Universidade do Minho
Escola de Engenharia
Departamento de Informática

Luís Filipe Fernandes Gomes

Connectionist Systems for Image Processing and Anomaly Detection

June 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Luís Filipe Fernandes Gomes

Connectionist Systems for Image Processing and Anomaly Detection

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Professor Cesar Analide

June 2021

COPYRIGHT AND TERMS OF USE FOR THIRD PARTY WORK

ACKNOWLEDGEMENTS

The writing of this dissertation would not have been possible without the support and contribution of many people and organizations.

Desejo expressar a minha infinita gratidão aos meus pais, Joaquim Edmundo Gomes e Júlia Maria Fernandes, sem os quais nada seria possível. Por todo o apoio, financeiro e emocional, e por todos os esforços para me manter a estudar, não apenas durante a escrita da dissertação, mas durante toda a minha vida de estudante. A toda a minha família, muito obrigado por se preocupar comigo e cuidar do meu bem-estar. [I would like to express my endless gratitude to my parents, Joaquim Edmundo Gomes and Júlia Maria Fernandes, without whom nothing would be possible. For all their support, financial and emotional, and for all their efforts to keep me studying, not only during dissertation writing but during all my student life. To my whole family, my thanks for concerning with me and caring for my well-being.]

I would like to thank my supervisor, Professor Cesar Analide, who proposed the dissertation subject, allowing me to work in interesting Artificial Intelligence areas, which have a strong presence and usability in nowadays solutions. Also for his availability to help me with research and development decisions, for the recommendations and for all the trust in my work.

The main case study, presented in chapter 6, is the follow-up of a project developed in a cooperation between the Department of Informatics and the Department of Civil Engineering of the University of Minho[1]. Regarding this project, I would like to express my very great appreciation to Professor Paulo António Pereira, Principal Investigator. I want to offer my special thanks to Professor Elisabete Fraga de Freitas for transmitting me all the needed knowledge in the pavement monitoring scope and for supervising my work with a critical view in relation to the civil engineering understanding. Again, to Professor Cesar Analide, my supervisor in the projects in respecting to the computer science scope. Additionally, my thanks are extended to the technical staff of the Department of Civil Engineering of University of Minho, namely, Engineer Carlos Alberto Palha, for his aid setting everything up for the software tests on the roads and during the data acquisition process.

The *Crowds Detection* research (section 7.3), is developed within the Project MARÉ (`https://www.mare-project.org/`). I wish to acknowledge the guidance provided by Professor Nuno Jardim Nunes, main project coordinator. I am particularly grateful for the assistance given by Professor Rui Maranhão Abreu, always present to help me doing my best. I want to thank my back-end partner, Rui Pedro Calheno, who worked in close proximity with me and who have supported me making decisions during the project. To everyone who worked in the project, it is also expressed my appreciation.

Regarding the *Mechanical Structural Design* study, I would like to thank Professor Sérgio Manuel Tavares for give me the chance to work in this application. I would like to express my great gratitude to Engineer João Paulo Ribeiro, my friend and colleague in this work, for his help with the domain understanding of Mechanical Engineering topics and for his valuable goodwill.

1 Part of this work was funded by scholarships *CTAC/UID/ECI/04047/2013* and *UID/ECI/4047/2019*, financed by national funds FCT/MCTES.

Many thanks to all teachers and professors for having taught me everything I know so far, empowering me to complete this work. To all my friends and colleagues, I would like to thank for their support and for the always needed relaxation moments. To everyone that took part in my academic and professional path and all the ones who walked it with me. To HUBS, the company that welcomed me in recent months, and to my entire team, especially to my manager, engineer Héctor Mouriño-Talín, for all the new things I learned from him, and to my colleague Filippo Lo Bue, who worked with me in close proximity. Thanks should also go to my kickboxing team and to my coach, *Mestre* Manuel Gomes, for helping me to develop a mindset of persistence and overcoming. Last but not least, I would like to manifest my esteem to my friend, Engineer Cláudia Sofia Correia, who generously offered feedback on the dissertation content, above all for her encouragement in moments when working and writing was difficult.

To all of you and to anyone I've forgotten,
my sincerely appreciation,

Luís Filipe Fernandes Gomes

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

RESUMO

A Inteligência Artificial (IA) e a Ciência de Dados estão cada vez mais presentes no nosso quotidiano e os benefícios que trouxeram para a sociedade nos últimos anos são notáveis. O sucesso da IA foi impulsionado pela capacidade adaptativa que as máquinas adquiriram e está estreitamente relacionada com a sua habilidade para aprender. Os sistemas conexionistas, apresentados na forma de Redes Neurais Artificiais (RNAs), que se inspiram no sistema nervoso humano, são um dos mais importantes modelos que permitem a aprendizagem. Estes são utilizados em diversas áreas, como em problemas de previsão ou classificação, apresentando resultados cada vez mais satisfatórios. Uma das áreas em que esta tecnologia se tem destacado é a Visão Computacional (*Computer Vision (CV)*), permitindo, por exemplo, a localização de objetos em imagens e a sua correta identificação. A Deteção de Anomalias (*Anomaly Detection (AD)*) é outro campo onde as RNAs vêm surgindo como uma das tecnologias para a resolução de problemas. Em cada área são utilizadas diferentes arquiteturas de acordo com o tipo de dados e o problema a resolver. Combinando o processamento de imagens e a deteção de anomalias, verifica-se uma convergência de metodologias que utilizam módulos convolucionais em arquiteturas dedicadas a AD. O objetivo principal desta dissertação é estudar as técnicas existentes nestes domínios, desenvolvendo diferentes arquiteturas e modelos, aplicando-as a casos práticos de forma a comparar os resultados obtidos em cada abordagem. O caso prático principal consiste na monitorização de pavimentos rodoviários por meio de imagens para a identificação automática de áreas degradadas. Para isso, dois protótipos de software são propostos para recolher e visualizar os dados adquiridos. O estudo de arquiteturas de RNAs para o diagnóstico da condição do asfalto por meio de imagens é o foco central no processo científico apresentado. Os métodos de *Machine Learning (ML)* utilizados incluem classificadores binários, *Autoencoders (AEs)* e *Variational Autoencoders (VAEs)*. Para os dois últimos modelos, práticas supervisionadas e não supervisionadas são também comparadas, comprovando a sua utilidade em cenários onde não há dados rotulados disponíveis. Usando o modelo VAE num ambiente supervisionado, este apresenta uma excelente distinção entre áreas de pavimentação em boas condições e degradadas. Quando não existem dados rotulados disponíveis, a melhor opção é utilizar o modelo AE, utilizando a distribuição de semelhanças das reconstruções para calcular o *threshold* de separação, atingindo *accuracy* e *precision* superiores a 94%). O processo completo de desenvolvimento mostra que é possível construir uma solução alternativa para diminuir os custos de operação em relação aos sistemas comerciais existentes e melhorar a usabilidade quando comparada às soluções tradicionais. Adicionalmente, dois estudos demonstram a versatilidade dos sistemas conexionistas na resolução de problemas, nomeadamente no projeto de estruturas mecânicas, possibilitando a modelação de campos de deslocamento e pressão em placas reforçadas; e na utilização de AD para identificar locais de aglomeração de pessoas através de técnicas de *crowdsensing*.

PALAVRAS-CHAVE Sistemas Conexionistas, Redes Neuronais Artificiais, *Deep Learning*, Ciência de Dados, Visão Computacional, Deteção de Anomalias, *Autoencoders*, *Variational Autoencoders*, Monitorização Automática de Pavimentos.

ABSTRACT

Artificial Intelligence (AI) and Data Science (DS) have become increasingly present in our daily lives, and the benefits it has brought to society in recent years are remarkable. The success of AI was driven by the adaptive capacity that machines gained, and it is closely related to their ability to learn. Connectionist systems, presented in the form of Artificial Neural Networks (ANNs), which are inspired by the human nervous system, are one of the principal models that allows learning. These models are used in several areas, like forecasting or classification problems, presenting increasingly satisfactory results. One area in which this technology has excelled is Computer Vision (CV), allowing, for example, the location of objects in images and their correct identification. Anomaly Detection (AD) is another field where ANNs have been emerging as one technology for problem solving. In each area, different architectures are used according to the type of data and the problem to be solved. Combining image processing and the finding of anomalies in this type of data, there is a convergence of methodologies using convolutional modules in architectures dedicated to AD. The main objective of this dissertation is to study the existent techniques in these domains, developing different model architectures, and applying them to practical case studies in order to compare the results obtained in each approach. The major practical use case consists of monitoring road pavements using images to automatically identify degraded areas. For that, two software prototypes are proposed to gather and visualise the acquired data. Moreover, the study of ANN architectures to diagnose the asphalt condition through images is the central focus of this work. The experimented methods for AD in images include a binary classifier network as a baseline, Autoencoders (AEs) and Variational Autoencoders (VAEs). Supervised and unsupervised practises are also compared, proving their utility also in scenarios where there is no labelled data available. Using the VAE model in a supervised setting, it presents a excellent distinction between good and bad pavement areas. When labelled data is not available, using the AE and the distribution of similarities of good pavement reconstructions to calculate the threshold is the best option with both accuracy and precision above 94%. The full development process shows it is possible to build an alternative solution to decrease the operation costs relatively to expensive commercial systems and improve usability when compared with traditional solutions. Additionally, two case studies demonstrate the versatility of connectionist systems to solve problems, namely in Mechanical Structural Design enabling the modelling of displacement and pressure fields in reinforced plates; and using AD to identify crowded places through crowd-sensing techniques.

KEYWORDS Connectionist Systems, Artificial Neural Networks, Deep Learning, Data Science, Computer Vision, Anomaly Detection, Autoencoders, Variational Autoencoders, Automatic Pavement Monitoring.

CONTENTS

ACRONYMS

AAE Adversarial Autoencoder
AD Anomaly Detection
AE Autoencoder
AI Artificial Intelligence
ANN Artificial Neural Network
AP Access Point
API Application Programming Interface
AUC Area Under the Curve
AUROC Area Under the Receiver Operating Characteristic

CNN Convolutional Neural Network
CPU Central Processing Unit
CSV Comma Separated Value
CV Computer Vision

DAD Deep Anomaly Detection
DB Database
DBMS Database Management System
DBN Deep Belief Network
DL Deep Learning
DNN Deep Neural Network
DS Data Science

EDA Exploratory Data Analysis

FPR False Positive Rate

GAN Generative Adversarial Network
GDPR General Data Protection Regulation
GPU Graphics Processing Unit
GUI Graphical User Interface

ICMP Internet Control Message Protocol
ILSVRC ImageNet Large Scale Visual Recognition Challenge
IQR Interquartile Range

JSON JavaScript Object Notation

KNN K-Neareast Neighbors

LSTM Long Short-Term Memory

MAE Mean Absolute Error
MAPE Mean Absolute Percentage Error
ML Machine Learning
MLP Multi-Layer Perceptron
MNIST Mixed National Institute of Standards and Technology
MSE Mean Square Error

PCA Principal Component Analysis

RBM Restricted Boltzmann Machine
REF Recursive Feature Elimination
ReLU Rectified Linear Unit
RMSE Root Mean Square Error
RNN Recurrent Neural Network
ROC Receiver Operating Characteristic

SGD Stochastic Gradient Descent
SOTA State-of-the-Art
SQL Structured Query Language
SSIM Structural Similarity Index Measure
SVM Support Vector Machines

TNR True Negative Rate
TPR True Positive Rate

VAE Variational Autoencoder

XML Extensible Markup Language

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

> *Knowledge of the past and of the places of the earth is the ornament and food of the mind of man.*
>
> Leonardo da Vinci

Artificial Intelligence (AI) is becoming increasingly present in our daily lives. The benefits that it has brought to society in recent years are remarkable. Its use in daily complex life tasks, like autonomous driving, proves that the human being has more and more confidence in this kind of solution. The success of AI models in the last few years is closely related to its ability to learn, Machine Learning (ML).

Connectionist Systems, commonly referred to as Artificial Neural Network (ANN), are systems inspired by the human nervous system. One of the most popular tasks performed by these models is pattern recognition. The network is fed with data samples as input and the corresponding data labels as output, learning the relations between them, during the training process. In image processing the inputs of the networks are the image pixels. In such a scenario, the main goal is to localise objects or classify the images through the correct identification of pixels patterns.

Anomaly Detection (AD), also known as outliers detection, is another group of problems where AI has an important role. This problem consists of the identification of unusual observations, that differ in some aspects from the rest of the data samples. Applying AD to the Computer Vision (CV) domain, one aspires to identify in a dataset a subset of images that has unique characteristics from most of the entries.

1.1 BACKGROUND

In the past few years, AI has gained greater visibility in the social media and imposed itself in the society, putting a lot of tools at our disposal. AI is everywhere now, serving the widest variety of purposes. For instance, it can be found in virtual assistants, self-driving cars, recommendation systems and even in the smartphone cameras, post-processing images and improving their quality and visually appealing. With the advances in computer architectures and the excellent results shown by this technology, there is a growing interest in the computer science community on this topic. Efforts are being made to expand the AI areas of application and to improve its performance even more.

ANNs are one of the ML model types that benefit from this reality since they need a great computational power to process the large amounts of data that are fed to them. Based on the human brain architecture, these

extremely parallel systems divide the complexity of a problem, computing the relationship of input-output pairs, aggregating the calculations of diverse nodes, called neurons.

CV is one of the specific areas where this truth is noticeable, making use of deeper ANNs to give machines the ability to understand the meaning of image pixels. Nowadays, there is a wide range of CV algorithms with real-life value, performing tasks like object detection, facial recognition or image segmentation. Since the methods of image processing imply a lot of vectors calculus, they can benefit from the vector instructions of many processors or even the more powerful capacity of Graphics Processing Units (GPUs).

Another field where ML can give its contribution is in tasks involving the detection of anomalous situations or occurrences inside a specific domain. In this problem, there are two main approaches depending on the data availability and the specific domain needs. In the unsupervised approach, the system can learn to absorb the patterns of the normal data and inferring that an anomaly occurs when some data doesn't fit the pattern. In the supervised paradigm, the systems receive information about the two cases for the data, normal and anomaly, and learn how to differentiate the patterns of the different classes.

Combining the CV and AD study fields, it is expected to find a method capable of identifying images that differ from what is initially expected. There are a lot of practical examples which need a solution where these models can be applied. The most obvious situations are related to the quality control process. The detection of defective items is a very important stage in every industrial process, improving the excellence of the product to the consumer.

This process can be directed to an infinity of areas. Every case that has a notion of *normal* and *abnormal* situations (and possibly a correspondent representation in images) can be treated the same way. The *abnormal* cases are mapped to the anomalies in the model. The following list provides some examples of real cases that correspond to the problem under analysis:

- **Manufacturing machines** - To detect anomalies in manufacturing systems (Kammerer et al., 2019);
- **Medical images** - To detect malignant tumors in mammogram, CT or PET images (Wei et al., 2018);
- **Road Pavements** - To detect degraded pavement parts in highway roads (Seraj et al., 2014);

The resultant identification can be used directly or indirectly, for example to discard some items automatically or to feed the results to a decision-making process.

For each specific mentioned problem, there are different key participants and different people interested in the problem's resolution. However, they are correlated, since in every case, there are service providers and consumers. The traditional methods to detect anomalies include the manual checking of each item. With a good automatic detection system, the service providers can reduce their costs (for example, the production cost in industries or maintenance costs in road pavements) and the consumers can have better service quality.

Taking the last specific example, the pavement classification problem in highway roads, one can see the customers as the drivers who use the infrastructures and the service provider as the governments or other private entities responsible for the road maintenance. Currently, it is difficult to monitor the highways pavements efficiently, since the process is done by observing the pavement directly, which is a very time-consuming and laborious task. Other available solutions have high associated costs, moving away from them the service providers, whose objective is to minimise the spending on road monitoring and conservation.

With all that in mind, the main objective of the present work is to study different connectionist systems models, building, applying them and proving their reliability to solve problems, specially in the CV and AD fields. The central work focus is to characterise the problem of pavement classification in highways, coming up with a robust methodology for solving similar problems and a low-cost prototype capable of classify the pavement in two main categories: *Good pavement* and *Bad pavement* (anomalies).

1.2 MOTIVATION

Data Science (DS) and ML are the areas that interest me the most. The idea of giving intelligence to computers and machines presented in movies was always compelling and curious to me. For my Master's degree, I chose *Data Science* and *Intelligent Systems* as specialisations. In both profiles, I applied a variety of ANN techniques to process data and extracted useful information from it. Based on that, the main motivation for this dissertation is to understand in depth how connectionist systems work and how they can be used to solve real-life problems.

Furthermore, the present work arises from the follow-up of a research project to solve the pavement monitoring problem to detect degradations. In that sense, the study of image processing and AD connected to ANNs are also suited to that problem context.

This work is important for those who are interested in AI related fields, considering the theoretical hypothesis presented. Also for the pavement conservation organisations, that need to reduce their costs when monitoring the roads networks, this work presents some solutions that can be further applied.

1.3 OBJECTIVES

This research aims to study the existing techniques used to solve problems in the following specific ML areas:

- Connectionist Systems - Use of Artificial Neural Network architectures to solve problems;
- Anomaly Detection - Detection of anomalies among a set of cases;
- Image Processing and Computer Vision - Simulate human vision in machines and give meaning to images inside a computer.

Figure 1.1 shows how these fields interact with each other in this research scenario.

Note that the Venn diagram does not represent all the subfields of each subject and it is not intended to assume that each shape expresses the whole study area. For example, there are AD techniques outside the ML field, which also include more than the 3 inner domains.

For each specific case, different aspects characterise the problem, and it is necessary to understand them and identify the difficulties in its resolution. The next step is to seek the current solutions available to solve the issue and evaluate its advantages and disadvantages.

To understand what can be improved from the current solutions, it is needed to investigate the used technologies in the described domains. From those technologies, some have to be chosen and possibly used together to reach the main goal.

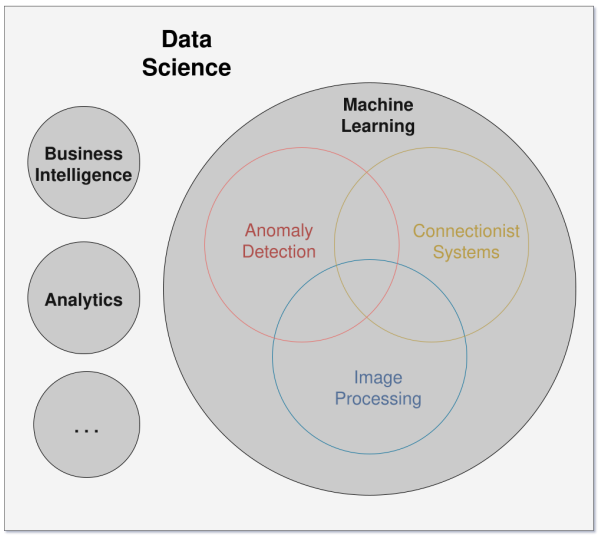


Figure 1.1: Research subjects Venn diagram.

The main goal of this research is to create a methodology that can distinguish anomalies from the rest of the cases, using ANNs. This objective can be unwinded into theoretical and practical sub-goals. The theoretical objectives are the ones related to the study of ML techniques and scientific proof:

T1) Study methodologies within the scope of problem solving using connectionist systems;

T2) Explore different techniques in CV and AD, focusing in ML approaches;

T3) Evaluate the performance of ANNs architectures in each domain;

T4) Compare empirically the proposed solutions.

Since the main practical problem that is intended to solve is related to pavement monitoring and classification, the following list shows the main practical goals:

P1) Study the existent solutions for pavement monitoring;

P2) Build a system to gather data of pavements, including pictures from the road;

P3) Build a system to visualise the collected data, providing the ability to understand the pavement conditions;

P4) Study the existent solutions for automatic pavement classification;

P5) Build ML models to identify pavement degradations automatically;

P6) Identify the best approach for the problem, comparing the different built models.

1.4 DOCUMENT OVERVIEW

This document is structured in three parts. The first part presents the theoretical fundamentals that serve as the base for the practical approaches presented in the second one. The third part presents appendices that can be useful during the reading of the document.

The chapters in **Part I** are structured in order to go from the general to the specific research topics, as follows:

Chapter 2 - Artificial Intelligence It is made a description about what is DS, why it is important nowadays and an overview about the field. It is also presented a formal definition for this area of study and a generic workflow that can be applied in DS projects.

The ML concept and what kind of models are included in this approach are also presented in this chapter. The distinction between learning paradigms is explored and how the respective models interact with data. It is presented a proposed workflow to build and train ML models using data, as sub-step of the DS workflow.

Lastly are presented performance metrics for AI models and useful tools in this area.

Chapter 3 - Connectionist Systems The goal of this chapter is to present the specific technology that will be applied in the practical use cases. Connectionist systems are the main focus of research in this work, whereby beyond the definition of these models, a brief historical overview and the way they appeared is presented. Furthermore, technical details about their internal functioning and their mathematical fundamentals are depicted in this chapter.

Chapter 4 - Image Anomaly Detection In the last chapter of the fundamentals part, a closer look is made at the detection of image anomalies. It is made the presentation about what is CV and how machines can perceive images, assigning meaning to them. Are also presented the most common used transformations made to images to improve the computer capability to understand pixels values. Finally, some applications of modern CV approaches are described, explaining in detail the use of convolutions inside ANN, and the State-of-the-Art (SOTA) work in this area.

Regarding the AD field, the formal definition of *anomalies* and their different types are presented. Additionally are shown technologies that can be applied in this area. It is also presented the related work and possible combinations of AD with other areas.

Tools to perform AD in images are presented in the last section of this chapter.

Part II shows the development of tools and models to reach the proposed goals. The chapters included in this part are the following:

Chapter 5 - Preliminary Studies The first practical case study is presented in this chapter. First, is presented some related work in the intersection of the 3 areas in study: Connectionist systems, Anomaly Detection and Image Processing. It is used the Mixed National Institute of Standards and Technology (MNIST) dataset to set up experimental models that are used to validate the proposed approaches. The two purposed workflows presented in Chapter 2 are used to drive the process, even though not all steps need to be taken (for example, the data gathering). In this way the chapter is organised accordingly the workflow steps, presenting the problem definition

and the path to the solution, using ANN models. The conclusion section presents the validity of the models, comparing their results.

Chapter 6 - Highway Pavements Monitoring In this chapter, it is analysed the main case study of this project, the pavement monitoring problem. It also follows the same proposed workflows as the previous chapter and it is roughly structured following the same pattern. In this case, all the steps are taken, and consequently, after the introductory section where the problem background is presented, it is presented the domain understanding, including: the presentation of the degradations in the pavement; the traditional solutions used to solve the monitoring problem; and related studies in this specific area. After the formal problem definition, the proposed solutions are presented in the respective sections, constituting a full prototype to pavement monitoring: the way data gathering and storage are performed, a user-friendly data analysis tool, as well as the models to classify the pavement. In the result analysis section is made the theoretical study for the ML and ANN field, about the different built models, depicting the consequences of different approaches to reach the final classification methodology. In the conclusion section, the models are compared presenting advantages and disadvantages for each method.

Chapter 7 - Additional Applications The final practical chapter presents additional use cases of ANN models in different use cases. The first use case, presents how connectionist systems can be used in the Mechanical Structural Design in order to design a reinforced structure. In this study the goal is to predict the stress and displacement fields on a plate for a given solicitation. The second use case lies in the intersection of AD and connectionist systems fields, and the main objective is to create ML models to detect crowded points in university campi during the COVID-19 pandemic using the data of wireless access points. This includes the use of different ANNs to detect anomalous situations, to estimate the current number of people in each place and to predict their future occupancy.

Chapter 8 - Conclusions The closing chapter presents the conclusions that can be inferred from the project as a whole. It is presented a critical view of the presented solutions, a comparison with the previous existent methods and a discussion presenting their main advantages and disadvantages. Also in this chapter are presented proposals for future work to follow up the present piece of research.

Figure 1.2 presents the prerequisites graph for the document. Each arrow indicates the recommended reading order, accordingly to the prerequisites of each chapter.

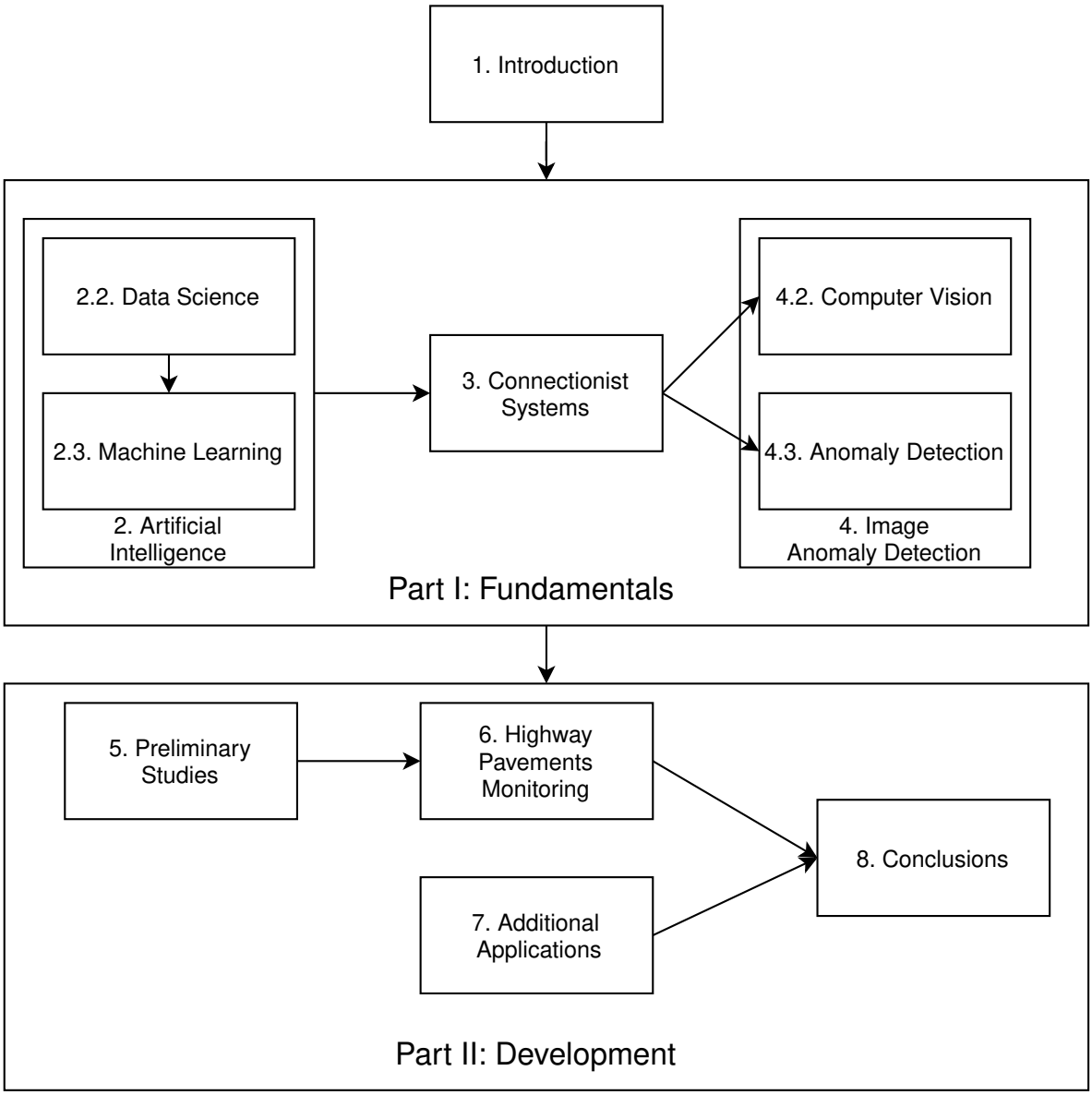


Figure 1.2: Document organisation graph.

Part III presents auxiliary results related to the experiments performed, and it is recommended to consult them while reading Part II.

Part I

FUNDAMENTALS

2

ARTIFICIAL INTELLIGENCE

> *An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behaviour.*
>
> Alan Turing

2.1 INTRODUCTION

In the last few decades, the world is undergoing very rapid technological changes. This truth is particularly evident when one looks at the IT sector. The capacity to compute information has grown exponentially with the increasing number of transistors per processor, as predicted by Moore's Law. Correspondingly, the prices of processors, memory and storage went down drastically due the expanding supply.

The shift from analogue to digital was an important factor to improve the ability to store and transmit data. As shown by Hilbert and Lopez (2011), digital format growth in the early 2000s is obvious. The digital percentage in telecommunications went from 19.8% (1987) to 99.9% (2007) and similar values are presented for storage, with 0.8% (1987) and 94% (2007).

Artificial Intelligence (AI) human behaviour and intelligence, using mechanism that are similar to the human reasoning and learning processes. It was defined by McCarthy (2007), the "father" of AI, as *the science and engineering of making intelligent machines*. The growth of computational power benefits this field on a large scale, since more powerful hardware expands the possibilities for the creation of AI applications.

In the first section of this chapter is presented a study area where AI is very important nowadays. Data Science (DS) uses AI in the form of Machine Learning (ML) models to get insights about data. The specificity of these models is depicted on the second section. Additionally are also presented performance metrics and a set of tools for AI.

2.2 DATA SCIENCE

The rise of the internet and new devices such as laptops, smartphones or even more recent ones as smartwatches have allowed people to stay connected 24 hours a day, consuming and generating massive quantities of

data at every second. Also, the ability to perceive and model our environment through sensors (e.g. temperature, humidity, sound and image) came to reinforce the importance of data in quotidian problem-solving (van der Aalst, 2016).

With the growing amount of data, the term "Big Data" is now an undeniable presence, especially in big technology companies. Big Data is not only referring to large quantities of data, but also to the correspondent computing resources that need to be allocated to it. Initially summarised by the 3 V's mnemonic - volume, velocity and variety - (Oussous et al., 2018; Sagiroglu and Sinanc, 2013), it is now common to find two more concepts in the literature, making a total of 5 V's (Ishwarappa and Anuradha, 2015):

- Volume - Extensive amounts of data are collected from every device with a internet connection. It is important to get this data processed and stored, to get insightful information from it.
- Velocity - Data is constantly being acquired at every second and it need to be used in a fast way. As an example, anomaly detection in bank transactions have to be fast enough, in near real-time, to notify the clients as soon as possible.
- Variety - Different sources and different data formats (e.g. text, image, sound and video) make the data heterogeneous in the way it is gathered. Structured and unstructured data can be find in the storage infrastructures.
- Veracity - Denotes the quality and accuracy of data;
- Value - Represents the ultimate goal of acquiring and storing data, turning it into an asset.

The expression "Data is the new oil." is widely used nowadays, denoting the its monetary significance. Furthermore, this analogy is also truth comparing the way these assets are managed. Oil is not profitable before being converted into other materials like gasoline or plastic. Likewise, data is useless when stored with no purpose. It is necessary to extract knowledge from it through Data Mining processes, and use it afterwards.

2.2.1 *Definition*

The role of Data Science is precisely to help organisations to transform their data into value. Among the many definitions for Data Science in the literature, Wil van der Aalst presents that concept in a very complete way (van der Aalst, 2016):

> *"Data science is an interdisciplinary field aiming to turn data into real value. Data may be structured or unstructured, big or small, static or streaming. Value may be provided in the form of predictions, automated decisions,models learned from data, or any type of data visualisation delivering insights. Data science includes data extraction, data preparation, data exploration, data transformation, storage and retrieval, computing infrastructures, various types of mining and learning, presentation of explanations and predictions, and the exploitation of results taking into account ethical, social, legal, and business aspects."*

The same author refers that a data scientist can use the available data to produce value in four different ways (van der Aalst, 2014):

- **Reporting** - Understand what events happened in the past;
- **Diagnosis** - Explain how and why they took place;
- **Prediction** - Anticipate future events, often based on the past ones;
- **Recommendation** - Interpret the data correctly and make valuable decisions from it.

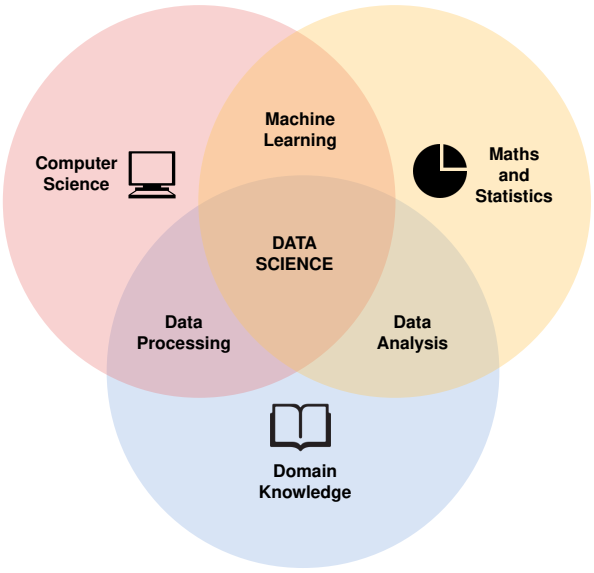


Figure 2.1: "The Data Science Venn Diagram" (Conway, 2010).

To complement these definitions, a visual abstraction of what is included in Data Science can be given by the "Data Science Venn Diagram" (Conway, 2010) presented in Figure 2.1.

As the diagram suggest, a data scientist has to understand the domain of the problem, being capable of process and analyse data, but also use ML techniques to get insightful information from it.

2.2.2 *Workflow*

There isn't an universal definition for the Data Science workflow in the literature; however there are some key steps that are common in every problem (Saltz, 2021). Based on the definition presented in previous sub-section, one can build a workflow for data science projects. Each data scientist will have their own way to work, and the presented workflow (Figure 2.2) is a personal view to better understand the steps and decisions made during this work.

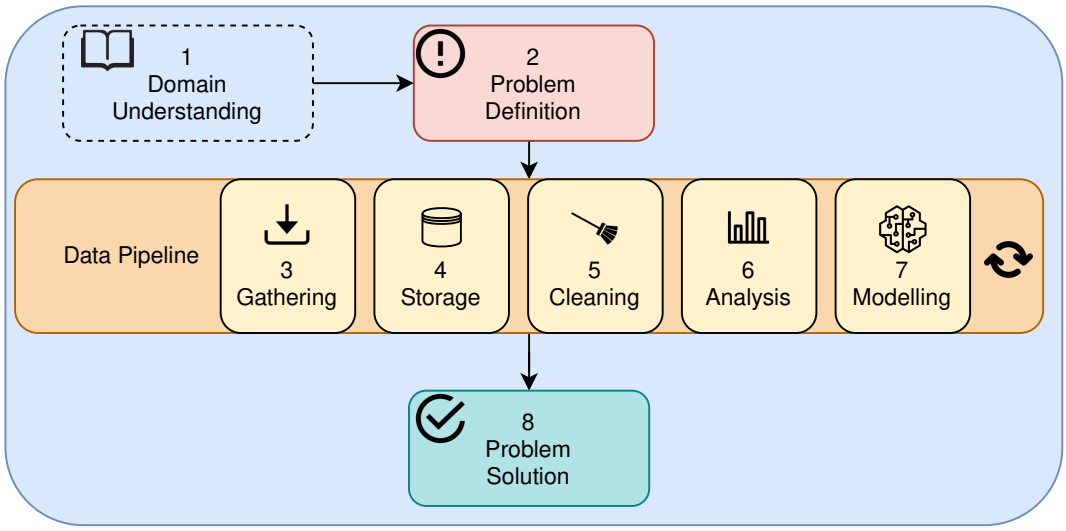


Figure 2.2: Proposed DS Workflow.

