Towards automated remote inspection of anomalies in offshore components.

TORAL QUIJAS, L.A.

2024

Towards automated remote inspection of anomalies in offshore components

Luis Alberto Toral Quijas

A report submitted as part of the requirements for the degree
of MRes in Computing: Data Science
at the School of Computing
Robert Gordon University
Aberdeen, Scotland

July 2024

Supervisor Prof Eyad Elyan, Dr Carlos Moreno-Garcia

Abstract

This dissertation marks a significant advancement in offshore structural inspections, focusing on the development, integration, and evaluation of advanced deep-learning models. The research encompasses a thorough literature review identifying innovation opportunities in deep learning for industrial inspections; the development of a General Classification Model using cutting-edge architectures for precise classification of circumferential welds; the design and training of an anomaly detection model to enhance fault identification; the implementation of a human-in-the-loop system for improved model accuracy and reliability; and a comprehensive evaluation of these models' real-world applicability.

The study not only showcases cutting-edge deep learning techniques for defect detection but also highlights critical research gaps, providing a guide for future investigation. The novel incorporation of human expertise with machine learning via a Human-in-the-Loop approach is a significant innovation, bolstering decision-making and potentially lowering error rates.

This research presents a comprehensive model that could serve as a benchmark in the field, valuable to both academics and industry professionals. It concludes by reflecting on the framework's successes and limitations, discussing its implications for offshore inspection practices, and suggesting future research directions and potential broader industry impacts.

Acknowledgements

I extend my heartfelt gratitude to the AISUS team's practical insights, Jan Stander's mentorship, the financial backing from Innovate UK, the collaborative platform provided by the North Scotland KTP Centre, Prof. Elyan's academic guidance, and Dr Carlos Moreno. Their encouragement has been instrumental in the successful completion of this dissertation. Their collective contributions have profoundly shaped this work and laid a strong foundation for future advancements in the field.

Declaration

I confirm that the work contained in this MRes project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed ..... Date .....

Luis Alberto Toral Quijas

Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction

In an era where the offshore industry prioritizes asset integrity and safety, this dissertation introduces a deep learning (DL) based framework to enhance inspection methods. This work, situated in the context of Aberdeen's significant offshore engineering advancements, proposes innovative solutions for the challenges faced in this sector.

Subsection Introductions:

1. Background: This section delves into the evolution of the offshore industry, highlighting the critical need for efficient inspection methods and the role of Aberdeen in this transformation.

2. Motivation: Discusses the driving forces behind this research, emphasizing the need for advanced inspection techniques to ensure structural integrity and safety.

3. Objectives: Outlines the primary goals of the dissertation, focusing on developing and applying a DL framework for improving offshore inspection processes.

4. Thesis Contribution: Details the unique contributions of this thesis to the field of offshore engineering, particularly in asset management and safety.

5. Thesis Structure: Provides an overview of the dissertation's structure, guiding the reader through the upcoming chapters and their significance in the broader context of this research.

1.1 Background

The offshore industry, integral to the global energy sector, is undergoing a substantial transformation in its approach to maintaining the safety and integrity of its infrastructure. Historically, this industry has predominantly relied on manual inspection methods. While foundational, these traditional techniques are increasingly recognized as inefficient, time-consuming, and unable to scale with the growing complexity of offshore operations.

Figure 1.1: View of Offshore Structure from the splash-zone area

Recent technological advancements, particularly in robotics and artificial intelligence (AI), including DL, offer promising solutions. Studies Sudevan, Shukla, and Karki 2018; Dias et al. 2022; De Tomi et al. 2014 have highlighted how technologies like wall-climbing robots and Unmanned Aerial Vehicles (UAVs) are revolutionizing inspection processes in the oil and gas industry, including those for vertical structures and offshore wind turbines. These technological innovations provide safer and more efficient alternatives to conventional methods, significantly reducing the risks and costs associated with traditional inspection processes.

Advancing AI's role in the offshore sector, a study from Alharam et al. 2020 titled 'Real Time AI-Based Pipeline Inspection using Drone for Oil and Gas Industries' showcases an innovative drone-based system for pipeline inspection. This system, equipped with a thermal camera and AI-based real-time processing, addresses the challenges of traditional human inspections by offering a safer, more efficient, and less intrusive method. Its ability to rapidly detect leakages and cracks in hard-to-access areas significantly reduces inspection costs and time, enhancing the safety and efficiency of offshore operations.

Adding to the technological strides in AI, the 'A Machine Learning Approach for

Big Data in Oil and Gas Pipelines' study emphasizes machine learning for enhancing pipeline inspections. It introduces an approach combining Magnetic Flux Leakage sensors with neural networks to process large data volumes, improving pipeline defect detection accuracy. This method not only streamlines the inspection process but also outperforms traditional methods, demonstrating the increasing relevance of AI in refining offshore inspection techniques A. Mohamed, Hamdi, and Tahar 2015

In addition to robotics, AI's role in sustainable development within the oil and gas sector has been increasingly recognized, as evidenced in the systematic literature review by Waqar, Othman, Shafiq, et al. 2023. This study underscores a positive trend in AI research related to oil and gas construction projects, highlighting AI's potential to enhance operational efficiency, reliability, and sustainability. Despite the challenges associated with AI, such as impacts on privacy and labour, its advantages in driving sustainable development are substantial and undeniable.

In line with these technological advancements, the application of DL, a branch of machine learning, is gaining traction in addressing inspection challenges in the offshore industry. For instance, Wu et al. 2021 developed a DL-based approach for Automatic Surface Defect Inspection (ASDI) that demonstrates high accuracy with limited training data. While this study primarily focuses on industrial applications, its methods can be effectively applied to the offshore sector, particularly for high-resolution image analysis of offshore structures. Such an approach could be instrumental in accurately detecting surface defects, crucial for maintaining the structural integrity of these installations.

Transitioning from general to specific applications, the study 'A DL-Based Ultrasonic Pattern Recognition Method for Inspecting Girth Weld Cracking of Gas Pipeline' enhances pipeline weld inspections. It addresses the limitations of Electromagnetic Acoustic Transducer (EMAT) technology by combining a deep Convolutional Neural Network (CNN) with a Support Vector Machine (SVM) classifier. This approach improves the signal-to-noise ratio (SNR) in detecting weld cracks and surpasses traditional methods, demonstrating DL's effectiveness in specialized inspection tasks Yan et al. 2020.

Furthermore, Langenkämper et al. 2020 demonstrated the effectiveness of DL, particularly Convolutional Neural Networks (CNN), in the visual monitoring of offshore windmill installations. This methodology allows for the accurate detection and classification of damage patterns, contributing significantly to the efficient inspection and maintenance of the growing number of offshore windmills.

Similarly, Xia et al. 2018 explored a DL-based image recognition and processing model

for electric equipment inspection. Their work highlights the potential of DL in automating and improving the accuracy of inspections, an essential aspect of maintaining offshore operations.

In conclusion, the offshore industry's transition towards DL-based inspection methods represents a necessary evolution driven by the need to address growing complexities and heightened safety standards. DL frameworks are emerging to complement pivotal technologies, promising to redefine this sector's inspection and maintenance paradigms. The subsequent chapters will delve into these technologies' theoretical underpinnings and practical applications in offshore inspections, highlighting their potential to significantly enhance safety and operational efficiency.

1.2 Motivation

The offshore industry faces challenges in ensuring the integrity and reliability of its structures. Traditional inspection methods have limitations in efficiency, risk, and comprehensiveness, especially under the strenuous conditions of offshore environments.

The motivation behind this work is twofold: firstly, to demonstrate how DL can enhance the precision and thoroughness of offshore structural assessments, thereby contributing to the overall safety of these critical installations. Secondly, this research seeks to showcase the potential of AI-driven approaches in revolutionizing traditional practices, setting a precedent for future technological integration in offshore engineering.

Through this investigation, this thesis aims to substantiate the premise that DL is a viable tool and a necessary evolution in the ongoing effort to uphold the highest safety and efficiency standards in offshore operations. The findings and developments presented herein aspire to contribute significantly to the body of knowledge in offshore engineering, potentially reshaping inspection practices and fostering greater operational effectiveness and safety.

1.3 Objectives

This dissertation aims to advance the state-of-the-art in automated visual inspection systems by developing, integrating, and evaluating DL models. The specific objectives set to achieve this goal are as follows:

1. **Examine and Critically Evaluate DL-based Methods:** Investigate how DL has been applied in the context of condition monitoring and remote inspection in industrial settings. This will involve critically evaluating existing literature's

methodologies, tools, and outcomes to identify gaps and opportunities for innovation in automated visual inspection systems.

