N° Ordre……../FHC/UMBB/2024

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

M'HAMED BOUGARA-BOUMERDES UNIVERSITY

**F**aculty of **H**ydrocarbons and **C**hemistry

**Master dissertation**

Presented by:

**HADIDI Anfel**

Field: Science, technology and hydrocarbons

Option: automation and electrification of industrial processes

# Automated detection and classification of defects in the outer surface of transportation pipelines

**Infront of jury board:**

Year of study: 2023/2024

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

M'HAMED BOUGARA-BOUMERDES UNIVERSITY

**F**aculty of **H**ydrocarbons and **C**hemistry

Department: automation and electrification of industrial processes
Field: Science, technology and hydrocarbons
Option: Automatic control

# MASTER dissertation

***Theme:***

# Automated detection and classification of defects in the outer surface of transportation pipelines

| | |
|---|---|
| **Presented by:** | **Favorable opinion from supervisor:** |
| HADIDI Anfel | Full name: |
| | signature: |

**Favorable opinion of the jury chairman**
Full name:

Signature:

**Stamp and signature**

# Acknowledgement

بِسْمِ ٱللَّٰهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

*In the name of God, the infinitely companionate and merciful.*
*Praise to God. Lord of all the worlds.*
*The compassionate, the Merciful. Ruler of the day of Reckoning.*
*You alone do we worship; you alone do we ask for help.*
*Guide us to the straight path,*
*The path of those who have received your grace;*
*Not the path of those who have brought down wrath, nor of those who wander ashtray.*

I would want to express my sincere gratitude to my university advisor DR. **M.Khebli,** whose advice and assistance have been crucial to me over my academic career.

I would want to sincerely thank my hardworking internship supervisors Mr. **A.Battikh** and Mr. **S.Chebbi**, for giving me the chance to use my knowledge in a practical situation. Your support, helpful criticism, and mentorship have been crucial in helping me develop my abilities and self-assurance.

I am incredibly grateful to my family for their constant love, support, and selflessness. My success has been largely attributed to your unwavering support, and I sincerely appreciate your unwavering faith in me.

To my cherished friends, **Djamila, Hadjer, Ines, Lynda, Samah, Yasmine and Nadjet,** thank you for being my pillars of strength and sources of joy. Your unwavering support, understanding, and laughter have made every challenge more manageable and every triumph more meaningful.

To you my best friend **Lyn** you have been my rock, my confidant, and my partner in all of life’s moments. Thank you for standing by my side through every high and low, and for being the embodiment of unwavering friendship. I am endlessly grateful for your presence in my life.

I also want to acknowledge the class of Gaza of 2024, whose lives were tragically taken, who worked so hard for their future, a future denied to them by relentless shadow of conflict. Their dreams of graduating, of celebrating hard-earned achievements with family will remain unfulfilled.

# Dedication

I bestow this piece of creation to:

Thy mother,

Thy who carried me, In thy loving arms, solace I find.

A gentle touch, a lullaby so sweet, A guiding light, in sacrifice complete.

With tireless hands and heart so pure, Through every storm, thy, hast endure.

Thy dreams set aside for mine, A beacon of strength, eternally divine.

Thy father,

A distant gaze, yet strength pure , Through silent storms, thy did endure.

A guiding light from shadows cast, A quiet presence, steady and steadfast.

In the embrace of thine unseen care, A legacy of love, beyond compare.

To thine essence intertwined mine,

Thine the echoes of laughs and cry, in the back of my head I find.

Thine parcelled out every moment, sank into the fields of the throny roses.

Thine in togetherness walked, where shadows blend and twilight composes.

Thine spirit, a mirror to my own, reflected in the silent whispers of the night.

In every shared glance, in every whispered word, we glimpsed a light.

To thou dear self, braved the darkest night, Through trials unspoken, emerged into light.

To thou self with every fall, thou didst rose anew, A testament to the strength within thou.

To thou warrior self, battles waged and fears thou hast war valiantly, In quiet resilience, triumph thou sought.

Anon, standing tall, scars turned to pride, A journey of courage you cannot hide.

Embrace the existence thou became, In thou reflection, see what thou hast gained.

# Acronyms

**AI:** Artificial intelligence

**ML :** machine learning

**MFL :** magnetic flux leakage

**EMAT :** electromagnetic acoustic transduction

**CCTV :** closed circuit television video

**NDT :** non-destructive testing

**NASSCO :** National steel and shipbuilding company

**CNN :** convolutional neural network

**MSRCP :** multiscale retiney with chromacity preservation

**SIFT :** Scale-Invariant Feature Transform

**YOLO :** you only look once

**OS:** open system module

**CV2 :** Open Source Computer Vision

**VGG :** visual geometry group

**ReLu :** Rectified Linear Unit

**SVM :** support vector machine

**OvO :** one vs one

**RBF :** radial basis function

**RF :** random forest

# Abstract

This dissertation explores the critical importance of defect detection in pipelines and advocates for the integration of artificial intelligence (AI) to enhance inspection capabilities. Traditional methods of pipeline inspection are prone to human error and lack scalability. By employing machine learning models, this study proposes a novel approach to pipeline inspection.

The integration of AI offers numerous advantages, including increased efficiency, accuracy, and scalability in defect detection. Through rigorous experimentation and evaluation, this research demonstrates the effectiveness of AI-driven approaches in enhancing pipeline integrity management.

Furthermore, the study emphasizes the sensitivity of AI-based defect detection systems and underscores the significance of feature engineering using deep learning techniques. By extracting rich features from pipeline images, the VGG architecture combined with machine learning models facilitates more robust and discriminative representations, enhancing the model's ability to detect subtle defects with high precision and recall.

This dissertation contributes to the advancement of pipeline inspection practices, highlighting the potential of integrating advanced technologies to ensure safer, more reliable, and cost-effective maintenance strategies in the industrial sector.

**Keywords:** pipeline inspection, artificial intelligence, machine learning, VGG16, support vector machine, random forest, YOLOv8.

## Résumé:

Ce mémoire explore l'importance critique de la détection des défauts dans les pipelines et plaide en faveur de l'intégration de l'intelligence artificielle (IA) pour améliorer les capacités d'inspection. Les méthodes traditionnelles d'inspection des pipelines sont sujettes à l'erreur humaine et manquent d'évolutivité. En employant des modèles d'apprentissage automatique, en particulier des architectures d'apprentissage profond comme le VGG (Visual Geometry Group), cette étude propose de nouvelles approches de l'inspection des pipelines.

L'intégration de l'IA offre de nombreux avantages, notamment une efficacité, une précision et une évolutivité accrues dans la détection des défauts. Grâce à une expérimentation et une évaluation rigoureuse, cette recherche démontre l'efficacité des approches basées sur l'IA pour améliorer la gestion de l'intégrité des pipelines.

En outre, l'étude met l'accent sur la sensibilité des systèmes de détection des défauts basés sur l'IA et souligne l'importance de l'ingénierie des caractéristiques à l'aide de techniques d'apprentissage profond. En extrayant des caractéristiques riches des images de pipelines, l'architecture VGG facilite des représentations plus robustes et discriminatives, améliorant la capacité du modèle à détecter des défauts subtils avec une précision et un rappel élevés.

Cette thèse contribue à l'avancement des pratiques d'inspection des pipelines, en soulignant le potentiel de l'intégration des technologies avancées pour garantir des stratégies de maintenance plus sûres, plus fiables et plus rentables dans le secteur industriel.

**Mots-clés** : inspection des pipelines, intelligence artificielle, apprentissage automatique, VGG16, machine à vecteurs de support, forêt aléatoire, YOLOv8.

## الملخص

(AI) تستكشف هذه الأطروحة الأهمية الحاسمة للكشف عن العيوب في خطوط الأنابيب وتدعو إلى دمج الذكاء الاصطناعي لتعزيز قدرات الفحص. الطرق التقليدية لفحص خطوط الأنابيب عرضة للخطأ البشري وتفتقر إلى قابلية التوسع. تقترح هذه الدراسة نهجًا جديدًا لفحص خطوط الأنابيب من خلال استخدام نماذج التعلم الآلي، وخاصةً بنيات التعلم العميق

يوفر تكامل الذكاء الاصطناعي العديد من المزايا، بما في ذلك زيادة الكفاءة والدقة وقابلية التوسع في اكتشاف العيوب. من خلال التجريب والتقييم الدقيق، يوضح هذا البحث فعالية الأساليب القائمة على الذكاء الاصطناعي في تعزيز إدارة سلامة خطوط الأنابيب.

تساهم هذه الأطروحة في تطوير ممارسات فحص خطوط الأنابيب، وتسلط الضوء على إمكانات دمج التقنيات المتقدمة لضمان استراتيجيات صيانة أكثر أمانًا وموثوقية وفعالية من حيث التكلفة في القطاع الصناعي

**الكلمات المفتاحية:** فحص خطوط الأنابيب، الذكاء الاصطناعي، التعلم الآلي، آلة ناقلات الدعم، الغابة العشوائية

# Content table

*Résumé:* ........ *VII*

*General Introduction:* ........ *1*

*1. Chapter 1: Pipeline Inspection* ........ *5*

**1.1. Introduction** ........ **5**

**1.2. The Evolution of Pipeline Inspection Through History** ........ **5**

**1.3. Photographic Pipeline Inspection** ........ **7**

1.3.1. Closed Circuit Television Video (CCTV) inspection ........ 7

1.3.2. Halliburton's visual inspection of pipelines and processing plants ........ 10

**1.4. Photographic inspection limitations:** ........ **12**

**1.5. The urge for AI models integration in defect detection:** ........ **13**

**1.6. Conclusion** ........ **15**

*Chapter 2: Related work* ........ *16*

*2. Chapter 2: Related work* ........ *17*

**2.1. Introduction** ........ **17**

**2.2. Pipeline technology inspection methods** ........ **18**

2.2.1. Automated defect detection for sewer pipeline inspection and condition assessment from videos and images ........ 19

2.2.2. Automatic Defect Detection System for Petrochemical Pipelines ........ 21

2.3. **Conclusion:** 25

*Chapter 3: Dataset preparation* 26

3. *Chapter3: Dataset preparation* 27

3.1. **Introduction** 27

3.2. **Dataset creation constraints** 28

3.3. **Image collection** 28

3.3.1. Image collection sources 29

3.3.2. The total number of the collected images 29

3.4. **Image adaptation** 30

3.5. **Data ordering** 31

3.6. **Data augmentation** 32

3.7. **Image processing and feature engineering** 34

3.7.1. Image processing 35

3.7.2. Feature engineering with VGG16 (Visual Geometry Group from Oxford) 36

3.7.3. Data visualisation 39

3.7.4. Storing data in arrays 42

3.8. **Conclusion** 42

4. *Chapter4: Machine learning model building* 45

4.1. **Introduction:** 45

4.2. **Support vector machine (SVM):** 46

4.2.1. Data linearity ..... 46

4.2.2. Kernel regulation in SVMs ..... 51

4.2.3. One vs One SVM ..... 54

4.2.4. Grid search ..... 55

4.2.5. Approaches taken and their respective results ..... 56

4.2.6. observations ..... 63

4.2.7. Explanation and discussion: ..... 65

4.2.8. conclusion ..... 65

**4.3. Random forest ..... 66**

4.3.1. Description of a random forest: ..... 66

4.3.2. Ensemble learning techniques: ..... 66

4.3.3. Random forest algorithm building ..... 68

4.3.4. Hyperparameters to Increase the Predictive Power ..... 69

4.3.5. Approaches taken and their respective results ..... 69

4.3.6. observations ..... 75

4.3.7. Explanation and discussion: ..... 77

4.3.8. Conclusion: ..... 78

**4.4. Comparison between the two models: ..... 78**

4.4.1. Strength of the SVM classifier: ..... 78

4.4.2. How can random forest perform well in some scenarios: ..... 79

4.4.3. Side by side comparison: 80

**4.5. Conclusion: 81**

***5. Chapter 5: optimisation and deployment 83***

**5.1. Introduction 83**

**5.2. Binary tree of SVMs with YOLOv8 83**

5.2.1. Binary tree of SVMs 84

5.2.2. Crack & dent segmentation using YOLOv8 84

**5.3. Results: 87**

**5.4. Discussion: 90**

**5.5. Deployment: 92**

**5.6. Conclusion: 94**

***General conclusion 95***

***6. General conclusion 96***

***7. References: 98***

# Table of figure

**Figure 1-1**: industriel pipe inspection crawler camera robot 8

**Figure 1-2**:HVAC Duct Inspection 8

**Figure 1-3**: Push Rod Cameras: 11

**Figure 1-4:** Endoscope Cameras view inside a pipeline 12

**Figure 1-5:**Hal-Vision Camera 12

**Figure 1-6**: Gas leakage detected by computer vision 14

**Figure 2-1**:images of different types of defects 22

**Figure 2-2**:YOLOv5 network structure 24

**Figure 3-1**: data preparation steps. 28

**Figure 3-2**: examples of the collected images from each class 30

**Figure 3-3**: A dented pipeline image before and after upscaling 31

**Figure 3-4**:class based ordering 31

**Figure 3-5**: nature based ordering 32

**Figure 3-6**: representation of data augmentation process 34

**Figure 3-7:** steps taken in image processing 35

**Figure 3-11** fine tuned features representation (all classes together) 41

**Figure 3-**10 fine tuned features representation (each class separately) 41

**Figure 3-9**: fine tuned features representation (all classes together) 41

**Figure 3-8:** fine tuned features representation (each class separately) 41

**Figure 4-1**:decision boundary and margin of 48

**Figure 4-2:** decision boundary of a nonlinear SVM. 50

**Figure 4-3:** example of different kernels plotting 53

Figure 4-4: Grid search initialization demonstration.: 55

**Figure 4-5:** SVM transfer learning X linear kernel confusion matrix 59

**Figure 4-6**:SVM transfer learning X RBF kernel confusion matrix 60

Figure 4-7: SVM fine-tuning X linear kernel confusion matrix 61

**Figure 4-8:** SVM fine-tuning X RBF kernel confusion matrix 62

**Figure 4-9:** general comparison between all the approaches 63

**Figure 4-10**:descriptive scheme of bootstrapping and boosting 67

**Figure 4-11**:random forest with set A of hyperparameters confusion matrix 71

**Figure 4-12**:random forest with set B of hyperparameters confusion matrix 72

**Figure 4-13**:random forest with set C of hyperparameters confusion matrix 73

**Figure 4-14:** random forest with set D of hyperparameters confusion matrix 75

**Figure 4-15:** Visualization of a part of a tree in the random forest of set A 76

**Figure 4-16:** SVM & RF approaches side by side comparison 80

**Figure 5-**1: classification and segmentation structure 84

**Figure 5-2**: segmentation of a crack 85

**Figure 5-3**: segmentation of a dent 85

**Figure 5-4:**Confusion matrix of binary classification "defected/ non defected" 88

**Figure 5-5**:Confusion matrix of binary classification "corroded/ non corroded" 89

**Figure 5-6:** YOLOv8 results on a dented unseen image 90

**Figure 5-7**: YOLOv8 results on a cracked unseen image 90

**Figure 5-8**: AzureML evaluation interface 93

**Figure 5-9:** AzureML evaluation results interface 94

# Tables

**Table 4-1**:SVM transfer learning X linear kernel results: 58

**Table 4-2**: SVM transfer learning X RBF kernel results 59

**Table 4-3**: SVM fine-tuning X linear kernel results 60

**Table 4-4:**SVM fine-tuning X RBF kernel results 61

**Table 4-5**: approaches taken for Random Forest classifier 70

**Table 4-6**: random forest with set A of hyperparameters results 70

**Table 4-7**:random forest with set B of hyperparameters results: 71

**Table 4-8:**random forest with set C of hyperparameters results 73

**Table 4-9:**random forest with set D of hyperparameters results 74

**Table 5-1**: results of binary classification "defected/ non defected" 87

**Table 5-2**:results of binary classification "corroded/ non corroded" 88

**Table 5-3:** results of segmentation "crack & dent" 89

# General introduction

# General Introduction:

Transportation pipelines are a critical component in any industry and their safety affects the well-being of people and the progress of society.

For this particular reason we notice that the research methods of their maintenance improve day by day, and the competition in the market increases. We can even notice the start of the implementation of various and well advanced deep learning models to have more precise inspection results.

In the light of this technological evolution, this subject caught my attention, the intrigue and thirst in me to get to know more about the field combined with my interest in all what concerns artificial intelligence and its applications on various domains, drove me to take the step and make this dissertation to open a wider research axis in the future.

This project was realised in collaboration with a Tunisian startup Industry X.0. It is founded in 2019 that aims to accelerate the digital transformation of various industries and businesses that are active in different domains, by increasing the potential and efficiency of teams through technological solutions. It provides them with real-time information on their quality, safety and customer service for a more efficient, connected and collaborative market.

**Contributions:** The goal from this project, is to integrate machine learning specifically computer vision in the oil & gas industry. Since it is a very sensitive research field and work area, the project's main focuses are:

1. Develop different approaches and algorithms to detect defects in the external surface of the pipeline.
2. Evaluate the robustness of support vector machines and random forest on low data number with a challenging quality.
3. Finally take the previously mentioned evaluation into consideration and work on optimising the outcomes and illuminating the issues faced.

**Summary:**

This dissertation is split into 5 chapters:

**Chapter 1: pipeline inspection**, you will find in this very first chapter a general overview about pipeline inspection and how did it evolve through history and an insinuation of the importance of artificial intelligence in this field;

**Chapter2: Related work**, in this part we will be talking about the previous researches and efforts that were put into developing new technics to detect and classify defects in sewer pipelines and a study case on a petrochemical one.

**Chapter 3: Dataset preparation,** you will dive into the first step taken in building any AI model, which is the data related part, starting by the data collection, data cleaning and pre-processing to the data augmentation and feature engineering. The tools and methods used will be explained and mentioned in detail.

**Chapter 4: Machine learning model building**, this chapter talks about the different algorithms and different approaches taken in developing each algorithm. Their respective results and their interpretation will be brought up alongside the issues and constraints faced.

**Chapter 5: Optimisation and deployment**, in this very last chapter an optimisation architecture will be proposed with the method used to deploy the model and make I available for users to use.  And we finish by a general conclusion.

# Chapter 1: pipeline inspection

# Chapter 1: Pipeline Inspection

## 1.1. Introduction

In this first chapter we will be diving into the pipeline inspection. We will firstly go through its history and how did it develop to talking about photographic inspection and going through its aspects and applications.

## 1.2. The Evolution of Pipeline Inspection Through History

To transport vital resources like oil, gas and water over extensive distances, it is imperative that pipelines are operated safely and efficiently. This has led to a revolution of pipeline inspection methods over the years, aimed at enhancing accuracy, efficiency and safety.

