UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Data Mining

Deep Learning for Audio Detection and Video Analysis in Railway Applications

Anno Accademico 2019/2020

Relatori
Ch.mo prof. Valeria Vittorini
Ch.mo prof. Carlo Sansone
Correlatore
Ing. Stefano Marrone

Candidato
Lorenzo De Donato
matr. M63000732

Abstract

In recent years, Artificial Intelligence (AI) is becoming increasingly present in the daily lives of individual citizens. So much so that, in many areas there is an increasing effort to invest in research in order to create and integrate "intelligent systems" within production processes, services, inspections, etc. In this thesis, we will analyse only a small part of the immense suite of tools and disciplines that Artificial Intelligence encompasses; in particular, we will focus mainly on some aspects of Deep Learning.

The work described in this Thesis has been conducted at the Department of Electrical Engineering and Information Technologies of the University of Naples Federico II, in collaboration with the research group currently working at the H2020 Shif2Rail project RAILS (Roadmaps for AI integration in the raiL Sector) and a railway company; as such the thesis addresses topics related to the application of Deep Learning to railways, and specifically to maintenance and defect detection. The purpose of this work is twofold: a) to assess the status of research on the usage of Deep Learning techniques in railway applications for maintenance purposes, with a specific focus on video analysis and audio detection, and b) to investigate the application of Deep Learning to a real world monitoring and maintenance scenario. At this aim, we have laid the groudwork to build a robust audio alarm detection system at level crossings.

Hence, after a brief introduction to Machine Learning and Deep Learning (Chapter 1), the Thesis presents a Systematic Literature Review on Deep Learn-

ing for video analysis and audio detection in railways (Chapter 2) and proposes a modular system to monitor and evaluate the health status of level crossings; as a first step, a deep neural network is implemented that is able to discern between audios related to the level crossing alarm and audios related to similar sounds coming from the surrounding environment (Chapter 3). The choice to leverage only on video and audio data is due to the necessity of using non-invasive sensors so as not to interfere with the behaviour of a safety critical system in operation.

Contents

Chapter 1

What is Deep Learning?

Artificial Intelligence (AI) is not a new concept, indeed its birth is dated around the middle of the twentieth century. From a theoretical perspective, AI encompasses those algorithms, methods and procedures oriented to create and characterize *intelligent* computer systems. Conceptually speaking, intelligence refers to the ability of an agent (e.g. a human being) to learn, understand, reason, plan, solve problems, etc. In the context of the Artificial domain, the most common definition of intelligence is based on the ability of an agent to pass the "the imitation game" (also known as Turing test), a test proposed by Alan Turing in his article entitled "Computing machinery and intelligence" (1950) [1]: a machine is deemed intelligent if it is indistinguishable from a human during a conversation with an impartial observer. Nowadays, different definitions of AI can be found in the literature, however, taking into consideration some of them [2, 3, 4] and the Turing's concept just presented, we can outline the following meaning: AI refers to those artificial agents capable to analyze the environment and take actions, emulating the human reasoning process, in order to achieve a specific goal. Although this is only an interpretation, it is a sufficient start point to understand which correlations bind Artificial Intelligence, Machine Learning and Deep Learning.

Machine Learning is only a particular form of AI that includes the ability to

learn from examples and improve with experience, without being explicitly programmed. When provided with sufficient data, a ML algorithm can learn to make predictions or solve more general problems. Systems supported by ML have the capability to improve the performance in specific tasks, based on previous experience or on provided data used to train the machine itself. The interest in ML comes from the fact that ML applications allow performing tasks such as classification, object detection, landmark localization, and so on, better than expert humans operators. In Tom Mitchell's Machine Learning [5], a more formal definition of ML is given: *"Machine Learning is the study of computer algorithms that improve automatically through experience"*. However, under the Machine Learning flag, there are several algorithms and models each of which has different characteristics and is able to solve determinate tasks. Among them, Artificial Neural Networks (ANNs) have the peculiarity of being inspired by the structure of the human brain; in particular, an ANN has a layered structure composed of various interconnected neurons (also called perceptrons) which interact with each other to perform the desired task. In recent years, the technological evolution has made it possible to obtain increasingly complex networks, leading to the statement of the so-called "Deep Neural Networks", a particular subset of ANNs characterized by a very deep structure (with several layers) and the ability to automatically extract features from data. Formally, *Deep Learning* "allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction" [6].

Fig. 1.1 shows connection described above between Artificial Intelligence, Machine Learning, Artificial Neural Networks and Deep Learning.

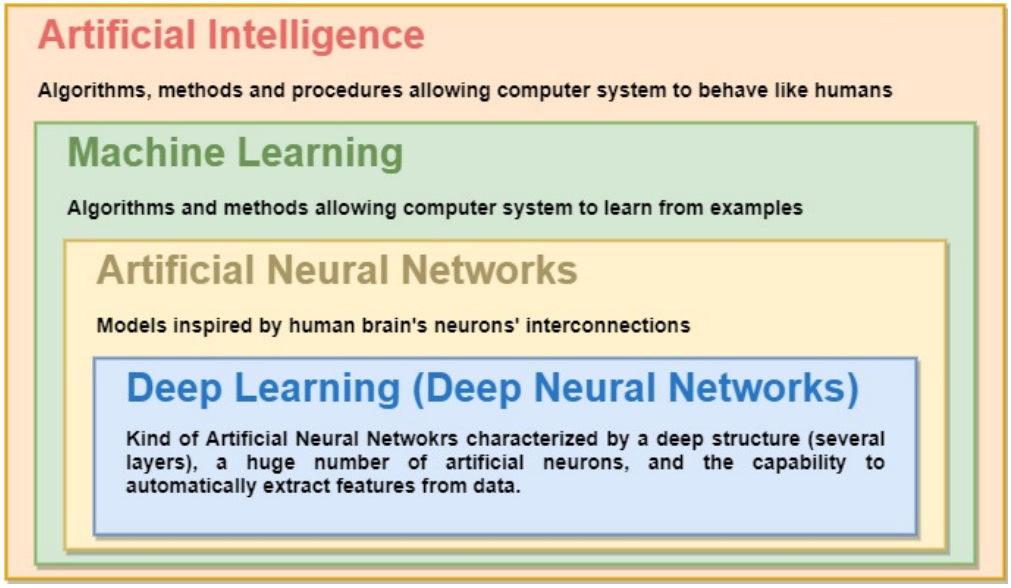


Figure 1.1: Deep Learning in the AI Field

1.1 From Machine Learning to Deep Learning

As discussed so far, Machine Learning is only a particular form of AI. It encompasses all the algorithms and models which allow artificial systems to learn from examples and the previous experience, where "learn" stands for *extracting knowledge*. How algorithms do that depends on the specific Learning Paradigm which, in general, can be classified in: i) *Supervised Learning*: leveraging on labelled data, the system should learn how to act correctly in unknown situations [7]; ii) *Unsupervised Learning*: this paradigm deals with finding hidden structures among unlabelled data [7]; iii) *Semi-supervised Learning*: in this case, the concept is to solve typical supervised tasks (e.g. classification) relying on not fully labelled dataset, using clustering approaches, allowing the possibility to leverage on huge datasets reducing the labelling costs [8]; iv) *Reinforcement Learning*: it may seem a kind of unsupervised learning since it not relies on examples of correct behaviour, however, it aims to find a suitable action through a trial and error process, based on the observations gathered from the interaction with the environment, that would maximize a reword (a particular function) [7, 9].

From another point of view, ML algorithms and techniques can be classified in relation to the tasks they are able to solve, such as: i) *Classification*, the task of building a model capable to automatically classify new instances attributing them the correct label; ii) *Regression*, differently from classification task, here, the model fits the data points in order to predict a numeric value for the new instances; iii) *Clustering*, the problem to split the instances into natural groups, samples in the same group should be characterized by a high similarity, otherwise, samples belonging to different groups will not; iv) *Dimensionality reduction*, the set of techniques aiming to reduce the dataset size, in terms of attributes (e.g. feature selection, principal component analysis) or in terms of samples (e.g. sampling), to obtain better model's performances and memory benefits.

The latter point leads us to wonder whether a dataset with many attributes (or features) can always bring benefits. In theory, it could be true, we will have more information from which extract knowledge; in practice, it is not always so. For example, correlated attributes, i.e. those features which trends or values are related in some way, often bring the same information to the model; in many cases, considering both of them leads to an increasing in complexity and bad performances [8]. Another example resides in irrelevant (non-discriminant) attributes, i.e. features which distribution is roughly constant so they are not discriminant in the sense that it is hard to make decisions relying on them. It is as these attributes distract the model. Moreover, "the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables" [10]; this phenomenon is known as the *curse of dimensionality* and, in the case of datasets, it means that it is very likely that if the number of features grows a lot, whatever they are significant or not, more samples are needed in order to well characterize a ML model. That is why, in almost all the cases, when using an ML-based approach, a good feature selection/extraction is the key to success. Such extraction can be performed manually, through some

ML-models or through specific techniques. Whichever mode is chosen, this always remains a highly delicate engineering process that must be performed accurately in collaboration with domain experts. Since it is an engineering choice, it must be programmed and evaluated by humans first, and this is one of the most troubling tasks beyond the choice of the ML model. However, as we will see in the next sections, such a problem will be automatically addressed by Deep Neural Networks.

Machine Learning encompasses a really huge number of algorithms and models, each of which, as already mentioned, can solve one or more tasks. The algorithms can be subdivided in multiple categories, among which we can cite: *Support Vector Machines (SVM)* (e.g. Non-Linear SVM and Support Vector Regression); *Tree-based Algorithms* (e.g. C4.5, C5.0 and Model Trees); *Ensemble Classifiers* (e.g. Random Forest, Rotation Forest and Additive Logistic Regression); *Bayesian Algorithms* (e.g. Naive Bayes, Bayesian Networks); *Association Rule Algorithms* (e.g. Apriori and FP-growth); *Polynomial Algorithms* (e.g. Linear Regression and Linear Discriminant Analysis); *Clustering Algorithms* (e.g. K-means, X-means and hierarchical/incremental clustering approaches); *Artificial Neural Networks* (e.g. Multilayer Perceptron and Deep Neural Networks); and so on. Detailed analysis and explanations concerning these algorithms can be found in [9] and [8]. Nevertheless, it is out of the scope of this thesis to make a complete survey of all the these ML techniques; therefore, in the following, we will introduce only Artificial Neural Networks (ANNs).

1.1.1 Artificial Neural Networks

An Artificial Neural Network is characterized by a structure, widely inspired to the human brain, in which the basic elements (neurons, also called *perceptrons*) are organised in layers and interact with each other to perform the desired task.

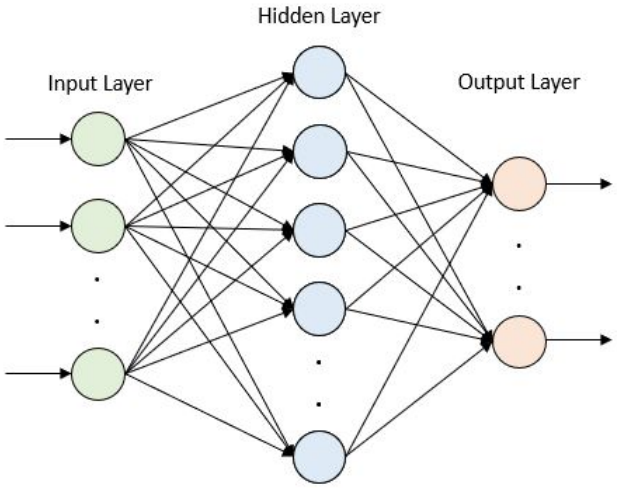


Figure 1.2: Artificial Neural Network Structure

The "simplest" form of perceptron, i.e. the *Linear Perceptron*, is able to perform linear classification. Assuming to have *linearly separable* data instances, belonging to two different classes, the *perceptron learning rule* iteratively find the separating hyperplane, i.e. the plane in the space of the features that separates the samples belonging to the two classes [8]. Actually, from a conceptual perspective, a neuron applies an *activation function* to its own inputs in order to produce an output; such function could be linear, as in the case of Linear Perceptron, or non-linear (e.g. a sigmoid). Anyway, combining more than one perceptron in a "hierarchical" structure, thus an Artificial Neural Network, it is possible to perform non-linear classification, both if the activation function is linear or not. As shown in Figure 1.2, such networks are composed of three kinds of layers: i) the input layer, that has as many neurons as there are attributes of the input dataset ii) the output layer, that has as many neurons as there are classes of the problem, and iii) an arbitrary number of hidden layers.

When it comes to Neural Networks, also considering the third point above, the learning process encompasses both the network's structure and the connection weights. In the following, we will focus on the weights' learning process, considering a fixed network structure (typically defined by a "trial and error" experimental

process). In the case of perceptron learning rule, weights were updated leveraging on the neuron's output. Nevertheless, since it is not possible to evaluate the outcome of the hidden units, it is necessary to adopt another solution. The concept is to leverage on the contribution that each neuron gives to the final prediction applying the *gradient descent* algorithm. It is an optimization procedure that aims to iteratively adjust a function's parameter, multiplying the function's derivative by a small constant (*learning rate*) and subtracting the results from the current parameter value; the process is repeated until the minimum is reached [8]. Actually, the learning rate can vary iteration by iteration and defines how quickly the process "should" converge; a small learning rate leads to a very slow learning process, otherwise, a high one, should induce oscillations. Anyhow, as mentioned, the gradient descent algorithm relies on the function's derivative, then, the function to optimize must be derivable; generally, in the case of ANNs, the squared-error loss function is considered. To find and optimize the weights of the whole network, the derivative of the loss function must be determined with respect to each weight. Without going deeper with details, the error on the output layer is propagated backwards in the network considering the *chain rule* and weights are updated according to its derivative and the learning rate; it is worth noting that, during the process, also the derivatives of the activation functions will be involved, then, they must be derivable too. The algorithm just described, given the error propagation mechanism, is called *Backpropagation*.

Regarding what has just been described, there is a quite important concept to discuss: the *Normalization* of the input data. Typically, data comes with features that have different scales and this could affect the training process. Since the weights' update is proportional to the derivative of the activation function in a given point which, in turn, is proportional to the feature in input, the higher feature will lead to higher weights' variation. Therefore, it could happen that

some weights can update "faster" than others. To avoid that, and then obtain similar variations in all the direction of the feature space, the normalization could help to pare features in order to obtain more or less the same scale.

Actually, there is another factor that could affect the training process, which concerns the order of samples. Indeed, considering two classes, if all the samples of one class are processed first, the network could "polarize". This means that, when the network begins to process the samples of the other class, some performances' oscillations could be registered. In order to avoid that, samples should be *shuffled* before to start the training.

Lastly, we should also consider what happens if a network is composed of a lot of neurons or it is trained for many epochs. If too many neurons are used in the network, they could define decision regions too liable to samples. This could lead to a model that will lack in generalization, i.e. it will not able to predict correctly new samples since it has adapted too much to the characteristics of the training data (*overfitting*). At the same time, if a network is trained for a lot of epochs, i.e. it is *overtrained*, the same issue could be registered as well. Commonly, to avoid that, and then obtain a model that generalizes correctly, the dataset is divided into tree sub-sets (*Holdout* procedure): Training, Validation and Test set. The Training set will be used to train the network, hence the name, and, conceptually, its error will decrease with the increase in the number of epochs. On the other hand, theoretically, the error on the Validation set could start to increase from a certain number of epochs. It is good practice to set the number of epochs to the one corresponding to the lowest error on the Validation set (*Early Stopping* approach, Fig. 1.3). If Validation and Test sets are representative of the whole dataset, such choice should lead to the lowest loss also on the Test set.

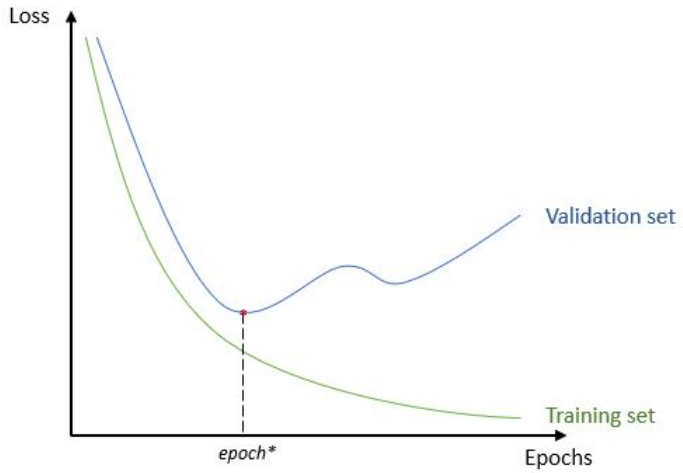


Figure 1.3: Early Stopping

1.2 Deep Neural Networks

Deep Neural Networks are a particular subset of ANNs characterized by a very "deep structure", i.e. they are composed of several hidden layers. The key aspect of those kind of networks is the automatic feature extraction [6]. While classical ML techniques make predictions from a set of features that have been prespecified by the user, Deep Learning Techniques can autonomously learn the best input representation. This characteristic is typical of Representation Learning Techniques that transform feature in input into some intermediate representation in order to make decisions; in the case of Deep Neural Networks, features undergo different transformation steps which gradually allow to obtain more complex and explanatory ones [8]. This process is known as *"Feature Learning"*. Difference between classical Machine Learning and Deep Learning techniques from this perspective are depicted in Fig. 1.4.

The key factors that contributed to the uprising of deep networks are, above all, the huge amount of data not available before and the easy availability of high-speed computation technologies (e.g. Graphic Processing Unit - GPU). Deep networks, being characterized by several hidden levels, are also characterized by a large amount of parameters. Their optimization and the iteratively processing

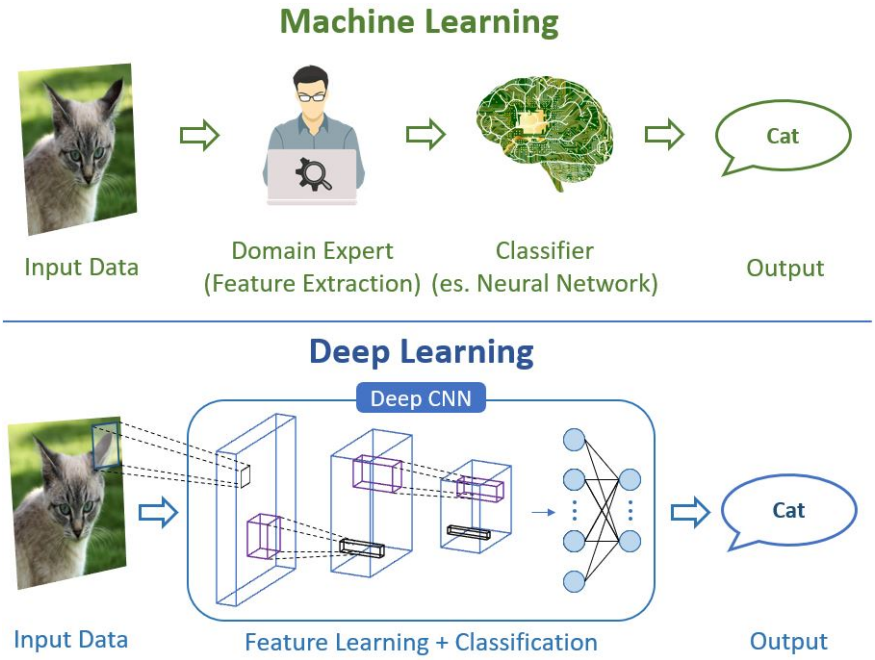


Figure 1.4: Classical Machine Learning vs Deep Learning

of the massive training dataset require a huge computational power that has been one of the biggest technical limitations in deep networks. Training requires a lot of time, fortunately, in the last period, the evolution of General-Purpose GPU (GP-GPU) has brought great benefits by greatly reducing the training times of these networks.

Deep Learning encompasses different approaches, which translate in a different deep structure and neurons' interconnections. From *Deep Convolutional Neural Networks* (the "deep" counterpart of the Multilayer Perceptron), a kind of feed forward networks characterized by layers which are not all fully connected, that are widely used in problems that involve image, video and audio analysis; to *Recurrent Neural Networks*, in which connections between neurons can form cycles, particularly useful in processing sequences of data, therefore to solve learning problems as writing recognition, speech recognition, and so on; to *Deep Boltzman Machines* and *Deep Belief Networks*, which are two types of Stochastic Deep Network in which the units correspond to random variables; to *Autoencoders*, networks used most of all for unsupervised training that can learn an efficient coding of its same

input, the concept is to reconstruct the input through intermediate representations that compress or reduce their size [8].

1.2.1 Convolutional Neural Networks

Deep Convolutional Neural Networks (D-CNN, or simply CNN) are a special kind of Feedforward Networks that are extremely successful for the image, video, but also audio, analysis. Such networks are typically composed of many layers with a lot of neurons, therefore, connect all the neurons within a layer to all the neurons of the previous layer (*Fully Connected Networks*) will lead to a model with high complexity. The solution lies in connecting a neuron only to a subset of neurons of the previous layer (receptive field).

Typically, the inputs of a CNN are multiple arrays representing colour images (three 2D arrays, one for each RGB channel), greyscaled images or audio spectrograms (a single 2D array), signal or sequences (1D array), etc [6]. In order to well understand CNNs, in the following, we will focus mostly on 2D images.

Each image will be processed by filtering it; each filter will be applied by multiplying a small portion (spatial region) of the image with a set of weights (filter or kernel), and then the result will be input to an activation function. In this way, a neuron will be connected only to the neurons of the previous layer belonging to that region. In addition, since this multiplication is repeated on the whole image leaving the set of weights unchanged, it can be implemented through convolution. The result is a CNN capable of learning both the filters and the classifier through particular optimization algorithms such as minibatch Stochastic Gradient Descent (minibatch SGD), RMSprop, Adam, and other algorithms, some of which implement an adaptive learning rate or rely on second-order methods. Since the detailed functioning of these algorithms is beyond the scope of this work, for further clarifications we refer to [11].

The architecture of Convolutional Neural Networks is composed of different intermediate convolutional and pooling layers performing feature learning/extraction, and a final classifies composed of few fully connected layers (Fig. 1.5).

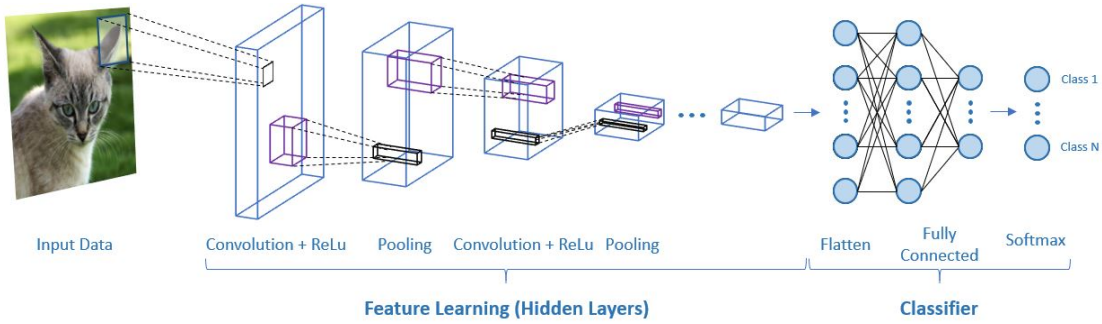


Figure 1.5: Convolutional Neural Network Architecture

The power and effectiveness of CNNs were demonstrated within the ImageNet Large Scale Visual Recognition Challenge [12], in 2012. The task was to recognize object categories in images found on the internet. The challenge included a huge imagery training dataset with about a million images. Since the CNNs' output is composed of a series of probabilities representing the probabilities that the input image belongs to the various possible classes, in ImageNet challenge, algorithms were evaluated on the "top-5 error", i.e. the percentage of time that the target label does not appear among the 5 highest-probability predictions. It was therefore shown that CNN far exceeded the capabilities of classic visual recognition methods; not only that, in this task, Deep CNNs are also able to outperform people [8].

CNN layers and parameters

Figure 1.5 shows, besides pictures, some keywords widely spread among the Deep Learning disciplines: Convolution, ReLu, Pooling, Flatten, Softmax and Fully Connected. Since CNNs rely on them, as also results presented in charter 3, could be interesting to conceptually understand them.

Convolutional Layers represent the core of this architecture. Stacking them sequentially, it is possible to extract a hierarchical set of features from the low to

the high level of details: the first convolutional layer, directly connected with the image data, extracts low-level features (e.g. edges); the central convolutional layers begin to learn slightly more complex patterns (e.g. nose, ears, headlights) since their input is represented by the feature map, i.e. geometrical feature extracted by the previous layer; at the end, the last convolutional layer will be able to learn high-level features like face, cars and so on.

Each convolutional layer is characterized by some parameters: the number of kernels (N), which have typically the same size and are square; the kernels' size (KxK); and a *stride* parameter (S). Assuming in input an array with WxHxD size, such a layer performs the convolution between the input array and each of its kernels (one-by-one) by sliding the kernel at each step by a number of positions indicated by the stride parameter. This would result in a 3D array of W*xH*xN size, where W* and H* are respectively less than W and H. To avoid this size reduction, the input array is often padded by a certain number (P) of levels (arrays) of zeros. The resulting array is then processed by the layer's activation function that could be a sigmoid, a hyperbolic tangent, a step function, and so on; however, typically, all the convolutional layers in CNNs are characterized by a Rectified Linear Activation Function (ReLu) that simply returns the maximum value between its input and zero.

The pooling layers performs a downsampling operation on the input volume. Practically, it reduces the size in terms of width and height (leaving the depth unchanged) of the input volume in order to be able to progressively reduce the network complexity (in terms of layers' size, parameters and operations), and also to make the extracted features space invariant (i.e. independent from the specific location where they have been detected [8]). Generally, Max Pooling or Average Pooling are performed, which respectively select the maximum value or the average of the values that fall within the filter (pool). In terms of parameters, such layers are characterized by the pooling filters' size (RxR) and a stride parameter (SP).

The concept is almost the same of the convolutional layer: the pool slides on the input according to SP, selecting, at each step, the max or avg value in the pool.

Typically, the two layers above are arranged in (convolutional) blocks including one or more convolutional layers and a final pooling layer.

On the other hand, the last layers are fully connected and constitute the classifier. Since the output of the last convolutional block is generally a 3D array, a Flatten layer is generally inserted between such block and the first fully connected layer in order to transform the 3D array in a 1D one. Finally, the last layer of the network is typically equipped with a *softmax* activation function which transforms the outputs of the network into probabilities. In practice, the network produces an array of score values with a dimension equals to the number of neurons of the output layer, which coincides with the number of classes; however, to obtain the probabilities of belonging to the different classes of the sample in input, such scores must be "normalized" (softmax).

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

where x_i are the scores.

In conclusion, if on the one hand CNNs are able to automatically learn features, on the other they are extremely complex to parameterize. Empirical choices must be made concerning the specific optimizer, the activation functions, the number of layers, the number of neurons per layer, etc. which translate in a very high number of parameters to set, generally through different experiments.

Fortunately, different studies have already been done and are available in the literature. As we will see later, many of them have produced CNN-based network structures working well for specific tasks which can be used to solve similar tasks (fine-tuning them) or as starting point to build more complex structures.

Building CNNs

What has just been explained suggests that there are different ways to train a CNN. Among these, we definitely have the training *from scratch*, i.e. the network is built and trained for the first time with the available data, or through *Transfer Learning (TL)* techniques that allow us to transfer the acquired knowledge from a certain domain, or a specific set of data in the same domain, to our task. We have also already seen how many parameters compete in the creation of a deep network; certainly, even applying TL techniques, such parameters will have to be considered, but in a different way. In other words, many of these parameters will only have to be fine-tuned (i.e. adjusted) to adapt the network to the task under examination; others, as maybe the size of the filters, could even be left unchanged.