The Domain Understanding is the background for the whole process. After defining the problem, there is a set of stages involving data, from gathering to its final utilisation for building models. This pipeline can be repeated as many times as needed to improve the final results until achieve a solution for the problem, changing also the order of the steps if necessary. The steps presented in Figure 2.2 are described in this sub-section as follows.

1 - Domain Understanding

The first to solve any problem is to understand it. This crucial phase is what makes possible all the other ones. All the roles involved in the Data science pipeline have to comprehend the business context. After having knowledge of the field, studying it with some depth and getting all the needed information, it is important to define the end objective of the process. This includes answering some questions:

1. What problem is meant to be solved and why is it a problem?
2. What are the current solutions (if any)?
3. What data can be used to solve it?
4. How can the performance of the results be evaluated?

These question are the start point for all the development. They will guide the data scientist, specially during the problem definition stage, that is the first practical step.

2 - Problem Definition

To define the problem, it is necessary to build an abstraction of the real world. This is achieved using data and its correspondent meaning. For example, to represent the temperature in a room based on thermometer measurements, there are many ways to do it. The data can be saved as an integer or a real value or even a text label, indicating the warmness of the room. These decisions are made in the problem definition step.

For each specific problem, the collected data can always be represented as a set of variables. Depending on the variable nature, it can be defined in different ways:

- **Qualitative** or categorical values are the ones that have classes associated with them. categorical variables can be ordinal, assuming an order, for example *low*, *medium* and *high* or nominal, without a order, as the colours of a car.
- **Quantitative** variables can be characterised by numerical values, assuming discrete or continuous ranges as the age and weight of a person, respectively.

When defining a problem, it is important to define all the variables and identify which ones are predictors (X) and responses (Y) for a specific situation. Predictors, also called independent variables or features, are the ones that are available to be measure directly. On the other hand, dependent or target variables can be deduced from the features. This terminology is used in supervised learning, where there is available data from the two groups to train the algorithms.

In prediction, the features and targets are the inputs and outputs of the model, respectively. In inference contexts, it is common to get the correlations between predictors and responses, extracting valuable information from it (Hastie et al., 2009).

As an example, in the *Advertising* dataset presented in James et al. (2013) there are three features (TV, radio and newspaper) and a target variable (sales). It can be useful to make some inference, like what advertising medium generates more revenue for the company, or to build a prediction model to forecast the sales based on advertising spending in each channel.

Prediction problems are categorised accordingly to the output variable. When the target is categorical, it is a classification problem. Classification problems can be subdivided into the following groups, based on their cardinality:

- Single label classification, where only one out of N labels can be assigned to each event. e.g. The colour of a car is either “Black”, “White”, “Grey” or “Other”;
- Binary classification that is a special case of single-label classification, with N=2, and the output being “Yes” or “No”, e.g. The presence of a dog in an image is either “True” or “False”;
- Multi-label classification, where the same entry can have zero or more labels assigned to it. e.g. The same person can have any combination of diseases, “Hepatitis B”, “Tuberculosis”, “Diabetes” and “Cholesterol”, or none at all.

Quantitative target variables are associated with regression problems. In this case, the models will provide a numerical approximation for each sample, as the *Advertising* example presented before.

3 - Data Gathering

Sometimes data scientist don’t really need to be aware of the data acquisition process, since there are a lot of scenarios where this stage is already set up. However, even in these cases, it is good to know by what means the data is being collected to better understand what to do in the next steps. When there are no existent solutions

to acquire data, it is necessary to create them. This usually involves the development of pieces of software that translates real-world events to a computational representation of them. This representation is the collected data.

Data can come from one or more sources, as multiple sensors collected in the same software, or even distributed solutions, where different programs collect different types of data. Data is also gathered in multiple formats, as text, image or sound. All the collected information has to be saved to be useful in the future. The gathering software is responsible to store this information with persistence in a way that it can be accessed later. This is commonly accomplished through some data storage solution.

4 - Data Storage

To store the data there are different solutions that can be used depending on the project requirements. In this step, the major decision to make is to choose the one that best fits the problem specifications. The following list presents some guidelines to make the right choice (Heller, 2019).

- Schema - What variables and correspondent types are needed? How they interact with each other?
- Data size - How much data will be acquired at an atomic level, i.e. what space is needed for each event stored event?
- Writing frequency - How much events are being acquired for a determined time interval?
- Scale - What is the maximum predicted data amount to store?
- Mutability - Can the data schema change over time?
- Reading needs - How will the data be used in the future? What is the reading frequency?

One of the most simple solutions to store data is to use files in the computer hard drive. The most common formats to store information in files are Comma Separated Value (CSV), Extensible Markup Language (XML) and JavaScript Object Notation (JSON). CSV files can be translated directly to a tabular configuration, since the values are stored as a grid, with columns representing the variables and rows representing each event. The schema is fixed and defined by the number of columns.

XML and JSON files are much more versatile than CSV as they allow nested information and different schemes for each event. However they tend to require more disk space for the same information, since they need to keep metadata for every element (Nurseitov et al., 2009; Breje et al., 2018).

For more robust solutions, where writing and reading frequencies are higher and data volume scales in size, the use of Database Management Systems (DBMSs) can be the right choice. There are different paradigms for the Database (DB) implementation. The conventional DB architectures uses the relational paradigm, which divides the data in different tables with fixed structure, using associations to connect the different entities. The language used to query this DBs is Structured Query Language (SQL), that uses relational algebra to get the values. Other solutions, included in the group of NoSQL (Not Only SQL) DBs, are becoming more popular due to their performance and ability to scale better (Li and Manoharan, 2013). Some examples include Key-Value Store, Column Family and Document-Oriented DBs (Sakr and Gaber, 2014).

5 -Data Cleaning

Before extracting useful information from the raw data, it has to be prepared to better reflect the events and to be used for a specific knowledge extraction or modelling tool. Raw data can have multiple problems, as incomplete or empty fields and inconsistent or wrong information. To solve the first problem, empty fields can be removed or replaced with a coherent value, as the average of the same field in the rest of the dataset, for example. The same solution can be used when some field is invalid (e.g. “Age” presents a negative value).

Sometimes, data need to be aggregated at some level to be easier to use in the next steps. Aggregations always introduce some error to data and they have to be well planned to reflect the correct information. As an example, when storing the sales events for a group of stores, it can be useful to aggregate them, performing a sum to get the totals by city. However, the same aggregation can be done with an average instead, representing a different piece of information.

With the available data, it is possible occasionally to extract more information from it, enhancing the dataset with more details. For example, the “Date” attribute can be unfolded into other fields as “Day of Week” or “Month”. These fields are more useful for the future modelling step, since more data leads to better models. Furthermore, this representation is more likely to expose the data patterns (e.g. “There are more sales on weekends”).

When performing these transformations, it is important to keep data as unchanged as possible, introducing the minimum amount of distortion to it.

6 - Data Analysis

It is essential to understand and get relevant insights from data. This process is called Exploratory Data Analysis (EDA), and the aim here is to discover patterns and perceive what values are in the dataset before start modelling it.

To start EDA, it is common to get a visualisation of some data examples and statistics for each variable, as mean, standard deviation or minimum and maximum values. For qualitative variables it is useful to understand how data points are distributed across different classes.

Moreover, the use of visual analysis with graphic representations of the data can also be helpful and easier to interpret. These visualisations can be applied to the data itself, or to statistical data values.

Distribution plots, as histograms or kernel density estimates, are good tools to visualise how variables are dispersed. Line plots show data variations, for example in time-series data, along a time axis. Scatter plots can show data points dispersion related to some other variable.

Other common plots are correlations matrices, that shows the correlations between variables. High correlations can sometimes represent duplicated information in a dataset. Box-plots can show summary statistics in a visual way, presenting the quartiles, minimum and maximum values and the existence of outliers.

7 - Data Modelling

The data pipeline leads to the construction of one or more models that will transform the data into a valuable asset. The modelling stage involves the selection of the model architecture, the selection of data to train and test the model and the model evaluation as well.

Model selection is a decision that has everything to do with the problem definition. There are different models, for regression and classification, with multiple use cases. The use of simpler statistical models, as linear regression, or the choice for heavier ML models, for example Deep Neural Network (DNN), has to take into account the available resources and the cost-benefice trade-off.

The model building is an iterative process, where the data scientist must experiment diverse configurations or even various model architectures to reach the best possible solution. There are some guidelines to improve the models efficiency, but each case has different needs. The models versions should be saved, to allow that they are reproducible in the future. For each model it is important to keep performance metrics and their results for future comparisons.

8 - Problem Solution

After the modelling step, it is expected to get a solution for the initial problem. Among the built models, the one that performs better in a test scenario will be the best candidate to be used in the real-world case.

The model itself could be the answer, but it isn't true every time. The results of the model can be used in indirect ways to achieve the goal, being necessary some processing after the model output. For example, if the objective is to predict the number of cars in a parking lot, a regression model will output numbers in a real interval ($\mathbb{R}$). This information have to be treated before its use, since there aren't negative numbers neither decimal places in object counts ($\mathbb{N}_0$).

Additionally, the model results can be incorporated into some system that allows a easier utilisation for non-programmers. This is typically accomplished by using a user friendly Graphical User Interface (GUI), that will receive the features, use the data model and show the final result to the user.

2.3 MACHINE LEARNING

In 1950, Alan Turing presented the imitation game (TURING, 1950), proposing the *Turing Test* where two players, human and machine, answer the question of a third player, the interrogator. The machine is considered to have intelligent behaviour if the interrogator is not able to distinguish them. The idea of *learning machines* is also presented in the same article, suggesting the idea of a computer that is able to learn by itself how to perform some tasks rather than only follow instructions.

While there are many approaches to solve problems in the AI field, ML is the most popular one, powered by the increasing of computational power and data availability, that have enhanced the possibility to solve harder problems, using increasingly complex models.

This new programming paradigm has changed the way data and rules are used by the machine, since the computer writes its own decisions based on previous experiences. These systems are commonly black-boxes for the programmer on account of the internal complexity of the models (Chollet, 2017).

Statistical models that learn from data, such as linear regression or K-Neareast Neighbors (KNN) are part of the ML world. More recently a new field inside ML is becoming very popular due to the good results of its models, namely DNNs. This area is called Deep Learning (DL) and it is the main area of study of the present document (see Chapter 3 for more details).

2.3.1 *Definition*

ML is an AI sub-field where the ability to learn is given to the machine. In a traditional computer program, it is specified to the computer what it is expected to do when some input is presented. Depending on the input, the program returns a programmed output, or an error if the input is not specified in its internal rules. On the other hand, in ML systems, instead of programming the actions, they are learned using data of the specific problem that is intended to solve.

ML systems are capable of recognising patterns in input-output pairs or in some cases just in the inputs. This task is performed using statistical rules, returning at the end of the learning process a model that can be used to solve the problem (Chollet, 2017).

2.3.2 *Learning Paradigms*

The ability to learn can be given to machines by using different approaches. There are two major paradigms to categorise ML algorithms, depending on how the learning process is managed regarding data availability (Goodfellow et al., 2016):

- **Supervised Learning** - There is available information about the expected outputs. This data, called *labels* or *annotations*, are used to achieve the desired results. The distinction between dogs and cats pictures is an example, since there is a label for each image.
- **Unsupervised Learning** - Only the features are used. Some properties can be extracted from them, as data distributions that are useful to understand the problem itself. As an example, the clustering of data points with similar characteristics in different groups.

In addition to the aforementioned methods, there are also the *self-supervised* and *reinforcement* learning paradigms (Chollet, 2017). A self-supervised setting uses the inputs to generate the labels. In reinforcement learning, a reward system is used to teach the program the best decisions to make.

2.3.3 *Workflow*

When training ML models, a few tasks that apply to all projects. As with the DS workflow, shown in Section 2.2, the presented scheme is a personal understanding of the common steps presented in the literature. It is divided in two axis: the *data axis* with all operations that will transform or use data; and the *model axis*, comprising the steps of model creation. The intersection of both axis is the heyday of the process, in which using the model becomes useful.

The set of stages presented in Figure 2.3 are performed during the modelling of the DS workflow. From the schematic view presented in Figure 2.3, it is possible to build a continuous by plan to follow in a fluent way. The proposed ML workflow is presented in Figure 2.4.

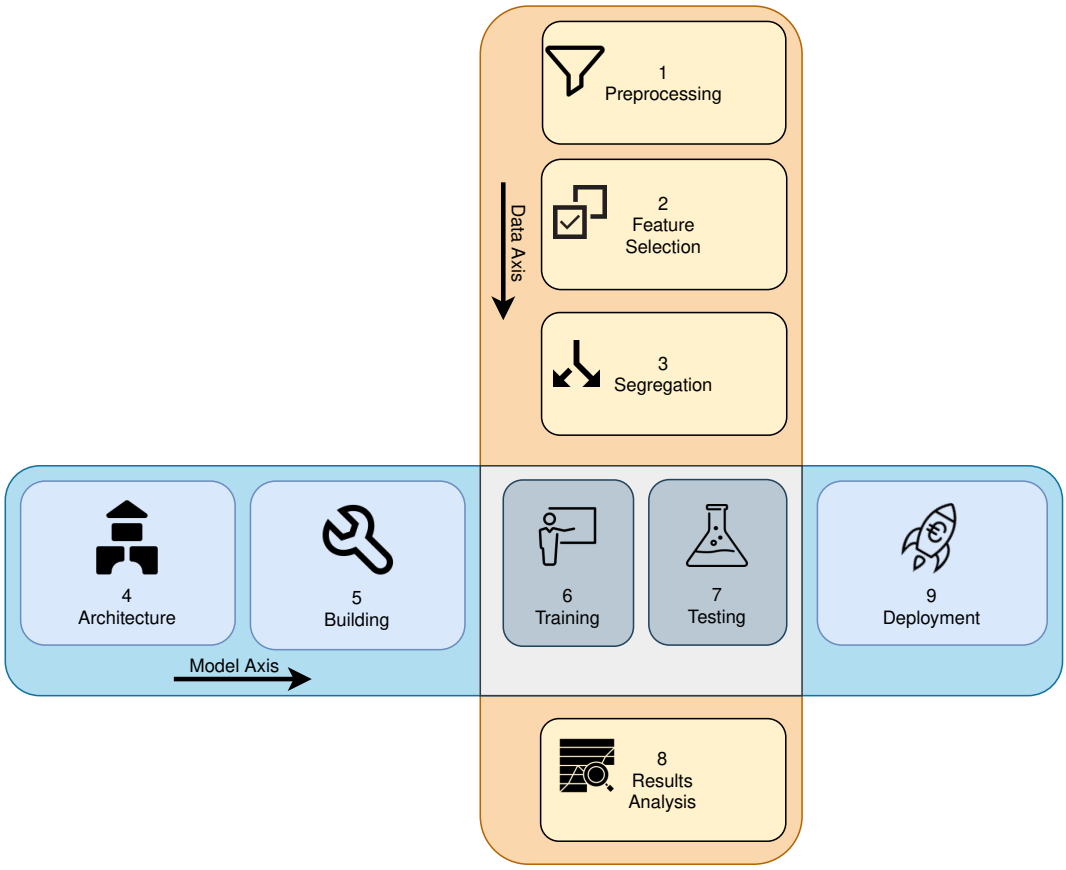


Figure 2.3: Proposed schematic view of the ML process.

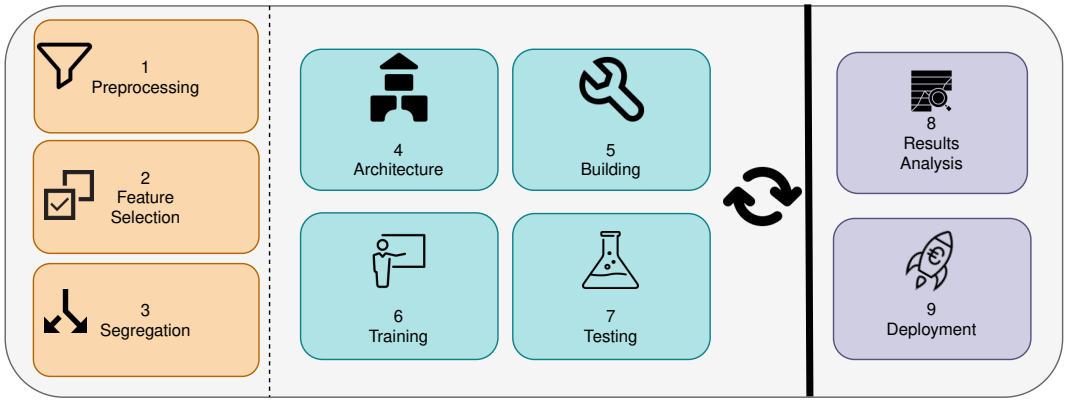


Figure 2.4: Proposed ML workflow.

1 - Data Preprocessing

Preprocessing includes all the operations that are needed to get the data ready to be consumed by a model. These actions may be mandatory, for example when dealing with missing values, or optional, for example when transforming data to improve the training performance.

One of the most common problems when preparing data is the existence of missing values in the dataset. Depending on the quantity and nature of such values, it is possible to use different strategies to solve the issue. Some common practises are drop the rows, if there are few missing entries; drop the column, when there are too

much lacking information for that feature, what makes it unusable; replace the values with some metric, as the average, median, maximum or minimum of the column.

Models doesn't often accept qualitative variables as inputs. It is necessary to encode them into some quantitative value. When the variable is qualitative ordered, representing some ordinal information, the most used method is label encoding. Here, for each class it is assigned a number, for example, "Low", "Medium", "High" are encoded as 1, 2, 3 respectively, maintaining the meaningful order of the classes. Oppositely, if variables are qualitative unordered, or categorical, they can be processed by an one-hot-encoding method, that transforms each class into a vector with 0's and 1's, where each vector position represents one class. For example, "Brown", "Blue", "Green" become [1,0,0], [0,1,0], [0,0,1], conferring equal influence for each class.

Other optional transformations are made to data in order to improve the models performance. Common conversions include scaling the variables to have the same intervals, for example to the $[0,1]$ domain, using the minimum and maximum values of the features, or modify data to follow a normal distribution (Garca et al., 2014).

2 - Feature Selection

To train a model successfully, it is important to select the characteristics that best define the problem, helping the algorithm to find patterns easily. This process can be made by hand or use techniques to calculate them.

For example, when modelling some data about a specific students class of primary school, features as the name or the student number can be dropped, since it will not bring useful statistical information for the model. The same way, the age is not a useful parameter if everyone in the class has nearly the same age. This is called the feature variance, and features can be selected using a variance threshold, where all the features with low variance will be discarded. Other methods for feature selection are Recursive Feature Elimination (REF) and Principal Component Analysis (PCA) (Chandrashekar and Sahin, 2014).

3 - Data Segregation

To get valid results when creating a new model, the data has to be segregated. One part of it is used to train the algorithm and the other to test its performance in data that is unknown to the model.

A model is under-fitting when it presents a low representational power and it cannot learn the patterns in the training data. When the model performs well on the training set but is inaccurate on the testing set, the model is over-fitting, what means that it has a poor capacity to generalise.

To solve these problems, it is common to modify the model parameters (more frequently called hyperparameters, since the *parameter* terminology is used for the variables learned during the training process) and train the model repeatedly. In this case, the segregation should return a third dataset, the validation dataset, that provide insights about the model during training used for the model tuning. Note that this is necessary because if the test data was used for that task, it could not be used to give an unbiased performance test, since the model was indirectly changed to fit that data.

4 - Architecture

The architecture is the first of the model axis and it is the beginning of its construction. Decisions about the skeleton of the model are taken at that moment, as the structure of the DNN.

While designing the model, the input and output layers have to be carefully defined, since they are the touch points with training and testing data. They need to receive the inputs and send the outputs in the same format that they are in the datasets. For example, if our data has three features and one target, then the network must has 3 and 1 neurons in the input and output layers, respectively.

The depth of the network, the type and the number of neurons in each layer are also decisions to be made at this stage. The activation functions to be used are other important architectural choice that can impact the future results of the model.

When this is finished, the model should be structurally prepared to receive data with the correct shape.

5 - Building

Before starting to train the model, a small is needed to prepare it to search for patterns in the presented data. This step, also called *compilation* by some frameworks, ensure that the network will try to converge accordingly to the defined criteria.

The selection of the loss function and the optimizer are the key decisions to make in order to get the model ready for the training. While the *architecture* builds the structure of the model, in the *building* phase, the network behaviour during the training process is defined. This specifies the way the network will learn.

6 - Training

The training dataset created in the segregation is now used to teach the model. One can use the validation data to track the training progress, more specifically the model performance and its capacity to generalize. Improvements can be made by repeating all the previous steps as many times as needed, modifying some of the earlier made decisions appropriately.

The training is achieved by passing the data through the model multiple times, minimising the loss function. The number of times this action is performed is another parameter that can be defined, the *epochs* count. Using the tracking metrics, it is visible sometimes that the model stops improving after a few training epochs. In those cases it is better to stop the training earlier, since the next iterations would be useless.

The outcome of this is a trained model, capable of using new data to return the best approximation for the desired outputs.

7 - Testing

The created model has to be tested before it can be used in a real-case scenario. For this purpose, the testing dataset is used as unknown data to the model. At this point, the network should be treated as a finished system, using the model to generate new data, the test predictions, that will be compared with the ground-truth.

The testing is the last interaction between data and the model during the development workflow.

8 - Results Analysis

Returning to the data axis, the outputs generated by the model are used to make comparisons with the expected values. The relation between them produce statistics that are used to evaluate the model, assigning a final score to it.

Depending on the problem definition and the selected performance metrics (see 2.4), it is advantageous to perform the analysis, adopting graphical tools to visualize the results.

The results analysis is the last checkpoint before deployment. If the results are not good enough, the whole process can be invalidated and start again. If the model passes this final experiment, it is ready to use in real problem-solving settings, where it is expected to perform similarly.

9 - Deployment

When the model is ready, it can be deployed, starting to generate value. The deployment is the where the model is integrated into other software that will use its outputs.

The deployment has to have in account the final use of the model, and in concordance with that, adapt the way it will be served. Depending on the problem needs, the model can be used to batch inference, where the results are precomputed and saved for future uses; or to online inference, where results need to be calculated in real time.

One popular example to deploy a model is to serve it creating an Application Programming Interface (API), which is the communication point to the model. Every software that needs to use the model results can use the entry-points to receive them. One of the benefits of this solution is that the internal behaviour can be modified without changing the way external entities contact with the model. For example, when more data is available, the training process can be improved and a new version of the model can be updated to serve without the need to adjust the remaining software.

2.4 PERFORMANCE METRICS

To know if an AI application is performing well, it is necessary to have one or more quantitative metrics, thus allowing more objective comparisons and improvement tracking. In every data science project, data modelling is one of the major steps. The built models intend to find patterns in data, either for inference or prediction. In inference models, it is important to find correlations that explain real phenomena. In prediction problems, more than understand how variables are related (interpretability), it is relevant to get accurate mappings between inputs and outputs, even without knowing the model's internal behaviour (explainability). For both cases, there are performance metrics to quantify the models performance, depending on the problem definition.

Focusing in prediction models, problems can be divided in two categories: regression and classification. In regression, the output data is a quantitative variable and the model should output a value that approximate the real observations. For classification, the model should choose the right class(es) for each presented case.

Two of the most common metrics for regression problems are the Mean Square Error (MSE) and the Mean Absolute Error (MAE). The MSE represents the average of the squared difference between real and predicted values. This metric is mathematically defined by Equation 1.

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

This metric cannot be so easily interpreted in the problems domain, once the error is shown in a different scale than the output variable. To solve this problem, the same metric can be rooted to translate the error back to meaningful scales. As presented in Equation 2, this operation originates the Root Mean Square Error (RMSE).

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

Other alternative is to use the absolute value of the errors instead of the square operation, as in the MAE metric in Equation 3, or convert it to a percentage using the Mean Absolute Percentage Error (MAPE) - Equation 4.

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

$$MAPE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4}$$

When dealing with images reconstruction processes, it is necessary to compare the original and the reconstruction images. One of the metrics to compare them is to use the MSE at a pixel level. However, other metrics as the Structural Similarity Index Measure (SSIM) presents a better performance in this task (Sara et al., 2019; Wang et al., 2004). The SSIM metric is defined in Equation 5.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5}$$

where μ_x and μ_y are the local means, σ_x and σ_y are the standard deviations and σ_{xy} is the cross-covariance for images x and y sequentially (Sara et al., 2019). C_1 and C_2 are stabilisation constants.

In classification problems, there are other ways to measure the model performance. The start point for evaluate any model in classification is to build the confusion matrix. The confusion matrix presents the relationships between the real classes and the predicted ones by the model. Table 2.1 is the general case for a binary classification problem.

Table 2.1: Confusion matrix example.

Real \ Prediction	True	False
True	True Positives	False Negatives
False	False Positives	True Negatives

There are four values in the matrix:

- **True Negatives (TN)** - Cases corrected classified as negative;
- **True Positives (TP)** - Cases correctly classified as positive;
- **False Positives (FP)** - Negative cases classified as positives, the equivalent of a type I error in statistics;
- **False Negatives (FN)** - Positive cases classified as negatives, the equivalent of a type II error in statistics;

Based on these values, there are different metrics to evaluate a model. The accuracy metric (6) measures the correct predictions on the entire dataset. It is often one of the most used metrics since a high accuracy represents few misclassified cases.

$$Accuracy = \frac{TP + TN}{Total} \tag{6}$$

The precision (Equation 7) is a major metric that presents, for all data points of a given class, how many of them are correctly classified. The recall (Equation 8) presents the percentage of correctly classified instances for all real data points of a specific class.

$$Precision = \frac{T_C}{T_C + F_C} \tag{7}$$

where C represents an arbitrary class, T_C the correctly predicted cases and F_C the incorrectly predicted cases for that class.

$$Recall = \frac{T_C}{T_C + F_{\neg C}} \tag{8}$$

where $F_{\neg C}$ represents the incorrectly predicted cases classified as not belonging to the C class.

Looking for the binary classification problem, the sensitivity and specificity metrics, presented in the Equations 9 and 10, are important metrics to understand how the model performs for each class. The sensitivity is defined as the percentage of real "Yes" cases that the model can predict. Oppositely, the specificity measures the same for the "No" class. For that reason the sensitivity and specificity are also called True Positive Rate (TPR) and True Negative Rate (TNR), respectively. Note that these metrics correspond to the recall for both classes of a binary problem.

$$Sensitivity(TPR) = \frac{TP}{TP + FN} \tag{9}$$

$$Specificity(TNR) = \frac{TN}{FP + TN} \tag{10}$$

The Receiver Operating Characteristic (ROC) curve is commonly used to get a summary of the performance of a binary classifier. It is the equivalent to use different thresholds to split the two classes and build multiple confusion matrices, calculating the TPR and False Positive Rate (FPR). The Area Under the Receiver Operating Characteristic (AUROC) applied to the graphic visualisation of this metric is used to get a overall accuracy of ML algorithms (Bradley, 1997). An Area Under the Curve (AUC) of 1.0 represents a perfect classifier, with TPR=1.0 and FPR=0.0.

To compare two distributions, g and f, the Kullback-Leibler Divergence can be used. Equation 11 presents the definition of this metric (Commenges, 2015).

$$D_{KL}(g\|f) = \int f(x) \log \frac{f(x)}{g(x)} dx \tag{11}$$

The relevance of each metric in the model evaluation depends on each specific case. For example, in some medical diagnosis, a model with higher recall and lower accuracy could be more valuable than the opposite, since identifying correctly a diseased person could be more important than inform a healthy person of an nonexistent disorder.

2.5 TOOLS

In AI projects there are a set of tools that can make the work a lot easier when dealing with data. Python™ is a versatile open-source programming language that is widely used for data-related operations. It is more readable and easier to understand than other languages, since the code syntax uses English keywords in the commands, avoiding also some punctuation, such as semicolons and curly braces.

Focusing on ML models train and test, there are two relevant tools that are used in this project. The first one, Scikit-Learn, is a Python module that presents a set of functions to help during the model evaluation. The second, WandB, is a web application to track and compare ML experiments.

Beyond the programming language, in this section are presented useful libraries and development tools to produce the desired results and in some cases improve the work productivity.

NumPy

NumPy stands for Numerical Python and it is a Python library that allows users to work with arrays (Harris et al., 2020). It is useful since arrays are often used in DS to represent all kinds of data. Considering that a big part of this library is compiled using C/C++, it processes arrays faster than the default operations performed with Python lists, using the memory locality to enhance the performance. It is also possible to use NumPy to manipulate matrices or higher dimensional arrays.

Pandas

When dealing with one or more data sources, it is important to have a structure that enables data manipulation in a easy way to extract information from it, empowering further visualisation and knowledge extraction. For this purposes, the *DataFrame* object from Pandas library is a flexible structure capable of read data from multiple formats, such as CSV, JSON or even SQL databases. The data is presented in a tabular format, organized to have labelled columns, commonly used to address each variable of the problem; and indexed rows, representing the data points (pandas development team, 2020).

The library is build on top of NumPy and it allows a variety of operations regarding the use of multiple data sets, such as the similar *joins* and *merges* performed in DBs. It is also easy to perform queries, obtaining different slices of data and subsets of columns. This tool is particularly important for the data cleaning and data analysis steps.

Matplotlib and Plotly

When performing EDA, it is important to use visual aids to better understand data. Furthermore, these abstractions are even more helpful when presenting conclusions to other people without technical know-how. Depending on the specific problem, different plots can be created to explain it. Among the most common are the line chart, bar chart, scatter plots and histograms. Matplotlib and Plotly are two Python libraries that take data as input and create the desired chart.

The Matplotlib library works with data represented in NumPy arrays. The *matplotlib.pyplot* submodule have a set of functions that change the figure to be presented, for example, adding a new plot to it or defining the label of each axis (Hunter, 2007). This library can also be used to visualize images.

An easy way to produce charts directly from a Pandas DataFrame is the use of the Plotly library, more specifically using the *plotly.express* submodule, that provides a high-level API to build the most common plots (Plotly Technologies Inc., 2015). Since it works with the DataFrame object, the column names can be used to specify the data to use in the chart. Additionally, the produced figures are interactive, which means that they can be manipulated, using for example the *zoom* and the *pan* tools, to move to different plot regions, getting information on the fly.

Jupyter

Jupyter is an open-source project that aims to improve the developers productivity through interactive computing. It born from the IPython Project and it includes various tools regarding user interfaces, such as the Jupyter Notebook, the JupyterLab or the Jupyter Console (Pérez and Granger, 2007). The Jupyter Notebook is a web-based application that permits to write Python code and Markdown annotations in the same place. The code can be divided in multiple cells and executed separately. This is very important in DS projects, specially for the modelling phase where various experiments can be tested with no need to rerun the full code (data loading, processing, etc.) over and over. Other advantage is that the resultant code output of each cell is directly attached to it, making debugging faster.

Combining the Jupyter Notebook with the Plotly library, a full interactive report can be built. This report can also be converted to HTML or PDF for offline visualisations. The Jupyter Notebook is a tool that is important for all the workflow stages regarding data operations (cleaning, analysis and modelling).

Scikit-Learn

Built on top of SciPy and NumPy libraries, Scikit-Learn presents a set of tools for predictive data analysis (Pedregosa et al., 2011). It presents a set of models either for regression, classification and clustering. Some examples include the traditional *Linear Regression*, Support Vector Machines (SVM) amd *K-Means*. Even if the objective is to use connectionist systems, it is useful to create simple ML models before training DNN, since they can serve as a baseline for more complex architectures.

The module also has some functions and objects that allows data preprocessing, such as the *StandardScaler* and *MinMaxScaler*, that convert the original data to a distribution with zero mean and unit variance or to the [0, 1] range, respectively. Other important feature of this library is the *train_test_split* function that provides a quick way to split data for the training and testing steps.

WandB

The ML workflow is an iterative process where the results are accomplished using the *trial and error* approach. Since there is no certainty that the next try will be the best, it is imperative to track the experiments making possible a future comparison between the different architectures.

The WandB web application can easily track all the experiments, the results and save the best models as well (Biewald, 2020). It has a Python library that will automatically log the desired information every time a new model is trained, with almost no changes to the code. To compare the results, an intuitive dashboard is presented, showing the logged metrics in a table where filtering is possible (for example, filter models with an precision higher than a given threshold). Also the plots are overlapped to give a better perspective on the differences for each model. When looking for each specific model it is possible to analyse the training steps evolution regarding the loss function (see chapter 3), validation metrics, and even system information, such as Graphics Processing Unit (GPU), Central Processing Unit (CPU) and memory usage and storage access.

3

CONNECTIONIST SYSTEMS

> *The real question is not whether machines think but whether men do. The mystery which surrounds a thinking machine already surrounds a thinking man.*
>
> B.F. Skinner

3.1 INTRODUCTION

In cognitive science, connectionism is a movement that aims to explain and model the functionality of the human brain. To represent the human nervous system are used Artificial Neural Networks (ANNs), expecting a similar result of the biological ones. As Barrow presents in the *Artificial Intelligence* book (1996), this is a bottom-up "mechanist" solution, based on the assumption that if one uses the same structure and components of an existing system, the created copy must have the same behaviour of the original one (Barrow, 1996).

Although there are lots of information about the human nervous system anatomy, it stills difficult to understand the exact way it works. The human brain, one of the most important components of the central nervous system, is capable of memorising and process the information it receives, learning over time how to get better in its functions.

The cells in the nervous systems are the neurons. The three basic components of a neuron are the cell body or soma, the dendrites and the axon. The dendrites are a branched structure near the soma that receive signals from other neurons. Electric impulses are conducted away from the soma by a thin and long channel, the axon. This message passing process is called a synapse. Synapses occur between an axon and one or more neurons (Standring, 2015).

Based on the different dendrites arrangements, there are distinct neuron morphologies, each of them with a specific function (Kulkarni and Firestein, 2012; Bota and Swanson, 2007). There is excessive production of dendrites in early development and, with the synaptic traffic variances, these structures can expand or contract (Wong and Ghosh, 2002).

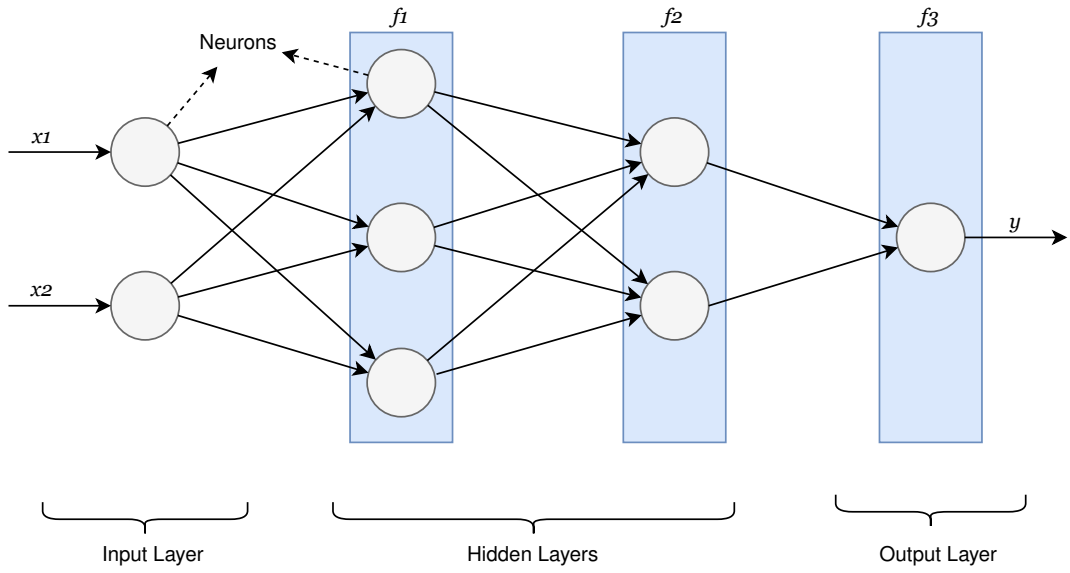


Figure 3.1: Example of a ANN as a composition of functions. In this case, $n = 3$, $x = [x1, x2]$ and $\hat{y} = [y]$.The θ parameters in each layer are represented by the arrows before the functions (blue boxes).