- Early Days: Relying on the Human Eye (1920s-1940s):

In those early days when pipes were first used for this purpose, inspections were mainly visual exercises done by human beings who inspected pipe surfaces for leaks and cracks. This was the simplest form of pipeline inspection but its shortcomings were glaring due to its incredibility and missing crucial part of the inside of the pipes due to the lack of tools and the imprecise human eye.

- The Rise of Non-Destructive Testing (NDT) and In-Line Inspection (1950s-1970s):

Developments in this area such as radiography and ultrasonics were a major step forward. This meant that internal checks could be carried out by methods that would not cause damage, making it possible to have a more precise reflection on issues such as corrosion, cracks or blockages.

Also, the advent of intelligent pigs transformed pipeline inspection activities. These robots moved in the pipes using various sensors and techniques to identify something unusual. This development has drastically increased productivity and made an overall assessment of inner conditions of the pipeline possible.

- Progress and Advancements (1980-present)

It was in these decades that Pipeline inspection gained steady advancements:

- **Better Evaluation Techniques for Oil Pipelines:** Existing methods like ultrasonography and radiography were enhanced while new ones like Magnetic Flux Leakage (MFL) and Electromagnetic Acoustic Transduction (EMAT)were developed. These improvements enabled more accurate and sensitive defect detection.
- **Digitization and Automation:** This meant that computerised data acquisition, analysis or interpretation was merged with the NDT and In-Line inspection, thereby improving on data management, response time as well as decision making process during inspection exercises.
- **Remote Sensing Inspection and Monitoring:** The development of remote sensing technologies such as drones and fibre optic sensors has facilitated non-destructive testing of inaccessible or dangerous environments which further promotes safety together with efficiency.

## 1.3. Photographic Pipeline Inspection

### 1.3.1. Closed Circuit Television Video (CCTV) inspection

#### 1.3.1.1. General overview

CCTV is becoming the most used defect detection technique in engineering domains. It involves using a small crawling robot to take pictures within the metal pipe and identify the flaws seen in the footage.

CCTV is used to inspect and assess the conditions of the pipes to identify and determine if the existing concrete pipe is suitable for rehabilitation or replacement. Performance requirements and qualification criteria of the new pipe were set in specifications. The use of CCTV during the conditions assessment determined that the pipes were still structurally sound. The concrete liner inside the pipes showed some areas with concrete spalling; however, it was determined that the cracks were not detrimental to the hydrologic performance of the pipes. Furthermore, existing deteriorated concrete pipes have a total of 35 joints (grouted spigot and socket but no rubber O-ring), with a maximum separation of six feet.

The use of closed-circuit television (CCTV) equipment dates back to the early 1960s when it first appeared as a means for security. The device proved useful and made it a business in which the demand for the said equipment grew rapidly worldwide. Based in Milton Keynes, UK, writer and director Tony Scott of TVS Ltd. in 1973 led a team of engineers to design the world's leading pipeline CCTV inspection vehicles. This was the company that invented CCTV for water inspection, and later several other companies started to be in the same field. The world's leading pipeline CCTV inspection vehicles came to be known as 'Renovator'. In production, more than 1700 'Renovators' have been manufactured and are operating worldwide as a direct result of Tony Scott's pioneering efforts to create TVS as a successful, well-respected company in the CCTV pipeline inspection industry.

***Figure 1-1***: *industriel pipe inspection crawler camera robot*

***Figure 1-2***:*HVAC Duct Inspection*

#### 1.3.1.2. CCTV uses:

This inspection method is a more cost-effective and suitable substitute for the hazardous and conventional method of using humans to inspect pipelines, particularly in cases when the pipeline is too narrow or risky for humans to enter. CCTV cameras continuously capture images and video of the pipeline's interior above the flow-line. Experts typically utilise the films and recordings they have obtained to inform their interpretation and determination of the pipeline's condition [8].

Unlike methods that require digging or breaking the pipeline, CCTV is a non-destructive way to assess its health. This minimises disruption and keeps repair costs lower.

Defects such as various forms of cracks and fractures, settled deposits, deformation in the pipeline cross section, wall collapses, misplaced joints, polished aggregates or surface abrasion, and tree root penetration can all be found with CCTV. Every pipeline is evaluated and assigned a rating according to the frequency of fault occurrences as well as the seriousness of each fault.

The operator scores each pipeline according to this rating, typically on a five-point scale [24]; nevertheless, numerous researchers have identified a number of drawbacks with this approach. According to Tuccillo et al. (2010), CCTV is limited to providing data above the flow-line. More significantly, it doesn't put a number on the identified flaws, which include surface damage, settling deposits, and deformation.

As a result, the researcher came to the conclusion that CCTV inspection is solely used as proof to find flaws.

#### 1.3.1.3. Practical use domains of CCTV:

CCTV pipeline inspection is used in a number of different industries:

- **Oil and Gas Pipelines**: To guarantee the secure and effective transportation of oil and gas, routine CCTV inspections are essential. They are able to identify obstructions, leaks, and corrosion that can cause harm to the environment or interfere with operations;
- **Chemical Pipelines**: Because the contents they transport are toxic, chemical pipelines need to be carefully inspected. CCTV aids in the detection of leaks, obstructions, and deterioration of internal linings to stop environmental contamination and safety risks;
- **Water Pipelines**: It's critical to have clean water sources available. Water pipeline corrosion, mineral accumulation, and leaks that may affect the quality of the water are found through CCTV inspections.

### 1.3.2. Halliburton's visual inspection of pipelines and processing plants

#### 1.3.2.1. General overview:

Before containment can be broken, inspecting the inside surfaces of pipelines and processing plants typically necessitates complete decontamination, which adds time and

the price of upkeep tasks. The initial inspection process can be optimised by using cameras, which lessens or eliminates the need for mechanical involvement [11].

Cameras can be moved around corners, across long distances, and have several camera heads to give a 360-degree picture of interior surfaces.

#### 1.3.2.2. Applications:

- Heat Exchanger tubing and shell inspection;
- Small vessel and tank inspection;

- Pipework and duct inspection;
- Flexible and rigid riser inspection;
- Caisson, cavity and machinery inspection.

#### 1.3.2.3. Features:

- Can be used in both hazardous and non-hazardous areas;
- Suitable for use in live process plants;
- Digital recording for immediate playback;
- Annotation and voiceover of video and still footage;
- Files compatible with Halliburton Integrity Management Software.

#### 1.3.2.4. Benefits:

- Requires little mechanical preparation and offers evidence of cleanliness prior to beginning operations;
- Provides location and severity details, improving repair work and preparations Monitoring and assessment of cleaning and remediation in real time.

#### 1.3.2.5. Inspection cameras types:

- *Push Rod Cameras:* There is a "stiff fibre rod" attached to these cameras. The rod is rigid enough to allow it to be pushed a considerable distance down a pipeline, but it is flexible enough to be stored on a reel, uncoiled, and then deployed into the pipeline. A rod counter is built into the camera systems so we can precisely track the location and distance of the camera. In addition, the cameras allow for easy digital recording for later playback and record

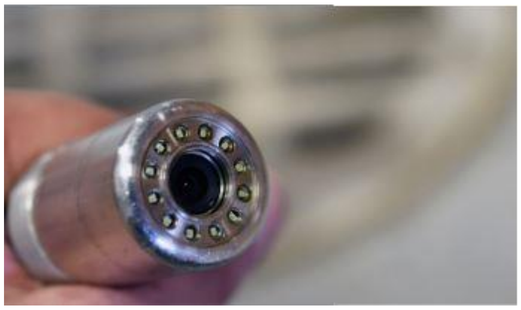


***Figure 1-3**: Push Rod Cameras:*

keeping, as well as direct sight of the interior surfaces. Because the "camera heads" are waterproof, they can be utilised in liquid-filled systems.

- *Endoscope Cameras:* When the item or entry point has to be inspected has a small diameter, Halliburton offers cameras that are suitable for these kinds of circumstances. There are two types of camera heads available: radial and forward-facing. This kind of camera features articulation functionality that enables us to look both upstream and downstream while conducting localised inspections of bigger pipelines using small bore instrument connections.

***Figure 1-4:*** *Endoscope Cameras view inside a pipeline*

- *Hal-Vision Camera:* This camera inspection system was created to examine the flexible risers' interior surfaces. The camera head can be pulled down and has sealing discs installed. fluid-pressurised riser or pipeline (similar to an inspection or cleaning pig). Through the use of fibre optic connections, the unit's high-definition camera may be controlled remotely. A single point of entry can be used to investigate an area up to three kilometres away.

***Figure 1-5:****Hal-Vision Camera*

## 1.4. Photographic inspection limitations:

Visual based inspection had its own limitations through the years due to some challenges such as:

- lighting and Visibility: Achieving proper lighting conditions for clear and detailed photographs within the narrow and potentially dark space surrounding the buried pipeline is extremely difficult;
- Image Interpretation: Even if captured, interpreting the images accurately is challenging due to factors like limited visibility, potential presence of debris, and the complexity of analysing external pipe conditions solely based on photographs;
- Safety and Cost: The safety risks associated with excavating and manipulating machinery around buried pipelines, coupled with the high costs involved, make photographic In-line inspection a viable option.

The previously stated issues faced in the vision-based pipeline inspection and with the development of technological optical or photographic tools combined with deep learning algorithms revived such methods and pushed the researchers to pursue such things.

## 1.5. The urge for AI models integration in defect detection:

Vision-based defect identification is a key component of industrial intelligence and is a necessary technology in contemporary production to ensure product quality. The advancements in industrial big data have made it possible for omnipresent sensors to gather defective photos. Furthermore, accuracy recognition has emerged as a popular area of study.

Numerous vision-based defect identification algorithms have been offered in the last several years, and some recently developed methods like deep learning have gained popularity and successfully solved a number of difficult situations.

As computer vision technology advances, industry and academic interest in vision-based defect recognition has grown. Vision-based defect recognition is a quick, affordable, and reliable method using computer vision algorithms. As a result, it is extensively used in a variety of industries, including steel, wood, ceramics, textiles, and architecture.

The goal of vision-based defect detection is to determine whether the provided pictures contain any faults as figure 1-7 represents. This procedure typically includes data pre-processing, feature extraction, and recognition. The goal of data pre-processing is to gather, clean, and standardise the pictures of defects. Generally speaking, recognition involves matching, detection, segmentation, and classification. Classification identifies the different sorts of defects, segmentation separates regions with and without defects, detection indicates the defect's position, and matching locates the template that most closely resembles the original. The key to all of these recognition jobs is feature extraction. Defect recognition performance may be improved using a skilful feature extraction technique, statistical approaches, structural methods, filter-based methods, and model-based methods that are the traditional categories into which vision-based defect identification techniques fall.

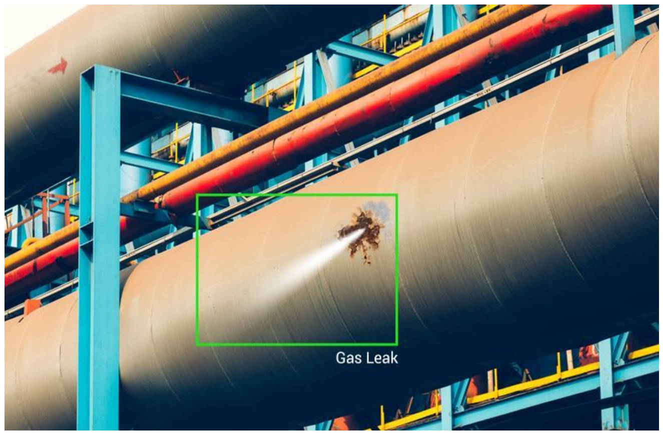


***Figure 1-6***: *Gas leakage detected by computer vision*

## 1.6. Conclusion

This chapter's focus was mainly to highlight the pipeline inspection in general through history and how it did evolve.

It tackled the development of different technologies and different methods to inspect precisely the transportation pipelines. One of the main methods that captured my interest and has a relation with the theme of this project is photogenic inspection.

As previously described and shown, photographic inspection also had its portion of development and advancement from only human's eyes view to well-developed robot ones. Even all the changesets that touched this sector still it has its own constraints, and by the inclusion of Ai in all aspects of life currently, the cruciality of developing new AIs in such domain is brought to light.

# Chapter 2: Related work

# Chapter 2: Related work

## 2.1. Introduction

Automation and the application of machine learning techniques is becoming more and more common in industry, especially in the maintenance and inspection of different terrains.

One of the latest researches and efforts made is to better the quality and the security of the subterranean infrastructure systems particularly, since they represent a crucial part in the human daily life and the ecosystem.

On another hand, the oil & gas industry has also a huge role in countries economics and providing a better transportation of petrochemical substances not only will elevate the country's economic and financial state but also grants a safer and a cleaner transportation on the environment, since hydrocarbons have been and still considered as the most pollutive.

In this chapter multiple papers of different researches have been presented that have been published yet not most of them are being applied due to the sensitivity and the complexity of the pipes networks.

## 2.2. Pipeline technology inspection methods

Several substitute technologies that can be used for pipeline inspections have surfaced: Sonar, laser, and closed-circuit television (CCTV) scans, among others. Pictures and videos gathered with very detailed condition information are provided during pipe inspections.

When combined with other pipe condition data, such as flow, gas information, and other quantitative measurements, municipalities can use these visual data as visual aids for maintenance decision-making, in addition to technicians or inspectors using them during and after pipe inspections.

As a result, the most often used technology for pipeline inspection is CCTV. In addition to single-sensor inspection methods like the traditional CCTV inspection described above, a number of smart pipe evaluation systems integrate numerous sensing technologies into the inspection process.

Even though CCTV is currently the most often utilised visual inspection technique in industry, human operators still view the photos and videos taken during inspection in order to identify and categorise defects. According to the standard pipeline condition grading system, human operators must be taught and qualified in order to recognize pipe faults by watching inspection films and/or photographs.

This current pipeline inspection and data interpretation practice is labour-intensive, error prone and time consuming.

### 2.2.1. Automated defect detection for sewer pipeline inspection and condition assessment from videos and images

#### 2.2.1.1. Research motive

The state of automated pipeline inspection and condition assessment must be advanced in order to routinely and proactively evaluate the conditions of sewer infrastructure systems in order to guarantee their structural integrity and uninterrupted services. Addressing realistic flaw detection that works with actual sewer inspection data is currently a crucial concern.

#### 2.2.1.2. General overview

The existing inspection methods limited the assessment of sewer pipeline conditions until recently. Numerous issues, including backups, leaks, and total loss of service, were brought on by the absence of frequent condition inspections and the delay in inspection cycles [4].

Such widespread degradation would have been detected, repairs would have been begun, and more serious and expensive damage would have been avoided with a prompt, accurate, and thorough inspection.

Municipal authorities are mandated by the **US EPA** to evaluate the state of their collection systems in order to have a better understanding of issues related to maintenance and operation.

The US EPA's requirement made it necessary to obtain current, trustworthy pipeline status data using appropriate and efficient techniques. It also made it necessary to evaluate the data efficiently, such as by identifying and categorising pipe faults.

The term **"defects"** in this research refers to either those anomalies, like areas of corrosion and cracks, or important pipe patterns or landmarks, like couplings and connections. The sewer pipeline inspection and condition assessment criteria published by **NASSCO** are consistent with this definition of a problem.

#### 2.2.1.3. Pipeline Internal Defect Detection Based on Computer Vision

Several automated detection systems have been proposed for pipeline defect identification. Moselhi et al [20] proposed a system with four modules: image acquisition, image processing, feature extraction, and defect classification, laying the groundwork for a computer vision-based pipeline internal defect detection system. Yang and Su [26] combined OTSU thresholding with morphological opening operation to segment CCTV video frames and created models for common defects. However, their approach is limited by strict guidelines regarding lighting conditions and filming perspective. Myrans et al [21] collected GIST features and applied the random forest algorithm, supplemented by the Order Oblivious Filter (OOF) and Hidden Markov Model (HMM) to enhance prediction accuracy, albeit at the expense of time efficiency. Halfawy et al [6] utilized morphological, threshold segmentation, and median filtering techniques, focusing on specific flaw types such as tree root intrusion-related pipe faults. Guo et al [5] proposed a change detection method involving image alignment, histogram matching, and image filtering, achieving high accuracy despite a relatively high false detection rate.

#### 2.2.1.4. Pipeline Internal Defect Detection Based on Deep Learning:

Several advancements have been made in pipeline problem identification using various deep learning techniques. Hassan et al [6] incorporated AlexNet into their approach, utilizing altered photos from CCTV footage of pipelines for training data. Wang and Cheng introduced DilaSeg-CRF, an end-to-end neural network combining deep convolutional neural networks (CNN) with fully connected conditional random fields (Dense CRF), with jointly trained segmentation and CRF layers parameters.

Subsequently, numerous CNN-based techniques have emerged. However, Kumar et al [19] observed limitations in structural flaw identification when comparing the Faster R-CNN model with YOLO v3 and Single-Shot Multibox Detector (SSD). On the other hand, Rao et al [23] demonstrated improved detection accuracy by combining CNN with non-overlapping windows, with findings indicating that deeper CNN models perform better albeit at the cost of longer inference times. These studies collectively contribute to the ongoing development of efficient and accurate pipeline defect detection systems.

### 2.2.2. Automatic Defect Detection System for Petrochemical Pipelines

#### 2.2.2.1. General overview

For the safety of industrial production, petrochemical pipeline defect identification is a crucial responsibility. Currently, the primary approach for pipeline defect detection is closed circuit television (CCTV), which records video of the pipeline's inner wall before manually identifying the problematic location. This method is labour-intensive and has a high false and missed detection rate [3].

#### 2.2.2.2. Proposed methods

##### 2.2.2.2.1. Images shooting

The pipe-climbing robot travels axially forward within the pipe to be investigated at a pace of three metres per minute. It is outfitted with a lighting source and camera equipment. At a frame rate of 15 frames per second, the robot can capture 180 metres of video in an hour, with a resolution of 2888 x 2888 pixels per frame. The inspection report provided by qualified pipeline maintenance staff is used to classify and label the pipeline inner wall fault area. The utilised pipeline video data mostly contains four types of surface faults; Figure 2-1 illustrates examples of the various categories.

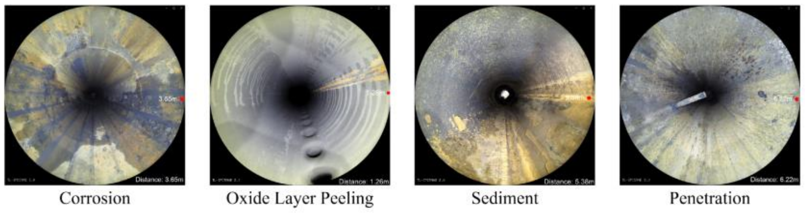


***Figure 2-1**:images of different types of defects*

In this paper, the original data set consists of 21 videos shot during the maintenance of the pipeline, including drainage pipes, distillation unit pipes, catalytic fractionation tower pipes, and reforming collection pipes.

#### 2.2.2.2.2. Image unfolding

The original ring picture of the inner wall of the pipeline must first be unfolded in order to precisely identify the problematic area and restore the distorted image.

