks. This dataset should encompass a good variability in terms of boulder types and sizes, lighting conditions, meteorological factors, and railway background landscapes.

3.2 Data Generation

Within this section, we will proceed to illustrate both the prerequisites that would have been placed on the data if, before the development of such a system, it had been possible to set up a data acquisition campaign. We will also describe the chosen images on which the boulder generative process was applied. In the previous chapter, in the analysis of the state of the art of datasets available for this purpose, an indisputable insufficiency became evident. This insufficiency, along with the impossibility of field acquisition, drove the creation of a synthetic dataset. In this section, we will describe the procedure used for the generation of this synthetic dataset and the results obtained, emphasizing the technological innovation developed in this aspect.

3.2.1 Definition of Required Data

If it had been possible to set up an image acquisition campaign to meet this specific task, it would undoubtedly have been easier to satisfy highly important prerequisites on images. The obstacle detection system, as repeatedly emphasized, regardless of specific implementations, should certainly require a preliminary phase of identifying the region of interest, which in this specific case is represented by the tracks. Subsequently, there should be a phase that involves inspecting the identified region to diagnose the possible presence of obstacles. If the identification of tracks were to be done through neural networks, it would be necessary to acquire a sufficiently extensive set of images, including various railway structures that are diverse enough in terms of the complexity of the railway system (intersections, turns, curves), and in terms of the track structure. Especially if we consider that many architectural aspects of the railway line are characteristic of each country.

Just as it would be necessary to take into account these peculiar diversities of the railway system, the obstacles under consideration also present enormous variability that needs to be addressed. Specifically, the type of rock, including its coloration and texture, is undoubtedly a factor of variability, as well as its shape and size. Additionally, the position of the obstacle could lead to different types of shading and to different level of integration of the obstacle into the surrounding environment.

In particular, as mentioned earlier, a boulder on the track could be deliberately placed, making it easily distinguishable from the surrounding environment (low integration level), or it could be the result of natural events such as landslides, making it difficult to distinguish from the surrounding environment (high integration level).

In addition to the aspects just mentioned that are specific to the tracks or to the obstacles, there are a series of other aspects that are common to both tasks, both the identification of the region of interest and the identification of the obstacle, for example no less negligible should be the variability to be achieved in the images in terms of the landscapes surrounding the tracks. These variations could cause confusion in the system in question. For example, for the individuation of railways there is a difference between the presence of natural landscapes or structures such as roads or sidewalks, that due to their geometric characteristics, might be more prone to confusion with the tracks themselves. Similarly, there might be objects in either scenario that could be confused with boulders. The same diversification should have been obtained in terms of weather conditions and lighting conditions, considering that even a simple weather condition like sunshine can bring about different challenges throughout the day, with changes in the position of the sun concerning the camera's viewpoint.

Considering that the hardware architecture within which our system operates, as will be detailed further, includes a camera mounted on board a vehicle, a factor that could significantly influence the performance of the system we are developing is the blurring caused by the vehicle's movement speed and the resulting camera oscillations that become integral to it.

So it will be desirable for the acquired dataset to be representative of these conditions, as they pose challenges to the system's performance. But, it is clear from the conducted analysis that managing to cover such a variety would be practically impossible. In fact, even with all the necessary equipment, to cover all this conditions, it would undoubtedly be necessary an enormous amount of time, a substantial financial investment and a risk dictated by the need to have a train driving towards an obstacle to capture realistic acquisitions. It becomes intuitive, therefore, to justify the absence of such an extensive dataset and also the lack of

a real dataset of boulders, which would require transporting them to the point of interest. These reasons leave room to consider generated datasets as plausible solutions.

In general, when possible, there is an attempt to avoid synthetic datasets, as it is clear that they are certainly exposed to a very high risk due to the fact that the target domain and the source domain (the domain of the object added synthetically) are different, and this could result in the realization of a system without a sufficiently high reliability. However, it is clear that if the alternatives are to use data from other domains or not have any data at all, this option is a valid alternative. So in the proposed method it was decided to use a generated dataset representing the rocks on the tracks, while for the images oriented to track detection, there are real datasets available in the state of the art, as reported in the previous chapter, so it has been decided to draw from them.

3.2.2 Image Generation Process

Having chosen the path of a synthetic dataset, a generation procedure was set up to be as effective as possible. As reported in the previous chapter, existing synthetic datasets were identified through a review of the state of the art. The procedures through which these datasets were generated vary, but there is a clear indication of the superiority of those generated using GANs compared to those generated through more traditional copy-paste methods. The latter, despite attempting to maintain the aspect ratio, lack details such as shadows, inevitably compromising their naturalness. Of course, none of such datasets were available, so an analysis was conducted to determine the preferred system for generation.

Therefore, the decision was made to utilize the generative component present in Photoshop. The latter will be detailed further, but briefly, it consists of a GAN integrated into Photoshop, providing users with extensive possibilities. Specifically, the user, completely unaware of the underlying mechanism, can choose a background image, identify a region with a variable shape, and describe through a simple phrase what they would like to be generated in that section of the image. As a result, in most cases, the desired object is well-integrated into the surrounding space in terms of coloration, shading, dimensions, and positioning, such as adherence to the

ground.

It is evident that despite the performance of this powerful tool, some generations may fail in terms of positioning or in terms of naturalness of the depicted subject. For this reason, estimating the number of images to be generated for the realization of the system in question it was also necessary take into account the number of generated images that would then be discarded. Therefore, considering this number and the machine time required by Photoshop for each generation, it arose the need to designate a procedure that could be automated as much as possible, in order to avoid requiring constant human intervention.

The first step done was to select which background images to use for the generation. For this purpose, it was decided to use two different datasets, namely RailSem19 [19] and FRSign [59]. The choice fell on these two for several reasons. Firstly, both datasets are publicly available, have a sufficiently high number of images, and consist of images captured from the front of the vehicle, adhering to the specifications of our interest. Both datasets have acquisitions made in sufficiently diverse weather and lighting conditions, as well as very high variability in terms of track structure (curvatures, overlaps, numerosity). Additionally, FRSign [59] is entirely collected in France, while RailSem19 [19] is composed of a series of frames extracted from videos recorded by train enthusiasts, covering geographical variability, the importance of which has been discussed earlier. Furthermore, the RailSem19 [19] dataset was created for semantic segmentation of tracks and obstacles on them, so it also has segmentation ground truth. The decision to use two datasets stems from the idea of creating, as will be better detailed later, two completely unrelated sets for training-validation and testing phases. This caution was specifically motivated by the fact that being an artificially generated dataset, it was necessary, using all the possibilities at our disposal, to make an accurate evaluation of the performance of the system obtained. Therefore, in this way, given the extremely diversified background scenarios between the two datasets, the generalization capability of the system is tested quite well.

Through the use of the tool provided by Photoshop, it became apparent that there is an enormous variability available in terms of generated boulders, including quantity, color, texture, and size. Therefore, these two achieved requirements aligned

well with the objectives identified during the data requirements identification phase. Another component of variability definitely came from the position of the obstacle on the track, and this was managed with the automatic generation procedure described below.

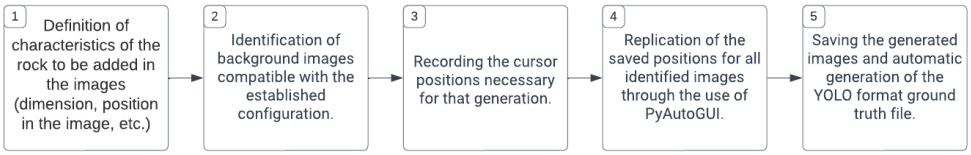

Figure 3.1: Required steps for the procedure of automatic generation of boulders that has been designed

The above scheme highlights the steps that were necessary for the generation in the most automatic way possible. Specifically, Photoshop did not provide APIs that could be simply called, so it was decided to use PyAutoGUI, which essentially allows defining specific positions on the screen and executing predefined mouse commands at those positions without human intervention. The procedure outlined above,was carried out independently for each of the two background image datasets used.

In particular, there is a first phase in which a specific configuration is decided regarding the size of the boulder to add and its position, whether in the center, on the sides, in the foreground, or in the background. Given this specific configuration, a selection was made among the available background images, aimed at identifying those that could be compatible with the particular configuration chosen. Each of the background images, acquired in a varied manner, presented its own peculiarities. After that, by running a script that recorded the mouse positions and actions performed to complete the generation of a boulder in the specified configuration on one of the selected background images, the actions were saved to a text file. Finally, through PyAutoGUI, the steps read from the text file were re-executed for each of the other selected images for that configuration, without requiring any further human intervention. So, in a cyclical manner, all of this was executed for each identified configuration and for each of the two datasets. Each time, the generated images were saved, and a final review and selection of all of them were carried out. Knowing the positions where the boulder was inserted during this procedure, it was also possible

to automatically generate the ground truth file containing the coordinates of the reference bounding box in YOLO format. For each generative cycle, the Photoshop tool generated three different images, all characterized by having the obstacle in the same position and the same background, so the only variation was represented by the specific rock generated. During the just-described procedure, another component of variability was introduced by making a random choice of the phrase indicating the object to generate from a set of possible phrases, such as "a rock", "a fallen rock", "many rocks", etc.

Figure 3.2: Some generated samples on images of FRSign.

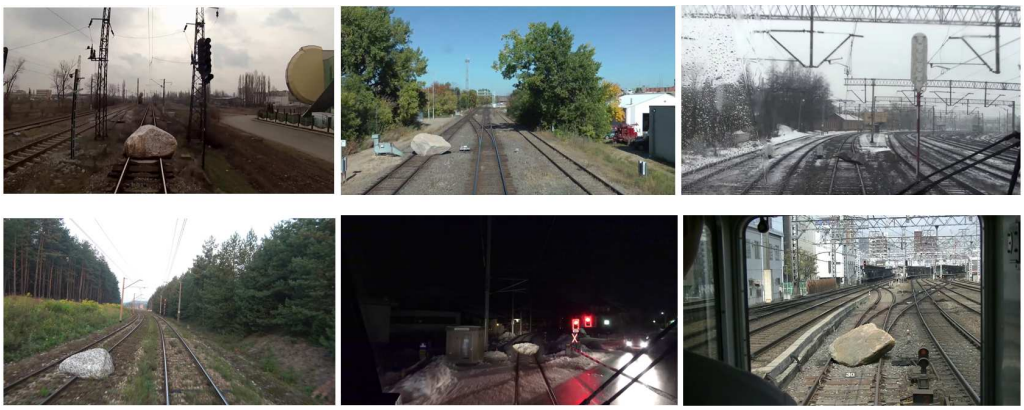

Figure 3.3: Some generated samples on images of RailSem19.

Starting from the displayed images, the achieved level of variability is visible in terms of track configuration, type, position, and size of the generated boulder, as well as the meteorological and lighting conditions of the surrounding space.

It is also important to emphasize that during the generation process, attention has been paid to the generation of boulders not only on the tracks but also in adjacent areas or between one track and another, so, in those areas where the presence of the boulder is not considered dangerous for the normal flow of the train. This is done to train the system and subsequently test it on such scenarios, ensuring that they are not recognized as false positives.

The only form of variability that it was not possible to maintain within this generation process was the blur typical of a moving train, given that the camera is mounted on the vehicle. Specifically, within both selected datasets, there were slightly blurred images simulating the movement of a traveling train. However, on these images, the GAN network provided by Photoshop, in most cases, could not perform a satisfactory generation. In these cases, the added boulder always appeared very sharp, becoming poorly integrated into the surrounding space. For this reason, it was decided to delegate this form of generalization to a subsequent step, and so, ensuring that samples of this kind were present through the application of a preliminary data augmentation procedure before feeding the samples into the network.

It is clear that the just-described automatic procedure has proven to be extremely efficient in terms of time. However, within the same configuration, the positions were fixed, so the generated images risked having limited variability. In the same way, it would not have made sense to introduce variability in the position through the generation of rocks in random positions in the image, indeed, this would have entailed the risk of generating numerous unusable images to be discarded. Therefore, this automatic procedure was used for generating less than half of the total images; all the others were generated manually, by choosing a correct position each time, introducing significantly greater variability in the samples, as the only images that continued to have exactly the same position were the three from the same generation cycle, and therefore, they also had the same background.

During the creation of the datasets, both for training and validation, as well as for testing, the need arose to label the obtained patches based on whether they contained the obstacle or not. This was done by exploiting the automatic generation of the ground truth file containing the coordinates of the bounding box of the generated object. In this way, it was sufficient to check for intersection between the bounding box associated with the obstacle and the coordinates describing the patches in the original image. This process was thus automated and made extremely faster. However, since the objects contained in the bounding boxes were not truly rectangular, there could have been borderline cases where the automatic algorithm would have claimed an overlap that was not actually present. To avoid these errors,

a manual review of all the produced labels associated with the patches was carried out.

3.2.3 Organization of the generated images

As specified earlier, for generating the images, it was decided to start from base images from two different datasets to achieve a complete lack of correlation between the data used for training-validation and those used for testing. In particular, the images contained in the RailSem19 dataset were used to create the training and validation sets. The images on which to generate the rocks were carefully selected to cover the maximum level of possible variability. The total number of generated images is 5500, which were subsequently divided into training and validation sets. Since the system for detecting boulders does not involve the direct analysis of the input image but the analysis of patches extracted from them through a procedure that will be detailed later, the division into the two sets was not done directly on the images but on the extracted patches. This split between the training set and the validation set was made to achieve a lack of excessive correlation between the two sets. Therefore, during the division, which had a percentage ratio of approximately 75%-25%, attention was paid to ensuring that all patches belonging to the same image, as well as all the generated images on the same source image (same background), all ended up in the same set, either the training or the validation set.

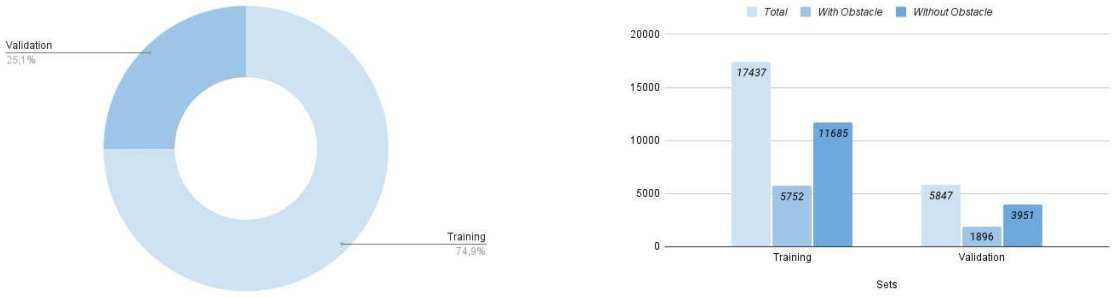

Figure 3.4: Percentage of patches among the various sets, number of total element, and number of samples containing obstacles.

As for the test set, it was obtained from FRSign images, for a total of 550 generated images.