3.2 DEFINITION

An ANN can be defined as presented in Equation 12, assuming f^* as the real relationship between x and y. The function f is defined with the choice of the ANN architecture. The learning process will determine the values of θ (Goodfellow et al., 2016).

$$\hat{y} = f(x, \theta) \tag{12}$$

where x represents the input data, $\hat{y}$ the predicted output data, f the approximate function of f^* and θ the learning parameters.

Assuming a deep feed-forward architecture, each layer represents a function that is applied consecutively to the input. In that sense, one can look at a DNN as an extremely complex composite function (Goodfellow et al., 2016).

From the representation on Figure 3.1, one can rewrite the ANN definition as shown in Equation 13, assuming that the f function in Equation 12 is the composition of n functions, where n represents the number of computing layers in the network (Goodfellow et al., 2016; Chollet, 2017).

$$\hat{y} = f_n...(f_2(f_1(x, \theta_1), \theta_2), ...\theta_n) \tag{13}$$

where f_i and θ_i represent the approximate function and the learning parameters of layer i, respectively.

3.3 HISTORY

The first attempt to reproduce an ANN was made in 1943 by Warren S. McCulloch and Walter Pitts (McCulloch and Pitts, 1943). In their work, they tried to explain and simulate the behaviour of the neurons with electrical

circuits. Six years later, Donald Hebb presented a learning model which pointed out that the strength of the connections between neurons is enhanced each time a synapse occurs (Hebb, 2005). Hebb proposed that memory becomes permanent with this activity and the information is represented in the changing neural network structure.

In 1958, based on the work of Hebb, Frank Rosenblatt developed the *Mark I Perceptron* (Rosenblatt, 1958). For each R-Unit is performed a weighted sum of the A-Units outputs. The most important feature of this network is the capability to learn over time, adapting these weights in order to get the desired output.

With the R-units introduced by Rosenblatt, Bernard Widrow and Marcian Hoff improved the weight update algorithm by using the Least-Mean-Squares learning rule (Widrow and Hoff, 1960). This process is more efficient since the weights update is based on the continuous values before the binary output is produced. The system created by Widrow and Hoff in 1960 was called ADALINE (ADAptive LInear Element). This was the first system to be used in a real-world problem.

Marvin Minsky and Seymour Papert wrote in 1969 theirs *Perceptrons* book. In that book, they expose several single-layer perceptrons limitations. Perceptrons were not able to establish non-linear input-output correspondences, like the XOR problem. This problem could be solved with a multi-layer network, but at this time there wasn't an effective learning algorithm for these kinds of networks.

The back-propagation algorithm was reintroduced by Rumelhart, Hinton, and Williams in 1986, solving the problem of the weights update in multi-layer networks (Rumelhart et al., 1986). The algorithm calculates the difference between the network's output value and the real value and propagates the error backwards. For each layer, it uses gradient descent to adapt the network weights. The back-propagation algorithm is not related to biological neural networks since there is no evidence in the human brain of such a mechanism. Although, this is a good way to optimise ANNs and solve tasks that can't be described by mathematical rules. These tasks include objects or faces recognition in images, speech understanding or classification problems.

3.4 ARTIFICIAL NEURAL NETWORKS

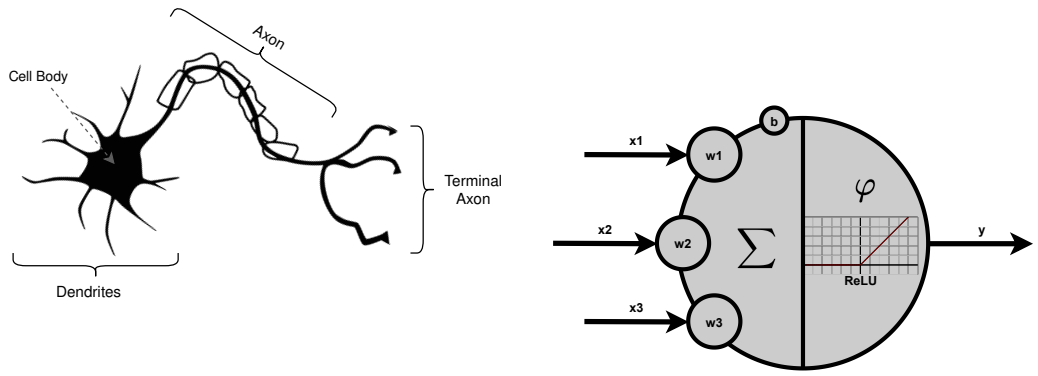
Nowadays, rather than models to understand the human brain, ANNs have become tools for an infinity of tasks, due to the big potential they demonstrate. With the computational power increasing, the data availability and the improvement of the algorithms, is now possible to create, train and use more complex models (Chollet, 2017). The models with various hierarchical layers are called DNN due to the successive representations of the data inside the network (Chollet, 2017). There are many DNN types, each of them with specific use cases.

The most popular DNNs are the deep feedforward networks or Multi-Layer Perceptrons (MLPs). These models are called feedforward because the information flows from the input to the output without cycles (Goodfellow et al., 2016). The intermediate layers, between input and output, are the *hidden* layers because the training data has no information about the expected output for each of them. The number of hidden layers and the number of nodes in each layer are parameters that have to be chosen on the ANN creation. MLPs has typically fully connected layers, where each node of a layer is connected to all nodes of the next one. As a function approximator (Cybenko, 1989), the MLP architecture is mainly used for simple tasks, such as regression and classification problems. Its use in more complex problems is now outperformed by other solutions.

3.5 ARTIFICIAL NEURONS

The artificial neuron structure is an attempt to reproduce the human nerve cell as presented in Figure 3.2a. The neuron receives information, processes it, and then forwards it to adjacent nodes. The process of transmitting information is called a synapse. In the artificial neuron, the information is processed in two stages. The first stage is a weighted sum of all the input data and the second is performed by an activation function, that decides if the neuron is activated or not, i.e. if the information will pass to adjacent neurons (Basheer and Hajmeer, 2000).

The activation function (φ) maps the weighted sum from the $]-\infty,+\infty[$ interval to the desired domain. One of the most popular activation functions is the Rectified Linear Unit (ReLU), which has an output domain of $[0,+\infty[$, activating the neuron only when the input is greater than 0 (see Section 3.6 for more details about activation functions). In the learning process, at the neuron level, the learning parameters are the weights from the input connections and the bias, defined as b. The bias is an additional parameter to improve the flexibility of neural networks.



(a) Natural Neuron structure. It receives information through the dendrites, processes it in the cell body and sends it to adjacent neurons through the axon.

(b) Artificial neuron example. The artificial neuron receives information from other neurons (x_i), processes it ($\sum$ and Φ) and sends the result (y) to other nodes.

Figure 3.2: Comparison between natural and artificial neurons.

Equation 14 provides a mathematical definition for the operations inside each artificial neuron.

$$output = \varphi(\sum_{i=0}^{n} w_i x_i + b) \tag{14}$$

where x_i and w_i are the input data from the neuron i of the previous layer and its associated weight, b is the bias and $output$ represents the processed information, the output of the neuron.

3.6 ACTIVATION FUNCTIONS

The activation functions are important to confer non-linearity to the network. If a linear function (Equation 15) were always used as activation, the ANN would perform as a simple linear regression model, not benefiting from stacking multiple layers (Chollet, 2017; Goodfellow et al., 2016).

$$f(x) = kx \tag{15}$$

where k is a constant.

In order to be possible to use the back-propagation algorithm in the learning process (see Section 3.7), the function must be differentiable. The derivative of the function is used to calculate the gradients direction to reach the minimum of the loss function using the gradient descent method (CAUCHY, 1847). One of the most common functions used in the intermediate layers of a network is the ReLU function, presented in Equation 16.

$$ReLu(x) = max(0, x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{16}$$

The ReLU is represented in Figure 3.3 with the respective derivative. There are some variants of this implementation as the *leaky ReLU* or the *parametric ReLU* (Maas et al., 2013; He et al., 2015b).

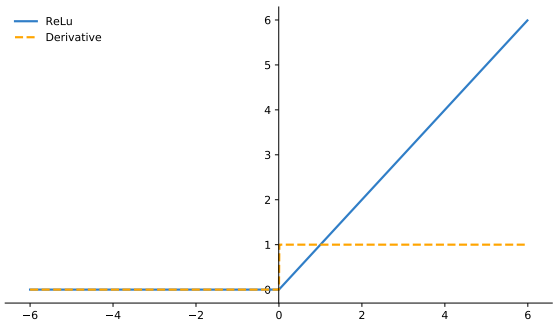


Figure 3.3: ReLU function and its derivative.

Additionally, the activation is important in the last layer of a network to determine the output interval, that will coincide with the activation function codomain. There are different activation functions, that could be used depending on the problem nature. For example, the *sigmoid* function (Equation 17) is used in binary classification problems where the output in the $]0,1[$ interval can be divided to split the two classes.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{17}$$

The sigmoid function is also useful for multi-label classification problems, where each of the neurons in the last layer represents a class and the output of the neuron is the probability of a case to belong to that class. Figure 3.4 shows the sigmoid function.

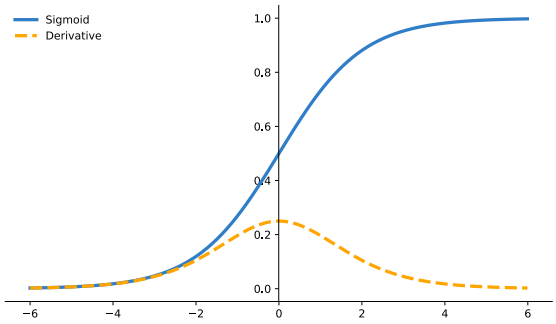


Figure 3.4: Sigmoid function and its derivative.

Other activation functions as the *hyperbolic tangent* are used for the same purposes. The *softmax* function is useful to represent probabilities distributions across multiple classes, that is, single label classification problems. In regression problems, a linear function can be used in the last layer to maintain the full output domain.

3.7 LEARNING PROCESS

The learning process is a loop, starting with all the weights set with random values, being adjusted in every iteration. Each iteration has two stages: the forward propagation, where the input data (x) is used to predict the output value ($\hat{y}$); and the weights update, using the back-propagation algorithm to calculate the error gradient for each layer, propagating it from the output to the other layers (Rumelhart et al., 1986). This process is determined by two main elements: the loss function and the optimizer.

The loss function defines a measurement of the network success in its task during the learning iterations. It uses the predicted results ($\hat{y}$) by the network in each step and the real values (y) to calculate the error.

The main goal is to minimize the loss function. Using the error gradient, the ANN must converge to its minimum. The optimizer defines the way that the weights in the network are updated based on the gradient calculated with the back-propagation algorithm. The optimizers implement some variant of Stochastic Gradient Descent (SGD) (Chollet, 2017; Goodfellow et al., 2016).

3.8 LOSS FUNCTIONS

A loss function, also called *cost function*, is the metric that the learning process uses to improve the network over iterations. It compares the expected and predicted outputs, returning a value that is expected to decrease to the minimum.

In regression, it is common to use the MSE and MAE metrics (presented in Section 2.4) as the loss function. Negative log-likelihood losses are used in classification problems. For example, the binary cross entropy loss, defined in Equation 18, allows to train a network for binary classification.

$$BinaryCE(\hat{y}, y) = -y \cdot log(\hat{y}) - (1 - y) \cdot log(1 - \hat{y}) \tag{18}$$

Note that one part of the equation will turn into 0 since one of the real labels (y) is either 0 or 1, cancelling out the first and second multiplications, respectively.

3.9 APPLICATION AREAS

ANNs have multiple application areas, specially in the DS field. They are ML models used to solve classification and regression problems with better results than classical statistical models.

In the Computer Vision (CV) field, the most widely used technique to deal with image data is the training of Convolutional Neural Networks (CNNs). This type of DNN has convolutional layers that process the information differently from fully connected layers. These layers can capture the pixels patterns of the input image. CNNs are more deeply presented in Chapter 4.2. Common uses to CNNs are image classification, object detection or facial recognition.

Other type of ANN is the Recurrent Neural Network (RNN) architecture introduced in 1986 (Rumelhart et al., 1986). These networks perform well with sequential data, like texts or time-series. Unlike feedforward networks, RNNs can have cyclic connections. The Long Short-Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) is a RNN type that has solved problems related with vanishing and exploding gradients. LSTMs can be used to speech and handwriting recognition.

Autoencoders (AEs) are neural networks capable of copy the input to the output, with a data bottleneck in its structure. AEs are used to information compression, Anomaly Detection (AD) and generative modelling (Goodfellow et al., 2016).

3.10 TOOLS

The building of solutions involving connectionist systems are not intended to rewrite the full logic implemented in ANN. Instead, there are frameworks that enable the programmer to use abstractions and focus his efforts in the decisions to make in order to improve the final results. In this section are presented two of those frameworks: . In this work, the Keras API is used with TensorFlow back-end.

TensorFlow

TensorFlow is an interface that provides the capability to define ML algorithms, allowing its further execution (Abadi et al., 2015). The supported languages are Python and C++ and the computations can be expressed through computational graphs.

The library is open-source and it is flexible when creating neural networks topologies. The models can be trained either in CPUs or GPUs, adapting the load to the specific machine where the experiments are being performed. The GPU support requires an NVIDEA®graphic card and the installation of software such as the CUDA toolkit and the cuDNN SDK.

Additionally, a visualization tool is also made available when working with TensorFlow models. The model training process can be logged through TensorBoard. It enables to track the training and validation metrics, such

as the loss and the accuracy and to see the model graph. It can be combined with WandB tool, that will use the same TensorBoard logs to group the metrics in the respective experimental run.

Keras

Keras is a high-level framework designed to become the construction of deep models easier (Chollet et al., 2015). The library is coded in Python and it is publicly available through an open-source licence. The Keras functionalities can also be used through the TensorFlow library, using the *tf.keras* module. It is also possible to use other back-end frameworks instead of TensorFlow, such as Theano or CNTK.

The API is user-friendly and modular, which ensures that the programmer can use multiple stacked "blocks", creating the desired DNN. When using Keras, the abstractions are on the layer level. It means that each "block" represents a layer, that can be parameterized to denote the desired architecture.

The Keras framework provides also a diversified range of tools for data preprocessing, specially for image, text and time-series data. Although the default building blocks are sufficient in the majority of the scenarios, the metrics, loss functions and even the layers themselves can be specifically tuned. For example, it is possible to build a custom layer that applies a specific function to the previous layer output (*Lambda* layer).

4

IMAGE ANOMALY DETECTION

> *The uneducated person perceives only the individual phenomenon, the partly educated person the rule, and the educated person the exception.*
>
> Franz Grillparzer

4.1 INTRODUCTION

When developing AI systems, it is common to specialize them for a specific task, depending on the problem to solve. If the problem involves visual data, represented in the form of images and videos, it is framed in the CV context. In this situations, the machine is intended to be equipped with the capability to perceive the world and describe it based on image properties (Szeliski, 2010).

AD is a task that involves the identification of unusual or unwanted situations in data. These situations are commonly referred as anomalies, outliers or exceptions. In a dataset, anomalies commonly represent a small percentage of the data. There are some sub-fields in AD (Alla, 2019): Outlier Detection aims to distinguish between the *normal* and *anomalous* cases; Novelty Detection is very similar to Outlier Detection, however the main objective is to detect unobserved data instead of rare observations (Markou and Singh, 2003a,b); Noise Removal is the area that aims to reconstruct the initial dataset removing unwanted data points (noise), for example in images or sound. In the present work, the AD term is mainly used to refer Outlier Detection.

The following sections show these two areas that are involved in detect anomalous situations in image sets. The first is focused in process and synthesize information from images, while the second provides an overview of AD techniques. In the last section are presented tools related with the study areas.

4.2 COMPUTER VISION

Imagery problems are challenging since, in most cases, an image is a representation of 3-dimensional objects in a 2D interface, where there is unknown information inevitably. Therefore, the CV algorithms have to be able to perceive the same object in different perspectives.

Animals in general perform the “seeing” task effortlessly, while computers need extremely complex computations in this perceptual function. Additionally, the solutions are specific for each use-case, considering that there is not a solution extensive enough to approximate the capabilities of biological vision systems (Szeliski, 2010).

4.2.1 *Definition*

CV is a field that aims to simulate human vision in machines, trying to give a meaning to input images. While it is easy for the human eye to perceive and distinguish different components in a scene, this task is challenging for computers, since they need to translate the pixels information into something useful in each different context (Prince, 2012).

To translate the visual data into a real world representation, it is necessary to build a model, that takes images as input and returns the desired information correctly mapped.

In the recent years, the input image is processed inside an ANN in order to get the relevant information from it. Currently, the best deep learning algorithms to perform CV tasks uses CNN architectures.

4.2.2 *Image Preprocessing*

As referred in Section 2.3.3 of Chapter 2.3, the data preprocessing step is important to get it ready for the modelling phase and to improve the model performance. The preprocessing in computer vision is equivalent, however the transformations are applied to the image pixels.

The objective in this process is to help the network to identify the images patterns easily, removing some of the unwanted variation that can be present in a image set. For example, in object detection, the same object can be exposed to different light conditions. By preprocessing the images, it is expected to emphasize the principal components that will help the model to distinguish the object, homogenizing the input images (Prince, 2012).

At the image level, size transformations can be performed to match the model architecture. At the pixel level, there are a variety of techniques to enhance the features in image. These operations are made using neighborhood filtering (convolutions), that will transform the pixel values using a filter applied to the original image (Szeliski, 2010). Figure 4.1 shows an example from a convolution operation.

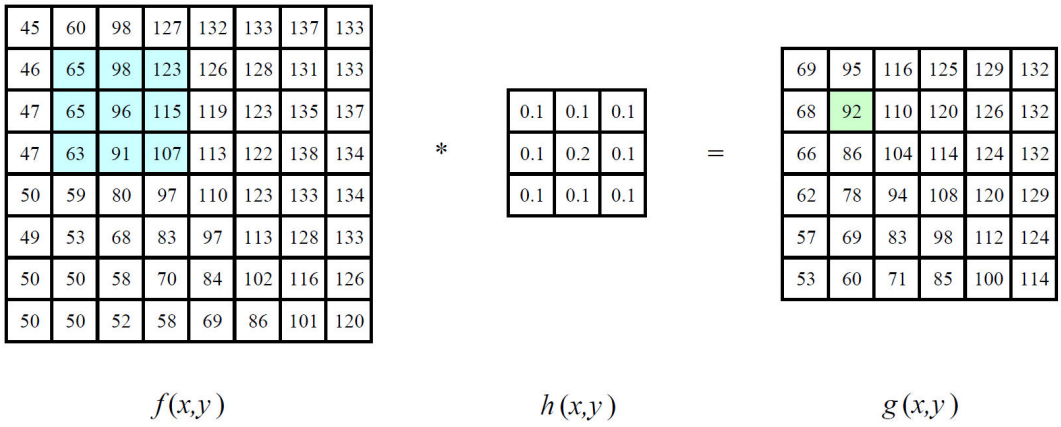


Figure 4.1: Image filtering operation example. "The image on the left is convolved with the filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel."(adapted from Szeliski (2010)).

In the following sections are presented some useful filters that are commonly used in image preprocessing. The majority of the transformations are presented in detail by Richard Szeliski in the book "Computer Vision: Algorithms and Applications" (Szeliski, 2010).

Smoothing

Smoothing is used to blur the image, removing noise from it and making the pixel values more related to its adjacent ones. The most used filters for blurring are the *Average* and *Median* filters, the *Gaussian* filter, and the *Bilateral* filter.

As the names suggest, the *Average* and *Median* filter use the respective metrics in the adjacent pixels set to compute the output pixel value. Figure 4.2 presents an example of applying one of these methods to an image (median blur).

(a) Original image. (b) Processed image.

Figure 4.2: Example of image smoothing using the median filter.

The *Gaussian* smoothing method uses a Gaussian distribution to perform a weighted computation of the output pixel, where the nearer pixels have more importance than more remote values, since the weights decrease with distance from the central location (Stockman and Shapiro, 2001).

The bilateral filtering is a non-linear filter that preserves the edges in the input image. It removes noise from the image, homogenizing the pixel values, performing a blurred effect maintaining the edges perceptible as presented in Figure 4.3b.

(a) Original image. (b) Processed image.

Figure 4.3: Example of image smoothing using the bilateral filter.

Thresholding

A thresholding operation modifies the pixels values, splitting the image at a certain threshold. Equation 19 shows that operation, where the image is polarized to have only two values, dividing all the tones in a binary output (light and dark).

$$f(p,t) = \begin{cases} 1, & \text{if } p \geq t \\ 0, & \text{if } p < 0 \end{cases} \tag{19}$$

where p represent each pixel value and t the threshold.

There are some variants of this operation, where the inverse operation is performed, reverting the light and dark values. The option to preserve some pixel values is also a possibility, truncating the rest of the pixels, as presented in Equation 20.

$$f(p,t) = \begin{cases} 1, & \text{if } p \geq t \\ p, & \text{if } p < 0 \end{cases} \tag{20}$$

Other methods apply adaptive thresholds to make changes in different image zones where illumination conditions can vary. Examples of the application of these variants are presented in Figure 4.4

(a) Original image.

(b) Processed image using Equation 19.

(c) Processed image using Equation 20.

Figure 4.4: Example of thresholding operations.

Edge Detection

Edges are important features when dealing with images. Edges are the boundaries between objects in a 2D representation. The Canny filter is one of the most popular edge detectors (Canny, 1986).

In addition to other steps, the convolution masks used to perform the edge detection are matrices such as the presented in 21 and 22, for the x and y directions, respectively (OpenCV Documentation, 2008).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{21}$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{22}$$

Figure 4.5 presents the transformations made by the canny filter.

(a) Original image. (b) Processed image.

Figure 4.5: Example of edge detection using the canny filter.

4.2.3 *Convolutional Neural Networks*

In 1989 Yann Lecun introduced CNNs by applying his approach to handwritten zip code recognition (LeCun et al., 1989). A CNN is a DNN that uses convolutions instead of matrix multiplications in one or more layers (Goodfellow et al., 2016). The CNN architecture have some specific characteristics. The most obvious is the presence of convolution operations inside the network. In CNNs, a convolution is an operation that transforms an input image in a feature map, by applying a kernel to it. The kernel has a fixed size smaller than the input image and is applied to the pixels like a sliding window. Similarly to the presented convolutions in the previous section, Figure 4.6 explains how a convolution operation is performed in detail inside a CNNs.

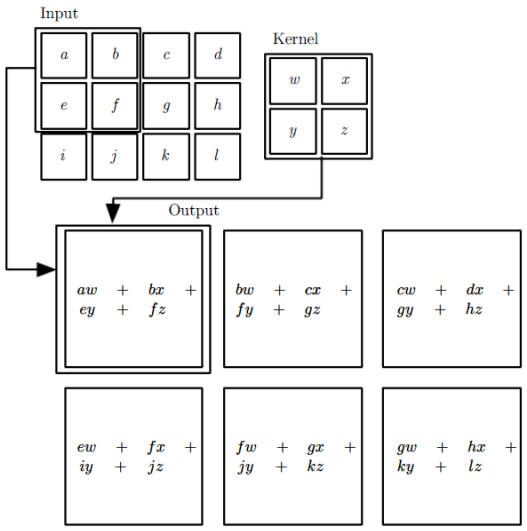
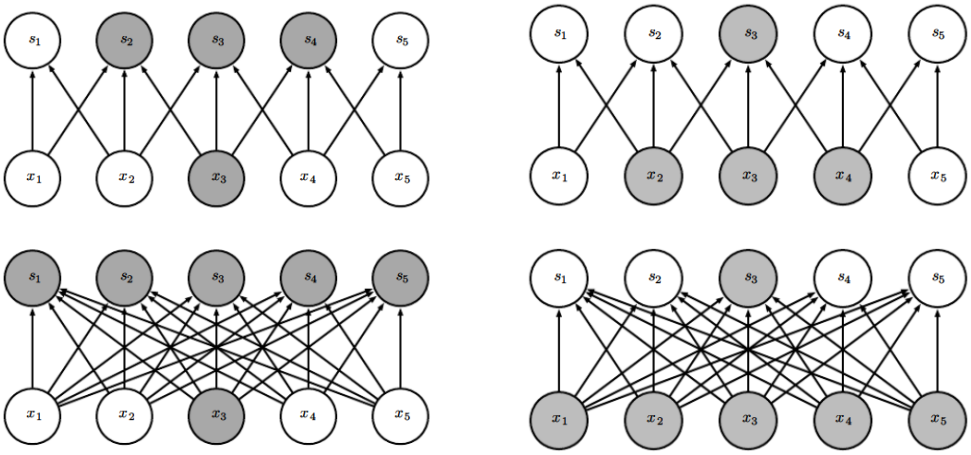


Figure 4.6: Convolution operation in CNNs(adapted from Goodfellow et al. (2016)).

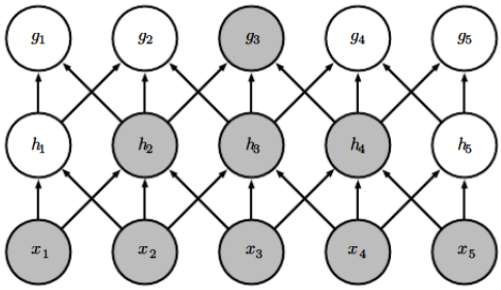
A convolutional network implements sparse connectivity and parameter sharing. In fully-connected networks, all nodes of one layer are connected to all nodes of the next one with different weights for each connection. The use of filters (kernels) in CNNs makes the connections between layers sparse, which means that the neurons in one layer only communicate with a limited number of nodes in the next one (Figure 4.7a). Consequently, each receptor neuron will only be fired by a restricted set of input nodes (Figure 4.7b). With that in mind, it is expected that the network learns the patterns in the input pixels by stacking up multiple convolutional layers. In Figure 4.7c, perhaps the *g3* node receptive field corresponds to only 3 neurons (*h2*, *h3* and *h4*), it is indirectly connected to 5 input nodes (*x1* to *x5*). It means that the deeper the layer, the more global will be the information presented in its nodes about the image. While the first layers process information about some parts of the image, like edges and other patterns, the deeper layers represent a compressed view of the image as a whole.

The parameter sharing property is acquired by having the same weight value in different node connections as shown in Figure 4.8. This is equivalent to say that the same kernel is used in each step of the sliding window, reducing the necessary memory to store these values. The parameter sharing property causes the convolutional layer to preserve the order of the input image in the output feature map. This is called equivariance to translation, because any image displacement will be reflected in the output.



(a) A neuron in convolutional layers (top) only activates a restricted number of neurons in the next one. In fully-connected layers all neurons are activated (bottom).

(b) The receptive field of a neuron is much smaller in convolutional layers (top). Each neuron is influenced only by a portion of the previous layers neurons.



(c) Deeper layers represent more generic information about the input image.

Figure 4.7: Comparison between convolutional and fully-connected layers focusing the different aspects of sparse connectivity (adapted from Goodfellow et al. (2016))).

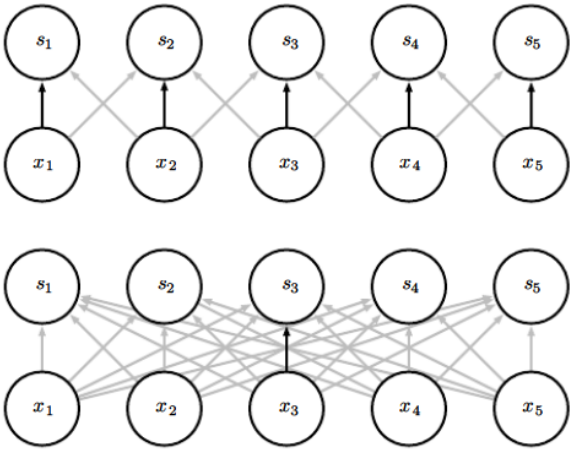


Figure 4.8: Parameter sharing in convolutional layers (adapted from Goodfellow et al. (2016))).

In every convolution layer is common to have a detector stage at the end of the layer, where an activation function, like ReLU introduces non-linearity to the output.

Other very used layer type in CNNs is the pooling layer and it is used after the convolutional layers. Pooling functions take the feature maps produced by the previous layer and create a summary of the features contained in each location. Like convolutional layers, it uses a sliding window to go through the image. There are different pooling functions, but the more commons are max-pooling, average-pooling and sum-pooling (Boureau et al., 2010). The pooling layers provide to CNNs the ability to be invariant to transformations of the input.

In the convolutional and pooling layers, one can decide the size of the sliding window (kernel size and pooling size, respectively) and the size of each step (strides). Depending on the kernel size and the selected strides, convolutional layers decrease the image size. To maintain the original size, padding can be used, filling the remaining rows and columns with zeros. In pooling layers, padding is not so common since a downsample is desirable in this case.

The most basic structure of a CNN for image classification is a sequence of interleaved convolutional, ReLU and pooling layers with a set of fully-connected layers on top. Figure 4.9 shows an example of this architecture type.

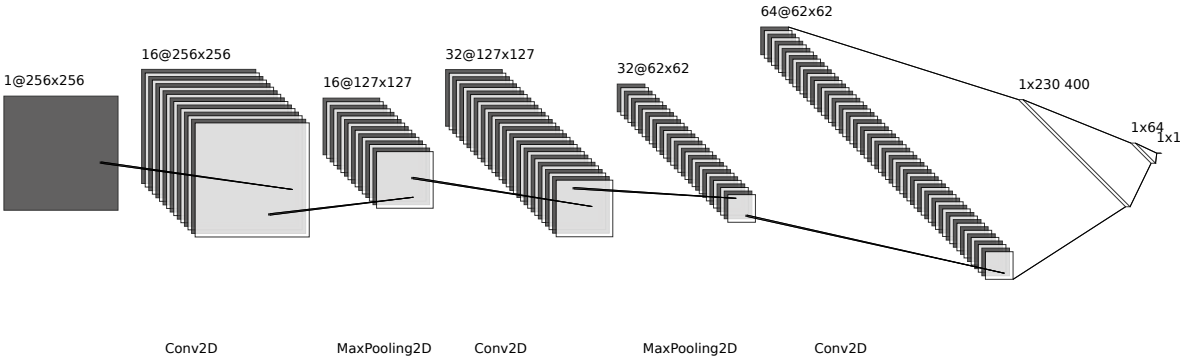


Figure 4.9: CNN architecture example.

Even though each specific task has a specific domain of solutions, an effective way to get good results with CNNs is to use parts of a pre-trained network. This is called transfer learning and can be more efficient than to train a full network from scratch. The ImageNet dataset is one of the biggest image sets with more than 14 million images. Some networks are trained with this dataset and they are available on frameworks, like Keras and Tensorflow, with the best weights configuration (Chollet, 2017).

4.2.4 *Related Work*

Some of the most important CNN architectures derived from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition. The AlexNet (Krizhevsky et al., 2012) was the State-of-the-Art (SOTA) in 2012 with a top-1 accuracy of 63.3%. The AlexNet marks the beginning of a new interest in CNN research. One year later, Matthew D Zeiler and Rob Fergus slightly improved this value to 64.0% with the ZFNet (Zeiler and Fergus, 2013). The ZFNet architecture is similar to the AlexNet. VGG (Simonyan and Zisserman, 2014) in 2014 get an accuracy of 74.5%, reducing the filter sizes and increasing the number of layers in the network (Aggarwal, 2018). In 2015, the second version of Google's Inception network (or GoogLeNet) get a 78.8% top-1 accuracy (Szegedy et al., 2015). The inception modules have different filter sizes and, since the weights are learnable, the network

can decide which one has more importance. This is useful to capture more than one level of granularity in the network (Aggarwal, 2018). In 2015 the Microsoft ResNet achieved 78.6% top-1 accuracy (He et al., 2015a). This accuracy is lower than the Inception V3 78.8%, but it was the winner of the 2015 ILSVRC competition. This architecture used 152 layers, approximately 7 times more layers than the previous work. ResNet uses connections between not adjacent layers in order to create shortcuts that permits the depth of the network to be increased without compromising the optimisation process (He et al., 2015a). The current best top-1 accuracy in ImageNet dataset is the Noisy Student network (88.4%) (Xie et al., 2019), updated in January 2020. In this work a self-training framework with two networks is used: a teacher and a student. The teacher model is trained in labelled images and create pseudo-labels for new unlabelled images. The student network uses both labelled and pseudo-labelled images in the training process and it is used as the teacher for the next iteration (Xie et al., 2019).

4.3 ANOMALY DETECTION

There are three methods that can be applied to identify anomalies. The first one is to train models in *normal* data, where the anomalies are detected by a difference in what the model is used to "see". Another method is to use labelled data points, and present that information to a model, that will split the two different distributions. The third method is to use probability distributions and detect low-probability instances, that correspond to anomalous situations (Alla, 2019).

The AD problem has a very important role in several domains. For example, the credit card fraud detection is one of the most popular problems, where the main objective is to detect fraudulent transactions to warn the respective owner (Aleskerov et al., 1997).

4.3.1 *Definition*

Anomaly detection is the process in which an algorithm is used to identify some patterns in data that indicate an anomalous situation. Anomalies can represent low-probability events, something that is not expected to occur or unwanted situations (Alla, 2019).

There are three types of anomalies (Chandola et al., 2009):

- Point Anomalies - There are isolated data entries that are anomalous to the rest of the dataset. These points are called outliers. An example is the amount spent in credit cards. If a transaction presents a very high value compared to the rest of the transaction values, it represents a point anomaly;
- Contextual Anomalies - The entry could be considered normal in other parts of the dataset, but is a strange data point in the context it appears. This is more common in data-series cases, for example the same temperature value can be normal in January and abnormal in July;
- Group or Collective Anomalies - There are groups of data entries that together create an anomaly. As example we can take the requests to an web server. If a lot of request are done in a short period of time, it can represent an anomalous behaviour, and a possible cyber-attack (e.g. DDOS), even if each individual request is normal.

4.3.2 *Deep Learning*

The use of ANNs to solve AD problems is attractive due to the good results they present in other fields. This utilisation of deep models in the AD field is called Deep Anomaly Detection (DAD). For each domain are used different techniques and DNN architectures.

Autoencoders

There are two main components in the AE architecture. The first is the encoder, that provide a dimension reduction over the input and the second is the decoder that makes the reverse process. The input and output in the autoencoder training process is the same (Chollet, 2017). An autoencoder is represented in Figure 4.10. One of the most popular applications of this architecture is AD (Chalapathy and Chawla, 2019).