2. **Develop an Image Classification Framework:** Create an image classification framework utilizing advanced DL architectures. This framework should be capable of accurately classifying various conditions in engineering images, focusing on versatility and adaptability to different types of engineering environments and challenges.

3. **Create Anomaly Detection Methods:** Develop and train a machine learning model specifically for anomaly detection within engineering images. This method should enhance the fault identification process in automated inspection systems, emphasising precision and reliability in diverse operational scenarios.

4. **Implement a Human-in-the-Loop API:** Establish a system for integrating human expert feedback into the model's decision-making process. This can be achieved through API integration, enabling a collaborative approach where human expertise supplements the automated system, thereby improving the model's accuracy and reliability.

1.4 Thesis Contribution

To succinctly summarize the key contributions of your thesis in bullet points, you can focus on the three most significant advancements your research offers. Here's a concise version:

1. **Examination and Critical Evaluation of DL Methods:** Thorough investigation of DL applications in condition monitoring and remote inspection in industrial settings, critically evaluating existing methodologies to identify gaps and opportunities for innovative solutions in automated visual inspection systems.

2. **Development of an Image Classification Framework:** Creation of a versatile and adaptable image classification framework using advanced DL architectures to accurately classify various conditions in engineering images across different engineering environments.

3. **Advancement in Anomaly Detection Techniques:** Development of a machine learning model for anomaly detection in engineering images, enhancing fault identification processes in automated inspection systems focusing on precision and reliability in diverse operational scenarios.

1.5 Thesis Structure

This thesis is structured into chapters, each targeting a specific objective:

Chapter 2 Literature Review: Focuses on Objective 1 by reviewing DL applications in industrial inspection and defect recognition. It critically evaluates existing literature to identify gaps and opportunities for innovation in automated visual inspection systems.

Chapter 3 Design: Corresponds to Objective 2, detailing the creation of an image classification framework using advanced DL architectures. The chapter discusses the framework's design and adaptability to various engineering environments.

Chapter 4 Implementation: Aligns with Objective 3 and Objective 4. It covers developing and training an anomaly detection model and implementing a human-in-the-loop system via an API, focusing on enhancing fault identification and model accuracy.

Chapter 5 Evaluation & Testing: Dedicated to the evaluation and testing of the developed models, covering performance metrics such as accuracy, precision, recall, and F1-score. This chapter ensures that the models meet the required standards of reliability and effectiveness.

Chapter 6 Conclusion: Synthesizes the research findings, reflecting on the achievements and limitations of the work. It discusses the broader implications for the future of automated inspection systems and suggests directions for future research.

Chapter 2

Literature Review

Before delving into the transformative role of DL and transfer learning in industrial inspections, it is essential to provide some context on the domain knowledge and AI techniques pertinent to this field. The offshore industry, integral to the global energy sector, demands rigorous inspection and maintenance practices to ensure the safety and integrity of its infrastructure. Over the years, AI techniques, particularly DL, have emerged as powerful tools in enhancing these inspection processes. DL, a subset of machine learning, involves training neural networks on large datasets to perform highly accurate tasks such as image and pattern recognition. This chapter aims to bridge the gap between traditional inspection methods and modern AI applications, setting the stage for a comprehensive understanding of how DL can revolutionize inspection practices in the offshore industry.

This Literature Review chapter delves into the transformative role of DL and transfer learning in industrial inspections, particularly emphasising the energy sector. We commence by exploring a range of DL methodologies, underscoring their significant advancements in enhancing the accuracy and reliability of defect detection in complex environments. The evolution of these technologies is traced from traditional manual methods to sophisticated, data-driven approaches. Emphasizing the importance of transfer learning, we highlight its critical role in overcoming challenges posed by data scarcity and harsh environmental conditions in the energy sector.

2.1 Inspection Challenges in the Energy Sector

The complexities highlighted in 'Digitisation of Assets from the Oil & Gas Industry: Challenges and Opportunities' underscore the broader context of challenges in the offshore energy sector. This paper delves into the hurdles of digitizing vital engineering documents like Piping and Instrumentation Diagrams (P&IDs), essential for risk assessments yet complex due to issues like document quality, skewed data distribution, and topology. It reviews advanced methodologies to address these challenges, illustrating the need for integrated digital solutions. Such technologies could streamline processes, enhance defect detection accuracy, and improve critical assets' safety and reliability. Moreno-Garcia and Elyan 2019.

In conjunction with these digitization challenges, it is crucial to acknowledge the unique environmental and geographical challenges faced in offshore operations. The paper 'Challenges for the Inspection of Pre-Salt Ultra-deep Offshore Production Facilities' offers insight into these specific challenges. Concentrating on the Brazilian Pre-Salt oilfields located in deep waters far from the coast highlights the necessity of using ROVs for inspections. It emphasizes the need for sophisticated management of technology and inspection processes in these demanding deep-sea environments. This research points out how extreme conditions and logistical complexities compound the challenges of maintaining operational integrity, reinforcing the imperative for innovative inspection methods and robust asset management strategies in the oil and gas industry. De Tomi et al. 2014.

Building on this understanding of the diverse challenges in the sector, managing ageing offshore energy infrastructure, especially in critical areas like the splash zone, becomes a focal point. Often characterized by harsh environmental conditions, these zones demand advanced inspection techniques to ensure infrastructure integrity and safety. Such areas are subject to varying submersion levels due to tides and winds, making them particularly challenging to inspect and maintain. Therefore, developing and integrating sophisticated inspection and digitization technologies are beneficial and essential for efficiently managing these vital offshore components.

These offshore components are categorized into three primary segments: topside, splash-zone, and sub-sea (see Figure 2.1). The topside is the area under the deck, typically dry and less exposed to marine conditions. In contrast, the splash zone experiences intermittent submersion due to tides and winds, making it a critical area for inspection. The sub-sea segment, typically the most extensive, requires thorough inspection due to its constant exposure to marine conditions and higher likelihood of anomalies.

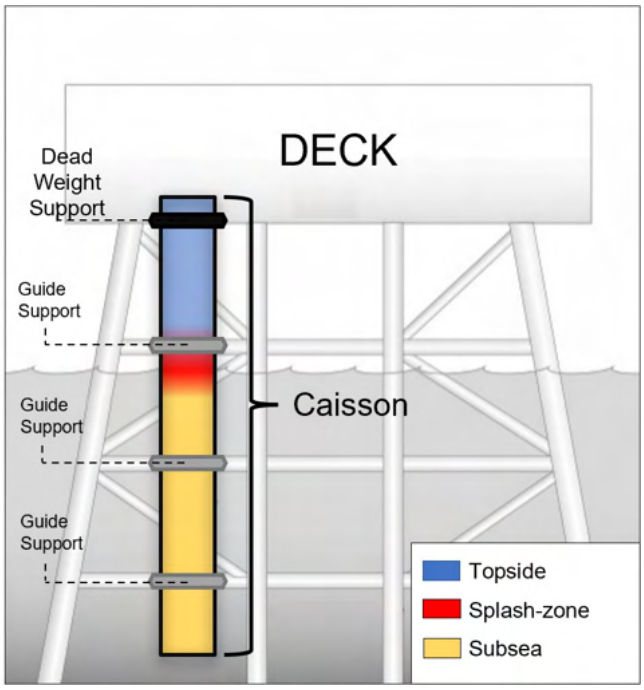


Figure 2.1: This figure provides a detailed overview of the different zones of an offshore component, illustrating the varied environmental conditions each section faces.

In regions like the United Kingdom Continental Shelf (UKCS), there has been an increasing concern over the past decades regarding the deterioration and failure of critical offshore components, including caissons, J-tubes, conductors, and risers. These failures not only pose risks to the structural integrity of the installations but can also lead to severe consequences like gas leaks and operational disruptions. These components' intricate nature and exposure to aggressive marine environments underscore the need for more sophisticated inspection methodologies.