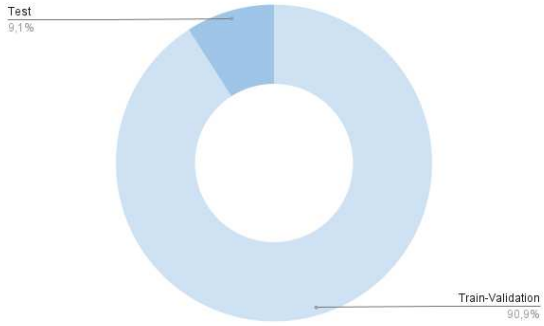

Figure 3.5: Percentage of samples used for train and evaluate the system and that used to test it.

In this case, care was taken to ensure that the only common element between the set used in the training-validation phase and that used in the testing phase was the methodology used for generation. Everything else, including background images, boulders, and their positions in the image, was completely different.

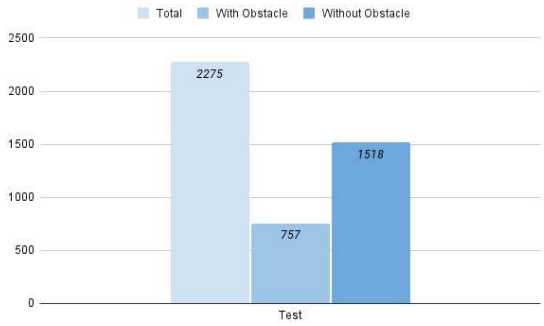

Figure 3.6: Total number of patches of Test Set, and relative number of samples with and without obstacles.

3.2.4 Technological innovations introduced

It is evident that the dataset generated in this thesis work is not the first of its kind. However, it certainly fits into a context where, although there are other existing datasets depicting boulders on tracks, they are not publicly available and not excessively large. Therefore, the innovation introduced in generating this dataset is primarily technological, residing in the tools used and how they were employed to achieve this goal.

The use of the GAN provided by the generative functionality of Photoshop has

allowed, unlike any other type of more traditional approach, such as copy and paste of the object, to obtain extremely realistic images. The results of this generation are boulders completely well-integrated into the target domain, namely the background image within which they were decided to be inserted. This perceptible naturalness is also evident when zooming into the image, deriving from the fact that the contours of the boulder are well-integrated into the scene, the object is perfectly aligned in terms of lighting and size relative to its position, and it is well-supported on the ground. Specifically, unsuccessful samples that did not meet all these prerequisites were discarded.

Furthermore, the use of background images with semantic segmentation of the tracks for a significant portion of the dataset, as well as the automatic generated ground truth files for the obstacle, leaves room for future developments for systems with different approaches than the one proposed.

So, compared to the analysis of the state of the art, it is evident that a first aspect of innovation is dictated by the introduction of this dataset, generated through a successful and well-varied mechanism, although not extremely extensive. The not excessively high number of data, coupled with a procedure that, in fact, imposes no limits, leaves room for a further challenge, which could be an expansion of this data collection, perhaps by including objects equally infrequently represented in the railway scenario, such as fallen tree trunks.

3.3 Definition of the proposed architecture

The following will proceed to illustrate the architecture of the proposed system, both from a hardware perspective (for example, details regarding the camera assembly) and from a software perspective. This will include a detailed explanation of the various software components of the system under analysis and the relationships that exist between each of them. Finally, the methodological innovations characteristic of the proposed system will also be detailed in comparison to the methods already present in the state of the art described in the previous chapter.

3.3.1 Operational Framework - Hardware architecture

The hardware architecture in which the proposed system is integrated assumes that the acquisition of images for analysis is carried out through a camera mounted in front of the vehicle, as previously specified. In fact, this is the operational framework in which the software system has chosen to be embedded, but it is certainly not the only possible solution. The decision was made considering a fixed camera mounted along the railway tracks as another possible alternative.

In this latter case, it would obviously have been necessary to plan for a greater number of installations. Assuming that the geographical area to be monitored is clarified based on the type, functionalities, and characteristics of the cameras being installed, a design for their arrangement would be necessary to ensure maximum coverage of the areas. Certainly, a solution of this kind poses significant challenges in terms of electrification and maintenance. This is because many of the areas to be monitored are likely to be far from inhabited centers, making them less accessible to operators. Additionally, since this surveillance system would tend to become well-integrated into the railway's infrastructure component, any maintenance activities would likely result in disruptions in travel and the need to establish alternative routes.

Certainly, a system of this kind, with cameras distributed along the area to be monitored, offers the possibility of not imposing excessively stringent constraints on the performance of the cameras to be used. In the sense that it is undoubtedly

useful to have a sufficiently strong focal length, but it is not necessary since obstacle detection is not linked in any way to the approach of the train. Furthermore, there are certainly fewer prerequisites for the artificial intelligence systems to be used on such cameras. In fact, each camera will, on average, capture images with the same background, with only variations in lighting and weather conditions and, furthermore, being a fixed camera, it is unlikely that the acquisitions made will suffer from blurring.

The disadvantage of such an approach, which led to the preference for the on-board vehicle camera system, is essentially the fact that the number of installations grows exponentially based on the extent of the area to be monitored. This undoubtedly requires increasing complexity in the network that needs to be set up to ensure that the various devices communicate. In particular, with this approach, assuming the software system is distributed, a communication mechanism needs to be established so that each surveillance device communicates the danger either to the trains passing through that section at that moment (which would require setting up a dynamic communication mechanism) or to the nearest stations, which would then communicate the danger to the trains passing through those sections when they arrive.

On the other hand, the proposed architecture involves installing the camera on board the vehicle, with the aim of creating a unified system well-integrated between hardware and software components, thus limiting potential communication latency. Specifically, in the case of obstacle detection, communication with the train itself would undoubtedly be more immediate, avoiding the involvement of railway stations and the need to electrify difficult-to-access areas. This setup would also ensure much easier maintenance, resulting in a significantly more scalable system.

However, the substantial advantages just outlined come at the cost of the system software's performance. It becomes crucial to have a system capable of generalizing well for various background scenarios, lighting and weather conditions, and resistant to blurring and acquisition noise in general. Furthermore, considering the specifications of the camera in terms of detection distance, it is still necessary to develop a system with a very good processing speed, as image sharpness, and therefore obstacle detection, increases proportionally with the decrease in distance

between the vehicle and the obstacle. Thus, finding a good trade-off between these aspects and the braking time required by the train, which is strongly dependent on the speed of movement, is essential.

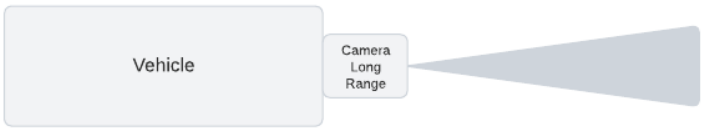

Figure 3.7: Proposed hardware architecture with a single long-range camera at the front of the vehicle.

Throughout the just-conducted description, we have discussed the on-board vehicle camera, referring to a single device. However, it is not necessarily the case; one could envision the use of long-range cameras for more extended depth visualizations, placing one at the front and one at the rear if the vehicle can travel bidirectionally. On the sides, the use of short-range cameras with less depth but wider vision could be considered, offering support in situations such as curves. Certainly, the frames collected from various video sources will need to undergo processing by the implemented software system. For generating or not the alarm, a combination mechanism could be employed. This could involve a mechanism of hard or soft voting among the output produced by the system analyzing the frames collected from different perspectives.

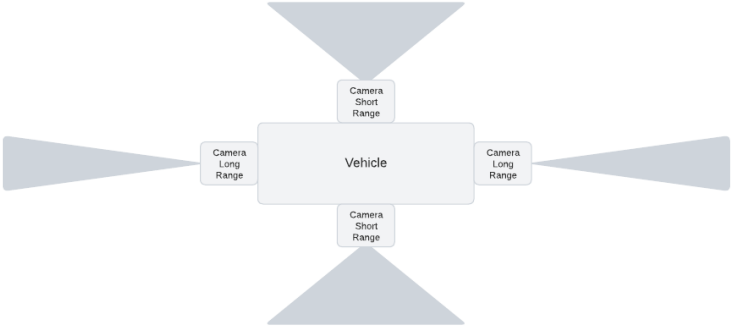

Figure 3.8: Proposed hardware architecture with multiple long-range and short-range cameras.

This hardware architecture will not be further detailed in terms of camera

specifications and quantity to be installed on board, but it is useful only for a detailed framing of the proposed system. In particular, indeed, to design such a hardware system in greater detail, it would be necessary to at least know the speed of movement of the train in question. So, it is assumed that the proposed software system is sufficiently scalable and therefore adaptable to a configuration with multiple cameras and a potential mechanism for combining the obtained outputs.

3.3.2 Software architecture of the proposed system

Within this section, we will proceed with an initial description of the high-level software architecture, identifying the main components and how they interact with each other. Subsequently, each individual module will be more detailed, highlighting the functionalities offered and the implementation with which it has been realized.

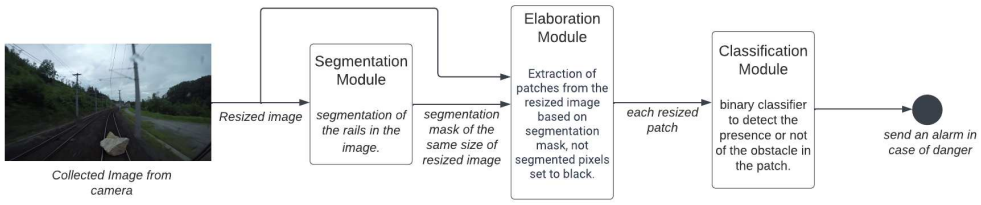

Figure 3.9: High Level Software Architecture.

In the above image, it is possible to visualize a high-level diagram of the software architecture of the implemented system. Reconnecting to what was mentioned for the hardware part, if the decision is made to use multiple cameras, the process we are about to detail must be executed for each of them and, of course, for each acquisition. Starting from the input image captured with the camera, it undergoes resizing to be compatible with the first module responsible for perform semantic segmentation on it. Specifically, as mentioned multiple times in previous sections, the task of rock detection on the tracks we are trying to accomplish consists of two main components: one focused on identifying the region of interest and another focused on analyzing the presence of obstacles in that region. In this regard, the input image will be initially analyzed by the segmentation module, which will generate a mask highlighting pixels belonging to the tracks. Once this mask is

obtained, the obstacle needs to be identified in the region of interest.

Compared to what has been analyzed in the state of the art, it is clear that various approaches can be taken. For example, a detector could be directly employed after the segmentation module to evaluate the overlap between the identified ROI and the predicted bounding boxes. Not wanting to use a classical detector within the proposed method, a processing module has been introduced. This module takes the image processed by the segmentation module and based on the segmentation mask given as output by the previous module, divides the source image into patches. Each patch corresponds to the minimum width rectangle, with a fixed height, to contain the segmented reference track portion.

Within the obtained patches, an additional step involves setting all pixels not included in the segmentation region to black, ensuring that these pixels are not effectively considered in the analysis.

At this point, the proposed solution involves taking each of these patches, resized as needed, and inputting them into the classification module, which will determine the presence or absence of an obstacle within that specific patch.

Segmentation Module

The segmentation module is essentially composed of a neural network dedicated to semantic segmentation. The employed neural network was a BiSeNetV2, and its implementation was found in a GitHub repository. This implementation likely deviates from the official one, as the authors of the official implementation have not provided implementation details.

Starting from the RailSem19 dataset [19], which was used for training this architecture, modifications were made concerning the considered classes and the dimension of the input image. All the input images were resized to 1024x512 in order to fit the network architecture.

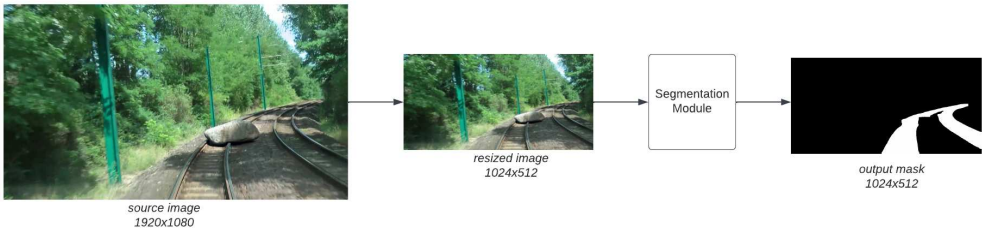

Figure 3.10: Steps of the Segmentation module. The Obtain mask is a binary mask.

All classes that did not represent objects related to the railway tracks were not taken into account, and training was conducted only on the "rail-raised" and "rail-track" classes, along with the "background" class. It is also important to specify that during the inference phase, an additional modification is made, combining the two classes that effectively represent the edges of the railway tracks and the area between them into a single class. This was done because, for the subsequent steps, such as obtaining patches and obscuring all other pixels related to the "background" class, it is not relevant to maintain this distinction.Therefore, the output of this network is essentially converted into a binary segmentation mask. On the output mask obtained from the network, a morphological operation of closure is further applied. This morphological operation involves the combination of two morphological operations, namely dilation followed by erosion.

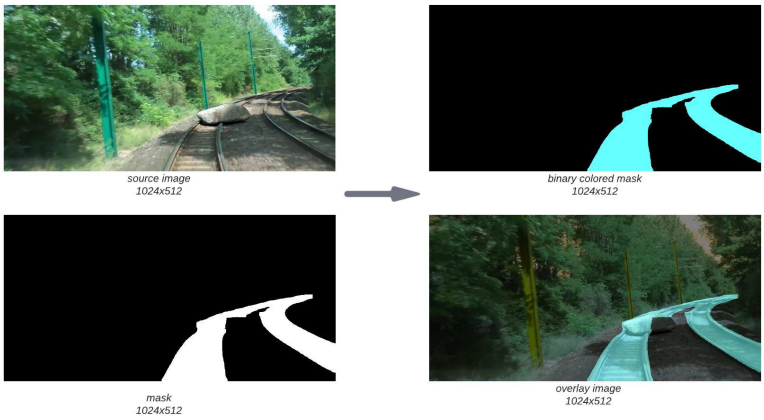

Figure 3.11: Starting from the mask and the color coding associated to classes it is possible to obtain the colored binary mask. Multiplying this values for a re-scaling factor and add this to source images pixels it is possible to obtain the overlay image.

The purpose of this morphological operation is essentially to fill in any areas

of the segmentation mask where holes have been created. This precaution was considered both to make the segmentation mask more uniformly shaped and to address a practical issue arising from the fact that the network was trained on images of railways (RailSem19) without interruptions on them. However, during inference, images with interruptions on the tracks (rocks) are presented to the network, which could lead to interruptions in the segmentation mask. The potential problem here is that patches are generated on areas recognized as railway tracks. So, if the presence of a rock causes the segmentation network to fail, the patches containing the rock won't exist. As a result, the danger won't be detected, rendering the system completely non-functional. However, it's important to note that, in general, the trained network is capable of generalizing well to images containing interruptions caused by rocks, seamlessly incorporating the rocks into the segmentation. Certainly, the introduced morphological operation provides an additional safeguard against a risk that could be completely mitigated by retraining the network on images with rocks, while keeping the ground truth unchanged.