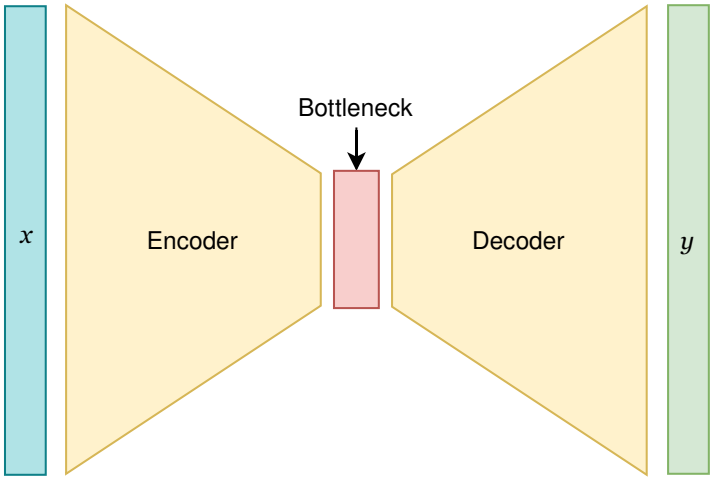


Figure 4.10: Autoencoder architecture.

The behaviour of the network consists in reconstruct the original data. In Figure 4.10, the original data is represented by x and the reconstructions by y. During the training process, the AE learns to reconstruct only the *normal* class instances, either if it is the predominant class in a mixed dataset with few anomalies, either if it is purposefully exposed exclusively to that class. In this scenario the network overfitting to that class is desirable. When the network is trained, the anomalies are found by comparing the original and reconstructed data. It is considered an anomaly if the reconstruction similarity is low (or the reconstruction error is high), according a defined threshold.

Variational Autoencoders

The Variational Autoencoder (VAE) bahaves very similarly to the AE architecture. Using the same principle of detecting the anomalies through the reconstruction errors, it differs from the AE in the latent space representation. Instead of using a simple tensor to code the inputs, this architecture uses a distribution and a sample from it to reconstruct each data point. Thereby, the reconstructions are not expected to be so similar since a random sample is performed. However, this approach have the advantage of grouping similar inputs in the latent space. Figure 4.11 shows the VAE architecture.

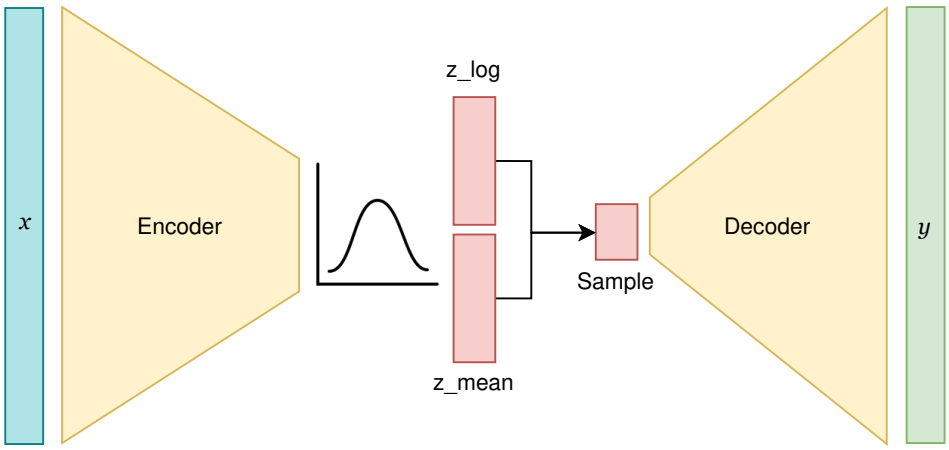


Figure 4.11: Variational Autoencoder architecture.

The *z_mean* and *z_log* variables represent the mean and variance of the latent distribution, respectively. The *Sample* is the random selected point from that distribution.

4.3.3 *Related Work*

In the survey made by Chalapathy and Chawla in 2019 (Chalapathy and Chawla, 2019) are presented some architectures used in AD. Deep Belief Networks (DBNs) are generative models composed by Restricted Boltzmann Machine (RBM) internal layers. DBN models are compared to SVM models, proving that the first is more efficient and scale better than the second (Wulsin et al., 2010). Generative models are also used in AD, using reconstruction probabilities to detect the anomalies. The most common generative architectures in this domain are VAEs and Generative Adversarial Networks (GANs) (An and Cho, 2015; Schlegl et al., 2017).

As seen in Chapter 4.2, CNNs are used in the feature extracting process for image data. These networks can also be in image AD (Minhas and Zelek, 2019), but also in other types of data such as text data (Gorokhov et al., 2017).

AD applied to sequential data has attracted also interest due to the various possible applications in time-series problems. LSTMs, a type of RNN model, presents performance enhancements over conventional methods when dealing with this type of data (Chalapathy and Chawla, 2019; Ergen et al., 2017). The models can be combined to improve performance and decrease the need to preprocess the raw data. Commonly AE are joined with other models to detect anomalies, depending on the task to perform. For example, in sequential data are used LSTM-AE and in images are used CNN-AE or CNN-VAE networks.

4.4 TOOLS

To perform DAD in image sets, two main auxiliaries. The first is the use of a framework to enable the design of DNN and the corresponding architectures that present good results in AD scenarios. These tools are presented in Section 3.10). The second is an image preprocessing mechanism to extract some important features from the image, making the ANN work more manageable.

OpenCV

The OpenCV library is designed for CV tasks. It is open source and it provides interfaces for Python, C++, Java and MATLAB programming languages (Bradski, 2000). There are thousands of algorithms implemented in the library, including ML approaches to the most diversity of tasks, such as object recognition or object tracking.

Regarding image processing, OpenCV has a specific module to help in the process, modifying 2D images. The image filtering module permits to apply linear and non-linear operations, representing the desired filters, to the images in a straightforward way.

Part II

DEVELOPMENT

5

PRELIMINARY STUDIES

> *I think that in the discussion of natural problems we ought to begin not with the Scriptures, but with experiments, and demonstrations.*
>
> Galileo Galilei

5.1 INTRODUCTION

AD is a common problem in the CV field. Due to the bias in some image datasets, it is frequent to have a small representation of some classes. AD can be treated as a supervised or unsupervised problem, depending on data availability. There are several approaches to solve this problem by using DNN. The majority of the proposed solutions use semi-supervised techniques to achieve their goal. These techniques include deep generative models as AEs and GANs.

In this chapter it is presented the first practical use case of using DNNs to detect anomalies in image sets. The objective of this experimental and introductory case study is to better understand the ideas underlying the application of connectionist systems to AD in images, building and improving the studied models. The data used in this scenario is the Mixed National Institute of Standards and Technology (MNIST) dataset, a simple image set known as the "Hello World" for CV and image processing. Based on the acquired results, the use of this simpler dataset allows to validate the proposed methodology and models, before applying them to more complex problems. As the inputs for the models are images, all of them have convolutional layers in their architecture, due to the good results of its application presented in the literature.

Therefore, the present work lies in the intersection of the specific areas explained in the chapters 3, 4.2 and 4.3. The white dot in Figure 5.1 represents a visual depiction about the research subjects involved.

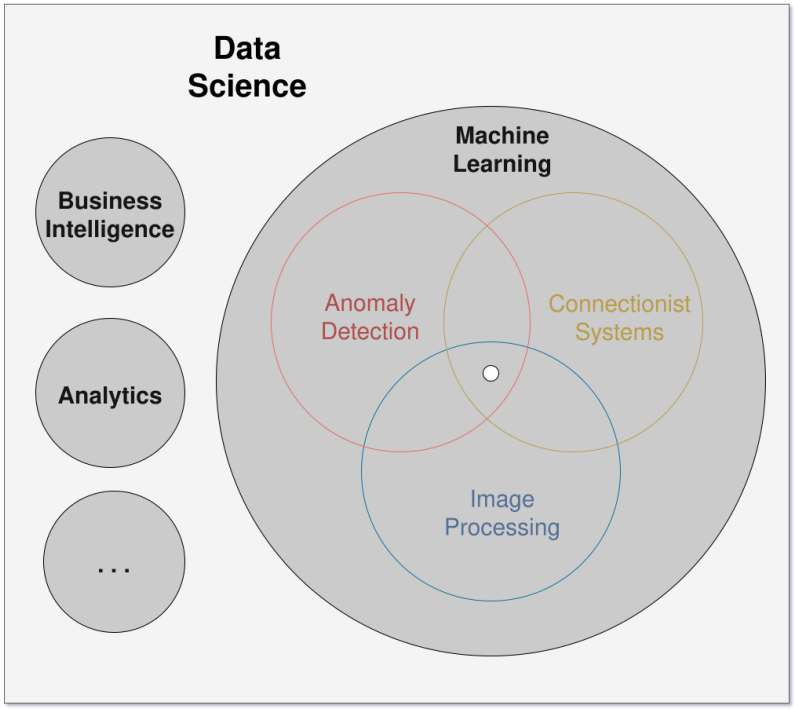


Figure 5.1: Research subjects Venn diagram - MNIST case study.

5.2 RELATED WORK

In 2017, Thomas Schlegl et al. (Schlegl et al., 2017) proposed the use of an unsupervised learning method to identify anomalies in imaging data. In their work, they use a deep convolutional generative adversarial network, a hybrid CNN and GAN, which is trained in non-anomalous data and is able to identify novelties through a return score, where higher scores represent images not seen in the training process. This work was applied to medical images of optical coherence tomography to detect and quantify disease markers. The results presented an 88.34% precision. In the same context, Schlegl et al. in 2015 (Schlegl et al., 2015) used CNNs with labelled data to solve the same problem with an accuracy of 95.98%. CNN models have been used in other contexts like industrial surface inspection (Staar et al., 2019) or AD in crowded scenes with surveillance cameras (Sabokrou et al., 2018). Minhas and Zelek also use CNNs with transfer learning to perform AD in 3 available datasets (CIFAR10, MNIST and Cement Crack datasets) (Minhas and Zelek, 2019). They use different base networks, like *DenseNet*, *ResNet* and *Inception*, and presented the results comparing the different architectures on the different datasets.

In November 2018, Akçay et al. present *GANomaly*, an ANN architecture that applies GANs to X-Ray Security Screening. They proposed an architecture with 3 subnetworks: a CNN-AE, an encoder and a discriminator. In 2019, the same authors trained a model using GANs and CNN-AE concepts with skipping connections between

convolutional layers (Akçay et al., 2019). The results obtained are better than the previous work in the CIFAR-10 dataset.

Other work with AE networks was done by Baur et al. in brain magnetic resonance images (Baur et al., 2019). The main objective is to detect anomalies comparing the input image with their reconstruction. For this purpose, the model has to learn how to reconstruct not-damaged brain parts and not be able to reconstruct the anomalies. They test various AE setups with combinations on variational or non-variational and dense or spatial configurations. The results show that perhaps the model can ignore damaged parts on the brain, it can't reconstruct some important brain parts.

Deep generative models and AEs have been used in group AD applied to images by Chalapathy et al. (Chalapathy et al., 2018). They used Adversarial Autoencoder (AAE) and VAE models to detect anomalous image sets. The presented case studies were the detection of tigers within cats; the detection of images with cats and dogs together in a dogs-vs-cats dataset; the identification of rotated pictures of cats; and the detection of stitched scene images. In this work are used convolutional layers to extract the image features. In this scenarios, almost every AAE models presented better results than VAE architectures.

5.3 PROBLEM DEFINITION

Using the existent classes of the MNIST dataset, a experimental problem can be generated. Thereby, using the classes representing the handwritten numbers "0" and "1" as "Normal" and "Anomaly" cases, respectively, the main goal is to produce models that are able to identify correctly the anomalous situations.

Translating the problem into variables, the system receives an image of a handwritten number, which represents 28x28=784 quantitative features, one for each pixel value; and the target will be a qualitative variable with two possible values: "Normal" and "Anomaly". Depending on the model architecture the target can be represented in different ways, for example using label encoding. As there are only two possible classifications, it is a binary classification problem.

5.4 DATA ANALYSIS

The MNIST dataset is a popular image dataset representing handwritten digits in black and white images with a size of 28x28 pixels.

The database is divided in two datasets, 60 000 images for training and 10 000 for testing, each of them with 10 classes, representing the numbers between 0 and 10. This dataset is a subset of a larger database and it was created by mixing two particular image sets: Special Database 3 collected from Census Bureau employees and Special Database 1 collected among high-school students LeCun et al. (1998). Figure 5.2 shows some examples from the dataset.

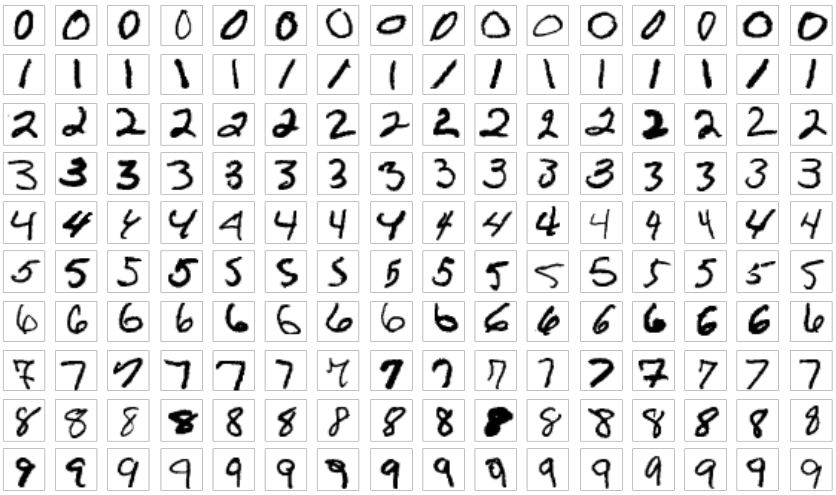


Figure 5.2: Examples of images from MNIST dataset.

The differences between instances of the same class, represented for each row in the image, are evident. Any built model has to be capable of perceiving the patterns by class, regardless of this variance.

The EDA process in the MNIST images is not so useful as in more complex use cases. Figure 5.3 shows the visualization of the histogram of some examples.

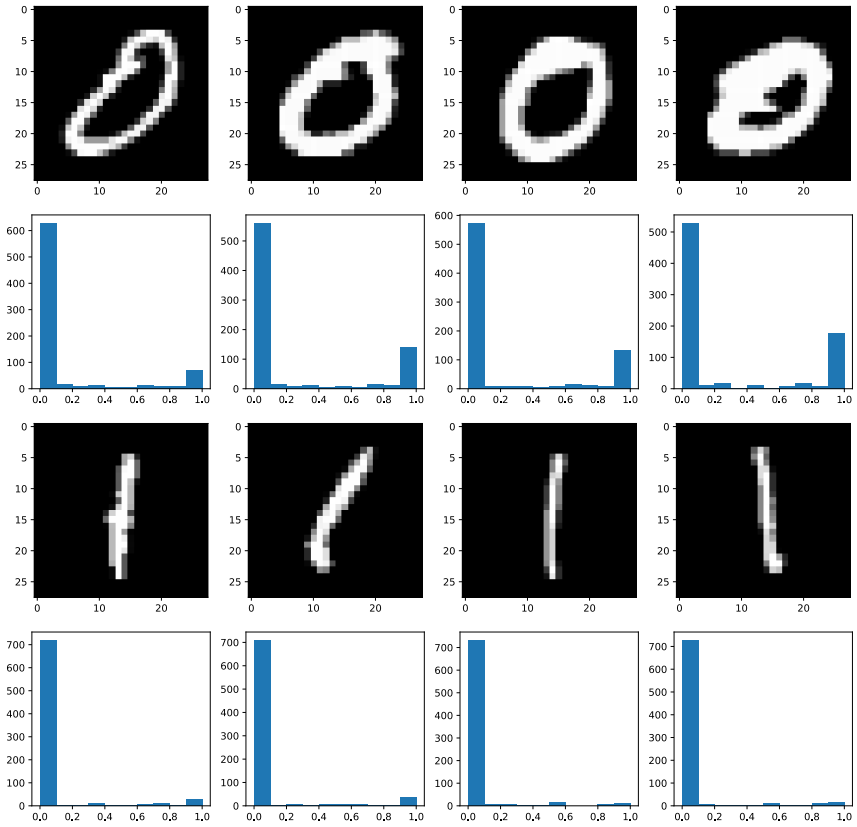


Figure 5.3: Histogram analysis of examples from the MNIST dataset.

Darker pixel values are presented on the left (near 0) and brighter pixel values are located on the right (near 1). As expected, the “Normal” class (“0”) - first and second rows - presents more white pixels than the “Anomaly” class (“1”) - third and fourth rows.

5.5 DATA MODELLING

This section shows the steps taken to build and train the models, using the defined ML workflow, more specifically the phases 3 to 8. There are 3 models presented: a classifier network, an AE and a VAE.

Data Segregation

The data is segregated 2 times. The first one using a script that splits the images into 2 directories, *training* and *testing*. The second, at runtime where the data is adapted depending on the model architecture and the approach needs. This permits a more flexible treatment of the data.

The segregation script stores 80% of the data in the *training* folder and the remaining 20% to the *testing* directory. This is logical since the models need big quantities of data to improve during training. The selected 80-20 proportion is commonly used in ML projects.

During the modelling phase, the data is segregated as follows:

- *training* - 90% to update the model weights (training), 10% to monitor the learning process (validation);
- *testing* - 60% to calculate the threshold, 40% to evaluate the model.

The data segregation is made specifically for each model architecture, depending if there is a need to use some part of the data for further calculations or not. The classifier model uses all the *testing* data for the final model evaluation, whereby the last 60-40 segregation is not performed in this case.

5.5.1 *Classifier*

The classifier model is the most straightforward solution in this case. Its architecture is composed by stacked convolutional layers to extract features from the image and a fully-connected classifier on top.

The number of filters used in the convolutional layers are 8 and 16 for the first and second layers, respectively. The filter size is 3x3 for both layers. In the middle of these layers, is used a *MaxPooling* 2x2. The classifier has two *Dense* layers, one with 32 neurons and the output with 1 single node. A possibility for the last layer was to use 2 nodes, one for each class, changing the loss function accordingly.

All layers present the ReLU as activation function, except the last one. In this case, the *sigmoid* function is used to get results in the [0,1] interval. The cut point to divide the classes is the middle of the interval (0.5). The result is expected to approximate as much as possible to the ground-truth labels, using the *binary_crossentropy* loss function and the *rmsprop* optimizer.

The training process is completed at the end of 50 epochs. After the 30th step, approximately, the validation metrics show a stabilisation in training whereby the train could be interrupted at this point since there are no significant improvements in the results (see Appendix A, Figure A.1). One alternative is to use the *EarlyStopping* callback that uses an arbitrary metric to stop the training when there are no significant improvements.

Figure 5.4 shows the raw predictions obtained with the classifier. The division is clearly achieved even with a simple ANN architecture and a few training steps (note that the validation accuracy was near 100% since the first training epoch).

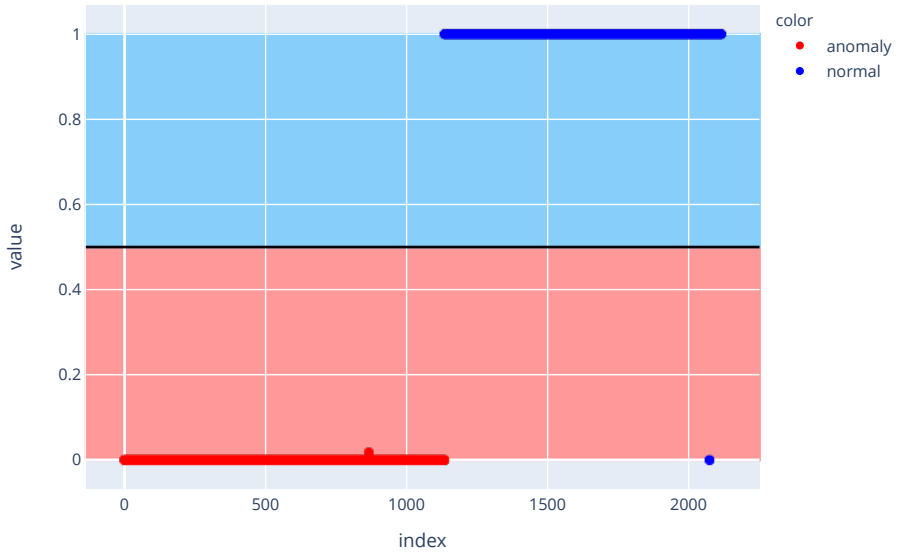


Figure 5.4: MNIST Classifier - Predictions.

The model is nearly perfect in the task of distinguish the two classes, misclassifying only 1 of the 2115 test cases (Table 5.1). The AUROC also show that the model can distinguish the classes almost perfectly (Figure 5.5). The confusion matrix can be found in Appendix B.

Table 5.1: MNIST Classifier - Classification Report.

	Precision	**Recall**	**support**
anomaly	1.00	1.00	1135
normal	0.99	1.00	980
Accuracy	1.00		2115

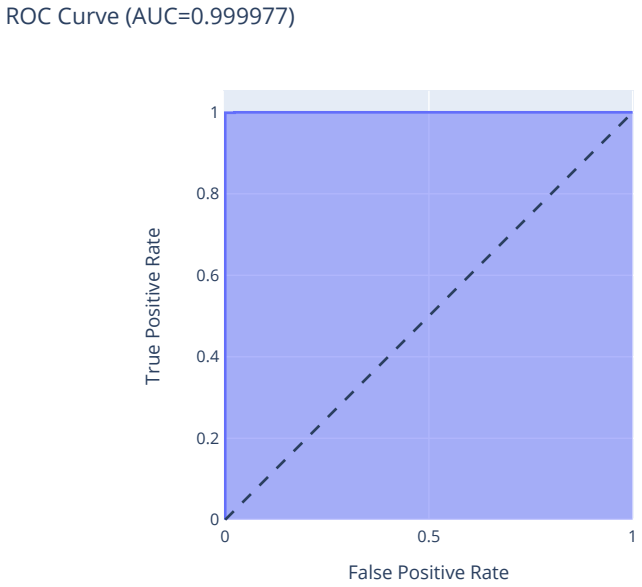


Figure 5.5: MNIST Classifier - AUROC.

5.5.2 *Autoencoder*

The AE model is used to build reconstructions from the original data classified as normal and a high reconstruction error indicates that an anomaly is found. This architecture is more complex than a classifier since it needs more operations to perform the reverse process until reach a reconstructed image.

The architecture can be divided in 3 parts:

1. Encoder - 2 *Conv2D* (32 and 64 3x3 filters) + *MaxPooling2D* layers (2x2 window);
2. Bottleneck - 2 *Dense* layers (40 and 196 neurons);
3. Decoder - 3 *Conv2DTranspose* (all with 64 3x3 filters) intercalated by 2 *BatchNormalization* layers.

The output layer returns an image using the *Conv2D* layer with just one filter (1 color channel). If the images were colored, the output layer would use 3 filters instead to represent the 3 color channels.

The *Flatten* and *Reshape* layers are the entry and exit points for the bottleneck, respectively. The first transform the data into a flat array to be processed by the *Dense* layers, the later modifies it to a specific target shape. The number of neurons in the last Bottleneck layer is very important, so that it can be reshaped properly accordingly the original image size. In this case, 196 outputs in the *Dense* layer are reshaped as 14x14x1, that can give a 28x28x1 image after decoded, using one of the *Conv2DTranspose* layers to double the image size, using strides of 2x2. The activation functions in all layers is the ReLU function, except in the last layer, where the *sigmoid* function retrieves a value in the [0,1] interval representing the darkness of each pixel.

The model is trained in 10 epochs with the *RMSProp* optimizer, and the MSE loss function, since the goal is to minimise the difference between the pixels values of the real and reconstructed images.

The results are not straightforward as in the classifier model and they are not the problem solution. In this case, some further steps must be taken to get the distinction between the two classes. Examples of reconstructions for both classes are presented in Figure 5.6.

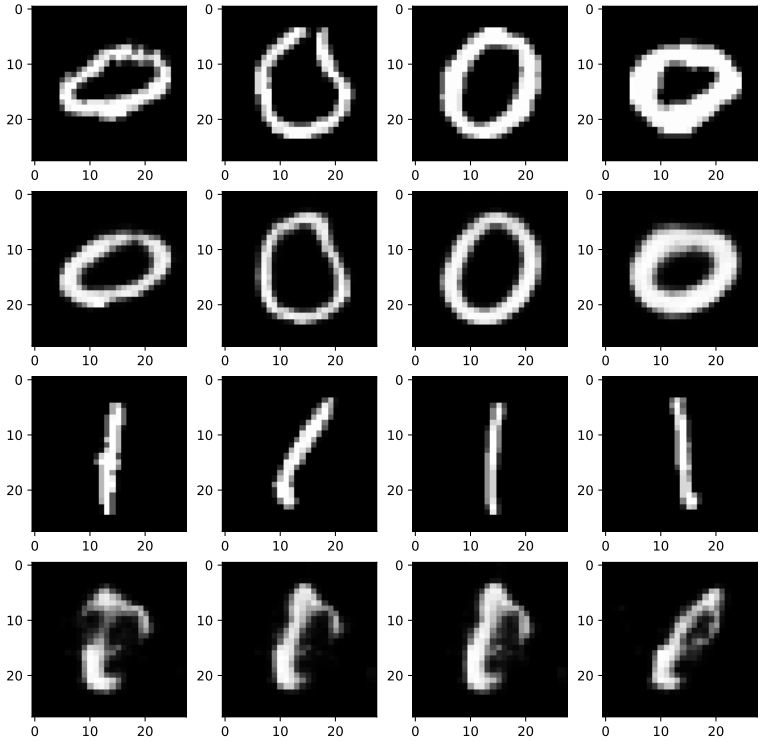


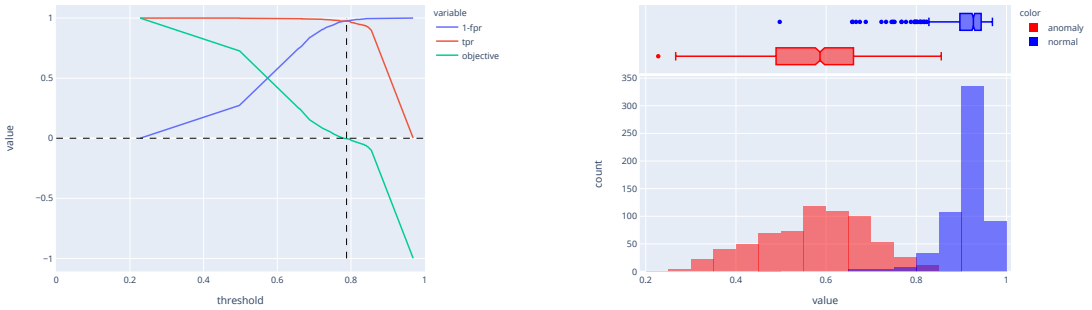
Figure 5.6: MNIST AE - Reconstruction examples. First and third rows are examples of the original images for "Normal" and "Anomaly"classes, respectively. Second and fourth rows are the respective reconstructions.

It is visible that the model is not able to reconstruct the anomaly cases. From these images, it is needed to calculate the likelihood between the input and reconstructed images, using the SSIM metric to be able to split the two classes.

In order to split accurately the images, it is necessary to choose the correct cut threshold. This is achieved using the AUROC plot (Figure 5.8) and finding the nearest point from the upper left corner, which would be a perfect classifier, with a TPR=1 and FPR=0. Since the domain of both metrics is the [0,1] interval, the objective is to maximize both the TPR and $1 - FPR$. This is achieved using the objective function defined in Equation 23, that represents the intersection between the 2 variables (Figure 5.7a).

$$\begin{aligned} TPR(t) &= 1 - FPR(t) \\ \equiv TPR(t) - (1 - FPR(t)) &= 0 \end{aligned} \tag{23}$$

where t represents the threshold.



(a) Supervised calculation method.

(b) Unsupervised calculation method.

Figure 5.7: Best threshold calculation using the values from the curve.

As shown in Figure 5.7a, the equality $objective(t) = 0$ is reached near the point $threshold = 0.79$.

Even though the overall accuracy drops to 98% face to the classifier model, the results still pointing to a very viable solution with a precision and recall values of 97% and 100%, respectively. Note that the support is lower than the classifier network since some data needs to be used in the threshold calculation.

In scenarios where the "Anomaly" data is not available or when there is a few instances of that class, it is useful to use a method that doesn't need labelled data. Using only the class of the training process ("Normal"), it is visualised the distribution of the reconstructions similarities.

The threshold is defined using the lower fence, since image reconstructions bellow this point should not belong to the "normal" class distribution. The lower fence is calculated using the distribution information: the quartiles and Interquartile Range (IQR). Equation 24 shows the calculation of the threshold as the lower fence, which leads to the 0.83 value. Figure 5.7b represents that information in the blue boxplot above the similarities distribution.

$$t = Q1 - 1.5 \cdot IQR \tag{24}$$

where t represents the threshold and $Q1$ the first quartile (percentile 25).

The "Anomaly" class is presented in the plot for display purposes only. It is not used for any calculations and it must be treated as unknown data.

To simplify the reference to both threshold classification methods, they can be classified using the analogy with supervised and unsupervised learning paradigms. Therefore, the supervised method is the one that needs labelled data to perform the division and the unsupervised method is the one that only uses the data distribution (assuming that the majority of data is from the "Normal" class). Note that the AE learning paradigm is self-supervised and is not related to the threshold calculation.

With the selected cut-points, the resultant predictions are shown in Figure C.2.

The metrics in the classification report (Table 5.2) show that both methods, supervised and unsupervised, produce credible results with good performance for this test experiment.

Table 5.2: MNIST AE - Classification Report.

	Supervised		Unsupervised		
	Precision	**Recall**	**Precision**	**Recall**	**support**
anomaly	0.97	1.00	0.94	1.00	454
normal	0.99	0.96	1.00	0.93	392
Accuracy	0.98		0.97		846

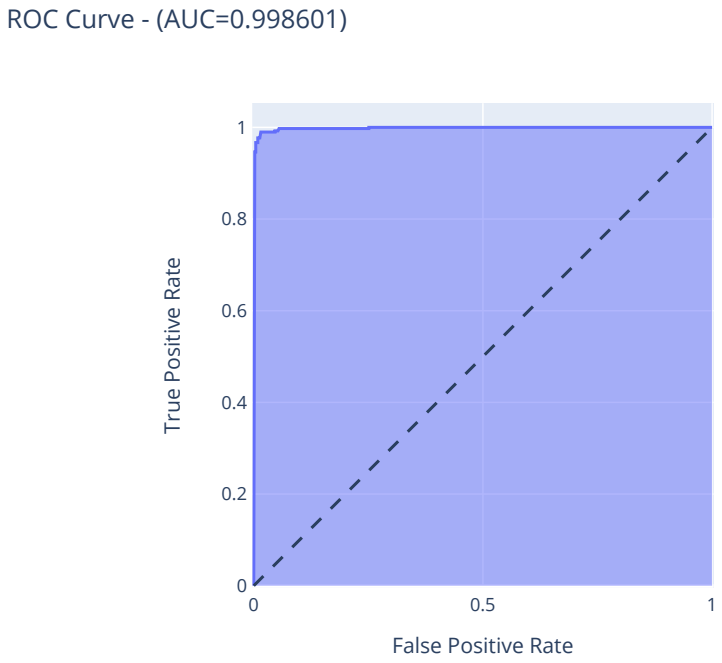


Figure 5.8: MNIST AE - AUROC.

5.5.3 *Variational Autoencoder*

The principle of using a VAE is the same of a conventional AE. The objective is to distinguish the anomalous cases, that are novelties for the model, by the differences in the reconstruction error. However, the internal behaviour of this model is different, since the outputs are generated from samples of a distribution. The building of a VAE is also more complex, considering the need to define proper layers to sampling the data and a specific loss function that combine the approximation of sampling distribution to a $N(0, 1)$ and the minimisation of the reconstruction error.

The encoder architecture is similar to the AE encoder structure, with a slightly difference. In this case, before the *Bottleneck*, there are two forked *Dense* layers, that represent the mean and standard deviation of the latent distribution. From this distribution, a sample is acquired using a *Lambda* layer that uses a previous specified sampling function.

The output of this layer is the VAE *Bottleneck* and therefore, the output of the encoder part. The decoder has the same behaviour of the AE decoder, building an image similar to the input from the latent representation.

However, the intermediate distribution representation is responsible for a increase in the model building phase. For the training process, a special loss function needs to be defined, using the sum of two losses: the Kullback–Leibler divergence measure to approximate two distributions (Equation 11) and the MSE to minimize the difference between the original and reconstructed pixel values (Equation 1).

The intermediate representation also difficult the network learning process introducing the vanishing gradient problem, where gradients used in the back-propagation algorithm lose significance in the first layers of the network.

The used optimizer is the *Adam* with a learning rate of 0.0005, preventing the problem of exploding and vanishing gradients in the weights updating process.

In this case, the reconstructions are not expected to be so similar as the AE, since they are not a perfect correspondence to the input, but an example from a set of inputs with the same characteristics. An example of the reconstructions can be seen in Figure 5.9.

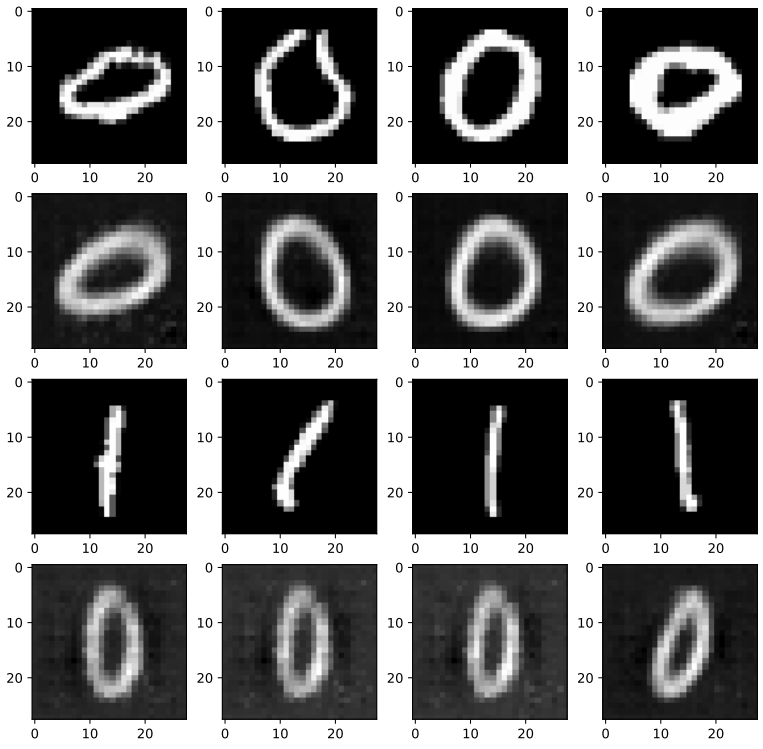


Figure 5.9: MNIST VAE - Reconstruction examples. First and third rows are examples of the original images for “Normal” and “Anomaly”classes, respectively. Second and fourth rows are the respective reconstructions.

Characteristics such as the roundness of the “zeros” and their orientation are well captured by the network. When trying to reconstruct the “ones”, the network recreate them as similar as possible, outputting “zeros” less round and oriented vertically. These are examples of feature vectors, that can be useful in other problems.

The results are acquired splitting the classes by the reconstruction error threshold, using the same supervised and unsupervised techniques of the AE (Figure C.3).

As in the previous model, the confusion matrices and the classification report are presented in Figure B.3 and in Table 5.3.