The image of the pipeline's inner wall displays characteristics such as darker inner wall, brighter inner wall, and black corners because the pipeline's data are mostly lighted by its own light source during photography. The threshold segmentation method divides the pipeline's inner wall, centre, and four corners, while the morphological method partially eliminates noise.

The Hough transform is used to identify the circular region and get the pipeline's centre coordinate $O1(u0,v0)$. Given the properties of the shooting data, the outer circumference of the ring is tangent to the image's edge; so, the outer radius of the ring is $r=L/2$, where L is the size of each image frame.

#### 2.2.2.2.3. Luminance correction and feature engineering

Since there is no lighting inside the pipeline and the light is generally dim, the pipe-climbing robot uses its auxiliary lighting device when taking images inside the petrochemical device's pipeline. This helps prevent uneven illumination and even serious reflection in some areas of the inner wall image that is taken.

The image is initially unfolded circularly in order to address the issue of unequal illumination in the image. Next, the multi-scale Retinex with chromaticity preservation (**MSRCP**) is applied to address the image distortion problem brought on by the hemispherical camera firing in the pipeline.

An image stitching method based on **SIFT** features is used to stitch consecutive frames to create a fully unfurled image of the pipeline inner wall in order to fully utilise the context information in the video.

Since the dataset wasn’t sufficient for the deep learning model, in order to increase the sample size by **5.5** times and randomly produce the defect patches on the original data, a sample enhancement technique based on **Cycle-GAN** is suggested.

###### 2.2.2.2.4. Defect Detection Based on YOLO v5

After stitching the photos of the inner pipeline wall, a deep learning-based object identification algorithm is used to find and identify the pipeline inner wall fault. Object detection, which is frequently used in face identification, autonomous cars, and other domains, is the process of locating and classifying one or more objects of interest in images and determining their category and location.

In this research, they applied the **YOLO v5** object detection method to the pipeline surface defect detection problem and obtained satisfying results.

#### 2.2.2.2.5. Structure of YOLOv5

The fundamental principle of YOLO is to directly regress the positions and categories of the bounding boxes at the output layer by feeding the full image into the network, which is akin to Faster-RCNN. Here is the precise procedure:

(1) The picture is separated into cells in the grids, which are used to anticipate the objects when the centres land there.

(2) B bounding boxes are projected for each grid cell, with a total of five values predicted for each bounding box, depending on $(x,y,w,h)$ and confidence.

(3) A total of C categories is anticipated for the category in each grid cell. The final output tensor, which predicts the locations and categories for B bounding boxes of $S{\times}S$ grid cells, is the $S{\times}S{\times}(5{\times}B{+}C)$ dimension. Figure 2-3 displays the YOLO v5 network structure.

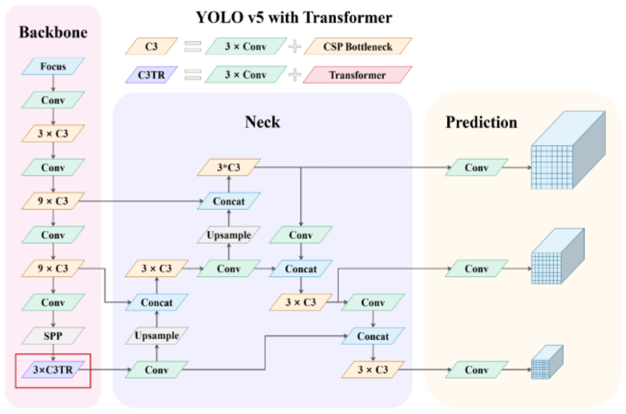


***Figure 2-2***:*YOLOv5 network structure*

## 2.3. Conclusion:

In this chapter various methods and approaches found in literature for pipeline defect detection based on image classification were highlighted.

However, literature is rich in various algorithms and datasets, yet none of them is applicable. The transportation and the piping field has a huge role in crucial industries, they are critical, very sensitive and can even be costly. It takes more intensive applications and deeper analysis for the realised models and techniques to be applied.

# Chapter 3: Dataset preparation

# Chapter3: Dataset preparation

## 3.1. Introduction

In this chapter all that concerns dataset preparation is going to be mentioned. The dataset is a crucial part in any model building and training. In order to have the most refined and well-structured dataset you need to pass by multiple steps so it gets ready to be fed into your specific model as figure 3-1 shows.

To begin, gather data from multiple sources, such as spreadsheets, databases, and APIs.

The data is then cleaned by eliminating the unnecessary images and get reordered in folders of train, test and validation, split into their adequate classes.

Next, in order to prepare the data for use with machine learning algorithms, you alter it using procedures like encoding and normalisation.

Lastly, you use methods like dimensionality reduction or feature engineering to minimise the complexity of the data while retaining the information that it may offer the machine learning model.

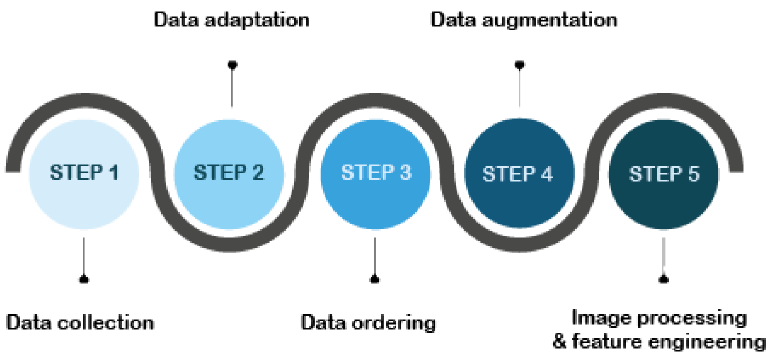


***Figure 3-1**: data preparation steps.*

## 3.2. Dataset creation constraints

The project focuses on developing a new approach using computer vision to detect and classify various types of defects in the external surface of a transportation pipeline.

However, it is important to shed light on the fact of the lack of data, computer vision is a domain that works with multidimensional data (2D images, 3D images, videos…Etc), that's where the first challenge resides.

Companies in the oil & gas industry don't provide direct images of their pipelines and use mostly ultrasonic waves to inspect their health. This pushed me to using alternatives and creating a full database based on deep searches and collecting, more details will be provided in the following parts.

## 3.3. Image collection

Collecting the adequate images to be fed into a machine learning model is a constraint itself, it takes deep searches into various sources and filtering to select the data that will give you the most optimal results.

### 3.3.1. Image collection sources

The images used to build this project were extracted from open sources:

- **Google search**: it is the commonly used search engine, it helps users find information easily and quickly and orients you towards helpful and useful websites;
- **Shutterstock[17]:** A leading platform offering millions of royalty-free stock photos, videos, and illustrations under subscription plans or individual purchases. Popular for commercial and creative use;
- **Istock[15]:** Similar to Shutterstock, iStock offers a vast library of high-quality stock photos, videos, and illustrations. Known for a curated collection and focus on premium content;
- **Pexel[16]:** Provides a large collection of high-resolution stock photos and videos available for free download under a permissive licence. Popular for personal and non-commercial use;
- **Flickr[12]:** Primarily a social networking platform for photographers, but also contains a vast collection of images and videos uploaded by users. Some users offer their content for free use under specific licences, while others retain copyright. Requires careful licence checking before using content;
- **Gemini Generator[13]**: A new and emerging service that uses artificial intelligence to generate images based on text descriptions. Currently in beta testing, it allows users to create unique visuals based on their prompts;
- **google scholar[14]:** Provides a comprehensive search engine for academic research, targeting scholarly articles, theses, books, abstracts, and court opinions.

### 3.3.2. The total number of the collected images

Non defected: 100 images

Defected:

- Cracked: 50 images;
- Dented: 50 images;
- Corroded: 50 images;

**Total:** 250 images.

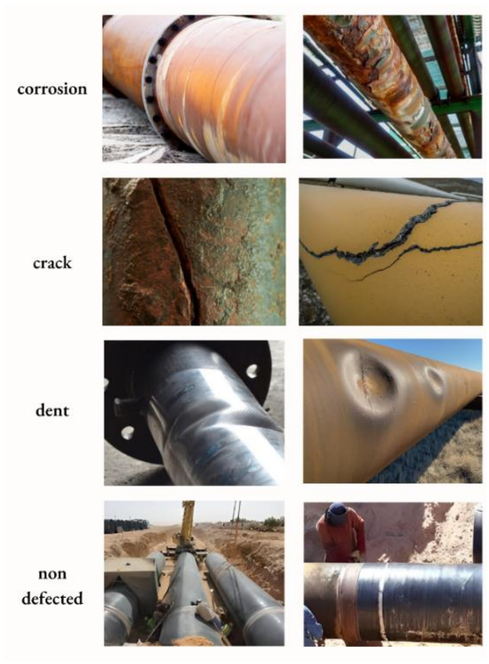


***Figure 3-2**: examples of the collected images from each class*

## 3.4. Image adaptation

- **Defect cropping:** after the images were collected they were generalised images. In order for our model to perform well on our classification task the specific defect in the image was cropped.

  NB: this process was done manually so the cropped area would be precise.

- **Quality enhancement(image upscaling):**

Images collected from open sources aren't always ones of good quality. The credibility of the Ai model depends on how much it could learn the most important feature. If the images are of a low quality our model won't be able to detect the defect and this will lead to an inability to learn.

The technique of raising a digital image's resolution is known as image upscaling. To put it another way, it's about enlarging an image. However, expanding the image alone would only produce a jumbled, fuzzy image. The goal of upscaling approaches is to produce a crisper, higher-quality image with more detail by analysing the current pixels and creating new ones that fill in the gaps using different algorithms.

There are many free websites that can be used in this process such as: zyro / Fotor / AI.imagesenlarger / upscayl / image enhancer / M.image enhancer.

The one used for this specific project is **Fotor**. It is a free website that allows various operations on the images, it is very easy to use and it gives very good results as an output.

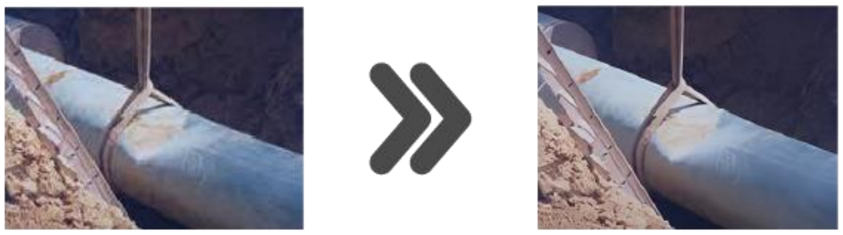


***Figure 3-3**: A dented pipeline image before and after upscaling*

## 3.5. Data ordering

- **Based on classes:** our images should be in a folder containing sub-files, named with their associated class.

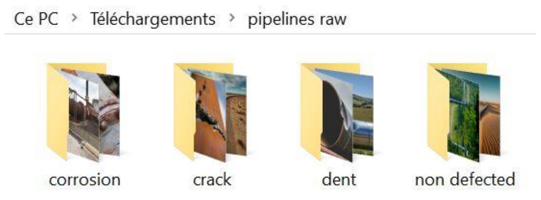


***Figure 3-4**:class based ordering*

- **Based on nature:** the data collected should be split into training data, test data and validation data. The training data will be augmented in the next step so we generate a larger number of images meanwhile the test and validation ones shouldn't be, the tes images are used to evaluate the performance of our model and the validation ones to double check.

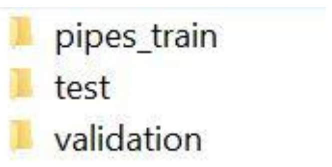

pipes_train
test
validation

*Figure 3-5: nature based ordering*

- Training: 160 image
- Test: 45 image
- Validation: 45 image.

## 3.6. Data augmentation

Having only 160 images for training was clearly not enough, which led me to apply data augmentation operations on my images to boost the number and have more varied data that my model will learn from. For this, I have used the ImageDataGenerator library from TensorFlow keras API.

- ImageDataGen [18]: Data augmentation involves applying arbitrary transformations such as rotations, flips, shifts, zooms, and other effects to generate variations of training images artificially. This practice is instrumental in mitigating the overfitting issue by expanding the quantity and diversity of data encountered during training. Through exposure to a wider range of examples, the model learns to generalize better on unseen data, identifying underlying properties that remain consistent across various distortions or variations. Consequently, data augmentation enhances the robustness and generalization capability of machine learning models, enabling them to perform more effectively on real-world datasets.

- Operations applied in this project:

- <u>Rotation_range</u>: Rotates the image by a random number of degrees within the specified range (in degrees). This helps the model learn to recognize objects from different orientations;
- <u>brightness_range</u>: varies the image brightness at random by a scaling factor that falls under the given range (tuple of two floats). This makes the model more resilient to changes in illumination;
- <u>Width_shift_range</u>: Randomly shifts the image horizontally by a fraction of its total width within the specified range (float or tuple of two floats). Values can be less than 1.0 to shift to the left or greater than 1.0 to shift to the right. This helps the model learn features that are not dependent on the object's position within the image;
- <u>Height_shift_range</u>: same as width_shear_range except for height;
- <u>shear_range</u>: Applies a shearing transformation to the image. This can distort the image slightly, helping the model learn features that are invariant to small geometric distortions. Specified as a float representing the shear angle in radians;
- <u>Zoom_range</u>: Randomly zooms in or out on the image within the specified range (tuple of two floats). Values less than 1.0 zoom in, and values greater than 1.0 zoom out. This helps the model learn features at different scales;
- <u>Horizontal_flip</u>: Randomly flips the image horizontally (mirroring the image left to right). This can help the model learn features that are independent of the object's orientation on the x-axis;

- **Fill_mode**: Specifies how to fill the pixels that become exposed due to random transformations like rotations or shifts.

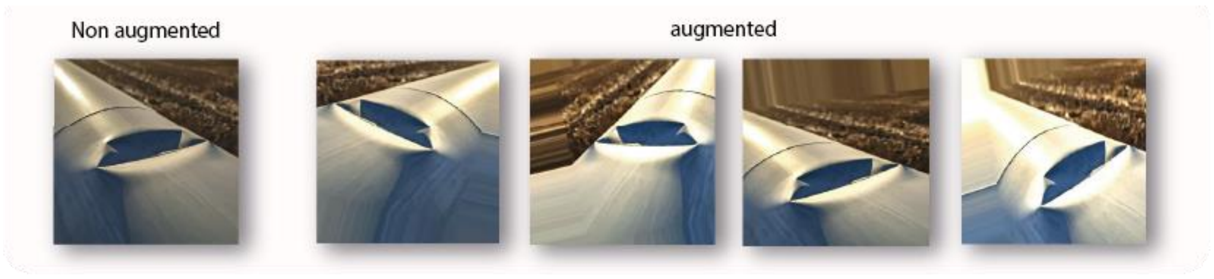


***Figure 3-6**: representation of data augmentation process*

- ➔ After applying the methods mentioned above my built model was regulated to generate for each image 30 other ones.
- ➔ Total number of images after augmentation: 7540 images.

## 3.7. Image processing and feature engineering

How do we unlock the information from images? This part is the answer to this question.

Machine learning models have their own specific language, they are unable to read images as they are, that's why we have to transform these images into numerical data and arrays in order for it to be read. The steps taken in these project are represented in the following figure 3-7

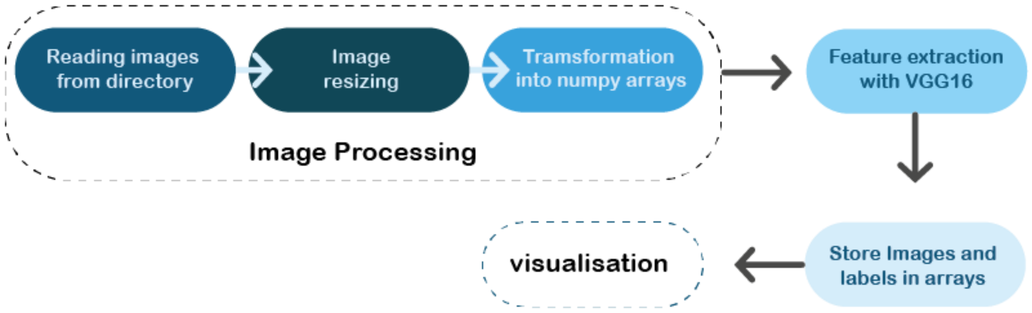

Reading images from directory
Image resizing
Transformation into numpy arrays
Image Processing
Feature extraction with VGG16
Store Images and labels in arrays
visualisation

***Figure 3-7:*** *steps taken in image processing*

### 3.7.1. Image processing

This step concerns bringing images from visualised pixels and colours into floats and arrays.

In this project I created a function called load_and_preprocess_images_with feature_engineering that operates all that is going to be explained, by calling this function on a certain image folder it directly applies all the necessary transformation and it stores the numerical information of the images in images array and labels in label array. This function is built as follow:

#### 3.7.1.1. Reading images from directory

In this part my code will iterate over all the training images folder and each sub-file and read the images.

The libraries that had a role in this step are:

➔ **Operating system module** (OS): [27] This module offers an operating system-dependent functionality in a portable manner. See open() if you only want to read or write a file; see the os.path module if you want to alter paths; and see the fileinput module if you want to read every line in every file on the command line.

See the tempfile module for producing temporary files and directories, and the shutil module for high-level file and directory operations.

#### 3.7.1.2. Operations on the image

In this step the images get resized to a 224px by 224px to make them easy for the next step, and transformed into a normalised numpy array that holds values that vary from 0 to 1. The libraries used are:

➔ **OpenCV** (cv2): [10] For real-time computer vision and image processing, cv2, sometimes referred to as OpenCV (Open Source Computer Vision toolkit), is a potent and extensively utilised open-source toolkit. Although it's mainly built in C++, it offers bindings for Python (via the cv2 module) and other languages.

Its functionality is to execute a variety of image and video processing tasks, such as: Filtering (blurring, edge detection, noise reduction), transformations(resizing, rotating, converting colour space);

➔ **Numpy arrays:** [9] A flexible data structure for storing and managing multidimensional arrays of homogeneous data types (e.g., all integers, all floats) is introduced by NumPy: the ndarray object. Compared to Python's built-in lists, which are less effective for numerical operations and can hold diverse data types, this is a major gain**.**

### 3.7.2. Feature engineering with VGG16 (Visual Geometry Group from Oxford)

VGG16 is a pre trained deep learning model, built by a group of Oxford experts, trained on the various dataset ImageNet. It can be used as a classifier on its own or as a feature extractor in transfer learning.