Elaboration Module

Most likely, among the various modules of the software architecture, this represents the core of the innovation in the proposed method. In particular, this module starts from the combination of the resized input image to be fed into the segmentation module and the mask produced as output by the same module, these two will have the same dimensions. For the patch extraction process, it begins with the requirement to extract rectangles on the image that, with a fixed height, have the minimum width to contain each track, using the connected components algorithm. The selection of heights is done preliminarily and is the same for all images. The adopted criterion starts from the idea of wanting to maintain a sort of perspective visualization, meaning that as one moves away from the foreground, the area of the extracted patches should decrease. The heights established as the best for obtaining a division of the image into horizontal slices were [160, 128, 96, 64, 32]. This combination was chosen after several attempts, as it represented a good trade-off between having patches that were not excessively large, collecting background elements not directly of interest, and having patches that were too

small, making any depicted object indistinguishable within them.

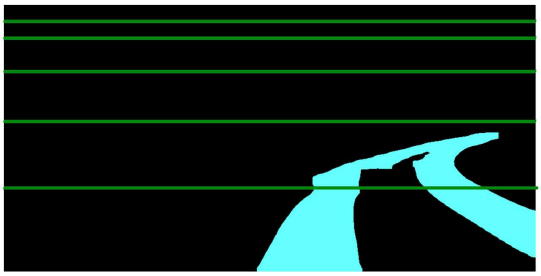

Figure 3.12: Heights of the slices done on the mask.

After that, on each of these image slices, more precisely on the corresponding segmentation mask, the connected components algorithm was applied. In this way, if there were pieces belonging to different tracks within a specific slice, this algorithm would identify them as separate components. At this point, the minimum bounding box to contain each of the connected components was identified. Given the coordinates of each of these bounding boxes, originally identified on the mask, they were then applied unchanged to the input image fed to the segmentation network, thus obtaining the desired crops. This process was iteratively repeated for each slice of the image containing pixels belonging to the tracks.

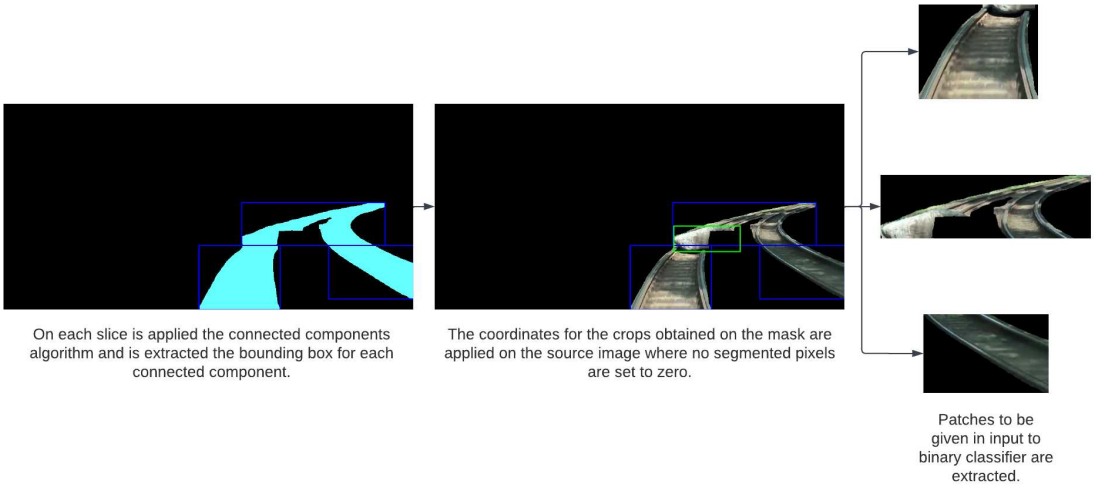

Figure 3.13: The entire process of patches extraction. The extraction of connected components is depicted on the colored mask, but only for visualization simplicity, as in reality, it is performed on the binary mask.

From the obtained crops, those whose size was smaller than a certain threshold

were eliminated. In addition, in the connected components algorithm, components that were entirely contained within other components were excluded. This could potentially be more harmful than helpful for the algorithm, not during training but during inference. In fact, it could happen that on two patches, one included in the other, a disagreement in the output generated by the system could occur, difficult to predict and therefore manage.

The need to divide the image into slices before applying the connected components algorithm arises from structural reasons in the images. In particular, the views of the tracks are frontal, not from above, so at a certain point, there is a convergence of even separate tracks at a point called the "vanishing point". This means that if the connected components algorithm were applied to the entire mask, it would identify a single large component, making it impossible to then divide the patches.

The idea of dividing into patches fundamentally arises from the need to address certain issues. Specifically, if a simple detector from the YOLO family had been used after image segmentation, it would have sufficed to determine the presence of an obstacle on the tracks by checking for intersection between the segmented region and the bounding box predicted by the obstacle detector. In the approach proposed here, however, intending to use a binary classifier downstream, there remained the problem of locating the detected object in order to consider it as an obstacle or not.

By using these crops, which should ideally contain only regions of the image related to the tracks and exclude others, if providing one of these to the binary classifier resulted in a positive outcome for the presence of a rock, it would be certain that the rock lies on the tracks and therefore poses a danger.

Certainly, since the shape of the tracks is often very irregular and influenced by perspective, it would be impossible to assume that the patches do not also contain regions of the image that do not actually belong to the tracks. However, this would expose the risk of false positives. A mitigation for this issue could have been to consider a smaller size for the patches, i.e., decrease the heights for the horizontal slices of the image. However, this would, in turn, have the negative aspect that often the rock itself would be divided among an increasing number of patches, each of which might contain parts so small that it becomes difficult for the binary classifier

to determine whether they belong to a rock or to another object or to the background soil.

To address this problem, the adopted solution was to set all pixels in each patch that were not attributable to tracks in the generated segmentation mask to black. In this way, it is possible to obscure the pixels not belonging to the region of interest, achieving the dual advantage of avoiding false positives (even if there were rocks or parts of them in non-interest areas, they would be hidden), and further simplifying the task of the binary classifier by focusing only on the region of interest.

The division into patches, in addition to meeting the aforementioned needs, also provides significant benefits to classification, mainly for two reasons. The first is that the number of generated images is not excessively high, certainly not to the point of covering scenarios with all possible configurations of rocks on the tracks. This could have resulted in gaps in the learning of the classifier. However, the division into patches completely frees the classifier from the surrounding scenario and from the position of the rock because it only perceives a very small part of it. The variability achieved in terms of background is more of a help to the segmentation module, which is still inevitable for the construction of correct patches. The second reason is that the classifier becomes accustomed to seeing not always the entire rock but often only small parts of it, and still distinguishes it from the surrounding space. This undoubtedly makes it more resistant to the presence of any obstructions or imperfectly successful segmentation. In this way, the classifier becomes accustomed not only to discriminate the object based on the edges that distinguish it from the surrounding environment but also, for example, based on a different texture.

Augmentation Procedure

What we are going to describe is not properly a module, in the sense of not being an architectural component that contributes to the definition of the final architecture. Instead, it is more of a pre-processing step that is performed on training images before submitting them to the binary classifier. So it is not part of the processing steps in which, during the inference phase, each image will be subjected. This pre-processing specifically involves the application of a data augmentation procedure primarily aimed at overcoming, as is commonly done, the

poor generalization that could arise from a dataset that is not excessively large. In particular, this procedure involves applying various transformations of variable magnitude chosen randomly, which may therefore not be applied to samples in the training set. This is done so that the network essentially sees many more samples that are effectively generated from modifications of existing training samples.

The possible modifications that have been chosen to apply are essentially aimed at compensating for those data acquisition prerequisites defined at the beginning of this chapter that could not be addressed through the used generation mechanism.

The first of the performed transformations involved the application of Gaussian noise, specifying the kernel size to be used and the standard deviation to create the kernel. The application of this transformation resolved the issue encountered regarding the inability to generate rocks on blurred images that would integrate well into the surrounding background, simulating captures that could be taken from a moving train. In this way, we started with images of sufficiently sharp background images, onto which rocks of similar sharpness could be applied. Subsequently, blur was applied to the resulting patches.

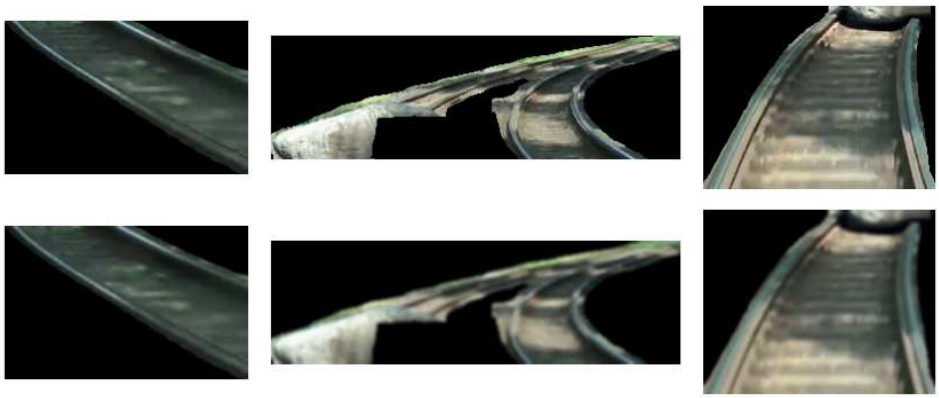

Figure 3.14: Examples of patches with the chosen blur applied.

It was also considered useful to apply a random conversion of patches to grayscale, defining the probability with which this transformation should occurs (0.2). The aim of applying this transformation stems from the intention to make the network capable of generalizing even on black and white images, which essentially represent the format that is closest to images that could be captured by a thermal

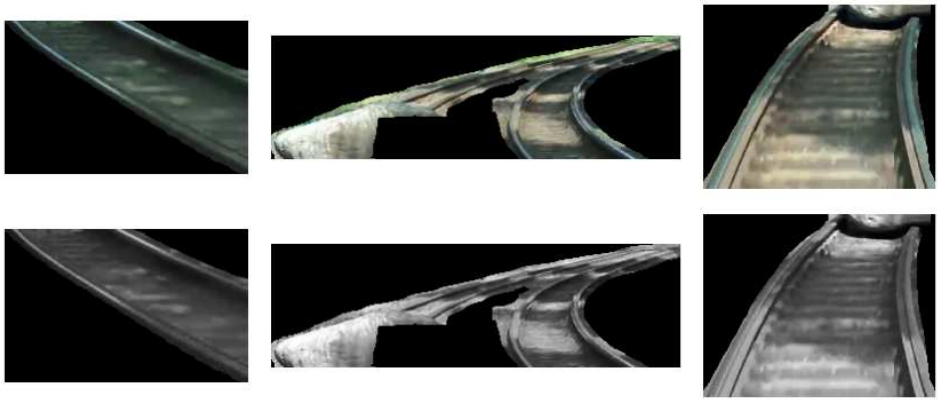

Figure 3.15: Examples of patches converted to grayscale if this transformation is applied with probability 1.

camera. This would make sense, for example, if the vehicle on which we are installing our system is an inspection vehicle that primarily operates at night. Of course, it would have been impossible to generate directly within thermal images of railway scenes, both because the available samples in such a scenario are significantly lower than those in the visible spectrum, and because the coloring of each object in a thermal image is related to the heat possessed by that object relative to others. Therefore, it would have been unthinkable to achieve faithful generation. In conclusion, the transformation to grayscale could be considered at least a good approximation and a workaround for a gap that could be filled only through an acquisition campaign and the development of a dedicated system.

In order to increase variations in terms of lighting conditions, for examples due to changing moments of the day or changing weather conditions, among the transformations, "ColorJitter" was also employed, aimed at randomly modifying the brightness, contrast, saturation, and hue of the image. Obviously, in this case as well, the resulting images will not accurately reflect the reality of actual weather conditions, or different moment of the day, but they will approximate it.

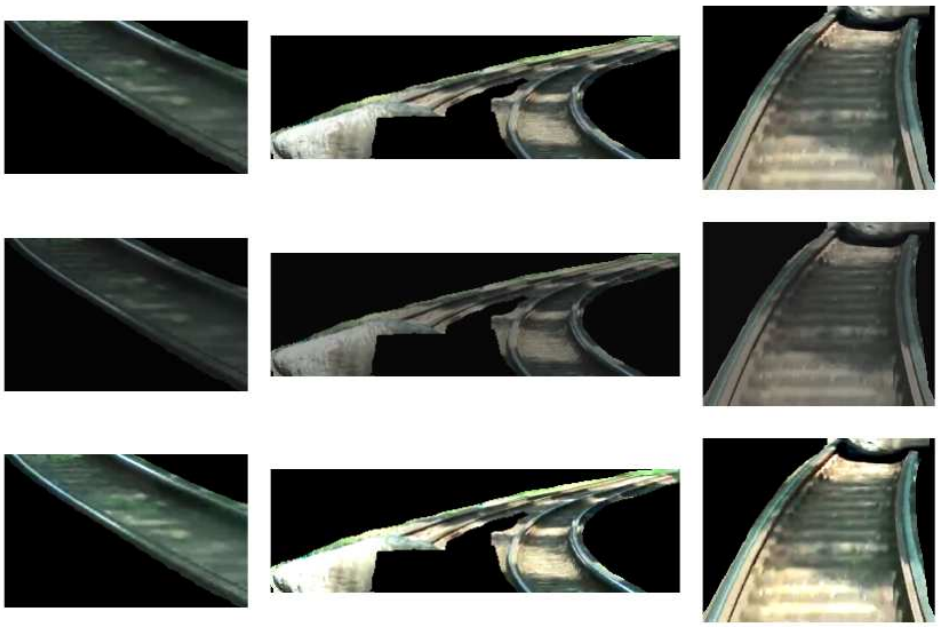

Figure 3.16: First row: original patches. Second row: examples of minimum values (0.8) of ColorJitter applied on patches. Third row: examples of maximum (1.2) values of ColorJitter applied on patches.

The last of the applied transformations was horizontal flipping, to be performed with a certain probability (0.5). This procedure was chosen as it would also contribute to increasing the variability of the samples. For example, assuming there are patches where, due to geometric reasons, other tracks are visible, this transformation would provide the opportunity to increase the number of positions where they are glimpsed, without being constrained by the availability of corresponding images. Similarly, regarding the curvature and inclination of the depicted tracks, this transformation would allow them to be seen both to the right and to the left.

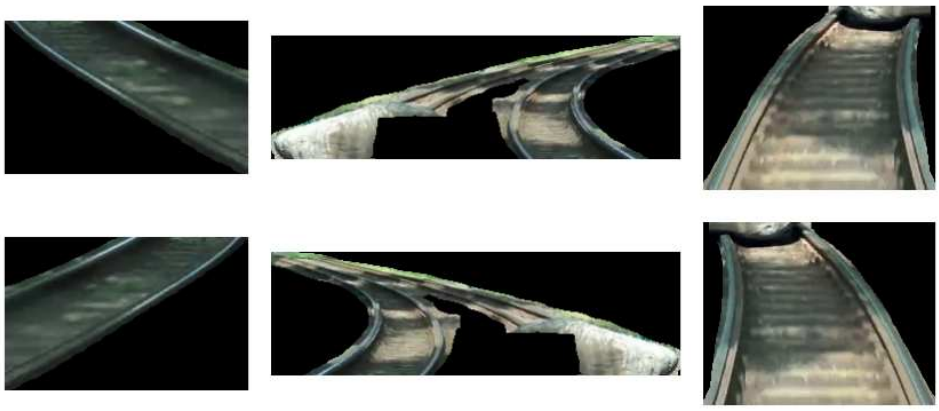

Figure 3.17: Examples of patches flipped horizontally if this transformation is applied with probability 1.