One of the biggest threats to the integrity of these components is internal corrosion. Due to the loss of coating or lining, the interior of these components is vulnerable to corrosion, and only a minimal amount of protection is provided by the external coating protection (CP) system. For instance, pump caissons can suffer severe corrosion damage caused by galvanic action between the pump and the caisson. Thus, internal corrosion shall be monitored by conducting a periodical internal inspection.Regulator 2021

The two most common inspection techniques in offshore operations are general visual inspection (GVI) and closed visual inspection (CVI). GVI, typically conducted by remotely operated vehicles (ROVs), serves to verify the presence of components and identify any major flaws, deformations, or damage without the need for pre-cleaning. In contrast, CVI is employed for a more detailed assessment of the component's condition and requires prior cleaning of marine growth to identify local defects or damage

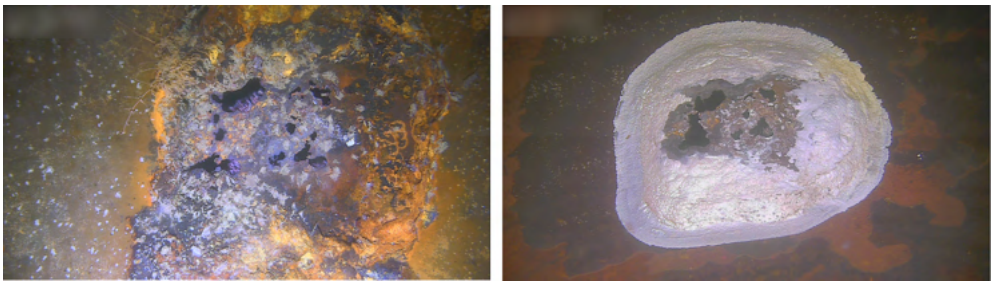


Figure 2.2: This figure illustrates a caisson undergoing inspection, showing the difference in anomaly visibility before (left) and after (right) the cleaning process, highlighting the importance of surface preparation in accurate defect detection.

accurately. This technique often results in a comprehensive report comprising still images of identified anomalies, which are critical for subsequent analyses (See Figure 2.2). Health and Executive 2009

The process of a full-caisson inspection is comprehensive, beginning with surface cleaning and preparation, followed by a remote ultrasonic inspection, and culminating in a remote visual inspection. The initial phase involves high-pressure water jetting to remove loose coatings and surface corrosion, with the resulting debris either released to sea or collected in recovery baskets. Next, robotic ultrasonic inspection tools are deployed to measure the caisson's thickness along its entire length, providing real-time data essential for preliminary condition assessment. The final stage involves remotely deploying inspection cameras to visually confirm any defects or irregularities detected during the ultrasonic inspection, with particular attention to the surface condition and circumferential welds.

In offshore structures, the integrity of welded components is of paramount importance. Welded joints inherently retain residual stresses, which can be as significant as the material's yield strength. In engineering structures, such tensile residual stresses are detrimental to structural integrity. Thus, a critical aspect of remote visual inspections involves assessing these welds, especially circumferential welds (CW), which are vulnerable to localized corrosion and fatigue. These stresses are often exacerbated by external forces such as ocean currents and tidal movements, making CWs a focal point in structural assessments. Y. Zhang et al. 2013a; Y. Zhang et al. 2013b.

The task of inspecting and assessing these components is not only critical but also time-intensive. Classifying circumferential welds from the images captured during an inspection is laborious and meticulous. Each circumferential weld, potentially manifesting different forms of deterioration or damage, must be examined precisely. An inspection engineer is tasked with capturing a comprehensive 360-degree view of the

Figure 2.3: Example of a 180° internal panoramic view of a component circumferential weld with defects.

inner circumference of the component, which typically results in a collection of 20 to 26 images for a mere fraction of a meter. Given the extensive length of these components, ranging from 30 to 70 meters, and the frequency of circumferential welds occurring every few meters, the volume of images generated for a single inspection can be substantial.

Moreover, the classification phase is just the precursor to the critical analysis of each identified weld. Post-classification, the engineer must scrutinize each image for defects, requiring a high level of expertise and an eye for detail to ensure no anomaly goes unnoticed. The defects in circumferential welds, such as cracking, corrosion, or deformation, are varied and may also be subtle and difficult to detect, adding further complexity to the task. The manual process, inherently prone to human error, is further exacerbated by factors such as poor lighting conditions, biofouling, and the turbidity of the water, which can obscure the visibility of the defects.

Therefore, this exhaustive and demanding process underscores the critical need for more efficient and reliable inspection methods. The imperative for innovation in this field is clear: enhancing defect detection accuracy, expediting the assessment process, and reducing human error risks are essential for safeguarding the safety and integrity of these vital sub-sea structures.

2.2 DL Applications in the Energy Sector

Advancements in DL have significantly enhanced monitoring and maintenance practices within the energy sector. These technological developments are pivotal for inspecting submerged structures and vast pipeline networks, ensuring their integrity and functionality. The application of DL extends to detecting minute surface anomalies, potentially indicating more significant underlying issues. Moreover, transfer learning has emerged as a powerful approach, enabling efficient and practical analysis in environments where conventional methods fall short. The forthcoming sections will delineate using DL for underwater inspection, pipeline integrity assessment, surface defect identification, and the application of transfer learning in challenging settings.

2.2.1 Underwater Monitoring and Corrosion Detection

The advent of DL has significantly advanced underwater monitoring and corrosion detection, which is essential for maintaining subsea infrastructure. While machine vision applications in these domains have been innovative, several challenges and limitations remain.

O'Byrne et al. 2018 explored semantic segmentation of underwater imagery using deep networks trained on synthetic data, focusing on bio-fouling detection. While their framework achieved commendable accuracy, the reliance on synthetic data may not fully capture the complexity of real-world underwater environments. The transition from synthetic to real-world data remains a significant challenge, potentially impacting the model's practical applicability.

De Masi et al. 2015 utilized machine learning to predict corrosion rates in subsea pipelines. Their approach, processing a substantial amount of inspection data, showed the feasibility of predictive models. However, the study's effectiveness in real-world scenarios may be limited by the variability in environmental conditions and the dynamic nature of subsea corrosion processes, which are not always predictable.

Soares et al. 2021 focused on classifying underwater corrosion levels by training a CNN with modified gamma variables in images. While their method showed high accuracy, the simulation of turbidity effects may not fully represent the diverse and unpredictable conditions encountered in natural underwater settings. Additionally, the study highlights the need for more robust models that can adapt to varying levels of visibility and environmental factors.

Adding to these advancements, a recent study by Smith, Coffelt, and Lingemann 2022 presents "A DL Framework for Semantic Segmentation of Underwater Environments." This research addresses the challenge of object classification and segmentation in underwater robotic missions, often hindered by water turbidity and light attenuation. The study proposes a unique solution involving the procedural creation of randomized underwater scenes and the generation of corresponding point clouds with semantic labels. This approach, using synthetic data for training a 3D segmentation network, shows promise in overcoming the limitations of traditional perception methods in harsh underwater conditions. The network's performance, validated on actual underwater point cloud data, signifies a step forward in enhancing the accuracy of underwater monitoring using DL.

The study presented by Prasad, Parikh, and Prasanth 2023 proposes an algorithm for detecting and classifying various underwater objects, emphasizing the versatility of DL in diverse applications. However, the effectiveness of such an algorithm in complex and

turbid underwater environments, where visibility is often compromised, is yet to be fully explored. The study allows further research to improve image clarity and object detection accuracy in challenging underwater conditions.

Transfer learning, as investigated in Pirie and Moreno-Garcia 2021 for underwater corrosion segmentation, shows promise in enhancing model performance. However, adapting models from surface to underwater scenarios raises questions about the generalizability of such approaches. The preprocessing techniques required for this transition need further refinement to ensure robust performance across different underwater environments.

Anastasios et al. Stamoulakatos, Cardona, Mccaig, et al. 2020 demonstrated using ResNet-50 CNN for the multi-label classification of underwater inspection images. While their model achieved high accuracy, the scalability of such a system in extensive, real-world underwater environments poses a challenge. Additionally, the study highlights the need for large, diverse datasets for training to ensure the model's effectiveness across various underwater scenarios.

These studies highlight the growing impact of DL in underwater monitoring and corrosion detection. They offer promising directions for future research, particularly in enhancing data realism, model generalizability, and robustness in diverse and challenging underwater conditions. Future research should focus on bridging the gap between controlled experiments and real-world applications, ensuring the practical deployment of these technologies in subsea infrastructure maintenance.

2.2.2 Pipeline and Structural Integrity Assessment

The integrity of pipelines and structural components is crucial in the energy sector, with corrosion as a primary challenge. While recent research leveraging DL shows promising results, it's essential to critically evaluate these advancements in the context of practical applications and inherent limitations.