13 convolution layers are stacked together in the VGG16 architecture to create an image categorization system. A kernel of dimension 3×3 with parameters that are learnable W and b is used to conduct the convolution process, passing it over each image's pixel x to produce an output y. The stride determines whether the kernel moves pixel by pixel or skips multiple pixels at a time. The function that represents the convolution operation's simplified form is as follows:

$$Y = f\ (\omega_{x} + b) \qquad (3\text{-}1)$$

The automatic feature extractor that the convolution layers provide extracts patterns for classifying different defects. Simple characteristics like edges are learned by the initial convolution layers, and these features are combined to generate complex features in the subsequent convolution layers. Rectified Linear Unit (ReLU), a non-linear activation layer that adds uncertainty, usually comes after each convolution layer. The maxpooling layer is used for down sampling in order to minimise the size of the activation map. At the top of this convolution layer stack is a classifier. It is a fully linked layer with 4096 neurons in this instance. Three fully connected layers are present with four neurons, one for each class. In this project two approaches are used:

➔ Feature extraction (transfer learning)

- **Pre-trained Model:** For image classification, for example, we use ImageNet, a robust model that has been trained on a sizable dataset. This model has picked up useful general traits that it can use for a variety of tasks;
- **Feature Extractor:** We just employ the pre-trained model's first layers, which are typically convolutional layers in image processing. These layers are useful for a variety of computer vision tasks since they capture generic and low-level properties like edges, forms, and textures;

- **New work Data**: For the particular work that interests us, such as dog breed classification, we have our own dataset. It's possible that this dataset is smaller than the pre-training one;
- **Feature extraction** involves running the freshly acquired task data through the trained model, up to a selected layer. These first layers extract features (e.g., visual features from photos) pertinent to the general domain;
- Benefits:
  - <u>Leverages Pre-trained Knowledge</u>: The pre-trained model is a useful place to start, particularly if there isn't much data available for the new task. Extracted features frequently contain important information that helps with the new assignment;
  - <u>Shortens Training Time:</u> Pre-trained features save computational resources by avoiding the requirement to train the entire model from scratch.

➔ Finetuning:

- **Pre-trained Model** (Akin to Feature Extraction): On a similar task, we apply a pre-trained model. The pre-training work should be as near to our new task as possible;
- **Unfreeze Certain Layers**: We let the layers—often completely connected layers—to be fine-tuned rather than freezing all of the pre-trained layers as we would in feature extraction. Usually, these layers capture more information unique to a given activity;
- **New Task Data:** We train using our newly acquired task data;
- **Fine-tuning**: We train the entire model, but the pre-trained layers learn at a significantly slower pace than the final layers. This lets the model adjust to the new task while guaranteeing that the previously learned information isn't entirely replaced;
- Benefits:

- Adapts to New tasks: By enabling the model to modify its pre-trained features for a given job, fine-tuning can possibly outperform feature extraction, particularly for tasks with greater variability;
- Leverages Pre-trained Knowledge: It makes use of the pre-trained model's knowledge, just like feature extraction does.

➔ Choosing between feature extraction and fine-tuning:

- **Data Size:** Feature extraction may be preferable to overfitting if you have a very small dataset for the new task. More data is needed for fine-tuning to enable effective adaptation;
- **Task Similarity:** Feature extraction may be sufficient if the new task and the pre-training task are substantially similar. Fine-tuning is perhaps more advantageous for more different jobs because it enables the model to adjust the features that have already been trained;
- **processing Resources:** Compared to fine-tuning, feature extraction often uses less processing resources and is faster.

### 3.7.3. Data visualisation

In this project's case, 4133 features were extracted in order to have an overview on my data. I had to pass it through a relevant features selection process. I extracted 10 features using the chi-squared statistical method and visualised only two.

- How chi-squared helps in feature selection for visualisation:
  - ➔ Chi-squared equation:

$$x^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \qquad (3\text{-}2)$$

  - ➔ Assesses Dependency:

Chi-square evaluates the statistical relationship between a target variable (the variable you're attempting to forecast) and a categorical characteristic;

Strong reliance is shown by a high chi-square value, which implies the characteristic may be informative for the prediction job;

Weak reliance is indicated by a low chi-square value, which reduces the feature's informativeness.

- ➔ Ranking of Features:

You can order the characteristics according to how dependent they are on the target variable by computing the chi-square statistic for each feature and its p-value (significance level);

Features that exhibit low p-values, which signify statistical significance, and high chi-square values are deemed to be more pertinent.

- ➔ The threshold for selection:

To choose a subset of features, you can choose a threshold for the p-value or chi-square value;

Features that are deemed informative and kept for model training are those that are above the threshold (top k features with highest chi-square, for example). Features that fall short of this cut off are ignored.

- Visualisation:
  - ➔ Class 0: corrosion;
  - ➔ Class1: crack;
  - ➔ Class2: dent;

➔ Class3**:** non defected.

Figures 3-8 and 3-9 show the visualisation of this project's features fine tuned visualised all together and separately, while the figures 3-10 and 3-11 show the raw features without any fine tuning.

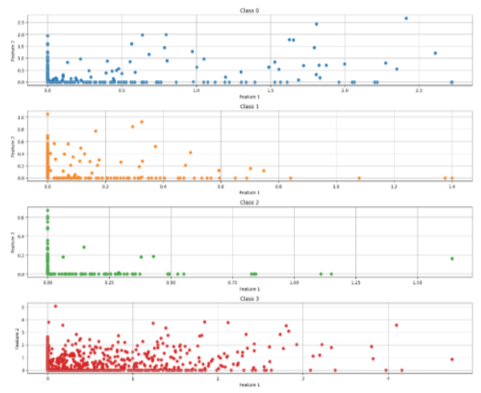


***Figure 3-8*** *fine tuned features representation (each class separately)*

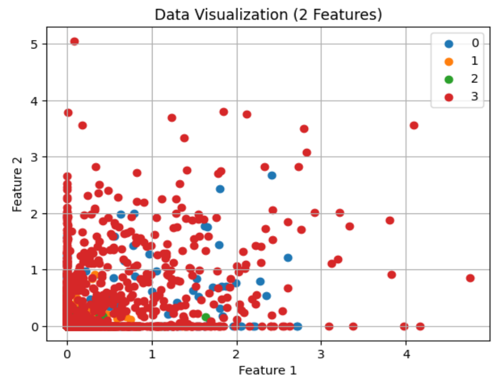


***Figure 3-9*** *fine tuned features representation (all classes together)*

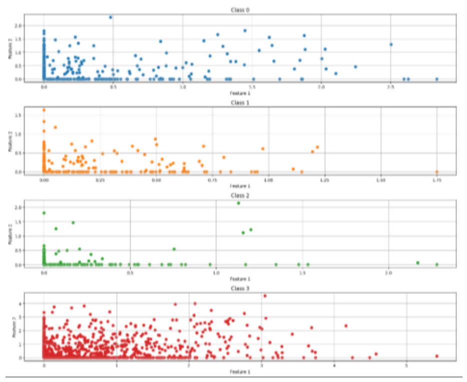


***Figure 3-11:*** *fine tuned features representation (each class separately)*

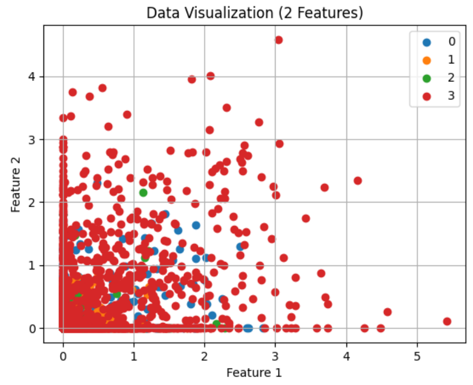


***Figure 3-10****: fine tuned features representation (all classes together)*

### 3.7.4. Storing data in arrays

Storing your processed images in arrays is a crucial step in order to jump to the next one.

In machine learning projects numpy is the one widely used. Why use numpy specifically?

➔ NumPy arrays provide a small and effective means of storing numerical data, such as pre-processed picture data.

➔ They make it possible for the data to be mathematically processed efficiently, which is essential for machine learning algorithms.

➔ They offer a uniform format that many machine learning libraries and tools can readily load and work with.

## 3.8. Conclusion

Data processing is a basic step in the field of machine learning that allows your data to reach its full potential. In conclusion, the following highlights its significance:

- **Extracts Useful Information:** Unprocessed data is frequently unstructured and devoid of the organisation required for machine learning models. Techniques for processing data that convert unprocessed data into a format that models can comprehend and use efficiently include feature extraction, normalisation, and cleaning.
- **Enhances Model Performance:** Accurate models that learn patterns and relationships are those that use clean, well-processed data. Better predictions, classifications, or other intended outputs from your machine learning system will result from this.

- **Reduces Training Time:** Data processing lowers the complexity of the data a model must handle by removing inconsistent and unnecessary information. This can greatly reduce the amount of computational resources needed and training timeframes.
- **Facilitates Feature Engineering Done Right:** Processed data is frequently used in feature engineering, the process of developing new features from pre-existing ones. The ability of a model to learn and generalise can be improved by identifying informative characteristics with the aid of processing techniques.
- **Improves Generalizability:** Properly handled data can aid models in learning from the training set and improving their ability to generalise to new examples. For real-world applications, where models must function well on data other than the training set, this is essential.

Data processing essentially serves as a link between strong machine learning models and unprocessed data. Through the act of converting data into an appropriate format and fixing errors, data processing enables models to learn efficiently and produce insightful predictions. In the next chapter we will be reviewing how this data and its types of pre-processing affect the model robustness.

# Chapter 4: Machine learning model building

# Chapter4: Machine learning model building

## 4.1. Introduction:

Machine learning and data mining methods are traditionally designed to tackle the problems separately. These algorithms are used for training the model separated on the same distribution and particular feature space.

A machine learning algorithm is used to train a model for a particular task, depending on the business case. The idea that training and test data must have feature spaces that are equal to the underlying distribution is a common one in the machine learning community.

However, in practice, this presumption might not remain true, necessitating the rebuilding of models in the event that characteristics and distribution change. Rebuilding the models and gathering relevant training data is a laborious task.

As it was stated in the previous chapter features were extracted and engineered using VGG16. This method is called transfer learning. It is commonly used in classification projects. It helps augment the robustness of the model and eliminates overfitting issues.

In this specific project the primary approaches and algorithms are going to be mentioned in the upcoming parts. It is important to mention that the trials and the search for the optimal model is never a stable study to make, yet the algorithms used can give us.

an overview about how our classification behaves and what are the missing details needed.

## 4.2. Support vector machine (SVM):

Vapnik et al.'s [25] statistical learning theory served as the foundation for the support vector machine, a novel learning technique that builds on a small number of samples from the data in the available training text to produce the best classification outcomes.

The foundation of SVM classification is the concept of decision hyperplanes, which provide decision boundaries in high-dimensional feature space or input space. SVM structures linear functions from a set of labelled training datasets (hyperplanes in either feature space or input space). The goal of this hyperplane is to separate the positive and negative samples.

The linear separator is typically built with the nearest negative and positive samples at a maximum distance from the hyperplane. This, intuitively, leads to accurate categorization for training data that is similar to yet distinct from the testing data.

Nonlinear difficulties in SVM can be resolved, By translating the n-dimensional input space into a high-dimensional feature space. In this high-dimensional feature space, at last, a linear classifier that functions as a nonlinear classifier in the input space is built.

### 4.2.1. Data linearity

#### 4.2.1.1. Linearly separable SVMs

Think about a binary classification issue where there are N training samples (data) [2]. A tuple ($x_i$, $y_i$) and (i = 1, 2,..., N) indicating each sample is present, where $x_i$=($x_{i1}$, $x_{i2}$,..., $x_{in}$) is in line with the i<sup>th</sup> sample's attribute set. Let $y_i \in\{-1, 1\}$, which is traditionally regarded as a class designation. A linear classifier's (separator's) decision boundary can be expressed as follows:

$$W^T x + b = 0 \tag{4-1}$$

where b is a bias term and w is the weight vector.

Although there are numerous linear separators, the SVM design objective is to specify a decision boundary that is as far away as possible

distant from any given data point. The classifier's margin is determined by this distance between the nearest data point and the decision boundary. With this design method, a (usually small) fraction of the data points that determine the separator's position fully identify the decision boundary for an SVM. We refer to these sites as support vectors. The margin and support vectors for the two class problems are displayed in Figure 4-1.

In the case that the training data exhibit linear separability, a pair (w, b) will exist whereby: $w^T$

$$If\ y_i = 1,\ then\ x_i + b \geq 1; \tag{4-2}$$

$$if\ y_i = -1,\ then\ w^T x_i + b \leq -1. \tag{4-3}$$

The formula for the linear classifier is:

$$f(x) = sign\ (w^T x + b) \tag{4-4}$$

The functional margin of the ith sample xi with regard to a hyperplane (w, b) is defined as in (2-7) for a given dataset and decision hyperplane:

$$y_i\ (w^T x_i + b) = \gamma_i \tag{4-5}$$

Then, the functional margin of any sample in the dataset with minimal functional margin is twice the functional margin of the decision boundary dataset.

The factor of two is derived by calculating the margin's entire width, as shown in Figure 2-18. The shortest path between a point and a hyperplane is known to be parallel to w since it is perpendicular to the plane. w/‖w‖ is a unit vector in this direction. Geometric margin, denoted by $\rho$ in Figure 2-18 , is the maximum width of the band that can be created to divide the support vectors of the two classes. Any xi sample's distance from the separator is equivalent to:

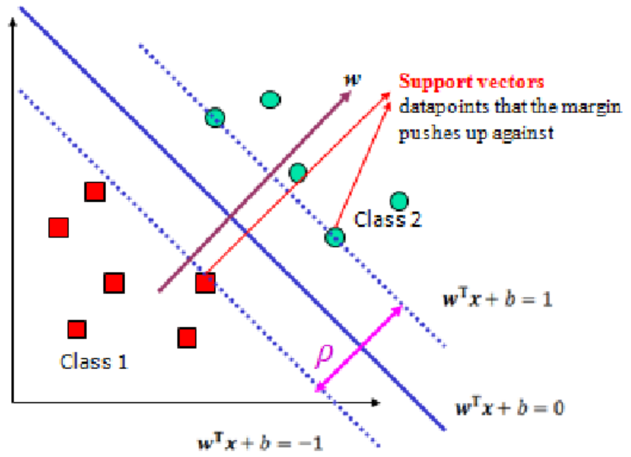


***Figure 4-1***:*decision boundary and margin of*

$$\gamma_i / \|w\| = 1/ \|w\| \qquad (4\text{-}6)$$

The best $w$ and b are determined by maximising this geometric margin (2-8) while designing a linear separator, as stated below:

$\rho = 2 / \|w\|$ maximised.

When $w$ is minimised, $\Phi(w) = ||w|| = w^{T.}$

For any *(xi, yi), i =1,..., N: s.t.* $yi\ (w^T x_i + b) = 1.$ *(4-7)*

To solve the aforementioned difficulties, one must create a dual problem in which each inequality constraint ($y_i\ (w^T x_i + b) >= 1$) in the primary problem is coupled to a Lagrange multiplier, αi:

Determine $a_1, \ldots, a_N$ so that:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \qquad (4\text{-}8)$$

is maximised with respect to α subject to the following constraints:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad s.t \quad \alpha_i \geq 0 \ \ for \ \ i = 1,2,3\ldots,N \qquad (4\text{-}9)$$

The solution of the dual problem αi must satisfy the condition $a_i\{y_i\ (w^T x_i - b) -1)\} = 0$ for

*i =1, 2, …, N.* Then solution to the primal is:

$$w = \sum_{i=1}^{N} \alpha_i yi\mathrm{x}_\mathrm{i} \qquad (4\text{-}10)$$

$$b = y_i - \sum_{i=1}^{N} \alpha_i y_i x_i^T x_i \ \ for\ any \ \ \alpha_i > 0. \qquad (4\text{-}11)$$

In the solution above, most of $\alpha_i$ are zero for data samples which do not support vectors. Each non-zero $\alpha_i$ specifies that the equivalent xi is a support vector. So, the classification function is then as given below:

$$f(x) = \sum \alpha_i y_i x_i^T x + b \qquad (4\text{-}12)$$

We don't need w explicitly in (11) since it relies on an inner product between the test point x and the support vectors xi. Finding class labels for any data points xj involves computing the inner products $x_i^T x_j$ .

### 4.2.1.2. Non linearly separable SVM

A support vector machine's (SVM) ideal hyperplane is defined to maximise generalisation. However, if the training set cannot be separated linearly, the intended classifier might not possess a strong capacity for generalisation, despite the fact that the hyperplanes are ideally selected. Therefore, the original input space is mapped into a high dimensional space called feature space in order to improve linear separability. [1]

The kernel trick, which involves mapping to a higher dimensional space, is made simple and effective by SVMs [22]. The SVMs linear classifier relies on the dot product of the vectors representing the data points.

Assume that the data are transformed using a mapping function Φ to some (potentially infinite dimensional) space H, where they take on the form Φ(xi)TΦ($x_j$). Non-linear operations in input space are similar to linear operations in H. In the event that x = ($x_1$, $x_2$) and \(x) = ($x_1$^2, $x_2$^2, √2$x_1x_2$). Fig.2 shows how this mapping is done. [2]

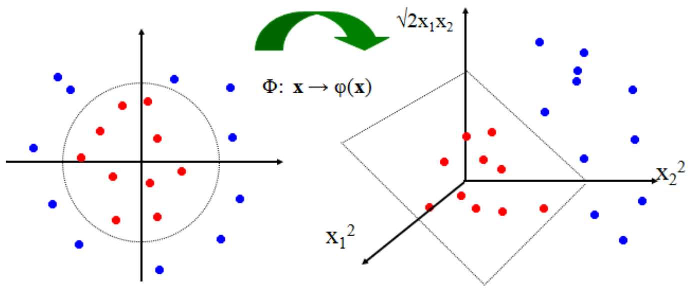


***Figure 4-2:*** *decision boundary of a nonlinear SVM.*

Let K($x_i$, $x_j$)=Φ($x_i$) T Φ($x_j$) as kernel function, so we change all inner products to kernel functions for training data.

Designing SVM is to find $\alpha_1$ ,…, $\alpha_N$ such that:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j K(xi, xj) \qquad (4\text{-}13)$$

is maximised under the following constraints with respect to α:

$$\sum_{i=1}^{N} \alpha_i y_i = 0,\ C \geq \alpha_i \geq 0 \ \textit{for all} \ \alpha_i, \qquad (4\text{-}14)$$

Then the classifier is as below:

$$f(x) = \sum \alpha_i y_i K(xi, xj) + b \qquad (4\text{-}15)$$

### 4.2.2. Kernel regulation in SVMs

A kernel, also known as a kernel function, is a machine learning function that accepts two data points as input and returns a value that indicates how similar the two data points are. In a higher-dimensional space, this similarity is frequently employed to quantify the non-linear interactions between the data points.

What kernels do is take as input two data points (xi and xj). These data points could be characteristics (features) of a thing or instance and it outputs a scalar value (K ($x_i$, $x_j$)) that illustrates how comparable the two data points are.