No further transformations were applied to the images to avoid the risk of making them excessively artificial. Other forms of image rotation or vertical flipping were also avoided as they would have represented scenarios that could never occur given the hardware configuration of the cameras, as we have already clarified.

Classification Module

The classification module is also comprised of a deep neural network that performs the binary classification task. Specifically, as described earlier, this network receives as input the patches generated by the previous module, resized to 224x224, and performs classification on them.

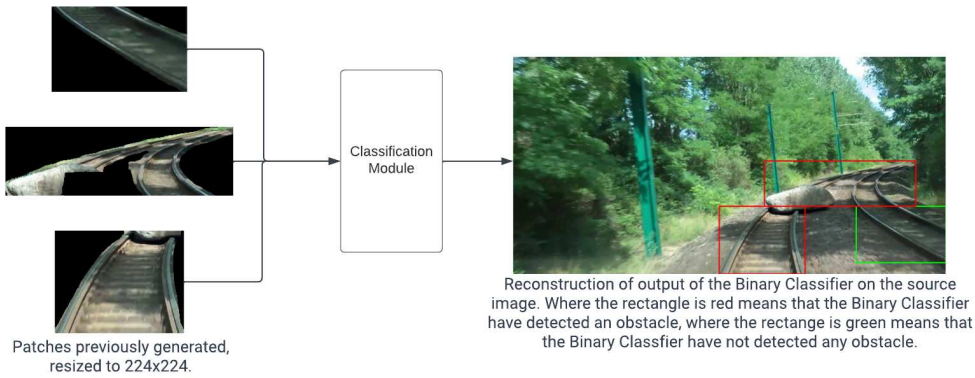

Figure 3.18: Visualization of the role of the binary classifier. Obviously, the image displayed as output is only a subsequent reconstruction og network's output, for the sake of visual understanding.

The binary classifier used consists of a backbone, which is the ResNet50, and an additional classification head formed by several layers. The image reported above is not a representation of all the layers contained in the proposed binary classifier, but rather the forward sequence traversed by the input element through the network. Furthermore, it is important to specify that the blocks indicated in the figure as *layer1/2/3/4* are not individual layers but rather blocks containing multiple layers, which have been grouped in this way for ease of visualization. .

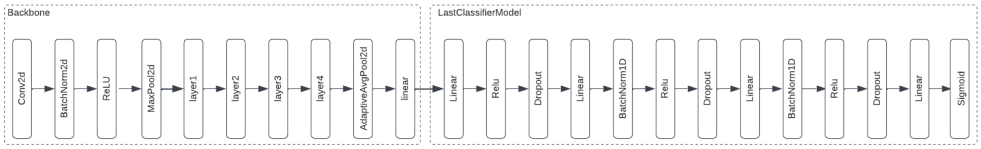

Figure 3.19: High level Binary Classifier Network.

Among the possible backbones for building the convolutional network, the ResNet50 has been chosen, which is one of the networks belonging to the family of Residual Networks. The power of this type of networks lies in the fact that they allow the incorporation of numerous convolutional layers without encountering the vanishing gradient problem, using the concept of "shortcut connections". These types of connections involve skipping certain layers and are the characteristic that distinguishes regular networks from residual networks. Generally, it is often thought

that increasing the number of layers should lead to an improvement in performance. This statement could be true if the phenomena of exploding or vanishing gradients did not exist. In particular the problem with the deep neural network is that, especially at the beginning of the training, the later layers have values that are quite far from their "destination" (from the optimal solution). So the gradient information that they propagate back can be much different from the one that would be seen near the optimum value. This phenomenon is called degradation and it could be considered as a consequence of the fact that earlier layers do not see directly the loss function but they only have an indirect view of this, on the base of which they optimize their weights. This could represents a problem especially if the successive layers are far from the optimal position, and thus the earlier layers waste time moving in a direction that is not the most direct path towards the optimal weights.

A possible solution to this problem is to make the gradient of the loss to arrive at the first layers with less intermediate steps, but this solution is in contrast with the increasing number of layers of deep neural networks. So, the only solution is to have a non-sequential architecture in which the gradient propagates through several paths, some of them longer, to get the benefits of deep learning and some other short, to address the degradation problem. Specifically, the basic idea is to use as a building block a group of layers that include a skip connection, so a connection that skips some of the layers.

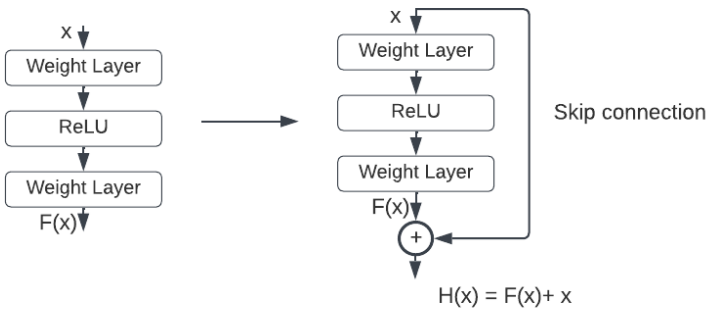

Figure 3.20: Using skip connections it is possible to pass from first scheme to second one.

In this way, during gradient propagation, the skip connections provide a "fast

lane" for the gradient to quickly arrive to the initial layers. So from another point of view is like that each building block instead of learning its own function H(x), is learning the residual F(x) = H(x)-x, so for this reason this technique is called residual learning. And the benefit of these shortcut identity mapping were that there was no additional parameters added to the model and also the computational time was kept in check.

The ResNet architecture follows two basic design rules. First, the number of filters in each layer is the same depending on the size of the output feature map. Second, if the feature map's size is halved, it has double the number of filters to maintain the time complexity of each layer. The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, known as a "bottleneck", which reduces the number of parameters and matrix multiplications. This enables much faster training of each layer. It uses a stack of three layers rather than two layers.

For the training of the choosen backbone it was decided to start not from random initial weights of the backbone, but from an already done pretrain of this network on ImageNet [14] dataset. According to this pretrain, the number of outputs of the last linear layer were equal to 1000 that is the number of classes used in the aforementioned dataset. So, in essence, this also justifies the addition of the layers contained in the classification head "LastClassifierModel".

The added layers certainly served to add a series of parameters that during the training phase would also change, carrying out the necessary domain transfer on the proposed model, notably, ImageNet is a dataset containing various types of objects but was not specific to rocks. But they also served to achieve a non-abrupt reduction, given that the number of neurons required in the output of our model is 1, as it is a binary problem. For this reason, normalization layers were added to perform batch normalization, in order to speed up the learning reducing coupling between the layers, so making the later layers independent of the mean and of the variance of the outputs of the earlier layers which undergoing large weights updates during the initial phases of the training. Linear layers were added to reduce the output dimension. Dropout layers were added to still control overfitting since the number of parameters in the network was not reduced compared to a not excessively

large dataset. The loss used during the training of the binary classifier network was the Binary Cross Entropy, whose formulation is reported above.

$$H(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

where $H(y, \hat{y})$ represents the binary cross entropy, N is the number of samples, y_i is the true label for the i-th sample (either 0 or 1) and $\hat{y}_i$ is the predicted probability of the i-th sample belonging to class 1.

3.3.3 Methodological innovations introduced

So far, it has been described the software architecture of the implemented system, and following the description of various modules, it is possible to observe a complete example of the processing chain to which each input image is subjected. Regarding the innovative contribution in technological terms, it is mainly attributable to the dataset generation procedure, while the proposed system itself presents innovations mainly of a methodological nature. Indeed, the technologies used are state-of-the-art, so the innovative component lies in the combination mechanism. The goal was to replace what could have been a simple detector downstream of the segmentation module with a more streamlined and efficient procedure. Therefore, a contribution in terms of computational benefits is evident, as complex networks were not used; instead, a binary classifier was employed, and the processing module does not utilize deep neural networks but relies on very simple and traditional operations.

Certainly, another benefit obtained is in terms of performance. In general, the use of a detector should offer greater success guarantees, as evidenced by the state of the art. However, if one were to use a pre-trained state-of-the-art detector, fine-tuning it with rock images would inherently require the network to identify the rock not only as an object different from the tracks but also discretize it concerning other objects it is accustomed to seeing. Thus, a significantly large number of samples would be needed, comparable to the number available for other classes. The same problem in terms of the number of samples would have been encountered, especially if one wanted to train a detector from scratch. Through the used procedure, the absence of many data samples is overcome because the binary

classifier is not trained on the entire image but on patches extracted from it, which are more than four times greater in quantity. Also, as mentioned earlier, the patch generation procedure avoids the need for extensive data, as it extracts only parts of the image that are concentrated on important subjects, avoiding influences from the surrounding environment that would otherwise require a dataset's variability (in geographical and structural terms) and quantity that would be truly difficult to achieve, enabling the network to be independent of it and have good generalization ability.

Furthermore, this thesis work focused on detecting rocks on tracks but fits into a broader context of the need to identify obstacles of any kind on the tracks, provided that they pose a danger to the train's operation. Making this assumption, it is clear that obstacles are all potentially dangerous in the same way, so classifying them may lose its relevance. Therefore, the implementation of this binary classifier could meet the broader need not to make the decision "rock present vs. rock not present" but rather "obstacle present vs. obstacle not present". To achieve such a goal, it would be necessary to have a discriminator that distinguishes the track from anything that obstructs it, and it is clear that the system implemented could succeed in this objective. Having trained the binary classifier to recognize the presence of rocks even in situations where only a small part of it is visible ensures that the rock is not recognized as such but as an obstruction of the track. Therefore, it is likely that such a system could generalize well even if the part of the obstacle it sees is not attributable to a rock but to any other object.

3.4 Additional Tools and Technologies used

Within this section, we will describe the remaining tools and technologies used for the creation of the dataset and the proposed software system. Regarding the state-of-the-art, neural networks that were employed, their respective losses, and the procedures used during training, although they could be part of this section, they have already been described earlier as components of the implementation.

3.4.1 Python

Python is high-level, versatile, and interpreted programming language that has gained immense popularity, particularly in the field of artificial intelligence (AI). It's an easily readable programming language with simple syntax that allows developers to express concepts in fewer lines of code compared to languages like Java or C++. This simplicity enhances code readability and maintainability. It also supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It is particularly useful because comes with a vast standard library that includes modules and packages for a wide range of functionalities and has also a large and active community of developers. One of the most important things is that python is platform-independent, meaning that Python code can run on various operating systems without modification. This enhances the portability of Python applications.

The popularity of this programming language in artificial intelligence is due to the richness of its ecosystem of libraries and frameworks, including TensorFlow, PyTorch, scikit-learn and Keras. These libraries provide pre-built modules for tasks such as data manipulation, model building, and evaluation. Another important aspect that is really useful in AI development is that Python can seamlessly integrate with other languages and technologies, this makes possible the usage of different tools and languages. This flexibility is final accompanied by the fact that Python follows an open-source philosophy, and this aspect fosters collaboration and innovation also within the AI community.

3.4.2 PyTorch

PyTorch is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). It has gained significant popularity in the machine learning and artificial intelligence communities due to its dynamic computational graph, flexibility, and user-friendly interface. Unlike some other deep learning frameworks with static computation graphs, PyTorch employs a dynamic computation graph through its dynamic neural network (DNN) capabilities. This allows for more flexibility in model design and easier debugging. In particular with PyTorch, computations are executed eagerly, meaning that operations are performed immediately upon invocation. This makes it easy to inspect and debug code by allowing developers to print intermediate values, inspect gradients, and modify the model dynamically during runtime. As for flexibility PyTorch allows for dynamic model building that, unlike frameworks that require a predefined static graph, is particularly beneficial in scenarios where the structure of the model depends on runtime conditions, such as in natural language processing tasks. PyTorch uses tensors as its fundamental data structure, similar to NumPy arrays. Tensors in PyTorch can be seamlessly converted to and from NumPy arrays, providing compatibility with the broader scientific computing ecosystem. It also provides the torch.nn module, which simplifies the process of building and training neural networks. It includes pre-defined layers, loss functions, and optimization algorithms, making it easy for developers to construct complex neural network architectures. It incorporates an automatic differentiation library called Autograd, a feature that automatically computes gradients for tensor operations, facilitating gradient-based optimization techniques used in training neural networks.

So its design principles are aligned with the iterative and experimental nature of research in artificial intelligence and machine learning, and it is the reason why it has been widely adopted in the research community.

3.4.3 Generative Fill in Photoshop

Generative Fill, part of the revolutionary and magical new suite of features based on Firefly and generative artificial intelligence, allows users to add, expand,

or remove content from images non-destructively, using text prompts in over 100 languages. Generative Fill is powered by Adobe Firefly, the family of generative creative models designed to be safe for commercial use. Firefly is trained on hundreds of millions of high-quality, professional images licensed by Adobe Stock, which are among the highest quality in the market. This tool, therefore, offers the ability to generate or remove objects, generate backgrounds, or expand images.

Chapter 4
Experimental validation and application aspects

Within this Chapter, we will present the chosen model as the result of an experimental process that involved the analysis and comparison of various possible solutions. The effectiveness of the proposed solutions naturally required the use of specific metrics, which will be explained later. These metrics were selected based on their appropriateness to the described problem, and all analyses were conducted using them. Finally, we will proceed to analyze the obtained results, examining their significance and characterizing them. This will lay the groundwork for tracking future developments that the proposed system could undergo. Concurrently, an analysis will also be conducted on the potential generalization ability of the proposed system in scenarios similar but not identical to the one considered for addressing the problem.

4.1 Description of the evaluation metrics for the results

The definition of the final system, as previously mentioned, occurred comparatively with other possible systems that differ in terms of training procedure or architecture type. Therefore, in order to make a choice based on objective parameters, some metrics were used, which will be explained below. The metrics used and subsequently presented were chosen from among the possible ones based on their relevance to the specific case and their utility. Certainly, the

conducted experiments focused on the binary classifier and not also on the semantic segmentation. Therefore, the metrics reported below are all related to binary classification.

4.1.1 Accuracy

Accuracy is a parameter often taken into consideration for the analysis of artificial intelligence systems. It is the percentage of correct classification that a trained model achieves, so in particular it is the number of correct predictions divided by the total number of prediction across all classes. So, accuracy is a proportional measure of the number of correct predictions over all predictions, where correct predictions are composed of true positive (TP) and True Negative (TN), while all predictions are composed of the entirety of positive (P) and negative (N) examples. In the composition of the positive examples are included true positives (TP) and false positives (FP), while in the composition of negative examples are included true negatives (TN) and false negatives (FN). So accuracy (ACC) can be defined as follow:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

So accuracy is a simple indicator of model performace and obviously of model error in fact it can be seen also as $(1 - error)$.