Training from scratch, as partially already mentioned, is affected by some issues related to two particular aspects:

i. Quality and size of the dataset: to well train a CNN, or a Deep Neural Network in general, i.e. properly tune millions of parameters, *in many cases* a huge dataset is required. The problem lies in the fact that it is not always possible to find proper examples, thus these kinds of datasets are difficult to build. Moreover, its construction will be expensive and time-consuming since those examples must be labelled too. A solution could be represented by data augmentation techniques that can be used to provide new data starting from available ones by applying simple transformations as could be, in case of images, translation, rotation, etc. [8, 13]. However, also in this case, directly learning so many parameters from only thousands of training samples will result in overfitting [13].

ii. Computational Power: to train a Deep Network in a reasonable time it is required a proper computational power that is not always achievable.

On the opposite side, through Transfer Learning it is possible to obtain on

small datasets more or less the same performances of CNN trained on large-scale datasets [13]. Taking a step back through Machine Learning techniques, their main problem lies in the basic assumption that the training and the test set are derived from the same feature space and, especially when it comes to numeric features (or attributes), it is assumed that the same feature in both the sets follows the same distribution. The problem arises when, and if, the distributions change; in this case, the model needs to be retrained but this is often expensive and, in some cases, impossible. This is the basic concept on which the Pan's et Al. paper "A Survey on Transfer Learning" [14] is based, according to which the use of Transfer Learning (TL) techniques could remedy this issue. Conceptually, these techniques involve transferring knowledge learned in previous tasks to new tasks, or else through TL techniques it is possible to transfer knowledge from a source domain or task to a target domain or task; practically, this is what people do to solve new problems in a faster and better manner [14]. Note that the domain indicates the origin field of the problem, otherwise the task indicates the problem itself that needs to be solved. However, as regards the TL, among the various ML Techniques, Deep Learning is the one that benefits most from it.

Generally speaking, Transfer Learning goes beyond the concept of both Supervised and Unsupervised Learning referring to these paradigms to cope with different situations. On the basis of the source and the target domains/tasks, we can consider three different TL approaches [14]:

1. *Inductive Transfer Learning*: source and target tasks are different, otherwise it doesn't matter if the source and target domains are similar or not. However, to induce a predictive model trained on the source domain, labelled data in the target domain are required;

2. *Transductive Transfer Learning*: the source and target tasks are the same but domains are not. In particular, this approach is used when labelled data

are not available in target domains while they are in the source domain.

3. *Unsupervised Transfer Learning*: it focuses on solving unsupervised problems (e.g. clustering, dimensionality reduction, etc.) in the target domain. It is similar to the first approach but tasks, even different, are related. Differently, there are no labelled data available in both source and target domains.

From a conceptual point of view, the purpose is to train a network on a "base" dataset and task, and then "transfer" the learned features to a second network to be trained on a "target" dataset and task. Once trained a base network, its first layers are adopted as first layers of another (target) network. At this point, one can choose how to proceed [15]:

i. The first approach consists in training the target networks on its dataset and "fine-tune", during the training, also the features copied from the base network;

ii. The second approach consists in "freezing" the copied features leaving them unchanged during the training phase. In this case, the pre-trained network can be seen as a feature extractor. It is important to note that this process will work if the learned features are both general and suitable on both base and target datasets.

To understand which is the best approach to use, considerations must be made concerning the size of the target dataset and the number of parameters in the copied layers. If the target dataset is small and the copied layers are characterized by a significant number of parameters, fine-tuning could lead to overfitting, thus the latter approach could be better than the former; otherwise, in case the dataset is sufficiently large or the number of parameters is small, then fine-tuning can be considered [15].

1.2.2 Deep Issues for Deep Learning

So far we have introduced both Machine Learning and Deep Learning, also marking the relationship between them. Although in previous sections we did emphasize some differences between them, it is interesting to understand which are the main Deep Learning challenges. Certainly, one of the greatest advantages of DL is its ability to automatically learn features. Traditional Machine Learning techniques have always had a big limitation in processing raw data; indeed, the feature extraction process has ever been a sensible engineering process, also requiring the attention of domain experts [6]. Before a ML model can correctly process data, they must be first transformed into a suitable representation.

Although the DL introduces a big advantage in the learning of the characteristics, its use usually translates into a greater number of data necessary for training and greater complexity than the classic ML techniques. By complexity, we mean:

- The number of computations necessary to characterize a model, which can be easily addressed using new technologies (e.g. GPU).
- The number of parameters to consider and tune to well train a deep network (as discussed in section 1.2.1);
- The comprehensibility of the model, which is still an open issue. As a matter of fact, this is an issue that affects the whole AI domain.

According to the latter point, several AI approachesare so complex to understand that they are considered as black-boxes. In other words, such approaches lead to the creation of systems for which the input and output are known but not the inference process, the reasoning, that the system has done to produce that output. A glaring example could be the difference between a Decision Tree and a Deep Network. Although the output model (a tree) of a tree-based algorithm may be deep and complex (in terms of the number of nodes), it is always possible

to go back, even if not easily, to the rule that led to the classification of a given sample (e.g. the class is 'A' because the first feature has a value included in a certain range, the second is equal to a certain number, and so on). When it comes to Deep Networks, it is not possible to understand which set of features or which rules the network used to deduce a class, so they can be considered as complex black-box models. Explanations supporting the output of a model are crucial, e.g., in medicine, experts require far more information from the model than a simple binary prediction for supporting their diagnosis [16]. The figure 1.6 summarizes the problem: actually, most of the outputs of ML-based systems are not directly comprehensible.

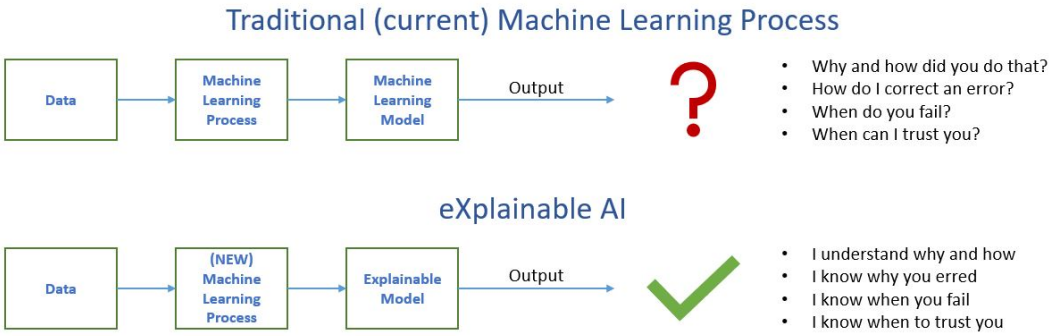


Figure 1.6: Explainable AI process compared against the traditional Machine Learning training schema[1].

Theoretically, Explainable AI (XAI) [16] deals with three particular concepts which should not be confused as they highlight different aspects of a given system:

- The Interpretability (also called Transparency) refers to a passive characteristic of a model referring to the level at which a given model makes sense for a human observer. It helps ensure impartially decision-making (detecting and correcting biases in the dataset) and provides robustness (highlighting potential adversarial perturbations);

- Explainability can be viewed as an active characteristic of a model, denoting

[1]Inspired to the DARPA image available at https://www.darpa.mil/program/explainable-artificial-intelligence

any action or procedure taken by a model with the intent of clarifying its behaviour;

- The Comprehensibility refers to the ability of a learning algorithm to represent its learned knowledge in a human understandable fashion.

It is clear that the more a system met these concepts, the more it can be understood and, maybe, trusted.

The same concern was expressed by DARPA (Defense Advanced Research Project Agency) that looks at the future considering Explainable AI, Explainable ML in particular, as the main foundation on which AI systems are built in order to properly understand, manage and also trust them. According to DARPA, ML techniques, both new and actual, will have to be capable to explain their rationale maintaining a high level of learning performnaces and this will be possible allowing them to produce more explainable models assisted by an "Explanation Interface" to communicate with users [17] (as figure 1.6 shows).

1.3 Deep Learning for Computer Vision and Audio Processing

In recent years, so-called Deep Learning approaches have demonstrated excellent potential in solving new and still unsolved challenges in a wide disparity of domains (e.g. Medical [18], Transportation [19, 20, 21], Video Surveillance [22], etc.). From a conceptual point of view, given its ability to extract information from high-dimensional data [6], Deep Learning has had great promising results in tasks related to:

- the image and video analysis (Image Processing and Computer Vision), such as Object Detection, Semantic Segmentation, Object Tracking, etc.;

- the audio signal analysis (Audio Signal Processing), such as Audio Classification, Sequence Transduction, Event Detection, also Automatic Speech Recognition (which is also an NLP facet), etc.;
- the natural language understanding, such as Sentiment Analysis, Speech Recognition, Question Answering etc.

For the purposes of this thesis, for reasons that we will explain in the next chapter, we focus mostly on Image Processing, Computer Vision and Audio Classification.

Theoretically, Computer Vision (CV) and Image Processing (IP) are strictly related. On one side, CV is often considered an AI discipline addressing AI applied to the visual world [23] which enables computer systems to automatically see, identify, and understand the visual world by mimicking human vision [24]; on the other side, Image Processing includes all those techniques and methods that aim to extract information from images, including all the image analysis life cycle steps such as image acquisition, image enhancement, image pre-processing and image segmentation [25, 26].

Even if nowadays, when it comes to CV and IP, it is a common thought to immediately refer to applications strictly related to AI (or DL), is not properly correct. Several CV applications (e.g., making computers able to perform tasks based on the analysis of visual inputs such as images) have been designed long before the development of modern AI techniques and do not rely on intelligent decision making (e.g., edge detectors) [27]. On the same page, Image Processing techniques does not necessarily have to rely on DL; just think to the Anisotropic Diffusion technique [28] which, relying on partial differential equations (PDE) [29], aims to reduce image noise without damage others significant information (e.g. edges, indeed it can be used to perform Edge Detection too).

Nevertheless, in the last years, DL techniques, CNNs in particular, have brought

so many benefits to become almost a de-facto standard in CV [24] and IP applications. Lastly, it is important to emphasize the relationship between CV, IP and DL; in practice, Computer Vision defines the macro tasks (e.g. object detection, scene segmentation) which can be addressed relying on Image Processing procedures, which, in turn, can rely on Deep Neural Networks.

It could be interesting to note that Image Processing is a sub-field of Signal Processing, as also Audio signal Processing is. Indeed, Signal Processing encompasses all the methods and the tools that allow analysing and model signals, extract information and patterns from them, classify or synthesize them, and also morphing them, whatever they are communication signals, images, audios or videos [30]. However, also in the case of audio or communication signal processing, Artificial Intelligence only represents a set of adjuvant technologies since Signal Processing is not totally a sub-field of AI; just think to the widespread Fast Fourier Transform which allows analysing a signal passing from the time domain to the frequency one, it is not an AI technique at all. Besides that, our purpose is not to characterize or analyse all the Signal Processing domain, indeed, in the following, we will focus only on the Audio Classification task.

1.3.1 Deep Learning Approaches

So far we have discussed two different network training modalities. On the one hand, we have the *training from scratch* which allows us to build a network suiting it on the particular task, but it also introduces some disadvantages: the tuning of the parameters (training) is a time-consuming and complex process which requires a lot of data and an adequate computational power. On the other hand, we have *Transfer Learning*, which allow us to achieve high performances even with small datasets and in a time-efficient manner. Nevertheless, we cannot apply Transfer Learning is none before us well trained the network from scratch relying on large

and qualitatively good datasets.

In recent years, Computer Vision applications are those that have most caught the attention of researchers. They proved, over time, the real potential of CNNs by creating disparate architectures testing them on various datasets (e.g. ImageNet [31], MS COCO [32], PASCAL VOC [33], etc.) commonly used as benchmarks in the homonymous challenges. Such networks have been created to address a particular task such as Object Detection, Image Classification, Semantic Segmentation, etc.; however, it is worth noting that another peculiarity of Deep Network lies in the fact that the same network can be adapted to solve more tasks, even if they are different from the one the network was created for.

When it comes to Image and Video analyses, the tasks which caught more attention are Image Classification, Object Detection, Object Tracking, and Semantic Segmentation, given their various applications in the domains mentioned above. The Image Classification is the classical classification task which aims to predict the label of an input image given an adequate training step; Object Detection aims to indicate various objects in an image, also indicating their location; Semantic Segmentation is a sort of object detection but it aims to recognize how the objects compose an image at pixel level; lastly, Object Tracking, as the name suggests, aims to track, then locate, objects moving within the environment.

Image Classification Networks

The Image Classification task is what led to the affirmation of Deep Neural Networks. The results obtained in the ImageNet Large Scale Visual Recognition Challenge [12], in 2012, demonstrated that CNNs can solve this task in a better way than the classic Machine Learning techniques. Indeed, **AlexNet** [34], trained on the ImageNet dataset, exhibited excellent performance marking, de facto, the line of the technological progress by shifting the focus from ML to DL techniques, CNNs in particular. Starting from these results, more and more researches have

been carried out in this direction and have led to the definition of some networks which, given their performances evaluated on the aforementioned datasets , have become almost a de-facto standard in addressing problems related to the classification of images. We will refer to these as state-of-the-art (SOTA) networks. Among the others, we can cite:

ZFNet [35] Which inspires to AlexNet making only a few edits in terms of parameters.

VGGNet [36] Which is a family of networks characterized by a very deep structure; e.g. VGG16 (16 layers) and VGG19 (19 layers), which are also the configurations that have achieved the better performances among those of the family. Nevertheless, more layers bring to a huge increment in terms of parameters; that is why VGG networks adopt only 3x3 shaped filters.

ResNet [37] Which introduces shortcut connections (residual connections) between two-layers blocks starting from a base structure inspired to VGG. These connections allow achieving deeper networks (ResNet-50/101/152) maintaining a less number of parameters with respect to the VGG networks.

Inception series Which indicates a set of networks which rely on the "Inception Modules", presented for the first time together with **GoogLeNet** [38], which are composed of different parallel convolution layers and a final filter concatenation layer which combine all the outputs (feature maps) in a single one. Following Inception Networks introduced small changes to obtain better performances: Inception-v2 [39] introduces Batch Normalization achieving a faster training phase; Inception-v3 [40] introduces Spatial Factorization and Asymmetric convolutions building new inception modules which are deeper and have smaller filters than the original ones. Lastly, Inception-ResNets [41] introduces residual connection inside the Inception blocks (residual inception blocks).

DenseNet [42] It proposes a different (*dense*, hence the name) connectivity pattern between layers. Indeed, direct connections between any layer and its subsequent layers are introduced to improve information flow. This means that a layer receives all the output feature maps of the preceding layers. The network structure is composed of an alternation of Dense Block and Transition Layers. To reduce the network complexity, and then improve the computational efficiency, Bottleneck layers (1x1 conv layers) before convolutional layers and a compress factor in each Transition Layer are introduced to reduce the number of feature maps (DenseNet-BC).

DarkNet The first version, DarkNet-19, was proposed together with YOLOv2 [43]. It is composed of 19 convolutional layers and, as for VGG, filters have all the same size (3x3). Moreover, 1x1 filters are used to compress the feature maps between conv layers, and Batch Normalization is implemented to stabilize training and speed up the convergence of the model. To further improve performances, a deeper version (DarkNet-53) has been introduced with YOLOv3 [44].

MobileNet [45] MobileNet is based on the concept of "depthwise separable convolution". Typically, a convolution layer simultaneously performs filtering and combination of inputs in a set of outputs. Otherwise, in this case, these steps are separated in a depthwise convolution, which applies a single filter to each input channel, and a pointwise convolution (1x1 filter) to combine the outputs of the previous step. This translates in a drastically reduced computation and model size (parameters). According to the experiments performed, despite having a very small number of parameters, such networks can match, more or less, the accuracy of networks as VGG16 and Inception-v3.

Object Detection Frameworks

The object detection task is a little bit more complicated with respect to the Image Classification one. To improve detection performances, the model must be trained on a dataset with object-level annotations (e.g. ImageNet), instead of only image-level annotations (i.e. image class or label). However, the object-level annotations collection is expensive, therefore a common scenario is to pre-train the model on a large dataset with image-level annotations and then fine-tune the network on a given detection dataset [46]. Above networks can be used as feature extractors to face the object detection task embedding them into two base frameworks [47, 46]:

- *Two-Stage Detectors*: such frameworks include a first step in which category-independent region proposals (also known as Region of Interest - RoI) are generated from the initial image; in the second step, a CNN is applied on such RoIs in order to extract features, then a category-specific classifier is applied to individuate the proposals' label and a bounding-box regressor is used to locate the object in the initial image. Although such models achieve high accuracy, they are typically slower than One-Stage Detectors. All the region-Based CNN frameworks belong to this category (R-CNN [48], Fast R-CNN [49], Faster R-CNN [50], R-FCN [51] and Mask R-CNN [52]);

- *One-Stage Detectors*: these frameworks are based on a single CNN able to directly predict the class probability and the bounding-boxes starting from full images. This allows real-time object detection at the expense of lesser accuracy then Two-Stage Detectors. The most popular One-Stage Detectors are those belonging to the *You Only Look Once* (YOLO) family (e.g. YOLO [53], YOLO9000 [43], YOLOv3 [44]) and the so-called Single Stage Detector (SSD [54]).

Semantic Segmentation Approaches

Networks for Image Classification have also had a central role in the Semantic Segmentation task. Indeed, transfer learning, starting from large-scale image dataset (e.g. ImageNet) pre-trained network, proved to be a suitable approach to address "pixel-level classification" [55, 56]. This approach has been used in combination with Fully Convolutional Neural Networks (FCN) structure. Practically, traditional classification network such AlexNet, VGG and GoogLeNet have been revisited replacing their Fully Connected layers with others Convolutional ones in order to obtain spatial maps instead of classification scores [57]; therefore, deconvolution is applied on such maps to obtain "dense per-pixel labelled outputs" [56]. The same approach has been used with DenseNets [58]. Similarly, the DeepLab networks [59, 60] rely on VGG-16 (or ResNet-101) in combination with fully connected conditional random fields (CRFs) in post-processing to better define the region boundaries. Even if there are other Semantic Segmentation approaches based on transferring knowledge from SOTA networks, our aim is not to write a detailed literature review on these applications. However, other worth mentioning approaches are the Region-based Semantic Segmentation [61, 62], which relies on R-CNN, and those which relies on Recurrent Neural Networks [63, 64]. Finally, SegNet [65] and U-Net [66] are both classifiable as *decoder variants* [56] since they present an Encoder-Decoder structure. SegNet has 26 convolutional layers in total, 13 for both the Encoding and Decoding network; the peculiarity is that the 13 convolutional layers composing the Encoding network are the same layers used in the VGG16, therefore it is still possible to use pre-trained weights. On the other hand, U-Net has a structure composed of 23 convolutional layers and has been explicitly created to face biomedical segmentation tasks.

Object Trackers

Lastly, for what concern Computer Vision tasks, we mentioned Object Tracking (OT). Simply, we can consider Object Tracking as an object detection task which must detect and locate the object continuously in time. Generally, there are two different OT tasks which are: i) Single Object Tracking (SOT), or Visual Object Tracking (VOT), aiming to track an arbitrary object which has been box-bounded in the first frame of the video, as also defined in the VOT2015 challenge [67]; and 2) Multiple Object Tracking (MOT), in which the object to track is not selected a priori then a preventive object detection step is required [68]. According to the VOT2018 challenge [69], VOT trackers can be subdivided into Short-Term (ST) and Long-Term (LT) trackers, which mainly differ in the fact that LT ones are able to re-detect the target object after its complete occlusion or exit from the scene. Therefore, according to this subdivision, two main challenges were proposed. For the sake of knowledge, we will cite only the winning trackers. The Short-Term challenge was won by the MFT [2] tracker, which is a version of the MHIT [70] framework; meanwhile, the ST Real-Time sub challenge was won by the SiamRPN [71], which also occupies the seventh position in the main challenge, that combine the Siamese Architecture proposed by SiamFC [72] and the Region Proposal Networks (RPN) proposed for the first time together with Faster R-CNN [50]. Otherwise, the Long-Term challenge was won by the MBMD [73] tracker, followed by DaSiam_LT [74]. The former is based on a SSD-MobileNet architecture and an MDNet-based [75] verifier; the latter is an extension of the just mentioned SiamRPN which is able to recover the detection after its occlusion. The other side of the coin is represented by the Multiple Object Tracking (MOT). Most of the MOT trackers are based on the tracking-by-detection approach: the tracking is performed through the detection of the different objects in the various

[2] https://github.com/ShuaiBai623/MFT

frames which are then associated so that each bounding box always contains the same target; even if there are other approaches, generally, the process combines an Object Detection algorithm and a Feature Extraction/Motion Prediction phase in order to obtain information about interesting objects and further track them [68]. For the sake of knowledge, we cite Simple Online and Real Time (SORT) algorithm [76], one of the first CNN-based trackers which also rely on object detected through Faster R-CNN framework (pre-trained on PASCAL VOC dataset) and won the MOT competition in 2015 for online object tracking [77], ant its deeper implementation (Deep SORT [78])which achieved good performances in MOT16. Lastly, the Tracktor [79] tracker also deserves a mention since it reached state-of-the-art performances on all the MOT15-16-17[3] challenges simply relying on the assumption that, with a high frame rate, objects move slightly from a frame to another. It does not need any tracking specific training or optimization and only leans on object detectors (Faster R-CNN); lastly, two extensions are also presented together with the base version, one of which involves the Siamese neural network to improve the online objects' re-identification.

Audio Detection and Classification

Meanwhile, for Computer Vision applications, we are interested in all the tasks cited above to address the needs of our project, regarding the Audio Signal Processing (ASP) applications we are mostly interested in Audio Detection and Audio Classification. Actually, the former task can be easily converted into a "binary classification" one where the classes suite something like "there is sound activity" or "there is not sound activity". Therefore, we will focus more on Audio Classification, precisely on Environmental Sounds field. The first step in to define an appropriate feature representation; among the others we can cite the mel frequency cepstral coefficients (MFCCs), the spectrum (e.g. log-mel spectrogram), the raw waveform

[3]for other and more recent results, please refer to https://motchallenge.net/

representation directly [80], and so on. Different examples can be found in the literature which applies new CNN-based architectures as feature extractor, e.g. in combination with MFCCs [81], or as a classifier, e.g. in combination with log-mel spectrogram [82, 83], and these are only a few examples. Moreover, also convolutional recurrent neural networks [84] and convolutional deep belief networks [85] have been used to face such a task. Moreover, it is worth noting that Image Classification networks, AlexNet, VGG, Inception-V3 and ResNet in particular, have also been used/analysed for Audio Classification [86, 87]. Indeed, for our experiments in Chapter 3, we will leverage on these results [87] to build our Alarm Classifier concerning the Level Crossing scenario.

Chapter 2

Deep Learning in the Railway Sector

The work presented in this thesis stems for the European project RAILS, although not directly involved. RAILS ("Roadmaps for AI Integration in the raiL Sector"), is a research project funded by the Shift2Rail (S2R) Joint Undertaking under the European Union's Horizon 2020 research and innovation programme. The main objective of RAILS is to investigate the potential of Artificial Intelligence (AI) in the rail sector and contribute to the definition of roadmaps for future research in next generation signalling systems, operational intelligence, and network management. For this aim, it investigates the potentiality of AI respectively in safety and automation, predictive maintenance and defect detection, traffic planning and management. The project is actively supported by a panel of industries and railway operators (the Advisory Board) bringing into RAILS additional knowledge and acting as a springboard for the research results towards industry and S2R members.

The work carried out in this thesis addresses one aspect of a real scenario suggested by Hitachi Rail STS, a leading company in the rail transportation industry and member of the RAILS Advisory Board.

The railway sector is crucial since different functionalities that systems must implement are identified as safety-critical. Generally, a system is identified as

safety-critical if the failure of one of its functionalities can lead to serious injury or loss of life, as well as the possibility of financial loss. Therefore, both the system itself in terms of its functionalities and all the processes needed for its realization (including the used tool suite), they must meet robust safety requirements being compliant with rigid standards, as CENELEC [88] can be for the railway sector. The EN50129 standard [89] defines a probabilistic quantification of "safety" in the railway domain by defining four Safety Integrity Levels (SIL) according to the admitted tolerable hazard rate (THR); the more the system is safety-related, the more restrictive are the requirements. For the sake of knowledge, for SIL4 (the most safety-related) systems a THR of about 10^{-9} failures per hour is required.

Beyond the specific standard and procedures, when a system results to be compliant with their requirements, it will result as "certified". It is very likely that the installation of additional components (e.g. sensors) within the system can lead to its invalidation, implying the need to repeat the certification process which could be extremely complex, but also time-consuming and cost-intensive. Since that, in the context of maintenance, inspection methods that are not invasive are often preferred. In addition to the sensors that the new systems could already be equipped with at the certification stage, it is important to clarify that not all systems currently in operation can always be modernized since it could be too expensive. Therefore, the application of inspection methods based on non-invasive sensors, such as cameras or microphones, which do not even need to be placed above existing systems, could be an excellent solution for the systems' status monitoring.

2.1 Smart Maintenance at Level Crossings: a Real World Case Study

Level Crossings (LCs) represent one of the most sensible and global issues in the rail sector. According to the European Union Agency for Railways (ERA) [90], "Level-crossings not only represent the physical intersection (of a railway track and a road), but also an intersection of responsibilities and interests"; indeed, they represent a safety risk for both railway and road users. In the 2020 "Report on Railway Safety and Interoperability in the EU" [90], ERA estimates that the number of fatalities and accidents at level crossings represents more than the 25% of all the EU railways' accidents; moreover, even if they are constantly decreasing in time (about 4% per year), the descendent trend is slower than the other types of railway accidents. In the same report, they also highlight that in order to increase safety, in recent years, attempts have been made to replace level crossings with subways or bridges. Nevertheless, this replacement process is too expensive given the huge number of LCs (about 105 thousand) within the EU-28 countries. Since that, it seems necessary to adopt other methodologies to increase safety and reduce accidents, such those proposed within the SAFER-LC project [91], concerning new warning signalling systems, speed bumps, but also detection and risk evaluation systems based on machine learning. Also, in [92] the authors gives a complete review of suitable obstacle detection technologies and their associated algorithms that can be used to support risk reduction of Level Crossings, and analyses the combination of obstacle detection sensors with intelligent decisions layers (such as Deep Learning model) as an opportunity to provide robust interlocking decisions.

Beyond this, also the maintenance is an important task to consider: guarantee the correct functioning of LCs should be the first step in order to ensure safety and transit availability. Given the huge impact that Level Crossings have on the railway system, as a safety-critical component, we have concentrated our efforts

investigating the possibility to monitor their activity applying DL-based methods through non-invasive approaches, as from the scenario proposed by Hitachi Rail STS.

Generally, Level Crossings can be subdivided into two macro categories: Passive and Active. *Passive LC* do involves neither signals to notice incoming trains to the road users nor barriers; otherwise, *Active LC* can involve warning signals, i.e. acoustic alarms and flashing lights (semaphore), and also user-side protections (gate arms, bars), which can be automatically triggered by the approaching train, or manually activated by human workers [93]. Focusing on Automatic Active LC, they work in a correct state if, once triggered by the train, the warning signals (alarm and traffic lights) are activated and, after some second, the bars begin to close. Once closed, one of two scenarios could take place: in the first, the alert bell warning device stops working as soon as the bars are closed; in the second, it continues to function until the train has passed. Therefore, as it is easily understandable, the level crossing is made up of several subsystems that must work properly together to ensure clear user-side notification and protection, thus a safe transit.