Table 5.3: MNIST VAE - Classification Report.

	Supervised		Unsupervised		
	Precision	**Recall**	**Precision**	**Recall**	**support**
anomaly	0.97	1.00	0.97	1.00	454
normal	1.00	0.97	1.00	0.97	392
Accuracy	0.98		0.98		846

The AUROC metric is depicted in Figure 5.10, showing also the good performance of this model separating the two classes.

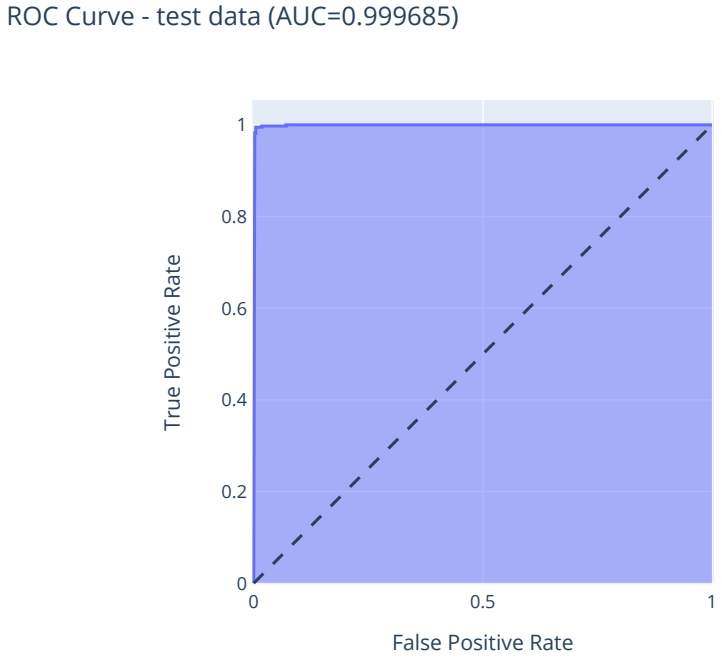


Figure 5.10: MNIST VAE - AUROC.

5.6 CONCLUSIONS

The use of AD techniques in CV has lead to a combination of different approaches, where convolutional layers are embedded in AD dedicated DNN architectures, extracting the features from the images while using the same principles.

Even in unsupervised scenarios, where there are few anomaly cases in the training data, it stills possible to build models that are capable of identify these cases. This is accomplished with the distribution of the cases reconstruction similarities, using its lower fence as the threshold to split the classes, since the model will not be capable of reproduce unseen data points (in this case, the anomaly ones). Only AE and VAE models are used in this approach, since the classifier needs the two classes during the training process.

Despite of the main purpose of the introductory use case is to validate and set up the models to improve them in the further steps, the results can be compared to get some insights of what approach could be the best. Table 5.4 shows the summary of the different approaches with the most useful metrics.

Table 5.4: MNIST models comparison by metric.

	Classifier	**AE**		**VAE**	
	Supervised	**Supervised**	**Unsupervised**	**Supervised**	**Unsupervised**
AUROC	0.99	0.99		0.99	
Accuracy	1.00	0.98	0.97	0.98	0.98
Precision	1.00	0.97	0.94	0.97	0.97
Recall	1.00	1.00	1.00	1.00	1.00

In the experimental case study, using the MNIST dataset, all the methodologies can be validated since the results show a clear distinction of the anomalous cases in every studied model architecture and threshold calculation. This is the starting point to apply these DNNs to more complex problems, increasing their complexity and representation power.

6

HIGHWAY PAVEMENTS MONITORING

> *Roads were made for journeys not destinations.*
>
> Confucius

6.1 INTRODUCTION

Highways are one of the most important assets in the daily life of modern societies, increasing the economic gains of many activity sectors, the citizens quality of life and the countries development, with special impact in urban areas (Rephann and Isserman, 1994). After the highway construction, the pavement tends to degrade due to different factors, like meteorological conditions, materials self-deterioration or the road wear by its utilisation.

It is proven that bad pavements impact not only the drivers comfort, but also their safety (D'Amico et al., 2018). The accident rate is correlated with the pavement condition, where higher values of roughness and rut depth increase crash rate in highways (Vinayakamurthy, 2017). A degraded asphalt leads also to an increase in the vehicles operation cost due to damage or depreciation on its mechanical components. Additionally, the road maintenance itself can cause traffic jams, driver stress and increase the fuel consumption (Silva et al., 2017).

Therefore, it is imperative to periodically monitor the highway pavements and identify the anomaly locations in the road. This information is useful to decide what measures can be adopted to repair damaged asphalt parts or to preserve healthy ones.

The chapter is divided in order to roughly follow the workflows presented in the chapters 2.2 and 2.3. After the domain study presented in the sections 6.2, 6.3 and 6.4, the steps in the data pipeline, including the modelling phase, are described in the sections 6.6 to 6.9. The result analysis is presented in Section 6.10 and the respective conclusions of the chapter are presented in Section 6.13.

6.2 ANOMALIES IN PAVEMENTS

Highway pavements can be classified as flexible or rigid, depending on the construction materials used. A flexible pavement is commonly built with bituminous material, while a rigid pavement use concrete in its composition (Tom V. Mathew, 2009). For each pavement type, there are several anomaly patterns with different levels of severity. The most common in both flexible and rigid pavements are represented in the following list (Research

& Development division of the Highway Department, 2013; Gabinete de Gestão da Rede - Estradas de Portugal, 2008).

- **Cracking** - Fissures due to fractures on the pavement. They can be isolated or connected and they can have different orientations on the road (parallel, perpendicular or diagonal to the road axis). In more severe cases, they can create a cracking mesh.

Figure 6.1: Example of pavement cracking (adapted from Research & Development division of the Highway Department (2013)).

- **Deformations** - Changes on road structure with surface modification. The pavement shape is adulterated, with a different height in the affected areas. This anomaly can be caused by traffic or environmental reasons. Can be called elevation defects.

Figure 6.2: Example of pavement deformation - rutting (adapted from Research & Development division of the Highway Department (2013)).

- **Surface Texture Deficiencies** - Modifications in the surface layer of the pavement caused by materials loss. Those losses can be macroscopic, provoking for example potholes, or microscopic with wear or disintegration of pavement elements.

Figure 6.3: Example of pavement surface texture deficiencies - pothole (adapted from Research & Development division of the Highway Department (2013)).

- **Material Movements** - Disintegration of materials from the pavement or rise of materials from the ground or lower layers to the surface.

Figure 6.4: Material movements in the pavement example - ravelling (adapted from Research & Development division of the Highway Department (2013)).

6.3 TRADITIONAL SOLUTIONS

The traditional methods to supervise the asphalt include the direct observation of the road, with manual annotations, which is a very rudimentary method. The data can be stored in paper format and it has to be processed afterwards, or digitally, where some systems like VIZIROAD® help in the data acquisition process, facilitating the later use of information. The VIZIROAD® system (Figure 6.5b) has two keyboards and a correspondent software to handle the data collection process (Picado Santos et al., 2006). These processes represent very time-consuming practices that are difficult to scale up and implement in larger highway roads networks. Fur-

thermore, the VIZIROAD® system is deprecated, since the software is not compatible with modern operating systems.

Other approaches use complex systems with 3D image capturing and laser profiling sensors to provide a more detailed report of the pavement conditions (Laurent et al., 2012; Wang, 2011). Laser technologies are integrated in some commercial solutions as PaveTesting® or Dynatest® [1]. These systems require a specific equipment attached to a vehicle and specialised workers to operate it. Perhaps the accurate results provided by these mechanisms, they represent an expensive solution, with high cost-benefit ratio, that is not compensatory for most entities.

(a) Pavetesting® lasers installation in a monitoring vehicle.

(b) VIZIROAD® system with 2 keyboards for data acquisition. In this solution, the pavement information is inserted by an operator based on visual observations of the road (Adapted from Picado Santos et al. (2006)).

Figure 6.5: Examples of existent solutions for highway pavement monitoring.

6.4 RELATED WORK

In order to mitigate the disadvantages of the existent solutions, different research approaches are being continually explored. One of the most used strategies is the application of smartphone devices to collect different kinds of data that is posteriorly used in data mining processes. In their 2017 work, Silva et al. (Silva et al., 2017) proposed a AD system, where all the data is collected by a smartphone. Here the anomalies are divided in 4 main groups (*unlevelled manholes*, *long bumps*, *short bumps* and *others*). The acquired data consists in 4 features (*timestamp*, *GPS location*, *speed* and *accelerometer data*) and 1 target variable (*anomaly*). Beyond these attributes, some more has been derived, like maximum, minimum and mean values for accelerometer x, y and z variables. For the predictions they used different algorithms, where SVM presented the best results with a subset of all features, based mostly in the accelerometer data. The model can predict correctly non-anomaly cases with a 97.5% accuracy, but performs poorly in classify the other anomalies compared to other algorithms like

1 https://pavetesting.com/
https://www.dynatest.com/

gradient boosting. In the work continuity, Silva et al. propose the use of the created models to build an anomalies detection system based on collaborative mobile sensing (Soares et al., 2018), where the data is acquired by many users. The proposed architecture and models are evaluated in a real-world scenario with different metrics and training settings and PCA techniques are used to get the most relevant variables for the problem (Silva et al., 2018).

In a similar way, the smartphone accelerometer data is used in other works to predict the road conditions. Yu-chin Tai et al. (chin Tai et al., 2010) used a smartphone attached to a motorcycle and an SVM algorithm to achieve a precision of 78,5%. Kyriakou et al. (Kyriakou et al., 2019) implemented a bagged trees classification model with a prediction accuracy of 98.84% for anomalies distributed in 5 classes. The same approach is taken by Fatjon Seraj et. al (Seraj et al., 2014) using SVMs, achieving an accuracy of 90%. Although using the same inertial sensors technology, Fatjon Seraj et al. (Seraj et al., 2015) presented a different point of view relatively to the previous mentioned works. It is assumed that the assumption that the driver pass through all the anomalies can be discarded. In order to get the AD, it is assumed that the driver will deviate for some of the degradations founded on the pavement, swerving the car from the expected route. The TPR to the swerve class is 70%.

The major problems in those approaches are related to difficulties found in the devices sensors. The sensors are heterogeneous depending on the device brand and model. The GPS data is little accurate in some cases and depending on the accelerometer sensitivity, it can detect activities that are not related with the pavement conditions. The use of inertial sensors is also dependent on some vehicle characteristics like the suspension system, which introduces even more noise to the data acquisition (Masino et al., 2017). Furthermore, anomalies like cracks cannot be detected with accelerometer data since they are more visual and they don't interfere in the car stability.

Instead of vibration-based methods, which are more vulnerable to unknown factors in the data, vision-based ones can be used to avoid the above mentioned problems. This approach has the advantage of providing a visual understanding of the observations, that can be used to understand how each instance is classified.

Several approaches using imaging methods had been also explored. Radopoulou and Brilakis (2017) present the distinction between different methods according the level of detail: presence, detection and measurement. The *presence* is the distinction between good and bad pavement, the *detection* focus on distinguish between different types of degradations and the *measurement* works on a more specific level to identify their severity. The used data can be 3D images (Zhang et al., 2018b,a) with highly gathering costs associated or 2D images (Cubero-Fernandez et al., 2017; Yang et al., 2019) where budget cameras can reduce the solutions costs.

For the present work, the main focus will be the *presence* of degradations in the pavement, using 2D images to achieve this objective. Similar to the approach presented by Wang et al. (2020) for detecting brain tumor anomalies, AE and VAE models will be used to detect the degradations.

6.5 PROBLEM DEFINITION

After the domain understanding step, translated into the previous chapters, answering the questions proposed in Section 2.2.2 of Chapter 4.3, helps to construct a formal problem definition.

1. *What problem is meant to be solved and why is it a problem?* The objective is to automate degradation detection in pavements;

2. *What are the current solutions (if any)?* The current solutions are either obsolete or expensive (see Section 6.3);

3. *What data can be used to solve it?* The proposed data used to solve the problem are images, labelled with different degradations and severity levels;

4. *How can the performance of the results be evaluated?* The solution performance is proportional to the AD models effectiveness, thus the test metrics of the models are used as evaluation measures.

Therefore, the main goal is to prototype a solution that is capable of recognising degradations in pavements automatically, comparing different methods and architectures. The gathered data labels are meant to be divided into two categories ("Good" and "Bad" pavement conditions) for the modelling phase. Later, the methodologies can be further enhanced in order to increase the granularity of distinction between anomaly classes. Ideally, the solutions applied to a real case scenario will reduce the operation costs comparing to existent solutions, while performing similarly.

Even if not used to train the models, it is useful to store a set of variables that represent the state of the road for each location. Additionally, this data is convenient to perform data analysis apart its use to the automated solution. For the variables representations, the following notations are used:

- **Timestamp** - Quantitative variable, representing the system timestamp;
- **Distance** - Quantitative variable, representing the distance travelled in kilometers;
- **Degradation Level** - Qualitative ordered variable, representing the degradation severity;
- **Location** - Qualitative nominal variable, representing the existence or not of the degradation in different transverse road positions (left, middle and right).

6.6 DATA GATHERING

The first step to build a system that can automatically identify the pavement condition is to acquire the data. In a first moment the system has not only to acquire images of the pavement, but also to capture information about the road condition. Later, this information is expected to be automatically generated using the ML models.

6.6.1 *Requirements*

To build the data collection software, it is necessary to take into account the requirements, regarding its future use and the data that will be generated. In general, the requirements raised can be summarised as follows:

1. Gathering of information on road degradation, such as its type and severity level;
2. Acquisition of information on observations that may exist along the route (such as viaducts or crosswalks);
3. Collection of photographic data associated with the degradations;

4. Alignment of the collected data with the travelled distance and the time elapsed since the beginning of the acquisition;

5. Record the collected data for future use.

These requirements serve as the basis for the development of the gathering application, from the user interface to how data is saved after collection.

6.6.2 *User interface*

An user interface is one of the most important aspects of building an application since it is in this component that the usability of software is reflected. The GUI developed is designed for easy use with a touch screen. Figure 6.6 shows the interface created. Note that the interface is in Portuguese, since it is to be used by Portuguese entities.

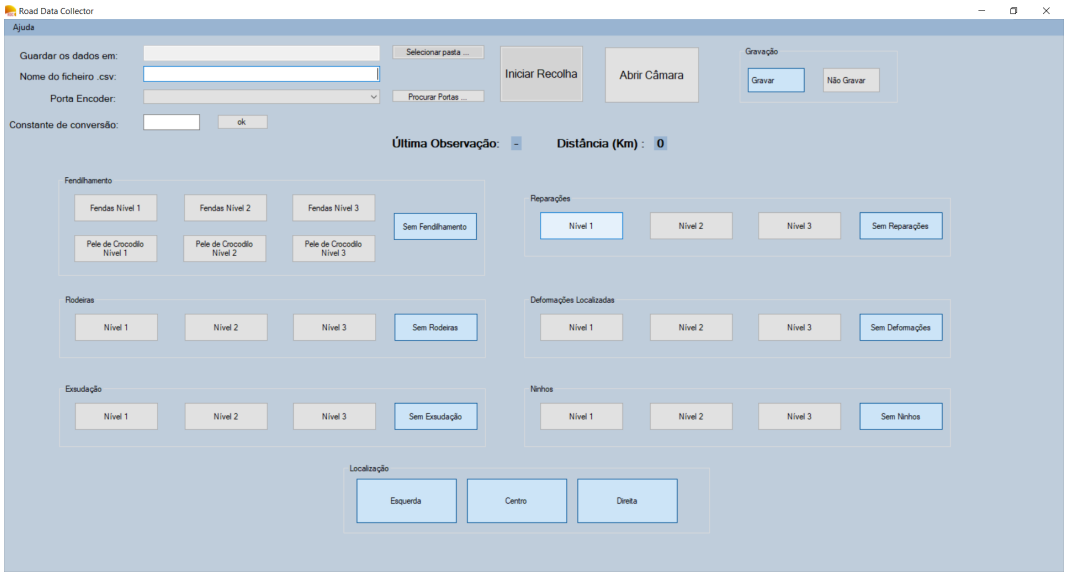


Figure 6.6: GUI of the gathering software.

To satisfy the first requirement concerning the search for degradation information, buttons are included for each degradation, so that the user can insert them as they appear on the pavement.

As mentioned in the second requirement, annotations are captured using keyboard inputs. The last inserted observation is shown in the *Última Observação* (Last Observation) display. Associated with the third requirement, the *Open Camera* button displays the image being captured by the camera, allowing the verification of a proper camera functioning.

Concerning the fourth requirement, the distance travelled so far is shown in the *Distância (Kms)* (Distance) display. The alignment of the collected data with the distance is achieved through an odometer that counts the distance from the beginning of the data gathering. This distance is captured through an encoder whose port must be indicated in the *Porta Encoder:* (Encoder port) field. The encoder is connected to an Arduino, that converts revolutions in the vehicle wheel into electrical signals, allowing to convert a revolution into a number, and

therefore to the travelled distance. Since it depends on the wheel radius, a conversion constant between the two values (revolution to distance) must be specified. Using the formula presented in Equation 25, the conversion constant represents the travelled distance for each wheel rotation, that is the wheel perimeter translated to kilometers. Notice that the appropriate sign must be also included, accordingly to the direction of rotation (positive to clockwise, negative otherwise). This is important because the direction of rotation will be different for different wheels (clockwise for left side, anticlockwise for right side).

$$k = \begin{cases} \frac{2\pi r}{10000}, & \text{if clockwise rotation} \\ -\frac{2\pi r}{10000}, & \text{otherwise} \end{cases} \tag{25}$$

where r represents the wheel radius.

To satisfy the fourth requirement, an internal *timestamp* is added to this information, which makes the association between all the data collected, including distance and as images. To save the information obtained, the user is asked for a folder (*Guardar os dados em:* (Save data in:)) where all the photos taken during the survey will be placed. In this same folder, a CSV file is created with a name customized by the user (*Nome do ficheiro .csv:* (CSV file name)) that stores an occurrence on each line. The data is thus persisted as required by the fifth requirement.

It is also possible for the user to insert the location of road pathologies in relation to their position in the width of the road, using the *Esquerda* (Left), *Centro* (Middle) and *Direita* (Right) buttons. Recording can be paused and resumed using the buttons in the *Gravação* (Recording) section.

6.6.3 *Architecture*

Internally the application is structured in a modular way allowing the internal modification of each component without changing the general behaviour of the system. Each component provides an API that is used by the other modules in order to cooperate in carrying out the desired tasks. Figure 6.7 shows the internal structure of the software.

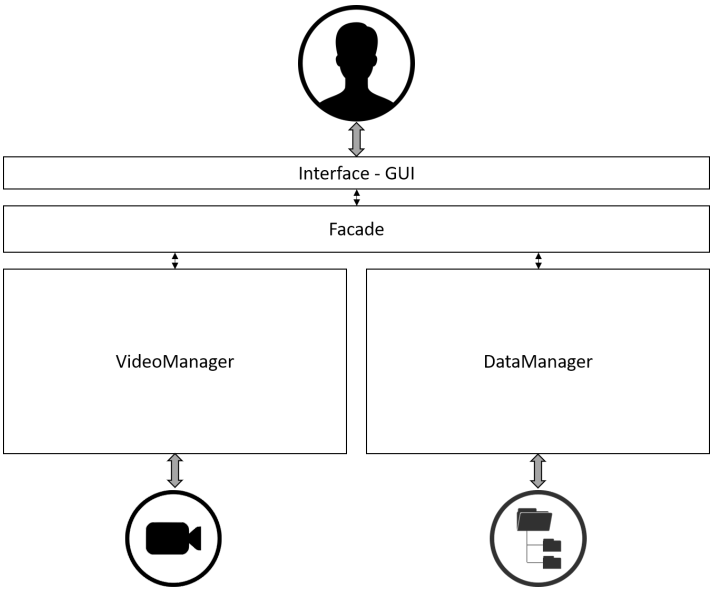


Figure 6.7: Internal structure of the gathering solution prototype.

The point of contact with the user is the graphical interface (GUI), shown in Section 6.6.2. Each request made in this component is sent to the *Facade* module, which is responsible for distributing tasks accordingly. When there is the need to capture images, the *VideoManager* component is called, which in turn communicates with the camera device. This module is responsible for receiving the image, saving it in the specified folder with the appropriate file name, denoting the timestamp of acquisition.

The *DataManager*, in turn, also receives orders from the *Facade*, however, it has the function of communicating with the file system, where the CSV file with the data is created, updated during the data acquisition and saved on the end. Note that the encoder data is captured directly on the interface via a COM port linked to the Arduino, so there is no need to build a specific module for that task.

6.6.4 *Execution Flow*

When the user interacts with the system, the flow of actions that is triggered between the different levels of the application translates into the following set of internal communications:

- When starting a new data gathering:
  1. *Interface* communicates the new event to *Facade*;
  2. *Facade* informs *VideoManager* to open the camera and *DataManager* to create a new CSV file;
  3. *VideoManager* gives an opening instruction for the active camera and *DataManager* creates a new file, leaving it opened for writing.
- When inserting occurrences of degradations:
  1. *Interface* communicates to *Facade* the anomaly entered and the distance marked on the odometer;
  2. *Facade* collects the system time (*timestamp*);

3. *Facade* inform *VideoManager* to capture an image and save it with the collected *timestamp* and *DataManager* to write a new entry with distance, *in* the CSV file timestamp and information about the degradations;
4. *VideoManager* and *DataManager* communicate with ther external systems, completing the requested actions.

- When finished collecting:
  1. *Interface* communicates to *Facade* the end of the data gathering;
  2. *Facade* informs *VideoManager* to close the camera and *DataManager* to close the CSV file;
  3. *VideoManager* gives the closing instruction for the camera and *DataManager* orders the file system to close the open descriptor.

6.7 DATA STORAGE

The generated data is saved in CSV format, storing an image for each event in the same directory. The CSV file is structured as shown in Figure 6.8.

A	B	C	D	E	F	G	H	I	J	K	L	M
Obs	TimeStam	Distância(	Fendilham	Pele Croc	Ninhos	Reparaçõ	Rodeiras	Deformaç	Exsudação	Esquerda	Centro	Direita
-	942359	0.003	0	0	0	0	0	0	0	1	1	1
-	942656	0.005	0	0	0	0	0	0	0	1	1	1
-	943062	0.007	0	0	0	0	0	0	0	1	1	1
-	943453	0.01	0	0	0	0	0	0	0	1	1	1
-	943750	0.012	0	0	0	0	0	0	0	1	1	1
-	943953	0.014	0	0	0	0	0	0	0	1	1	1
-	944140	0.016	0	0	0	0	0	0	0	1	1	1
-	944359	0.018	0	0	0	0	0	0	0	1	1	1
-	944656	0.021	0	0	0	0	0	0	0	1	1	1
-	944953	0.023	0	0	0	0	0	0	0	1	1	1
-	945156	0.025	0	0	0	0	0	0	0	1	1	1
-	945343	0.027	0	0	0	0	0	0	0	1	1	1
-	945562	0.029	0	0	0	0	0	0	0	1	1	1
-	945750	0.032	0	0	0	0	0	0	0	1	1	1
-	945953	0.034	0	0	0	0	0	0	0	1	1	1
-	946140	0.036	0	0	0	0	0	0	0	1	1	1
-	946359	0.038	0	0	0	0	0	0	0	1	1	1
-	946546	0.04	0	0	0	0	0	0	0	1	1	1
-	946750	0.042	0	0	0	0	0	0	0	1	1	1
-	946953	0.044	0	0	0	0	0	0	0	1	1	1
-	947250	0.047	0	0	0	0	0	0	0	1	1	1
-	947562	0.05	0	0	0	0	0	0	0	1	1	1
-	947781	0.052	0	0	0	0	0	0	0	1	1	1
-	947953	0.055	0	0	0	0	0	0	0	1	1	1

Figure 6.8: Structure of the generated CSV file.

The following columns are presented in the created file:

- **Obs (A)** - Keys associated with notes captured from the keyboard;
- **Timestamp (B)** - System internal timestamp associated with the each event;
- **Distance (C)** - Distance travelled in kilometers;
- **Anomalies (D-J)** - Levels of each pathology (Cracking, Crocodile Skin, potholes, Repairs, Rutting, Localized Deformations and Exudation);

- **Location (K-M)** - Binary variable indicating the occurrence of pathologies in the left, central and right blocks of the road, with pathologies referring to the blocks where the value 1 appears.

Each image is saved with the correspondent timestamp from the file. Figure 6.9 shows some examples of a folder with captured images.

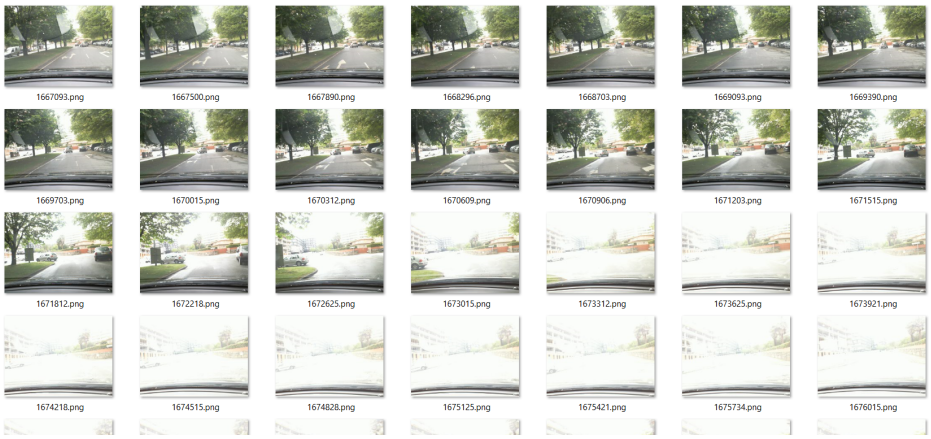


Figure 6.9: Examples of captured images.

The camera is located inside the vehicle, due to the need to it to be attached to the laptop and its wire length. The used camera is a low-price webcam (Microsoft® LifeCam 6CH-00002).

Although the present gathering conditions are not the best, the software prototype is proven to perform well in acquiring all the needed data, acquiring images and text information consistently along the way.

6.8 DATA ANALYSIS

Itinerary maps are the conventional way to record the degradations present on a given road segment. As shown in Figure 6.10, for each pathology the 3 possible levels are shown, their location and extension given by the beginning and the end of the horizontal bars and their occupation of the road in terms of percentages. Some information is omitted due to privacy concerns.

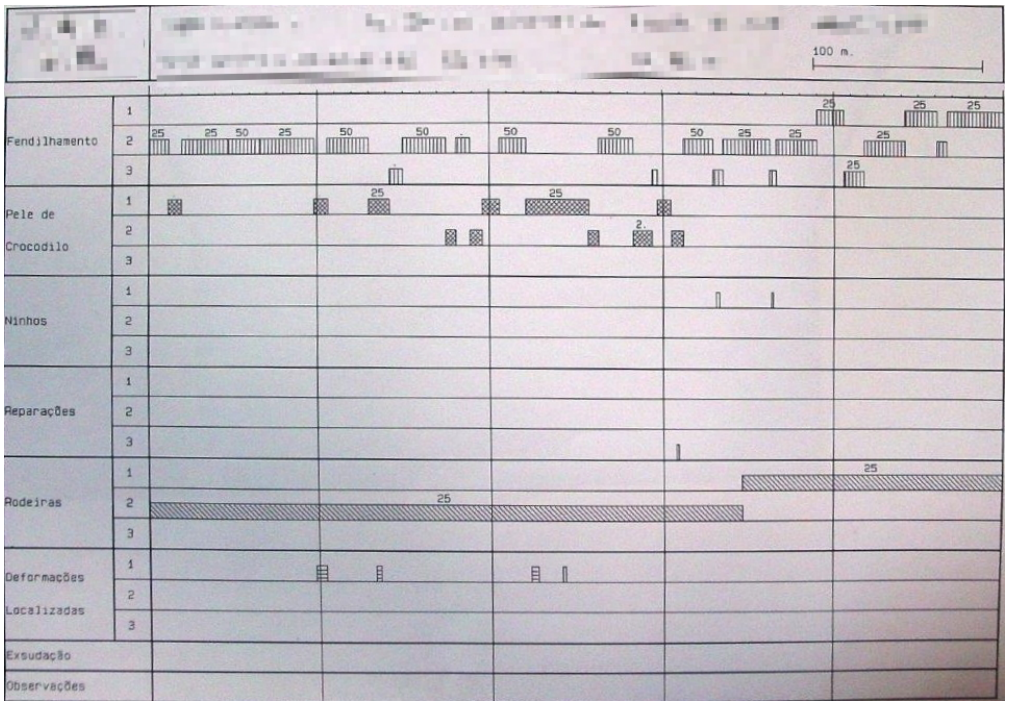


Figure 6.10: Sample of an itinerary map.

As an example, we can see that in the initial 100 meters of the represented collection, the anomalies are as follows:

- **Cracking** (“Fendilhamento”) - Level 2 in almost the entire length, occupying 25% or 50% of the road width;
- **Crocodile Skin** (“Pele de Crocodilo”)- Level 1 interspersed with the previous crack;
- **Rutting** (“Rodeiras”) - Level 2 at 25% of the track width over the entire length;
- **Localized Deformations** (“Deformações Localizadas”)- Level 1 in the final meters of the road extension.

The conventional type of representation shown in the 6.10 shows some limitations since it is static. To improve the user’s perception of the degradations present on the pavement, it is necessary to create a data analysis system. The goal is to allow the interactive visualisation of the road pathologies and their mapping to the real road.

6.8.1 *Requirements*

The requirements that best reflect the needs for building the data visualisation *software* are the following:

1. Choice of the file and directory with the degradation and image data;
2. Representation of anomalies in a graphical way, with their typology, severity, location and possible annotations;
3. Possibility to navigate through the entire road length, with different visualisations scales;

4. Modification of the degradations of interest that are visible in the representation;

5. Presentation of the image of the pavement corresponding to the graphic representation.

6.8.2 *User interface*

For a future user convenience, a graphic interface was developed that meets the requirements intended for the application. A graphical interface is shown in Figure 6.11.

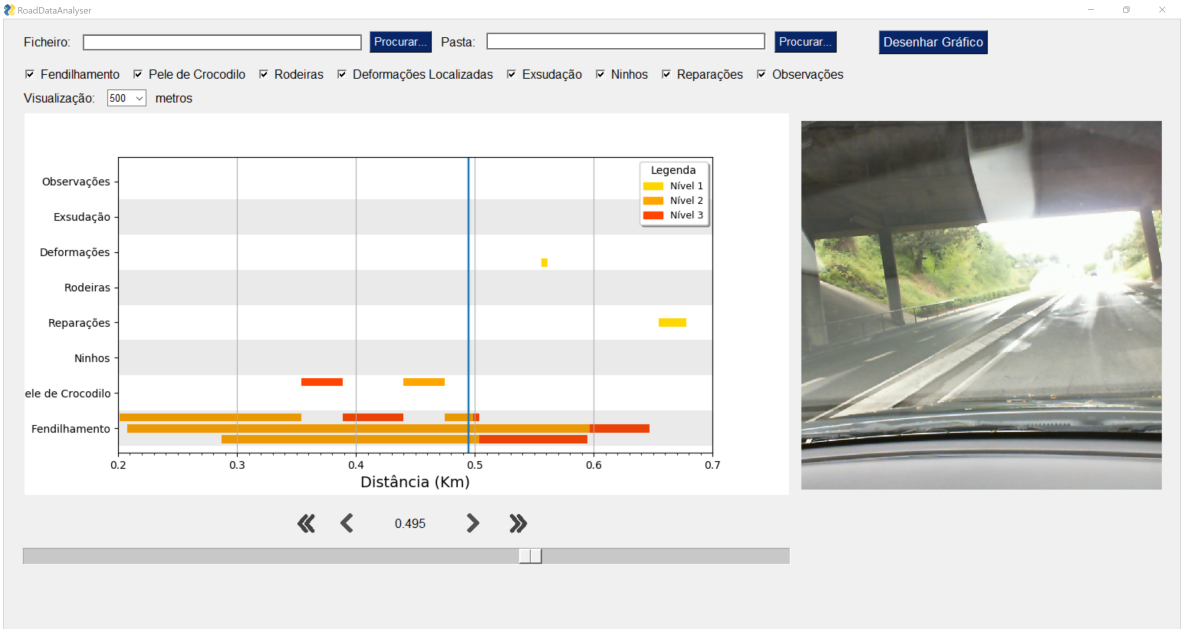


Figure 6.11: GUI of the data analysis software.

For choosing the data file and the folder with the photos, there are two text boxes, where their absolute paths are placed. This input is done using the *Procurar...* (Search) buttons that take the user to the operating system's folder search engine. These components meet first requirement.

When the user presses the *Desenhar Gráfico* (Draw Plot) button, the graph corresponding to the chosen data file and the first image captured in the choice appears. The following characteristics represent each anomaly:

- **Typology** - Displayed on the left side of the plot, on the vertical axis. Each type of degradation is shown horizontally in the graph. The grey/white dividers separate degradations on the visualisation.

- **Location** - It is shown according to its distance from the start of the indication. This distance is shown on the horizontal axis. The length of the bar gives the extent of the anomaly. Each anomaly has 3 bars at the top, centre and bottom, representing the left, centre and right of the strip, respectively. This representation is intuitive in the sense that it gives a sense of perspective of a road seen from above, walking towards the increasing distance.

- **Severity** - Identified with the colours in the label, accordingly to the level of severity presented by the anomaly.

Additionally to anomalies, some notes can be found in the plot, on the *Observations* row. These are represented by arbitrary keyboard characters and have a specific meaning for the operator. The second requirement is satisfied with these features.

To change the graph display parameters, 3 elements are required, which correspond to requirements 3 and 4. The first of the elements is the check-boxes with the degradations that allow managing their visibility in the plot. When activating or deactivating each cheackbox, the corresponding line in the graph appears or disappears, respectively. The second element is the visualisation scale, which can be changed using the *Visualização* (Visualization) drop-down. The possible scales are 100, 500, 1000 and 5000 meters, allowing a more specific or more general view of the section under analysis. The last element is the slider placed at the bottom of the graph, allowing the user to move along the plot, modifying the viewing window and consequently the section of road represented.

Finally, to ensure compliance with the last requirement, an image of the real pavement is displayed, accompanied by a blue line on the graph and the distance this line is, indicating its position on the road. This line cuts the graph allowing to verify which anomalies are associated. To move this line, the single arrows jump one event, while the double arrows jump 10 events.

6.8.3 *The Pavement Dataset*

As presented in Figure 6.9, the captured images in the gathering process don't have enough quality to be used in the data modelling step. The camera is near the laptop inside the vehicle, instead of near the pavement, and its sensitivity to light is not adequate to get the degradations in the pavement.

Consequently, the input data used for training the model is a public dataset of road pavement images Balaji et al. (2019). The set is originally divided into 2 groups:

- *Crack - Non-Crack* with pavements images in good condition;
- *Non-crack - Crack* - with images of cracked pavement.