Although accuracy is a very simple and summary metric, it is particularly meaningful in cases of balanced classification tasks, so in situations in which the number of classes appear in comparably equal numbers. Indeed, in situations where this is not the case, there is a risk of obtaining a method that performs well in terms of accuracy simply because the events belonging to the class with a higher error rate occur very infrequently. However, this would mean that even a system with very high accuracy, when placed in a real-world context where the less frequent cases are of interest, would be entirely useless.

4.1.2 F1-Score

So, accuracy was still reported for the various proposed models, but it was considered only partially. Instead, the F1-Score was introduced and used. To correctly define this metric, it is necessary to take into account precision and recall. In particular, the former refers to how many of the positive predictions made by the model are correct, while the latter refers to how many of the actual positive samples in the dataset were correctly identified by the model.

Precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Often, these two aspects offer a trade-off in the sense that improving one comes at the expense of the other. Specifically, having higher precision makes the classifier more stringent in assigning the positive class, so it means it is mostly right in saying

positive, thereby reducing recall. Conversely, a higher recall implies having a more permissive classifier that assigns the positive class more easily, so it is mostly right in saying "negative", resulting in a reduction in precision. To achieve a perfect classifier, the ideal would be to maximize both.

The F1-Score combines precision and recall using their harmonic mean, and maximizing the F1-Score implies simultaneously maximizing both precision and recall. Thus, the F1-Score has become the choice of researchers for evaluating their models in conjunction with accuracy. The harmonic mean of a set of numbers is equal to the ratio of the number of values considered to the sum of the reciprocals of the numerical values, so considering to have n values $x_1, x_2, \ldots, x_n$, their harmonic mean H will be:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}}$$

Therefore, the F1-Score is calculated as follows and has a value between 0-100%, where a larger value indicates a higher quality of the classifier:

$$\text{F1-Score} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Starting from this formulation, it is clear why the harmonic mean is used to calculate the F1-Score rather than the arithmetic or geometric mean. In particular, the harmonic mean encourages similar values for precision and recall, meaning that the more precision and recall values differ from each other, the lower the harmonic mean will be.

Therefore, based on these considerations, the greater detail and reliability of this metric compared to simple accuracy are evident. This is why it was decided to use it more extensively for model evaluation.

ROC Curve

The ROC (Receiver Operating Characteristic) Curve condenses in one chart the relation between False Positive rate and False Negative rate for different values of the operating parameter τ. So, the two axes of the chart are the false positive rate or (1-specificity of the test) and true positive rate or sensitivity of the test and each point of the curve represents a different value of τ.

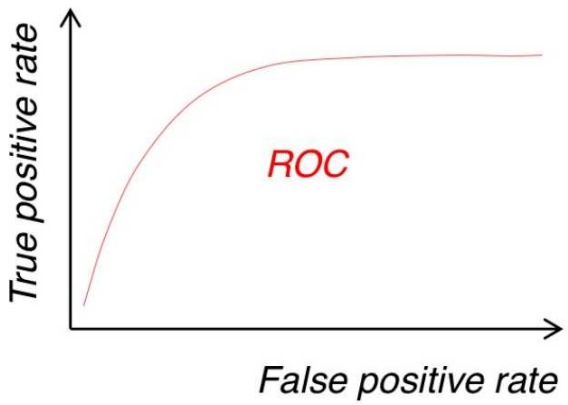

Figure 4.1: ROC Curve.

In general, a test is more accurate the closer its ROC curve approaches the upper-left corner of the chart. Furthermore, the point closest to this corner represents the cutoff value that maximizes both sensitivity and specificity of the test, so the area under the curve (AUC) represents a measure of how accurate a test

is. In fact if the AUC is 0.5, the test is not informative (the curve would correspond to the bisector), if, on the other hand, the AUC has a value between 0.5 and 1, the test is increasingly accurate. So this chart could be used to evaluate the performance of a system for example by comparing the values of their respective AUCs. However it is important to consider that this method has its limits, because there could be cases where tests have different ROC curves but equal AUC values, for which it is necessary to do a comparison across different intervals or ranges.

It could be useful, in some cases, to find the cut-off value that maximizes both sensitivity and specificity of the test, known as the optimal cut-off.

Intuitively, if we want to maximize both sensitivity and specificity simultaneously, as both are dependent on the cut-off value (c), we could maximize a function given by their sum and then find the cut-off value for which this sum attains the maximum value. One method to determine this cut-off is the Youden's index, defined as the maximum distance between the ROC curve and the bisector (chance line), it is denoted as:

$$J = \max_{c}\{\text{sensitivity}(c) + \text{specificity}(c) - 1\}$$

Note that maximize $\text{sensitivity}(c) + \text{specificity}(c) - 1$ is equivalent to maximize $\text{sensitivity}(c) + \text{specificity}(c)$. However, the given definition of Youden's index is interesting because it can be easily represented on the graph (a vertical segment indicating the maximum distance between the ROC curve and the bisector).

The importance of the ROC Curve also lies in the fact that in many systems FP and FN does not have equal weight, so changing the value of τ it is possible to move along the curve tuning the binary classifier for having more precision or more recall. In fact the system will choose "positive" if the probability is greater than τ, else "negative", so for increasing values of τ the system will have higher precision and lower recall.

4.1.3 Assessment on image

Certainly, the metrics that have just been presented, and more specifically the F1-Score, were used for the selection of the best model by evaluating on patches, which were what our system was trained on. The metric we are about

to describe, on the other hand, aims to assess the various systems developed during the experiments with a greater focus on a practical aspect. In other words, it is a more pragmatic metric oriented towards evaluating the model's performance in a real-world application scenario.

Specifically, in previous chapters, we have already discussed why we proceeded to divide the image into patches during the processing stage and the benefits that an approach of this kind has had. However, from an application perspective, what matters is the diagnosis of obstacles within the entire image, regardless of what happens with individual patches. This might be a simplification for the implemented system, but it is indeed what is needed during the system's usage.

Considering that the goal of this system is to recognize the presence of obstacles, and therefore the hazardous situation, and based on this, to generate an alarm, the capability of our system to generate alarms when they are really needed and the ability to refrain from generating them when there is no actual danger, will be evaluated.

We will assume that we have calibrated the system so that the detection of just one positive patch is sufficient to trigger the alarm. Therefore, from a practical standpoint, it could be considered that it is sufficient that at least one of the patches really positive in an image is recognized as such by our system, to ensure that the generated alarm is not unwarranted.

So, it becomes possible to define the F1-Score also for images, establishing a new interpretation for TP, TN, FP, FN in relation to the entire image rather than patches. In particular, the images can be divided into two categories: positive (P) when a boulder is present, and negative (N) when the boulder is absent. Based on this, True Positives (TP) will be the images that were positive, where at least one of the really positive patches was recognized by our system. False Negatives (FN) will be those images containing the boulder in which none of the really positive patches was recognized. True Negatives (TN) will be the images that did not contain any boulders, and where indeed all patches were correctly recognized as negative. False Positives (FP) will be those images not containing boulders where at least one patch was recognized as positive, or those samples containing boulders where, however, only patches different from the genuinely positive ones were recognized as positive.

This interpretation of false positives also takes into account positional information, which, of course, could be managed differently. So, provided that these meanings are changed, the definitions of precision, recall, and F1-Score remain the same.

4.2 Experimentations

Within this section, a description and analysis will be provided only for the implemented models that prove to be more significant both in terms of the possibility of making comparisons and in relation to the presented results. The experiments carried out moved in two main directions, namely, variations involved both the proposed architecture or training procedure and the dataset used. Specifically, since the dataset was generated, there was no initial constraint in terms of the number of samples to be created. Instead, it was more of an incremental process during which, by evaluating the results obtained on the models during training and analyzing the errors made, there was an awareness of the potential need to increase the number of samples or the need to include samples that met specific requirements.

Obviously, as one would expect, experiments conducted with a greater number of parameters yielded more satisfactory performances. Therefore, in the analysis of the conducted experiments, those performed with non-final versions of the dataset will be excluded. Variations in experiments concerned architecture and training procedure, and in particular the specific backbone to be used, any layers to be added, and parameters such as batch size or learning rate.

In particular, it was decided to start with an initial backbone, which was the ResNet50. The reasons that led to the choice of this network were primarily dictated by the awareness of using a relatively small amount of data compared to what is generally required for training a deep neural network. Hence, it was thought that the use of a network that aimed to limit the vanishing gradient problem through the employment of skip connections could be beneficial, an issue that we would otherwise surely encounter. The benefits of using skip connections and the vanishing gradient problem have already been extensively detailed in the previous chapter.

Among the available networks belonging to the ResNet family, with pre-trained weights, ResNet50 was initially chosen as it represented the right compromise between the achieved accuracy and the number of parameters it consisted of. Specifically, despite a minimal loss in terms of accuracy, there is a substantial reduction in parameters compared to choosing ResNet152, which is the largest among them all.

Several subsequent evaluations were carried out to decide which layers should be added as the classification head. The purpose of these layers was undoubtedly to gradually reduce the output dimension and, above all, to add additional parameters. Even if the backbone were to be left unchanged with its original weights, these parameters could be modified for the specific task under analysis.

In the previous chapter, the chosen final configuration was reported, which proved to be the best and consisted of dropout layers to limit overfitting, linear layers to reduce the output dimension, and batch normalization layers to reduce coupling between layers.

With all the aspects just described fixed, the two parameters subjected to further evaluations were the batch size and the learning rate. The learning rate indicates the step of weight modification in the stochastic gradient descent optimization procedure. This parameter has a value that can oscillate between 0 and 1. During the gradient propagation phase, it determines what modification to make to the weights based on the obtained error. Clearly, this parameter controls the learning speed. A large learning rate could cause fluctuations in model performance, even leading to weight divergence, while a very small learning rate might prevent the algorithm from converging or cause it to get stuck in a non-optimal point.

To support such a delicate aspect, what is generally done, and was also used in these experiments, is the use of momentum. Momentum accumulates an exponential moving average based on previous gradients, adding inertia to the update procedure. This forces the updates to proceed in a direction similar to previous changes, thus avoiding unpleasant oscillations.

Given that the various training epochs are all different from each other and, therefore, require different weight updates, what is generally done, and was also done in these experiments, is to use a dynamic learning rate. In particular, a mechanism of decreasing the learning rate at certain intervals of epochs was employed in these experiments, defined by a certain multiplicative factor.

In particular, specific values were identified, either because they were recommended by various documentations or because they were combinations that had already been tested. All experiments were conducted with the same configuration. In particular, SGD (Stochastic Gradient Descent) was chosen as

the optimizer, with a learning rate of 1×10^{-3} and a momentum of 0.9. For the dynamic update of the learning rate, a step size of 10 was used, and the value of gamma was 1×10^{-4}.

The batch size, on the other hand, determines the number of samples to work on before updating the weights. Choosing this aspect also required a compromise, in the sense that a small batch size could lead to incorrect weight updates, especially for a single sample, perhaps an outlier. On the other hand, with large batch sizes, significantly greater computational resources are required, and there are greater challenges in finding global optima. Therefore, a good trade-off in batch sizes to try has been between 64 and 128.

With these parameters fixed, the initial experiments conducted involved using ResNet50 as the backbone, modifying the number of trainable parameters, and applying data augmentation to the training samples. Regarding the number of trainable parameters, they were either all made trainable, starting from the weights of the backbone trained on ImageNet, or only the last of the ResNet50 blocks and the additional layers were unlocked. Other tested configurations with an even lower number of trainable layers will not be reported, as they did not yield good results. This is evident considering that the pre-existing weights were tuned for different classes than those under examination.

After finding the best configuration with ResNet50 as the backbone and keeping the settings unchanged, two other backbones were tested: a larger one, ResNet101, and a smaller one, MobileNetV2, in order to test the obtained benefits. All the trainings have been done for 100 epochs.

4.2.1 Backbone ResNet50 - Partial Retraining

Here are reported the results obtained leaving unchanged the hyperparameters of the training procedure, having the ResNet50 as backbone and the added layers already described. A comparison that has been conducted and is reported below concerns the use or non-use of the dynamic data augmentation procedure described, before training. Regarding the trainable parameters, in this experiment, we started with the weights of the ResNet50 trained on ImageNet. However, all the additional layers were then made trainable, and only the last part of the backbone starting

from the block 4, that is the last block.

Training Results	Validation Results	Real TestSet Results	Generated TestSet Results
Loss: 0.1392 Acc: 95.2400	Loss:0.1595 **Acc: 94.185** TP:1691 TN: 3816 FP:135 FN:205 precision: 0.9261 recall: 0.8919 **F1-score: 0.9087**	TP:18 TN:21 FP:1 FN:14 precision: 0.9474 recall: 0.5625 **F1-score: 0.7059**	TP:620 TN:1369 FP:166 FN:137 precision: 0.7888 recall: 0.8190 **F1-score: 0.8036**
Loss: 0.0480 Acc: 98.3828	Loss:0.1325 **Acc: 95.2112** TP:1731 TN:3836 FP:115 FN:165 precision: 0.9377 recall: 0.9130 **F1-score: 0.9252**	TP:18 TN:21 FP:1 FN:14 precision: 0.9474 recall: 0.5625 **F1-score: 0.7059**	TP:609 TN:1401 FP:117 FN:148 precision: 0.8388 recall: 0.8045 **F1-score: 0.8213**

Table 4.1: Results of Backbone **ResNet50** with added layers, retraining of **only last block** of the backbone and of **all added layers**, with a batch size of **64**, the first row is without augmentation while the second is with it.

Training Results	Validation Results	Real TestSet Results	Generated TestSet Results
Loss: 0.1367 Acc: 95.3604	Loss:0.1749 **Acc: 93.6891** TP:1681 TN: 3797 FP:154 FN:215 precision: 0.9161 recall: 0.8866 **F1-score: 0.9011**	TP:18 TN:20 FP:2 FN:14 precision: 0.9 recall: 0.5625 **F1-score: 0.6923**	TP:591 TN:1352 FP:149 FN:166 precision: 0.7986 recall: 0.7807 **F1-score: 0.7896**
Loss: 0.1144 Acc: 95.9855	Loss:0.1467 **Acc: 94.3219** TP:1678 TN:3837 FP:114 FN:218 precision: 0.9364 recall: 0.8850 **F1-score: 0.9100**	TP:16 TN:21 FP:1 FN:16 precision: 0.9412 recall: 0.5 **F1-score: 0.6531**	TP:578 TN:1422 FP:96 FN:179 precision: 0.8576 recall: 0.7635 **F1-score: 0.8078**

Table 4.2: Results of Backbone **ResNet50** with added layers, retraining of only **last block** of the backbone and of all **added layers**, with a batch size of **128**, the first row is without augmentation while the second is with it.

4.2.2 Backbone ResNet50 - Complete Retraining

Here are reported the results obtained leaving unchanged the hyperparameters of the training procedure, having the ResNet50 as backbone and the added layers already described. A comparison that has been conducted and is reported below concerns the use or non-use of the dynamic data augmentation procedure described before training. Regarding the trainable parameters, in this experiment, we started with the weights of the ResNet50 trained on ImageNet. However, all the additional layers were then made trainable, as well as the entire backbone.