It exists four types of kernels:

- Linear kernel:

This kernel is the most basic one; it computes the dot product of the two original space data points. It takes the features to have a linear relationship.

It has no additional parameters to adjust, is computationally efficient, and is simple to interpret (similarity based on dot product).

Its function is:

$$K(x_i, x_j) = x_i^T x_j \qquad (4\text{-}16)$$

- <u>Polynomial kernel with degree d:</u>

Before applying the power, this kernel adds a constant c and elevates the dot product of the data points to a power d. In contrast to the linear kernel, it enables the learning of more intricate non-linear correlations.

Compared to the linear kernel, it can capture a larger range of non-linear interactions. On another hand you must select specific and adequate degree d and constant C as hyperparameters, which can have a big effect on performance. You can experience the "curse of dimensionality" in high dimensions as a result of features growing exponentially.

It is defined by the following function:

$$K(x_i, x_j) = (x_i^T x_j + 1)^d \qquad (4\text{-}17)$$

- <u>Radial basis function (RBF) kernel (σ is a positive parameter for controlling the radius)</u>

This kernel regulates the impact of neighbouring points by using the Euclidean distance between the data points and a parameter called sigma. It is useful for modelling non-linear connections with decision boundaries that are smooth.

It works well and is adaptable for a variety of non-linear issues. Smooth decision boundaries may result in improved generalisation.

nonetheless, it might be more difficult to understand than the polynomial or linear kernels.

It can be expressed by the following function:

$$K(x_i, x_j) = exp\,(-\|xi - xj\|^2 / 2\sigma^2) \qquad (4\text{-}18)$$

- <u>Sigmoid kernel:</u>

The tanh function is used for non-linearity in this kernel, which is comparable to the polynomial kernel. It is less popular than the RBF kernel but can be helpful in certain situations. Some non-linear relationships can be modelled by it.

although for some parameter values, it is susceptible to numerical instability.

It is mathematically modelled by:

$$K\,(x_i, x_j) = tanh\,(\delta x_i{}^T x_j + r) \qquad (4\text{-}19)$$

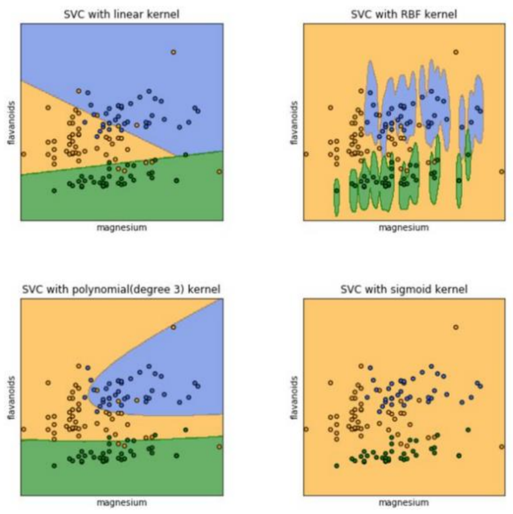


***Figure 4-3:*** *example of different kernels plotting*

### 4.2.3. One vs One SVM

- <u>Description:</u>

The "one against one" technique, sometimes referred to as "round robin," "all pairs," or "pairwise coupling," entails creating one SVM for each pair of lessons.

Consequently, c(c-1)/2 SVMs are trained to separate the samples of one class from the samples of another class for a problem with c classes. When classifying an unknown pattern, the maximum voting method is typically used, with each SVM voting for one class.

- <u>Probability estimation:</u>

The task is to express the global posterior probabilities $\hat{P}(\omega_j | x)$ as functions of the local posterior probabilities $\hat{P}(\omega_j | f_{j,j'}((x))$, where $f_{j,j'}(x)$ denotes the output of the SVM trained to distinguish class $\omega_j$ from class $\omega_{j'}$. Each SVM's output is mapped into probability using a sigmoid function. Numerous approaches have been put out in the literature.

- <u>Advantage of OvO SVM:</u>

Possibility of improved handling of data imbalance: OvO may be more resilient to class imbalance in situations where certain classes have noticeably less data points than others than One vs all. This is so that minority classes may benefit from improved decision boundaries as a result of each SVM's concentration on a particular class pair.

### 4.2.4. Grid search

In machine learning, grid search is a popular method for hyperparameter tweaking. It seeks to determine which model's hyperparameter combinations yield the greatest results for a certain task.

The logic behind the grid search functionality:

- You specify a range of possible values for each hyperparameter to be explored. This range can be defined as a list of discrete values or a continuous range with a certain step size;
- The grid search will train your specified model on all the possible combinations between the hyperparameters, so it gives you the most optimal ones. It works in a probabilistic logic.

**Example:**

➔ Consider two hyperparameters  1 and 2 such as:

Hyperparameter 1 = [a, b, c]  / hyperparameter 2 = [x, y, z]

➔ We perform a grid search on this given data, we can visualise this process by the following figure:

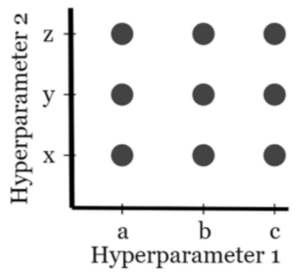


*Figure 4-4Grid search initialization demonstration.:*

- ➔ This will be having 9 combinations of both of our hyperparameters.
- ➔ By next our predefined model (eg: SVM, Random Forest, Stochastic gradient descent. Etc) will be trained on all the previously provided combinations.
- ➔ By the end only one combination will be outputted and it is considered as the optimal solution for our model.

### 4.2.5. Approaches taken and their respective results

In the following part four approaches are going to be introduced:

- ➔ Transfer learning for feature engineering with a linear kernel for the SVM classifier
- ➔ Transfer learning for feature engineering with an RBF kernel for the SVM classifier
- ➔ Fine-tuned features for feature engineering with a linear kernel for the SVM classifier
- ➔ Fine-tuned features for feature engineering with an RBF kernel for the SVM classifier

The evaluation matrices used are:

➔ **Precision**: calculates the percentage of real positives compared to expected positives and more simply, it indicates the proportion of things that your model correctly identified as positive;

$$Precision = \frac{TP}{TP + FP} \tag{4-20}$$

➔ **Recall**: calculates the percentage of real positives that were successfully detected, it indicates the percentage of all real positive cases that your model successfully recognized;

$$Recall = \frac{TP}{TP + FN} \tag{4-21}$$

➔ **F1-score:** balances the value of precision and memory by combining them into a single metric. An excellent balance between detecting genuine positives and averting false positives is indicated by a high F1-Score;

$$F1 - score = \frac{2.Precision\ .Recall}{Precision + Recall} \tag{4-22}$$

➔ **Accuracy:** calculates the percentage of samples that are correctly identified (including positives and negatives). This straightforward metric may be deceptive in datasets that are unbalanced;

➔ **Macro average:** determines the average F1-score, recall, and precision for each class. helpful in assessing performance on unbalanced datasets where accuracy may not always be dependable;

➔ **Weighted average:** determines the precision, recall, and F1-score weighted average; the weights are determined by the class support (number of samples). assigns classes with more samples a higher weight;

➔ **Support:** shows how many data points there are overall for each class. It facilitates comprehension of how your data is distributed throughout the various classes;

➔ **Confusion matrix:** a table that shows how well a categorization model performs with a given set of data. Actual class labels are shown in rows, and anticipated class labels are shown in columns. The number of instances where the model predicted the label in the relevant column and the true label fell in a certain row is displayed in each cell.

A. Transfer learning X linear kernel:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | validation accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 0.98 | 0.97 | 0.97 | 208 | 98,36% | 62,6% | 0.98 | 0.98 |
| crack | 0.96 | 0.98 | 0.97 | 167 | | | | |
| dent | 0.98 | 0.98 | 0.98 | 173 | | | | |
| non defected | 0.99 | 0.99 | 0.99 | 486 | | | | |

***Table 4-1**:SVM transfer learning X linear kernel results:*

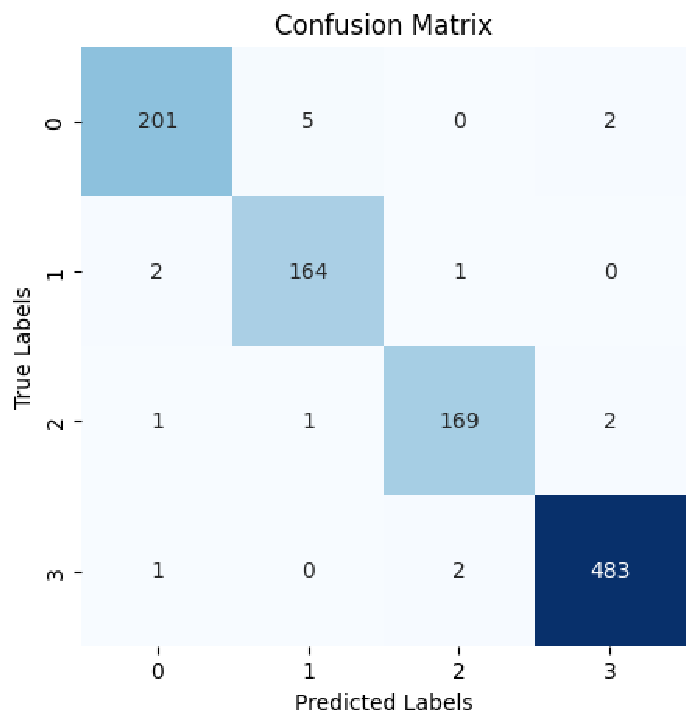


***Figure 4-5:** SVM transfer learning X linear kernel confusion matrix*

B. Transfer learning X RBF kernel:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | validation accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 0 | 0 | 0 | 208 | 47% | 28% | 0.17 | 0.33 |
| crack | 0 | 0 | 0 | 167 | | | | |
| dent | 0 | 0 | 0 | 173 | | | | |
| non defected | 0.47 | 1 | 0.64 | 486 | | | | |

***Table 4-2**: SVM transfer learning X RBF kernel results*

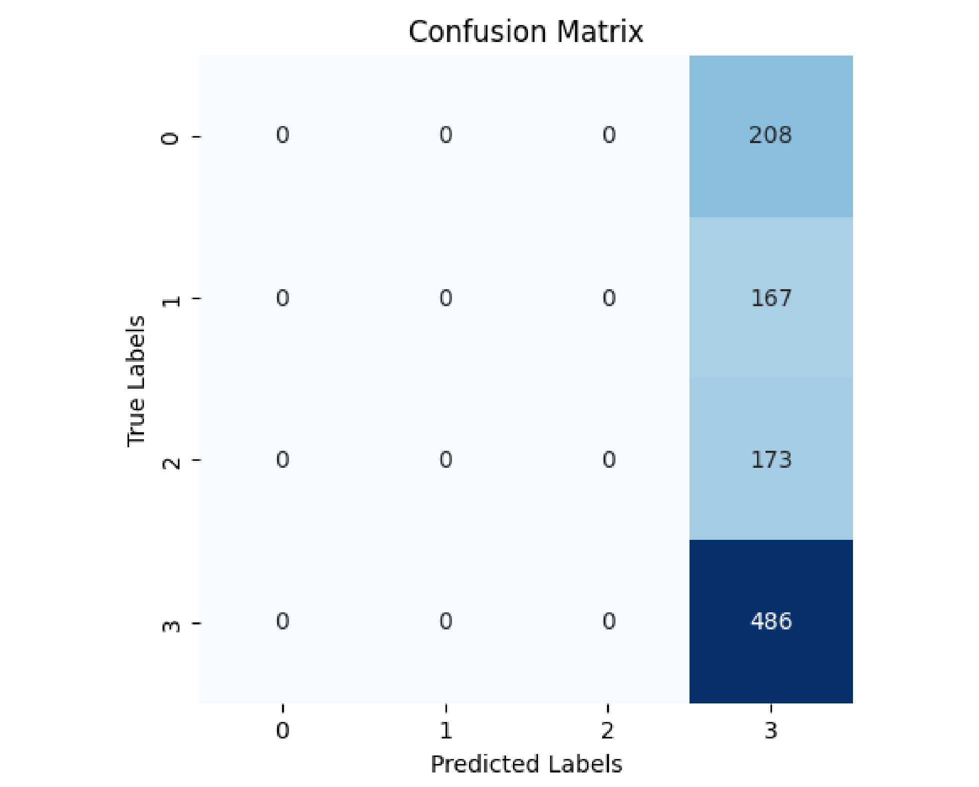


***Figure 4-6***:*SVM transfer learning X RBF kernel confusion matrix*

C. Fine tuning X linear kernel

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | validation accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 0.99 | 0.96 | 0.98 | 216 | 98% | 47,2% | 0.98 | 0.98 |
| crack | 0.97 | 0.97 | 0.97 | 213 | | | | |
| dent | 0.97 | 0.98 | 0.97 | 212 | | | | |
| non defected | 0.99 | 1 | 0.99 | 606 | | | | |

***Table 4-3***: *SVM fine-tuning X linear kernel results*

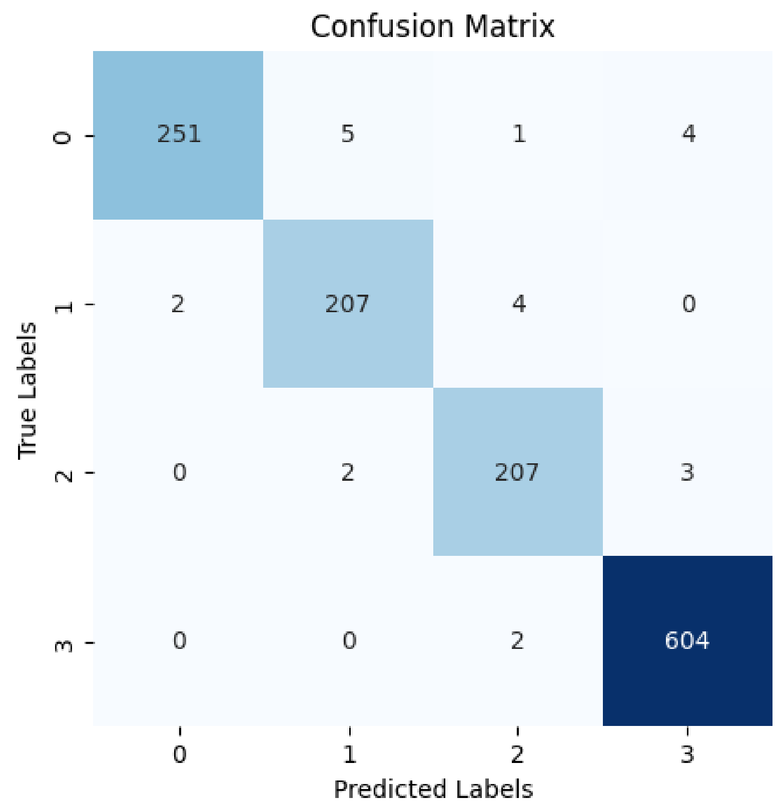


*Figure 4-7: SVM fine-tuning X linear kernel confusion matrix*

D. Fine tuning X RBF kernel:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | validation accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 0 | 0 | 0 | 216 | 47% | 20% | 0.16 | 0.33 |
| crack | 0 | 0 | 0 | 213 | | | | |
| dent | 0 | 0 | 0 | 212 | | | | |
| non defected | 0.47 | 1 | 0.64 | 606 | | | | |

***Table 4-4:****SVM fine-tuning X RBF kernel results*

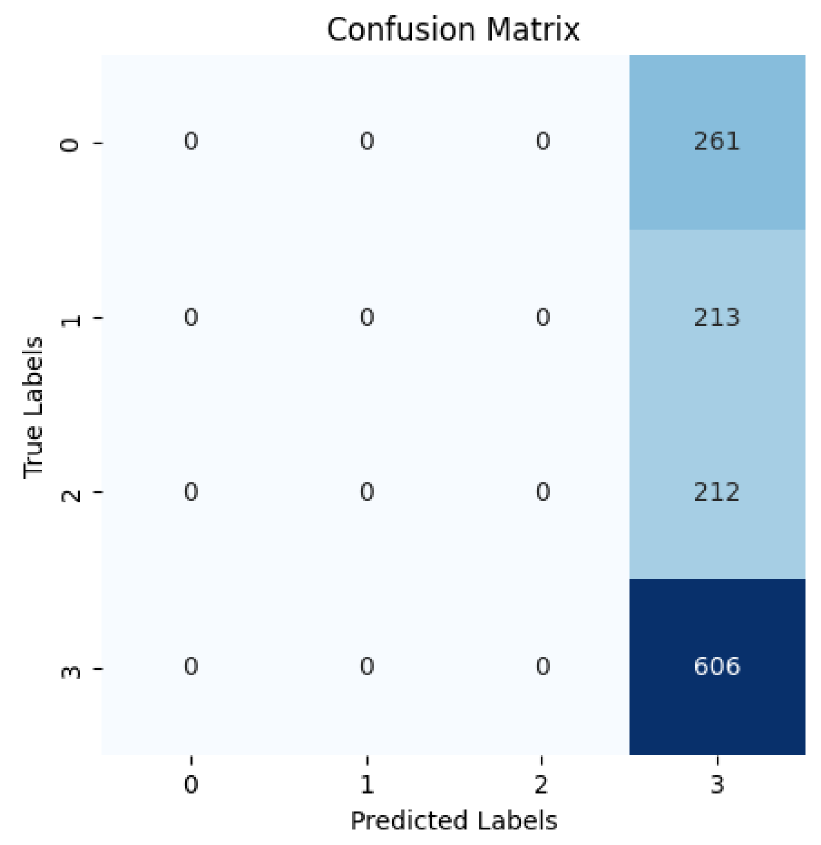


***Figure 4-8:*** *SVM fine-tuning X RBF kernel confusion matrix*

➔ General comparison between all the approaches:

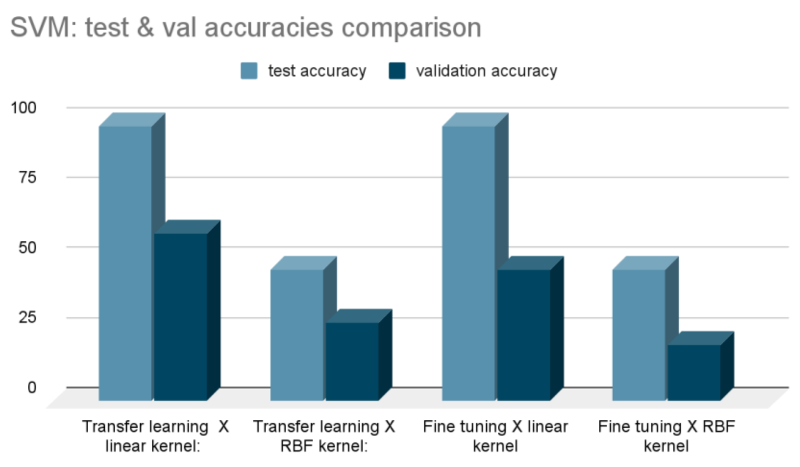


***Figure 4-9:*** *general comparison between all the approaches*

### 4.2.6. observations

The exploration of different types of training the SVM kernel led me to make general observations on different aspects that were set as variables and how changing them would change the performance of my model. The key observations and details are as follow:

A. <u>How kernel tuning affected the performance:</u>

- We have noticed a drastic change in both of the results where the linear kernel surprisingly performed better than the RBF one;
- From the classification report as the tables show we can notice that the scores of the linear kernel for all classes are high whereas with the RBF kernel are zero except for non defected classes;

- When it comes to the confusion matrix we can notice that it gave us a clear insight on how our model classified each class as its true class on the linear kernel.
- For the RBF kernel we can notice that our model classified all classes as non defected and it didn't actually perform on the other classes.