Beyond the particular implementation of these subsystems, which could also vary from model to model, we are interested in evaluating the LC health status through Deep Learning approaches basing only on video and audio data. This approach also has the great advantage allowing us to consider the whole Level Crossing System as a black-box, focusing only on its behaviour, i.e. just evaluating its visible and audible output estimating if it deviates from the expected and nominal functioning.

2.1.1 An Introduction to Smart Maintenance

Focusing on the rail sector, most of the maintenance activities are performed on a scheduled basis, this could allow the system to fail between two adjacent inspections[1]. Moreover, as will be emphasized by some papers analysed in the following Systematic Literature Review (section 2.2), common inspection methods involve manual and visual examination; such process results to be very slow and limited to the capability of human workers which often lead to a poor defect identification. Therefore, in the last years, the attention has been moved toward new technologies and "smart" approaches, in order to perform predictive maintenance instead of a corrective one.

The Smart Maintenance (SM), conceptually, is defined in [94] as *"an organizational design for managing maintenance of manufacturing plants in environments with pervasive digital technologies"*; it is also seen as a multidimensional concept based, among the others, on a *data-driven decision-making*. Within the Shift2Rail H2020 SMaRTE [95] project, a clear scope is associated with smart maintenance, i.e. *"to improve the current railway train maintenance systems, through the integration of predictive data analysis algorithms and online optimization tools within an improved Condition Based Maintenance (onwards CBM) strategy"*; where Condition Based Maintenance is based on a maintenance program exploiting the information collected through nondestructive testing methods, in order to recommend maintenance decisions in a proactive fashion. In this work we apply this definition to the whole railway domain, and not only to trains.

Therefore, we can summarize the Smart Maintenance as encompassing all those activities and technological "intelligent" tools aiming to monitor system data and proactively evaluate its health status. Smart maintenance could include subactivities such as Defect Inspection, Detection, Forecasting, etc. when performed

[1] https://cordis.europa.eu/project/id/777627

to prevent the whole system failure through "smart" data analyses.

Data is the key factor, and managing Big Data Streams is the first task that should be faced. With the advent of IoT, ever-increasing real-time streams of data, coming from different sources, must be managed also because not all the data are qualitative optimal. For the sake of knowledge, among the very different and heterogeneous applications that can be found in the literature, the authors in [96] implement a Machine Learning-based (online support vector regression) Condition Based Maintenance task to estimate remaining the useful life of axle bearings, leveraging on data coming from several on-board sensors. In [97], the authors proposed a decision-tree based method to evaluate the health status of train truck and wheels, leveraging on data collected by various kind of sensors along the railway line. Recent reviews about data-driven Smart Maintenance approaches are evaluated in the next section. Nevertheless, if, on one hand, the use of external sensors *could* provide more (useful) data, it should strongly impact the time and the effort needed to the development of AI-bases solution in railways. Indeed, the use of *new* sensors (i.e. not already available and/or whose signal are accessible) usually results in the need for a new certification process, increasing costs and the investment of further time on systems that have been compliant with very established standards for years.

Nonetheless, cameras and microphones can be installed in a non-intrusive way (e.g. by using a different power line) without any "tampering" with the existing systems. This difference allows us to install as many image/audio sensors as needed, without i) requiring a new certification process and ii) impacting on the system safety. Considering that these kinds of sensors are often already installed for security reasons, it is extremely interesting and advantageous using them for maintenance purposes.

2.1.2 LC Health Status Monitoring: a Plausible Approach

The analysis of level crossing functional state in a sensor-free manner is a challenging task involving the design of several modules each intended to capture a particular aspect. Focusing on the use of Deep Learning, and considering an Automatic Active LC as presented above, a possible approach consists of the following modules:

- Alert Bell Module, intended to determine whether the bell is ringing or not, and its intensity. Although this task might appear straightforward, it involves taking into account a wide set of possible scenarios hard to anticipate. LCs are located both in urban and peripheral environments and, to the best of our knowledge, each country adopts a different kind of sound. Therefore, if on one side the sound activity detection could be faced easily, especially if the microphone is positioned as close as possible to the Alert Bell, it could be not straightforward to recognise the sound as an LC Alarm. As we will see in 3, some LC Alarms are too similar to other sounds that can be recorded in urban and peripheral environments;

- Light Signal Module, intended to determine whether the light signal is blinking or is steady, and its intensity. This task can be very hard if the light signal occupies only a portion of the image. Thus, the ideal situation is to have a camera totally dedicated to this module.

- Bar Analysis Module, intended to determine whether the bar is moving as expected or if an anomaly is present. This is the hardest module, since there is nothing similar in the literature (also papers focusing on the analysis of moving parts are probably too different to be effectively leveraged), to the best of our knowledge; further analyses and reviews must be performed in this sense.

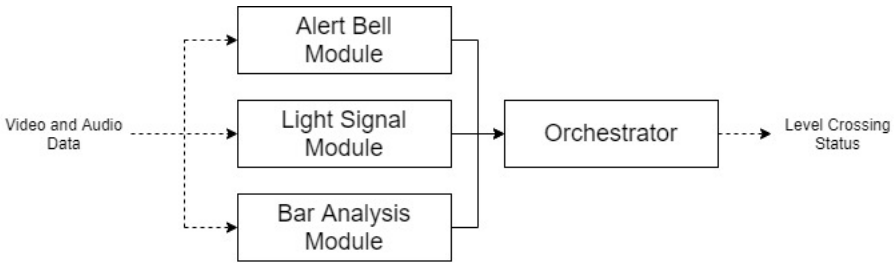


Figure 2.1: Whole System Overview

The latter module, in respect of the others, introduces others issues. First, if Alert Bell Detection/Classification and Light Signal Recognition can be faced as classification problems (to detect the functioning) supported by a regression approach (to evaluate the intensity), the Bar Analysis must be faced through a Motion Tracking approach, also evaluating the shape, speed and fluidity of the track in order to detect the health status. Moreover, data collection could not be straightforward. Indeed, if for the first two tasks a transfer learning approach could be easily adopted, generating a dataset for Bar Analysis could not be so simple since publicly available videos often adopt different camera point of views, they could be recorded by hand-holding cameras, then could introduce jitter between video frames, etc.

Clearly, all the modules above must be orchestrated by a coordinator module, implementing the logic needed to check the Level Crossing state. An overview of the whole system is shown in Figure 2.1.

Nevertheless, the whole process has such a complexity that it is out of reach for a thesis work. Indeed, we will focus only on the fist module presented, performing only the Audio Classification for Level Crossing Alert Bells, in order to assess that such a task can be well performed through a Deep Learning approach even re-using pre-built networks. Such purpose have also been directed form the following Systematic Literature Review.

2.2 Systematic Literature Review

On the grounds set out in section 2.1.1, also motivated by the desire to bring significant innovations to the researchers' community, the present Systematic Literature Review (SLR) has been drawn up in order to identify the current degree of integration of Deep Learning for Smart Maintenance in the Railway Sector based on audio and video analysis. To the best of our knowledge, no study has already been done in this direction.

Some previous works provide a review of AI/DL based approaches to address some aspects of the railway domain. For example, in [98] a novel framework based on computer vision and pattern recognition has been proposed to perform risk management in railway stations. The decision layer is based on a convolutional neural network to identify risks. There is also an interesting review of the application of convolutional neural networks in the construction field and crack detection, that are interesting from a transferability point of view. On the other hand, [99] recently drafted a survey on the Machine Learning applications for rail track maintenance, stating which algorithm, whether shallow or deep learning-based, have been used to detect structural defects, i.e. due to physical deterioration of the tracks, or geometry irregularities (e.g. misalignment). It states that, although deep learning algorithms have been widely adopted in recent years to identify structural defects, there are nevertheless different shortcomings to address such as the lack of benchmarks or labelled datasets, and the difficulty of finding defective observations. An interesting and complete wide-spectrum systematic literature review of Machine Learning methods applied to predictive maintenance is reported in the recent work in [100]. This analysis is conducted on the papers published from 2009 to 2018 and confirms that predictive maintenance experienced an increasing interest from researchers mainly from 2013. This work confirms that few paper proposed solutions specifically tailored for railway systems, and it also

confirms that the integration of predictive maintenance techniques with the latest sensor technologies avoids unnecessary replacement of equipment, saves costs and improves the safety, availability and efficiency of maintenance processes. A similar conclusion has been already reached in [101] in 2010. Lastly, another interesting and quite deep systematic literature review, in terms of Prognostic and Health Management (PHM), have been performed in [102]. For each PHM subfield, i.e. Fault Detection, Fault Diagnosis, and Prognosis (Remaining Useful Life estimation), it classifies paper according to the specific Deep Learning approach (CNN, AE, RBM, and RNN) and the dataset type they rely on (vibrational, time-series, imagery, and structured), validating the "universal applicability" of DL to all these kinds of inputs for PHM. Anyhow, it also emphasizes other DL challenges concerning, among the others, the model complexity and its tricky characterization/selection ("the use of deep learning is still an art"), the lack of labelled data, and also the fact that most of the analysed papers were about datasets gathered from bench-scale experiments which possibly lead to poor applicability in the real world; moreover, no image-based approaches have been identified to face the prognosis task.

2.2.1 Research Questions and Automated Search

The related work presented above shows a particular inclination, especially in the last few years, in the use of Deep Learning approaches as data analysis techniques for smart maintenance applications. Therefore, considering the increasing interest in DL and the constraint on the sensor, i.e. not using external sensors that may result in the need for a new certification process, through this SLR we would highlight the Smart Maintenance applications *based on Audio or Video analyses* which have been adopted in the Railway Sector. Considering this scenario, we would answer the following questions:

RQ1 For which components of the whole railway system (e.g. Infrastructures, Rail Tracks, etc.) have DL techniques oriented to Smart Maintenance already been applied?

RQ2 For which Smart Maintenance tasks (e.g. defect detection, fault diagnosis, condition based maintenance, etc.) have Deep Learning-based solutions already been adopted in the rail sector?

RQ3 What are the most commonly used Deep Learning-based approaches?

RQ4 Is it possible to leverage pre-built networks or creating them from scratch is strictly needed?

RQ5 Are there any available public datasets? And on what task/data?

The proposed study was conducted on three well-known scientific literature databases: IEEE Xplore Digital Library[2], Scopus[3] and ACM Digital Library[4].

The definition of the search query started with the identification of the search keywords, that have been organized in the following sets:

- S_1 = {"rail*", "metro" , "subway"}
- S_2 = {"smart maintenance", "predictive maintenance", "condition based maintenance", "fault diagnosis", "fault detection", "fault forecast*", "fault prediction", "fault prevention", "defect inspection", "defect detection", "health monitoring", "health status", "remaining useful life", "condition monitoring"}
- S_3 = {"deep learning", "deep reinforcement", "deep neural", "convolutional neural network", "recurrent neural network"}
- S_4 = {"computer vision", "vision", "image*", "video*", "sound", "acoustic", "audio"}

[2]https://ieeexplore.ieee.org
[3]https://www.scopus.com
[4]https://dl.acm.org

The asterisk means that also words which begin with the specified keyword are included, e.g. "rail*" includes "railways", "rails", "railroad", etc. The first set specifies the keywords related to the rail domain; the second set identifies the activities related to maintenance and defect/fault detection; the third set specifies keywords related to DL; the last set limits the applications to image and audio processing. Starting from these sets, the paper of interest should contain at least one keyword for each set. Hence, the search query is given by the logical disjunction (i.e., or) among keywords of each set, combined by the logical conjunction (i.e., and) at a higher level. The search query has been executed on the metadata in the command search of IEEE Xplore, and on title-abstract-keyword on both Scopus and ACM Digital Library.

On the automatically obtained results we applied the following exclusion criteria:

E1 works that are not written in English;

E2 works not related to predictive maintenance and DL;

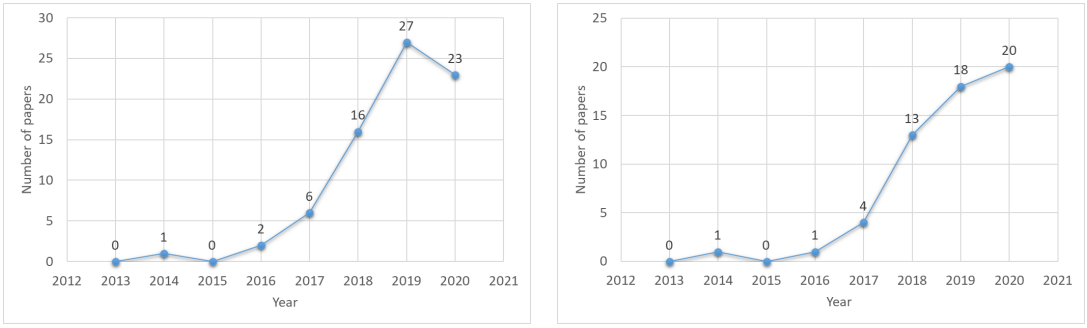
E3 works that do not present a focus on the rail sector;

E4 works that present results leveraging on data coming from sensors that are not cameras or audio sensors;

E5 works that do not propose a specific architecture but perform only a review (e.g., conference reviews and editorials);

E6 works that do not present any type of experimentation or comparison results, and make only propositions.

Evaluated results are dated between 2014 and August 2, 2020 (date of the last SLR update). Actually, no lower limit has been applied on the publication date, since, once E5 has been applied, no results are dated before 2014. Indeed, it is



(a) All papers resulting from the query (b) Available and English papers

Figure 2.2: Papers' trend per year from 2014 to 2 August 2020

worth noting that, before 2014, only five conference reviews (in 2009) were found matching the query on Scopus; therefore, we can assert that no journal/conference papers across the three digital libraries have been published before that year. However, without considering the exclusion criteria above, the IEEE Xplore library has returned 40 papers, Scopus 70 papers and ACM Digital Library only 1 paper. These results have thus been refined by carefully removing duplicated papers (i.e. the same paper can be found both on Scopus and IEEE Xplore) and applying the E5 criterion; then, 75 unique papers have been obtained, which trend per year is shown in Fig. 2.2a. Nevertheless, we found that some papers were available only in the Chinese language (6), meanwhile, others were not publicly available (12). Living them aside, we obtained 57 papers which have been published as 27 journal papers and 30 conference papers. The trend per year is shown in Fig. 2.2b.

On the obtained results, further refinements were conducted according to the remaining exclusion criteria. Indeed, between those 57 papers, there are some that are not fully compliant with the purpose of this literature review.

In [103], the authors propose a DarkNet-53-based network for tiny defect detection on the car-body surface; leveraging on these results, one year later, almost the same authors propose in [104] another DL-based quality assessment system for the painted car body. On the basis of the E3 exclusion criterion, such papers have been excluded from further analyses.

Considering the E4 exclusion criterion, other papers have been excluded. References [105] and [106] present two DL-based approaches to perform fault diagnosis concerning railway turnouts basing on a 2-D greyscale representation of current signals; the former relies on CNNs to detect faults, the latter on Deep Convolutional Auto-Encoders (DCAE) and Logistic Regression. Similarly, in [107] authors propose a new method for IGBTs open-circuit fault diagnosis relying on a greyscale image representation of current signals and a CNN-based network. Moreover, [108] propose an approach based on CNN for fault diagnosis of high-speed train bogies, analysing 2D greyscale representation of vibration signals.

Three more papers were left aside form this SLR according to E2. Reference [109] proposes a four-stage Big Data framework for rail inspection focusing mostly on data homologation through a Map Reduce-based approach and the MongoDB environment since data are collected from multiple sources (e.g. 3D-laser cameras and IMU sensors). Differently, [110] proposes a learning rate scheduling based on Hyperbolic-Tangent Decay for DL-based classification network; although it demonstrates promising results on different SOTA networks (section 1.3.1) tested on different widespread datasets, in our humble opinion, it is out of purpose for the proposed Systematic Literature Review. Lastly, in [111], authors propose a deep network to enhance the quality of catenary images, improving quality and visual effects. It is subdivided into two stages: the first perform multi-scale feature extraction, meanwhile, the second performs multi-feature fusion.

Lastly, although reference [102] proposes a very interesting and wide-spectrum review on the DL-based approaches for PHM applications, as already mentioned, it has been considered as related work, also because it does not entirely focus on the railway sector (E3, E6).

In view of the above considerations, this refining process has led to 48 papers which will be analysed in detail in the following.

2.2.2 Content Analysis and Results

Performing the in-depth analysis of the resulting papers, we noticed that they mainly refers to four railways' macro areas: *Rail Track*, *Catenary & Pantograph*, *Tunnel & Subway* and *Train Bogie & Frame*. Nevertheless, since three papers fell outside this classification, we have grouped them under a fifth general category, trivially named *Other Areas*. Table 2.1 and the pie-chart in Fig. 2.3 show the distribution of the papers given the five areas just mentioned. We will subdivide the following content analysis on the basis of this classification.

Table 2.1: Papers per Railway Macro Area

Railway Macro Area	**Papers**	**Papers**
Rail Track	[112],[113],[114],[115],[116],[117],[118],[118],[119],[120],[121]†, [122],[123], [124],[125]$^{\sim}$,[126]$^{\sim,\dagger}$,[127]	17
Catenary & Pantograph	[128]$^{\sim}$,[129]$^{\sim}$,[130]$^{\sim}$,[131]$^{\sim}$,[126]$^{\sim}$,[132]$^{\sim}$,[133],[134],[135],[136],[137], [138],[139],[140],[141],[142],[143],[144],[145]	19
Train Bogie and Frame	[146],[147],[148],[149]	4
Tunnel & Subway	[127]*,[150]$^{\sim}$,[151],[152],[153]	4 (1*)
Other Areas	[154],[155],[156]	3

$[x]^{\sim}$ only preforms type classification or object identification (no defects)
$[x]^{\dagger}$ performs tests on data collected on specimens (laboratory tests)
$[x]^{*}$ relies on data not related to the railway environment

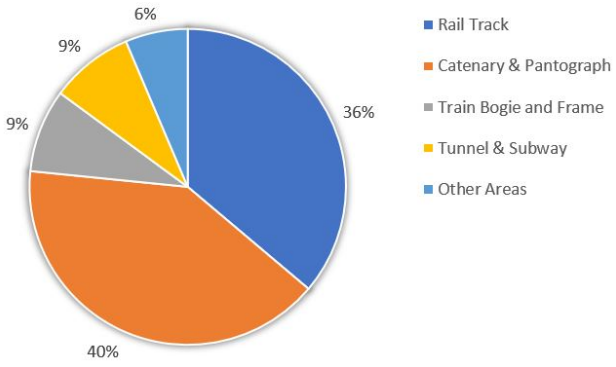


Figure 2.3: Papers' Distribution per Railway Macro Area

For each paper, the particular addressed maintenance task and the used DL approach have been evaluated. Moreover, where specified, also the adopted data acquisition system and database have been highlighted.

According to the content of the papers, we have grouped all the maintenance tasks in two main categories:

- Surface Defect Detection (SDD): as the name suggests, it deals with the detection of surface defects through vision-based methods. DL-based approaches have been used to detect defects or imperfections (e.g. cracks) on the surfaces of rails' heads[5], Tunnels' Lining, etc.

- Defect Inspection (DI): the analysis is not limited to the object's surface imperfections. This category encompasses all those models which, relying on visual or audio analysis, are also able to detect different kinds of faults as broken objects, missing objects, etc.

Similarly, the Deep Learning approaches have been subdivided in:

- Classification (C): the paper adopts an approach based on a CNN built from scratch or a SOTA network, exclusively to perform whatever kind of classification;

- Object Detection (OD): the proposed method is based on CNN or frameworks commonly adopted for object detection. It is important to note that such frameworks typically include a feature extractor (e.g. a CNN), performing object localization and also labelling (a class is associated with each detected object). This means that Object Detection architectures can both involve a feature extraction network and (in series or in parallel) a classifier;

- Semantic Segmentation (SS): the described model is based on a Semantic Segmentation approach. It is worth noting that object detection tasks can also be performed through SS.

Although some papers do not introduce explicitly a new approach for Smart Maintenance based on Deep Learning, we decided to keep them inside the SLR since they lay the groundwork for future implementations, presenting an interesting object detection or classification method; indeed, also this can be seen as DL applied

[5]The rail's head is the top part of the rail, the one which is directly in contact with train's wheels

to maintenance purposes. Such papers will be highlighted in the following tables as $[x]^{\sim}$.

Lastly, it is worth mentioning that models can be trained and tested on already available datasets or on data collected from scratch, moreover, public datasets can also be used to pre-train the proposed architecture. In case of data collected from scratch, systems for data acquisition can work in two modalities: online, i.e. the system have been proposed to work on the train or as a trackside component, and off-line, i.e. it must be used when no trains run on the rail line. In the following analyses, we will also highlight public datasets, if any, used for model training, or dataset built from scratch that authors have made available to the researchers' community.

Rail Tracks

Rail Tracks are the core element of the railway system. Correct health monitoring and pre-emptive maintenance are essential for the whole system safety. Common inspection methods involve manual and visual examinations, resulting in a very slow process limited to the capability of human workers which often lead to poor defect identification. Therefore, in order to improve maintenance, researchers began to test some DL-based applications in order to improve detection speed and accuracy, moved by the impressive results such techniques achieved in the last years.

Concerning the Rail Track area, we have obtained 17 papers which introduce innovative or improved methods to monitor and/or analyse different the Rail Track components. Indeed, the Rail Tracks, beyond turnout and switches, are mainly composed of rails, sleepers, fasteners, welded joints and ballasts. In Table 2.2, we classified all the papers according to these components and the Smart Maintenance task they address; some of them present solutions for one component only, meanwhile others propose applications for multi-target maintenance, considering

Table 2.2: Deep Learning for Smart Maintenance in Rail Tacks

Rail Track Component	**Smart Maintenance Task**		**Deep Learning Approach**		
	Surface Defect Detection	*Defect Inspection*	*Classification*	*Object Detection*	*Semantic Segmentation*
Rails	[157]	-	[157]	-	-
Rails Head	[112],[113], [114],[115],[116], [117],[118],[118], [119],[120]	[121]†	[112],[113], [121]†,[114], [115],[117]	[116],[119], [120]	[117], [118]
Fasteners	-	[122],[123], [124],[119]	[125]~,[124]	[122],[123], [124],[119]	-
Welded Joints	-	-	-	[126]~,†	-
Sleepers	[127]	-	-	-	[127]

$[x]^{\sim}$ only preforms type classification or object identification (no defects)
$[x]^{\dagger}$ performs tests on data collected on specimens (laboratory tests)

more than one component at the same time. Also, it is worth noting that some papers could present only approaches to object detection or classification introducing, de facto, a marginal contribution to the smart maintenance; such papers will be reported as $[x]^{\sim}$. Moreover, other papers perform tests only on laboratory data, they have been reported as $[x]^{\dagger}$.

In order to read the table correctly, it is important to note that a given task (e.g. Defect Inspection), on a specific subject (e.g. fasteners) can be performed through an object detection, a semantic segmentation, or a classification approach, or even a combination of them; for example, surface defect detection can be performed classifying the images in accordance with the labels or detecting the defect within the image. Therefore, the same reference will appear under both the "Smart Maintenance Task" and "Deep Learning Approach" macro columns. Lastly, if a paper appears under more than one sub-columns of "Deep Learning Approach", it means that it presents a multiple-stage approach, or simply more than one DL-based approaches. Approaches based on traditional Image Processing Methods are not reported.

The oldest paper we obtained from our research process, dated 2014 according to Scopus, deals with rails heads surface defect detection. Indeed, in [112], authors

implemented a CNN-based classifier training it from scratch starting from photometric stereo images (with a 16x16 shape), acquired through an ad-hoc system, in order to classify cavities or small hills on the rails' head surface. They also conducted experiments pre-training the network with weights obtained from a sparse auto-encoder trained in an unsupervised manner, showing that pre-training could improve recognition performance; moreover, data augmentation is used to evaluate both the approaches, showing that, after this step, the two networks achieve more or less the same error rate. Three years later, in 2017, a new CNN-based method to address the same task has been proposed in [113], leveraging on rails' profile obtained by processing AT C5-1600CS19-500 3D laser camera data collected on the field. The resulting CNN-based classifier has achieved 98% of accuracy on the test set, considering a binary classification task ("healthy" or "faulty" rail profile). In the same year, [121] proposed a rails' health status classification based on CNN and Acoustic Emission (AE) recognition. Practically, AE signals are pre-processed through performing the Fast Fourier Transform (FFT) in order to obtain 2D frequency spectrum matrices on which the network has been trained. Nevertheless, tests have been performed in the laboratory on a rail test specimen by stressing it until it cracks and recording audio events, labelling them as "safe" or "unsafe" according to the healthy status of the test specimen. CNN results have also been compared with MLP and SAE methods, showing higher accuracy.

Given that, the majority of the papers have been published starting from 2018. For what concern *rails*, [157] proposes a defect inspection method to validate manufactured rails in the quality assessment step. Mostly, authors presented a systematic procedure to configure CNN-based classifier evaluating different issues. Indeed, the collected image dataset resulted to be quite unbalanced, therefore different network configurations (roughly 12), in terms of the number of layers and feature vector dimensions, were tested in order to obtain a significant defect Recall (77.41 for the best configuration). The images composing the dataset were

obtained through four cameras which, with different angles, have been able to capture the whole rail surface; these images are then been labelled manually according to six defect classes. Nevertheless, the problem has been faced as a binary classification one considering defective and non-defective images.

On the other hand, different applications have been presented for what concerns *rail's heads* surface defect detection on the field. In [114], the same authors of [113] presented a three-stage pipeline to address the rails' heads' health monitoring task. The first stage aims to detect and remove the blurring effects from the images basing on an IMU unit and the AHRS algorithm; in the second stage, a CNN built from scratch is trained to detect surface defects in a binary fashion (healthy or faulty image); in the third stage, the CNN is tested on new images. A transportation device, manually driven on the rail line, has been equipped with the necessary hardware to collect data and train the CNN at the same time. Experiments showed that the whole process, from blur detection to classification, could be performed on-line at 40 frames per second which means that the system is able to analyze images collected from a device moving at 144km/h with an estimated accuracy of 97.7% (the real-time operating speed has been determined around 72Km/h). A quite similar approach is proposed by [115]. Images are collected by a robot running on the rails and a first local image processing and classification is performed onboard (on-the-fly). Once the onboard CNN architecture detects a defective image, the location is saved and the image is sent to the cloud for post-processing. The network has been tested on the Type-I dataset (authors cite [158, 159]), increasing the number of images cropping and/or rotating them, achieving an accuracy rate of 97%. Also, in [120], the authors propose an approach based on YOLOv3 architecture in order to improve the detection speed (about 0.15s), maintaining a recognition rate of 97%. Training and Test have been performed on images captured by a camera installed above the rail of the train.

Similarly, [116] proposes a new network, called MOLO, in order to perform surface defect detection on rails heads also under poor lighting conditions. It based on the YOLOv3 core idea: "turn target detection issues into target regression issues". Therefore, MOLO combined the YOLOv3 regression idea, multi-scale prediction and the loss computation method with a feature extractor based on MobileNetV2. It is capable to detect three types of defects with a Mean Average Precision (mAP) of 87.4% at 60 FPS (results show that it achieves better performances than YOLOv3 itself). Unfortunately, it seems that no information about the dataset creation is given. Such approaches were based on the concept to consider a classification process as an object detection task, relying on OD approaches. Differently, [117] and [118] focus on Semantic Segmentation Networks. The former proposes a new network, called TrackNet, in order to face the high false alarm rate issue of the traditional commercial-of-the-shelf visual-based track inspection system (COTS VTIS). Such a network implement a multi phase deep learning approach: in the first stage, a U-Net is used to extract rail tracks and locate the Regions of Interest (RoI); therefore, Image Processing tools are used to crop the portion of potential faulty region; then, such patches are fed into a CNN-based classifier which classifies them into True or False alarms. Two SOTA networks have been used as a classifier: ResNet and DenseNet, pre-trained on the ImageNet dataset. The training phase has been performed on COTS VTIS images in two modalities: in the first, pre-trained weights have been frozen, then only the last layers have been tuned; in the second, all layers are unfrozen and a typical training has been executed. This results on accuracy of roughly 90% on the DenseNet-based configuration. Nevertheless, the authors also highlight three limitations: only one type of defect is considered, the machine that labels RoI introduces unnatural noise, and the network is tested only on vertically aligned rail tracks. On the other hand, [118] proposes a network based on SegNet able to cope with different surface defects achieving a detection rate of 100%. Images are

collected through CCD industrial cameras on a testing vehicle arranged near the rail; images are then pre-processed in order to obtain a greyscale representation, reduce noise and enhance rail's defects through a histogram matching method.