Training Results	Validation Results	Real TestSet Results	Generated TestSet Results
Loss: 0.0235 Acc: 99.3978	Loss: 0.1191 **Acc: 96.3400** TP:1764 TN:3869 FP:82 FN:132 precision: 0.9556 recall: 0.9304 **F1-score: 0.9428**	TP:19 TN:21 FP:1 FN:13 precision: 0.95 recall: 0.5938 **F1-score: 0.7308**	TP:633 TN:1396 FP:122 FN:124 precision: 0.8384 recall: 0.8362 **F1-score: 0.8373**
Loss: 0.0477 Acc: 98.5376	Loss: 0.0901 **Acc: 97.0754** TP:1799 TN:3877 FP:74 FN:97 precision: 0.9605 recall: 0.9488 **F1-score: 0.9546**	TP:21 TN:20 FP:2 FN:11 precision: 0.9130 recall: 0.6563 **F1-score: 0.7636**	TP:641 TN:1450 FP:68 FN:116 precision: 0.9041 recall: 0.8468 **F1-score: 0.8745**

Table 4.3: Results of Backbone **ResNet50** with added layers, **complete retraining** of the **whole network** with a batch size of **64**, the first row is without augmentation and the second is with it.

Training Results	Validation Results	Real TestSet Results	Generated TestSet Results
Loss: 0.0657 Acc: 98.1992	Loss: 0.1115 **Acc: 96.5281** TP:1756 TN:3888 FP:63 FN:140 precision: 0.9654 recall: 0.9262 **F1-score: 0.9454**	TP:19 TN:21 FP:1 FN:13 precision: 0.95 recall: 0.5938 **F1-score: 0.7308**	TP:628 TN:1422 FP:96 FN:129 precision: 0.8674 recall: 0.8296 **F1-score: 0.8481**
Loss: 0.0285 Acc: 99.1971	Loss: 0.0922 **Acc: 97.1438** TP:1799 TN:3881 FP:70 FN:97 precision: 0.9625 recall: 0.9488 **F1-score: 0.9556**	TP:18, TN:20 FP:2 FN:14 precision: 0.9 recall: 0.5625 **F1-score: 0.6923**	TP:652 TN:1424 FP:94 FN:105 precision: 0.8740 recall: 0.8613 **F1-score: 0.8676**

Table 4.4: Results of Backbone **ResNet50** with added layers, **complete retraining** of the **whole network** with a batch size of **128**, the first row is without augmentation and the second is with it.

4.2.3 Backbone MobileNetv2 - Complete Retraining

Here are reported the results obtained leaving unchanged the hyperparameters of the training procedure, but placing a network with fewer parameter as backbone and the added layers already described. Among all the possible networks that satisfies this requirements it has been chosen MobileNetv2, because MobileNet networks family are generally used for situations in which real-time responses are required, and also because in terms of number of parameters and in terms of reached accuracy MobileNetv2 is a good trade-off between MobileNetv3-small and MobileNetv3-large. The only experiment that is done with this backbone is that with the same configuration of the best model obtained using ResNet50 among that

reported above.

Training Results	Validation Results	Real TestSet Results	Generated TestSet Results
Loss:0.0399 Acc: 98.7555	Loss:0.0917 **Acc:97.2293** TP:1794 TN:3891 FP:60 FN:102 precision:0.9676 recall:0.9462 **F1-score: 0.9568**	TP:12 TN:21 FP:1 FN:20 precision:0.9231, recall:0.375 **F1-score:0.5333**	TP:638 TN:1429 FP:89 FN:119 precision:0.8776 recall:0.8428 **F1-score: 0.8598**

Table 4.5: Results of Backbone **MobileNetv2** with added layers, with configuration with which best model with ResNet50 has been obtained so using **augmentation procedure**, **complete retraining** and a batch size equal to **64**.

4.2.4 Backbone ResNet101 - Complete Retraining

Here are reported the results obtained leaving unchanged the hyperparameters of the training procedure, but placing a network with more parameter as backbone and the added layers already described. Among all the possible networks that satisfies this requirements it has been chosen the ResNet101, as increasing the number of parameters made the vanishing gradient problem even more prominent, the decision was made to stay within the realm of residual networks. However, an attempt was made to understand the improvement achieved in terms of performance if a larger network were chosen. The only experiment that is done with this backbone is that with the same configuration of the best model obtained using ResNet50 among that reported above.

Training Results	Validation Results	Real TestSet Results	Generated TestSet Results
Loss:0.0580 Acc: 97.9698	Loss:0.1038 **Acc:96.7163** TP:1772 TN:3883 FP:68 FN:124 precision:0.9630 recall:0.9346 **F1-score: 0.9486**	TP:17 TN:21 FP:1 FN:15 precision:0.9444 recall:0.5313 **F1-score:0.68**	TP:618 TN:1391 FP:127 FN:139 precision:0.8295 recall:0.8164 **F1-score: 0.8229**

Table 4.6: Results of Backbone **ResNet101** with added layers, with configuration with which best model with ResNet50 has been obtained, so using **augmentation procedure**, **complete retraining** and a batch size equal to **64**.

4.2.5 Comparison of obtained results

Among the experiments conducted using ResNet50 as the backbone, the best results were undoubtedly obtained in situations where a complete retraining of the network was performed. This approach certainly ensured that all parameters, initially adapted to a generic classification problem over a very broad set of classes like those contained in ImageNet, were modified to become more oriented towards the specific problem at hand. Similarly, trials using the data augmentation procedure before training achieved better performance on the test set. This is easily justifiable considering that the modifications included and extensively described in the previous chapter undoubtedly provided greater variability to the samples, attempting to cover scenarios close to real-world situations. The benefits are, of course, visible on the generated test set, which is completely uncorrelated with the images in the training and validation sets. As for the results reported on what has been termed the "Real Test Set" they are significantly worse compared to those obtained on the generated test set. This should not be interpreted as a failure of the system to generalize to real-world scenarios, which would be highly concerning. In reality, for the sake of completeness of the proposed work, it was deemed necessary to conduct a test on real images. However, the difficulty of obtaining these samples has been extensively discussed earlier, so the test was conducted on a very small number of images, specifically 17 images, and therefore 54 patches, with scenarios that are also extremely different from the framework in which we have situated ourselves, with a camera on board a vehicle. In many cases, the errors can be attributed to incorrect segmentation of the tracks, causing patches containing obstacles not to be effectively included among the generated patches. Therefore, among the experiments conducted with ResNet50 as the backbone, the best results on both test sets were obtained by including the augmentation procedure, making the entire network trainable, and using a batch size of 64, which probably proved to be a good trade-off as discussed earlier. Once this winning configuration was identified, it was then replicated by solely replacing the backbone. As seen, neither of the two backbones actually achieved any improvement; on the contrary, there is a visible performance decline. However, below are other comparison metrics used to analyze the three models obtained.

In particular, regarding aspects related to the performances of these three models, a comparison was made not only based on the previously reported results, namely the F1-Score on the two tests, but also on the results obtained on the entire image, which we remind is the most pragmatic metric defined above. Since the test with two networks of different sizes was conducted to compare both the performance benefits gained in terms of an increase in the number of parameters and to potentially visualize the level of performance degradation in relation to the decrease in inference time, the comparison below includes both the number of parameters for each of these architectures and the mean with the standard deviation obtained by performing inference on a sufficiently large number of samples, namely those contained in the generated test set. Obviously, since the purpose was purely comparative, the time taken into consideration is solely the inference time taken by the network, that is, given the input, the time required to obtain the output, with the same input size fixed.

As seen from the above table, the model that utilized ResNet50 can be considered the best in terms of performance on both tests. It has a reasonable number of parameters, and the inference time compared to the smaller backbone is not excessively higher. At the same time, the model that employed ResNet101, despite having a higher number of parameters and longer inference time, did not actually show any improvement in performance, probably because the dataset is quite small and so a bigger model is more prone to overfitting.

A note must be made regarding the fact that MobileNetV2 achieves better results in the image-based evaluation on the Generated TestSet. However, this is not sufficient to prefer this model over the one based on ResNet50, as the performances recorded on the patches show an unacceptable degradation. This is a critical consideration since the primary metric to consider is the performance on patches, given that our classifier was trained on patches rather than entire images.

Backbone	Parameters	Inference Time (microseconds)	Real TestSet Results	Generated TestSet Results
MobileNetv2	3504872	mean: 4118.0136 std: 2275	Evaluation on **patches**: TP:17 TN:21 FP:1 FN:15 precision: 0.9444 recall: 0.5313 **F1-score: 0.68** Evaluation on **images**: TP:10 TN:0 FP:1 FN:6 precision: 0.9100 recall: 0.625 **F1-score: 0.7410**	Evaluation on **patches**: TP:618 TN:1391 FP:127 FN:139 precision: 0.8295 recall: 0.8164 **F1-score: 0.8229** Evaluation on **images**: TP:485 TN:22 FP:7 FN:36 precision: 0.9857 recall: 0.9309 **F1-score: 0.9576**
ResNet50	25557032	mean: 14939.0365 std: 4797.8760	Evaluation on **patches**: TP:21 TN:20 FP:2 FN:11 precision: 0.9130 recall: 0.6563 **F1-score: 0.7636** Evaluation on **images**: TP:13 TN:0 FP:1 FN:3 precision: 0.9286 recall: 0.8125 **F1-score: 0.8667**	Evaluation on **patches**: TP:641 TN:1450 FP:68 FN:116 precision: 0.9041 recall: 0.8468 **F1-score: 0.8745** Evaluation on **images**: TP:485 TN:0 FP:28 FN:37 precision: 0.9454 recall: 0.9291 **F1-score: 0.9372**
ResNet101	44549160	mean: 24765.6154 std: 7241.5714	Evaluation on **patches**: TP:12 TN:21 FP:1 FN:20 precision: 0.9231 recall: 0.375 **F1-score: 0.5333** Evaluation on **images**: TP:9 TN:0 FP:0 FN:8 precision: 1 recall: 0.5294 **F1-score: 0.6923**	Evaluation on **patches**: TP:638 TN:1429 FP:89 FN:119 precision: 0.8776 recall: 0.8428 **F1-score: 0.8598** Evaluation on **images**: TP:486 TN:0 FP:27 FN:37 precision: 0.9474 recall: 0.9293 **F1-score: 0.9382**

Table 4.7: Comparison between various backbones. The evaluation on patches have been made on 2275 or 54 samples, respectively for Generated TestSet and for Real TestSet, while the evaluation on images have been made on 550 or 17 samples, respectively for Generated TestSet and for Real TestSet.

4.3 Significance of the results obtained on the best model

Within this section, we will conduct a more detailed analysis of the results obtained with the selected best model. Indeed, the numerical results obtained are certainly indicative of the functioning of the proposed model, but a more critical analysis necessitates an examination of the errors made by the system in order to identify potential improvements for future developments. We will also assess the system's potential for generalization. Specifically, it has been described how the use of patches effectively frees the binary classifier from recognizing the rock as such, leading it instead towards the possibility of recognizing the obstacle as an interruption of the track. This is because the patches only see a small part of the rock and therefore cannot recognize the edges or the shape, but at most the texture. We will analyze whether this level of generalization compared to other possible obstacles has already been achieved or if, in future developments, fine-tuning the binary classifier on other types of obstacles should be considered.

4.3.1 Analysis and classification of errors

Considering the importance of a visual analysis of errors in each of the tests, in order to effectively identify the strengths and limitations of the implemented system, after performing inference on the tests with the best model obtained, a manual inspection of the results was carried out. The outcomes were classified into some identified macro-categories, and the recurrence of each class was highlighted in relation to the total number of errors committed.

In addition to the normal distinction between false positives and false negatives, we proceeded to identify more specific classes that could guide towards a possible solution to the identified problem. The following table contains all the types of errors identified, with different color-coding depending on whether that specific error can be attributed to a false positive or a false negative.

Segmentation Failure	Too small part rock in the image	Confusion with other track-related items	Unrecognized but clearly visible rock	Confusion with the front part of the train	Very small rock in the distance

Table 4.8: Types of errors considered.

Specifically, errors attributable to false negatives are:

- Segmentation Failure: this refers to the fact that often the segmentation network, unchanged within this thesis work, does not comprehend some parts of the track where the rock is present. This results in the fact that no patches containing the rock are actually created, making the classification not incorrect but entirely non-existent.
- Too Small Rock Part in the Image: this refers to the fact that, when dividing into patches, very often small contributions resembling very small rocks appear within the patch, making their detection extremely difficult.
- Unrecognized but Clearly Visible Rock: this refers to errors where a sufficiently large part of the rock is present in the patch but is not correctly recognized. These are concrete errors of the implemented classifier.
- Very Small Rock in the Distance: this refers to the presence of rocks or parts of them that are extremely small and located in the distance. This means that the extracted patches are also very small and likely to be subject to blurring, given the need to resize the input image.

Errors attributable to false positives are:

- Confusion with other track-related items: this refers to the potential presence of structural objects on the tracks themselves that are mistakenly confused with rocks.
- Confusion with the front part of the train: this are related to the fact that the segmentation network often incorporates the structural metallic parts characteristic of the train's front into the generated mask. These are perceived as track obstructions and consequently treated as obstacles by the classifier.

Error analysis was conducted for both test sets, and in particular this analyisis is useful for the Real TestSet in order to support the previously described justifications. The error analysis has been performed on both, both on errors evaluated at the patch level and those at the image level, as many error classes could significantly be affected by such variation.

Analysis and classification of errors on Generated Test Set

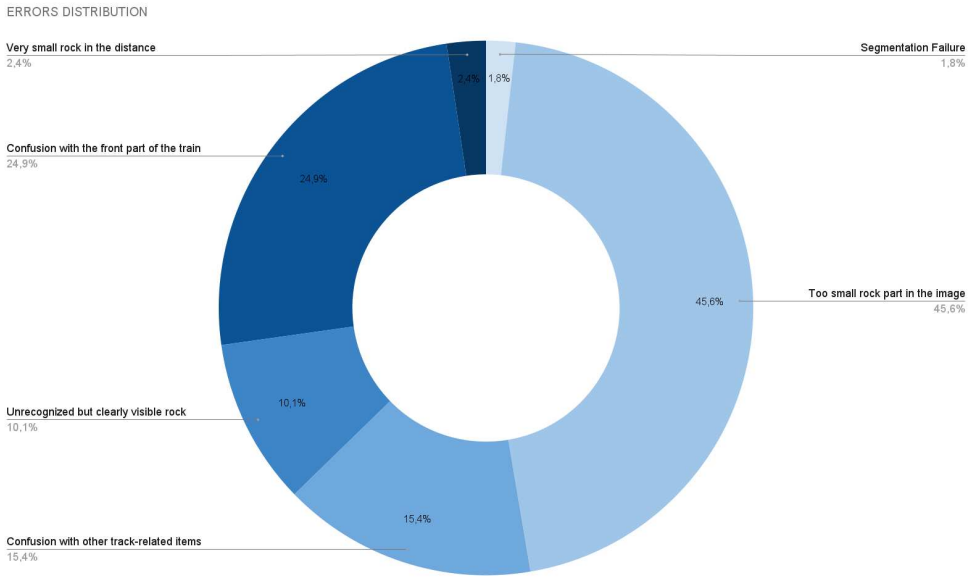

Figure 4.2: Distribution of the patches errors on the Generated Test Set.