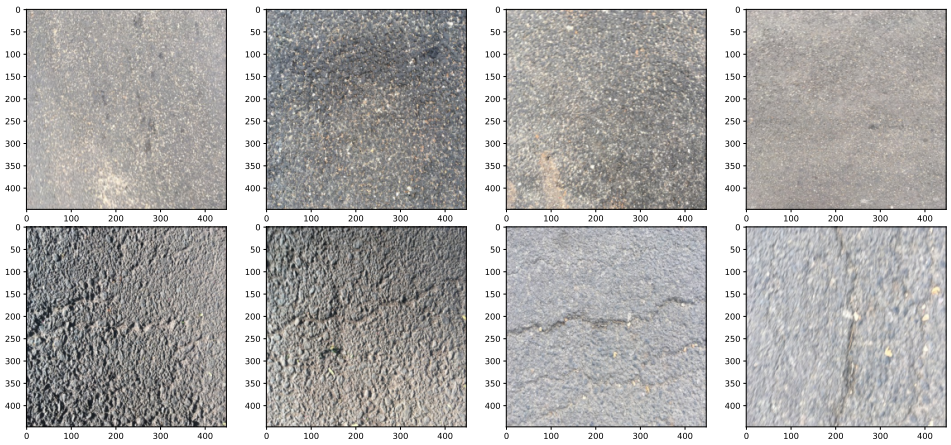


Figure 6.12: Examples of pavement images from the Pavement Crack dataset Balaji et al. (2019): pavement in good condition (top) and degraded pavement (bottom).

The images are homogeneous in terms of measurements, having a size of 448x448 pixels with 3 color channels (RGB). The dataset has a total size of 400 images. There are 200 images in each of the groups, what means that the dataset is balanced.

Examples of the images of the two groups can be seen in Figure 6.12.

To better understand the differences in the features of the input images, a pixel value histogram study can be performed. Figure 6.13 show some picture examples and the respective histograms.

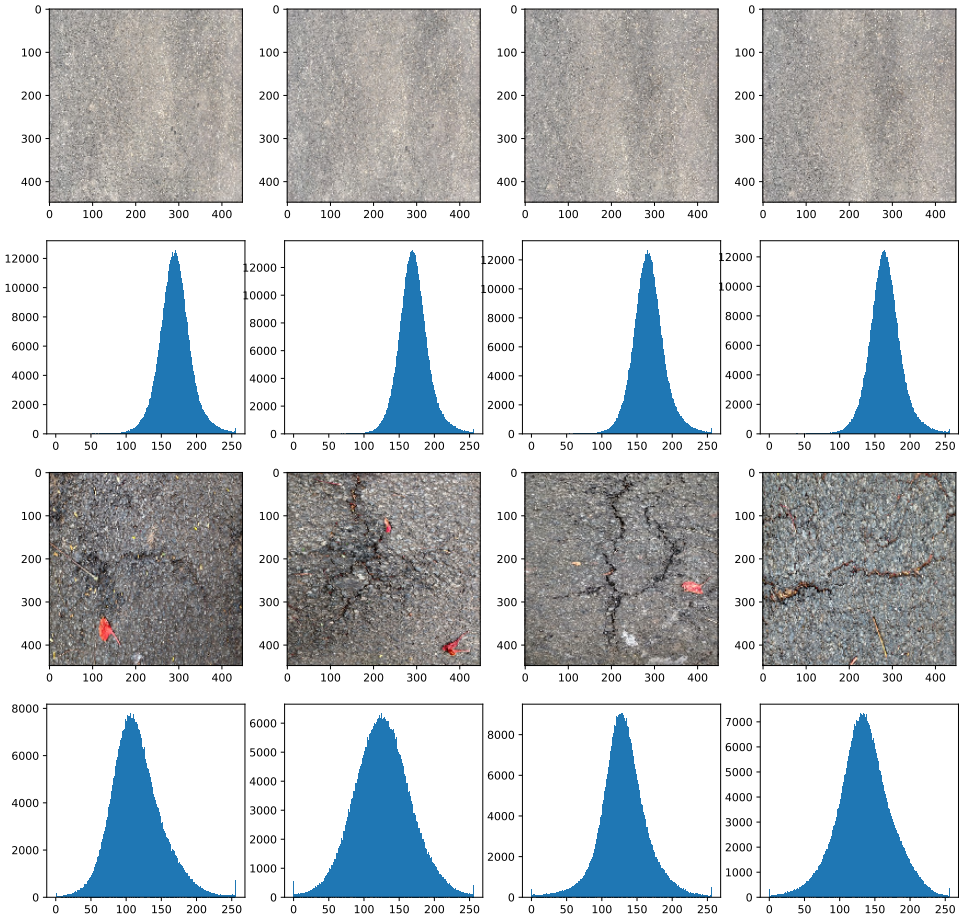


Figure 6.13: Examples of histograms from the "Normal" and "Anomaly" classes (first and third rows, respectively); and respective histograms (second and fourth row, respectively).

The histograms show the distribution of pixels darkness (0-black, 1-white). It helps to define the filters for preprocessing in the next step. From the histograms, it is clear that the "Anomaly" cases present darker values than the "Normal" cases. The distribution of "Normal" cases shows that their values are approx. in the range [100,256], while the "Anomaly" class presents wider values, in the full range [0,256]

6.9 DATA MODELLING

The modelling phase is the principal component in this project. The present section is subdivided to show the decisions made during the construction of each model accordingly to Figure 2.4, where the ML workflow is displayed.

The *preprocessing* and *segregation* steps are presented in sections 6.9.1 and 6.9.2, respectively. The *feature selection* step is omitted since the input data are images and it is made inside the networks automatically.

The phases in the model axis and the result analysis are grouped and presented in a section for each created model.

6.9.1 *Data Preprocessing*

As the pavement is very heterogeneous when closely observed, the patterns found on it are difficult to find through a neural network. Due to the complexity of the problem, the images need to be preprocessed in order for the model to be able to recognize the patterns of the anomalies.

The applied transformations need to have in account the real world and the observations that the human eye can perceive. As a consequence, a particular assumption based on the previous analysis of the images is assumed: "*The degradations present darker pixel values than the rest of the picture.*".

In this way, the treatment of the images was carried out so that the most relevant aspects for detecting the degradations were highlighted. After trying several filters, the ones that have the best results are the following:

- Loading of black and white images and resizing to 256x256, discarding color information, which is irrelevant in this case;
- Use of the *bilateralFilter* filter from the *OpenCV* library (Bradski, 2000), since it blurs the image (*blurring*) preserving its contours. This filter is effective insofar as the degradations are evidenced at the same time that the pixels corresponding to the asphalt are blurred, removing patterns that could create confusion in the training of the network (Tomasi and Manduchi, 1998).
- Use of the *threshold* filter in the *OpenCV* library (Bradski, 2000), truncating the value of certain pixels. In this case, the darkest values are truncated for the same color (black), since anomalies in general have a darker color than the rest of the asphalt, due to differences in light incidence.

The 6.14 figure shows sample images before and after pre-processing.

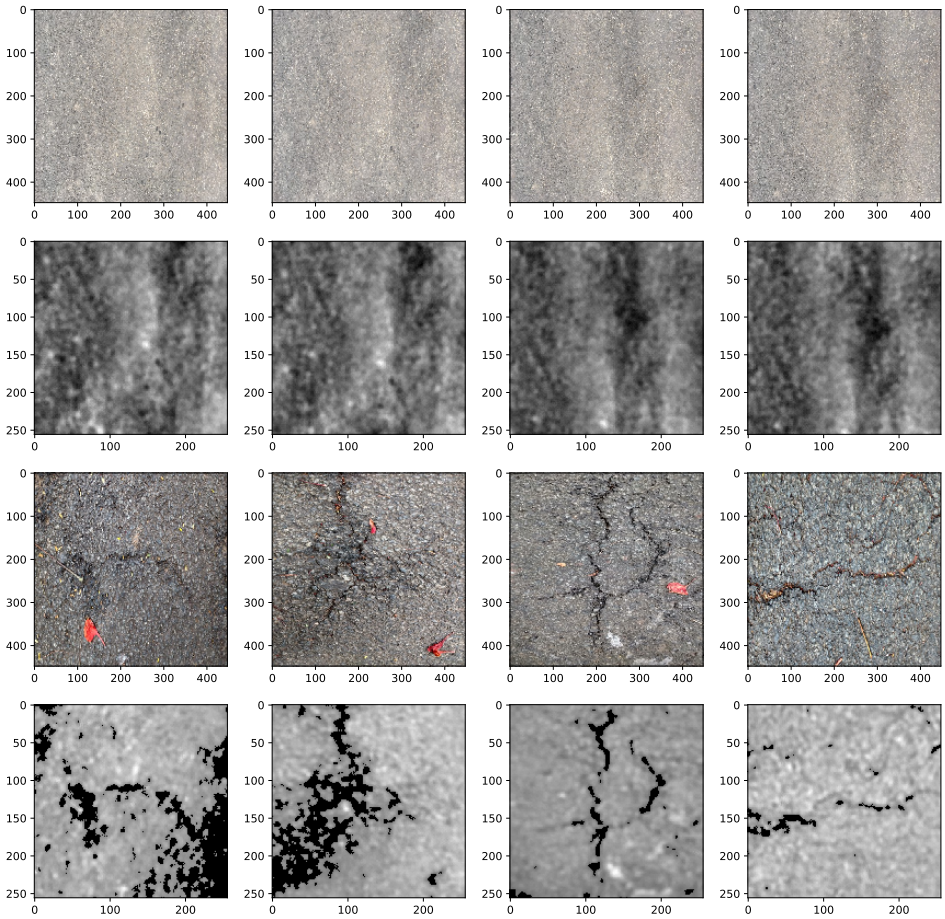


Figure 6.14: Examples of the preprocessing step. The first and third rows show the "Normal" and "Anomaly" original images, respectively; the second and fourth rows show the same images after being processed.

This treatment aims to facilitate the work of the neuronal network in learning the patterns, while minimizing the computational capacity required, since the number of operations to be performed is proportional to the size of the image and the number of color channels that is reduced from 3 (RGB) to 1 (*gray-scale*). Note that in the processed images, the degradations stand out clearly.

6.9.2 *Data Segregation*

To start the training process, it is necessary to divide the dataset into *training* and *testing*. The *training* data is used by the model as a way to learning the patterns found in the set of images. The *testing* data is an indicator of the model's performance. Without it, there would be a possibility that overfitting would not be detected, that is,

the neural network could adapt to specific patterns of the *training* set, but behave inconsistently when exposed to new pavement images.

In addition to this division, the training data is subdivided in order to integrate a set of *validation* data that is used to monitor the training process. The data is segregated in different ways depending on the model that is intended to train. For the self-supervised models, an additional division needs to be made. In these models, the reconstructions metrics are used to split the classes, using a calculated threshold. To perform an unbiased model evaluation, different images has to be used in the threshold calculation step and the model testing. The split percentages are the same used for the preliminary case study (see Chapter 5).

6.9.3 *Classifier*

The classifier model is a CNN, where the inputs are images with dimensions 256x256x1 (width x height x color channels). The output variable is a value between 0 and 1 that will represent one of the two classes, with the division point of the classes being 0.5.

The architecture is composed by 4 *Conv2D* layers (16,32,64 and 64 3x3 filters), interspersed by 3 *MaxPooling* layers (2x2). Comparing to the experimental case study, the representational power of the network is increased, since the images to process are more complex.

All the layers but the last use the ReLU activation function. The last one uses the *sigmoid* function to give the result in the desired domain - [0,1].

As it is visible, the information inside the network is being processed in such a way that the dimensions of the image decreases. On the other hand, the number of filters applied is increasing. Therefore, the information is being processed hierarchically, with more specific areas of the image being processed.

The last three layers presented, do not do image processing and refer to the classifier itself, which is composed by densely connected layers. The first one is a *Flatten* layer that returns a 1D tensor, the second is a *Dense* layer with 64 nodes and the last one is the output layer composed by a single *Dense* layer as well.

The configuration of the model building is achieved by using the *binary_crossentropy* loss function and the *rmsprop* optimizer. The metric to follow during the training process is the accuracy, since it expected to maximize this metric while the loss function decrease.

The training process is finished after 50 epochs using the training dataset to update the weights and the validation dataset to supervise the model evolution. The use of the *EarlyStopping* callback can be useful to automatically finish the training process when there are no significant results.

The results are presented in Figure 6.15. The network can distinguish the good pavement cases very well, misclassifying only one “Normal” case and presenting a good confidence in all the others, since the predictions are close to 1.0. On the other hand, it has some difficulties to classify a few instances of degraded pavement with some values near the 0.5 cut-point.

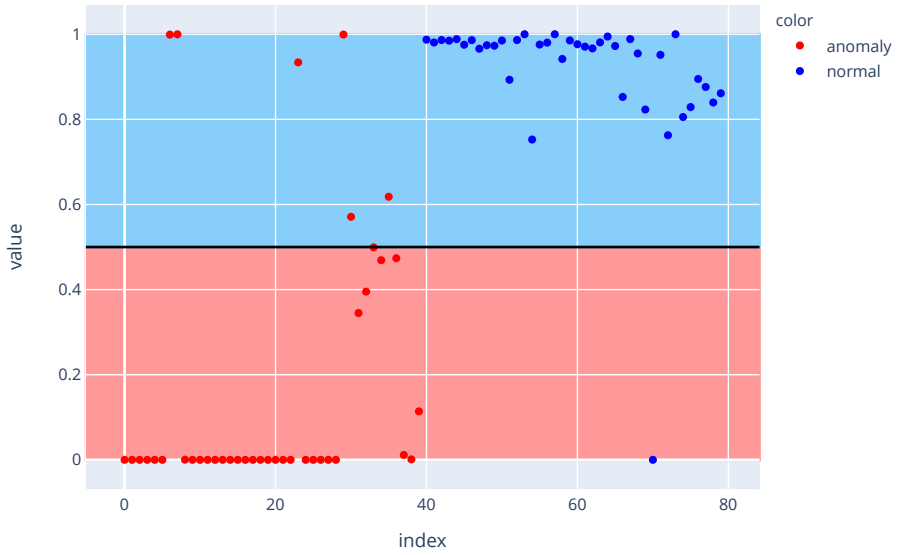


Figure 6.15: Pavement Classifier - Raw predictions.

6.9.4 *Autoencoder*

As the strategy presented in the preliminary case study (see Chapter 5), the autoencoder model reconstructs the good pavement images and it is expected that images representing bad conditions are poorly reproduced.

As expected, although similar, the created model is more complex than the MNIST AE.

The same structure is maintained, modifying the layers, increasing the representational power:

1. Encoder - 3 *Conv2D* (16, 64 and 32 3x3 filters) + *MaxPooling2D* layers (2x2 window);
2. Bottleneck - 2 *Dense* layers (40 and 1024 neurons);
3. Decoder - 3 *Conv2DTranspose* (all with 64 3x3 filters) intercalated by 2 *BatchNormalization* layers.

The *Conv2D* layer is the output of the model that returns an image comparable to the input.

In this case, the encoder performs better with an additional convolutional layer. The decoder has the same number of layers as the initial MNIST model, however they are configured in a different way. in every *Conv2DTranspose* layer are used strides of 2 to duplicate 3 times the initial size of 32x32 pixels into the original size ($(32 \times 32) \cdot 2^3 = 256 \times 256$). The activation functions used in all layers is the ReLU. The output layer uses the *sigmoid* function to retrieve the pixel darkness.

Figure 6.16 show examples of reconstructions using the AE network. The network is capable of reconstruct with a some degree of similarity, although it is visible its difficulty to present a similar behaviour when some degradation is present in the image.

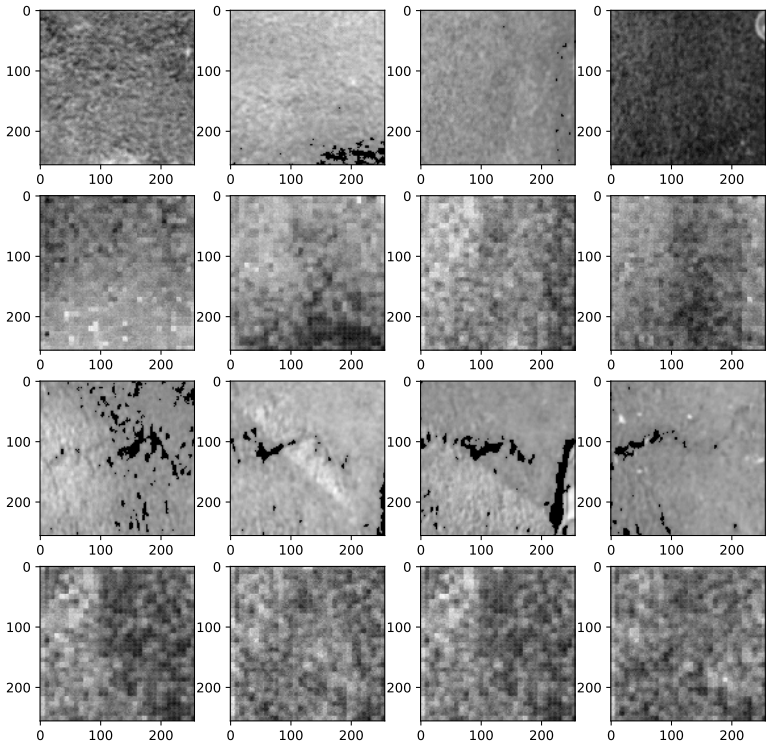
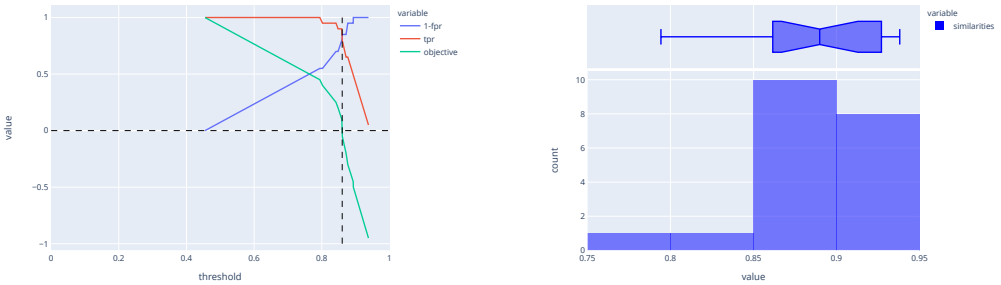


Figure 6.16: Pavement AE - Reconstruction examples. First and third rows are examples of the original images for “Normal” and “Anomaly”classes, respectively. Second and fourth rows are the respective reconstructions.

From the reconstructions of the pavement images, the SSIM metric is used to calculate the likeness between them and the original input.



(a) Supervised calculation method.

(b) Unsupervised calculation method.

Figure 6.17: Pavement AE - threshold calculation methods.

Figure C.5 presents the predictions for both supervised and unsupervised approaches based on the calculated similarities. Using the unsupervised threshold calculation, that can be used in situations where the majority of the data are good pavement images, the results are not so good as the presented in the supervised method.

6.9.5 *Variational Autoencoder*

The base VAE model built in the experimental use case predictions is adjusted to address the pavement case problem. In this case, maintaining the encoder structure with 5 stacked sets of *Conv2D*, *BatchNormalization* and *LeakyReLu* layers, the model capacity increases substantially. Since the input images dimensions are considerably larger than the ones presented in the MNIST dataset, their intermediate representations and the tensor after the *Flatten* layer are bigger as well. Comparing the relative dimensions, the difference in the image sides is approximately 9 times (28×28 to 256×256), what is reflected in the number of pixels to be processed by the input layer, that increases almost 84 times (784 to $65,536$). The full encoder, including the two *Dense* layers to represent the mean and variance of the latent distribution increases the learning parameters from around 100,000 to more than 1,100,00. The output of the encoder is acquired using the sampling function previous defined.

The decoder part follows the same behaviour of the experimental case study. Since there are no *Dense* layers in this part of the network, the parameters increasing is not so noticeable.

The model reconstructions are presented in the Figure 6.18. Compared to the AE model, it presents a finer grain in the output images. Also in the “Anomaly” case, the model cannot reconstruct the degradations, outputting random patterns in some cases.

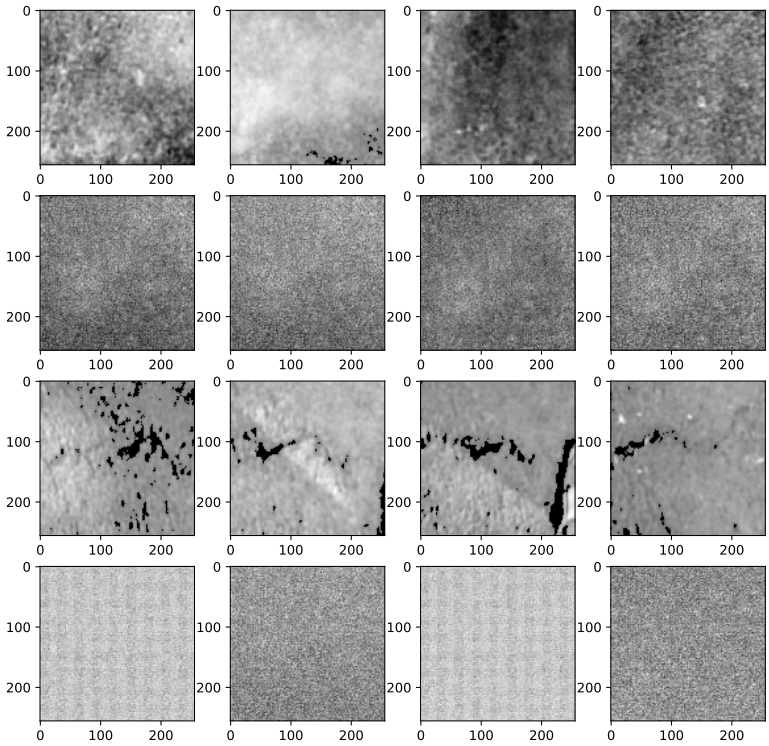
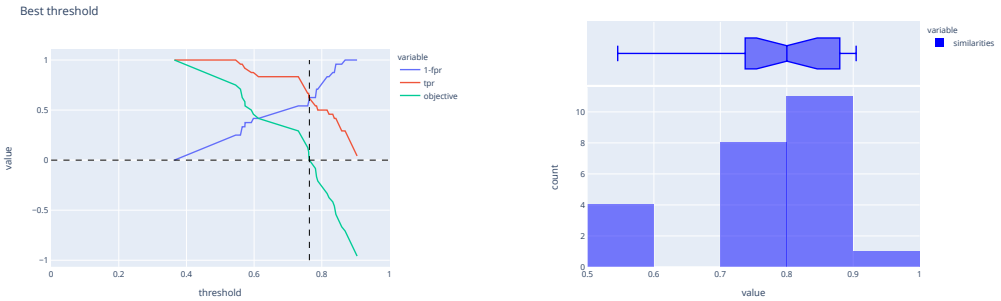


Figure 6.18: Pavement VAE - Reconstruction examples. First and third rows are examples of the original images for “Normal” and “Anomaly”classes, respectively. Second and fourth rows are the respective reconstructions.

Using the supervised approach, the best threshold is 0.76, when TPR=0.63 and FPR=0.38. The unsupervised method returns a much lower threshold of 0.54. Figure 6.19 presents both threshold calculation methods.



(a) Supervised calculation method.

(b) Unsupervised calculation method.

Figure 6.19: Pavement VAE - threshold calculation methods.

The predictions for both methods based on the SSIM metric are presented in Figure C.6. Using the supervised threshold, it is possible to correctly predict all the test cases. The unsupervised threshold is not so accurate in

the classes division. Note that what makes the unsupervised method perform worse are the four similarity values between 0.5 and 0.6, that are clearly out of the rest of the distribution, but are not classified as outliers due to the small dataset size.

6.10 RESULTS ANALYSIS

6.10.1 *Classifier*

The AUROC presented in the figure 6.20, with an AUC of 90%, gives the model a very good separation capacity. With the default 0.5 split value, the confusion matrix shows good results in the test set.

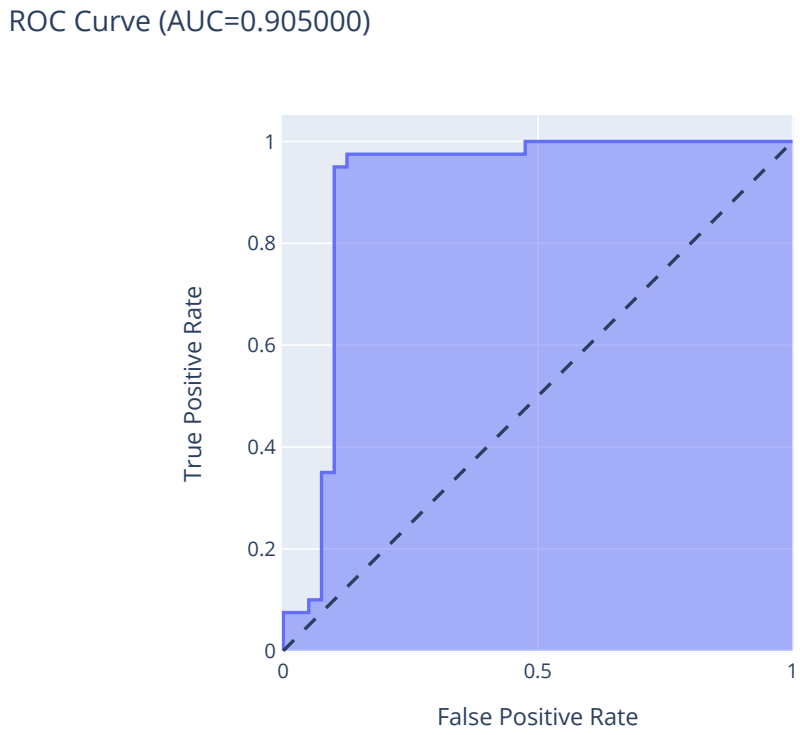


Figure 6.20: Pavement Classifier - AUROC.

As the Table 6.1 shows, in relation to the class “Anomaly”, 85% of the real cases of degradation are identified by the model. In turn, the model guarantees with a 97% success rate that a case predicted to be anomalous is in fact corresponding to a pavement image in poor condition.

Table 6.1: Pavement classifier - Classification report.

	Precision	Recall	support
anomaly	0.97	0.85	40
normal	0.87	0.97	40
Accuracy	0.91		80

The prediction capacity of the model in the two classes presents an accuracy of 91%, which means that in the vast majority of cases the image of the pavement will be inserted in the correct class.

6.10.2 *Autoencoder*

Based on the similarities presented in Section 6.9, the confusion matrices can be created, as presented in Figure B.5.

The results of this model using both classes to calculate the threshold present an improvement in the anomaly recall relatively to the classifier model from 85% to 94%. The full results are presented in the classification report in Table 6.2.

Table 6.2: Pavement AE - Classification reports.

	Supervised		Unsupervised		
	Precision	**Recall**	**Precision**	**Recall**	**support**
anomaly	0.94	0.94	1.00	0.62	16
normal	0.94	0.94	0.73	1.00	16
Accuracy	0.94		0.81		32

Figure 6.21 presents the AUROC metric, which is 99%, representing a very good model capacity to split both classes.

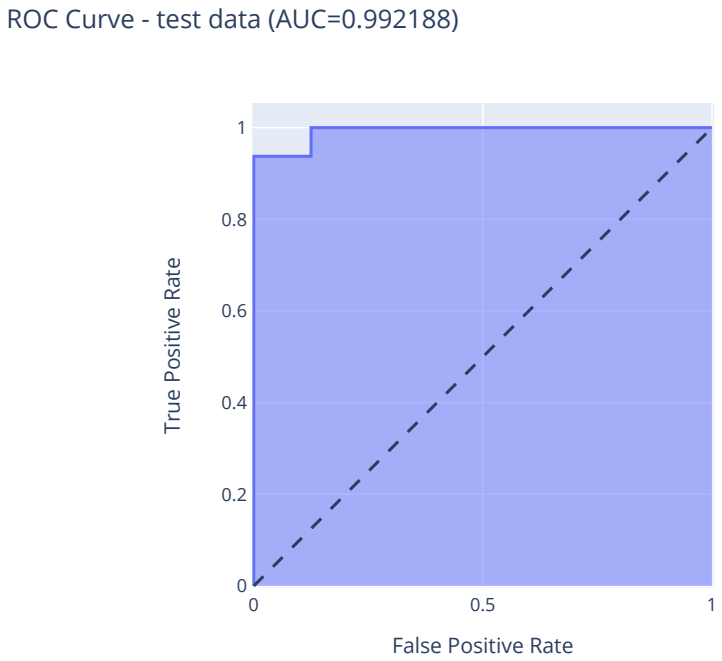


Figure 6.21: Pavement AE - AUROC.

6.10.3 *Variational Autoencoder*

According to the results shown in Figure C.6 the confusion matrices are calculated. The supervised method presents a perfect class distinction with all the instances predicted correctly. The unsupervised method tends to classify the majority of the cases as normal, misclassifying a considerable part of pavement degradations.

As expected, using the test data, the AUROC metric presents an AUC of 1.0 (Figure 6.22).

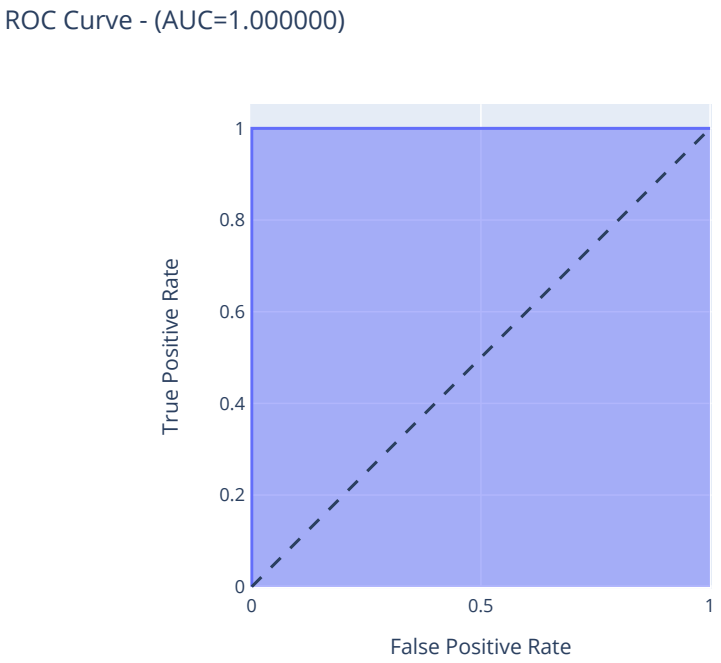


Figure 6.22: Pavement VAE - AUROC.

According to the results of the unsupervised method presented in Table 6.3, the recall for the normal class and the anomaly class precision are 1.0, since all the good pavement images are correctly classified and all the images classified as anomalous represent in fact degraded pavement.

Table 6.3: Pavement VAE - Classification reports.

	Supervised		**Unsupervised**		
	Precision	**Recall**	**Precision**	**Recall**	**support**
anomaly	1.00	1.00	1.00	0.44	16
normal	1.00	1.00	0.64	1.00	16
Accuracy	1.00		0.72		32

6.11 PROBLEM SOLUTION

Using the proposed solutions proposed in the present chapter, it is possible to identify a full problem solution prototype to automatically monitor the pavement. The data gathering software, used in a first moment to acquire

data to train the models (image and eventually annotations), is then used only to get the pavement images that will be classified by the selected trained model. The data visualization software is used as an interactive report, presenting the road conditions returned by the model to the users. Ideally, the data analysis software would be the unique point of contact with the final user, since the data acquisition and pavement classification would be hidden and performed automatically.

6.12 DISCUSSION

The obtained results with the ANNs are good in the scientific context that they are presented. However the path to get to the final results was not linear, as it may be incorrectly inferred from the lack of bad results in the document structure. On total, more than 250 experimental models were trained, using different architecture and building decisions, as layers, activation functions, loss functions and optimizers. On average, the models training time was around 15 to 20 minutes, with the greatest registered training time to be more than 2 hours. The hardware settings used during the process were the following: *Intel® Core™ i7-7700HQ CPU @ 2.80GHz* processor; *NVIDIA® GTX 1050* graphic card; 16GB of RAM. Also to choose the cut-points different methods were tried, using different objective functions and metrics.

With all the training events in mind, some discussion questions can be appointed:

1. *Why are some of the training times so high?* The highest training times were observed in the first iterations of modelling. At that point, the graphic card was not being used and the images were fed to the network without preprocessing, what made it to use the full image size (448 x 448), tripling also the color channels (RGB images).

2. *The images reconstructions in the pavement case don't seem to be very similar. How are the results so good?* The important in reconstructions for AD is not the absolute quality of the reconstructions, but the difference between both classes. The bigger the difference, the better. Even if the reconstructions are not very similar when looking at the images, the important pixel level similarities are captured by the model, being also reflected when calculating the SSIM metric between the original and reconstructed images.

3. *What were the criteria to select the image preprocessing filters?* Initially, a group of filters were selected to preserve the main characteristics of the pavement and the degradations. From this group, using the classifier model with a fixed architecture, multiple tests were performed varying only the different filters combinations and parameters. The select filters are the ones that lead to the best results.

4. *With so high AUROC values for the AE models, why some of the results are worst than the classifier, that presents a lower AUROC?* The worst results are presented mainly in the unsupervised setting. This is an evidence that there is room to improve the threshold calculation methods, specially the unsupervised methodology. One of the main causes for this, is the lack of data. With more images to calculate the threshold, it is expected to achieve better results.

5. *How is it possible for the VAE model to get perfect results? Are they cherry-pick?* The results of the VAE model are indeed really good, but they are not cherry-pick.The scores are obtained with the automatic random selection from the original dataset. However, it is important to note that the support for all the

trained models is relatively small and there may be fluctuations on the scores when dealing with bigger test datasets.

6. *There are other available datasets containing pavement images. Why aren't they used to increase the support?* The selected dataset provides the images in similar settings of the real use case relative to the preferred camera positioning. Other datasets, even though with more images, show different perspectives of the pavement, that are not suitable for this case.

Relatively to the last two presented cases, similar iterative approaches have been used, with lower training times (maximum of approx. 5 minutes), since the numerical problems are much lighter in computational terms than using images.

6.13 CONCLUSIONS

The main goal of the case study presented in this chapter is to demonstrate the viability of a low-cost, fully automatic pavement monitoring system, using image data to detect degradations.

As proved in the previous sections, it is possible to build software that acquires image data with low/no human interaction, using budget cameras. This program can evolve to be integrated into an embedded system, that acquires data automatically when the vehicle starts to move.

Regarding the ANN models, it is also proved that any of the models are liable to be used, depending on the context and on the importance given to each metric. In Table 6.4 are shown the overall results in order to compare the different models and approaches, from a degradation detection perspective (recall and precision for the anomaly class).

Table 6.4: Pavement models comparison by metric.

	Classifier	**AE**		**VAE**	
	Supervised	**Supervised**	**Unsupervised**	**Supervised**	**Unsupervised**
AUROC	0.91	0.99		1.00	
Accuracy	0.91	0.94	0.81	1.00	0.72
Precision	0.97	0.94	1.00	1.00	1.00
Recall	0.85	0.94	0.62	1.00	0.44

Looking only for the capacity to distance the classes from each other, that is independent from the threshold calculation method, the VAE and the AE present better results than the classifier, as the AUROC metric presents.

The model that performs better in a generic view is the VAE architecture using the supervised threshold calculation method, which presents a perfect distinction between both classes. This is the best option to take when all the possibilities are available, namely the possibility to have training data for both classes. In this setting, the AE also performs well than the classifier if the selection criteria is the model accuracy or recall. If the model precision is most important, the classifier must be chosen over the AE.