Bastian et al. 2019 introduced a DL framework using Convolutional Neural Networks (CNNs) for pipeline corrosion detection. While their approach marks a significant step in defect identification, the model's generalizability across different pipeline materials and environments remains a question. Additionally, the reliance on high-quality image data may limit its applicability in less ideal conditions often encountered in real-world scenarios.

Hoang and Duc 2019 combined image texture analysis with a machine-learning algorithm for automating corrosion detection. Their method's integration of traditional

and modern techniques is innovative; however, the model's performance in varied lighting and weather conditions, typical in outdoor pipeline environments, needs further exploration. The effectiveness of the meta-heuristic optimization in diverse settings also warrants additional validation.

A novel approach proposed by Bhavani et al. 2022 that integrates DL with traditional non-destructive testing methods advances pipeline inspection methodologies. This research enhances magnetic flux leakage (MFL), a common technique for pipeline inspection, using DL to analyze patterns in MFL signals. By classifying these signals through a convolutional neural network and assessing damage indices based on enveloped MFL signal and threshold values, this method proposes a more quantitative and real-time approach to pipeline integrity assessment. Integrating DL in this traditional NDT method underscores the potential for advanced technologies to enhance the accuracy and efficiency of pipeline inspections.

Using aerial imagery, Ortiz et al. 2016 developed new descriptors for detecting coating breakdown corrosion. Their method emphasizes the importance of feature extraction in machine vision. Nonetheless, the accuracy of these descriptors in differentiating between corrosion and similar-looking anomalies in complex backgrounds is an area that could benefit from more research, especially considering the vast range of textures and colours present in outdoor environments.

Mazzella et al. 2020 offered a model combining machine learning with geospatial analysis for pipeline wall loss risk estimation. While this approach is comprehensive, the challenge lies in its scalability and adaptability to different pipeline networks with varying data availability and quality. Additionally, integrating real-time data for dynamic risk assessment could enhance the model's utility.

These studies represent a significant stride in applying machine vision and DL to assess pipeline and structural integrity. They offer robust tools for the energy sector yet underscore ongoing research's need to address challenges like model generalizability, performance in diverse and unpredictable conditions, and scalability. Future research should focus on enhancing these models' robustness and adaptability, ensuring their effective deployment in varied and real-world settings, where conditions are often far from ideal. This progression is essential for developing truly reliable and efficient tools for structural integrity assessment in the energy sector.

2.2.3 Surface Defect Detection and Classification

The detection and classification of surface defects are crucial in maintaining the integrity and safety of infrastructural components. Profound learning advancements have notably improved automated surface defect recognition, yet a critical evaluation of these methods reveals areas for further development.

Fu et al. 2019 developed a Convolutional Neural Network (CNN) model for steel surface defect classification using a SqueezeNet pretrained architecture and a multi-receptive field module. While their model achieved high accuracy, the challenge remains in its application to diverse surface materials under varying environmental conditions. The model's performance in real-world industrial settings, where data may be more irregular and less controlled, is an essential area for future research.

Ren, Hung, and K. C. Tan 2018 introduced a generic approach to automate surface inspection, creating a defect heat map using a pretrained DL network. This method streamlined the defect detection process and demonstrated adaptability to different datasets. However, the study's reliance on pretrained networks raises questions about the model's specificity and sensitivity in unique defect scenarios, suggesting a need for customization in applications with distinct defect characteristics.

Similarly, Altabey et al. 2022 further extends the application of DL in surface defect detection. This research focuses on efficiently and accurately monitoring pipeline surface cracks using a CNN algorithm, enhancing the traditional Magnetic Flux Leakage (MFL) method. The approach involves detailed analysis and classification of MFL signals, applying DL to improve the identification and localization of pipeline cracks. While demonstrating promising results, this method's effectiveness in varying pipeline conditions and its computational efficiency in large-scale applications warrant further investigation and optimization.

Building on the theme of specialized defect detection, Medak et al. (2022) Medak et al. 2022 proposed "DefectDet," a novel DL architecture for detecting defects in ultrasonic images, commonly used in non-destructive testing (NDT). This architecture is tailored to handle the extreme aspect ratios often found in ultrasonic images, employing a lightweight feature extractor and a modified detection head. Tested on an extensive in-house dataset, their architecture achieved superior performance over state-of-the-art methods while also reducing inference time, showcasing the potential of DL in specialized NDT applications.

Lin, Chou, and Cheng 2023 tackled dataset variability and model generalization in Automated Optical Inspection (AOI). Their DL-based General Defect Detection Framework (DL-GDD) addressed insufficient defect samples and achieved high accuracy and

low false rates, demonstrating DL's potential in AOI.

Zhu, Ge, and Z. Liu 2019 combined CNN with random forest classifiers for weld surface defect identification. Their approach effectively learned high-level features and provided robust predictions. Still, integrating machine learning techniques like random forests with DL models may require further refinement to optimize performance and computational efficiency.

Atha and Jahanshahi 2018 compared various CNN architectures for corrosion assessment on metallic surfaces. Their findings indicated the superiority of CNNs, particularly when fine-tuned, over traditional corrosion detection methods. However, the fine-tuning process demands extensive datasets and computational resources, which could be limiting factors in some practical applications.

Huang, Li, and D.-m. Zhang 2018 proposed a DL-based method for identifying cracks and leakage in metro shield tunnels using a two-stream algorithm for semantic segmentation. While their method outperformed traditional segmentation methods, the increased computational time required for processing poses a challenge for real-time applications.

Collectively, these studies underscore the effectiveness of DL in surface defect detection and classification. They offer promising directions for maintenance and safety in various sectors, including energy. Future research should focus on enhancing the models' applicability in diverse real-world scenarios, improving computational efficiency, and addressing the challenges of dataset variability and environmental conditions. This progression is crucial for advancing reliable and efficient tools for surface defect assessment in practical applications.

2.3 Review of DL Frameworks for Inspection

To advance DL and computer vision for analysing offshore inspection images, our strategy is leveraging the most effective models tailored for such intricate tasks. The architectures of choice are the Vision Transformer (ViT) for its novel approach to image classification and EfficientNet for its optimized balance of model depth, width, and resolution.

2.3.1 Vision Transformer

The Vision Transformer (ViT) introduces an innovative approach to image classification, applying transformer models from natural language processing. Diverging from

the conventional CNNs, ViT leverages a self-attention mechanism, allowing it to analyze various parts of an image about each other, enhancing the overall understanding of the image's content Dosovitskiy, Beyer, Kolesnikov, et al. 2020.

ViT processes an image by dividing it into fixed-size patches, linearly embedding them along with position encoding. These embedded patches are then inputted into transformer encoder layers, which consist of multi-head self-attention and fully connected neural networks. This structure allows ViT to capture detailed features and broader contextual information within the image Dosovitskiy, Beyer, Kolesnikov, et al. 2020.

A key advantage of ViT is scalability, improving performance with larger training datasets, and its ability to benefit from transfer learning. This makes it well-suited for tasks with extensive data, such as image classification on large corpora like ImageNet or JFT-300M Dosovitskiy, Beyer, Kolesnikov, et al. 2020, Kolesnikov, Beyer, Zhai, et al. 2019.

In practical applications, Komijani et al. (2022) Komijani et al. 2022 demonstrated ViT's adaptability by using it for multi-label classification of steel surface defects, achieving an impressive weighted F1 score of 92%. This highlights ViT's capability to handle complex classification tasks in various industrial settings.

Furthering the application of ViT, Liu et al. (2023) Y. Liu et al. 2023 developed the Fast Multi-Path Vision transformer (FMPVit), an optimized version for welding defect detection. FMPVit addresses the limitations of standard transformer models, such as large parameter sizes and computational resource demands, by enhancing local feature detection capabilities through an advanced architecture. Validated on a dataset of weld seams, FMPVit showed significant improvements over traditional models.

By expanding on these advancements, Chen et al. (2024) Chen et al. 2023 introduced a hybrid method that combines YOLOv5 and ViT to detect pipeline defects. This approach, employing a cascaded DL framework, surpasses YOLOv5 alone in accuracy, illustrating the effectiveness of combining different architectures for intricate defect detection tasks.

ViT's comprehensive image analysis and sequence-to-sequence classification approach are highly beneficial for offshore inspections, where identifying subtle anomalies is crucial. Its scalability and ability to fine-tune on specific datasets make ViT an apt choice for analyzing the integrity of structures like circumferential welds, thereby enhancing the reliability of remote visual inspections.