For what concern rail's *fasteners defect detection*, four papers focus mostly on this task. One of them, reference [125], focuses mostly on the recognition of the rail fastening systems. Indeed, it proposes an intelligent information processing system to automatically recognize the fasteners' type between six fastening systems (e.g. DO2, KB, etc.). Images are acquired through a rail detector car equipped with four video cameras and an automated workstation that pre-processes the video stream; then, zones with fasteners are sent to the diagnostic ANN (based on VGG) that detect the type of the rail fastening systems. The achieved overall accuracy has been estimated around 97.2%. Differently, [122] proposes an architecture based on YOLOv3 network able to detect the type of fastening system and also evaluate the "faulty" or "normal" status. Images are collected through a GoPro Hero 7 Black action camera built on an inspection vehicle, also equipped with a GPS module, operating at the speed of 20Km/h during the night. Captured images were then sent to a back-end server for storing and classification. Captured images were hand labelled to train/test the network, achieving a precision of 89% and a recall rate equals to 95% on rail fasteners object (defective or not) recognition. Also, a test on the field was performed and, when a defective fastener was found, the GoPro GPX is used to mark the location of that fastener. Similarly, also in [123] authors implements a DL-based fasteners' defect detection. However, they performed such task considering UAV images (with HD cameras attached), flying at 30m on one side of the Beijing–Shanghai high-speed railway line, which also collects information about the location of the images. Authors tried different architectures for the defect detection stage based on i) YOLOv3 (with DarkNet-53 as feature extractor), ii) a modified version of YOLOv3, iii) Faster RCNN (with

VGG16 as feature extractor), and iv) Feature Pyramid Network (FPN) combined with CNN. From the performed experiments, FPN-based detection network results to have the best mAP (95.78%), whether YOLOv3 and Improved YOLOv3 based detectors results to be the fastest (a detection time of 0.01s). Also, [124] performs a comparative study in relation to the fasteners defects detection task. Images are captured manually through DLSR camera, perpendicularly to the rail track, and are related to complete, broken or missing fasteners. Fasteners are located within the image basing on traditional Image Processing techniques (e.g. wavelet transform). After this process, a dataset containing only fasteners images has been obtained. Then, three different classifiers have been presented: the first is based on Dense-SIFT, bag-of-visual-word (BOVW) model, Spatial Pyramid Decomposition and SVM; the second is based on VGG16 convolutional layers; the last one is based on Faster R-CNN. The latter configuration is also able to detect fasteners on its own without relying on image processing algorithms; this results in a faster comprehensive speed (0.23s for bot detection and classification versus above 2.20s for the other approaches). Concerning the accuracies, the first method results in a classification accuracy of 99.26%, the VGG16-based network achieved an accuracy of 97.14%, meanwhile, the latter approach achieved a mAP of 97.9%. One year later, almost the same authors proposed a new comparative study [119] but focusing on both rails head and fasteners defect detection. Images are captured in the same way described in the previous work, such as the object detection process. Moreover, the first approach proposed in this work is based on the same techniques and algorithm of the first architecture proposed in the previous one. What differs, behind the task, is the second proposed architecture called TLMDDNet (Track Line Multi-target Defect Detection Network) which is based on the YOLOv3 architecture, modifying it introducing scale reduction and further feature connections between layers. Moreover, other improvements have been introduced using Dense Blocks (proposed in DenseNet) within the TLMDDNet; this

has been called DC-TLMDDNet. This last model achieved less complexity, faster detection speed and better performances (mAP equals to 99.61% on 9 classes between normal and defective rails head/fasteners) in respect to the other two approaches.

The last two papers we found concerning rail tracks deals with rail *thermite welded joints* detection [126], and *concrete and sleeper cracks* width estimation [127]. In particular, the former reference proposes an object detector based on YOLOv3 for the welded joints detection task; nevertheless, it has been trained and tested on images collected in the laboratory on thermite welded specimen. On the other hand, in [127], the authors propose three Semantic Segmentation models based on the SegNet structure in order to predict the width of cracks in concrete and sleepers. SegNet convolutional blocks are based on the VGG16 ones, with constant dimensions for kernels equals to 3x3. Authors modified the shape of the kernels by introducing rectangular kernels to make these more compliant with a crack shape. Therefore, some blocks in the original SegNet were modified in order to create three models: the first one (CK Model 1), simply introduced rectangular kernels in some layers; instead, CK Model 2 can be seen as a serial extension of the CK Model 1 since in the same layer two sets of perpendicular kernels were used; lastly, CK Model 3, introduced kernel size variation within the same convolutional block. Tests were performed on two datasets: one with trick cracks and the other with thin ones. The first one ([160]) is composed of concrete cracks images, meanwhile, the second has been built from scratch on rail tracks sleepers. Results show that the CK Model 2, with an adequate kernel size selection, achieves better performance than the other models and the original SegNet for what concerns the width crack estimation.

Considering this first stage, based on *rail tracks*, our understanding is that

researchers are running towards DL-based approaches able to achieve always better performances in combination with an adequate training/evaluation speed in order to perform tasks in real-time. At the same time, most of the papers share the same concerns about unavailable data. Indeed, in most cases, data are collected from scratch through different kinds of systems and equipment except for concrete cracks in [127], but it is not related to rail tracks, and the Type-I dataset that the authors in [115] used to test their model for Rails Head surface defect detection. However, beyond some comparative studies, the lack of a shared dataset reflects in the difficulty to decree the best model, if any, since no benchmark performance can be evaluated. Nevertheless, some of them achieve optimal and promising results, also in terms of training speed and real-time applicability, on their own datasets. Another important result that we obtained concerns about the implementation of smart maintenance through Audio Processing. Indeed, in our research, we found only one paper ([121]) that relies on Audio Events to perform defect inspection, moreover, it analyses audio recorded in the laboratory.

Lastly, in order to answer RQ2, 3, and 4, we refer to Fig. 2.4. Different approaches have been presented leveraging on traditional Image Processing techniques to pre-process images, or even to extract features and region of interest form images (e.g. [124]). Nevertheless, we want to emphasize the usage of Deep Learning techniques to perform maintenance tasks, also understanding if SOTA networks have been involved in such processes. Therefore, in Fig. 2.4, we show a chart that correlate the maintenance task on the specific Rail Track component (SDD - Surface Defect Detection, DI - Defect Inspection), with the used DL approach (if a Classification, an Object Detection or a Semantic Segmentation one), and the specific network that authors used or inspire to; "CNN" means that the network has been built from scratch, without leveraging on neither the architecture nor the idea of SOTA ones. Lastly, as we already explained, some papers did not introduce direct contributions to maintenance, "simply" proposing classification

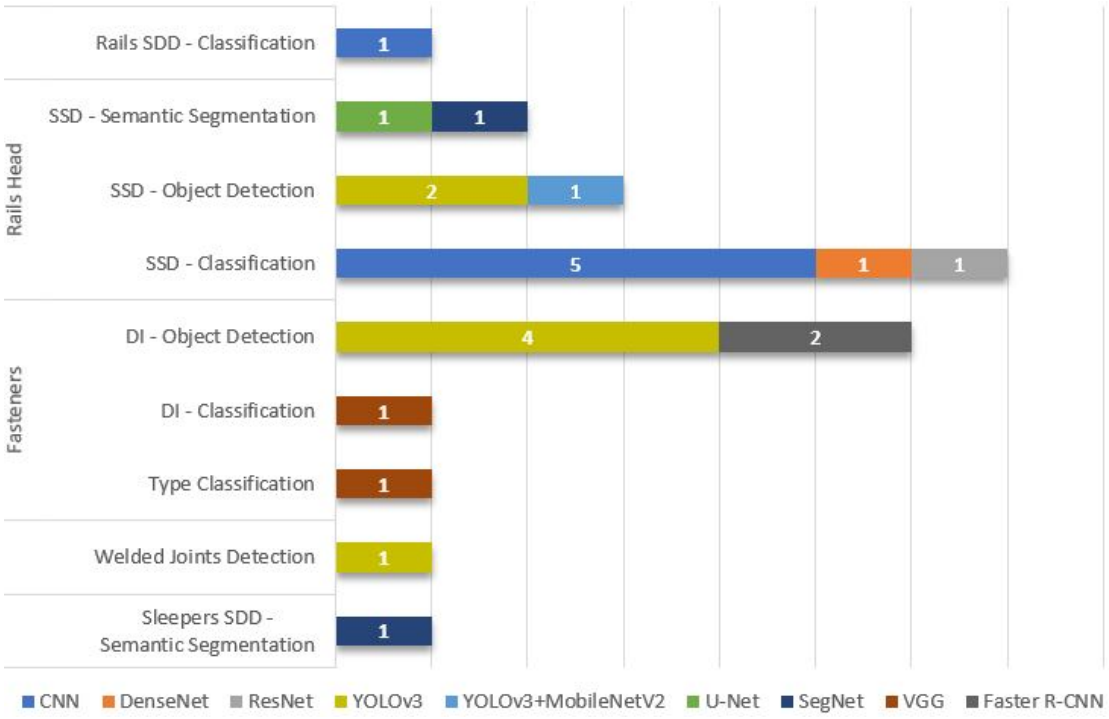


Figure 2.4: Deep Networks in Rail Track Maintenance

or object detection approaches; these are also included in the chart as "Fasteners Type Classification" and "Welded Joints Detection". Concerning the other rows, the "-" means "through", then "Rails Head SSD - Object Detection" means "Rails Head Surface Defect Detection through an Object Detection approach".

Tunnel & Subway

For this rail area, only 5 papers were obtained. Actually, we already discussed one of them ([127]) in the above subsection. Nevertheless, concrete crack analyses have been conducted on a database [160] composed of images collected through a smartphone with a constant distance of 0.2m from the concrete which origins (e.g. if from railway tunnels or not) have not been specified. Therefore, we will include it in Table 2.3 with an asterisk since it could probably be used to estimate cracks in tunnels lining, but have been trained/tested on general concrete cracks. Moreover, since the approach based on Semantic Segmentation estimate the width counting the number of pixels within it, it is clear that all the image must be collected with

the same distance between concrete and camera (as has been done).

Also in the case of Tunnels & Subways, traditional inspection methods involve human workers or some traditional image processing methods which, given complex situation (e.g. uneven illumination, image noise, etc.), results in non-optimal performances in terms of both accuracy and time-cost.

Beyond that, for this rail area, we will draft a single table only since the main focus is Surface Defect (e.g. cracks, leakages) Detection on Tunnels' Lining. It is also important to note that this is similar, if not equal, to an object detection task. In our understanding, following papers will not implement an "estimation" mechanism of the cracks (as [127] for width estimation) or defect intensity through DL approaches, they will "limit" themselves to classify/detect/locate these defects in the concrete.

Actually, [150] conducted analyses on images collected by a Kinect to estimate cracks defect intensity, nevertheless, it proposed an approach based on traditional Image Processing techniques. Indeed, after an images' pre-processing given the Kinect inclination to be easily affected by external inferences, a DL-based approach (a CNN classifier) thus used only to classify the faulting lines in vertical, horizontal, upward tilt or downward slope.

On the other hand, the other papers propose approaches able to detect more than one defect/object within tunnels images. Reference [151] proposes a two-stage Semantic Segmentation approach based on a Fully Connected Network built starting from the VGG16 architecture. In many cases, crack and leakages can overlap; therefore building a single model able to detect both of them results in poor accuracy. Therefore, the proposed system is composed of two FCNs: one capable of detect cracks, the other capable of detecting leakages. Results are compared with those obtained through other approaches (region growing algorithm and adaptive thresholding algorithm) demonstrating very higher accuracy and inference speed compared with them. Images for training and testing have been

acquired by a Moving Tunnel Inspection (MTI-200a) system, developed by the authors, which captures tunnel's surface images through 6 cameras while moving on the rails; images are then labelled and divided into a crack dataset and a leakage dataset. As reported on the ScienceDirect website, research data are not publicly available, but they could be on request[6]. Differently, [152] uses a two-stage approach (inspired by the R-FCN framework) to detect leakages, cracks ad scratches. In this case, images are collected through the Movable Tunnel Inspection (MTI-100) system and two datasets were built. The first dataset is used to pre-train the Fully Convolutional Network (based on GoogLeNet and VGG16 core ideas) composing the fiest stage of the model. This network has been considered as a traditional classifier (with a last fully connected layer) and has been trained to classify images according to five classes: leakage, crack, segment joint, pipeline and lining. Such network showed better classification accuracy in relation to the classical AlexNet, GoogLeNet and VGG16 (the former did not converge). Once the proposed network has been trained, the last FC layers have been dropped and replaced to obtain the last feature map used as input for the next stage. Including RPN, position-sensitive RoI pooling, softmax and bounding box regression, such stage performs object classification and localization. Proposed approach achieved a detection rate of 94.4%, an accuracy of 86.6% and an efficiency of 0.266s, showing optimal performances also compared with Faster R-CNN and traditional methods. Finally, some test were performed to understand the correlation between the size of the input images and the performances; obtained results showed that, although the detection accuracy was almost the same, the location accuracy increases with smaller images. Lastly, reference [153] proposes a Semantic Segmentation based approach to detect cracks in tunnels' lining referring to the DeepLab-v3 framework. Authors built two datasets starting from the images captured by a proprietary im-

[6]https://www.sciencedirect.com/science/article/abs/pii/S0886779817310258?via%3Dihub#ec-research-data

Table 2.3: Tunnel's Lining Surface Defect Detection Approaches

Paper	**Purpose**	**Deep Learning Approach**		
		Classification	*Object Detection*	*Semantic Segmentation*
[127]*	Crack width estimation	-	-	- SegNet
[150]~	Crack orientation identification	CNN	-	-
[151]	Cracks and Leakages detection	-	-	VGG16
[152]	Cracks , Leakages and Scratches detection	-	R-FCN (GoogLeNet +VGG16)	-
[153]	Cracks detection	-	-	- DeepLabV3 (ResNet-18)

[x]* relies on data not related to the railway environment
$[x]^{\dagger}$ performs tests on data collected on specimens (laboratory tests)

age acquisition system equipped with eight industrial linear array CCD cameras. Datasets were annotated manually (authors assert that this is the first annotated dataset for tunnel cracks, but it is not totally clear if it is available or not): the first contains only cracks (tunnel crack dataset), the second contains cracks, structural seams, water stains, and scratches (augmented tunnel crack dataset). The built system relies on ResNet-18 as encoder; experiments show that the IoU on the first dataset is equal to 0.371, meanwhile, in on the second dataset (considering only cracks) it achieves 0.408. To increase performances, authors weighted the labels giving a factor of 4 for cracks, meanwhile, the others are left to 1; this resulted in an IoU of 0.421 (the inference time was estimated to be 51.2ms). Other experiments were performed leveraging on MobileNet-v1 (IoU 0.464, speed 42.6ms) and CrossNet (IoU 0.452, speed 44.7ms) as decoders.

What has just been discussed is summarized in Tab. 2.3. For each paper, the purpose and the DL approach have been indicated. The cells contain the particular architecture/framework/network to which the proposed model is inspired; "CNN" means that the network was built from scratch.

Catenary & Pantograph

The Pantograph-Catenary system has a central role in electrical trains, indeed, they provide them with the necessary power to operate. Therefore, a correct maintenance routine is vital for the correctness and safety of railway transportation. As for Rail Tracks, also for the power supply system, traditional inspection methods rely on human workers, which involve the same issues of poor accuracy and time-cost. Moreover, catenaries and pantographs are composed of a lot of "small" components which are difficult to detect (e.g. fasteners, insulator, dropper, etc.). Since that, different papers demonstrated that applying a defect detection approach directly on the whole image, will result in a poor accuracy as well. Indeed, as we will see, many of them proposed a multi-stage approach which first extracts/detects the interesting component from the original image and then performs defect inspection analysing only these image's patches.

The catenary is the fixed infrastructure located near, and upon, the rail track; on the other hand, the pantograph is a train component, placed on the roof. The power to the train is provided through the contact between the pantograph slide plates and the catenary contact wire.

Since there are a lot of components (including screws, wires, etc.) we cannot subdivide papers dealing with each of these. Therefore, in the following analyses, we will classify papers according with three Catenary-Pantograph system sub-areas: *Pantograph and Arcs*, *Catenary Support Device* (CSD) and *Catenary Wires*. The former will encompasses all those papers that deal with pantograph's defect inspection or arc[7] detection; the latter will encompasses all those papers that deals with wire (e.g. droppers, contact wire, message wire, etc.) defects or detection. On the other hand, the second one is a real sub-structure of the Catenary system that supports wires above the railways. Such structure includes a lot of other

[7]An arc can be trivially defined as an abnormal sparkling event that occurs between the pantograph slide plate and the contact wire

Table 2.4: DL for Smart Maintenance in Catenary & Pantograph

Catenary & Pantograph sub-area	**Smart Maintenance Task**		**Deep Learning Approach**		
	Surface Defect Detection	*Defect Inspection*	*Classification*	*Object Detection*	*Semantic Segmentation*
Pantograph and Arcs	[145]	[139],[141], [142],[135]	[145]	[132]~,[139], [141],[142], [135]	-
Catenary Support Device	[138]	[137],[133], [144],[135], [140]	[138],[133], [135]	[126]~,[138], [137],[131]~, [130]~,[133], [144],[135], [129]~,[140]	[138],[137], [144]
Catenary Wires	-	[143],[134], [136]	[136]	[128]~,[143], [134],[136]	-

$[x]^{\sim}$ only preforms object identification (no defects)

components (e.g. insulators, brace sleeves, brace sleeves screw, steady arm, etc.) that, as already indicated above, are quite small in respect to the whole image; clearly, we placed in this category all those papers that deals with defect inspection of such components. In Table 2.4 we classify, as we have done for Rail & Track, the 19 resulted papers according to the aforementioned sub-areas and the Smart Maintenance task they address.

As we mentioned at the beginning of this chapter, we are mostly interested in maintenance application through DL-based approaches which rely exclusively on audio and video. Nevertheless, among the results obtained for this rail area, we got the reference [132] which, moved by the difficulty in acquiring clear images and an adequate number of insulator defect samples to properly feed a deep network, only detects insulators through a DL-approach, meanwhile, the defect detection is performed through traditional Image Processing algorithms (Contour Features and Gray similarity matching). However, even if only in part, it relies on DL to approach smart maintenance, therefore it has not been excluded (given E2). Moreover, also [129] relies on image processing techniques. First, images acquired on the field concerning catenary suspension device in metro power supply systems are pre-processed. Then, interesting areas (insulators and bolts) are cropped using

the template matching algorithm based on grey value. Once that, a SURF-based (Speed Up Robust Feature) BOF (Bag of Features) model is used to detect defects. Also, an AlexNet-based detection method is presented to further improve the detection efficiency. Similarly, [128] implements an approach to detect droppers in high-speed railway improving the Faster R-CNN framework by involving ResNet as basic feature extractor, introducing a balanced attention feature pyramid network (BA-FPN) and a mixed attention block to obtain the fusion feature of multilevel feature maps, and a center-point rectangle loss (CR Loss) function to accelerate the convergence of the model. The model has also been evaluated on VOC2012 and MS-COCO 2014 datasets achieving state-of-the-art performances; then, it has been tested on a dataset collected from the high-speed rail 2C system achieving, also in this case, optimal performances compared to others SOTA frameworks/networks even if it is not so suitable with real-time detection (6.3-7.9 FPS). Lastly, reference [126] presents an approach based on the Faster R-CNN in order to detect brace sleeve screws (a little screw belonging to the catenary support system). Faster R-CNN has been improved introducing two concepts: discrimination feature maps and Proposal feature maps. Then, VGG16 and ResNet-101 have been tested as feature extractor relying on the Catenary-500 dataset. It has been collected through the GJ-2 inspection vehicle and 12 objects (brace sleeve screw included) within images have been manually labelled using the LabelImg tool [161]; nevertheless, it is not clear if the dataset is publicly available or not. The ResNet-based configuration seems to be slightly better than the VGG16-based one, both in terms of mAP and AP on brace sleeve screw, even if a little bit slower.

Actually, also the "simple" object detection can be seen as a form of smart maintenance at the time when it locates visible defects within the images, or detect particular events, as happened for crack detection in the previous section. In this respect, reference [139] proposes an Arc detection system based on CNNs. Indeed, the constant contact between the pantograph friction bands and the catenary

contact wire can cause erosion on one, or both, the components; arcs, according to the authors, are the most important reporters of such wear and tear phenomenon. Images were obtained from video frames captured by a camera located on the top of the train pointing to the catenary-pantograph contact point. For the same purpose, in [141], authors propose a Faster R-CNN based architecture able to detect both pantograph and arcs within high-speed train lines. The Faster R-CNN architecture has been modified relying on the fact that the camera was fixed on the roof of the train, so there is no need to predict region proposal with different aspect ratio; moreover, the ZFNet has been used as training network in order to address the processing speed. Then, arch detection is performed relying on the picture of the pantograph area. A four-step alternating training process is performed relying on ImageNet to pre-train the networks. Also, the proposed system involves the height of pantograph calculation (to detect a bow drop) and a SIFT-based approach to geographically locate the fault on the railway line. Before the training, parameters are initialized by a model trained on the VOC2007 dataset, then tests have been done on images related to different environmental conditions; comparing the results with other SOTA frameworks, the proposed model achieved the best results in terms of detection average precision both on pantograph and arcs detection (up the 99%), even if YOLO remains the fastest network. Other tests have been performed to show the fault localization feature. Similarly, adopting an object detection approach, [142] proposes a two-stage approach based on SSD with a VGG16 base structure to detect pantograph horns. In the fists stage (called circle), the detection through the SSD model is preformed; then results are also sent to the second circle which, through the detection of the pantograph head skeleton, is able to improve the whole detection score. Images were acquired by four cameras, two located on the catenary structure, two on the roof of the train. Experiments show that fusing the two cycles results, the mean IoU actually improves. Through horn detection, the system is also able to detect faults as pan-

tograph loss and divation. Lastly, for what concerns the "Pantograph and Arc" sub-area, [145] proposes an approach to estimate the wear of Pantograph Slide. During the running, the Slide Plate constantly slides, indeed, over the contact wire; this certainly causes wear over time. Therefore, authors propose a CNN-based approach in order to classify the sliding plates according to their defect (normal, over abrasion, partial abrasion, and grove-shaped abrasion). Images for training and testing are collected by a specific system on the metro line that acquires images when triggered by the train; nevertheless, such images are mostly related to "normal" pantographs, therefore other defective images are collected with a handhold DSLR camera. The proposed network, called PDDNet, achieves the best overall accuracy (90.63%) compared with other SOTA networks (AlexNet, LeNet, and an optimized version of AlexNet). Nevertheless, such authors highlighted, it is not sufficient to classify defects only, it is also necessary to quantify them (the thickness of the plate). Therefore, they also present a procedure, based on traditional Image Processing methods, to estimate the thickness over the pantograph's slide.

Concerning *Catenary Wires*, the *droppers* are the components that have caught the attention of researchers. Droppers act as powerful connectors between the message and the contact wire; their failure (e.g. bending, break, deformations, etc.) can lead to unpleasant consequences such the burn out of the contact wire, because of possible short-circuit, damage to pantographs, since droppers can disentangle with them during the train run, and also serious accidents [143]. Therefore, droppers' monitoring and defect inspection are two important maintenance topics in the railway sector.

In [143], authors propose a complex system for droppers defect inspection. Actually, it subdivided possible defects in (evident) faults and (tiny) defects. In order to detect the first ones (e.g. broken droppers), an object detector is proposed based on Faster R-CNN and ResNet-18 as the feature extractor. If a dropper is

faulty, the system will classify it as such; otherwise, if identified as "normal", it will be analysed by a second system, based on traditional Image Processing algorithms, in order to identify tiny defects (e.g. little foreign objects, broken strands). Results show that the fist system has a detection accuracy of 99.98%; meanwhile, the accuracy of the whole system exceeds 95%. Moreover, the whole system performs fault-defect detection in 0.2s per image. The same task is addressed by [134]. Here, authors propose a two-stage detection system: in the first stage a light version of YOLOv3 is used to extract only regions with droppers (called slings); then, a Faster R-CNN based network is used to detect foreign objects or hard bending (with a precision rate of 90%), in parallel with another block, based on traditional Image Processing technologies (OTSU, digital morphological operation and self-define linear difference detection), which aims to detect non-stress defect (with a precision rate of 80%). Also [136] addresses the dropper defect inspection task. Since the droppers can be easily obscured by others catenary components, resulting in poor detection accuracy, authors propose a three-stage method: in the first stage, a CenterNet [162] based object detector (with FPN and DLA [163] network) is used to detect current carrying rings (connection rings between droppers and contact/message wires); then, if they are classified as "healthy", the dropper region is cut leveraging on the top and bot rings positions; finally, a ResNet-34 is used to classify the droppers (normal or faulty). Among the tested classifiers, ResNet-34 achieved the best accuracy (99%) and speed (12ms) performances. In all the three just mentioned papers, data are acquired through inspection vehicles, nevertheless, their model is not clear.