B. how fine tuning affected the performance:

- The inclusion of the fine tuning on the features didn't significantly improve the performance of our model when it comes to training;
- We can notice that the accuracy of validation got lower when we fine tuned the features while the one of the training stayed as high as it was.

C. changing various hyperparameters during the training:

- The previously showcased values are the ones depending only on changing two training factors. In Fact performing a grid search allowed to freeze the other factors;
- Before applying the grid search we tried to play on other hyperparameters and we noticed that C (cost) parameter with a linear kernel had no effect while increasing it helped the RBF accuracy to augment but led it to overfit;
- Exploring with a different range of gamma parameters with the RBF kernel didn't indicate a great or an important improvement.

D. training and validation accuracies:

- In an adequate performing model the training and the validation accuracies should be almost the same, in our case we can clearly notice the huge drop in accuracy from training to validation.

### 4.2.7. Explanation and discussion

**A- Linearly separable data:** taking the change in performance between linear and RBF kernels can potentially showcase that our data is linearly separable. What led to sustain this opinion is the irrelevance of the cost values changing with the linear kernel because there's no need to adjust the margin for few misclassifications.

**B- overfitting:** In this case so many results show that our model is overfitting in all the previously proposed approaches and here's a breakthrough of it:

- When it comes to the best-case scenario "transfer learning X linear kernel" we noticed that the accuracy between training and validation isn't the same or even close to be. It means that this model is learning the images as whole or both of relevant and irrelevant features which lead to its incapacity of generalisation;
- We noticed also that validation accuracy got lower when we fine-tuned our features, this is because fine tuning increases the complexity of our features, amplifies noise and adds more irrelevant data to our training inputs, which makes our model more sensitive. This all will lead our model to perform poorly in the generalisation scenario;
- The reason behind our trained model on an RBF kernel performed so poorly and couldn't classify each class adequately can be also a potential problem of overfitting. As it was stated before our data is highly to be considered linearly separable, though in another hand if it wasn't the case, we can assume that the model learnt only the "non defected class" and was capturing pipes as whole in the other classes rather than the specific defected area.

### 4.2.8. conclusion

We conclude from these results that the SVM algorithm even though it performed well on some aspects but it still had its own constraints that left it non optimal. This opens the door to try on in this multi classification problem other algorithms that might perform better.

## 4.3. Random forest

### 4.3.1. Description of a random forest

The Random Forest Algorithm's broad appeal can be attributed to its versatility and ease of use, which allow it to effectively address problems related to both regression and classification. The method is a useful tool for a variety of machine learning predictive tasks because of its strength in handling complicated datasets and mitigating overfitting.

The ability of the Random Forest Algorithm to handle data sets with both continuous variables as in regression and categorical variables as in classification is one of its most crucial properties. In tasks involving regression and classification, it performs better. We will learn how random forests operate in this tutorial and apply them to a classification job.

### 4.3.2. Ensemble learning techniques

#### 4.3.2.1. Booting (bootstrap aggregation)

Using replacement data from the original dataset, bootstrapping entails generating multiple training datasets. This implies that a data point may be chosen more than once in a single training set, and that certain data points may be completely eliminated.

These "bootstrapped" datasets have been created, and as a result, each dataset differs somewhat from the original data. This introduces variation into the process of training.

Next, each bootstrapped dataset is used to train a learner (such as a decision tree). As a result, several models are produced, each with a slightly different learning curve.

Each model predicts something about a new data point during prediction, and the final forecast is usually the average (classification) or sum (regression) of the predictions made by each model separately.

It is used mostly in random forest algorithms.

#### 4.3.2.2. Boosting:

Using boosting, an ensemble is progressively built, with each new model learning from the mistakes of the older ones.

Using the original data, the first model is trained. Next, using a different set of data, a second model is trained with an emphasis on data points that the first model incorrectly identified. These incorrectly identified points have higher weights, which causes the second model to focus more on them.

This approach is iterative, with each new model concentrating on the "hard" examples that proved difficult for the earlier models to solve.

The ultimate forecast is frequently a weighted total of the predictions made by each separate model, much like bagging.

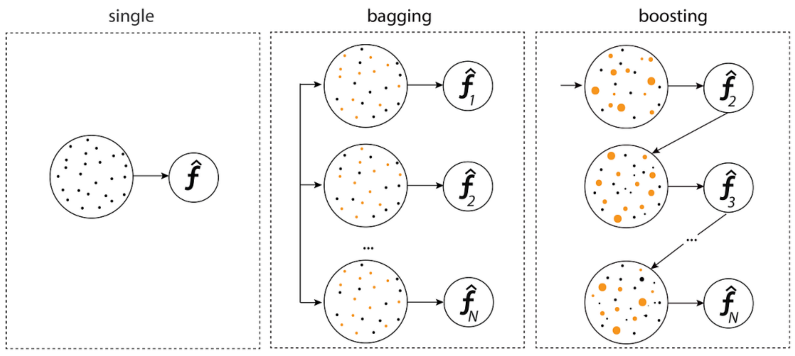


***Figure 4-10**:descriptive scheme of bootstrapping and boosting*

### 4.3.3. Random forest algorithm building

- Constructing the Forest
  - Random Sampling (with Replacement): The algorithm selects a random sample (with replacement) from the initial training data for every tree in the forest. This implies that a data point may be chosen more than once inside a single tree, resulting in forest diversity. Usually, the size of the original dataset (or a subset of it) is the same as the number of data points drawn;
  - Feature Subset Selection: A random subset of features is selected from the complete feature set at each node of the tree. Typically, this subset is equal to the square root of the total features. By doing this, the tree-building process is made more random and overfitting on any particular feature is avoided;
  - Best Split Selection: To divide the data into two child nodes, the algorithm determines which split among the selected features is the best. The split that optimises a certain criterion—variance reduction for regression, for example, or Gini impurity for classification—is selected;
  - Recursive splitting: Until a stopping requirement is satisfied, the process of choosing the optimal split and producing child nodes continues. This could be a minimal amount of data points in a node, reaching a specific depth in the tree, or attaining a sufficiently pure node (all data points belong to the same class).
- Educating Several Trees:

To build many decision trees (usually hundreds or thousands), you repeat steps 2(a) through 2(d) one after the other. The result is a diversified ensemble of trees, with each tree having a unique random feature subset and random data sample.

- Forming Forecasts:

Classification: Each tree in the forest classifies a new data point independently in order to make a forecast. Out of all the individual forecasts made by each tree, the final prediction has the highest frequency (majority vote).

### 4.3.4. Hyperparameters to Increase the Predictive Power

- **n_estimators**: The number of trees the algorithm constructs prior to averaging the predictions;
- **max_features**: The most features that a random forest will take into account before splitting a node;
- **mini_sample_leaf:** Ascertains the bare minimum of leaves needed for an internal node to split;
- **Criteria**: In each tree, it is how should the node be divided (Log Loss/Entropy/Gini Impurity);
- **max_leaf_nodes:** Each tree's maximum number of leaf nodes.

### 4.3.5. Approaches taken and their respective results

The upcoming table 2-5 shows different values and choices of the random forest hyperparameters.

| | Approaches taken | | | |
|---|---|---|---|---|
| | A | B | C | D |
| n_estimators | 2000 | 1500 | 2000 | 1500 |
| min_samples_split | 10 | 8 | 8 | 10 |
| min_samples_leaf | 2 | 4 | 4 | 2 |
| random_state | 42 | 42 | 42 | 42 |
| oob_score | TRUE | TRUE | TRUE | TRUE |

***Table 4-5**: approaches taken for Random Forest classifier*

N.B: The evaluation matrices used are the same as the SVM ones except A different validation set is not used by Random Forest. Rather, it uses "out-of-bag" data points, that is, data points that were not chosen for a specific tree's training to calculate the generalisation error of the model. Overfitting can be avoided by keeping an eye on this error throughout training.

A. Set A of hyperparameters:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | oob_score | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 1 | 0.6 | 0.75 | 304 | 81% | 0.2 | 0.8 | 0.82 |
| crack | 0.91 | 0.71 | 0.8 | 258 | | | | |
| dent | 0.93 | 0.64 | 0.76 | 267 | | | | |
| non defected | 0.73 | 1 | 0.84 | 722 | | | | |

***Table 4-6**: random forest with set A of hyperparameters results*

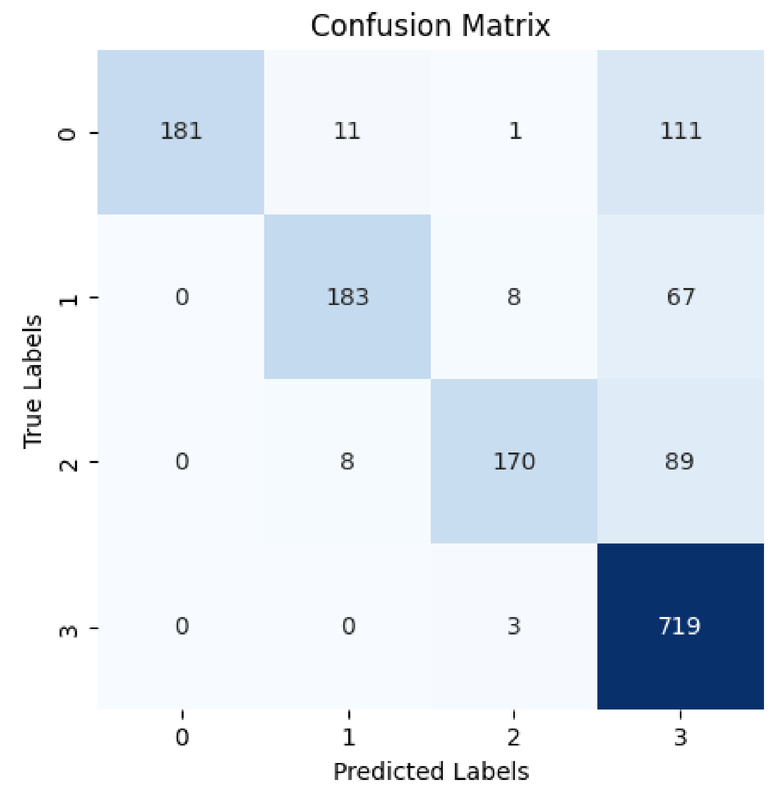


***Figure 4-11**:random forest with set A of hyperparameters confusion matrix*

B. Set B of hyperparameters:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | oob_score | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 1 | 0.58 | 0.73 | 304 | 79,17% | 0.2 | 0.78 | 0.8 |
| crack | 0.91 | 0.66 | 0.77 | 258 | | | | |
| dent | 0.91 | 0.61 | 0.73 | 267 | | | | |
| non defected | 0.71 | 0.99 | 0.83 | 722 | | | | |

***Table 4-7**:random forest with set B of hyperparameters results:*

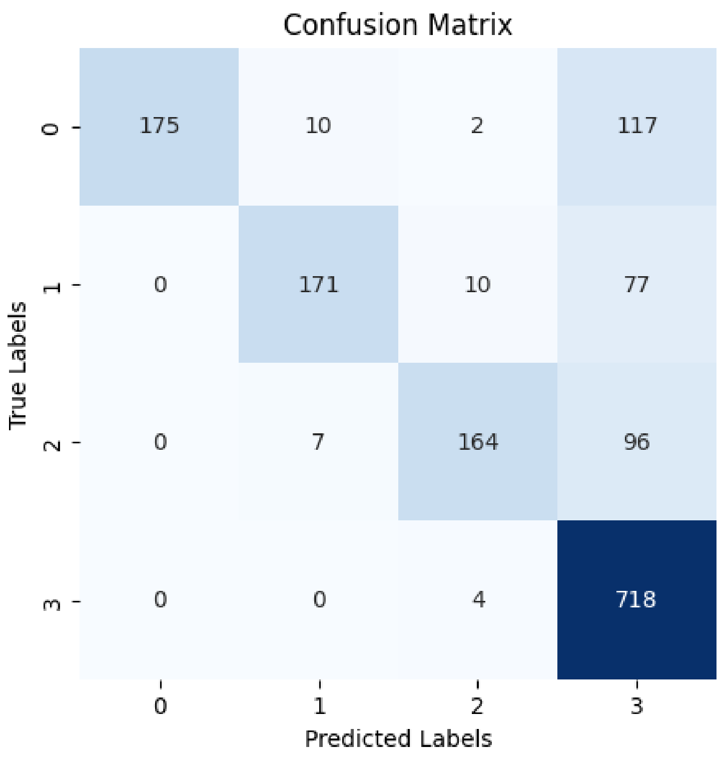


***Figure 4-12***:*random forest with set B of hyperparameters confusion matrix*

C. Set C of hyperparameters

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | oob_score | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 1 | 0.56 | 0.72 | 304 | 79.11% | 0.2 | 0.78 | 0.8 |
| crack | 0.91 | 0.66 | 0.76 | 258 | | | | |
| dent | 0.92 | 0.63 | 0.75 | 267 | | | | |
| non defected | 0.71 | 0.99 | 83 | 722 | | | | |

***Table 4-8:****random forest with set C of hyperparameters results*

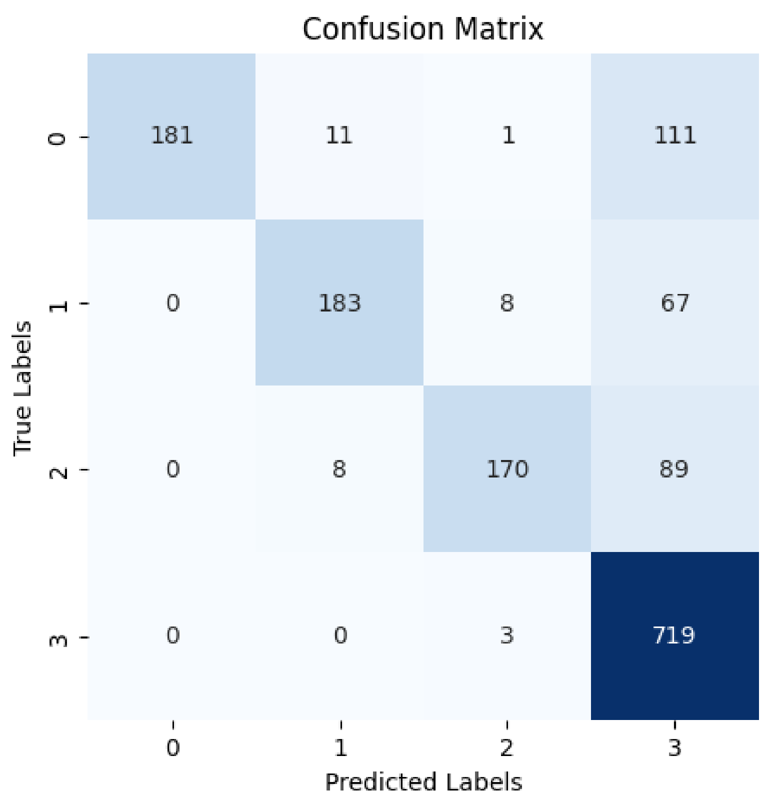


***Figure 4-13****:random forest with set C of hyperparameters confusion matrix*

D. Set D of hyperparameters:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | oob_score | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| corrosion | 1 | 0.59 | 0.74 | 304 | 80.53% | 0.19 | 0.8 | 0.82 |
| crack | 0.9 | 0.72 | 0.79 | 258 | | | | |
| dent | 0.94 | 0.63 | 0.76 | 267 | | | | |
| non defected | 0.73 | 1 | 0.84 | 722 | | | | |

***Table 4-9:****random forest with set D of hyperparameters results*

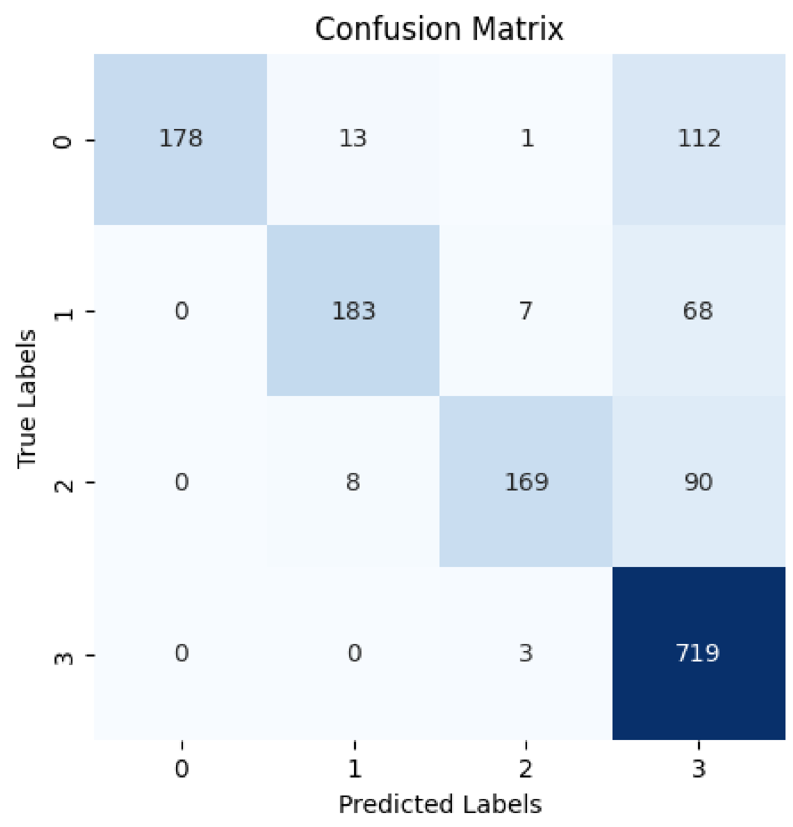


***Figure 4-14:*** *random forest with set D of hyperparameters confusion matrix*

### 4.3.6. observations

**A. changing the n_estimators:**

This hyperparameter specifies the number of the decision trees in our random forest. We notice that with the sets A /D or B/C even the number of estimators are changed yet it didn't much affect the performance of our model.

**B. changing min sample split:** The min sample splits controls the number of how few samples are needed to split an internal node in a Random Forest decision tree.

We notice that augmenting the number of samples betters the performance of our model.