In conclusion, ViTForImageClassification, with its global receptive field and scalability, is poised to set new standards in offshore inspection, transforming image-based

structural analysis through advanced DL techniques.

2.3.2 EfficientNet

EficientNet represents a paradigm shift in the scaling of convolutional neural networks (CNNs), employing a compound scaling method for balanced adjustments in network depth, width, and resolution. This harmonized scaling enhances model capacity and accuracy M. Tan and Le 2019.

EfficientNet's innovative scaling technique improves performance without disproportionately increasing computational costs. Starting with the baseline EfficientNet-B0, the model is scaled to more potent versions like EfficientNet-B7, depending on the complexity of the problem and computational resources available M. Tan and Le 2019.

Li et al. (2023) Hoang and Duc 2019 and de Moura et al. (2022) de Moura et al. 2022 demonstrated EfficientNet's application in submarine pipeline inspection and deep-water oil-spill monitoring, highlighting its adaptability in diverse scenarios.

Further showcasing EfficientNet's versatility, Yu et al. (2022) Yu et al. 2022 explored its application in the intelligent detection of forging defects. They developed an improved model using EfficientNet as the backbone and Feature Pyramid Network (FPN) as the fusion layer, enhanced with an Attention Mechanism and optimized by a Particle Swarm Optimization algorithm. This approach significantly improved the detection accuracy of automobile steel forging defects, achieving a mean Average Precision (mAP) of 95.69% and an F1 score of 0.94. The study exemplifies EfficientNet's potential in industrial quality control, offering a highly accurate and efficient solution for defect detection in manufacturing processes.

EfficientNet's adaptability in applications ranging from environmental monitoring to industrial defect detection underlines its suitability for offshore inspection image analysis. Its ability to efficiently scale and capture a wide range of features makes it a valuable tool for customized approaches to analyzing complex visual data.

In conclusion, incorporating EfficientNet into our computational pipeline for offshore inspection and environmental monitoring tasks leverages its balance between efficiency and performance, enabling accurate analysis and decision-making in diverse offshore and marine applications.

2.3.3 You-Only-Look-Once (YOLO)

The YOLO (You Only Look Once) series of models is among the most prominent architectures for real-time object detection. YOLO's fundamental innovation lies in its

unified architecture, which simultaneously predicts bounding boxes and class probabilities in a single forward pass through the neural network. This design allows for rapid object detection, making YOLO highly suitable for real-time analysis applications like video surveillance, autonomous vehicles, and industrial inspections.

YOLO models divide the input image into a grid, and each grid cell predicts a certain number of bounding boxes. For each bounding box, the model computes a confidence score reflecting an object's presence and the box's accuracy. In parallel, the model predicts class probabilities for each grid cell. This approach allows YOLO to capture spatial context and object classification highly efficiently.

You Only Look Once (YOLO) is an innovative object detection method that revolutionized the field of computer vision with its introduction in 2015 by Joseph Redmon et al. Unlike traditional object detection systems which employ separate models for classification and localization, YOLO unifies the entire object detection pipeline into a single neural network, enabling it to perform detection in real-time. The architecture processes the input image only once (hence the name) through a single convolutional neural network (CNN) and divides the image into a grid. Each grid cell predicts a certain number of bounding boxes and confidence scores for those boxes, including class probabilities. This end-to-end learning approach allows YOLO to predict bounding boxes and class labels simultaneously for all classes, making it exceptionally fast and efficient compared to its predecessors.Redmon et al. 2016

YOLOv8, developed by Ultralytics, builds upon the strengths of previous versions, offering several improvements. It inherits the unified, single-stage detector characteristic, enhancing it with a more refined architecture and advanced features. YOLOv8's design is optimized for speed and accuracy, with substantial improvements allowing it to outperform its predecessors. It provides state-of-the-art object detection capabilities with increased precision and real-time performance. It is ideal for applications requiring rapid and reliable visual recognition, such as offshore inspection tasks. Gašparović et al. 2023a

This latest iteration maintains compatibility with both CPU and GPU environments, ensuring accessibility and scalability. The YOLOv8 model stands out for its ability to immediately handle complex object detection, classification, and segmentation tasks. It is equipped with a suite of pre-trained weights and can be fine-tuned on custom datasets to enhance its performance on specialized tasks, such as identifying defects in welds during inspections. The versatility and power of YOLOv8 lie in its advanced engineering, which integrates lessons learned from earlier versions and incorporates new techniques from current research to push the boundaries of what's possible in real-time

object detection.

2.3.4 Transfer Learning

Transfer learning has emerged as a powerful tool in machine learning, particularly in environments where data collection is challenging or where the environment is harsh and unpredictable.

Training DL models from scratch is a resource-intensive task that often requires large datasets and significant computational power. Sarker 2021 In the context of offshore inspections, where data can be scarce and computational resources are limited, this becomes a substantial challenge. Transfer learning addresses these challenges by utilizing a model pre-trained on a large dataset and adapting it to the specific task at hand, thereby leveraging the generic features learned by the model on a broad range of images.

Ali et al. Ali et al. 2020 leveraged transfer learning to accurately detect cracks on concrete surfaces. By using pre-trained models like MobileNet and Inception V2, they demonstrated the potential of transfer learning to adapt to new, similar tasks effectively.

Liang Lu 2021 evaluated various DL methods, including transfer learning, to identify pavement cracks in visual images. By comparing different datasets and models, Liang's work underscores the adaptability of transfer learning to diverse image-based detection tasks.

In the context of building inspections, Perez et al. Perez, Tah, and Mosavi 2019 utilized a pre-trained CNN classifier to detect and locate building defects. Their work showed how transfer learning could reduce the need for extensive data labelling while still achieving high accuracy in defect classification.

Mohammed et al. Y. S. Mohamed et al. 2019 applied transfer learning to estimate steel crack depth from 2D images, demonstrating that neural networks can accurately estimate physical attributes from visual data. Akira et al. Oyama et al. 2021 employed DL-based methods, including transfer learning, to detect rust in pressure pipes from 2D images, highlighting the method's efficacy in pattern recognition tasks even with complex visual patterns.

Lastly, in the study by authors in Pirie and Moreno-Garcia 2021, transfer learning was explored for segmenting underwater corrosion from imagery. The study indicates that when fine-tuned with task-specific data, pre-trained models can improve segmentation results, showcasing the potential of transfer learning in underwater imaging applications.

These studies collectively show the versatility of transfer learning in adapting pre-trained models to new domains, particularly in harsh environments where traditional data gathering and processing methods are insufficient or impractical.

2.4 Challenges and Limitations of Current DL Approaches

This section critically evaluates the challenges and limitations inherent in applying DL techniques to the energy sector, as identified in the preceding sections of this literature review.

2.4.1 Data Scarcity and Quality

The efficacy of DL models is heavily contingent on the availability and quality of training data. However, as identified in Section 2.1, the offshore energy sector frequently grapples with data scarcity and inconsistent data quality, posing significant challenges for these models.

Data Acquisition Challenges: Collecting extensive datasets in offshore environments is a formidable task due to various factors. Logistical hurdles, high operational costs, and the inherently hazardous nature of offshore environments contribute to the scarcity of data. Additionally, the sporadic nature of inspections and maintenance activities results in datasets that are limited in size and sporadically collected, leading to temporal gaps in the data.

Annotation and Labeling Issues: Another critical aspect is the labour-intensive and expert-driven process of annotating and labelling the data, which is essential for supervised learning models. The complexity and specialized nature of offshore inspection data require expert knowledge for accurate labelling, further complicating the data preparation process.

While techniques such as data augmentation, synthetic data generation, and transfer learning have been proposed to mitigate these issues, they have limitations. Data augmentation can introduce biases, synthetic data may not capture the full complexity of real-world scenarios, and transfer learning relies on the assumption that pre-trained models on large datasets apply to the specific needs of the offshore sector.

2.4.2 Environmental and Operational Variability

The offshore energy sector is characterized by its challenging and variable environmental and operational conditions, which significantly impact the application and effectiveness

of DL models.

Impact of Environmental Conditions: As highlighted in Section 2.3, offshore environments pose unique challenges such as poor visibility, underwater turbidity, and extreme weather conditions. For instance, the studies on underwater monitoring and corrosion detection (O'Byrne et al. 2018, De Masi et al. 2015) illustrate how environmental factors like bio-fouling and turbidity can drastically affect the quality of visual data, posing a significant challenge for image-based DL models. These conditions can lead to substantial inaccuracies in tasks like defect detection and structural assessment.