Then, different approaches have been presented to perform defect inspection on *Catenary Support Device* components. In [131], authors propose a comparative study between four frameworks to perform catenary support components detection. In particular, they compared YOLOv2, SSD, and the Faster R-CNN framework both with VGG16 and ResNet-101 as a feature extractor. Training

and tests were made on a dataset collected by high-resolution cameras mounted on the roof of an inspection vehicle, then 12 categories (objects) were manually annotated. Such categories encompass insulator, brace sleeve, steady arm base, messenger wire base, etc. Results show that the Faster R-CNN with ResNet-101 had the best performances in terms of mAP (0.797); nevertheless, none of the tested architecture achieved good precision on small components (e.g. the best AP for brace sleeve screw has been estimated about 0.5), underlining the difficulties of such frameworks in extracting the features of small-scale targets. After that, they propose some fault detection methods to evaluate the status of those components dividing them into two categories: common image processing methods, and deep learning methods. Without going deeper with details, we will focus only on the papers that came out from our research too. Reference [130] proposes a method based on the Markov Random Field model to perform loose strands diagnosis in isoelectric lines. Data were collected during the night by the JX-300 inspection vehicle, then an architecture (called ILNET, Isoelectric Line Network) have been built relying on Faster R-CNN and ZFNet in order to detect isoelectric line areas. Therefore, the ICM/MPM method to perform image segmentation to further identify loose strands. On the other hand, in [133], a three-stage model completely DL-based has been proposed to detect fastener defects on the catenary support device. The first stage involves a modified version of the SSD framework to detect areas related to the interesting objects; then, in the second stage, a light version of YOLO (with fewer parameters) has been built in order to extract fasteners/clips from input images (the areas obtained in the previous step); finally, in the third step, a CNN is used to classify defects in relation to the fasteners images. For each step, several architectures have been tested showing that the ones selected achieve the best performances.Data for training and test were acquired thorough the XLN4C-01 inspection vehicle during the night. As also mentioned above, a multi-stage approach has been adopted in different cases. Reference [138] proposes

a two-stage method to detect surface defects in Catenary Insulators. In the first stage, insulator regions are detected through a Faster R-CNN (based on VGG16) and are cropped from the images. These images are then analysed by a second architecture which combines a deep denoising autoencoder (DDAE) and a deep material classifier (DMC, a segmentation approach inspired to [164] and [165]) which are trained through a multi-task learning approach, sharing the firsts convolutional layers. The surface defect detection is obtained combining the output classification score of the DMC and the anomaly score determined by the DDAE. The model has been tested on images captured by the KCIS-01 catenary inspection vehicle, showing optimal performances (an F1-score of 95%). A few months later, almost the same authors propose in, [137], a semantic segmentation based approach to perform defect detection in contact wire clamps (CWS) and Split Pins (SP), two components of the Contact Wire Support (CWS) system. The method involves two stages: in the first one, a Faster R-CNN based object detector recognize the RoIs related to the two objects above; then, such areas are cropped and analysed in the second stage by a catenary components segmentation network (CCSN) inspired to the Mask R-CNN architecture. CCSN involves a ResNet-50 as backbone (instead of an FPN) architecture and two Bayesian neural networks as mask branches which involve Monte Carlo dropout to obtain model uncertainty. Beyond implementation aspects, the defects are detected evaluating the geometric characteristics of the segmentation masks. Such network, compared with other segmentation approaches (DeepLab, U-Net, and Mask R-CNN), obtained best performances on both segmentation accuracy (a mIoU of 90.1& for CWC and 89.6% for SP) and, consequently, defect detection accuracy (an F1-score of 95.5% for CWC defects and 98.2% for SP ones). Data for training and testing has been collected by the XLN4C-02 catenary inspection vehicle. Moreover, [144] presents a three-stage method based both on object detection and semantic segmentation to detect defects in catenary Split Pins (SP). Since Split Pins are so small in re-

spect of the whole image, an end-to-end classifier could achieve bad performances. Therefore, a first improved YOLOv3 network is adopted to detect interesting regions, the ones that contain SP; then, the semantic segmentation is performed on those areas through a model based on an improved version of DeepLabV3+; lastly, defects are classified by evaluating the segmentation shapes (missing pins are detected directly in the second stage, on the basis of the segmentation colour). For all the three stages, comparisons were made with other SOTA frameworks/networks and image processing based algorithms. Results show that the considered architectures achieved the best performances. Data for training/testing were collected by HD cameras mounted on 4C detection vehicles at night and manually labelled for both the object detection and segmentation stages; in order to increase the applicability of the model, the datasets contain catenary images of different lines and different environments.

Then, other kinds of architecture have been presented to solve defect inspection task for CSD components. [140] presents a pyramid fusion architecture to detect defects in catenary insulators (bird preventing) and fasteners, relying on ResNet-101 as backbone network. It has been tested and compared with other frameworks (YOLO, Faster R-CNN with various feature extractors, etc.) on VOC2007 (railway dataset), and on a dataset composed of images collected by an inspection vehicle on the high-speed railway line between Beijing and Shanghai (images were manually labelled including bird-preventing, broken and loosening pins). On both the dataset, the proposed model achieved the best mAP, 80.1% on the former and 81.2% on the latter. Lastly, [135] proposes a Spatial Transformer And Bilinear Low-Rank (STABLR) model to detect defects in the catenary system. Practically, it is composed of two stages: in the first one, a Spatial Transformer Network [166] is used to learn the invariant representation of the original image and locate the target object at the same time; then, two parallel networks are used to extract features on which classification is performed. Different couples were tested, the

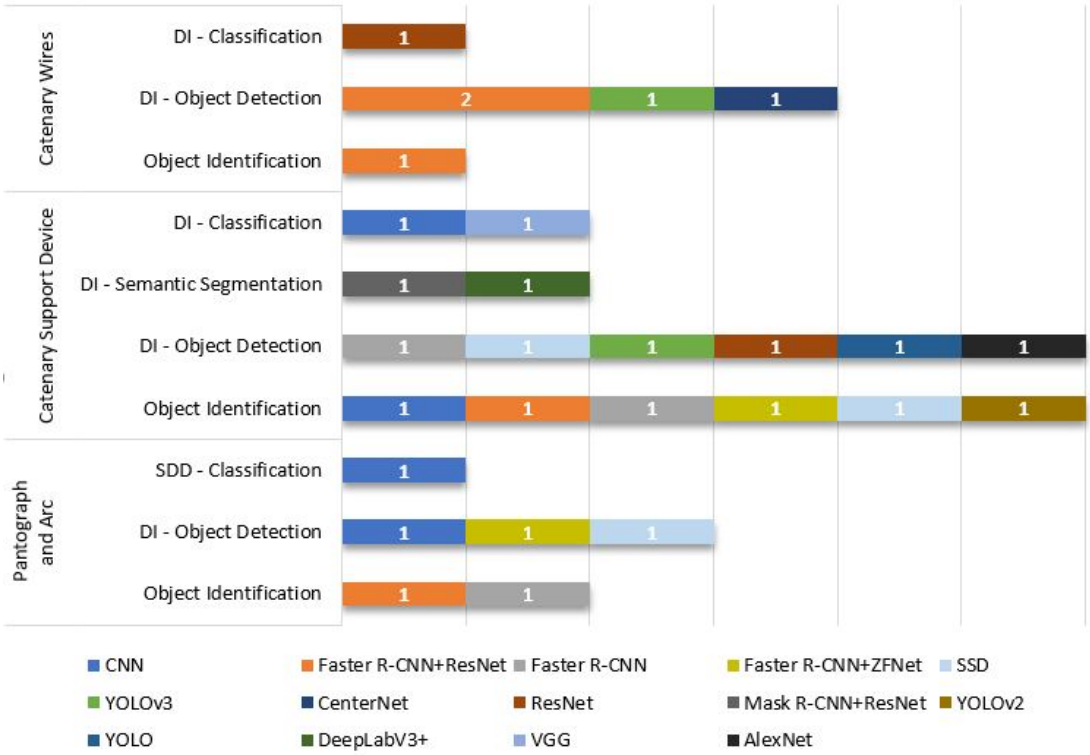


Figure 2.5: Deep Networks in Catenary & Pantograph Maintenance

VGG16+VGG19 couple obtained the best accuracy (94.09%). The tests were conducted on data collected by the Sydney Train Maintenance Center and contain images related to five components and relative defects, 10 classes in total (splice, insulator, knuckle, and two types of dropper).

Downstream this analysis, also in this case researchers mostly built datasets from scratch or relied on data that are not public. Indeed, only two datasets came out, the VOC2007 (railway dataset) and the one used in [151] that seems to be available on request. On the other hand, if for the Rail Track area we obtained readable results about the SOTA network/framework used in the various implementations, in this case, since the approaches are more articulated and complex, we obtained some sparse results as showed in Figure 2.5; it is important to note that "Object Identification" stands for applications oriented to target detection only, without introducing a direct contribution to the maintenance process.

The small-scale object detection has introduced some difficulties, since traditional frameworks and networks did not perform well without any improvement, especially for Catenary Support Devices' components [131]. Therefore, researchers have experimented different architectures, some of which go behind the "simple" modification of SOTA frameworks, or the consideration of multiple-stage systems, such as the two last reference: [140] built an architecture in order to leverage on more than one feature map in the defect detection process, i.e. the output of four blocks in the ResNet-101 backbone networks are fused in order to perform detection, meanwhile, [135] uses two VGG networks in parallel to extract features. Moreover, also other kinds of DL approaches have been used, as Deep Denoising Autoencoders (DDAE) in [138], to perform surface defect detection in catenary insulators.

Finally, it is worth noting that, in case of multi-stage approach, YOLO and Faster R-CNN based architecture have been mostly used in the first stage just to detect the area of interest, i.e. the area in the whole image containing the component of interest; the defect of the component under examination have then been detected in the next stages.

Train Bogie & Frame

The Train Bogie is a complex component with many elements that could also provide train traction and brakingOn the other hand, by Frame (chassis) we mean all the underlying part of the train, also including bogies, which allow the train to run correctly and safely. Therefore, an appropriate maintenance routine is mandatory to ensure train safety avoiding serious accidents. As for the above areas, inspection tasks have usually been performed by skilled workers supported, or not, by traditional vision systems resulting in low detection and poor efficiency. To address such issues, in the last years (2019 and 2020), researches have started to test DL-based applications, also trying to implement online monitoring systems.

Table 2.5: Defect Inspection Approaches for Train Bogies and Frames

Paper	**Purpose**	**Deep Learning Approach**	
		Object Detection	*Semantic Segmentation*
[146]	Freight Train Bogie Components Object and Defect Detection	Faster R-CNN	-
[147]	Real-Time Freight Train Bogie Components Object and Defect Detection	Faster R-CNN	-
[148]	Freight Train multiple defect detection	ResNet-101	ResNet-101
[149]	Bolt-Loosening detection in train's bogie	CNN	CNN

As for the *Tunnel & Subway* area, also in this case we will draft a single table since all the papers deal with small (in relation with the whole train) object/defect detection within Bogies and Train Frame. Therefore, in Table 2.5, for each paper the purpose and the DL approach have been reported, also indicating the particular framework/network on which authors relied.

The main issues when facing defect detection for train freight components, according to [146], reside in a low images quality and the complexity of the background of the images. These are composed of too many small objects, therefore is difficult to achieve fast and accurate fault detection. Authors propose a defect detection framework based on Faster R-CNN which involves a VGG16-based feature extractor and a Multi Region Proposal Network (MRPN, instead of the traditional RPN of Faster R-CNN) inspired by the idea of inception module in GoogLeNet and HyperNet; the shared convolutional layers are pre-trained on ImageNet. The core idea is to slide a set of anchors over multi-level convolutional feature maps. The model was tested on four datasets (authors cite [167], nevertheless, it is not accessible to us) which are related to 1) Cut-out cock handle, 2) Dust collector, 3) Fastening bolts, and 4) Bogie block key. The proposed model showed the best correct detection rate (100%) on databases 2 and 3 compared to others approaches (e.g. Faster R-CNN+ZFNet, Faster R-CNN, SSD+VVG16, etc.) and a false detection rate of respectively 0.35 and 0; for what concern databases 1 and

4, although not achieving the best detection performances, the proposed model has the lowest false detection rate (0.47 and 0). This model has been then modified and improved in [147] for real-time fault detection introducing a novel multilevel feature fusion strategy based on the MRPN and MRoI pooling, furthermore, they introduced a model reduction scheme (MRS) in the top layers of the fault detection network in order to reduce model complexity and improve speed. They also proposed an image acquisition system composed of a device placed in the middle of the rails, and two devices on the rails' sides, composed of CCD cameras and an auxiliary light device. When a train passes through, images are acquired and sent to a server system for the analyses. The proposed model has also been tested on the same datasets presented above and a new "Angle cock" dataset, showing great performances, low resources requirement and fast processing speed (0.058s per image) also compared with other frameworks (e.g. SSD+VGG16, R-FCN, Faster R-CNN, etc.). Another multi-defect (lost pin, lost bolt, lost rivet, broken chain, broken wire, and foreign object) detection system for freight trains has been proposed by [148]. Authors built a three-stage model: in the first stage, a ResNet-101 based network designed to extract hierarchical features; in the second stage, a RPN is involved to detect and classify coarse regions, basing on the multi-scale features; lastly, a FCN, also based on ResNet-101, makes instance-level predictions on coarse defect regions. The dataset used to train the network was collected by a Chinese company through 12 high-speed camera placed on the tracks (under the train); unfortunately, it seems not available. Performances have been evaluated and also compared with other approaches (including also the one proposed in [146]), showing the highest overall performances (80.48% recall, 78.20% precision, and 79.32% F1 score). Moreover, a transfer learning approach has also been tested, fine-tuning the network in order to detect oil leaks and scratches; also there, great results were achieved (89.7% recall, 86.37% precision, and 88.00% F1 score). Differently, reference [149] mostly focuses on bolts. It proposes a system for bolt-

loosening detection in key bogies component (Axle Box, Traction Motor and Gear Box) collecting data in an online fashion, i.e. while trains are running. Indeed, data are acquired through two binocular systems installed on both the sides of the rail tracks which are triggered at train passing. First, a CNN-based system detects regions of interest (the ones with bolts) in the components mentioned abobe; then, another CNN extracts bolts edges; lastly, to detect single or multiple bolt loosening, a 3D reconstruction method is used to calculate the distance between the bolt cap and the mounting surface. The first CNN-system is trained and tested on images captured by the proposed binocular system; meanwhile, the CNN for edge detection have been trained on the BSD500 dataset [168]. Results show optimal performances with a relative error less than 1.42%, moreover, the processing time to complete fault detection considering a train with 8 cars is about 4.6 minutes, 1.1s per image (within the China's requirement of 5 minutes). Nevertheless, the surface must be clean for proper analysis.

Other Areas

This category includes three papers, two of them are related to the EMU Train key components defect detection, meanwhile, the last is related to the detection of faulty rail valve. They are summarized in Table 2.6.

By EMU Train key components, [155] and [156] mean brake disk, brake caliper, tractor, side suspension, under suspension, and plate bolt. Such elements are significantly different from those of the "traditional" trains. Therefore, the authors propose a system able to cope with EMU components defect detection called

Table 2.6: Other Areas - Defect Inspection Approaches

Paper	**Purpose**	**Deep Learning Approach**		
		Classification	*Object Detection*	*Semantic Segmentation*
[154]	Faulty rail-valve detection	-	-	U-Net
[155],[156]	EMU Key Components defect detection	Inception-ResNet-v2	Faster R-CNN (ResNet-101)	-

MPDD. The structure is based on the Faster R-CNN framework, nevertheless, the feature extractor is based on ResNet-101 (instead of VGG16). Moreover, extracted regions are cropped and processed in order to obtain "super resolution images" thorough RAISR, SRGAN (image resolution increment through Generative Adversarial Networks) and biquadratic interpolation. Then, a classification network based on Inception-ResNet-v2 is used to classify defects. Then, the output of the first network and the last network are combined to both locate and classify the defect. Data are collected on the field by EMU inspection system (TDES in [156]). From the whole dataset, composed of 12 classes between normal and defective components, a second dataset composed of plate bolts only has been extracted in order to verify the super-resolution enhancement contribution to the classification performances. Indeed, according to the authors, the optimal performances achieved by the proposed method (79.2% mAP), roughly 12% higher in respect to the best results of the other tested methods, are due to such enhancement and the ability to detect defects also in small objects. Lastly, transferring potential are also evaluated testing MPDD on the Northeastern University (NEU) surface defect dataset; it achieved the best performances also on this dataset. Nevertheless, the detection time is higher with respect to the other frameworks.

On the other hand, reference [154] proposes a faulty rail-valve (traditional trains) detection through a two-step segmentation approach named FaultNet. Leveraging on U-Net, authors build a two-stage segmentation system. Images are collected through a VTIS system which allows an image collection always from the same distance; therefore, the valve has roughly the same dimension in all the images. The first network performs segmentation on the whole image, detecting the segmented mask which highlights the valve; then, both the mask and the image are cropped in order to obtain only the regions in which the valve appears. These are then processed by the second network that performs a High Resolution

Segmentation. The mask in output is then processed by an image processing algorithm to detect the fault. The two-stage segmentation network is also compared with U-Net on the SMRT dataset showing the best performances; meanwhile, the whole system has been compared with ResNet and DenseNet, trained on cropped images and full images, showing the best accuracy (97.26%). Nevertheless, the system can work only with train-valve binary classification and only if there is a single valve per image oriented in a specific way.

2.2.3 Discussion

All the papers mentioned in the above SLR implement, in some way, a DL-based approach to perform or facilitate maintenance operations. However, there are also approaches based on traditional Image Processing techniques both as regards image pre-processing (e.g. image enhancing, noise reduction, etc.) and as regards image segmentation [124] and defects' evaluation [117, 143, 134, 130, 154]. Nevertheless, it has been assessed that approaches entirely based on IP are limiting in unfavourable conditions such as those present in tunnels (e.g. uneven illumination, image noise, etc.).

On the other hand, many object detection approaches have been implemented. Both for Rail Tracks and Tunnels, but especially for Catenaries. In fact, these are composed of very different components, even very small, such as to make a direct defect detection on the captured images inefficient. Therefore, in the latter case, different multi-stage approaches were evaluated in which the first stage provided for the detection of the area related to the specific component of which to assess the status. Among the various object detection methods, the most used frameworks were those belonging to the YOLO family (mostly Yolov3) and those region-based family (mostly Faster R-CNN). Such frameworks have been used both as defect detectors, evaluating defects starting from the original images, and as first-stage

architecture in the multi-stage approaches, in order to detect the region related to the component of interest.

Finally, concerning datasets, we noticed that in most cases data have been collected from scratch through different kinds of systems, hand-labelled and private (this is probably also due to disclosure agreements with data providers). However, in some cases, external datasets have been mentioned on which models have been trained and/or tested. In the following we will mention all these datasets according to the rail area of interest:

- Rail Track: in [115] the Type-I dataset [158, 159], which consist of images related to surface defects on rails heads, has been used to validate the model.

- Tunnel & Subway: in [127], the proposed architecture has been trained on the [160] dataset, although in our understanding these are related to generic concrete structures cracks. On the other hand, [151] builds a dataset from scratch related to defects on tunnels lining from a proprietary acquisition system, data seems to be available on demand[8].

- Train Bogie & Frame: [149] relies on the BSD500 dataset [168], to train the bolt edge detection model; vice versa, [146] tests the proposed architecture for defect detection on four datasets presented in [167], however, these do not seem to be accessible.

- Catenary & Pantograph: the VOC2007 railway dataset has been used by [140] to test the proposed model for defect detection of insulators and compare it with other architectures; meanwhile, [126] refer to a dataset, the Catenary-500, used to test the proposed model for the detection of brace sleeve screws, however, even in this case, it is not really clear whether the dataset is available or not.

[8] https://www.sciencedirect.com/science/article/abs/pii/S0886779817310258?via%3Dihub#ec-research-data

The outlined SRL shows a strong deficiency in the use of audio-based techniques to carry out maintenance activities. In fact, among the 48 articles that have been evaluated, only one ([121]) tries to lay the groundwork for possible use of audio signals for maintenance operations through laboratory analyses. This "scarcity" of results, if on the one hand it is linked to the small number of digital libraries we considered, on the other, it is probably due to the fact that when dealing with audio data tasks are more related to Audio Detection or Classification. Indeed, typical tasks are *acoustic scene classification*, i.e. define a single label to associate with a single audio, *acoustic event detection*, i.e. which aims to individuate the start and the end instants of an audio event, and *tagging* (or polyphonic event detection), i.e. associate to a single audio multiple labels related to concurrent audio events [80].

Different studies can be found in the literature about the application of convolutional networks for environmental sound classification [169, 170], recurrent neural network for polyphonic sound events detection [171], large-scale audio classification [87], etc. Moreover, among the others, in 2013 the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge [172] was created in order to encourage researchers to investigate solutions for tasks such as acoustic scene classification and detection of sound events within a scene; in the last year more tasks have been added (e.g. urban sound tagging, sound detection in the domestic environment, etc.)[9]

Therefore, considering the outcome of the Systematic Literature Review, and leveraging on a VGG-based network (VGGish), which in turn is inspired by the results obtained in [87], we will present an Audio Classification network able to distinguish LC Alarms form similar sounds laying the groundwork to the realization of the Alert Bell Module described in section 2.1.2.

[9]http://dcase.community/challenge2020/task-sound-event-detection-and-separation-in-domestic-environments

Chapter 3

Level Crossing Alarm Classification

When facing a problem, a good starting is to analyze it from different perspectives to find a suitable solution. Such points of view include also the nature of the problem and the data availability.

When dealing with Deep Learning, or Machine Learning in general, data availability is one of the greatest issues to face. This problem is also aggravated by the fact that, in many cases, a very large amount of data is required to properly train a ML/DL model. Even if this issue can be bevelled involving approaches such as Transfer Learning (section 1.2.1), without data no analyses are possible.

Data unavailability could be due to the following reasons: data have not been collected yet or such datasets are not public. In both cases, data must be totally or partially collected from scratch.

Therefore, the first step we performed has been the dataset creation. To the best of our knowledge, no dataset about Level Crossing Alert bells has ever been built; on the other hand, different general audio datasets are available online.

Since we decided to face our problem as a Binary Classification task, the dataset we made is composed of samples belonging to two different classes: Level Crossing Alert bell class (LC Alarm) and Other Audios class (NoLC Alarm). The latter consist of audios belonging to different classes, quite similar to an LC Alarm,

which are available on the *AudioSet* website[1]. These choices have been made since distinguishing two similar audios could be more tricky than assessing whether there is a sound activity in a specific instant or not. The chosen sounds, that belong to the NoLC Alarm class, are those that can be easily found near the Level Crossing or that are quite similar to an LC Alarm: sirens, various kinds alarms, bells, etc. The latter have been considered mostly because Italian LC Alarms are quite similar to these sounds, as UK ones are quite similar to sirens and other kinds of alarms.

In the following sections, the whole process of data collection/dataset creation, the proposed network with the related considerations in terms of data preprocessing and post-processing, and the obtained results will be addressed and described. However, it is important to note that this work have been performed with the aim to prove that the use of Deep Learning techniques could be a suitable choice to solve our task. Further implementation and improvements must be made in order to both achieve better performances and use the system to properly evaluate the Level Crossings' activities.

3.1 Dataset Construction

When dealing with Deep Learning applications, one of the most complex and time-consuming steps is *data acquisition*, since gathering data could be not always straightforward. In some situations, data must be collected on the field with the possibility to interfere with the operations of the system under examination. Moreover, data acquisition could be time expensive since the collection of an adequate amount of data depends on the occurrence of the events. There are cases in which events of interest are extremely *rare* or are poorly predisposed for acquisition by nature.

[1]https://research.google.com/audioset

Fortunately, when it comes to audio and video data acquisition, such complications are limited by the fact that a microphone or a camera would not affect operability, indeed they can be installed outside the system. Moreover, thanks to platforms as YouTube, in some cases it is possible to collect real data without specialized experimentation.

As mentioned above, to the best of our knowledge, no dataset about Level Crossings Alert bells is available in the literature, otherwise, a huge audio dataset that includes a multitude of sounds is available on the AudioSet website.

3.1.1 AudioSet

AudioSet[2] is a collection of about 2 millions hand-labelled audio clips extracted from YouTube videos, most of which have a duration of 10-seconds. The dataset includes very different classes of audios such as human, animal, environmental, etc., organized according to a specific ontology [173]. It considers about 632 classes, but, to the best of our knowledge, "only" 527 are provided with the dataset.

The whole dataset has already been split by the authors into three sets:

- Evaluation Set: which is composed of about 20 thousand segments and provides at least 59 samples for each class;
- Balanced Train Set: which characteristics are more or less the same as the Evaluation one;
- Unbalanced Training Set: which contains about 2 millions of samples with no restrictions on the number of samples per class.

Each of these sets is provided as a CSV file with the same structure. Audios metadata are then provided as a tabular dataset composed of 4 columns: *YTID*,

[2]The dataset is made available by Google Inc. under a Creative Commons Attribution 4.0 International (CC BY 4.0) license (https://creativecommons.org/licenses/by/4.0/). The ontology is available under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license (https://creativecommons.org/licenses/by-sa/4.0/)

which is the YouTube ID of the video from which the audio has been extracted; *start_seconds* and *end_seconds*, respectively the instant, expressed in seconds, in which the audio clip starts and ends; *positive_labels*, which indicates all the classes associated to the audio. Moreover, since classes in such sets are coded, another CSV file is provided containing information to decode them.

Starting from the Balanced Train Set, we extracted significant samples for our NoLC Alarm class. However, before to describe the extraction process, it is important to emphasize that AudioSet data are also provided under the form of 128-dimensional audio features which are stored as TensorFlow Records. We will not use such representation, but we will rely on the network, *VGGish*, used to extract them to build our classifier, as described in section 3.2.

NoLC Alarm Class

As mentioned before, AudioSet data are provided as a tabular dataset. Starting from these values, audio must be downloaded from YouTube. However, a few aspects need to be considered:

- The dataset contains a lot of classes, many of which are not of interest;
- Different classes are associated with each sample;
- AudioSet associates a quality measure, expressed in percentage, to each class. It indicates the quality of the labelling evaluated on a subset of segments of each class. A 100% quality asserts that each audio of the subset "also" belongs to the class under examination, otherwise it could not.

Above points have led to some marginal issues addressed both with the local execution of some python scripts (available on GitHub[3]) and manual check.

Since we don't need all 527 classes to address our problem, we selected only 19 of them which encompass *sirens* (civil defence, ambulance, fire trucks, emergency

[3]https://github.com/jim-schwoebel/download_audioset

vehicles in general), *alarms* of various kinds (cars, smoke detector, etc.), *bells* (church bells, cowbells, bicycle bells, etc.), and so on. These classes have been chosen for the following reasons:

- They can be easily recorded near level crossings since are related to the urban or peripheral environment;
- They can be easily confused with LC Alarms.

According to the latter point, it is worth noting that to the best of our knowledge, each EU country has it owns LC Alarm. This means that to build a global Alarm Classifier, working for all the countries, we should collect for each of them at least a couple of hundreds of samples for both the LC and NoLC Alarm classes. This is one of the further improvements that must be performed starting from this work. However, for this experimentation, we focused our attention on Italian and United Kingdom LCs, simply because their audios can be easily found on YouTube through quick researches; moreover, the former are quite similar to bells (of any kind), meanwhile the latter are similar to sirens or other alarms.

Besides the above digression, once the 19 classes have been selected, the AudioSet tabular dataset has been revised accordingly. Starting from this new tabular dataset, the aforementioned python script has been used to download the audio clips. The script works following 4 main steps:

i. Create a folder for each class in the dataset;
ii. Download the audio of the video related to the YTID value in WAV format;
iii. Trim the audio according to *start_seconds* and *end_seconds* values;
iv. If the audio belongs to more then one class, it is copied into all the related folders.

About 1170 duplicated audios were collected and filtered in order to obtain abut 760 single audios. Since not all of the selected classes had a quality of 100%, such 760 audios have been manually evaluated. Some of them were characterized by too much noise or no sound at all for most of the duration, meanwhile, others were not entirely compliant with the associated class. For example, the class sound persisted in the audio for a very short time (1-2 seconds out of 10) meanwhile the rest of the audio was about other not-of-interest classes (e.g. human speech). Also, since the aim was to create a fairly balanced dataset, 500 audios were selected manually (listening to them one-by-one) to build the NoLC Alarm Class set. This also comes from the fact that, as we will see later, we collected "only" 347 LC Alarm samples.

However, we want to emphasize that the audios selected downstream of the described process are not noiseless. Many of them are affected by noise related to environmental conditions (e.g. wind, water, etc.) and/or urban traffic. Finally, the main common characteristic that audios share is that more than 90% of them have a duration of 10 seconds, meanwhile, others have a less duration (9, 5, 3 seconds).

3.1.2 LC Alarm Class

If for NoLC Alarms we were able to extract information from a pre-built dataset, to the best of our knowledge, no Level Crossing Alarm dataset is available online. Therefore, we built it manually leveraging on the YouTube platform and the python scripts mentioned in the above section, also structuring the tabular datasets as AudioSet proposes.

In order to check if the proposed approach could be used to solve our task, we focused mostly on Italian[4] and United Kingdoms[5] Level Crossings, since their

[4]https://www.youtube.com/channel/UClNobGtQtZncqcoK7YwBB2Q
[5]https://www.youtube.com/channel/UCv_ecq6N75xWfCWA2yGWD7Q

videos can be easily found on two YouTube channels collected in playlists.