**C. changing min sample leaf:**

The min sample leaf determines the number of samples required to split an internal node in a Random Forest decision tree as shown in the figure 4-15. We can notice that increasing it while reducing the min sample splits decreased our model's performance.

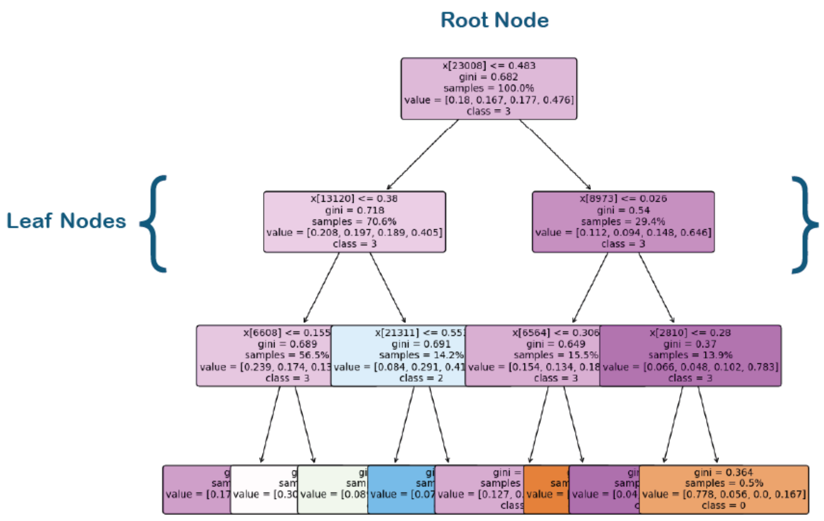


***Figure 4-15:*** *Visualization of a part of a tree in the random forest of set A*

**Remark:** The choice of varying the previously stated hyperparameters came from applying a grid search on the RFclassifier, I took close values to the optimal ones to showcase their influence on the overall performance.

### 4.3.7. Explanation and discussion:

A. **Tree Structure and Overfitting:** The Random Forest's individual trees' structures are directly impacted by the values of min_samples_split and min_samples_leaf. Too low of these values can result in:

- **Trees that are Shallow:** If too few samples are permitted for splitting, the trees may grow excessively shallow and fail to depict the nuanced relationships present in the data. This may reduce the overall accuracy of the model;
- **Overfitting:** Trees that are able to split on extremely small subsets of data may acquire patterns unique to the training set that are poorly generalised to new data.

B. **Ensemble averaging:**

Random Forests average the predictions from multiple decision trees. This inherent averaging helps mitigate overfitting to some extent. Even if some individual trees overfit due to low min_samples_split or min_samples_leaf values, the averaging process can help reduce the overall impact.

C. **Effect on individual trees vs. whole Forest:**

Although n_estimators boost the total number of trees, it has no direct effect on how each tree is arranged. Without the proper values for min_samples_split and min_samples_leaf, each tree remains vulnerable to overfitting.

### 4.3.8. Conclusion:

min_samples_split and min_samples_leaf have a more direct impact on the structure and potential overfitting of individual trees within the Random Forest.

n_estimators influence the overall complexity of the forest, but its impact on overfitting is often less pronounced compared to the other two parameters.

It's crucial to tune all three parameters (min_samples_split, min_samples_leaf, and n_estimators) through techniques like grid search or randomised search to find the optimal combination for your specific dataset and task. This ensures you achieve a good balance between model complexity, reducing overfitting, and achieving optimal performance.

## 4.4. Comparison between the two models:

### 4.4.1. Strength of the SVM classifier:

- High Accuracy: Based on my results, SVM performed well on my particular task. When the data can be efficiently transferred to a higher dimension for separation or is linearly separable, support vector machines (SVMs) are excellent at identifying the best hyperplanes for classification, resulting in high accuracy.
- SVMs prioritise maximising the margin between classes, which can result in strong generalisation on data that hasn't been seen before. This is so that there is less chance of overfitting to the training set since the model learns the most discriminative features.
- Interpretability: Compared to Random Forests, SVMs, especially linear SVMs are easier to understand. By identifying the decision boundary and the most significant data points, I could get understanding of the reasoning behind the model.
- Memory Efficiency: Random Forests need storing all trained trees in memory, but SVMs can be more memory economical for smaller datasets.

### 4.4.2. How can random forest perform well in some scenarios:

- **Non-linear Data:** SVMs may handle non-linear data by using kernel functions, however selecting the appropriate kernel and setting its hyperparameters might be difficult. By integrating several decision trees, Random Forests automatically manage non-linearity and may even outperform other methods when applied to datasets that are naturally non-linear.
- **High-Dimensional Data:** SVMs can become computationally costly and prone to overfitting when dealing with datasets that have a large number of features. When feature significance ratings are used to identify irrelevant features, Random Forests can be more resilient to high dimensionality.
- **Noisy Data:** When compared to SVMs, Random Forests are typically more resilient to noisy data. They can lessen the impact of individual noisy data points by averaging predictions from several trees.
- **Imbalanced Classes**: In situations when one class has substantially less samples than the other, Random Forests are frequently a better fit for managing imbalanced datasets. Compared to SVMs, they are less prone to problems with class imbalance.

### 4.4.3. Side by side comparison:

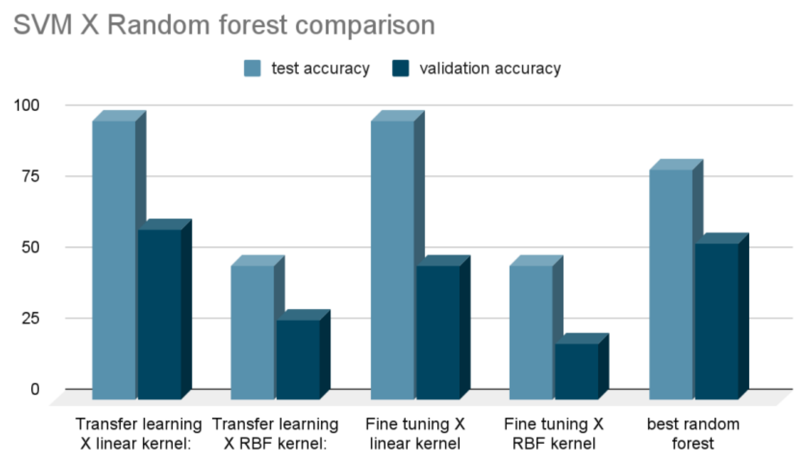


***Figure 4-16:*** *SVM & RF approaches side by side comparison*

As figure 4-16 shows the best cases of the SVM and the random forest gave almost similar results, yet they both showcased overfitting issues. This is due to the lack of data, random forest usually deals with big amounts of data and in my case my dataset wasn't big enough.

Since SVMs performed well on my task, it appears that the data may be a good fit for SVMs because of either the kernel function's efficacy or its intrinsic linear separability as it was previously concluded. Random Forests, however, might be more advantageous in handling non-linearity, large dimensionality, noisy data, or imbalanced classes, depending on the features and priorities of my particular dataset.

## 4.5. Conclusion:

This chapter discussed the two most powerful machine learning classification algorithms: Support Vector Machines (SVMs) and Random Forests. We investigated their inner workings, seeing how SVMs methodically design hyperplanes to distinguish classes and how Random Forests leverage crowd wisdom via ensembles of decision trees.

The empirical data revealed the strengths of both methods. SVMs, with their emphasis on maximizing margins, obtained high accuracy, especially when dealing with linearly separable data. Random Forests, on the other hand, demonstrated their adaptability by successfully managing nonlinear difficulties and noisy data.

Our comparison investigation revealed further pros and limitations. While SVMs provide interpretability and memory efficiency, they might be difficult to use due to nonlinearity and high dimensions. Random Forests excel in these areas but may take more memory and be less interpretable.

While both algorithms have inherent resistance, the search for optimal performance necessitates more investigation. In the following chapter, we'll go on an optimization. journey, studying approaches avoiding the hazards of overfitting, guaranteeing our chosen model generalises flawlessly to previously unexplored data journey, studying approaches avoiding the hazards of overfitting, guaranteeing our chosen model generalises flawlessly to previously unexplored data

# Chapter 5: optimisation and deployment

# Chapter 5: optimisation and deployment

## 5.1. Introduction

In the previous chapter various approaches to classify and detect the defects in pipelines were mentioned, we have noticed that they didn't give the desired applicable results.

In this chapter a new method is going to be introduced with a whole new algorithm.

## 5.2. Binary tree of SVMs with YOLOv8

The logic behind this approach is disassociating the classes and making the classification by steps or as a tree.

As figure 5-1 shows our model will firstly detect if our input is of a defected pipeline or of a non defected one, by that if the pipe is defected it will see if it is corroded or not and at the end it will do a segmentation of the cracks and the dents in both the corroded or the non corroded pipelines as the following figure shows:

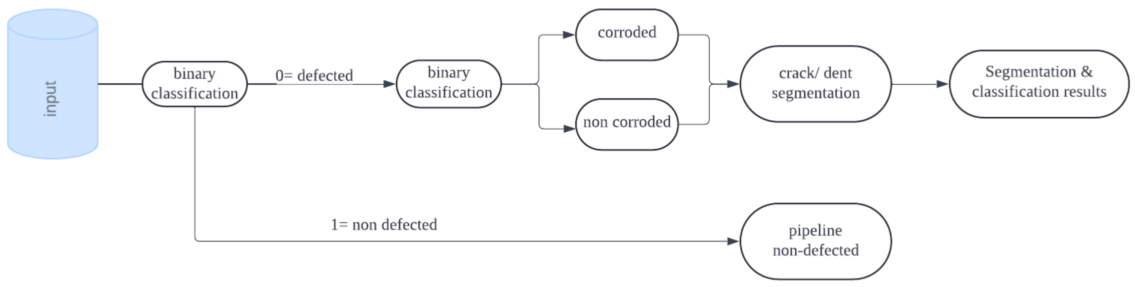


***Figure 0-1**: classification and segmentation structure*

### 5.2.1. Binary tree of SVMs

To do the first classifications, the models that are trained are both binary classification with SVM. We will firstly build an SVM model with a linear kernel to classify whether our pipelines are defected or non defected. Another separately built model will also classify a certain data of pipelines if they are corroded or non corroded.

N.B: check chapter4 for all the details that concern Support Vector Machines.

### 5.2.2. Crack & dent segmentation using YOLOv8

#### 5.2.2.1. Data annotation

For this specific task, a data annotation was necessary to identify the segments or the parts of cracks “figure 5-3” and dents “figure 5-2” in the corroded and the non corroded pipelines.

The software used for this process is Roboflow. You can directly highlight the part of your defect and store it in the database.

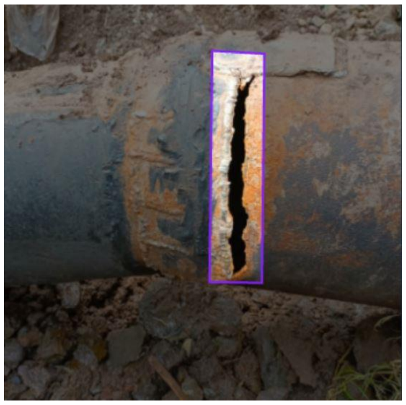


***Figure 0-3**: segmentation of a crack*

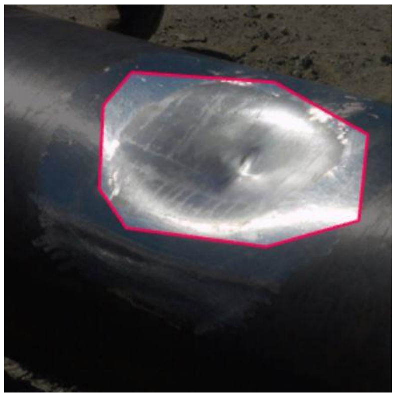


***Figure 0-2**: segmentation of a dent*

#### 5.2.2.2. Data augmentation

Roboflow also offers the possibility to do different types of augmentations. After saving your dataset you can set the augmentation factor as you want. The operations made on the dataset to vary it were:

- Flip: Horizontal;
- 90° Rotate: Clockwise, Counter-Clockwise;
- Rotation: Between -15° and +15°;
- Grayscale: Apply to 20% of images;
- Brightness: Between -15% and +15%;
- Exposure: Between -10% and +10%.

#### 5.2.2.3. YOLOv8 concept

YOLOv8, which is well recognized for its object detection skills, may also be used for instance segmentation jobs.

In normal object detection, YOLOv8 generates bounding boxes around objects in an image and assigns them class labels. This method identifies the presence of objects and their locations but does not provide precise information about their shapes.

Instance segmentation advances object detection. It seeks to not only recognize and locate things, but also to precisely define their borders or shapes at the pixel level. By forecasting a segmentation mask for every object in the picture, YOLOv8 is able to do instance segmentation. This mask is a grayscale image with values set to each pixel that represent different item classes. Background pixels are usually assigned a value of 0, whereas individual object pixels have distinct values (usually encoded using one-hot encoding). You may determine an object's presence and class as well as its exact shape and interactions with other objects in the image by examining the segmentation mask.

- Evaluation matrices of the YOLOv8:
  - **Box_loss**: represents the loss incurred when attempting to anticipate an object's bounding box in a picture. A smaller box_loss value suggests that the bounding boxes of the objects in the image better match the ground truth (actual) bounding boxes predicted by the model. IOU (Intersection over Union) functions are used by YOLOv8 to quantify the overlap between predicted and ground truth bounding boxes. In order to update the model's weights during training, the loss function computes the amount that the predicted boxes depart from the ideal boxes. This value is then back propagated;

  - **Seg_loss:** reflects the loss incurred when attempting to forecast segmentation masks for visual objects. Grayscale pictures known as segmentation masks include each pixel's value indicating whether it belongs to the background or a certain class of objects. When the seg_loss is smaller, it means that the predicted masks of the model more closely match the actual shapes and bounds of the objects in the image. YOLOv8 may use specific loss functions appropriate for segmentation applications, taking into account aspects such as object boundary preservation and pixel-wise accuracy;

- **Cls _loss:** represents the loss incurred in the process of classifying items found in the image. YOLOv8 assigns a class label to every object it finds. The degree to which these predicted labels agree with the objects' actual class labels is measured by the class_loss. A smaller cls_loss shows that the objects the model identifies are being correctly classified by the model. This is frequently the loss function used in classification tasks such as cross-entropy loss;

- **Instances:** The model is trained using data that includes annotations for the items that are seen in the pictures. Each object's location, size, and class are described in these annotations. The total number of objects that are truly present in the current image or batch of photos is known as the number of ground truth instances.

## 5.3. Results:

A. Binary SVM of the defected and non defected class:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | validation accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| **defected** | 0.89 | 0.93 | 0.92 | 30 | 85% | 80% | 0.87 | 0.89 |
| **non defected** | 0.9 | 0.8 | 0.83 | 15 | | | | |

***Table 0-1**: results of binary classification "defected/ non defected"*

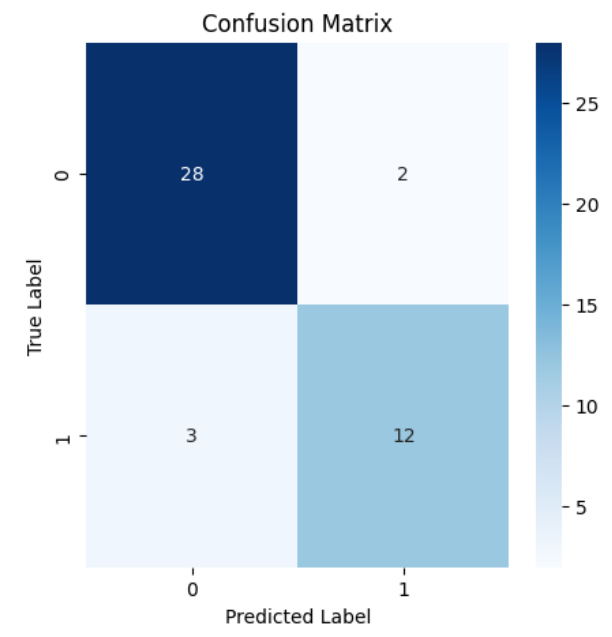


***Figure 0-4:****Confusion matrix of binary classification "defected/ non defected"*

B. Binary SVM of corroded pipeline and no corroded:

| classes | Precision | Recall | F1-Score | Support | Training Accuracy | validation accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|---|---|---|
| **corroded** | 0.81 | 0.87 | 0.84 | 15 | 84% | 83% | 0.83 | 0.83 |
| **non corroded** | 0.86 | 0.8 | 0.83 | 15 | | | | |

***Table 0-2****:results of binary classification "corroded/ non corroded"*

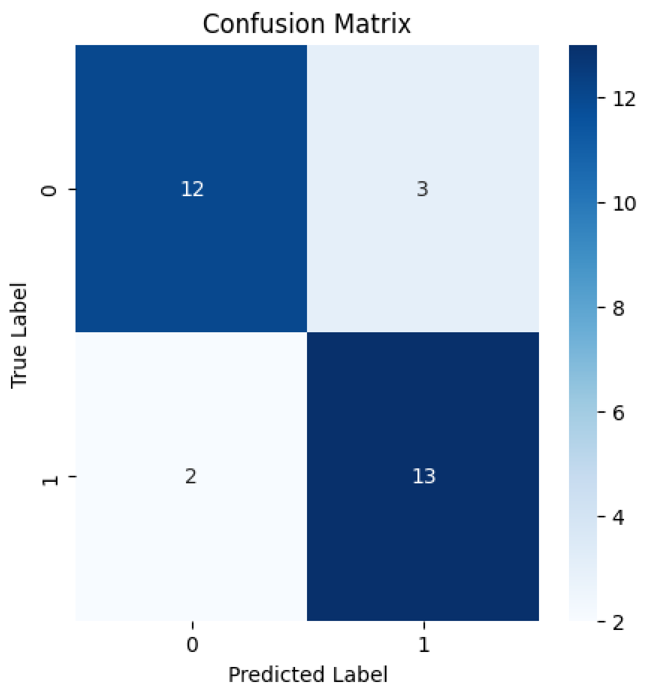


***Figure 0-5***:*Confusion matrix of binary classification "corroded/ non corroded"*

C. YOLOv8 results on some epochs:

| epochs | box_loss | seg_loss | cls_loss | Instances |
|---|---|---|---|---|
| **1514/2000** | 0.43 | 0.81 | 0.31 | 11 |
| **1518/2000** | 0.44 | 0.8 | 0.32 | 10 |
| **1523/2000** | 0.44 | 0.74 | 0.31 | 11 |

***Table 0-3:*** *results of segmentation "crack & dent"*

Some detected images:

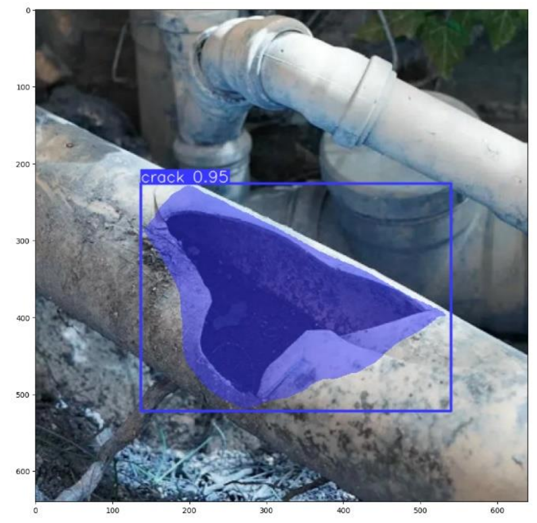


***Figure 0-7**: YOLOv8 results on a cracked unseen image*

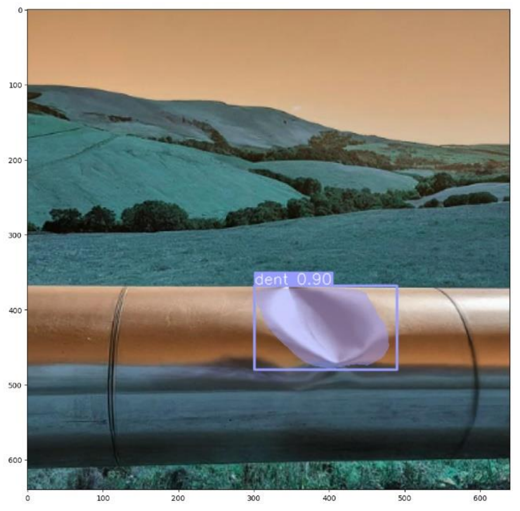


***Figure 0-6:** YOLOv8 results on a dented unseen image*

## 5.4. Discussion:

**A. Defected and non defected classes:**

- The previous results suggest that the model is doing well on the classification problem;
- The model's high precision (0.86-0.90) for both classes suggests that it can predict courses with a fair degree of accuracy;
- Additionally, recall is strong (0.80-0.93), indicating that the model detects the majority of real events in the data;
- A balanced picture is presented by F1-scores of approximately 0.83, which validate a good trade-off between recall and precision;
- With an overall accuracy of 0.89, the model is properly categorising the majority of the data points;

- Taking the gap between both, validation and training accuracy is acceptable, which means that our model is able to generalise on unseen data.