Variability in Operational Settings: The operational settings in the energy sector vary widely, as evidenced by the different inspection techniques used for diverse structures like pipelines, caissons, and risers. Each structure and environment demands specific inspection methods, as indicated by the studies on pipeline integrity and structural assessment (Bastian et al. 2019, Hoang and Duc 2019). The success of DL models in one scenario does not guarantee their effectiveness in another, underscoring the need for adaptable and robust models.

Challenges in Model Training and Adaptation: Training DL models to handle this environmental and operational variability is challenging. The models need to be robust enough to handle diverse and often suboptimal data conditions. The pipeline and structural integrity assessment study using aerial imagery (Ortiz et al. 2016) demonstrates the complexity of extracting relevant features in varied lighting and weather conditions.

Current DL approaches often rely on controlled or ideal conditions for training, which may not represent real-world offshore environments. Techniques such as augmenting training data with environmental noise and using simulation environments offer potential solutions, but they may not fully capture the complexity and unpredictability of real-world conditions.

2.4.3 Integration with Existing Systems

Integrating DL solutions into the established operational frameworks of the offshore energy sector presents a complex challenge, as noted in various parts of the literature review.

Compatibility Issues: A critical hurdle, as indicated in Section 2.3, is the compatibility of DL models with existing inspection and monitoring systems. The specialized nature of offshore operations often relies on legacy systems that may not be readily compatible with advanced AI technologies. For instance, the integration of AI-based corrosion detection or structural assessment tools (De Masi et al. 2015, Bastian et al.

2019) into existing pipeline inspection workflows requires careful consideration of data formats, software platforms, and operational protocols.

Workflow Modification and Training: Incorporating AI solutions often necessitates significant modifications to existing workflows, as seen in the application of models like YOLO for real-time object detection (Redmon et al. 2016). This integration is not just a technical challenge but also an operational one, requiring the staff retraining and potentially restructuring inspection processes. The change management aspect, including training personnel to work alongside AI systems, is critical to successful integration.

Interdisciplinary Collaboration: Effective integration demands a collaborative approach that combines expertise in AI and DL with domain-specific knowledge from the energy sector. This interdisciplinary collaboration is essential for addressing the practical and technical nuances of integrating DL into existing systems.

Resistance to Technological Changes: Another significant barrier is the resistance to change within organizations. The introduction of AI-based methods can be met with skepticism, especially in a field where traditional methods have been long-established and trusted. Overcoming this resistance requires demonstrating the clear benefits of AI integration, such as improved accuracy, efficiency, and safety in inspections.

While the integration of DL presents a path to enhanced efficiency and accuracy in offshore operations, the practical challenges of such integration must not be underestimated. Solutions need to be user-friendly, cost-effective, and easily adaptable to existing operations. Moreover, ensuring the security and reliability of these AI-integrated systems is paramount, given the high-stakes nature of offshore energy operations.

2.4.4 Summary

This section critically examines the multifaceted challenges that DL faces in the offshore energy sector. A key takeaway is addressing data scarcity and quality. Effective data collection methods and the development of robust models that can handle data quality issues are essential. Innovative strategies to maximize the utility of limited data are also crucial.

The generalizability and adaptability of DL models emerge as critical concerns. Models must be tailored to withstand offshore environments' diverse and dynamic conditions. This necessitates a deep understanding of these unique operational settings and the development of flexible models that can adapt to varying conditions.

Environmental and operational variability in the offshore sector significantly impacts

DL applications. Models must be resilient to changes in environmental factors such as visibility and weather conditions and operational factors like equipment variability and inspection protocols. Creating models that effectively navigate these complexities requires a collaborative approach, blending domain-specific insights with cutting-edge AI research.

Lastly, the integration of DL technologies into existing offshore systems presents its own set of challenges. It requires a nuanced approach that balances technical compatibility with the need for operational adjustments and sensitivity to organizational culture. Successful integration hinges on developing solutions that are not only technically sound but also align with the practical realities and workflows of the energy sector.

In conclusion, overcoming these challenges is key to unlocking the transformative potential of DL in offshore inspections and the broader energy sector. Future efforts should be directed towards creating integrative, adaptable, and robust DL solutions, paving the way for their practical and smooth adoption in this high-stakes and complex domain.

2.5 Conclusions

In conclusion, this Literature Review chapter has delved deeply into the growing areas of DL and transfer learning in industrial inspections, particularly emphasising the energy sector. We have covered a wide range of research, showcasing how these advanced technologies significantly improve the accuracy, reliability, and efficiency of identifying and categorizing defects. From ensuring the integrity of pipelines to detecting underwater corrosion, the studies reviewed demonstrate the flexibility and effectiveness of DL models, even in environments where data is limited.

By incorporating advanced methods such as the Vision Transformer and EfficientNet into our research approach, we show our commitment to utilizing the most effective technologies for complex offshore inspection tasks. This strategic choice and our investigation of the YOLO architecture and the practical advantages of transfer learning place our research at the forefront of innovation in industrial inspections.

Moreover, our thorough examination of the current challenges in the energy sector's inspection processes, particularly in offshore settings, highlights the pressing need for more efficient, reliable, and innovative methods. The complexity and critical nature of these inspection tasks call for advanced solutions that can enhance the safety and integrity of essential infrastructures and reduce the time and human resources needed.

As this chapter concludes, we are ready to apply these insights to the practical aspects

of our research. The comprehensive review we have conducted lays a solid foundation for our work. It guides us toward developing and implementing DL solutions that can significantly improve inspection practices in the energy sector. Therefore, this chapter serves as a detailed guide and a driving force for future progress in remote offshore inspections.

Chapter 3

Design

This chapter presents our DL (DL) framework, developed to automate the remote visual inspection of offshore assets. The focus is on enhancing the standard of inspections by improving their accuracy, efficiency, and consistency.

The framework is designed to streamline the traditionally labour-intensive process of visual inspections, including reviewing inspection images, identifying and classifying defects, and detailed documentation. The framework aims to reduce the manual effort involved in these tasks by automating anomaly detection and classification. This chapter will first introduce the data used in our system and then explain the methodology and the specific models employed.

3.1 Proposed Framework

The proposed DL framework is a sophisticated, API-driven process that enhances offshore visual inspection operations. The framework automates several steps of the traditional inspection process, as illustrated in Figure 3.1, which depicts the current practices of a remote visual inspection workflow.

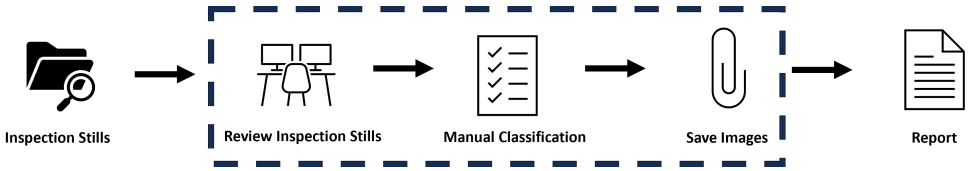


Figure 3.1: Current Practices of a Remote Visual Inspection Workflow

Upon uploading images to the API, the system initiates a pre-processing sequence to prepare the data for analysis. The images are then processed through a two-tiered model

system. Initially, a General Classification discerns images featuring circumferential welds from those without. Subsequently, an object detection model scans for anomalies within the images. This process significantly reduces the manual effort traditionally required in offshore visual inspections.

Engineers can review and refine these automated predictions to ensure accuracy. Post-validation, images are tagged and archived, streamlining the process for engineers who then incorporate only the final, vetted images into their reports. This process is further illustrated in the corresponding diagram (Figure 3.2), showcasing the seamless integration of AI with expert oversight, enhancing the efficiency and effectiveness of offshore asset inspections.

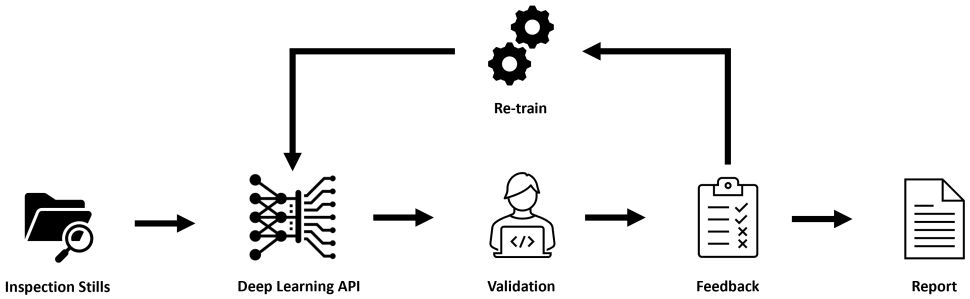


Figure 3.2: Proposed Framework

The development of our inspection framework is strategically segregated into three pivotal stages, each designed to build upon the insights and advancements of the preceding one, culminating in a robust tool for practical application by engineering professionals.