When looking for scenarios where there are no labelled data available, for example, when the data is acquired from a road that is known to be in good conditions *à priori*, the models to be used are the AE and VAE. In such unsupervised scenario, the VAE performs equal to the AE only in terms of precision, getting worse results in all the others metrics. The best model in this case in a overall view would be the AE.

Finally, to wrap the information and present it to the user, an interactive software is used, improving the traditional approach of multiple paper sheets with roads information. The full end-to-end solution represents an improvement over the traditional methods, removing the need to make visual foot-on-ground surveys; and over the commercial solutions that use costly systems to acquire the data. The costs involved in the present solution are mainly the price of budget cameras.

Note that the presented methodologies are a first approach to solve the problem and do not represent a fully functional solution. They can be further explored to acquire even better results.

7

ADDITIONAL APPLICATIONS

> *Although this may seem a paradox, all exact science is based on the idea of approximation. If a man tells you he knows a thing exactly, then you can be safe in inferring that you are speaking to an inexact man.*
>
> Bertrand Russell

7.1 INTRODUCTION

Connectionist systems have a variety of applications beyond their utilization to detect anomalies in image sets. DNNs are often used to approximate real values as the target variable in regression problems based on different input features. In this chapter are presented two problems that need this type of predictions.

The first case, presented in section 7.2, uses ANNs to estimate the stress and displacement fields of a plate subjected to pressure. In section 7.3 is shown the second additional case study, regarding the detection of crowds using crowd-sensing, more specifically Wi-fi Access Points (APs) data.

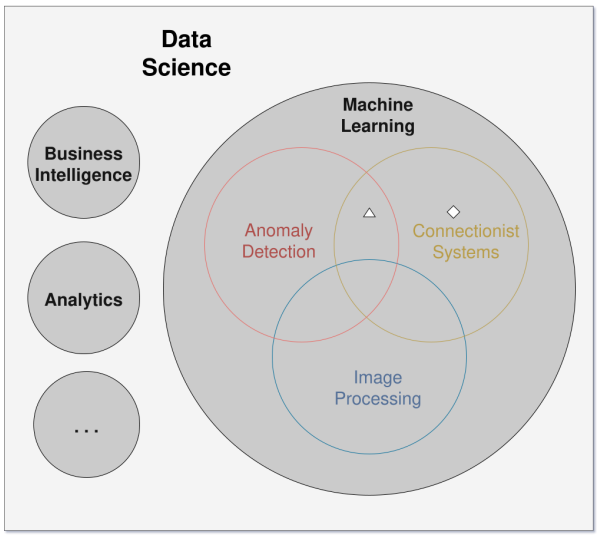


Figure 7.1: Research subjects Venn diagram. The diamond and the triangle represent the *Mechanical Structural Design* and *crowds detection* case studies, respectively.

7.2 MECHANICAL STRUCTURAL DESIGN

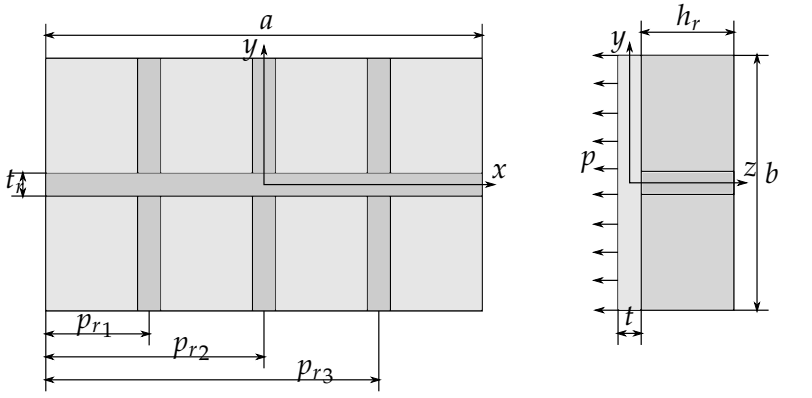
The finite element method allows to carry out mechanical design, for different conditions, obtaining results very close to the real ones (Zienkiewicz et al., 2013). However, when it is intended to simulate complex structures, the simulation time becomes excessively high and a set of computational resources are required. Despite the advancement of technology, with increasingly faster processors, the problem still remains. In this way, new techniques have been used to carry out the mechanical design (Mohamed, 2018).

One such technique is DL algorithms, namely DNNs. Through these methods, it is possible to significantly reduce computational costs, obtaining results with minor approximation errors. A fully detailed analysis of this practical use case can be found in Ribeiro et al. (2021a). Also an in-depth comprehension of the mechanical structural design and other reduced order models for that use case are explored in Ribeiro (2020) and Ribeiro et al. (2021b).

Problem Definition

The goal of this study is to apply ANN to design a reinforced structure, as can be seen in Figure 7.2. Reinforced structures are formed by a plate with reinforcements in the horizontal and vertical direction. These structures make it possible to reduce the plate thickness, for a given solicitation, which reduces the amount of material and thus the associated production cost.

Similarly to the approach used to solve the pavement classification problem, where the MNIST dataset is used as a preliminary study to build simpler models, three situations were established in the present scenario, increasing the difficulty up to the third situation, the case under analysis. The first problem corresponds to a plate with a central hole, a classic case of mechanics; the second to a fixed plate; and the last case to the reinforced



(a) Front view (b) Side view

Figure 7.2: Vertically and horizontally reinforced panel subjected to uniformly distributed pressure.

plate. For each case, it is necessary to take into account the material properties, the geometric properties and and the applied load (Timoshenko and Goodier, 1951; Ugural, 2020).

Note that the pressure applied to the plate, the material properties and the geometric properties of the plate are constant. Only the influence of reinforcements is being studied. For the case study, it is necessary to define the following variables:

- *Reinforcement thickness (t_r)* - Thickness of vertical and horizontal reinforcements;
- *Reinforcement height (h_r)* - Height of vertical and horizontal reinforcements;
- *Number of vertical reinforcements (n_r)* - Number of vertical reinforcements, the number of horizontal reinforcements has a constant value equal to one;
- *x coordinate (x)* - Defines the x position, where the fields will be calculated;
- *y coordinate (y)* - Defines the y position, where the fields will be calculated;
- *Displacement (w)* - Displacement value for a given point;
- *Von Mises stress (σ_{vmis})* - Von Mises stress for a given point;

Data Gathering

The data is acquired using finite element software, namely Ansys®. Where for each of the three cases, numerical simulations were carried out to obtain the stress and displacement field.

In Figure 7.3, it is possible to observe an example of the geometry implemented in the Ansys® software to obtain the stress and displacement fields.

Data Storage

The data are stored in CSV files, which are used later to train the models. In Figure 7.4 a part of an example of a CSV file is visible. The first two columns represent respectively the x and y coordinates of the point where

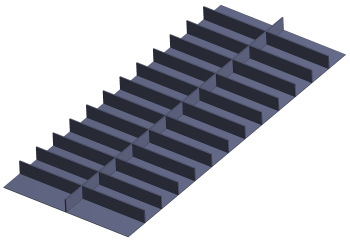


Figure 7.3: Example of implementation of the reinforced panel in Ansys® software.

the displacement perpendicular to the plate and the von Mises stress are calculated, represented in the last two columns respectively.

4204.000000	1169.062500	16.351171	67.059189
4204.000000	1091.125000	15.959534	74.610071
4204.000000	1013.187500	15.337763	91.969140
4204.000000	935.250000	14.499136	106.983175
4204.000000	857.312500	13.463440	121.920209
4204.000000	779.375000	12.255676	137.145962
4204.000000	701.437500	10.905736	153.236736
4204.000000	623.500000	9.447763	171.537507

Figure 7.4: Example of implementation of the reinforced panel in Ansys® software.

Data Cleaning

The data obtained from the simulations are assumed to be the ground-truth for the models. Since the data acquisition is made in a controlled environment, there are no errors on the stored information. Therefore, the data cleaning step can be ignored in this case.

Data Analysis

With the results obtained from the various simulations, it is possible to obtain the box-plot diagrams for each variable under study, as shown in Figure 7.5. As it is visible, the coordinates variables have big values ranges when compared to the remaining variables.

Data Modelling

Two neural networks were constructed to obtain the stress and displacement fields, respectively. It is necessary to normalize the input data, since as shown in Figure 7.5, the input variables have different orders of magnitude. Thus, the values of the variables x and y are divided by 100 and the values of the variable h_r by 10.

Then, these data are divided into two sets, training data and test data. In both cases, 80% of the data is defined as the training data and the rest the test data. Additionally, 20% of the training data is divided and used as validation set.

The neural networks are made up of *Dense* layers and an output layer that allows obtaining a single continuous value. With the exception of the last one, all other layers use the ReLU activation function.

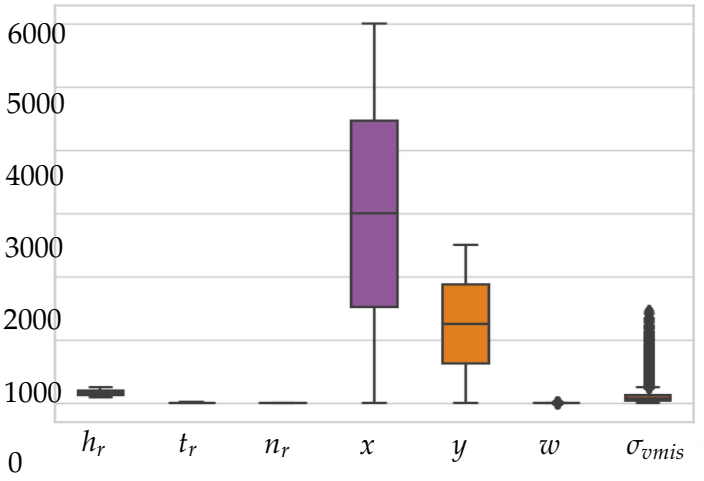


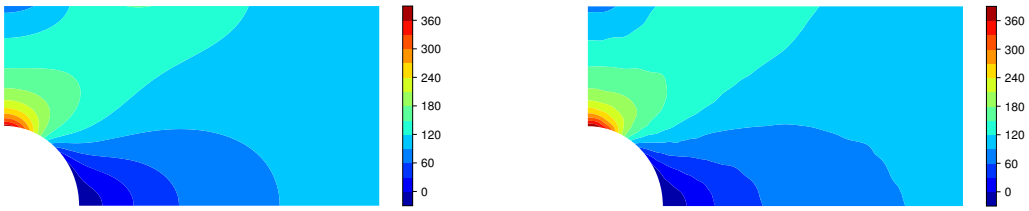
Figure 7.5: Box-plot diagrams for each of the variables.

In addition, it is necessary to build both models. For this, the optimizer *rmsprop* was chosen and the following metrics are used to monitor the model during the training process: MAE, MSE and MAPE. The loss function is MAE.

It is defined that the maximum number of epochs is 200 and 1000, for the network that provides w and σ_{vmis}, respectively. Regarding the *batch_size*, it was defined that it would be 32 and 256 for the network that provides w and σ_{vmis}, respectively.

Problem Solution

In figures 7.6, 7.7 and 7.8 it is possible to observe the results obtained from the ANNs and the results of the simulations in Ansys® for the three applications. As it can be seen, regardless of whether it is a stress or displacement field, the results obtained with DNNs are very close to those of the simulations. The biggest differences are found at the limits of the regions.

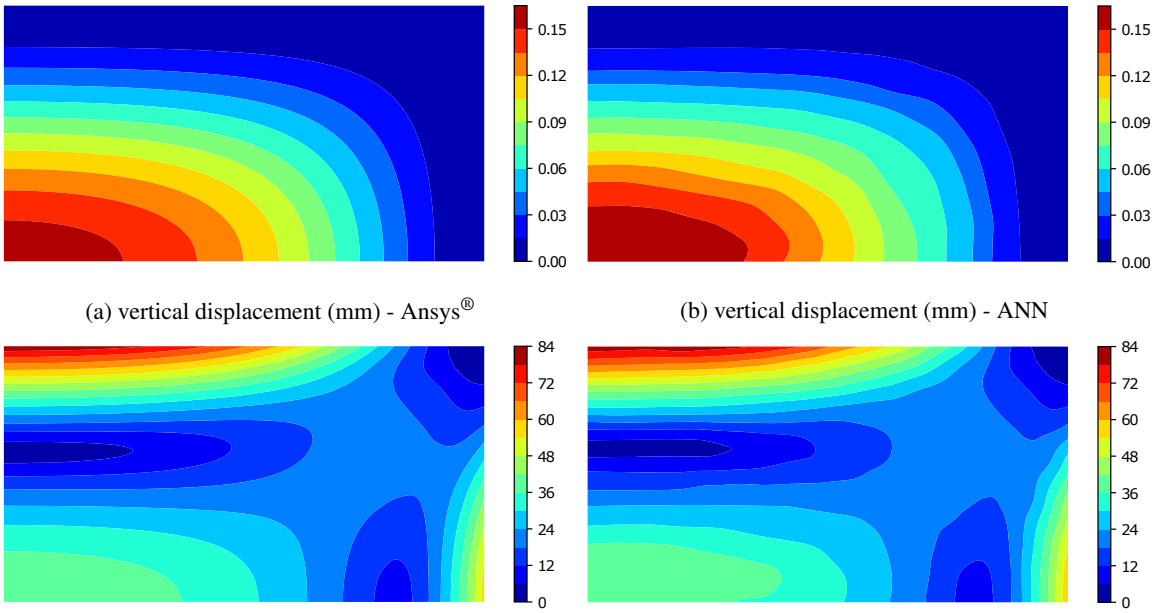


(a) von Mises stress (MPa) - Ansys®

(b) von Mises stress (MPa) - ANN

Figure 7.6: Comparison of the von Mises stress fields - plate with central hole (top-right quarter section).

In addition, ANNs are faster than simulations. For the case in study, ANNs were 40 times faster than the software. The forecast error is acceptable considering the complexity of the problem, presenting an average percentage relative error of 11.3% for displacement field, and 26.8% for stress field.



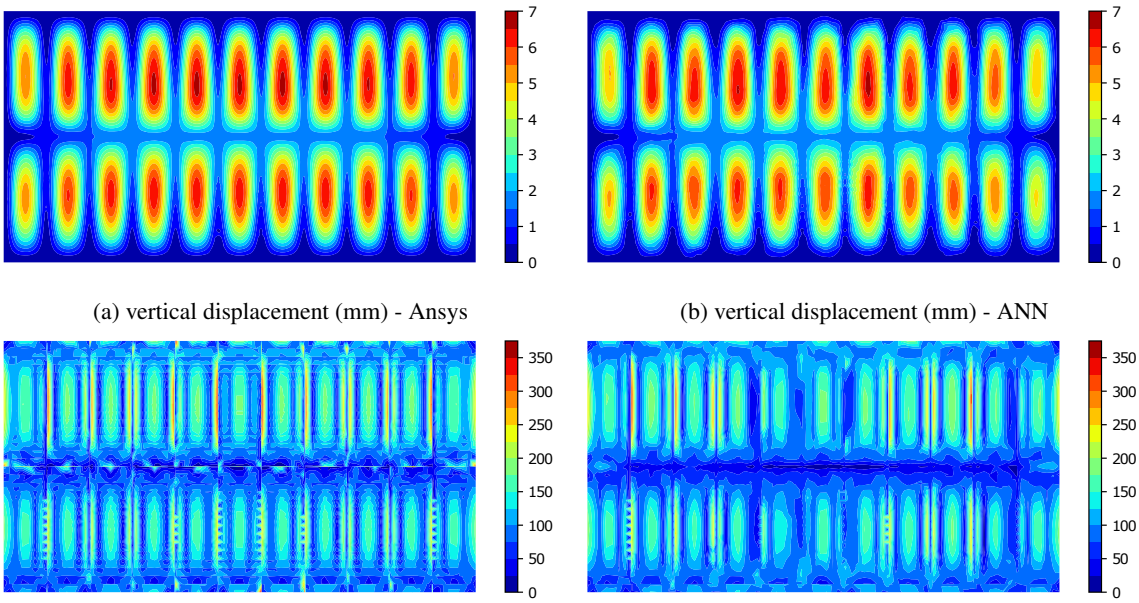
(a) vertical displacement (mm) - Ansys®

(b) vertical displacement (mm) - ANN

(c) von Mises stress (MPa) - Ansys®

(d) von Mises stress (MPa) - ANN

Figure 7.7: Comparison of the displacement and the von Mises stress fields - fixed panel (top-right quarter section).



(a) vertical displacement (mm) - Ansys

(b) vertical displacement (mm) - ANN

(c) von Mises stress (MPa)- Ansys

(d) von Mises stress (MPa) - ANN

Figure 7.8: Comparison of the displacement and the von Mises stress fields - reinforced panel.

7.3 CROWDS DETECTION

During the COVID-19 pandemic, one of the most accurate practices to reduce the risk of infection and the spread of the virus is to maintain physical distance from people. In crowded places, the probability of getting too close to others is high, thus increasing the possibility to get infected.

The *MARÉ-Project* objective is to mitigate the pandemic consequences, helping the people to follow their lives as normal as possible. Using APs data about the connected devices and ML algorithms, it is possible to detect the crowded places. This information is given by a mobile app to users, that can safely decide the best places to go and the ones to avoid.

Problem Definition

The main problem to be solved is the detection of crowded places, warning the users with this information. An event represents a temporal snapshot of the monitored places. The most important variables for each event that are useful to characterize the problem are the following:

- *Devices* - Quantitative variable representing the number of devices connected devices to a given AP;
- *StrengthAvg* - Quantitative variable representing the signal strength average of connected devices to a given AP;
- *People* - Quantitative variable representing the ground-truth count of people;
- *Place* - Qualitative variable representing each place (for example, the building of the event)
- *Timestamp* - Quantitative variables representing temporal information (month, hour, day of week, etc.)

Data Gathering

The data is acquired periodically using the Internet Control Message Protocol (ICMP) protocol to get the number of devices connected to each AP. Additionally, some more data is acquired, as a list with the signal strength for each connected device. There is not collected any information related to the device itself, preserving the anonymity and respecting the General Data Protection Regulation (GDPR).

The ground-truth is acquired in place, using visual counting estimates of the people for each hour.

Data Storage

The acquired data is stored using a *MongoDB* database with different collections, containing the information about the events and the APs metadata. The stored data is used both to train the ML models and to inform the user about the events in the last weeks for each specific location.

The ground-truth data is saved in CSV files, which are then used to train the models.

Data Cleaning

Before the data is used in the modelling phase, it is processed to extract only clean information from it. An example of the data after cleaning is displayed in Figure 7.9. Examples in this stage include:

- Unwind the events timestamps to get more temporal information (for example hour, day of week, etc.)
- Remove events with gathering errors (for example null values);
- Group the events information by place and time (for example, by building and hour);
- Discard useless data (for example, remove data from night time, since there are no crowds to train the models - the device counts are all near 0).

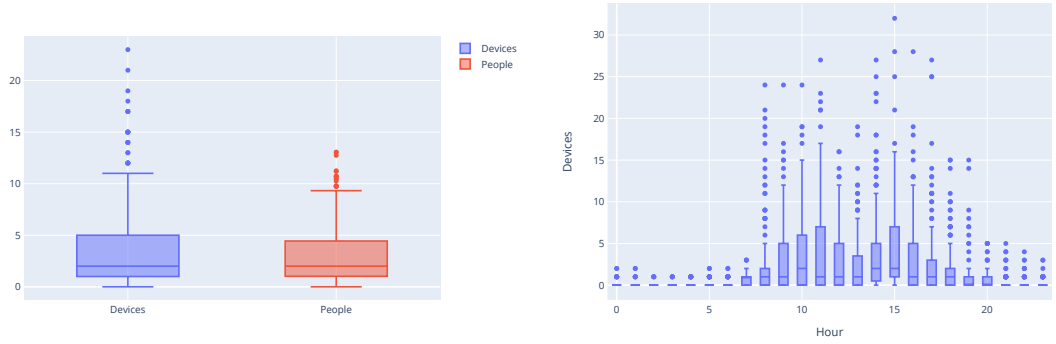
timestamp	people	building	devices	ap	sstrength_avg	dayofweek	hour	month	quarter
2020-10-22 19:00:00	10	[redacted]	15.0	[redacted]	-56.250000	3	19	10	4
2020-10-21 19:00:00	0	[redacted]	2.0	[redacted]	-68.750000	2	19	10	4
2020-10-22 18:00:00	17	[redacted]	1.0	[redacted]	-74.500000	3	18	10	4
2020-10-22 11:00:00	14	[redacted]	7.0	[redacted]	-66.090909	3	11	10	4
2020-10-21 17:00:00	30	[redacted]	1.0	[redacted]	-86.000000	2	17	10	4

Figure 7.9: Crowd sensing data after the cleaning process.Some information is omitted due to privacy concerns.

Data Analysis

To better understand the problem and the data to be used in the modelling phase, some EDA can be performed. In this section are presented some of the most important

In Figure 7.10a, it is presented the differences between the number of devices and the count of people (ground-truth) for a given location. From Figure 7.10b, it can be inferred that the number of devices is only relevant during working hours, being near 0 for periods before 7 AM and after 8 PM. Note that the figures are not from the same place.



(a) Comparison between *devices* and *people* variables.

(b) Example of devices distribution during the hours of the day for a certain location.

Figure 7.10: Visual data analysis of the data related to the devices count.

Using the data from different hours, it is possible to calculate the average of connected devices for each AP. Figure 7.11 shows an example of an arbitrary location with the distribution of devices for each AP in percentage relatively to the building. Some information is truncated due to privacy reasons.

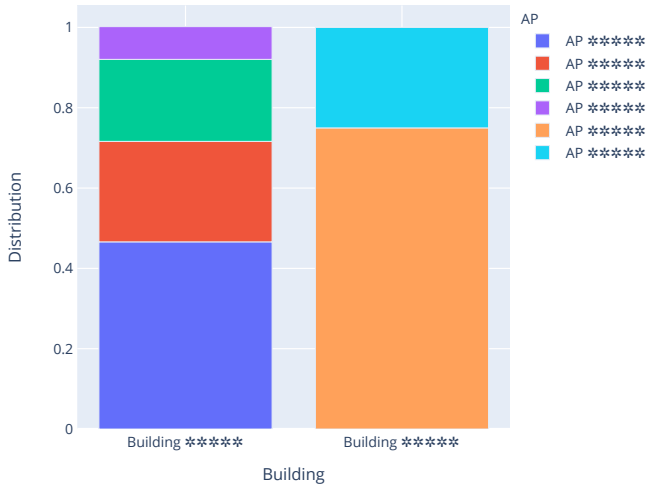


Figure 7.11: Example of devices distributions by AP for each building.

Data Modelling

To solve the present problem, some ANNs are trained with different goals. All the networks in this case use only *Dense* layers.

The first trained network is a regression model, that uses the count of people for each building (ground-truth) as the output variable. To estimate the number of people, it uses the APs data and the temporal information, understanding the people's behaviour in different places and time. The results of this model presents a test RMSE of 5.95, which means that the approximation fails in average by 6 people in each building.

Using the same type of model to estimate the number of people by AP, the test RMSE is 1.7. Since the ground-truth count of people is only available for the building as a whole, the AP people's count used to train the model is an approximation using the distribution of devices by AP, presented in the previous section.

To detect crowd places where the ground-truth for the building is not available, a self-supervised approach is followed. An AE network is created using the same features as the previous models, as both input and output of the network. Since the crowd situations are not frequent, the model is able to understand the normal occupancy for each place, detecting abnormal situations by higher reconstruction errors. Figure 7.12 shows the results for one of the places. Using a threshold of $MSE = 40$, the network is capable of detecting the anomalous situations when the place limit is exceeded (in this case, the critical capacity of this place is near 20).

Problem Solution

The solution in this case is not the direct output of the model. The results need to be processed before they be presented to the end user. When showing the estimate number of people in each place, it is needed to round the

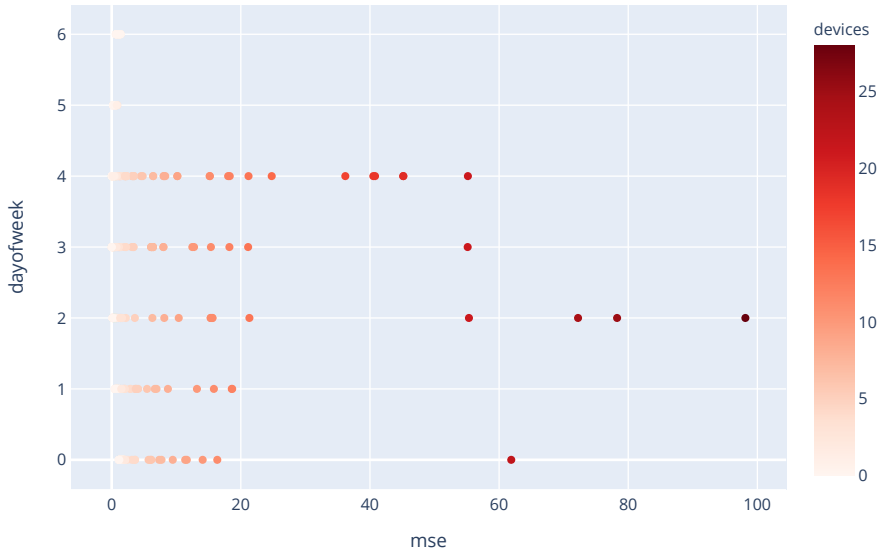


Figure 7.12: Example of AE model results for a specific location.

models outputs to a integer and remove numbers below zero (since the regression model outputs a number in $\mathbb{R}$). The models are deployed internally in the back-end and its results are available through an API that is used by the app front-end.

Additionally, the results can be divided in categories (presented as colors, for example), representing different levels of people concentrations in each place, depending on the front-end decisions.

7.4 CONCLUSIONS

Regarding the first application presented on this chapter, it was concluded that ANNs are important for the mechanical design, since they can obtain stress and displacement fields more quickly than the traditional method, with small approximation errors. Note that this was only a first approach and that new techniques could be used to improve the results obtained.

The models presented in the second use case also prove that connectionist systems have a great versatility to deal with multiple types of data, providing useful information even when ground-truth data is missing to train the models.

8

CONCLUSIONS

> *Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning.*
>
> Albert Einstein

The present dissertation is mainly concerned with the study of connectionist systems, specially to their application to AD and image processing. In this work, several use cases were studied in order to present the versatility of this technology. The first case, the preliminary MNIST experimental problem (Chapter 5), was used to set up and validate the models in a simpler scenario. With the results presented by the built networks it can be concluded that the studied architectures are appropriated to solve the type of problems under analysis, since they perform well in its simplified version.

For the main use case, the pavement monitoring problem (Chapter 6), a full end-to-end project is developed, with software prototypes that are well suited to accomplish the proposed objectives. The ML models used in the introductory study are upgraded to embrace the new challenges of this more complex domain, as well as the image preprocessing, that is needed in this settings.

The last two additional cases presented in Chapter 7 prove that connectionist systems are one of the best learning approaches due to its wide range of applications, opening new opportunities in different research domains.

This chapter is divided in four sections. Based on the defined objectives (section 1.3 - Chapter 1), sections 8.1 and 8.2 present the theoretical and practical conclusions that can be drawn from this work. In theoretical conclusions are presented the findings about the ML models in a scientific view. In practical conclusions, are presented the real-world solved problems using the mentioned techniques. In section 6.12, it is made a critical appraisal of the results. The proposed future work is presented in section 8.3.

8.1 THEORETICAL CONCLUSIONS

The use of connectionist systems is increasingly gaining importance in problem solving. All the use cases presented in this dissertation use ANNs to achieve each of their defined purposes. Depending on the input and output data, different architectures and methodologies are used.

T1) When dealing with numerical data, the best results are acquired using multiple stacked *Dense* layers in a feed-forward setting. As demonstrated in other CV areas, as object detection or image classification, CNNs are the SOTA architecture to deal with data that has a grid-like topology, such as the pixels of an image. The experiments done in previous work regarding AD using connectionist systems have shown a convergence in the best architectures currently used. Methodologies to treat AD problems frequently use AE models and its variants as self-supervised methods to find anomalous instances in datasets. Applying the same concepts to the CV field, better results have been accomplished when using architectures combinations, namely CNN with AE and VAE;

T2) The built models show good results in all use cases, bearing in mind that the proposed solutions are novel methodologies in each of the application domains. The trained AE variants (used in the pavement monitoring use case) achieve AUROC values near 1.0, which means that using an adequate cut-point calculation method, it is possible to find the anomalies with almost no errors. The threshold calculation methods proposed in this dissertation clearly show that if labelled data is available, the performance to detect the presence of anomalies is enhanced, as expected.

T3) Regarding the models of the additional use cases, the test errors presented for the mechanical design problem (11.3% in displacement field and 26.8% in stress field) are acceptable taking into account the trade-off between them and the improvements in the processing time, that is 40 times faster than the conventional software. Also in the crowds detection problem, a deviation of approximately 2 people is negligible when approximating the real number of people for each place in that context. The results of the semi-supervised AE architecture to automatically detect anomalous crowd situations show a high correlation between the reconstruction errors and the actual high number of detected devices in that area, which allows to identify these situations correctly;

T4) The empiric experiments made in all these contexts prove the relevance of ANNs and its good performance when considerable amounts of data are available for the training process. Connectionist systems are indeed one of the most adaptable technologies nowadays when the problem is well defined to adapt the variables accordingly, performing appropriately even in scenarios with incomplete information.

8.2 PRACTICAL CONCLUSIONS

From a problem-solving perspective, this dissertation presents several contributions, specially in the software engineering and ML domains, but also for the studied application contexts. In Chapter 2.2 is proposed a full pipeline that fits the development of DS projects. When the modelling phase of the pipeline is meant to use ML models, a more refined view of this step is introduced, dealing with the model construction and the data preparation (Chapter 2.3). The proposed methodologies were used in all the demonstrated cases.

P1) The traditional solutions for pavement monitoring presented in Section 6.3, include systems as the PaveTesting® and VIZIROAD® systems, that are either expensive or deprecated;

P2) Based on previous solutions, namely the VIZIROAD®, the gathering system presents improvements, since it is compatible with modern operating systems and it is prepared to work with touch screen laptops and tablets, providing the ability to acquire image data;

P3) The visualization tool, beyond presenting a more convenient way to access the stored data than the traditional paper files, also provides the possibility to visualize the road pictures and degradations associated with each location;

P4) The existent automatic pavement classification solutions presented in Section 6.4, use different approaches to achieve this goal, specially using accelerometer data instead of images. Using image data, pixel level and 3D image methods are also commonly explored;

P5) The ML models trained are a novel strategy to detect pavement degradations automatically. For this context, assuming that the acquired data is correctly labelled, the best architecture is the VAE using the supervised threshold calculation method, that is expected to distinguish accurately the bad pavement images.

P6) The solution presented to monitor and classify the pavement comprehends three main parts: a data gathering software, a prototype to visualize it and automatic pavement classification models. The best model to use in this setting is the VAE model, perfectly distinguish both classes with accuracy of 100%. When anomaly data is not available to train the models, the best option is to use the AE, using the distribution of normal instances to calculate the cut-point. The accuracy in this case drops to 81%, however the precision stills being 100%. Comparing the methodologies, it can be concluded that all of them have high confidence when the cases are classified as anomaly, since the precision is always above 94% (related with T4).

Comparing the full proposed solution with the available commercial ones, for example PaveTesting® and Dynatest®, it is clear that the involved costs are extremely reduced, either during the development, either when utilizing it. The proposed used instruments are only budget cameras and a mid-range laptop. The laptop in the experimental setting was used to acquire the data, train the models and also visualize it.

Additionally to the initial proposed practical objectives for this dissertation, two more case studies were conducted. Even though the aim of these applications were to observe the behaviour of connectionist systems in different domains, one of them also lead to a practical real-world solution. This contribution is reflected in the back-end of a mobile application developed within the scope of the *MARÉ* project, where the solution for the crowds detection problem is incorporated.

8.3 FUTURE WORK

The present dissertation opens a wide spectrum of future work, either for research, specially in the ML field, or in software development directed for the covered areas. Regarding the studied research topics, the following work directions are proposed:

- Improve the studied methodologies scores, specially in a unsupervised scenario, where there is no labelled data to train the models. This can be done exploring the existent AE and VAE models, that use only

normal data to be trained, and modifying the threshold calculation algorithms. Examples of alternatives to calculate the threshold is the use of z-scores and the Empirical Rule, when the SSIM data is a Gaussian distribution, or the Chebyshev's Theorem otherwise (Shafer and Zhang, 2012).

- One of the biggest issues of the present work is the lack of data, with a relatively small dataset of 400 images divided to training, validation, threshold calculation and testing. One possibility to solve this is to explore data augmentation techniques. Using image transformations (e.g. rotations, shifts, inversions, etc.) more images can be created, increasing the dataset size. Since data augmentation can be performed automatically during training runtime by some DL frameworks (TensorFlow Documentation, 2021), it is proposed to reduce the training dataset size (that will be augmented inside the network), using a greater number of original images to the rest of the phases (Chollet, 2017).

- Another possible approach to study the architectures is to use different pavement datasets. Even though other publicly available datasets were not suitable for the practical use case context, the study of the presented methodologies with other pavement perspectives is also encouraged with the possibility to apply them in other application settings.

- The GAN framework is also pointed as an alternative solution when dealing with AD. It is suggested the analysis of this approach to detect the presence of degradations in images. The set up could be identical to the one used in the AE variants, using the generator and its reconstruction errors to detect anomalies (Schlegl et al., 2017; Mattia et al., 2019).

- Generative models such as VAE and GAN architectures provide a reduced representation of the images. Some characteristics are encoded in that representation as feature vectors, which means that modifying a specific variable in the latent space will be reflected in the generated image (White, 2016). It is proposed to train similar networks with pavement images to discover feature vectors that can reflect the pavement conditions spectrum.

- Transfer learning refers to the use of pretrained models, applying them to a different domain, *transferring* the already learned information (Pan and Yang, 2010). This is a common practice in CV problems, freezing some convolutional layers already trained with images and training only a part of the network that will be specific for that domain (Chollet, 2017). It can be used also in this case, reducing the training times and eventually improving the results.

When looking from a software development perspective, the following steps can be taken:

- Improve the pavement data acquisition software to get data from multiple cameras simultaneously. This would be more adequate, since with three cameras it would be possible to acquire images for the left, middle and right traversal locations on the road. Also the hardware conditions can be adapted to acquire images near the ground and to calibrate the light exposure differences during the gathering process.

- Use GPS data to localize the acquiring location. In the present work an odometer is used to measure the travelled distance and to label the pavement information and pictures. An alternative is to use GPS data, depending on the reliability of this technology and its precision for each scenario.

- Build models to distinguish between the degradations. After the images are filtered from the built networks, that detect the presence of degradations, a possible next step is to identify the different classes of degradations (Uslu et al., 2011). This is proposed to be done using a multi-label classifier, since in the same image could eventually be present more than one degradation (Chen et al., 2019).

- The full use case pipeline would be closed if it is possible to acquire also the measurement of each classified degradation. It is accomplished by building models to get the severity of degradations, after they have been correctly detected and classified. Pixel-level approaches are commonly used to understand the pixels that are part of the degradation (Augustauskas and Lipnickas, 2020). Using these techniques and further image analysis it is possible to understand how much pixels are identified, and properly assign a severity level.

- Last but not least, it is suggested the deployment of the models in a integrated solution that is continuously receiving the acquired data, processing it and classifying the pavement. Using a similar interface of the proposed data analysis software, the information is then presented to the user in a easy to use solution.

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer International Publishing.