As can be seen from the above graph, the higher percentage of errors is primarily driven by the presence of extremely small portions of the rock within the image, which tend to create confusion in the classifier. Obviously, it is clear that, as mentioned in the definition of the image metric, what is done on the individual patch is not crucial in the application context. Instead, it is more important that if there is a rock in the image, it is detected in at least one of the patches that compose it. Therefore, even if the rock is not very visible in one patch and causes the classifier to fail, there would certainly be patches where the rock portion is larger, ensuring the recognition of the danger.

The second highest error percentage is instead dictated by confusion with the front part of the train. In this regard, it is necessary to specify that the segmentation

network had been trained on RailSem19 [19], while the Test Set was generated from FRSign [59] where the visualizations, although still frontal, included the front part of the train. The segmentation network was probably not well-prepared for this scenario.

As for the other two lower percentages, they can be attributed either to confusion with other parts of the track or to actual misclassifications. To mitigate such issues and prevent them from leading to false positives or false negatives, a decision rule based on majority voting could be proposed, for example, on the outputs obtained from various patches. However, this would be more of a palliative measure for a problem that, to be definitively resolved, would require an expansion of the training samples in terms of variability. Seeing track features in various situations, both within patches classified as positive or negative, could help the network avoid errors.

Regarding the lower percentages, they are attributable to segmentation errors that do not correctly create the patch or the presence of rocks in the distant background. As for the first issue, it could be easily addressed by retraining the segmentation network. Regarding the second issue, it would certainly require an expansion of the dataset to allow the train to have the proper braking time.

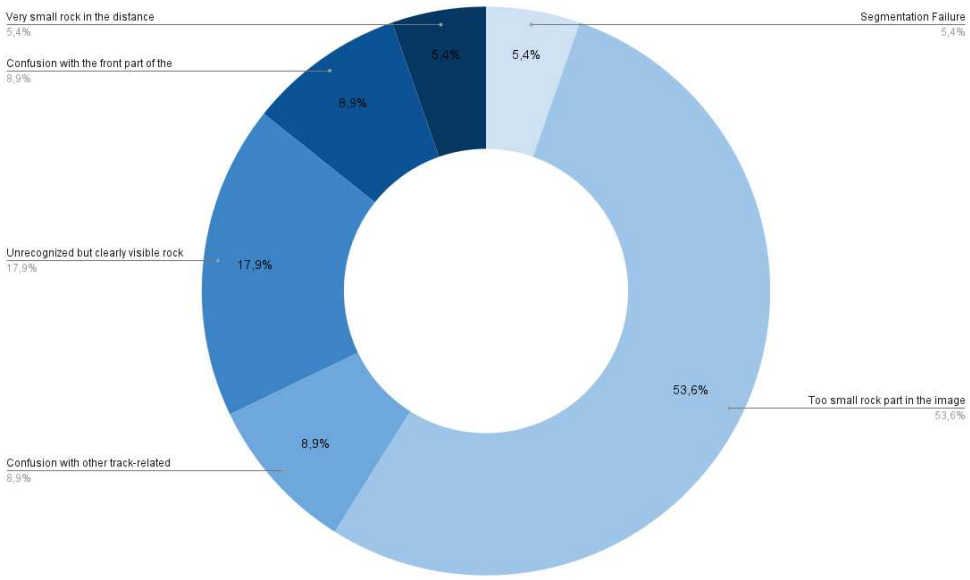

Figure 4.3: Distribution of the images errors on the Generated Test Set.

As it is possible to see, the percentages remain approximately constant, but

of course, they need to be analyzed from a relative perspective. From a numerical standpoint, however, conducting an analysis on the images results in a significant reduction of errors. For instance, considering the most frequent ones, namely *too small rock part in the image* only a few remain. Specifically, all those characterized by having a larger portion of the rock in other patches have been eliminated in this way. However, situations still persist where the part of the rock included in the segmentation is too small, preventing the classifier from identifying it correctly.

Analysis and classification of errors on Real Test Set

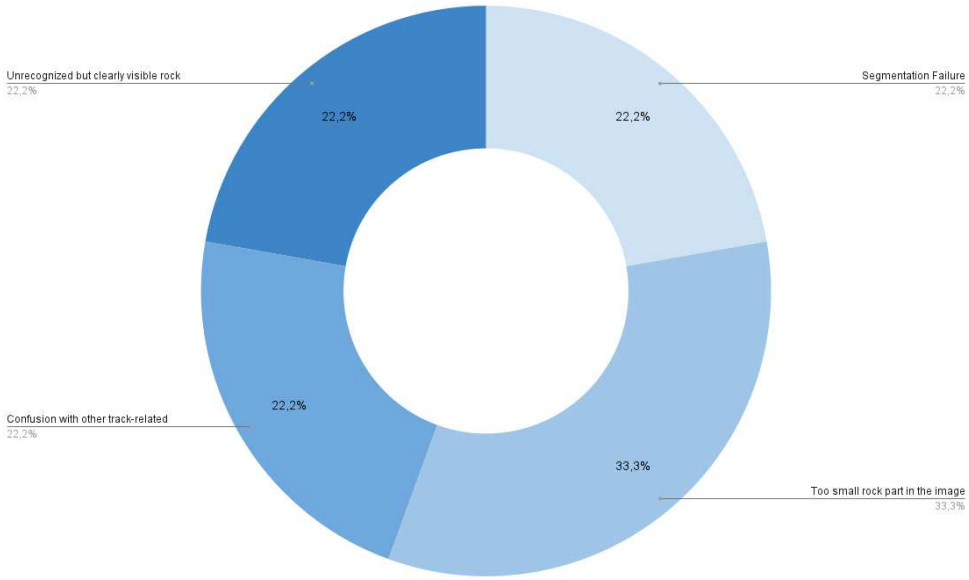

Figure 4.4: Distribution of the patches errors on the Real Test Set.

As you can see in the above graph, some errors do not appear at all, such as a very small rock in the distance and confusion with the front part of the train. This is because these situations are not present in the test, whose limited quantity and inadequacy of images have already been highlighted. Nevertheless, the considerations previously made for the errors encountered in the generated test set remain valid and also the proposed mitigations.

In this case as well, it's essential to consider that the visualization is done in percentage terms, so in relative terms. However, there is a clear decrease in errors, to the extent that the only ones that still remain are those related to poor segmentation

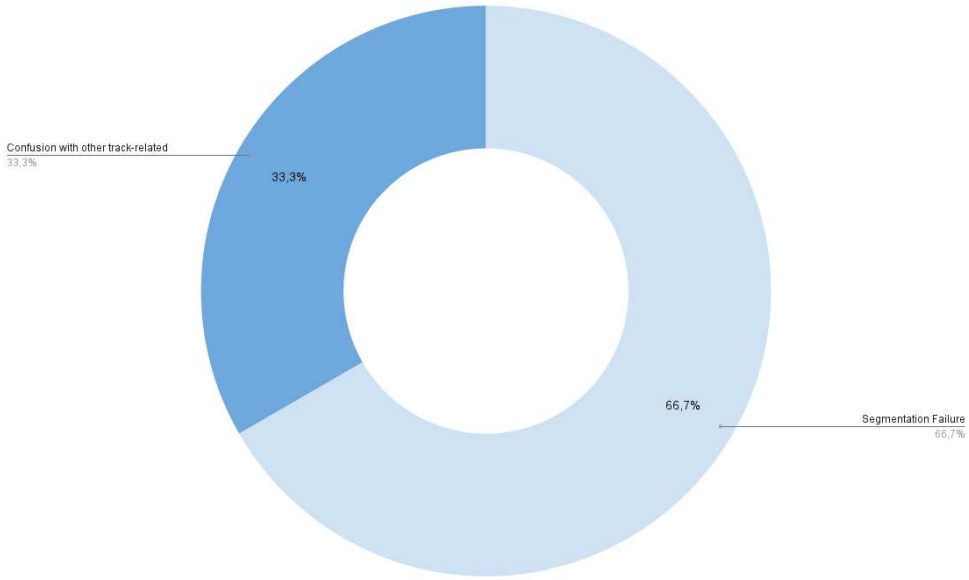

Figure 4.5: Distribution of the images errors on the Real Test Set.

or confusion with possible other elements of the railway structure. For these errors, possible mitigations have already been proposed previously.

4.3.2 Generalization capability

Also previously, we mentioned that a potential strength of the proposed approach could lie in its generalization capability. Specifically, by training the classifier not on entire images but only on patches, it should have acquired a discrimination ability that allows it to identify the boulder not because it recognizes it, but simply as an object obstructing the normal structure of the track. It is reasonable to think that the classifier may not be able to recognize the boulder as such because it has seen the entire edges only a very limited number of times. Therefore, an additional experiment we wanted to conduct was to test the system's ability to recognize other obstacles, different from boulders, present on the tracks, even though it was not trained for it. This experiment was primarily subjected to qualitative analysis, meaning that starting from a sequence of frames, the results obtained by the system were visually analyzed. It is clear that the system cannot be considered fully functional in these cases, but there is certainly a good foundation. Therefore, by perhaps retraining from the currently available weights, further fine-tuning on images containing other types of obstacles could lead to very

satisfactory results. In any case, this experimentation attests to the versatility of the employed approach.

4.3.3 Possible future improvements

From the error analysis, it is clear that the main avenues that would need to be explored for improving performance are primarily two. One should certainly be the retraining of the segmentation network, not included in the objective of this thesis, by incorporating images of tracks with rocks placed as obstacles within the dataset, while keeping the ground truth segmentation mask unchanged. This way, it would ensure that the presence of rocks does not disrupt the segmentation mask. Of course, this would also require an increase in the number of images where rocks are visible but not on the tracks, and in these cases, the segmentation mask should obviously not include them. This would avoid the opposing problem where rocks not present on the tracks would also be segmented, posing a very high risk of false positives.

The second avenue should be the expansion of the dataset generated to increase the number of samples and their variability, also in terms of possible railway scenarios and related structures. This expansion would certainly ensure greater accuracy of the discriminator. The enrichment of samples should also be carried out on images containing rocks in the distance. Although these cases are already present in the generated dataset, they require special attention because, practically speaking, while it is true that in the sequence of subsequent frames and thus as the train approaches, the obstacle will undoubtedly be recognized, the vehicle also has a significant braking time. Therefore, it is necessary for the alarm diagnosis to be generated in a timely manner. Obviously, the distance and small dimensions create an indistinguishable background that complicates the classifier's work, which must be trained on this aspect.

Another extremely important aspect that makes sense to focus on is the possibility of tuning the system. Currently, the development and analysis of the system have been carried out considering false positives and false negatives at an equal level of severity. In many binary classification situations, the two errors actually have extremely different impacts, and this is also the case here. Specifically,

false positives could result in the inability of the train on which the system is installed to travel efficiently, perhaps experiencing continuous slowdowns. At the same time, while this is only a prerequisite for efficiency, false negatives compromise people's lives and potentially damage the railway infrastructure in an irreparable way. By adjusting the operating threshold of the system, it could be possible to limit the presence of one type of error over the other, as was also discussed during the metrics analysis.

Finally, an additional mitigation to address the presence of false positives or negatives could involve establishing a rule for combining the outputs obtained on various patches, treating them as if they were outputs of a multi-task system. Conditions for triggering an alarm could then be defined based on a possible majority rule among the various outputs, also considering patch dimension.

An additional improvement to the established system could be achieved by including the obstacle's distance parameter from the train or possibly identifying the positioning of the rock in the image. For example, determining if it is located on the track where the analyzed vehicle is or on another vehicle. This could be done by extracting such information in relation to the placement of the patch in the original image and conducting subsequent analyses, even if purely geometric in nature.

Chapter 5

Conclusion

The importance of the railway system in the economic and service sectors for humans has always been unquestionable. Given the countless sectors and people relying on it, ensuring its safety is crucial, especially with the anticipation of a growing prevalence of high-speed trains. Enabling efficient automation of the guidance systems for such vehicles, alongside human operations, would undoubtedly enhance safety levels. To achieve this, it is necessary to identify potential obstacles that could disrupt the normal flow of the train, facilitating their removal or, at the very least, improving traditional track inspection methods prone to human errors that can result in loss of lives and irreversible catastrophes.

This thesis aimed to address this need by recognizing the necessity of a system as described and analyzing research gaps in certain aspects of this field. Consequently, a real-time video analysis system was developed to detect the presence of rocks on the tracks. To meet this objective, a synthetic dataset was created, and an innovative architecture suitable for the examined system was implemented. The architecture consists of several modules: a segmentation module focused on identifying the region of interest represented by the railway tracks, a processing module that extracts patches from the segmentation mask, and a final module that performs binary classification on each of these patches. As detailed earlier, the innovation in this work lies in both the dataset generation methodology and the proposed architectural solution, which, by deviating from the use of a single deep neural network, offers a more efficient solution than a classic object detector. This solution's efficiency lies not only in practical aspects such as reduced inference time and computational power but also in providing a solution to generalization

problems that would have been encountered with a small dataset. Through the use of patches, it was possible to increase the number of samples and make the network completely independent of positioning and irrelevant background aspects. To achieve such independence with a traditional approach, an extremely larger dataset would have been required.

The obtained results are satisfying and allow for possible improvements with minimal effort. In any case, the current results are highly significant. The system was evaluated on both a real dataset, where despite image incompatibility and limited quantity, the results were good, and on a generated dataset completely unrelated to the training dataset, as the background images were entirely different. Additionally, the number of rocks not detected in any of the patches is extremely low, as testified by an additional metric analyzing per image.

Error analysis reveals that the errors attributable to the system's malfunction are actually very few. The use of patches also equips the developed system with greater generalization capabilities, as demonstrated with other types of obstacles. The system is not recognizing the object per se but rather as an obstruction on the tracks. Generalization results could be further improved by simply fine-tuning and introducing different obstacles into the dataset, leaving the processing procedure completely unchanged. A redefinition of performance could also be achieved by implementing a decision rule between the outputs obtained on various patches to determine when it is reasonable to raise an alarm.

Some of the issues identified in the developed system could undoubtedly be mitigated by improving the segmentation network, as detailed earlier, and by expanding the dataset used. The approach used for dataset creation imposes no limits, so with sufficient time availability, this expansion would certainly be possible.

Regarding potential future developments based on the system created, one consideration is distance estimation. It might be thought that not using a detector makes the system lack the contribution of obstacle positioning in the scene. However, this contribution could still be obtained using patches, as it is possible to trace back to the position covered in the original image for each of them. Obtaining such a contribution could be crucial for a system of this kind, allowing for decisions, through geometric transformations, on how far the detected obstacle is from the vehicle and

deciding when to start braking in consideration of the vehicle's current speed.

In conclusion, the developed system can be considered satisfactory in terms of the achieved results and the innovation of the proposed method. It also provides numerous opportunities for enrichment and improvement that, due to temporal constraints and available resources, have remained unexplored at this point.