The first stage centres on the training of the General Classification. In this foundational phase, we focus on constructing and honing a binary classification system that can distinguish between images containing circumferential welds and those without. Through a rigorous process involving data collection, preprocessing, and augmentation, we train two sophisticated neural network architectures: the Visual Transformer (ViT) and EfficientNet. Upon training, these models undergo a thorough evaluation to ascertain their efficacy, with the best-performing model being selected for subsequent stages.

Transitioning to the second stage, we delve into the realm of Object Detection. Leveraging the YOLOv8 framework, we train our model to detect the presence of welds and pinpoint and classify various defect types within these images. This stage is characterized by meticulous data annotation, model training, and validation, ensuring that our object detection system operates with high precision and accuracy.

Finally, the third stage is integrating and optimising Models into an API. Here, the

trained models are encapsulated within a user-friendly application programming interface (API) specifically tailored for use by engineers in the field. This API is a conduit between the complex underlying machine learning models and the end-users, providing a seamless, interactive platform for real-time image analysis and inspection. Through iterative feedback and performance monitoring, the API is continually refined to meet the dynamic demands of practical engineering applications, ensuring it remains a state-of-the-art tool in the industry.

Each stage of the framework is designed as a step in the process and a gateway for iterative improvement, fostering a system that evolves in response to real-world data and expert feedback.

3.2 Data

Remote Operated Vehicles (ROVs) are pivotal in our offshore visual inspection, enabling access to subsea infrastructures' internal and external aspects, see Figure 3.3. These ROVs, guided by centralizers for internal and crawlers for external inspections, are meticulously manoeuvred via gravity-fed hoist chain mechanisms to adapt to the underwater installations' varying conditions and complex designs. The data acquisition from these vehicles is diverse, yielding imagery in standard definition (SD), high definition (HD), and even 4K resolutions. Our dataset is constructed from these varying qualities of imagery, ensuring a detailed and comprehensive range of visual data for our DL framework.

Figure 3.3: Inspection Vehicles for Visual Inspection

The visual data retrieved by our Remote Operated Vehicles (ROVs) is critical for identifying anomalies during offshore inspections. The imagery, which ranges from standard definition to 4K resolution, is showcased in Figure 3.4, demonstrating the varying clarity and detail captured. A higher resolution, such as 4K, significantly enhances the visibility of anomalies, providing superior-quality data. Nonetheless, environmental factors and lighting conditions remain challenges that can impact image quality, regardless of

resolution.

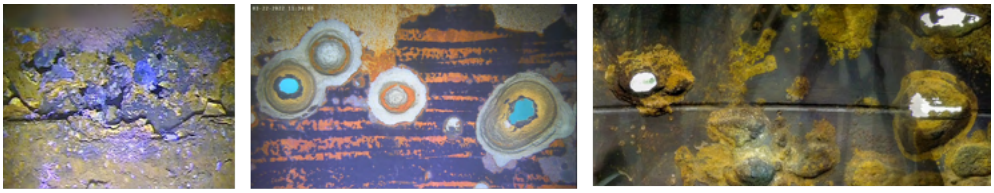


Figure 3.4: Comparison of SD, HD, and 4K Image Resolutions from Remote Inspection Vehicles

Each captured image is timestamped and appended with pertinent metadata, including the date, time, operator information, asset details, and the component tag. This metadata serves as a digital ledger of the inspection, providing valuable context for each image. The metadata can be embedded within the JPEG format, known as EXIF data, or overlaid directly onto the image. After inspections, images are stored on hard drives and transported offshore for analysis and reporting. In our dataset, the metadata section of the images is blurred for security reasons, to maintain anonymity and protect sensitive data. An example of such an image with the metadata section blurred can be seen in Figure 3.5.

Figure 3.5: Example of blurred section of inspection images

3.3 Methods & Experiments

Our methodology embraces a three-stage approach for automating the classification and detection of anomalies in offshore inspection imagery, operationalized through DL models and human expertise.

3.3.1 Stage 1 - General Classification Model

The development of the General Classification Model (GCM) begins with a rigorous data collection and pre-processing routine, ensuring a well-defined substrate for training sophisticated models such as the Visual Transformer (ViT) and EfficientNet. These

models undergo a comprehensive training process, which includes data enhancement techniques for optimal feature recognition. The evaluation stage assesses each model's performance, with the best-performing model selected for real-world application or further refinement (see Figure 3.6).

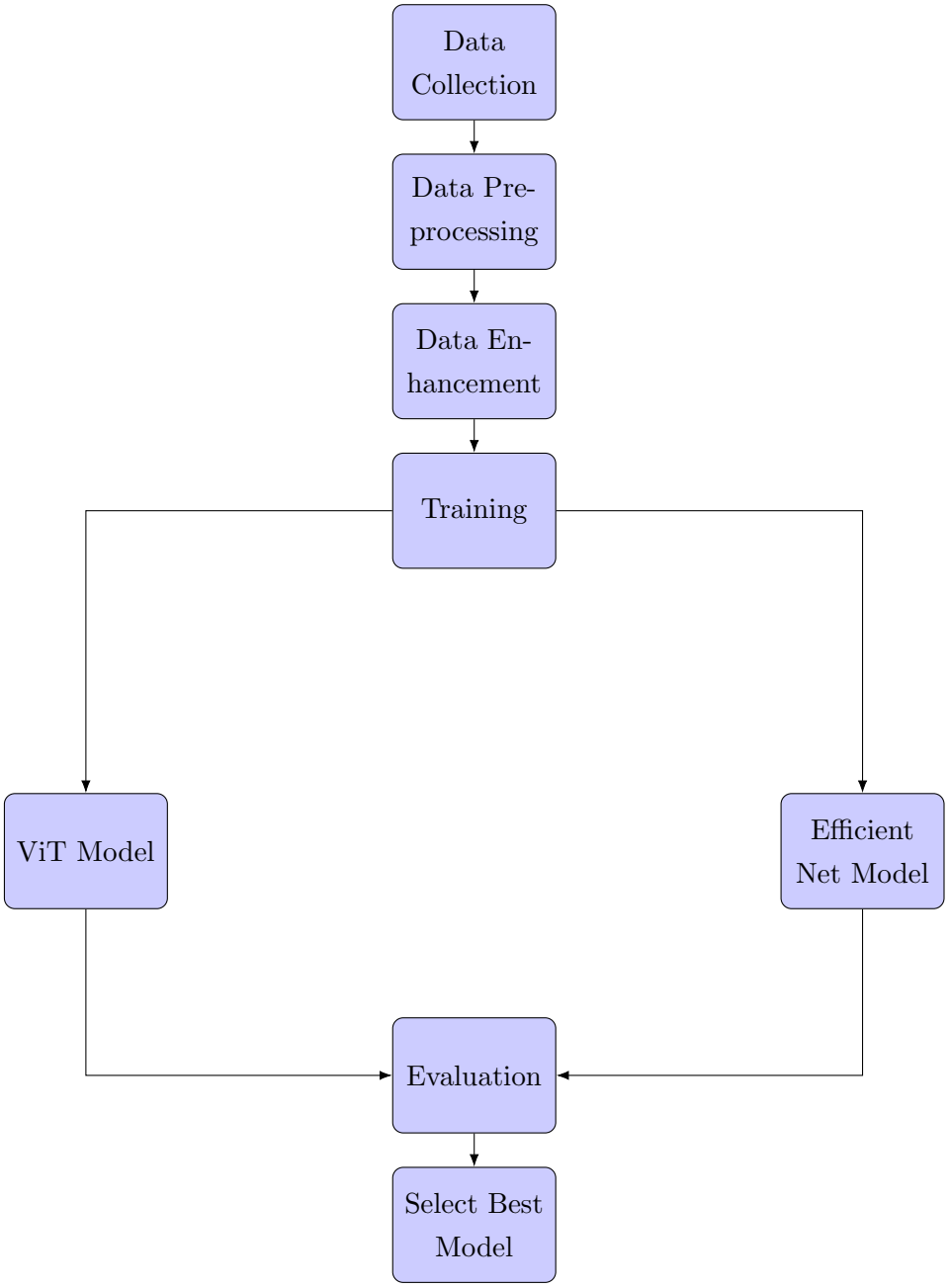


Figure 3.6: Flowchart of the General Classification Training Process

Model Architectures

The **EfficientNet B0** architecture introduces a systematic approach to scaling convolutional neural networks. At its core lies the innovative use of MBConv blocks, which implement mobile inverted bottleneck convolutions with depthwise separable convolutions. This technique significantly streamlines the network's complexity and size. With a strategic balance of network depth, width, and resolution, EfficientNet B0 effectively learns diverse features while conserving computational resources. Its culmination with global average pooling and fully connected layers equips it for various image classification tasks, achieving high accuracy with fewer parameters.