We watched many of these videos selecting from each of them one or more 10-seconds audio clips (where the LC Alert Bell rings). Also in this case, we did not consider only "clear" sounds, indeed we collected alarms at different intensities also considering those in which the LC Alarm is slightly perceptible compared to the urban, environmental or train passing noise. Moreover, we considered the year of video publication as constraint, selecting only those uploaded after 2019.

At the end of this process, we got 347 10-seconds LC Alarm audio clips which are subdivided as follows: 165 Italian LC Alarm audios, 165 United Kingdom LC Alarm audios, and 17 audios related to LCs of other European countries. Fig. 3.1 shows an overview of these audio clips in terms of country and year in which the videos have been uploaded.

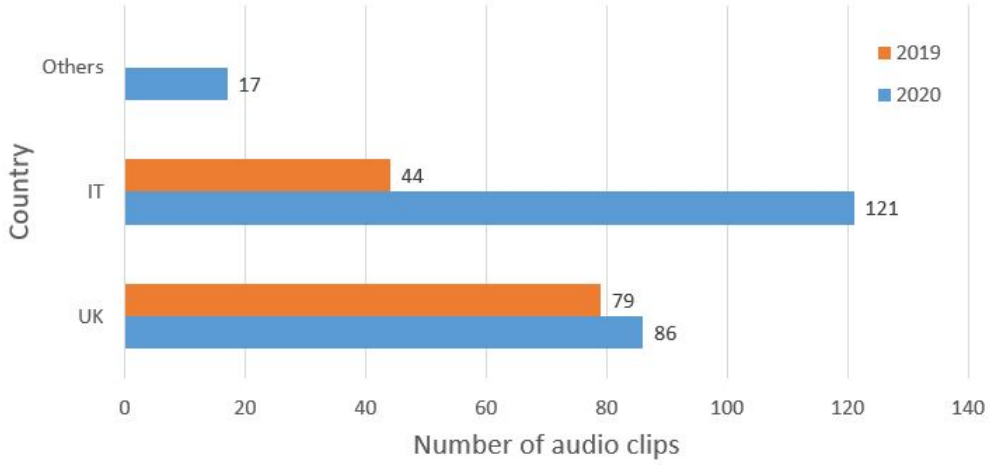


Figure 3.1: LC Alarm audio clips

In conclusion, downstream the whole dataset creation process, we obtained 847 audio clips: 500 belonging to the NoLC Alarm class, and 347 belonging to the LC Alarm one.

3.2 Model Definition

As already mentioned, AudioSet data are also provided as 128-dimensional features extracted using a VGG-based acoustic model [87], trained on a preliminary version of YouTube-8M[6]; a TensorFlow model of such network, named VGGish, is available on GitHub[7]. Actually, also a Keras implementation is available both on GitHub[8] and as Python library[9]. Since that, we decided to re-use the VGGish network as a base network for our classifier; in particular, we leveraged on the Keras implementation (in Python). We have chosen this network also because the authors shared the pre-trained weights of the network.

To perform all the following steps (data pre-processing, network training, etc.), we used Google Colaboratory, a hosted Jupyter notebook which allows running python scripts providing free access to computing resources like GPUs. This allowed us to perform all the experimentation with no added costs and in a time-efficient manner given the computational power of provided resources.

3.2.1 VGGish-based network

VGGish is a Deep Neural Network composed of 15 layers between convolutional, maxpooling and fully connected. An overview, with all the parameters, is shown in Fig. 3.2, meanwhile, a block representation is presented in Fig. 3.3.

Clearly, the input layer (first layer) has no parameters as it has the only task to present to the rest of the network the features related to the input samples. Conversely, the last fully-connected layers are necessary in order to produce the tensor that will be post-processed in order to obtain the features that the authors have made available as features representative of the entire dataset; these layers will then be replaced to build our classifier.

[6]https://research.google.com/youtube8m/index.html
[7]https://github.com/tensorflow/models/tree/master/research/audioset/vggish
[8]https://github.com/beasteers/VGGish
[9]https://pypi.org/project/vggish-keras/#description

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 96, 64, 1)]       0
_________________________________________________________________
conv1 (Conv2D)               (None, 96, 64, 64)        640
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 48, 32, 64)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 48, 32, 128)       73856
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 24, 16, 128)       0
_________________________________________________________________
conv3/conv3_1 (Conv2D)       (None, 24, 16, 256)       295168
_________________________________________________________________
conv3/conv3_2 (Conv2D)       (None, 24, 16, 256)       590080
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 12, 8, 256)        0
_________________________________________________________________
conv4/conv4_1 (Conv2D)       (None, 12, 8, 512)        1180160
_________________________________________________________________
conv4/conv4_2 (Conv2D)       (None, 12, 8, 512)        2359808
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 6, 4, 512)         0
_________________________________________________________________
flatten_ (Flatten)           (None, 12288)             0
_________________________________________________________________
fc1/fc1_1 (Dense)            (None, 4096)              50335744
_________________________________________________________________
fc1/fc1_2 (Dense)            (None, 4096)              16781312
_________________________________________________________________
fc2 (Dense)                  (None, 128)               524416
=================================================================
Total params: 72,141,184
Trainable params: 72,141,184
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.2: VGGish Network (Layers-Parameters)

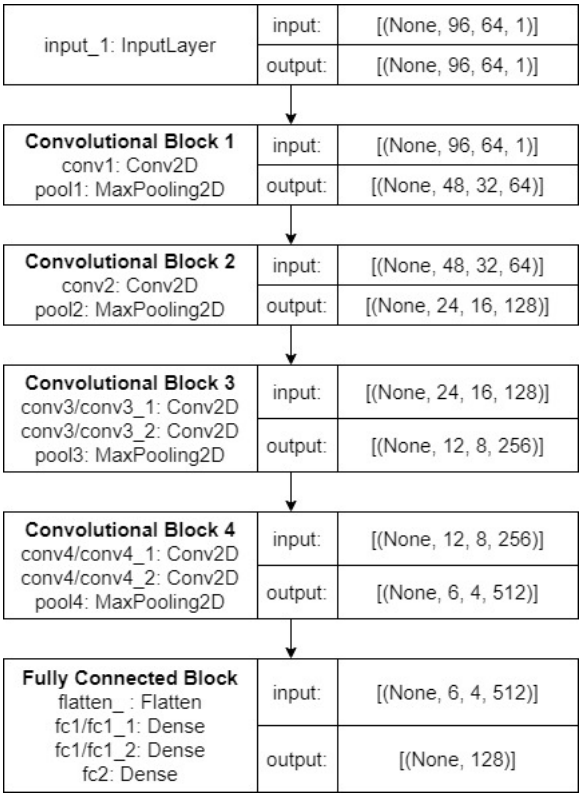


Figure 3.3: VGGish Bock Representation

However, before proceeding, it is necessary to analyze some characteristics of the network. First, evaluating the block representation (Fig. 3.3), it is notable that the network requires [96, 94, 1] shaped tensors as input. Theoretically, it can be seen as a sort of image with 96x64 dimensions and depth equals to one. As we will show later, such tensors are nothing then more Mel Spectrogram's chunks; briefly, the audio will be converted into its Mel Spectrogram from which different chunks (or examples)will be extracted in order to feed and evaluate the network.

Other aspects to take into account are:

- Each Conv2D layer agrees with the following characteristics: a 3x3 Kernel size (or filter size), a stride equals to one, a zero-padding (indicated as *same* in Keras) which allows each layer to produce an output feature with the same shape (height x width) of the input one, and a *relu* activation functions.

- Each MaxPooling2D layer have a Pool size and a stride both with 2x2 di-

mensions.

The pre-computed weights, available both in *ckpt* format (mostly used in Tensorflow implementations) and in *h5* format (for Keras), are related to the whole network just presented, also covering the Fully Connected layers. However, in order to generate a new classifier, the FC block in Fig. 3.3 must be replaced with others.

Typically, before placing fully connected layers on the top of the network, the output features of the last convolutional block are transformed and brought from multiple dimensions to a single one. To be specific, as shown in Fig. 3.2, the output tensor at the last convolutional block has four dimensions [None, 6, 4, 512]. Actually, the first dimension (None) is related to the batch size under examination, i.e. to the number of samples the batch is composed of. Therefore, the features' dimensions are dictated by the last three values that are respectively the height (H), the width (W), and the channels (C, also known as depth). After the last convolutional block, a Flatten Layer takes in input a tensor of dimensions (batch_size, H, W, C) and returns a tensor of dimensions (batch_size, H*W*C), which can then be placed in input to the last FC layers. When the fully connected block is dropped to create a new classifier, the Flatten layer is replaced by a Global Pooling Layer that performs, more or less, the same transformation. Practically, considering the python code of the Keras implementation of the VGGish network, by setting some parameters it is possible to automatically drop the fully connected block and choose between Globalaveragepooling2D (by default) or Globalmaxpooling2D layer. Such layers simply perform pooling considering as pool size the same size of the tensors' slices. Considering the example above, if the output tensor of the convolutional layer has dimensions [None, 6, 4, 512], the global pooling layer applies a max or avg pooling considering a pool of 6x4 size. Therefore, as output, it will produce a [None, 1, 1, 512] shaped tensor, then [None, 512].

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 96, 64, 1)]       0
_________________________________________________________________
conv1 (Conv2D)               (None, 96, 64, 64)        640
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 48, 32, 64)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 48, 32, 128)       73856
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 24, 16, 128)       0
_________________________________________________________________
conv3/conv3_1 (Conv2D)       (None, 24, 16, 256)       295168
_________________________________________________________________
conv3/conv3_2 (Conv2D)       (None, 24, 16, 256)       590080
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 12, 8, 256)        0
_________________________________________________________________
conv4/conv4_1 (Conv2D)       (None, 12, 8, 512)        1180160
_________________________________________________________________
conv4/conv4_2 (Conv2D)       (None, 12, 8, 512)        2359808
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 6, 4, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
=================================================================
Total params: 4,499,712
Trainable params: 0
Non-trainable params: 4,499,712
```

Figure 3.4: VGGish Network Convolutional Layers only

To build our classifier, we considered the default option. Then, we dropped the FC block obtaining the network in Fig. 3.4 on the top of which we added two more layers:

- A first Dense Layer (fc_layer) with 128 kernels ad a relu activation function;
- A last Dense Layer (prediction) that has only two neurons, one per class, and a softmax activation function. The motivation of this choice are reported in section 3.4.1.

The characteristics of the classifier are shown in Fig. 3.5 and 3.6. Considering the first one, since this screenshot has been made on the pre-trained network freezing the parameters of convolutional blocks, it is possible to notice that the network has about 4.5 millions of *Non-trainable params*. Indeed, the *Trainable params* (about 66 thousand) are those related to the last two layers. Lastly,

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model (Functional)           (None, 512)               4499712
_________________________________________________________________
fc_layer (Dense)             (None, 128)               65664
_________________________________________________________________
prediction (Dense)           (None, 2)                 258
=================================================================
Total params: 4,565,634
Trainable params: 65,922
Non-trainable params: 4,499,712
_________________________________________________________________
```

Figure 3.5: LC Alarm Classifier (Layers-Parameters)

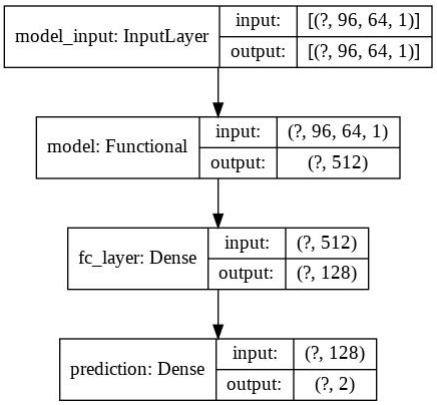


Figure 3.6: LC Alarm Classifier Block Representation

The "model: Functional" is the VGGish base network presented in Fig. 3.4

for a binary classifier, a single neuron in the output layer would suffice; it will ideally predict 0 if the particular input belongs to the first class, vice versa, it will predict 1. However, in our case, we preferred to use one-hot coding to make the classifier more robust: [1,0] will represent the LC Alarm class, meanwhile, [0,1] will represent the NoLC Alarm one. This involves two neurons in the output layer, each of which will produce the value of a single bit.

3.3 Data Pre-processing

As already mentioned, the VGGish admits in input [96, 64, 1] shaped tensors. To obtain such representation, audio pre-processing is required.

As a first step, leveraging on the functionalities offered by python libraries such as TensorFlow, soundfile and NumPy, the numerical features were extracted from

```
['LC_23_start_0_stop_10.wav'
 array([[    0,     0],
        [    0,     0],
        [    0,     0],
        ...,
        [-4132, -2202],
        [-4071, -2172],
        [-4678, -3058]], dtype=int16)
 44100]
```

Figure 3.7: Tabular Data Sample

each of the WAV audio. Therefore, a tabular dataset has been created in which each entry represent a particular audio through three features (Fig 3.7):

- *Audio name*;
- *WAV Features*: a Numpy Array with [frames x channels] shape, which represents the amplitude of the wave signal frame by frame. The number of frames is given by the sample rate (often equals to 44.1 kHz in our case) multiplied by the duration of the audio expressed in seconds (generally 10s), meanwhile, the number of channels comes from the audio format: if *stereo*, as in most cases, then it is equal to 2, otherwise, if *mono*, it is equal to 1. Generally, almost all arrays will have shape [441000, 2];
- *Sample Rate*: which varies depending on the audio but is mostly 44.1 kHz.

Both the WAV Features array and the Sample Rate have been obtained through the soundfile.read() function. The features array is then *normalized* so that all values are included in [-1;1] range, and a *stereo-to-mono conversion* is performed where required. The normalization is often a recommended step when dealing with neural networks, since, as already described in section 1.1.1, the weights updating process is quite sensible to feature with different scales. On the other hand, the stereo-to-mono conversion is performed simply averaging the values to obtain, for each audio, a single spectrogram (otherwise we should have two). However, as also shown in Fig. 3.8, there is not much difference between the waves of the two channels.

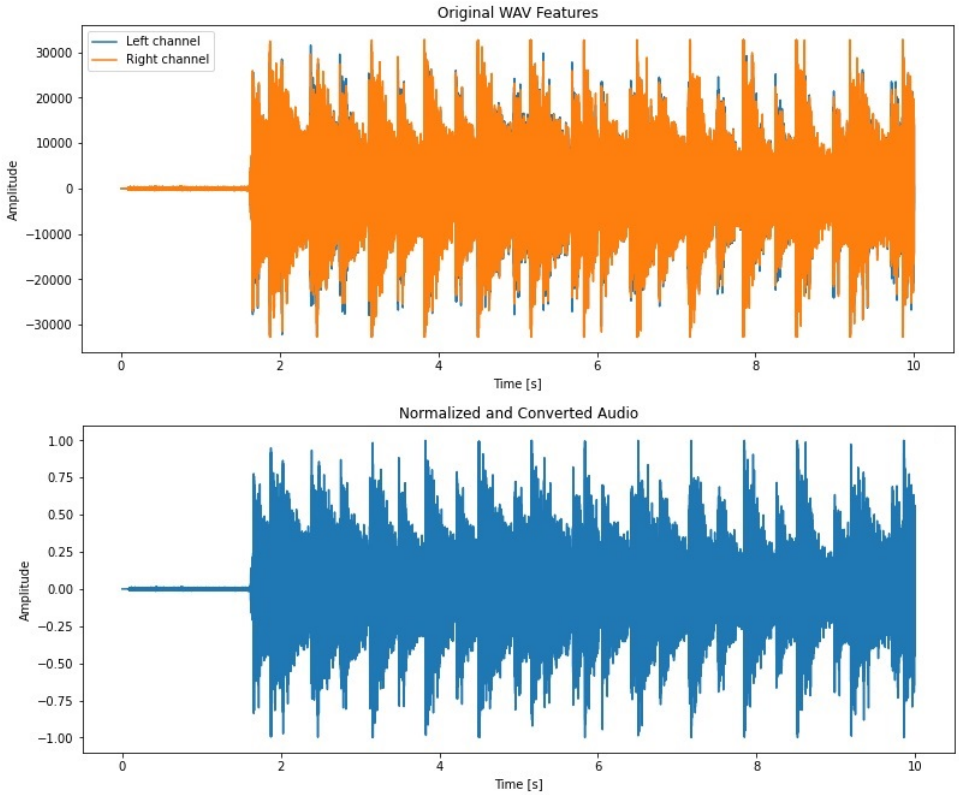


Figure 3.8: Audio Processing

The top chart shows the waves of the two channels overlapped. As it is easily notable, such waves are quite similar. Otherwise, the bottom chart shows the wave after the normalization and conversion steps; it remains, in form, similar to the two above.

Once these transformations are performed, it is necessary to compute the Log Mel Spectrogram. The choice of such representation is purely related to the network and the pre-computed weights that we used. Since the network has been obtained by analysing this kind of spectrograms, it is necessary to adopt the same representation in order to be able to use the network properly, also considering the same parameters. Moreover, such spectrograms have been derived from audio with a sample rate of 16000 Hz; therefore, also the resampling has been performed where the audio's sample rate was different from 16000 Hz (in almost all cases).

3.3.1 Mel Spectrogram

The spectrogram gives information about both the time and frequency domain. While the Fast Fourier Transform (FFT) allows evaluating which kinds of frequencies characterize an audio, the spectrogram allows us to also understand when in time such frequencies appear.

When it comes to non-periodic signals, the spectrogram can be obtained simply applying the Short-Term Fourier Transform (STFT) to the waveform. The STFT is nothing more than an iterative application of the FFT to different chunks (sets of frames) of the waveform. To obtain a spectrogram, three principal parameters have to be set: the window length, the window hop and the windowing function. The window length defines the width, in terms of samples, of the different chunks. Starting from the first frame, the FFT is computed considering the wave in the chunk and the windowing function (multiplying them). Then, the next chunk is evaluated moving the window forward by a value equals to the window hop. Typically, the hop is less than the length in order to obtain overlapped chunks. How these FFTs are merged to obtain the spectrogram is beyond this thesis, what we want to emphasize is the process of feature extraction in order to feed the classification network.

To leverage on the VGGish network and its weights, data processing has to match the one performed to compute features used to train this network. Then, we considered the same parameters, i.e. a window length equals to 25ms, a window hop of 10ms, and the periodic Hann window as the windowing function.

Once the spectrogram has been obtained, the Mel (melody) spectrogram has been derived "simply" mapping the frequencies considering the Mel Scale [174]. This is nothing more than a different representation of frequencies related to the human perception of frequency variation. Typically, humans do not perceive equally the same frequency variation at high and low frequencies. The *Mel Scale*

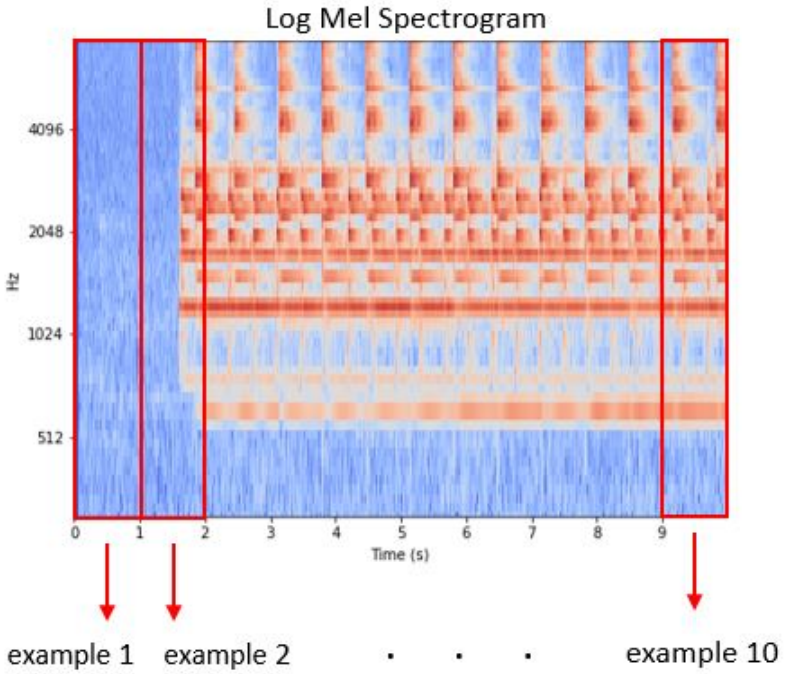


Figure 3.9: Log Mel Spectrogram and Examples

have been defined by performing an empirical experiment obtaining that the same variation on a different point on such a scale is perceived in the same manner by humans. Practically, the conversion is done through a proprietary function which maps the 257 frequency bins into 64 mel bands covering frequencies between 125 and 7500 Hz. Once that, the logarithmic computation of the Mel Spectrogram features is performed and [998, 64] shaped features have been obtained.

Finally, as the last step, these features are divided into K non overlapped examples with shape [96, 64, 1], where K is equal to the number of frames divided by 96. Since this could not be an integer number, the last values of the Log Mel Spectrogram will be dropped to obtain examples with equal size. Theoretically, each example will contain 96 10ms frames, then each of them represents roughly 1 second of audio (Fig. 3.9).

Table 3.1: Training, Validation and Test sets

Set / ***Class***	*Training*	*Validation*	*Test*	*Original Dataset*
LC Alarm	247 (40.6%)	50 (41.7%)	50 (41.7%)	347 (40.9%)
NoLC Alarm	360 (59.4%)	70 (58.3%)	70 (58.3%)	500 (59.1%)
Total	607 (71.7%)	120 (14.15%)	120 (14.15%)	847

3.4 Network evaluation

To evaluate the network, the original dataset has been subdivided into three subsets: Training, Validation and Test set. It is important to note that this subdivision has been made considering the tabular dataset obtained through the first pre-processing step as described in section 3.3 above. Hence, it is performed at the audio level and not at frame one. Moreover, we tried to maintain constant the ratio between the two classes meanwhile splitting the dataset. Table 3.1 reports the contents of the three sets: the original dataset is composed of 347 LC Alarm audios (roughly the 40.9% of the total), and 500 NoLC Alarm audios (59.1%); the Validation and Test sets have the same number of audios, 120 in total, 70 of which (58,3%) belonging to the NoLC Alarm class and 50 belonging to the other one (41,7%); lastly, the Training set is composed of 247 LC Alarm audios (40.6%) and 360 NoLC Alarm audios (59.4%). We will refer to these sets, at the audio level, as *"Rough" sets*.

Once *Rough sets* have been created, the remaining pre-processing steps have been performed to convert them into *"Flat" sets*: these are composed of all the examples of 960ms related to the audio belonging to the corresponding Rough set. Each of these frames has been automatically labelled considering the class associated with the reference audio adopting the one-hot coding: the code [1;0] have been associated to the LC Alarm class, meanwhile, the NoLC Alarm class have been represented through the [0;1] coding. Table 3.2 details the composition of the *Flat sets*: the Flat Training set is composed of 6003 examples, 2470 of which

Table 3.2: Flat Training, Validation, and Test sets

Set / ***Class***	*Flat Training*	*Flat Validation*	*Flat Test*	*Original Flat Dataset*
LC Alarm	2470	500	500	3470
NoLC Alarm	3533	677	678	4888
Total	6003	1177	1178	8358

belong to the LC Alarm class (actually, the number of the audios times 10 frames) and 3533 NoLC Alarm examples (not all the NoLC Alarm audios have a duration of 10 seconds); the Flat Validation Set is composed of 500 e 677 samples, which respectively belong to the LC Alarm and NoLC Alarm class; lastly, the Flat Test set is composed of 500 LC Alarm examples and 678 NoLC Alarm ones.

3.4.1 Network Training & Validation Results

Based on what has been described so far, the network will be able to predict second-by-second the content of a particular audio clip. This is extremely important as the network will be probably able to analyze a video, with further improvements, evaluating the sound behaviour in real-time second after second.

When training a network, some of the parameters to consider and evaluate depend on choices made in terms of network optimizer, loss function and evaluation metric:

- Optimizer: we considered the Adam [175] optimizer since it has been used for the VGGish training. We also used the same hyperparameters: *learning rate* equals to 10^{-4} and *epsilon* equals to 10^{-8}.

- Loss Function: we used the Categorical Cross-Entropy (CCE) loss function. Typically, the CCE loss is used for multi-task classification problems: having a one-hot coding of the classes, CCE is computed thorough the negative summation, over all the classes, of the products between the bit of the one-hot coded class (0, 1) and the logarithm of the outputs of the network.

- Evaluation Metric: we evaluated the Categorical Accuracy. Even if, as the CCE loss function, also the Categorical Accuracy is used for multi-class classification problems, the one-hot coding allows us to rely on it since accuracy is computed evaluating the position of the max value in the prediction array in relation to the max value in the one-hot coding representation of the classes. Nevertheless, in our case, this metric coincides with Binary Accuracy.

Beyond these, further considerations we made during our tests are related to the pre-computed weights. Indeed, we tested the network in three different ways:

1. Loading and freezing the weights for the convolutional layers;
2. Loading the weights without freezing them;
3. Training the network from scratch (without pre-loaded weights).

Such tests have been performed considering different batch sizes, monitoring the loss and the accuracy to the advance of the epochs.

Network Training with frozen weights

In this section, we will discuss all the results we obtained training the network considering the pre-computed weights for the convolutional layers and freezing them. *Freeze* means that during the training process the weights will not undergo any variation. The different parameters configurations we considered for this analysis are shown in Table 3.3; they differ only for the *Batch Size*. Lastly, it is important to note that all the analyses have been conducted considering *Flat* sets and not Rough ones (section 3.4).

Table 3.3: Parameters configurations with frozen weights

Configuration	**Weights**	**Adam Optimizer**		**Loss Function**	**Evaluation Metric**	**Epochs**	**Batch Size**
		Learning Rate	Epsilon				
Config 1	Frozen	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	300	30
Config 2	Frozen	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	300	60
Config 3	Frozen	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	300	100

Evaluating the first configuration (*Config 1*), we obtained the results shown in Fig. 3.10.

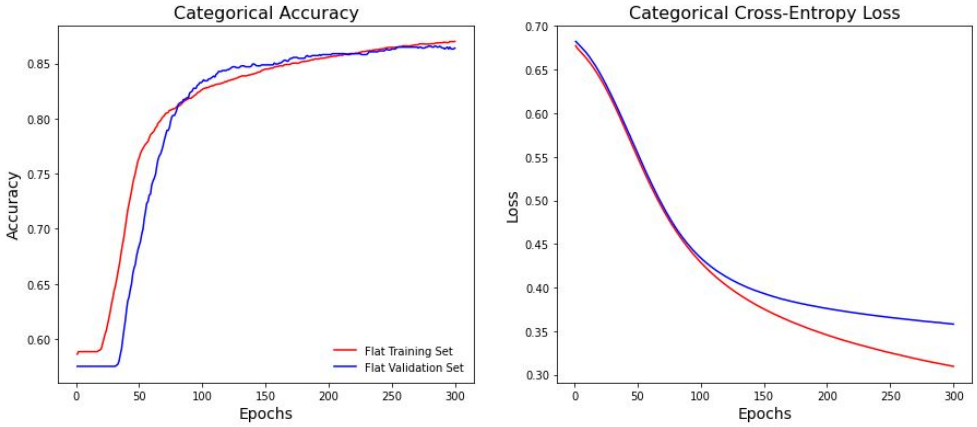


Figure 3.10: Performances of the first configuration (Config 1)

Although Flat Validation accuracy after approximately 100 epochs starts to increase very slowly, almost as if it had reached a sort of plateau near 0.85, looking at the losses, we can assert that the network can be further trained. Theoretically, after a certain number of epochs, the validation loss could assume an ascending trend; however, this is not always true in practice. Same trends have been encountered for the other two parameters configurations (Fig. 3.11 and Fig. 3.12).

To understand which configuration produces the best results, we must evaluate them in some comparative charts as shown in Fig. 3.13. The top two charts show the Flat Training Accuracy (left) and the Flat Validation Accuracy (right) of all the configurations; on the other hand, the two bottom charts show the losses.