Akçay, S., Atapour-Abarghouei, A., and Breckon, T. P. (2019). Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection.

Aleskerov, E., Freisleben, B., and Rao, B. (1997). CARDWATCH: a neural network based database mining system for credit card fraud detection. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*. IEEE.

Alla, S. (2019). *Beginning anomaly detection using Python-based deep learning : with Keras and PyTorch*. Apress, New York.

An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18.

Augustauskas, R. and Lipnickas, A. (2020). Improved pixel-level pavement-defect segmentation using a deep autoencoder. *Sensors*, 20(9):2557.

Balaji, A. J., Balaji, G. T., Dinesh, M., Binoy, N., and Ram, D. S. H. (2019). Asphalt crack dataset. https://data.mendeley.com/datasets/xnzhj3x8v4/2.

Barrow, H. (1996). Connectionism and Neural Networks. In *Artificial Intelligence*, pages 135–155. Elsevier.

Basheer, I. and Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31. Neural Computting in Micrbiology.

Baur, C., Wiestler, B., Albarqouni, S., and Navab, N. (2019). Deep autoencoding models for unsupervised anomaly segmentation in brain MR images. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 161–169. Springer International Publishing.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Bota, M. and Swanson, L. W. (2007). The neuron classification problem. *Brain Research Reviews*, 56(1):79–88.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 111–118, Madison, WI, USA. Omnipress.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Breje, A.-R., Gyorödi, R., Gyorödi, C., Zmaranda, D., and Pecherle, G. (2018). Comparative study of data sending methods for XML and JSON models. *International Journal of Advanced Computer Science and Applications*, 9(12).

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.

CAUCHY, A. (1847). Methode generale pour la resolution des systemes d'equations simultanees. *C.R. Acad. Sci. Paris*, 25:536–538.

Chalapathy, R. and Chawla, S. (2019). Deep learning for anomaly detection: A survey.

Chalapathy, R., Toth, E., and Chawla, S. (2018). Group anomaly detection using deep generative models.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection. *ACM Computing Surveys*, 41(3):1–58.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Chen, Z.-M., Wei, X.-S., Wang, P., and Guo, Y. (2019). Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

chin Tai, Y., wei Chan, C., and Hsu, J. (2010). Automatic road anomaly detection using smart mobile device. In *2010 15th Conference on Artificial Intelligence and Applications*.

Chollet, F. (2017). *Deep Learning with Python*. Manning.

Chollet, F. et al. (2015). Keras. https://keras.io.

Commenges, D. (2015). Information theory and statistics: an overview.

Conway, D. (2010). The data science venn diagram.

Cubero-Fernandez, A., Rodriguez-Lozano, F. J., Villatoro, R., Olivares, J., and Palomares, J. M. (2017). Efficient pavement crack detection and classification. *EURASIP Journal on Image and Video Processing*, 2017(1).

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

D'Amico, F., Calvi, A., Bianchini Ciampoli, L., Tosti, F., and Brancadoro, M. G. (2018). Evaluation of the impact of pavement degradation on driving comfort and safety using a dynamic simulation model. *Advances in Transportation Studies*, 1:109–120.

Ergen, T., Mirza, A. H., and Kozat, S. S. (2017). Unsupervised and semi-supervised anomaly detection with lstm neural networks.

Gabinete de Gestão da Rede - Estradas de Portugal (2008). Catálogo de Degradações dos Pavimentos Rodoviários.

Garca, S., Luengo, J., and Herrera, F. (2014). *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gorokhov, O., Petrovskiy, M., and Mashechkin, I. (2017). Convolutional neural networks for unsupervised anomaly detection in text data. In *Lecture Notes in Computer Science*, pages 500–507. Springer International Publishing.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

Hebb, D. (2005). *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis.

Heller, M. (2019). How to choose the right database for your application. https://www.infoworld.com/article/3452894/how-to-choose-the-right-database-for-your-application.html. Last checked on Aug 24, 2020.

Hilbert, M. and Lopez, P. (2011). The world's technological capacity to store, communicate, and compute information. *Science*, 332(6025):60–65.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Ishwarappa and Anuradha, J. (2015). A brief introduction on big data 5vs characteristics and hadoop technology. *Procedia Computer Science*, 48:319–324.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

Kammerer, K., Hoppenstedt, B., Pryss, R., Stökler, S., Allgaier, J., and Reichert, M. (2019). Anomaly detections for manufacturing systems based on sensor data—insights into two challenging real-world production settings. *Sensors*, 19(24):5370.

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

Kulkarni, V. A. and Firestein, B. L. (2012). The dendritic tree and brain disorders. *Molecular and Cellular Neuroscience*, 50(1):10–20.

Kyriakou, C., Christodoulou, S. E., and Dimitriou, L. (2019). Do vehicles sense pavement surface anomalies? In *Proceedings of the 2019 European Conference on Computing in Construction*. University College Dublin.

Laurent, J., Hébert, J. F., Lefebvre, D., and Savard, Y. (2012). Using 3d laser profiling sensors for the automated measurement of road surface conditions. In Scarpas, A., Kringos, N., Al-Qadi, I., and A., L., editors, *7th RILEM International Conference on Cracking in Pavements*, pages 157–167, Dordrecht. Springer Netherlands.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

LeCun, Y., Cortes, C., and Burges, C. J. (1998). The mnist database. `http://yann.lecun.com/exdb/mnist/`. Last checked on May 19, 2020.

Li, Y. and Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. In *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. IEEE.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Markou, M. and Singh, S. (2003a). Novelty detection: a review—part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497.

Markou, M. and Singh, S. (2003b). Novelty detection: a review—part 2: neural network based approaches. *Signal Processing*, 83(12):2499–2521.

Masino, J., Thumm, J., Frey, M., and Gauterin, F. (2017). Learning from the crowd: Road infrastructure monitoring system. *Journal of Traffic and Transportation Engineering (English Edition)*, 4(5):451–463.

Mattia, F. D., Galeone, P., Simoni, M. D., and Ghelfi, E. (2019). A survey on gans for anomaly detection. *CoRR*, abs/1906.11632.

McCarthy, J. (2007). What is artificial intelligence?

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Minhas, M. S. and Zelek, J. (2019). Anomaly detection in images.

Mohamed, K. S. (2018). *Machine Learning for Model Order Reduction*. Springer International Publishing, Cham.

Nurseitov, N., Paulson, M., Reynolds, R., and Izurieta, C. (2009). Comparison of json and xml data interchange formats: A case study. In *22nd International Conference on Computer Applications in Industry and Engineering 2009, CAINE 2009*, pages 157–162.

OpenCV Documentation (2008). Canny edge detector. https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html. Last checked on May 19, 2019.

Oussous, A., Benjelloun, F.-Z., Lahcen, A. A., and Belfkih, S. (2018). Big data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, 30(4):431–448.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

pandas development team, T. (2020). pandas-dev/pandas: Pandas.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pérez, F. and Granger, B. E. (2007). IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29.

Picado Santos, L., Ferreira, A., and Pereira, P. (2006). Estruturação de um sistema de gestão de pavimentos para uma rede rodoviária de carácter nacional. *Engenharia Civil*, 26:45–59.

Plotly Technologies Inc. (2015). Collaborative data science. https://plot.ly.

Prince, S. J. D. (2012). *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, USA, 1st edition.

Radopoulou, S. C. and Brilakis, I. (2017). Automated detection of multiple pavement defects. *Journal of Computing in Civil Engineering*, 31(2):04016057.

Rephann, T. and Isserman, A. (1994). New highways as economic development tools: An evaluation using quasi-experimental matching methods. *Regional Science and Urban Economics*, 24(6):723–751.

Research & Development division of the Highway Department (2013). Guidance Notes Catalogue of Road Defects. *Research & Development division of the Highway Department*, pages 1–53.

Ribeiro, J. (2020). Modelos de ordem reduzida na análise estrutural de transformadores de potência. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto.

Ribeiro, J., Gomes, L., and Tavares, S. (2021a). Artificial Neural Networks Applied in Mechanical Structural Design. *Journal of Computation and Artificial Intelligence in Mechanics and Biomechanics*, 1(1):14–21.

Ribeiro, J. P., Tavares, S. M., and Parente, M. (2021b). Stress–strain evaluation of structural parts using artificial neural networks. *Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications*, page 146442072199244.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z., and Klette, R. (2018). Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 172:88–97.

Sagiroglu, S. and Sinanc, D. (2013). Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE.

Sakr, S. and Gaber, M. (2014). *Large Scale and Big Data: Processing and Management*. Auerbach Publications, USA.

Saltz, J. (2021). What is a data science workflow? https://www.datascience-pm.com/data-science-workflow/. Last checked on Aug 24, 2020.

Sara, U., Akter, M., and Uddin, M. (2019). Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 07:8–18.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Niethammer, M., Styner, M., Aylward, S., Zhu, H., Oguz, I., Yap, P.-T., and Shen, D., editors, *Information Processing in Medical Imaging*, pages 146–157, Cham. Springer International Publishing.

Schlegl, T., Waldstein, S. M., Vogl, W.-D., Schmidt-Erfurth, U., and Langs, G. (2015). Predicting semantic descriptions from medical images with convolutional neural networks. In *Lecture Notes in Computer Science*, pages 437–448. Springer International Publishing.

Seraj, F., van der Zwaag, B., Dilo, A., Luarasi, T., and Havinga, P. (2014). Roads: A road pavement monitoring system for anomaly detection using smart phones. In *Proceedings of the 1st International Workshop on Machine Learning for Urban Sensor Data, SenseML 2014*, pages 1–16. Springer.

Seraj, F., Zhang, K., Turkes, O., Meratnia, N., and Havinga, P. J. M. (2015). A smartphone based method to enhance road pavement anomaly detection by analyzing the driver behavior. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers - UbiComp '15*. ACM Press.

Shafer, D. and Zhang, Z. (2012). *Introductory Statistics*. Saylor Foundation.

Silva, N., Shah, V., Soares, J., and Rodrigues, H. (2018). Road anomalies detection system evaluation. *Sensors*, 18(7):1984.

Silva, N., Soares, J., Shah, V., Santos, M. Y., and Rodrigues, H. (2017). Anomaly detection in roads with a data mining approach. *Procedia Computer Science*, 121:415–422.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Soares, J., Silva, N., Shah, V., and Rodrigues, H. (2018). A road condition service based on a collaborative mobile sensing approach. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE.

Staar, B., Lütjen, M., and Freitag, M. (2019). Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, 79:484–489.

Standring, S. (2015). *Gray's Anatomy: The Anatomical Basis of Clinical Practice*. Elsevier Health Sciences.

Stockman, G. and Shapiro, L. G. (2001). *Computer Vision*. Prentice Hall PTR, USA, 1st edition.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition.

TensorFlow Documentation (2021). Data augmentation. https://www.tensorflow.org/tutorials/images/data_augmentation. Last checked on Feb 05, 2021.

Timoshenko, S. and Goodier, J. N. (1951). *Theory of Elasticity*. McGraw-Hill. Google-Books-ID: 11ISAAAAIAAJ.

Tom V. Mathew (2009). Introduction to pavement design - lecture notes in transportation systems engineering. https://www.civil.iitb.ac.in/tvm/1100_LnTse/401_lnTse/plain/plain.html. Last checked on Nov 24, 2019.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846.

TURING, A. M. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460.

Ugural, A. (2020). *Plates and Shells: Theory and Analysis*. CRC Press, 4 edition.

Uslu, B., Golparvar-Fard, M., and de la Garza, J. M. (2011). Image-based 3d reconstruction and recognition for enhanced highway condition assessment. In *Computing in Civil Engineering (2011)*. American Society of Civil Engineers.

van der Aalst, W. (2016). *Process Mining: Data Science in Action*. Springer Publishing Company, Incorporated, 2nd edition.

van der Aalst, W. M. P. (2014). Data scientist: The engineer of the future. In *Enterprise Interoperability VI*, pages 13–26. Springer International Publishing.

Vinayakamurthy, M. (2017). Effect of pavement condition on accident rate. Master's thesis, ARIZONA STATE UNIVERSITY.

Wang, K. (2011). Automated survey of pavement distress based on 2d and 3d laser images. *Mack-Blackwell Rural Transportation Center*.

Wang, N., Chen, C., Xie, Y., and Ma, L. (2020). Brain tumor anomaly detection via latent regularized adversarial network.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Wei, Q., Ren, Y., Hou, R., Shi, B., Lo, J. Y., and Carin, L. (2018). Anomaly detection for medical images based on a one-class classification. In Petrick, N. and Mori, K., editors, *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 105751M.

White, T. (2016). Sampling generative networks: Notes on a few effective techniques. *CoRR*, abs/1609.04468.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs.

Wong, R. O. L. and Ghosh, A. (2002). Activity-dependent regulation of dendritic growth and patterning. *Nature Reviews Neuroscience*, 3(10):803–812.

Wulsin, D., Blanco, J., Mani, R., and Litt, B. (2010). Semi-supervised anomaly detection for EEG waveforms using deep belief nets. In *2010 Ninth International Conference on Machine Learning and Applications*. IEEE.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2019). Self-training with noisy student improves imagenet classification.

Yang, F., Zhang, L., Yu, S., Prokhorov, D. V., Mei, X., and Ling, H. (2019). Feature pyramid and hierarchical boosting network for pavement crack detection. *CoRR*, abs/1901.06340.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

Zhang, A., Wang, K. C. P., Fei, Y., Liu, Y., Chen, C., Yang, G., Li, J. Q., Yang, E., and Qiu, S. (2018a). Automated pixel-level pavement crack detection on 3d asphalt surfaces with a recurrent neural network. *Computer-Aided Civil and Infrastructure Engineering*, 34(3):213–229.

Zhang, A., Wang, K. C. P., Fei, Y., Liu, Y., Tao, S., Chen, C., Li, J. Q., and Li, B. (2018b). Deep learning–based fully automated pavement crack detection on 3d asphalt surfaces with an improved CrackNet. *Journal of Computing in Civil Engineering*, 32(5):04018041.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. (2013). *The Finite Element Method: its Basis and Fundamentals*. Butterworth-Heinemann.

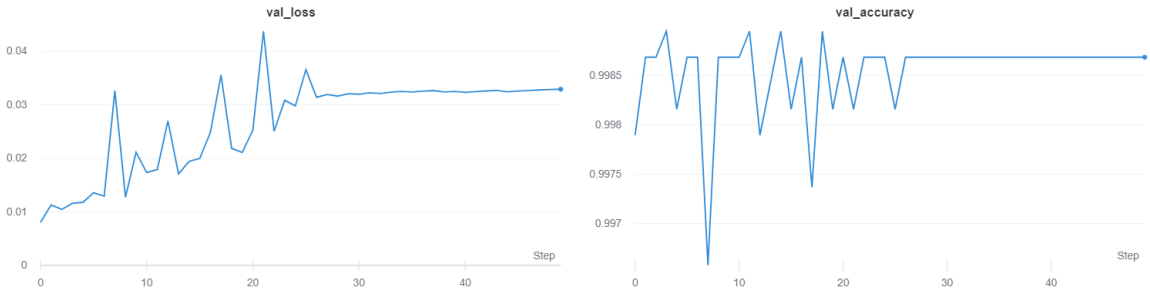
Part III

APPENDICES

A

VALIDATION METRICS

A.1 MNIST



(a) Validation loss. (b) Validation accuracy.

Figure A.1: MNIST Classifier - Validation metrics during the classifier training process.

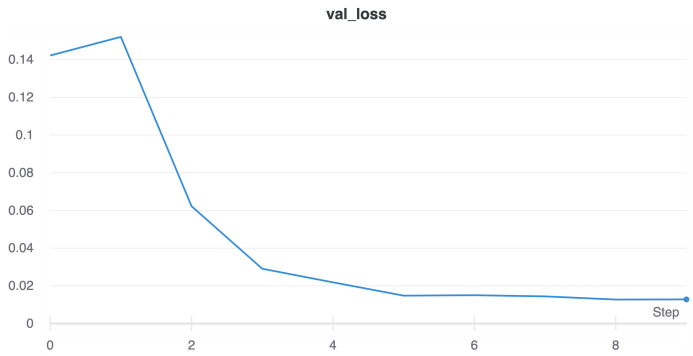


Figure A.2: MNIST AE - Validation loss during the training process.

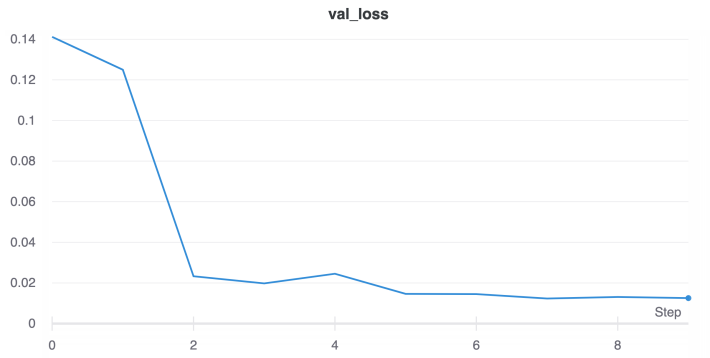
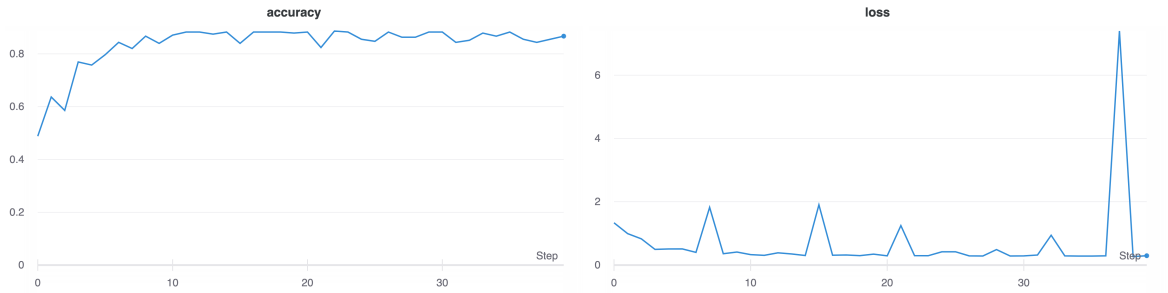


Figure A.3: MNIST VAE - Validation loss during the training process.

A.2 PAVEMENT



(a) Validation loss.

(b) Validation accuracy.

Figure A.4: Pavement Classifier - Validation metrics during the classifier training process.

Figure A.5: Pavement AE - Validation loss during the training process.

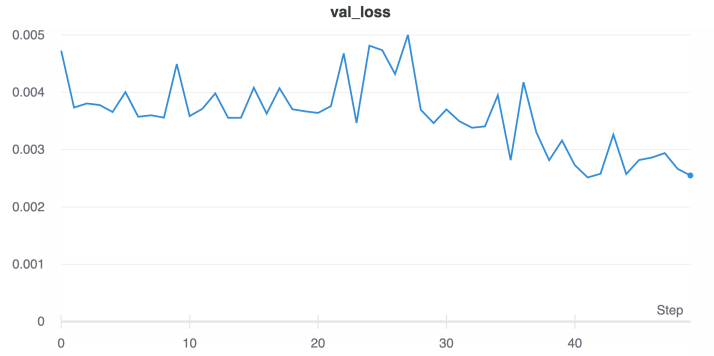


Figure A.6: Pavement VAE - Validation loss during the training process.

B

CONFUSION MATRICES

B.1 MNIST

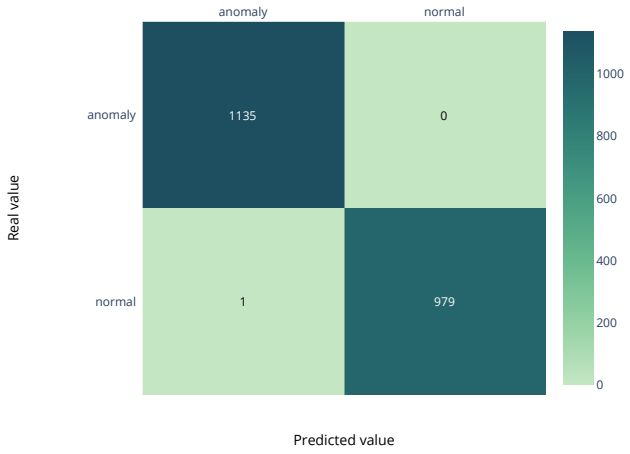
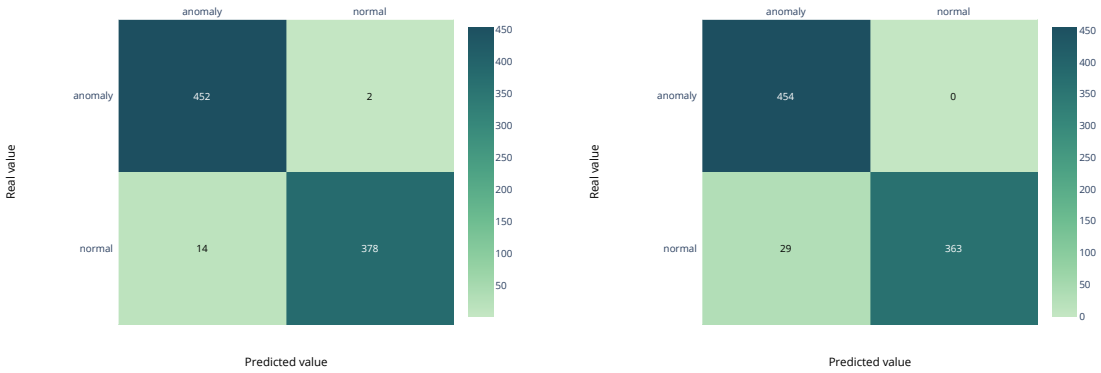


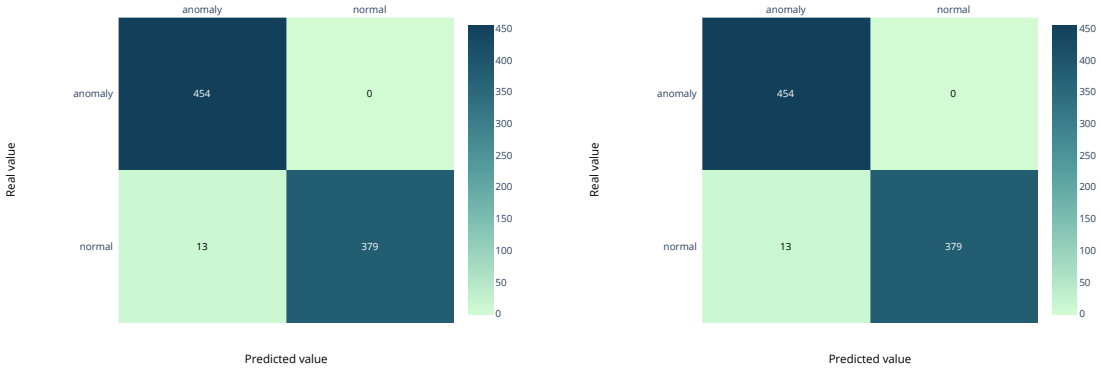
Figure B.1: MNIST Classifier - Confusion Matrix.



(a) Supervised threshold calculation.

(b) Unsupervised threshold calculation.

Figure B.2: MNIST AE - Confusion matrices.



(a) Supervised threshold calculation.

(b) Unsupervised threshold calculation.

Figure B.3: MNIST VAE - Confusion matrices.

B.2 PAVEMENT

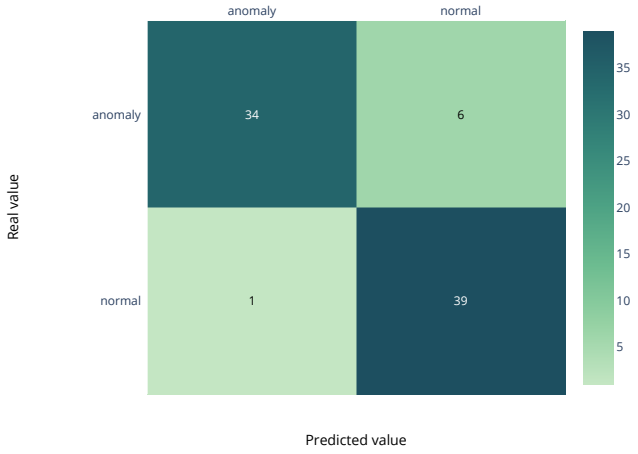
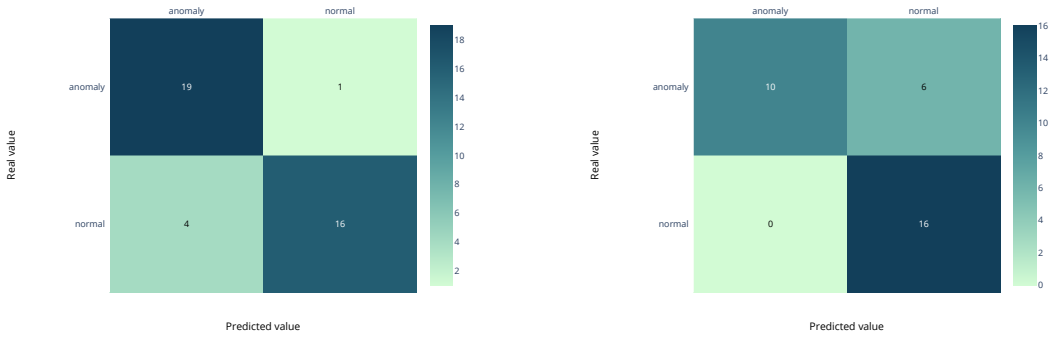


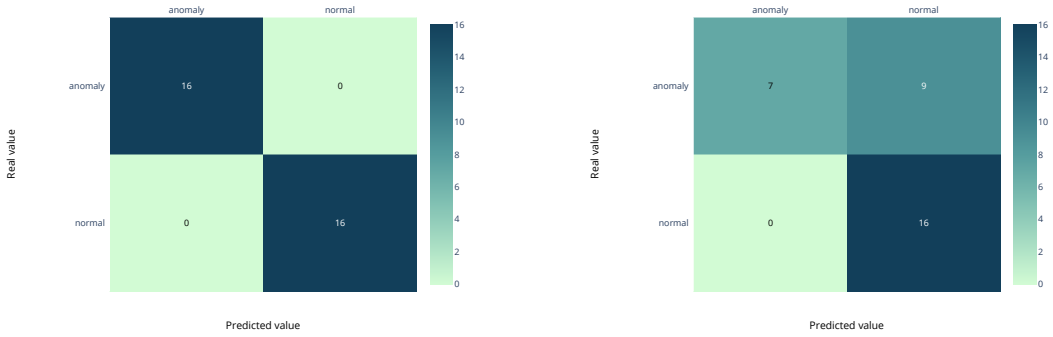
Figure B.4: Pavement Classifier - Confusion Matrix.



(a) Supervised threshold calculation method.

(b) Unsupervised threshold calculation method.

Figure B.5: Pavement AE - Confusion matrices.



(a) Supervised threshold calculation method.

(b) Unsupervised threshold calculation method.

Figure B.6: Pavement VAE - Confusion matrices.

C

PREDICTIONS

C.1 MNIST

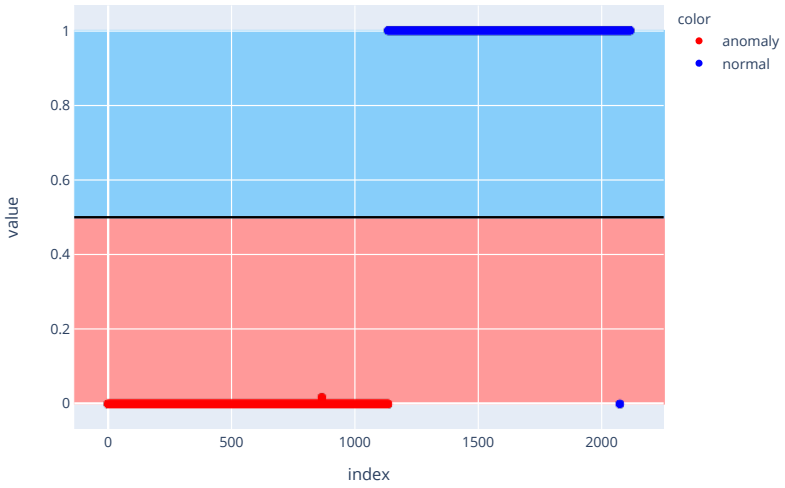
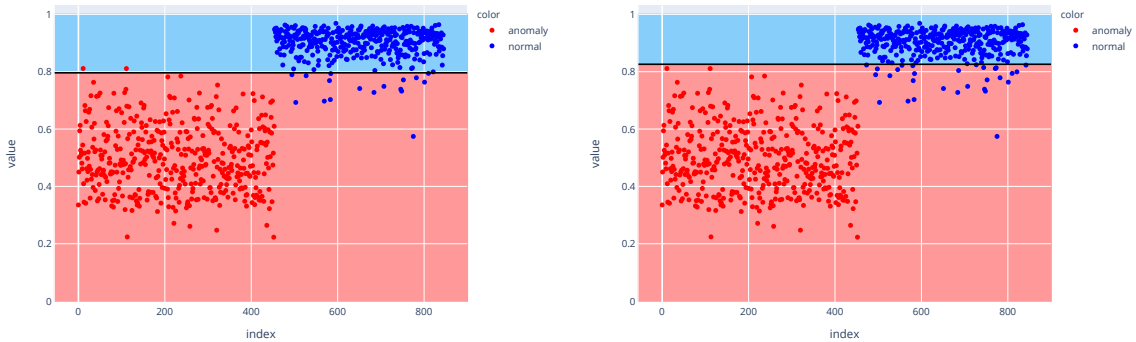


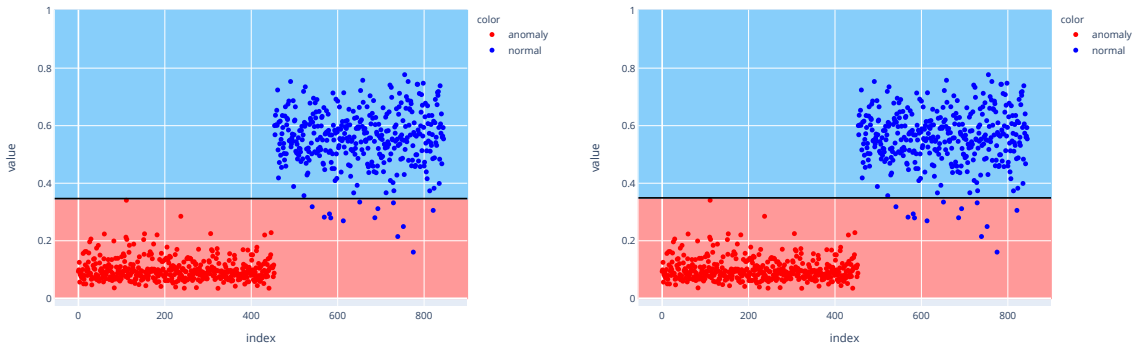
Figure C.1: MNIST Classifier - Predictions.



(a) Supervised threshold calculation.

(b) Unsupervised threshold calculation.

Figure C.2: MNIST Autoencoder - Predictions.



(a) Supervised threshold calculation.

(b) Unsupervised threshold calculation.

Figure C.3: MNIST VAE - Predictions.

C.2 PAVEMENT

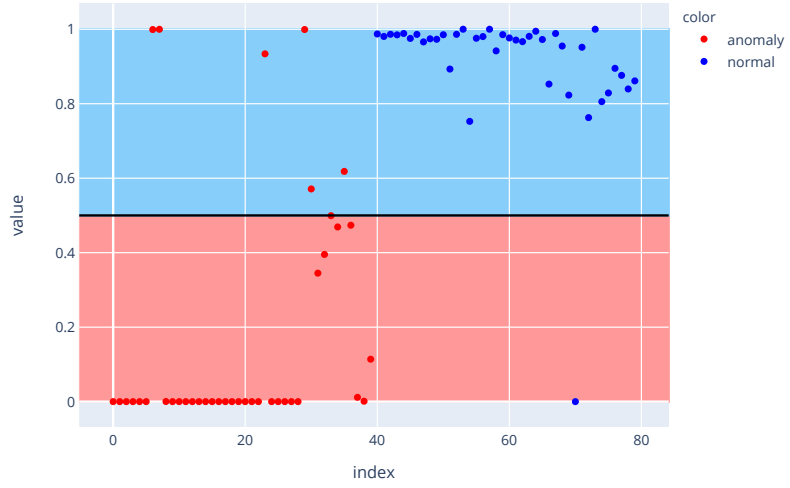
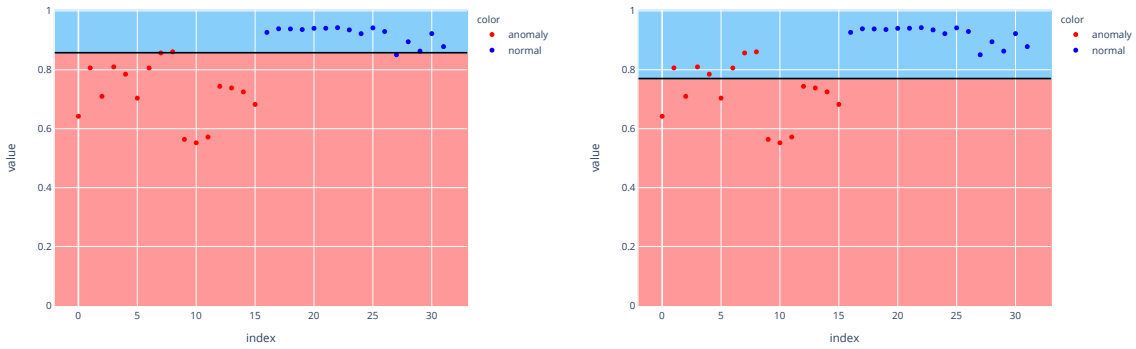


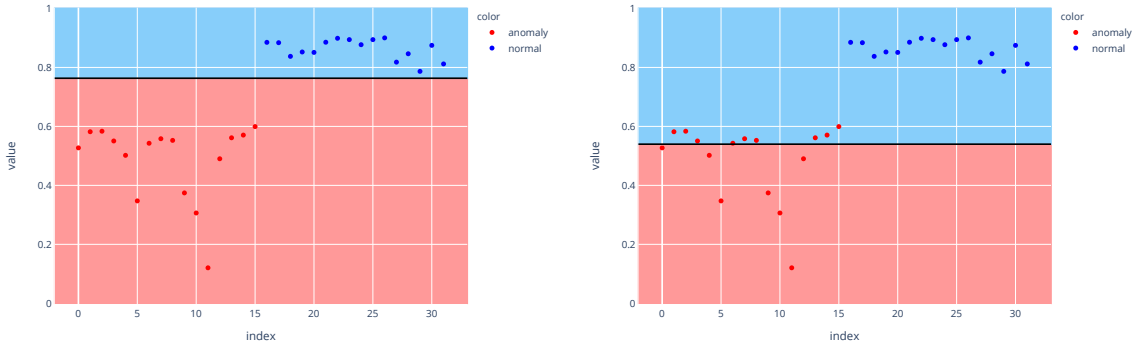
Figure C.4: Pavement Classifier - Raw predictions.



(a) Supervised threshold calculation method.

(b) Unsupervised threshold calculation method.

Figure C.5: Pavement AE - Reconstruction similarities and division of the classes.



(a) Supervised threshold calculation method.

(b) Unsupervised threshold calculation method.

Figure C.6: Pavement VAE - Reconstruction similarities and division of the classes.