**B. Corroded and non corroded pipelines:**

- Balanced Performance: For each class, recall and precision are both in the high 0.8 range, indicating that the model finds the majority of real objects while avoiding a large number of false positives;
- F1-Scores: For both classes, scores of 0.83 or higher show a solid balance between recall and precision;
- Accuracy: With an overall accuracy of 0.83, the model appears to be accurately categorising the majority of data items;
- Validation and training accuracy are almost the same which indicates that the model performed well and learnt the features.

**C. YOLOv8:**

- In all, the model is running through 2000 epochs; the previous results demonstrate development from epochs 1514 to 1523;
- box_loss (0.43 - 0.44): This shows the loss incurred when attempting to forecast bounding boxes for picture objects. Although there's certainly opportunity for improvement, the numbers here (around 0.4) indicate the model is making some headway in learning to predict bounding boxes. Better overlap between the anticipated bounding boxes and the ground truth (actual) boxes is indicated by a lower box_loss;
- Seg_loss (0.74 - 0.81): The values here (around 0.8) are rather high but during the training it keeps on decreasing until it reaches lower points at the end of the batches;
- Cls_loss (0.31 - 0.32): This is the loss that comes from categorising items that are found in the picture. These numbers, which are close to 0.3, indicate that the model may have some potential for classifying the items it finds. Reduced cls_loss suggests that the model performs better at classifying the items it detects correctly;

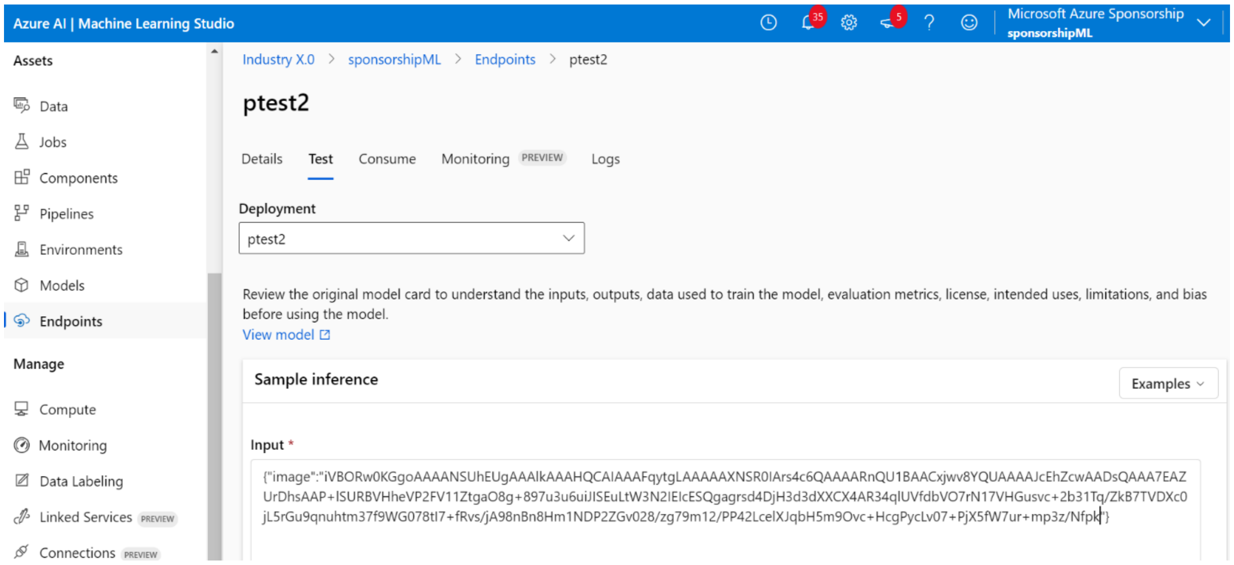
- This is a reference to the quantity of items found in the image that is currently being processed. The numbers (10 - 11) indicate that only a small number of objects are being detected by the model in each image.

## 5.5. Deployment:

The process of making a trained machine learning model available for practical application is known as a deployment. In order to receive user requests and produce predictions, it entails packing the model, creating its environment, and putting up the necessary infrastructure. In this process the previously trained models will be linked and associated as a tree, so the model will perform the way it is supposed to. And it goes through the following process:

- **Training and Evaluating the Model**: First, we used historical data to create and train our machine learning models. We built three models as it was stated before; two binary classifications using SVM algorithm combined with YOLOv8 for crack/dent segmentation;
- **Model Serialization**: After a model has been trained and verified, it must be saved in a file format that is easily loaded and utilised in production settings. This process is known as serialisation. For this project we saved our models as pytorch (a python API) format to be used later in our deployment script;
- **Deployment script:** the script is written in a development environment and it follows a certain logic to ensure the well coordination of our trained model and to be able to give the desired results when deployed. Our code started firstly by decoding the images into base-64 encoding which is a method for encoding binary data using a set of 64 printable ASCII characters to be used as JSON format. Afterwards a detection loop was initialised; it is structured in a way as fig 5-1 indicates;

- **Containerization (Optional)**: To guarantee consistency and portability across various contexts, ML models are frequently deployed inside containers. The model can be packaged together with its dependencies and runtime environment into a single container image using containerization technologies like Docker;
- **Deployment to Production Environment and strategy**: In our case we used azureML, it also supports the idea of deployment slots. Multiple versions of the model can be deployed into distinct environments (such as development, testing, and production) using deployment slots, all without interfering with the current production environment;
- **Execution of Deployment:** After everything is ready, the deployment procedure is carried out. This includes setting up the application servers, updating databases as appropriate, and moving the produced code and related files to the production environment. And it shows on the application as figure 5-2 shows the input and figure 5-3 indicates the output of JSON:



ptest2

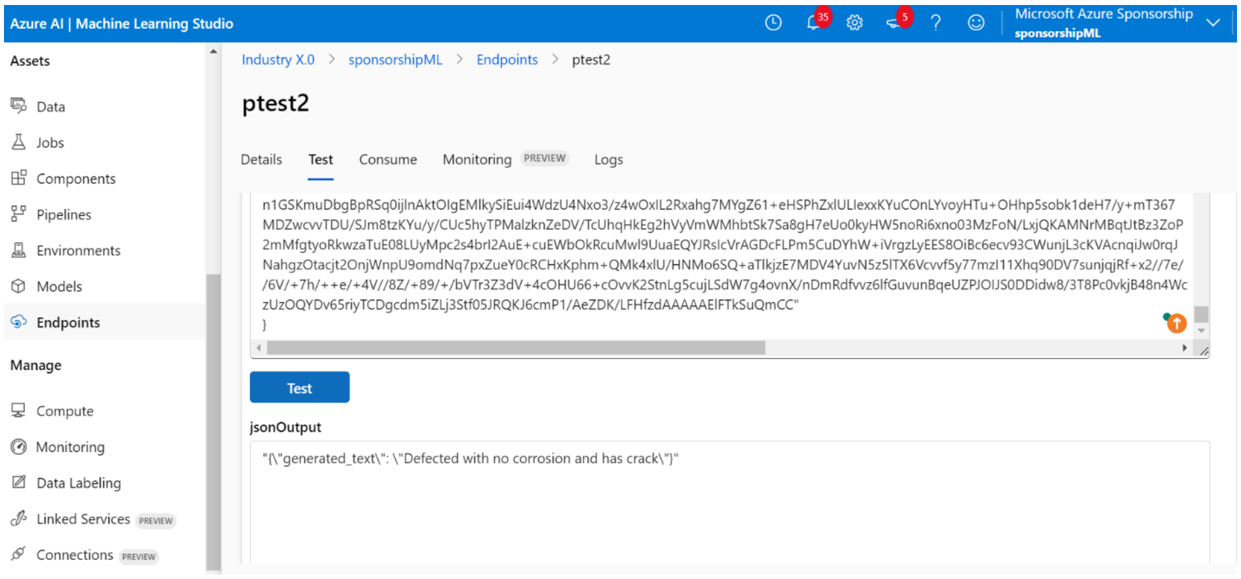
Deployment: ptest2

Review the original model card to understand the inputs, outputs, data used to train the model, evaluation metrics, license, intended uses, limitations, and bias before using the model.

Input *

{"image":"iVBORw0KGgoAAAANSUhEUgAAAlkAAAHQCAIAAAFqytgLAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsQAAA7EAZUrDhsAAP+lSURBVHheVP2FV11ZtgaO8g+897u3u6uiJISEuLtW3N2IElcESQgagrsd4DjH3d3dXXCX4AR34qlUVfdbVO7rN17VHGusvc+2b31Tq/ZkB7TVDXc0jL5rGu9qnuhtm37f9WG078tI7+fRvs/jA98nBn8Hm1NDP2ZGv028/zg79m12/PP42LcelXJqbH5m9Ovc+HcgPycLv07+PjX5fW7ur+mp3z/Nfpk"}

*Figure 0-8*: *AzureML evaluation interface*



ptest2

```
n1GSKmuDbgBpRSq0ijlnAktOIgEMlkySiEui4WdzU4Nxo3/z4wOxIL2Rxahg7MYgZ61+eHSPhZxlULIexxKYuCOnLYvoyHTu+OHhp5sobk1deH7/y+mT367
MDZwcvvTDU/SJm8tzKYu/y/CUc5hyTPMalzknZeDV/TcUhqHkEg2hVyVmWMhbtSk7Sa8gH7eUo0kyHW5noRi6xno03MzFoN/LxjQKAMNrMBqtJtBz3ZoP
2mMfgtyoRkwzaTuE08LUyMpc2s4brI2AuE+cuEWbOkRcuMwl9UuaEQYJRsIcVrAGDcFLPm5CuDYhW+iVrgzLyEES8OiBc6ecv93CWunjL3cKVAcnqiJw0rqJ
NahgzOtacjt2OnjWnpU9omdNq7pxZueY0cRCHxKphm+QMk4xlU/HNMo6SQ+aTlkjzE7MDV4YuvN5z5lTX6Vcvvf5y77mzI11Xhq90DV7sunjqjRf+x2//7e/
/6V/+7h/++e/+4V//8Z/+89/+/bVTr3Z3dV+4cOHU66+cOvvK2StnLg5cujLSdW7g4ovnX/nDmRdfvvz6lfGuvunBqeUZPJOIJS0DDidw8/3T8Pc0vkjB48n4Wc
zUzOQYDv65riyTCDgcdm5iZLj3Stf05JRQKJ6cmP1/AeZDK/LFHfzdAAAAAElFTkSuQmCC"
}
```

jsonOutput

```
"{\"generated_text\": \"Defected with no corrosion and has crack\"}"
```

*Figure 0-9: AzureML evaluation  results interface*

## 5.6. Conclusion:

In this chapter we viewed how we could build alternative models and find new ways to overcome our overfitting issues.

These previous processes need to always stay monitored and surveilled, even after deployment it may include updating documentation, notifying stakeholders, and gathering feedback from users.

# General conclusion

# General conclusion

Our dissertation delves into the application of sophisticated machine learning methods for pipeline inspection, with an emphasis on enhancing the precision of feature extraction and categorization. In order to detect and classify pipeline anomalies, we first employed Random Forest classifiers and Support Vector Machines (SVM) after feature engineering with the VGG16 convolutional neural network. Although these techniques showed promise, they suffered greatly from overfitting, especially because of the intricate nature of the inspection data and the scarcity of samples with labels.

In order to tackle these problems, we suggested an improved approach that makes use of the YOLOv8 object identification framework along with two SVM binary classifications. This dual-SVM method sought to improve binary classification precision, while YOLOv8 offered strong object identification capabilities, greatly enhancing the significantly improving the overall performance of the inspection system.

Important conclusions and contributions from this study include:

- **Feature Extraction using VGG16**: Using VGG16 for feature extraction helped to extract detailed information from pipeline inspection photos, which made the classification jobs that  followed easier;

- **Performance of SVM and Random Forest:** The need for more reliable techniques was highlighted by the initial models' good baseline performance but propensity for overfitting;
- **Optimized Approach with Dual SVM and YOLOv8:** By combining YOLOv8 with a two-stage SVM binary classification framework, overfitting was significantly reduced. Combining these two improved the model's generalization from the training set to unknown inspection circumstances, which raised classification accuracy and dependability;
- **Overfitting Mitigation:** The dual-SVM method addressed the overfitting problems seen with the original models by managing the complexity and unpredictability of the dataset by decomposing the problem into easier binary categories.

The integration of YOLOv8 with dual-SVM classifications presents a promising direction for future research, potentially extending to other areas of industrial inspection and defect detection. Future work could further refine these models and explore additional techniques to enhance robustness, scalability, and real-time application potential. In summary, this dissertation shows that combining deep learning-based feature extraction with a strategic optimization of classification models can significantly improve the effectiveness of pipeline inspection systems.

## References:

[1] Abe, S. (2005). *Support vector machines for pattern classification* (Vol. 2, p. 44). London: Springer.

[2] Chamasemani, F. F., & Singh, Y. P. (2011, September). Multi-class support vector machine (SVM) classifiers--an application in hypothyroid detection and classification. In *2011 sixth international conference on bio-inspired computing: theories and applications* (pp. 351-356). IEEE.

[3] Chen, K., Li, H., Li, C., Zhao, X., Wu, S., Duan, Y., & Wang, J. (2022). An automatic defect detection system for petrochemical pipeline based on cycle-gan and yolo v5. *Sensors*, *22*(20), 7907.

[4] Chen, Z., & Hutchinson, T. C. (2007). Urban damage estimation using statistical processing of satellite images. *Journal of computing in civil engineering*, *21*(3), 187-199.

[5] Guo, W., Soibelman, L., & Garrett Jr, J. H. (2009). Automated defect detection for sewer pipeline inspection and condition assessment. *Automation in Construction*, *18*(5), 587-596.

[5] Guo, W., Soibelman, L., & Garrett Jr, J. H. (2009). Automated defect detection for sewer pipeline inspection and condition assessment. *Automation in Construction*, *18*(5), 587-596.

[6] Halfawy, M. R., & Hengmeechai, J. (2014). Automated defect detection in sewer closed circuit television images using histograms of oriented gradients and support vector machine. *Automation in Construction*, *38*, 1-13;

[7] Hassan, S.I.; Dang, L.M.; Mehmood, I.; Im, S.; Choi, C.; Kang, J.; Park, Y.S.; Moon, H. Underground sewer pipe condition assessment based on convolutional neural networks. Autom. Constr. 2019, 106, 102849;

[8] Hawari, A., Alamin, M., Alkadour, F., Elmasry, M., & Zayed, T. (2018). Automated defect detection tool for closed circuit television (cctv) inspected sewer pipelines. *Automation in Construction*, *89*, 99-109.

[27] https://docs.python.org/3/library/os.html3

[9] https://numpy.org/

[10] https://opencv.org/

[11] https://www.adeq.state.ar.us/downloads/WebDatabases/InspectionsOnline/103062-insp.pdf

[12] https://www.flickr.com/

[13] https://www.gemini.com/

[14] https://www.googlescholar.com/

[15] https://www.Istock.com

[16] https://www.pexel.com

[17] https://www.shutterstock.com

[18] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[19] Kumar, S. S., Wang, M., Abraham, D. M., Jahanshahi, M. R., Iseley, T., & Cheng, J. C. (2020). Deep learning–based automated detection of sewer defects in CCTV videos. *Journal of Computing in Civil Engineering*, *34*(1), 04019047.

[20] Moselhi, O., & Shehab-Eldeen, T. (1999). Automated detection of surface defects in water and sewer pipes. *Automation in Construction*, *8*(5), 581-588.

[21] Myrans, J., Everson, R., & Kapelan, Z. (2019). Automated detection of fault types in CCTV sewer surveys. *Journal of Hydroinformatics*, *21*(1), 153-163.

[22] Pavithra, C., & Saradha, M. (2024). Classification And Analysis Of Clustered Non-Linear Separable Data Set Using Support Vector Machines. *Migration Letters*, *21*(S4), 901-913.

[23] Rao, A. S., Nguyen, T., Palaniswami, M., & Ngo, T. (2021). Vision-based automated crack detection using convolutional neural networks for condition assessment of infrastructure. *Structural Health Monitoring*, *20*(4), 2124-2142.

[24] Su, T. C., Yang, M. D., Wu, T. C., & Lin, J. Y. (2011). Morphological segmentation based on edge detection for sewer pipe defects on CCTV images. *Expert Systems with Applications*, *38*(10), 13094-13114.

[25] Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

[26] Yang, M. D., & Su, T. C. (2009). Segmenting ideal morphologies of sewer pipe defects on CCTV images for automated diagnosis. *Expert Systems with Applications*, *36*(2), 3562-3573.