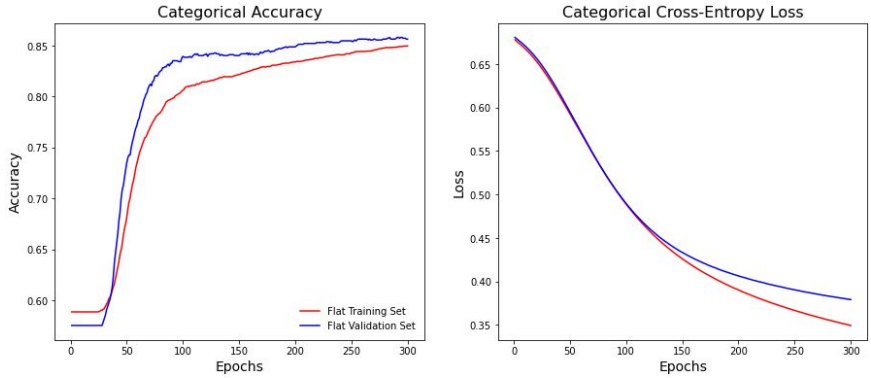


Figure 3.11: Performances of the second configuration (Config 2)

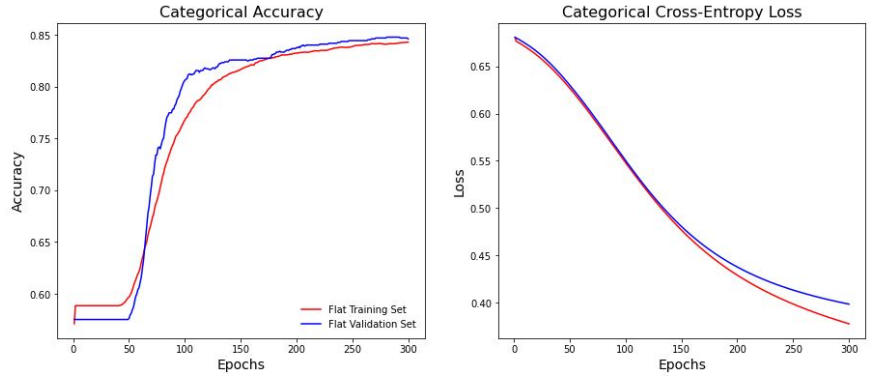


Figure 3.12: Performances of the third configuration (Config 3)

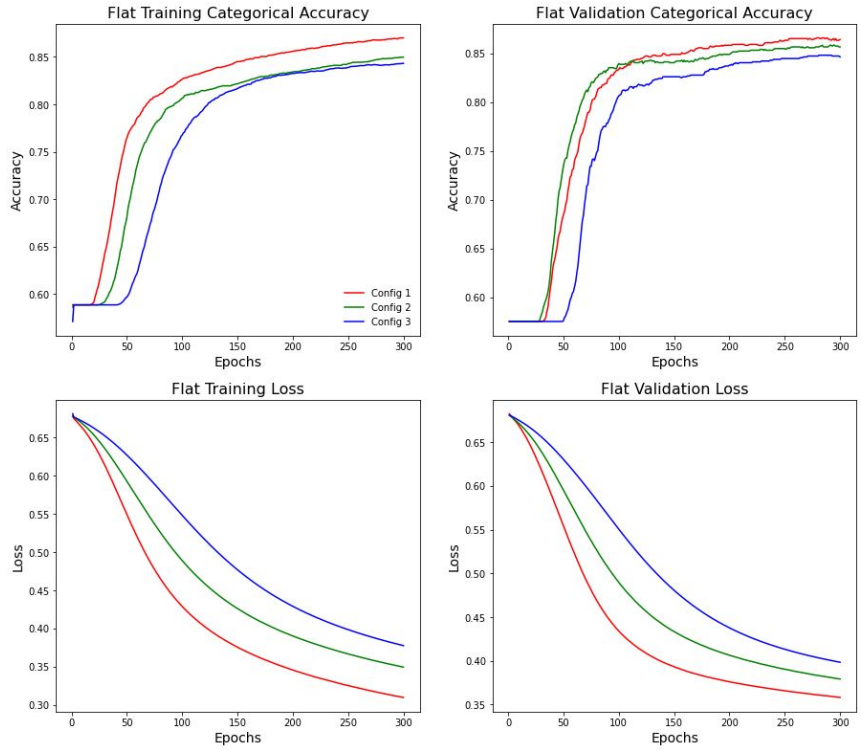


Figure 3.13: Comparing the Frozen weights training configurations

As easily understandable, all of these charts show that the first configuration has the best performances. Such results could be related to the fact that in the first case we have a smaller, but probably sufficiently large, batch size that allows the network to extract knowledge without losing too many information and then obtain an optimal last layers' weights updating.

Once performance at the frame level has been evaluated, to obtain those related to the audio level classification, some post-processing tasks are required. For each epoch, frames have been presented to the network differently. In particular, for each entry in the Rough Validation set we created a different batch containing the frames belonging to the entry under examination. Then, we got as many batches as the number of audios. Such batches have then been presented to the network one by one obtaining, for each of them, an accuracy value. If the accuracy of the batch is greater then 0.5, the batch, and then the audio, has been correctly classified. Basing on this principle, the accuracy related to the Audio Classification task have been computed looking to the batches accuracy epoch-by-epoch. The trend of this accuracy, concerning the first configuration, is shown in the left chart in Fig. 3.14 as "Validation Set (Audio level)". As can be seen, such accuracy is higher than the others since some misclassification errors on the single frames are absorbed by the batch. In other words, a batch which has been correctly classified will probably have some misclassification error at the frame level, but such errors will not be perceived externally.

The chart on the right in Fig. 3.14 compares the Audio level accuracy of the different configurations and, even in this case, the first configuration achieves the best performances.

Considering the analysis conducted so far, we can assert that around the 300 epochs, but also in perspective, Configuration 1 has the best overall performances. The accuracy and loss values of the last 5 epochs are reported in Table 3.4.

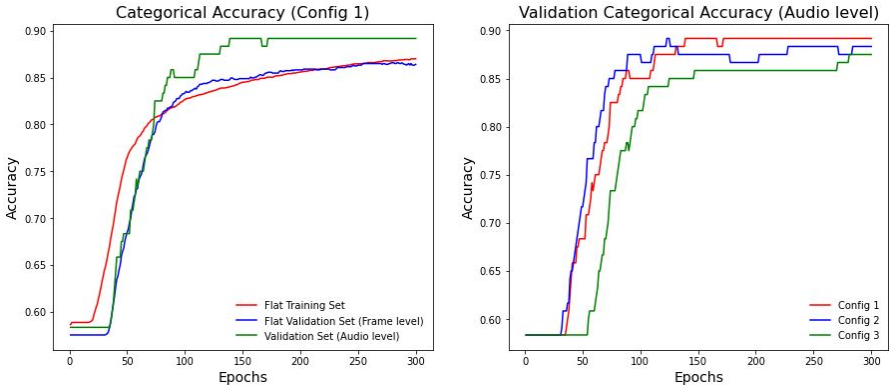


Figure 3.14: Validation Accuracy at the Audio Level

Table 3.4: Configuration 1 - Last 5 epochs values

Epoch	**Categorical Accuracy**			**Categorical Cross-Entropy Loss**	
	Flat Training	*Flat Validation (Frame Level)*	*Validation (Audio level)*	*Flat Training*	*Flat Validation (Frame Level)*
296	0.86989838	0.86491078	0.89166665	0.3108044	0.35868487
297	0.86989838	0.86321157	0.89166665	0.3105799	0.35860419
298	0.87006497	0.86321157	0.89166665	0.31029809	0.35849842
299	0.87006497	0.86321157	0.89166665	0.31003213	0.35830021
300	0.87006497	0.86406118	0.89166665	0.30975685	0.35815579

Network Training without freezing the weights

In this section, we will analyze the results about the configurations shown in Table 3.5. What differs from Table 3.3 are: the number of epochs, and the weights consideration. Indeed, weights are loaded but not frozen (they will update during the training process). We limited the training to 100 epochs since, as we will show below, the Flat Validation Loss starts growing after roughly 60.

The time required to train and evaluate the network, at the frame level, increases in respect of the first configurations. Indeed, they required roughly 3 or

Table 3.5: Parameters configurations with not frozen weights

Configuration	**Weights**	**Adam Optimizer**		**Loss Function**	**Evaluation Metric**	**Epochs**	**Batch Size**
		Learning Rate	Epsilon				
Config 4	Not Frozen	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	100	30
Config 5	Not Frozen	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	100	60
Config 6	Not Frozen	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	100	100

5 (then 4 in average) seconds to perform the computations needed for a single epoch, now the required time ranges from 9 to 11 seconds; the whole time roughly doubles. This is clearly due to the number of weights that must be updated during the training process. In the first three cases, "only" 66 thousand weights had to be updated; on the contrary, in this case, weights are roughly 4.5 million. Making an estimation, on 100 epochs, if weights are frozen, the whole training+validation step requires about 8 and a half minutes, otherwise, without freezing the weights, it requires about 17 minutes.

According to results, Fig. 3.15 shows a network with unstable behaviour. If, on one hand, the accuracy has a quite theoretical trend, with the Flat Training one upon the others settling around 1 after a certain number of epochs; on the other hand we can note several oscillations, especially in the sixth configuration. Such results seem to validate the theoretical thesis that the used dataset is not qualitative optimal or quantitative large enough to obtain a proper characterization of the whole network. Indeed, as also described in section 1.2.1, when implementing a Transfer Learning approach, if the original dataset (millions of videos) is too large in respect of the target dataset (few hundreds of 10s audios), fine-tune the whole network could result in oscillations.

However, it can be noted how after few epochs both the validation accuracies reach values very close to 0.9, meanwhile, the Training accuracy exceeds it. Moreover, in Table 3.6 are shown all the accuracy and loss values related to a theoretically optimal choice of the number of epochs given the lowest Flat Validation Loss value (Early Stopping).

In conclusion, we can assert that without freezing the weights, the network parameterization is quite complex and will certainly be the subject of future studies. Actually, in this sense, another test has already been done simply lowering the learning rate of the optimizer from 10^{-4} to 10^{-6} for the fifth configuration. Nevertheless, the network still behave in an unstable manner (Fig. 3.16).

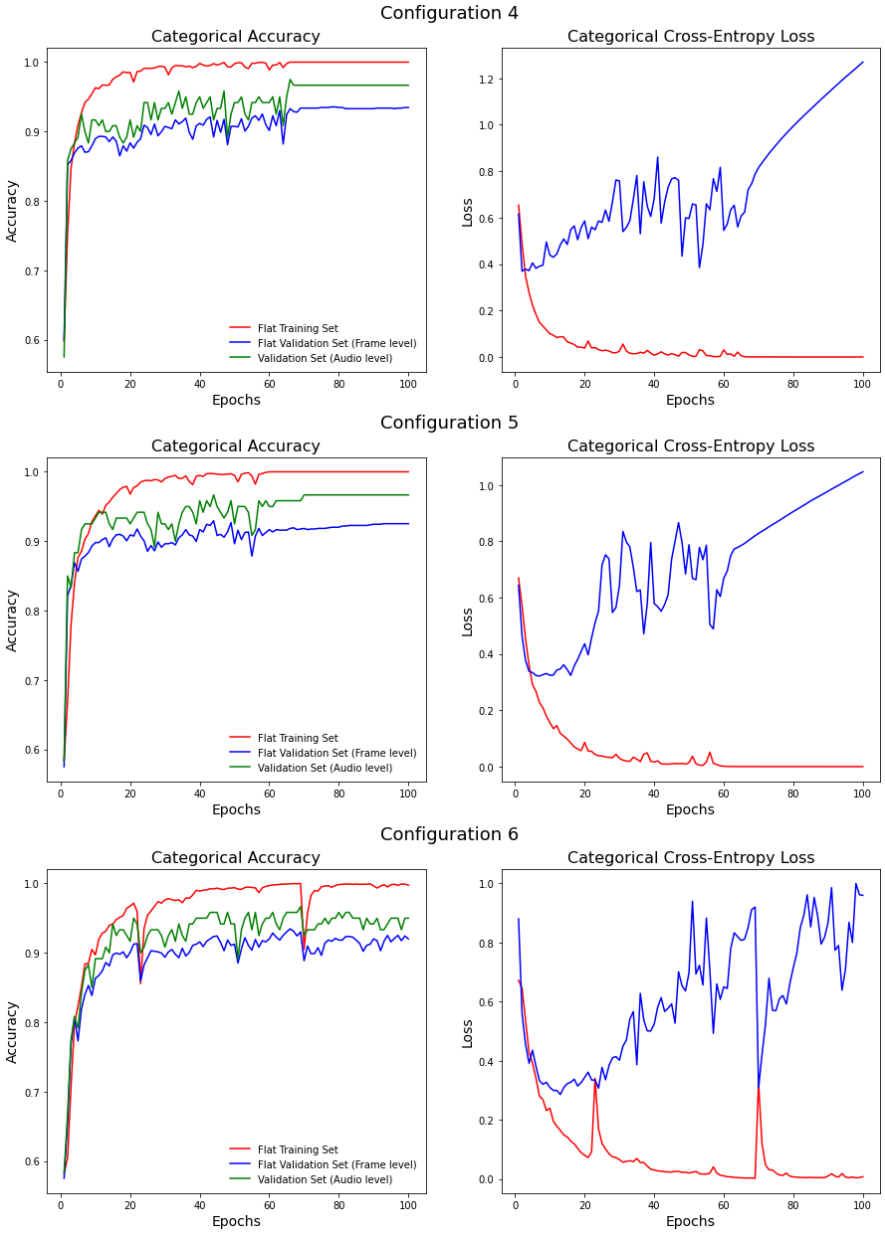


Figure 3.15: Network training results with not-frozen weights

Table 3.6: Early Stopping results for not-frozen weights configurations

	Epoch	**Categorical Accuracy**			**Categorical Cross-Entropy Loss**	
		Flat Training	*Flat Validation (Frame level)*	*Validation (Audio level)*	*Flat Training*	***Lowest Flat Validation (Frame level)***
Config 4	2	0.75145763	0.85216653	0.85833335	0.48700946	0.36946564
Config 5	7	0.90288192	0.87850469	0.925	0.22769725	0.32147234
Config 6	13	0.93136763	0.88615125	0.90833336	0.16468197	0.28566950

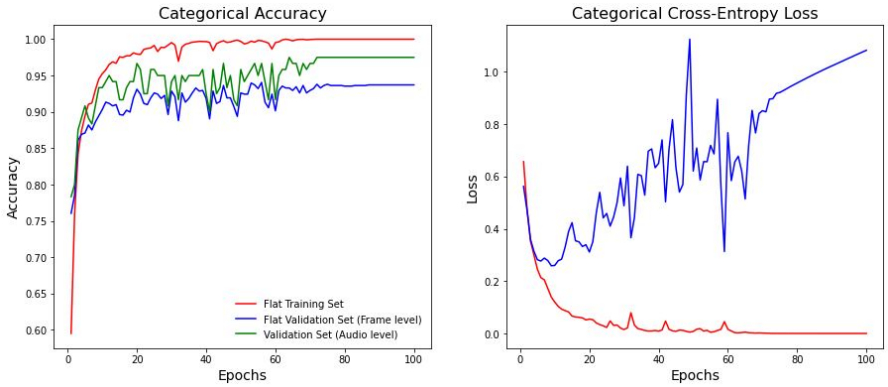


Figure 3.16: Configuration 5 with lower learning rate

Downstream these analyses, although the network reaches better performances in less time in respect to the first three configurations, it results too unstable to guarantee adequate generalization and persistence.

Network Training without pre-computed weights

To perform these last tests, we studied the network without loading the pre-computed weights. As for the above analyses, even in this case, three configurations have been tested (Table 3.7); the results are shown in Fig. 3.17.

The same considerations as in the previous section apply to this analysis, as well as the conclusions. The instability of the network does not allow us to rely on it despite the qualitatively good results achieved during Validation by selecting an appropriate number of epochs (Table 3.8). Actually, the lowest Flat Validation loss for the seventh configuration was registered at 28 epochs; however, the network results to be already too unstable, then, 17 epochs should be a more correct choice.

Table 3.7: Parameters configurations without pre-computed weights

Configuration	**Weights**	**Adam Optimizer**		**Loss Function**	**Evaluation Metric**	**Epochs**	**Batch Size**
		Learning Rate	Epsilon				
Config 7	NO	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	100	30
Config 8	NO	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	100	60
Config 9	NO	10^{-4}	10^{-8}	Categorical Cross-Entropy	Categorical Accuracy	100	100

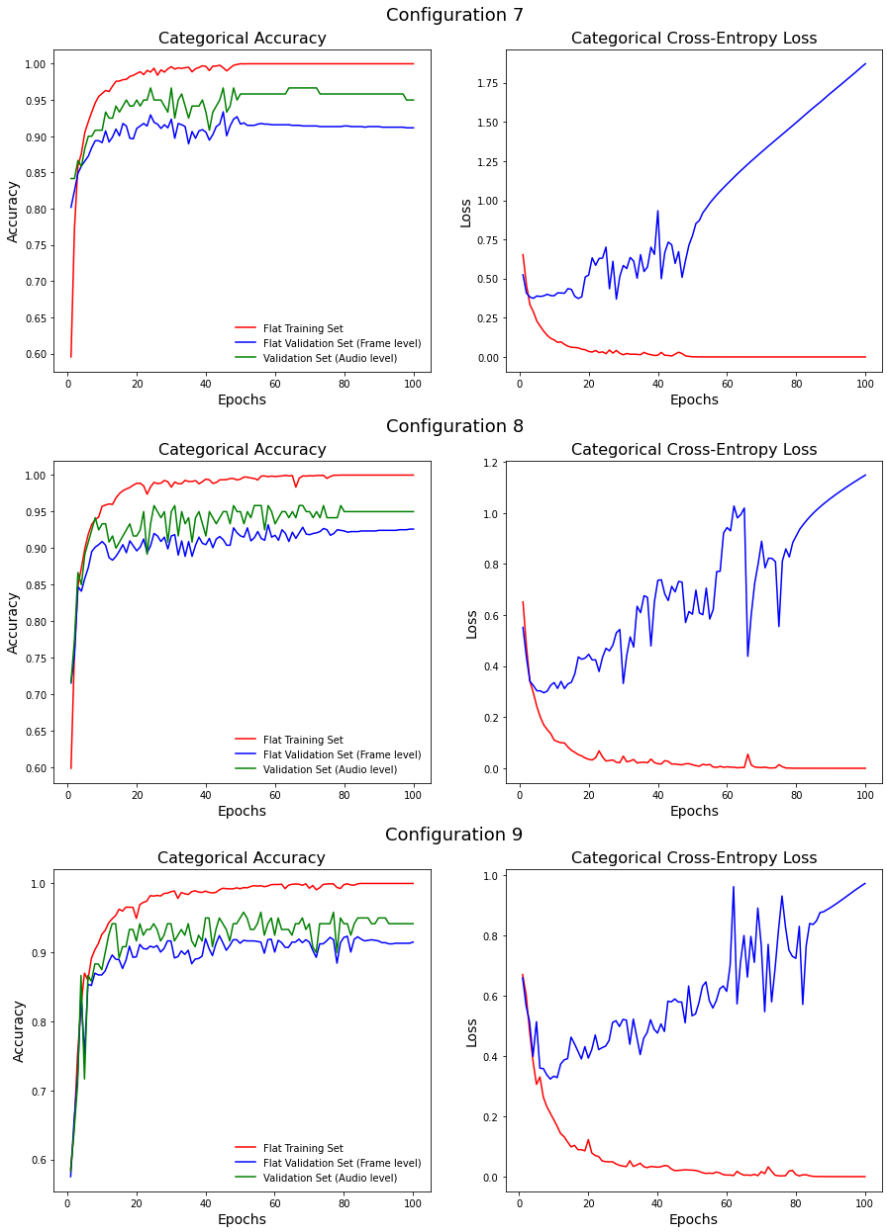


Figure 3.17: Results for network training without pre-computed weights

Table 3.8: Early Stopping results for not-frozen weights configurations

	Epoch	**Categorical Accuracy**			**Categorical Cross-Entropy Loss**	
		Flat Training	*Flat Validation (Frame level)*	*Validation (Audio level)*	*Flat Training*	***Lowest Flat Validation (Frame level)***
Config 7	17	0.97851073	0.91418862	0.95	0.05778701	0.37324047
Config 8	7	0.93253374	0.89549702	0.925	0.17015323	0.29571691
Config 9	9	0.91287690	0.86745965	0.8833333	0.21072910	0.32414805

3.4.2 Network Testing - Results

Taking into consideration all of the analyses described in section 3.4.1, we can assert that the first configuration presents the better trade-off between performances and network stability. However, downstream the 300 training epochs, we obtained a network which was further trainable since the Flat Validation loss still showed a decreasing trend (Table 3.4).

Therefore, we trained the network for 200 more epochs. Accuracy and loss trends are shown in Fig. 3.18, meanwhile, their values for the last 5 epochs are summarized in Table 3.9.

The table shows that even with 200 more epochs, results still remain stable with a very small increase in accuracy. Indeed, we got an increment in Flat Training accuracy and Validation accuracy at the audio level of roughly 0.01; meanwhile, we obtained only a 0.005 increment on the Flat Validation accuracy. Therefore,

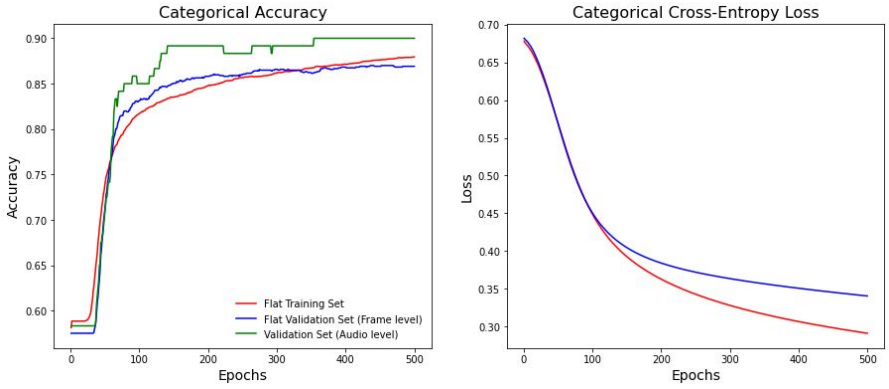


Figure 3.18: Config 1 (500 epochs) results

Table 3.9: Configuration 1 (500 epochs) - Last 5 values

Epoch	**Categorical Accuracy**			**Categorical Cross-Entropy Loss**	
	Flat Training	*Flat Validation (Frame Level)*	*Validation (Audio level)*	*Flat Training*	*Flat Validation (Frame Level)*
496	0.87906044	0.86915886	0.9	0.29149526	0.34074166
497	0.87906044	0.86915886	0.9	0.29135829	0.34064647
498	0.87922704	0.86915886	0.9	0.29122182	0.34055152
499	0.87956023	0.86915886	0.9	0.29108557	0.34045654
500	0.87956023	0.86915886	0.9	0.29094973	0.34036171

since it is not so worthy to roughly double the training time to obtain such small gains, we tested the network on the Flat Test set considering the fist configuration with 300 epochs.

We combined Flat Training and Flat Validation sets in a single Flat Joint Training set that we used to train the network, for 300 epochs, evaluating its performances on the Flat Test set. Fig. 3.19 shows accuracy and loss trends. However, in this case, we have that the Flat Test accuracy, after very few epochs, is always greater than the Flat Join Test Set; at the same time, the loss is always lower.

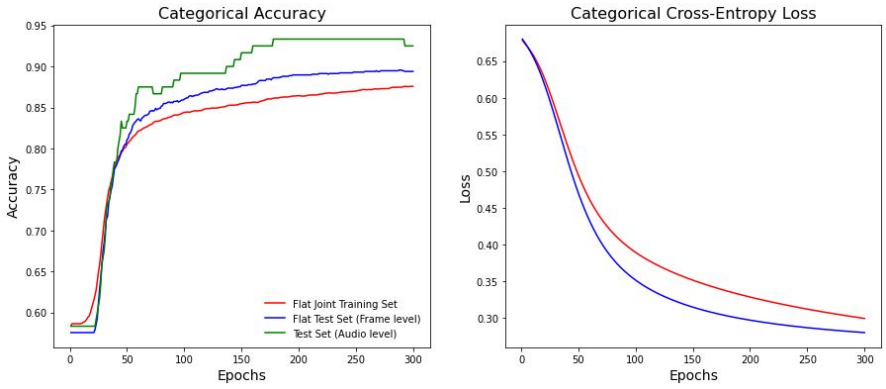


Figure 3.19: Config 1 - Test Results

Moreover, considering the charts in Fig. 3.20, the model seems to have optimal generalization performances. Evaluating the chart on the left, the accuracy on the test set (blue line) has a trend quite similar to the accuracy on the validation set (yellow line) but is "constantly" higher. At the same time, evaluating the losses (the chart in the middle), we can notice how the loss computed on the test set (blue line) is always lower than the loss on the validation set (yellow line); moreover, they have a descendent trend. Then, the network seems to be able to correctly rely on new samples since adding the Validation Set to the Training one in Testing phase, i.e. adding new knowledge, the accuracy increase, while the error the network performs in the classification phase decreases. These results seem to indicate a correct network characterization and generalization.

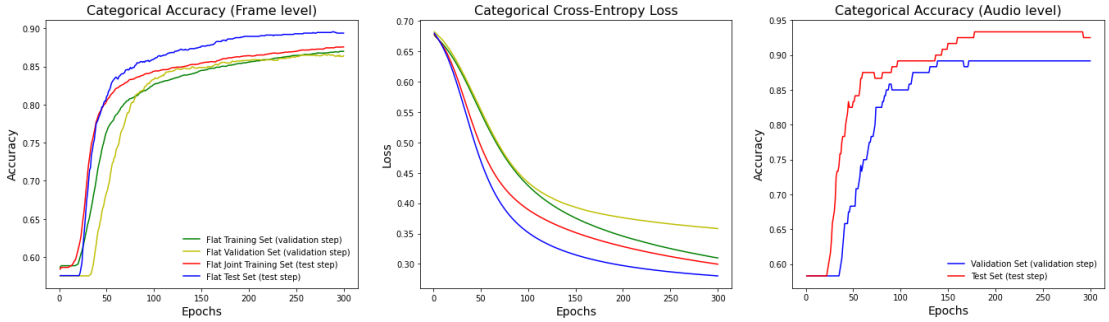


Figure 3.20: Config 1 - Comparing Validation and Test Steps

Lastly, tables in Fig. 3.21 show the confusion matrices of the network predictions at the audio level (3.21a) and at the frame one (3.21b).

		Actual Class	
		LC Alarm	*NoLC Alarm*
Prediction	*LC Alarm*	45	4
	NoLC Alarm	5	66

(a) Audio Level

		Actual Class	
		LC Alarm	*NoLC Alarm*
Prediction	*LC Alarm*	431	56
	NoLC Alarm	69	622

(b) Frame Level

Figure 3.21: Confusion Matrices

Considering the LC Alarm class as the *Positive* class and the NoLC Alarm class as *Negative* one, according to the value in these tables, we can derive three key measures:

- *Recall*: which indicates the ratio between the number of Positive samples correctly classified (True Positives, TP) in respect of the total number of Positives samples, also considering the misclassified ones (False Negative, FN):

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

- *Precision*: which indicates the ratio between the number of Positive samples correctly classified (True Positives, TP) in respect of the total number of samples classified as Positives, also considering the misclassified Negative samples (False Positives, FP):

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

- *Accuracy*: the metric we have evaluated until now in the above tests. It indicates the ratio between the number of samples correctly classified (True Positives and True Negatives) in respect of the total number of samples in the test set:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FP}$$

Therefore, at the end of our experimentation, we obtained a network that in testing phase achieved:

- At the Audio Level: a Recall equals to 0.9183, a Precision equals to 0.8823, and an Accuracy of 92.5%;
- At the Frame Level: a Recall equals to 0.8850, a Precision equals to 0.8620, and an Accuracy of 89.38%

Chapter 4

Conclusion

In this thesis, we tried to lay the groundwork for the integration of Deep Learning techniques within the Level Crossings Maintenance routine.