Acknowledgements

A termine di questo mio percorso voglio ringraziare tutti coloro che ne hanno fatto parte. Grazie alla Prof. Alessia Saggese per le attenzioni che mi ha riservato in questi ultimi mesi, per essere sempre stata pronta a darmi utili consigli, per la sua gentilezza, professionalità e competenza, che sono state per me fondamentali. Insieme a lei ringrazio anche il prof. Antonio Greco, altro relatore di questo elaborato di tesi. Grazie ad entrambi quindi per la passione con cui affrontate il vostro lavoro e per la disponibilità destinatami in questi anni.

Grazie ai miei genitori per essermi stati sempre accanto senza giudizi o condizionamenti, per avermi lasciata libera ma mai sola, per avermi insegnato che ogni successo è tale solo se guadagnato con onestà e per avermi insegnato a guardare alle mie debolezze con tenerezza e a riprendere il cammino verso i miei sogni.

Grazie a mio fratello Valerio per la dolcezza e l'amore con cui mi guarda dal primo giorno che ci siamo visti. Grazie per credere in me incodizionatamente.

Grazie quindi alla mia famiglia, per me porto sicuro, spalle forti sui cui poggiarmi quando mi sento barcollante, per me siete amore incondizionato, spero di avervi reso fieri di me.

Grazie ai miei nonni, a chi oggi avrei voluto fosse ancora qui, a festeggiare con orgoglio il mio traguardo.

Grazie a Mattia, compagno di studi, di sogni e di giorni. Grazie per esserci stato nei momenti più belli e più brutti di questi anni, per aver condiviso con me ogni sorriso e ogni sconfitta, ma soprattutto per avermi offerto un posto sicuro nel suo cuore puro.

Grazie ai miei amici Nando e Vito, per aver condiviso con me le ansie prima di ogni esame e le soddisfazioni dopo ogni successo, siamo arrivati ad essere insofferenti uno dell'altro per il troppo tempo passato insieme, come spesso accade tra fratelli. Ma

proprio come fratelli abbiamo imparato a conoscere le forze e le debolezze di ognuno, a conoscere cosa si celasse dietro ogni “fate come volete” e dietro ogni telecamera spenta. Non mi sento di dire che rifarei tutti gli esami solo per rifare progetti con voi, preferisco ricordarli con la nostalgia del passato, ma siete stati il gruppo migliore che potessi desiderare.

Grazie a Mariarosaria, con cui ho condiviso molto di questi ultimi mesi. Grazie perché dopo esserci squadrate dall’alto al basso con sospetto, appena conosciute, abbiamo scoperto la bellezza di guardare insieme gli altri dall’alto verso il basso, anche perché, inutile nasconderlo, le nostre prospettive visuali si completavano a vicenda.

Grazie a Giuseppe e a Francesco, con cui ho condiviso intere giornate di questi ultimi mesi, perché mi avete fatta sentire accolta dal primo momento in cui sono entrata nel laboratorio, e avete reso tutto più magico, siete preziosi.

Grazie a Marco per aver condiviso con me il suo lato più spensierato ed allegro, resta sempre come sei.

Grazie a Roberta per aver rallegrato e addolcito viaggi fatti di stanchezza e di noia.

Grazie ad Antonio, Umberto, Sabatino, Mattia, Antonio, Rosanna, Stefano, Giampaolo, Speranza, Giuseppe, Ernesto e Pierpaolo per aver colorito questi ultimi mesi, siete stati una bellissima scoperta.

Grazie a Gerardo e Marco per aver condiviso con me questi anni di università, vi voglio bene.

Grazie agli amici di sempre Alessia, Alessia, Serena, Vincenzo ed Emiliano, anche se la vita ci porta lontano e ci fa perdere le nostre abitudini, avrete per sempre un posto speciale nel mio cuore.

Grazie quindi a tutti voi, ogni pezzetto di queso traguardo porta il nome di qualcuno di voi.

References

[1] Hui Wang et al. "Transportation Safety Improvements Through Video Analysis: An Application of Obstacles and Collision Detection Applied to Railways and Roads". In: *Transactions on Engineering Technologies*. Ed. by Sio-Iong Ao et al. Singapore: Springer Singapore, 2018, pp. 1–15. ISBN: 978-981-10-7488-2.

[2] Frederic Maire and Abbas Bigdeli. "Obstacle-free range determination for rail track maintenance vehicles". In: *2010 11th International Conference on Control Automation Robotics Vision*. 2010, pp. 2172–2178. DOI: 10.1109/ICARCV.2010.5707923.

[3] Jürgen Wohlfeil. "Vision based rail track and switch recognition for self-localization of trains in a rail network". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 1025–1030. DOI: 10.1109/IVS.2011.5940466.

[4] Fatih Kaleli and Yusuf Sinan Akgul. "Vision-based railroad track extraction using dynamic programming". In: *2009 12th International IEEE Conference on Intelligent Transportation Systems*. 2009, pp. 1–6. DOI: 10.1109/ITSC.2009.5309526.

[5] Zhongli Wang et al. "An inverse projective mapping-based approach for robust rail track extraction". In: *2015 8th International Congress on Image and Signal Processing (CISP)*. 2015, pp. 888–893. DOI: 10.1109/CISP.2015.7408003.

[6] Zhongli Wang et al. "Geometry constraints-based visual rail track extraction". In: *2016 12th World Congress on Intelligent Control and Automation (WCICA)*. 2016, pp. 993–998. DOI: 10.1109/WCICA.2016.7578298.

[7] Michael Gschwandtner, Wolfgang Pree, and Andreas Uhl. "Track Detection for Autonomous Trains". In: *Advances in Visual Computing*. Ed. by George Bebis et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 19–28. ISBN: 978-3-642-17277-9.

[8] M. Alper Selver et al. "Camera based driver support system for rail extraction using 2-D Gabor wavelet decompositions and morphological analysis". In: *2016 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*. 2016, pp. 270–275. DOI: 10.1109/ICIRT.2016.7588744.

[9] M. Alper Selver et al. "Predictive modeling for monocular vision based rail track extraction". In: *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. 2017, pp. 1–6. DOI: 10.1109/CISP-BMEI.2017.8301928.

[10] Zhangyu Wang et al. "Efficient Rail Area Detection Using Convolutional Neural Network". In: *IEEE Access* 6 (2018), pp. 77656–77664. DOI: 10.1109/ACCESS.2018.2883704.

[11] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling*. 2015. arXiv: 1505.07293 [cs.CV].

[12] Yin Wang et al. "RailNet: A Segmentation Network for Railroad Detection". In: *IEEE Access* 7 (2019), pp. 143772–143779. DOI: 10.1109/ACCESS.2019.2945633.

[13] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[14] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[15] Yan Zhang et al. "DFA-UNet: Efficient Railroad Image Segmentation". In: *Applied Sciences* 13.1 (Jan. 2023), p. 662. ISSN: 2076-3417. DOI: 10.3390/app13010662. URL: http://dx.doi.org/10.3390/app13010662.

[16] Tobias Pohlen et al. *Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes*. 2016. arXiv: 1611.08323 [cs.CV].

[17] Mohamed Amine Hadded et al. "Application of Rail Segmentation in the Monitoring of Autonomous Train's Frontal Environment". In: *ICPRAI 2022, International Conference on Pattern Recognition and Artificial Intelligence*. ICPRAI 2022, International Conference on Pattern Recognition and Artificial Intelligence, Paris, FRANCE, 01-/06/2022 - 03/06/2022. Paris, France: Springer, June 2022, pp185–197. DOI: 10.1007/978-3-031-09037-0_16. URL: https://hal.science/hal-03875603.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[19] Oliver Zendel et al. "RailSem19: A Dataset for Semantic Rail Scene Understanding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[20] Changqian Yu et al. *BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation*. 2020. arXiv: 2004.02147 [cs.CV].

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2014. arXiv: 1411.4038 [cs.CV].

[22] Liang-Chieh Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. arXiv: 1606.00915 [cs.CV].

[23] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV].

[24] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[25] Francesco Visin et al. *ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation*. 2016. arXiv: 1511.07053 [cs.CV].

[26] Xiaodan Liang et al. “Semantic Object Parsing with Graph LSTM”. In: *CoRR* abs/1603.07063 (2016). arXiv: 1603.07063. URL: http://arxiv.org/abs/1603.07063.

[27] Liang-Chieh Chen et al. *Attention to Scale: Scale-aware Semantic Image Segmentation*. 2016. arXiv: 1511.03339 [cs.CV].

[28] Arij Zouaoui et al. “RailSet: A Unique Dataset for Railway Anomaly Detection”. In: *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*. Vol. Five. 2022, pp. 1–6. DOI: 10.1109/IPAS55744.2022.10052883.

[29] Hyunsu Kim et al. *Exploiting Spatial Dimensions of Latent in GAN for Real-time Image Editing*. 2021. arXiv: 2104.14754 [cs.CV].

[30] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 2016. arXiv: 1604.01685 [cs.CV].

[31] Gerhard Neuhold et al. “The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes”. In: Oct. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.

[32] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[33] Andreas Geiger et al. “Vision meets robotics: the KITTI dataset”. In: *The International Journal of Robotics Research* 32 (Sept. 2013), pp. 1231–1237. DOI: 10.1177/0278364913491297.

[34] M Ukai. “Obstacle detection with a sequence of ultra telephoto camera images”. In: *WIT Transactions on The Built Environment* 74 (2004).

[35] L. A. Fonseca Rodriguez, J. A. Uribe, and J. F. Vargas Bonilla. “Obstacle detection over rails using hough transform”. In: *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*. 2012, pp. 317–322. DOI: 10.1109/STSIVA.2012.6340602.

[36] Jonny A. Uribe, Luis Fonseca, and J. F. Vargas. “Video based system for railroad collision warning”. In: *2012 IEEE International Carnahan Conference on Security Technology (ICCST)*. 2012, pp. 280–285. DOI: 10.1109/CCST.2012.6393573.

[37] Hiroki Mukojima et al. “Moving camera background-subtraction for obstacle detection on railway tracks”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 3967–3971. DOI: 10.1109/ICIP.2016.7533104.

[38] Ryuta NAKASONE et al. “¡b¿Frontal Obstacle Detection Using Background Subtraction and Frame Registration¡/b¿”. In: *Quarterly Report of RTRI* 58.4 (2017), pp. 298–302. DOI: 10.2219/rtriqr.58.4_298.

[39] J. Vazquez et al. “Detection of moving objects in railway using vision”. In: *IEEE Intelligent Vehicles Symposium, 2004*. 2004, pp. 872–875. DOI: 10.1109/IVS.2004.1336499.

[40] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[41] Mingyang Yu, Peng Yang, and Sen Wei. “Railway obstacle detection algorithm using neural network”. In: *AIP Conference Proceedings* 1967.1 (May 2018), p. 040017. ISSN: 0094-243X. DOI: 10.1063/1.5039091. eprint: https://pubs.aip.org/aip/acp/article-pdf/doi/10.1063/1.5039091/14160466/040017_1_online.pdf. URL: https://doi.org/10.1063/1.5039091.

[42] Rajiv Kapoor, Rohini Goel, and Avinash Sharma. “Deep learning based object and railway track recognition using train mounted thermal imaging system”. In: *Journal of Computational and Theoretical Nanoscience* 17.11 (2020), pp. 5062–5071.

[43] Deqiang He et al. “Urban rail transit obstacle detection based on Improved R-CNN”. In: *Measurement* 196 (2022), p. 111277. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2022.111277. URL: https://www.sciencedirect.com/science/article/pii/S0263224122005206.

[44] Tao Ye et al. “Automatic Railway Traffic Object Detection System Using Feature Fusion Refine Neural Network under Shunting Mode”. In: *Sensors* 18 (June 2018), p. 1916. DOI: 10.3390/s18061916.

[45] Tao Ye et al. “Railway Traffic Object Detection Using Differential Feature Fusion Convolution Neural Network”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.3 (2021), pp. 1375–1387. DOI: 10.1109/TITS.2020.2969993.

[46] Tao Ye et al. “Autonomous Railway Traffic Object Detection Using Feature-Enhanced Single-Shot Detector”. In: *IEEE Access* 8 (2020), pp. 145182–145193. DOI: 10.1109/ACCESS.2020.3015251.

[47] Tao Kong et al. “RON: Reverse Connection with Objectness Prior Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5244–5252. DOI: 10.1109/CVPR.2017.557.

[48] Juan Li, Fuqiang Zhou, and Tao Ye. “Real-World Railway Traffic Detection Based on Faster Better Network”. In: *IEEE Access* 6 (2018), pp. 68730–68739. DOI: 10.1109/ACCESS.2018.2879270.

[49] Yuchuan Xu et al. “Real-time Obstacle Detection Over Rails Using Deep Convolutional Neural Network”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 1007–1012. DOI: 10.1109/ITSC.2019.8917091.

[50] Samira Badrloo et al. “Image-Based Obstacle Detection Methods for the Safe Navigation of Unmanned Vehicles: A Review”. In: *Remote Sensing* 14 (Aug. 2022), p. 3824. DOI: 10.3390/rs14153824.

[51] Danijela Ristić-Durrant et al. “Artificial intelligence for obstacle detection in railways: Project smart and beyond”. In: *Dependable Computing-EDCC 2020 Workshops: AI4RAILS, DREAMS, DSOGRI, SERENE 2020, Munich, Germany, September 7, 2020, Proceedings 16*. Springer. 2020, pp. 44–55.

[52] Weixun Chen, Siming Meng, and Yuelong Jiang. “Foreign object detection in railway images based on an efficient two-stage convolutional neural network”. In: *Computational Intelligence and Neuroscience* 2022 (2022).

[53] Ling Guan et al. “A Lightweight Framework for Obstacle Detection in the Railway Image Based on Fast Region Proposal and Improved YOLO-Tiny Network”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–16. DOI: 10.1109/TIM.2022.3150584.

[54] Omkar Vivek Sabnis and R Lokeshkumar. “A novel object detection system for improving safety at unmanned railway crossings”. In: *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*. Vol. 1. IEEE. 2019, pp. 149–152.

[55] Muhammad Asad Bilal Fayyaz and Christopher Johnson. “Object detection at level crossing using deep learning”. In: *Micromachines* 11.12 (2020), p. 1055.

[56] Ke Gong et al. “Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 932–940.

[57] Guilherme Kano, Tiago Andrade, and Alexandra Moutinho. “Automatic detection of obstacles in railway tracks using monocular camera”. In: *International Conference on Computer Vision Systems*. Springer. 2019, pp. 284–294.

[58] Mauro José Pappaterra et al. “A Systematic Review of Artificial Intelligence Public Datasets for Railway Applications”. In: *Infrastructures* 6.10 (2021). ISSN: 2412-3811. DOI: 10.3390/infrastructures6100136. URL: https://www.mdpi.com/2412-3811/6/10/136.

[59] Jeanine Harb et al. *FRSign: A Large-Scale Traffic Light Dataset for Autonomous Trains*. 2020. arXiv: 2002.05665 [cs.CY].

[60] Guilherme Kaname Reis Kano. *Automatic Detection of Railway Track Obstacles using a Monocular Camera*. 2020. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997260777/dissertacao.pdf.

[61] LI Yundong et al. “Multi-block SSD based on small object detection for UAV railway scene surveillance”. In: *Chinese Journal of Aeronautics* 33.6 (2020), pp. 1747–1755.

List of Figures

List of Tables