After a first problem statement, we performed a Systematic Literature Review with the aim to emphasize the actual degree of integration of Deep Learning for Smart Maintenance in the Railway Sector based on audio and video analysis. Results show that Deep Learning is a suitable solution to improve the current inspection routines in relation to different railways' infrastructures (tracks, catenaries, tunnels, etc.) and trains' components (e.g. bogies). It could help to improve maintenance tasks both in terms of time and accuracy, indeed the analysed studies achieved promising results in these directions. As expected, several papers have stressed the great limitation due to the lack of public data, therefore, in most cases, datasets were built from-scratch through disparate image acquisition systems. Nevertheless, although emphasizing the issue, they not specified if collected data are available to the researches' community, then we suppose not. Only one of them seems to give the possibility to access its data on request. Finally, only in one paper, the authors relied on audio data to estimate the health status of railways' components (Rail Tracks in particular), but the tests were carried out in the laboratory on rails' specimens.

Hence, a first step towards the application of DL techniques for Smart Maintenance at Level Crossings was performed. We decided to address, as a first issue, the audio classification task to discern LC Alarms from other similar sounds easily detectable near Level Crossings. Even if the final purpose of the whole project is to adopt DL-based approaches to address smart maintenance tasks, including the estimation of the remaining useful life of the level crossing system, in the case under examination, it is more correct to refer to Deep Learning techniques for Audio Detection and Classification.

Leveraging on state-of-the-art results, we implemented a classifier based on a VGG-like network called VGGish. Such a network, which is based in turn on the results achieved in [87], has been used to extract features from the AudioSet dataset. In our examination, we have removed the top fully connected layers building others two layers in order to perform our classification. Then, we considered some pre-trained weights for the convolutional layers. Lastly, we trained the network on a dataset composed of two classes: LC Alarm audio clips, collected from scratch and hand-labelled leveraging on some YouTube videos, and NoLC Alarm audio clips taken from the AudioSet. The tests we performed differ mainly in weights' consideration: in the first case we loaded the pre-trained weights and freezing them; in the second case, we loaded the weights but we allow them to variate during the training process; in the last case, we tested the network without loading the weights. The results we obtained show that in the last two cases the network has an unstable behaviour, therefore, they seem to validate the theoretical thesis that the dataset is not sufficiently large or sufficiently optimal to characterize a network with above 4.5 millions of parameters. Otherwise, in the first case, we obtained a correct behaviour of the network with promising results that must be improved through further analysis in the future.

Therefore, results validate an approach completely based on Transfer Learning. Indeed, the best validation performances were obtained considering the VGGish

network, loading its pre-trained weights freezing them and replacing only the last layers. Indeed, such configuration provides the best trade-off between accuracy and network stability. These results were also validated in the test stage, showing that the network achieved optimal generalization performances.

Chapter 5

Future Work

As introduced in section 2.1.2, the whole project is much wider than shown in this thesis work. It must involve also cost-effective analyses to evaluate if the use of Deep Learning, together with cameras, audio sensors and the support instrumentation, is a suitable solution to monitor the disproportionate number of level crossings.

Assuming that such an analysis will produce advantageous results, future works are related to the whole implementation of the proposed modular system.

First, the proposed LC Alarm Classifier must be improved by enabling it to face a three-class problem. Actually, the network as-is, trained on the whole dataset, has already been evaluated on another, although singe, YouTube video[1]. Figures 5.1 and 5.2 show that if no alarm is registered, the outputs of the network are quite low and similar (around 0.5); otherwise, while the Alarm bell is ringing, the LC Alarm output reaches values higher than 0.9 (networks outputs have been obtained in an off-line fashion). Nevertheless, by-the-book, it is not a correct approach since the model may not be robust to any disturbance. Therefore, a three-class (No Alarm, LC Alarm, and NoLC Alarm) classifier is needed.

As the above images show, some steps toward video analysis have already

[1]https://www.youtube.com/watch?v=GWC_RWE8a6k&t=21s&ab_channel=UKLevelCrossingsChannel

Figure 5.1: Level Crossing in a Steady State

Figure 5.2: Level Crossing in a Working State

been performed. The object detector that has been used is a YOLOv4-based detector trained on the COCO dataset (with 80 classes), which is available both on GitHub[2] and as Google Colaboratory Notebook[3]. Such network has been used without any edits, only for demonstrative purposes. The concept is to rely on this network, or similar ones, to detect the LC Traffic Lights in order to evaluate its status (blinking or steady). Then, the Bar Analysis Module must be implemented through a similar network or through an object tracking approach in order to

[2]https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial
[3]https://colab.research.google.com/drive/1_GdoqCJWXsChrOiY8sZMr_zbr_fH-0Fg?usp=sharing

estimate the remaining useful life of such bars.

Lastly, the Orchestrator Module must be implemented in order to evaluate the health status of the whole Level Crossing System leveraging on the outputs of the modules just mentioned.

Chapter 6

Acknowledgements

This thesis describes part of the research carried out during my internship at the University of Naples "Federico II". During this period I had the opportunity to collaborate with the international research group conducting the H2020 Shift2Rail project RAILS (Roadmaps for AI Integration in the raiL Sector) and interact with researchers from European universities in the Netherlands, United Kingdom and Sweden. I would like to thank Dr. Nikola Bešinović from the Delft University of Technology and Dr. Zhiyuan Lin from the University of Leeds for welcoming me in the RAILS project; I would like to express my gratitude to Prof. Francesco Flammini from the Linnaeus University, his guide was fundamental to take my first steps in the rail sector. A special thank goes to Hitachi Rail STS, in the person of Eng. Claudio Mazzariello, for having suggested the real scenario addressed in this thesis and for the time spent introducing and discussing its various aspects. Finally, thanks to Prof. Valeria Vittorini and Prof. Carlo Sansone, as well as to Eng. Stefano Marrone, for their close and constant supervision during all the activities I carried out in this period.

Bibliography

[1] A. M. Turing, "Computing machinery and intelligence," in *Parsing the Turing Test*, pp. 23–65, Springer, 2009.

[2] A. Annoni, P. Benczur, P. Bertoldi, B. Delipetrev, G. De Prato, C. Feijoo, E. Fernandez Macias, E. Gomez, M. Iglesias, H. Junklewitz, M. Lopez Cobo, B. Mertens, S. Nascimento, S. Nativi, A. Polvora, I. Sanchez, S. Tolan, I. Tuomi, and L. Vesnic Alujevic, "Artificial intelligence: A european perspective," tech. rep., Joint Research Centre (Seville site), 2018.

[3] E. Commission *et al.*, "Communication from the commission to the european parliament, the european council, the council, the european economic and social committee and the committee of the regions," *Artificial Intelligence for Europe. Brussels*, 2018.

[4] B. Copeland, "Artificial intelligence," 2019.

[5] T. Mitchell, *Machine Learning*. McGraw-Hill International Editions, McGraw-Hill, 1997.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2017.

[8] I. Witten, E. Frank, M. Hall, and C. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Elsevier Science, 2016.

[9] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[10] L. Chen, *Curse of Dimensionality*, pp. 545–546. Boston, MA: Springer US, 2009.

[11] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*, vol. 1. MIT press Massachusetts, USA:, 2017.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] D. Han, Q. Liu, and W. Fan, "A new image classification method using cnn transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43 – 56, 2018.

[14] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[15] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3320–3328, Curran Associates, Inc., 2014.

[16] A. B. Arrieta, N. D'iaz-Rodr'iguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garc'ia, S. Gil-L'opez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, tax-

onomies, opportunities and challenges toward responsible ai," *ArXiv*, vol. abs/1910.10045, 2020.

[17] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, 2017.

[18] G. Litjens, T. Kooi, B. Bejnordi, A. Setio, F. Ciompi, M. Ghafoorian, J. van der Laak, B. van Ginneken, and C. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017.

[19] Y. Yuan, Z. Xiong, and Q. Wang, "An incremental framework for video-based traffic sign detection, tracking, and recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1918–1929, 2017.

[20] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[21] L. Liu and R.-C. Chen, "A novel passenger flow prediction model using deep learning methods," *Transportation Research Part C: Emerging Technologies*, vol. 84, pp. 74–91, 2017.

[22] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[23] N. Ahuja and R. Horaud, "Introduction to the special volume on computer vision," *Artificial Intelligence*, vol. 78, no. 1, pp. 1 – 3, 1995. Special Volume on Computer Vision.

[24] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, pp. 309 – 320, 2019.

[25] S. Shawal, M. Shoyab, and S. Begum, "Fundamentals of digital image processing and basic concept of classification," *International Journal of Chemical and Process Engineering Research*, vol. 1, pp. 98–108, 06 2014.

[26] F. Y. Shih, *Image processing and pattern recognition: fundamentals and techniques*. John Wiley & Sons, 2010.

[27] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision* (K. Arai and S. Kapoor, eds.), (Cham), pp. 128–144, Springer International Publishing, 2020.

[28] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[29] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 421–432, 1998.

[30] S. V. Vaseghi, *Advanced digital signal processing and noise reduction*. John Wiley & Sons, 2008.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[33] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[35] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 818–833, Springer International Publishing, 2014.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[41] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[42] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[43] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[44] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[45] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[46] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[47] P. Soviany and R. T. Ionescu, "Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction," in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 209–214, IEEE, 2018.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[49] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[50] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[51] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, pp. 379–387, 2016.

[52] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[54] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[55] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[56] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing Journal*, vol. 70, pp. 41–65, 2018.

[57] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[58] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *CoRR*, vol. abs/1611.09326, 2016.

[59] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016.

[60] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[61] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International journal of multimedia information retrieval*, vol. 7, no. 2, pp. 87–93, 2018.

[62] S. Gould, T. Gao, and D. Koller, "Region-based segmentation and object detection," in *Advances in neural information processing systems*, pp. 655–663, 2009.

[63] A. Graves, S. Fernández, and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in *Artificial Neural Networks – ICANN 2007* (J. M. de Sá, L. A. Alexandre, W. Duch, and D. Mandic, eds.), (Berlin, Heidelberg), pp. 549–558, Springer Berlin Heidelberg, 2007.

[64] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville, "Reseg: A recurrent neural network-based model for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[65] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[66] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.

[67] M. Kristan, J. Matas, A. Leonardis, et Al., "The visual object tracking vot2015 challenge results," vol. 2015-February, pp. 564–586, 2015.

[68] G. Ciaparrone, F. Luque Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[69] M. Kristan, A. Leonardis, J. Matas, et Al., "The sixth visual object tracking vot2018 challenge results," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[70] S. Bai, Z. He, T. Xu, Z. Zhu, Y. Dong, and H. Bai, "Multi-hierarchical independent correlation filters for visual tracking," *CoRR*, vol. abs/1811.10302, 2018.

[71] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[72] L. Bertinetto, J. Valmadre, J. Henriques, A. Vedaldi, and P. Torr, "Fully-convolutional siamese networks for object tracking," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9914 LNCS, pp. 850–865, 2016.

[73] Y. Zhang, D. Wang, L. Wang, J. Qi, and H. Lu, "Learning regression and verification networks for long-term visual tracking," *CoRR*, vol. abs/1809.04320, 2018.

[74] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 103–119, Springer International Publishing, 2018.

[75] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4293–4302, 2016.

[76] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *CoRR*, vol. abs/1602.00763, 2016.

[77] L. Leal-Taixé, A. Milan, I. D. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *CoRR*, vol. abs/1504.01942, 2015.

[78] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *CoRR*, vol. abs/1703.07402, 2017.

[79] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," *CoRR*, vol. abs/1903.05625, 2019.

[80] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[81] H. Lu, H. Zhang, and A. Nayak, "A deep neural network for audio classification with a classifier attention mechanism," 2020.

[82] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2015.

[83] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[84] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 121–125, IEEE, 2018.

[85] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, pp. 1096–1104, 2009.

[86] L. Nanni, G. Maguolo, S. Brahnam, and M. Paci, "An ensemble of convolutional neural networks for audio classification," *arXiv preprint arXiv:2007.07966*, 2020.

[87] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[88] CENELEC, "En50126: Railway applications-the specification and demonstration of reliability," *Availability, Maintainability and Safety (RAMS)*, 2001.

[89] CENELEC, "En 50129: Railway application - communications, signaling and processing systems - safety related electronic systems for signaling," 2003.

[90] E. U. A. for Railways, "Report on railway safety and interoperability in the eu," 2020.

[91] A. Boniou, "About safer-lc–safer-lc project," 2020.

[92] M. A. B. Fayyaz, A. C. Alexoulis-Chrysovergis, M. J. Southgate, and C. Johnson, "A review of the technological developments for interlocking at level crossing," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, p. 0954409720941726, 2020.

[93] L. Khoudour, M. Ghazel, F. Boukour, M. Heddebaut, and E.-M. El-Koursi, "Towards safer level crossings: existing recommendations, new applicable technologies and a proposed simulation model," *European transport research review*, vol. 1, no. 1, pp. 35–45, 2009.

[94] J. Bokrantz, A. Skoogh, C. Berlin, T. Wuest, and J. Stahre, "Smart maintenance: an empirically grounded conceptualization," *International Journal of Production Economics*, vol. 223, p. 107534, 2020.

[95] SMaRTE (Smart Maintenance and the Rail Traveller Experience) project, "Deliverable 3.2 - experience map in passenger journeys," 2018.

[96] E. Fumeo, L. Oneto, and D. Anguita, "Condition based maintenance in railway transportation systems based on big data streaming analysis.," in *INNS Conference on Big Data*, pp. 437–446, 2015.

[97] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transportation Research Part C: Emerging Technologies*, vol. 45, pp. 17–26, 2014.

[98] H. Alawad, S. Kaewunruen, and M. An, "A deep learning approach towards railway safety risk assessment," *IEEE Access*, 2020.

[99] M. C. Nakhaee, D. Hiemstra, M. Stoelinga, and M. van Noort, "The recent applications of machine learning in rail track maintenance: A survey," in *International Conference on Reliability, Safety, and Security of Railway Systems*, pp. 91–105, Springer, 2019.

[100] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019.

[101] H. M. Hashemian, "State-of-the-art predictive maintenance techniques," *IEEE Transactions on Instrumentation and measurement*, vol. 60, no. 1, pp. 226–236, 2010.

[102] L. Zhang, J. Lin, B. Liu, Z. Zhang, X. Yan, and M. Wei, "A review on deep learning applications in prognostics and health management," *IEEE Access*, vol. 7, pp. 162415–162438, 2019.

[103] F. Chang, M. Liu, M. Dong, and Y. Duan, "A mobile vision inspection system for tiny defect detection on smooth car-body surfaces based on deep ensemble learning," *Measurement Science and Technology*, vol. 30, no. 12, 2019.

[104] F. Chang, M. Dong, M. Liu, L. Wang, and Y. Duan, "A lightweight appearance quality assessment system based on parallel deep learning for painted car body," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 8, pp. 5298–5307, 2020.

[105] P. Zhang, G. Zhang, W. Dong, X. Sun, and X. Ji, "Fault diagnosis of high-speed railway turnout based on convolutional neural network," in *2018 24th International Conference on Automation and Computing (ICAC)*, pp. 1–6, 2018.

[106] X. Chen, G. Zhang, W. Dong, X. Sun, and X. Cheng, "A novel fault diagnosis method for high-speed railway turnout based on dcae-logistic regression," in *2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 318–323, 2020.

[107] H. Wang, C. Zhang, N. Zhang, Y. Chen, and Y. Chen, "Fault diagnosis for igbts open-circuit faults in high-speed trains based on convolutional neural network," in *2019 Prognostics and System Health Management Conference (PHM-Qingdao)*, pp. 1–8, 2019.

[108] C. Huang, N. Qin, D. Huang, and K. Liang, "Convolutional neural network for fault diagnosis of high-speed train bogie," in *2019 Chinese Control Conference (CCC)*, pp. 4937–4941, 2019.

[109] Y. Santur, M. Karaköse, and E. Akin, "Big data framework for rail inspection," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–4, 2017.

[110] B. Hsueh, W. Li, and I. Wu, "Stochastic gradient descent with hyperbolic-tangent decay on classification," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 435–442, 2019.

[111] Y. Chen, B. Song, X. Du, and N. Guizani, "The enhancement of catenary image with low visibility based on multi-feature fusion network in railway industry," *Computer Communications*, vol. 152, pp. 200–205, 2020.

[112] D. Soukup and R. Huber-Mörk, "Convolutional neural networks for steel surface defect detection from photometric stereo images," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8887, pp. 668–677, 2014.

[113] Y. Santur, M. Karaköse, and E. Akin, "A new rail inspection method based on deep learning using laser cameras," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–6, 2017.

[114] Y. Santur, M. Karakose, and E. Akin, "An adaptive fault diagnosis approach using pipeline implementation for railway inspection," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 26, no. 2, pp. 987–998, 2018.

[115] N. AlNaimi and U. Qidwai, "Iot based on-the-fly visual defect detection in railway tracks," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 627–631, 2020.

[116] H. Yuan, H. Chen, S. Liu, J. Lin, and X. Luo, "A deep convolutional neural network for detection of rail surface defect," in *2019 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pp. 1–4, 2019.

[117] A. James, W. Jie, Y. Xulei, Y. Chenghao, N. B. Ngan, L. Yuxin, S. Yi, V. Chandrasekhar, and Z. Zeng, "Tracknet - a deep learning based fault

detection for railway track inspection," in *2018 International Conference on Intelligent Rail Transportation (ICIRT)*, pp. 1–5, 2018.

[118] Z. Liang, H. Zhang, L. Liu, Z. He, and K. Zheng, "Defect detection of rail surface with deep convolutional neural networks," vol. 2018-July, pp. 1317–1322, 2019.

[119] X. Wei, D. Wei, D. Suo, L. Jia, and Y. Li, "Multi-target defect identification for railway track line based on image processing and improved yolov3 model," *IEEE Access*, vol. 8, pp. 61973–61988, 2020.

[120] S. Yanan, Z. Hui, L. Li, and Z. Hang, "Rail surface defect detection method based on yolov3 deep learning networks," in *2018 Chinese Automation Congress (CAC)*, pp. 1563–1568, 2018.

[121] X. Zhang, K. Wang, Y. Wang, Y. Shen, and H. Hu, "An improved method of rail health monitoring based on cnn and multiple acoustic emission events," in *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 1–6, 2017.

[122] Y. Lin, C. Hsieh, W. Huang, S. Hsieh, and W. Hung, "Railway track fasteners fault detection using deep learning," in *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pp. 187–190, 2019.

[123] P. Chen, Y. Wu, Y. Qin, H. Yang, and Y. Huang, "Rail fastener defect inspection based on uav images: A comparative study," *Lecture Notes in Electrical Engineering*, vol. 640, pp. 685–694, 2020.

[124] X. Wei, Z. Yang, Y. Liu, D. Wei, L. Jia, and Y. Li, "Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 66–81, 2019.

[125] S. P. Orlov, R. V. Girin, and A. V. Piletskaya, "Intelligent information processing system for monitoring rail tracks," in *2019 III International Conference on Control in Technical Systems (CTS)*, pp. 233–236, 2019.

[126] Y. Liu, X. Sun, and J. Pang, "A yolov3-based deep learning application research for condition monitoring of rail thermite welded joints," pp. 33–38, 2020.

[127] J. Lee, S. Hwang, I. Choi, and Y. Choi, "Estimation of crack width based on shape-sensitive kernels and semantic segmentation," *Structural Control and Health Monitoring*, vol. 27, no. 4, 2020.

[128] Q. Guo, L. Liu, W. Xu, Y. Gong, X. Zhang, and W. Jing, "An improved faster r-cnn for high-speed railway dropper detection," *IEEE Access*, vol. 8, pp. 105622–105633, 2020.

[129] Y. Li, D. Wei, X. Wei, K. Wu, Y. Liang, and X. Shen, "Defects detection of catenary suspension device based on image processing and cnn," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1756–1761, 2019.

[130] Z. Liu, L. Wang, C. Li, and Z. Han, "A high-precision loose strands diagnosis approach for isoelectric line in high-speed railway," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1067–1077, 2018.

[131] Z. Liu, J. Zhong, Y. Lyu, K. Liu, Y. Han, L. Wang, and W. Liu, "Location and fault detection of catenary support components based on deep learning," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 1–6, 2018.

[132] P. Tan, X.-F. Li, J.-M. Xu, J.-E. Ma, F.-J. Wang, J. Ding, Y.-T. Fang, and Y. Ning, "Catenary insulator defect detection based on contour features

and gray similarity matching," *Journal of Zhejiang University: Science A*, vol. 21, no. 1, pp. 64–73, 2020.

[133] J. Chen, Z. Liu, H. Wang, A. Núñez, and Z. Han, "Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 2, pp. 257–269, 2018.

[134] J. Cui, Y. Wu, Y. Qin, and R. Hou, "Defect detection for catenary sling based on image processing and deep learning method," *Lecture Notes in Electrical Engineering*, vol. 640, pp. 675–683, 2020.

[135] H. Huang, J. Xu, J. Zhang, Q. Wu, and C. Kirsch, "Railway infrastructure defects recognition using fine-grained deep convolutional neural networks," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–8, 2018.

[136] C. Huang and Y. Zeng, "The fault diagnosis of catenary system based on the deep learning method in the railway industry," pp. 135–140, 2020.

[137] G. Kang, S. Gao, L. Yu, D. Zhang, X. Wei, and D. Zhan, "Contact wire support defect detection using deep bayesian segmentation neural networks and prior geometric knowledge," *IEEE Access*, vol. 7, pp. 173366–173376, 2019.

[138] G. Kang, S. Gao, L. Yu, and D. Zhang, "Deep architecture for high-speed railway insulator surface defect detection: Denoising autoencoder with multitask learning," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 8, pp. 2679–2690, 2019.

[139] G. Karaduman, M. Karakose, and E. Akin, "Deep learning based arc detection in pantograph-catenary systems," in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 904–908, 2017.

[140] J. Liu, Y. Wu, Y. Qin, H. Xu, and Z. Zhao, "Defect detection for bird-preventing and fasteners on the catenary support device using improved faster r-cnn," *Lecture Notes in Electrical Engineering*, vol. 640, pp. 695–704, 2020.

[141] Y. Luo, Q. Yang, and S. Liu, "Novel vision-based abnormal behavior localization of pantograph-catenary for high-speed trains," *IEEE Access*, vol. 7, pp. 180935–180946, 2019.

[142] Y. Shen, Z. Liu, and L. Chang, "A pantograph horn detection method based on deep learning network," in *2018 IEEE 3rd Optoelectronics Global Conference (OGC)*, pp. 85–89, 2018.

[143] P. Tan, X. Li, Z. Wu, J. Ding, J. Ma, Y. Chen, Y. Fang, and Y. Ning, "Multialgorithm fusion image processing for high speed railway dropper failure-defect detection," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2019.

[144] J. Wang, L. Luo, W. Ye, and S. Zhu, "A defect detection method of split pins in the catenary fastening devices of high-speed railway based on deep learning," *IEEE Transactions on Instrumentation and Measurement*, pp. 1–1, 2020.

[145] X. Wei, S. Jiang, Y. Li, C. Li, L. Jia, and Y. Li, "Defect detection of pantograph slide based on deep learning and image processing technology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 947–958, 2020.

[146] Y. Zhan, K. Linb, H. Zhan, Y. Guo, and G. Sun, "A unified framework for fault detection of freight train images under complex environment," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 1348–1352, 2018.

[147] Y. Zhang, M. Liu, Y. Chen, H. Zhang, and Y. Guo, "Real-time vision-based system of fault detection for freight trains," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 7, pp. 5274–5284, 2020.

[148] L. Xiao, B. Wu, Y. Hu, and J. Liu, "A hierarchical features-based model for freight train defect inspection," *IEEE Sensors Journal*, vol. 20, no. 5, pp. 2671–2678, 2020.

[149] J. Sun, Y. Xie, and X. Cheng, "A fast bolt-loosening detection method of running train's key components based on binocular vision," *IEEE Access*, vol. 7, pp. 32227–32239, 2019.

[150] Z. Yang, X. Gao, and Haibing Xia, "An improved faulting detection algorithm for subway tunnel segment," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 1710–1716, 2016.

[151] H.-W. Huang, Q.-T. Li, and D.-M. Zhang, "Deep learning based image recognition for crack and leakage defects of metro shield tunnel," *Tunnelling and Underground Space Technology*, vol. 77, pp. 166–176, 2018.

[152] Y. Xue and Y. Li, "A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 8, pp. 638–654, 2018.

[153] Q. Song, Y. Wu, X. Xin, L. Yang, M. Yang, H. Chen, C. Liu, M. Hu, X. Chai, and J. Li, "Real-time tunnel crack analysis system via deep learning," *IEEE Access*, vol. 7, pp. 64186–64197, 2019.

[154] R. S. Pahwa, J. Chao, J. Paul, Y. Li, M. T. Lay Nwe, S. Xie, A. James, A. Ambikapathi, Z. Zeng, and V. R. Chandrasekhar, "Faultnet: Faulty rail-valves detection using deep learning and computer vision," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 559–566, 2019.

[155] B. Zhao, M. Dai, P. Li, and X. Ma, "Data mining in railway defect image based on object detection technology," in *2019 International Conference on Data Mining Workshops (ICDMW)*, pp. 814–819, 2019.

[156] B. Zhao, M. Dai, P. Li, R. Xue, and X. Ma, "Defect detection method for electric multiple units key components based on deep learning," *IEEE Access*, pp. 1–1, 2020.

[157] D. F. García, I. García, F. J. dela Calle, and R. Usamentiaga, "A configuration approach for convolutional neural networks used for defect detection on surfaces," in *2018 5th International Conference on Mathematics and Computers in Sciences and Industry (MCSI)*, pp. 44–51, 2018.

[158] Q. Li and S. Ren, "A real-time visual inspection system for discrete surface defects of rail heads," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 8, pp. 2189–2199, 2012.

[159] J. Gan, Q. Li, J. Wang, and H. Yu, "A hierarchical extractor-based visual rail surface inspection system," *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7935–7944, 2017.

[160] S. Li, X. Zhao, and G. Zhou, "Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 7, pp. 616–634, 2019.

[161] W. Liu, Z. Liu, A. Núñez, L. Wang, K. Liu, Y. Lyu, and H. Wang, "Multi-objective performance evaluation of the detection of catenary support components using dcnns," *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 98 – 105, 2018. 15th IFAC Symposium on Control in Transportation Systems CTS 2018.

[162] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *CoRR*, vol. abs/1904.07850, 2019.

[163] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2403–2412, 2018.

[164] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 153–164, 2017.

[165] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[166] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015.

[167] Y. Zhang, "Hierarchical feature matching of fault images in tfds based on improved markov random field and exact height function," *M.S. thesis, Department of Mechanical Engineering, Hubei University Technology, Wuhan, China*, 2017.

[168] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[169] J. Salamon and J. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[170] K. Piczak, "Environmental sound classification with convolutional neural networks," vol. 2015-November, 2015.

[171] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," vol. 2016-May, pp. 6440–6444, 2016.

[172] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An ieee aasp challenge," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 1–4, IEEE, 2013.

[173] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, (New Orleans, LA), 2017.

[174] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[175] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.