On the other hand, the**Vision Transformer (ViT)** model, particularly the ViT-Base-Patch16-224 variant, revolutionizes image classification by applying transformer architecture to images. Core features of this model include dividing an image into fixed-size patches and linearly embedding each of them, akin to words in a sentence, for natural language processing. These patches are then processed by a series of transformer encoder layers, which use self-attention mechanisms to weigh the importance of different image parts. Positional embeddings are added to retain the order of the patches. The model is topped with a Multi-Layer Perceptron (MLP) head for the final classification. This approach allows the ViT to focus on relevant parts of the image for better classification performance.

Training Strategy EfficientNet model

Model Initialization The EfficientNet B0 model, a part of TensorFlow's Keras applications, is the foundational architecture for the image classification task. The model leverages a transfer learning approach using pre-trained weights from the ImageNet dataset. This method is advantageous for harnessing the robust feature extraction capabilities inherent in models trained on extensive and diverse datasets like ImageNet.

Data Augmentation Strategy Image augmentation enhances the model's ability to generalize and perform accurately on varied data. Introducing transformations and variations in the training images makes the model more adept at handling real-world variations in the input data.

Model Configuration In the initial phase, the EfficientNet B0 model is configured with its pretrained weights frozen (`model.trainable = False`). This freezing of weights ensures that the initial learning phase focuses exclusively on the newly added layers, thus allowing these layers to learn task-specific features without altering the already learned representations. The model's top is customized with layers designed

to pool, normalize, and regularize the features, culminating in a dense layer with a sigmoid activation function, ideal for binary classification tasks.

Initial Compilation and Training The model is compiled using the Adam optimizer with a learning rate set at 1×10^{-4}, and a binary cross-entropy loss function. This initial training phase, spanning 25 epochs, allows the newly added layers to adjust their weights based on the specific characteristics of the training data. In contrast, the frozen layers preserve their pre-learned feature representations.

Model Fine-Tuning Post the initial training, the model undergoes a fine-tuning process where the top 20 layers are unfrozen (excluding BatchNorm layers), allowing these layers to adjust their weights more nuancedly. During this phase, the learning rate is reduced to 1×10^{-5}, facilitating finer adjustments and preventing overfitting. This fine-tuning is critical for tailoring the model more precisely to the specific features of the dataset in question.

This training strategy, encompassing transfer learning and fine-tuning, is designed to optimize the EfficientNet B0 model for the specific requirements of the image classification task. By leveraging pre-trained weights and fine-tuning the model to the nuances of the dataset, the strategy aims to achieve high accuracy and robustness in the classification performance.

Training Strategy ViT model

Model Initialization The Vision Transformer (ViT) model is initialized using the pre-trained ‘google/vit-base-patch16-224-in21k‘ model. This approach utilizes transfer learning, where the pre-trained model, trained on a large and diverse dataset, provides a robust starting point for feature extraction. The ViT model is known for its effectiveness in handling image data through its unique transformer architecture.

Model Configuration The ViT model in this study is adapted for image classification with a custom classifier head. The model architecture includes a dropout layer for regularization and a linear layer for classification. The number of output classes is set based on the dataset used. This customization is crucial for tailoring the pre-trained ViT model to the specific requirements of the image classification task.

Training Preparation An essential aspect of the training strategy is preparing the data and model for the training process. This includes setting up the feature extractor designed explicitly for the ViT model, configuring the optimizer, and defining the loss function. The feature extractor is critical in pre-processing the images to a format

suitable for the ViT model. The Adam optimizer is chosen for its effectiveness in handling sparse gradients, with a learning rate of 2×10^{-5}, a value selected to balance the speed and stability of the learning process. The cross-entropy Loss function, which is suitable for classification tasks with multiple classes, is used.

Training Process The training process involves iterating the dataset in batches, applying the feature extractor to each batch, and feeding the processed data into the model. The model's parameters are updated based on the computed loss. Additionally, the training process includes evaluating the model on a separate test dataset at regular intervals. This evaluation step is crucial for monitoring the model's performance and generalization capabilities.

Computational Considerations The use of GPU for training is an important consideration, given the computational intensity of the ViT model. The model is transferred to the GPU, significantly speeding up the training process. This approach highlights the computational requirements of training sophisticated models like the Vision Transformer.

The training strategy for the Vision Transformer model is designed to leverage the strengths of transfer learning while also customizing the model to fit the specific image classification task. The plan encompasses data preprocessing, model configuration, and iterative training and testing, all aimed at optimizing the model's performance for the dataset.

Evaluation Metrics

In machine learning, particularly in binary classification tasks, model performance evaluation transcends beyond mere accuracy. A comprehensive understanding of a model's effectiveness necessitates a multifaceted approach encompassing several key metrics. These metrics provide insights into various aspects of the model's predictive capabilities, each addressing specific characteristics of the model's behaviour. This section outlines the critical evaluation metrics utilized in assessing the performance of binary classifier models. These metrics include accuracy, precision, recall, and F1 score. Their collective interpretation offers a holistic view of the model's performance, balancing the trade-offs between different types of errors and capturing the nuances of prediction in a binary classification context.

Accuracy: This is one of the most intuitive performance measures. It is the ratio of correctly predicted observations to the total observations and provides a quick snapshot of the model's overall correct predictions. However, it can be misleading when class

distributions are imbalanced.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Precision: The positive predictive value is the ratio of correctly predicted positive observations to the total predicted positives. It shows the model's ability not to label as positive a sample that is negative. This metric is critical when the cost of a false positive is high.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

Recall (Sensitivity): This measures the ratio of correctly predicted positive observations to all actual positives. It shows the model's ability to find all the positive samples, making it crucial when false negatives occur unacceptably high.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3.3}$$

F1 Score: The F1 Score is the harmonic mean of precision and recall, considering false positives and false negatives. It is an excellent way to show a class has low precision and recall. This score is instrumental when you need to balance precision and recall.

$$\text{F1 Score} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3.4}$$

Negative Predictive Value: The ratio of correctly predicted negative observations to the total predicted negatives. It measures the model's ability to identify negatives and is important in scenarios where false negatives are highly risky.

$$\text{Negative Predictive Value} = \frac{TN}{TN + FN} \tag{3.5}$$

False Positive Rate: The ratio of incorrectly predicted positives to the total negatives. This is significant in contexts where false alarms are to be minimized.

$$\text{False Positive Rate} = \frac{FP}{FP + TN} \tag{3.6}$$

False Discovery Rate: This is the proportion of false positives among all positive predictions and is important when the consequences of false discoveries need to be controlled.

$$\text{False Discovery Rate} = \frac{FP}{FP + TP} \tag{3.7}$$

False Negative Rate: This indicates the proportion of incorrect positives classified as negatives. It's crucial in applications where missing out on actual positives is costly.

$$\text{False Negative Rate} = \frac{FN}{FN + TP} \tag{3.8}$$

Matthews Correlation Coefficient (MCC): The Matthews Correlation Coefficient is a metric that evaluates the performance of a binary classification model by considering both the correct and incorrect predictions. It's valued for providing a balanced assessment, which is beneficial when predicted classes have significant size differences.

$$\text{Matthews Correlation Coefficient} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.9}$$

3.3.2 Stage 2 - Anomaly Detection Model

Advancing to the Anomaly Detection Model, we implement the YOLOv8 framework to identify critical defects from the visual data. This stage includes meticulous data annotation to train the model in detecting and categorizing anomalies. The iterative training process refines the model's accuracy, culminating in a validation phase that ensures the model's efficacy and readiness for deployment in practical scenarios (refer to Figure 3.7).

YOLOv8 Anomaly Detection Model Training Flowchart

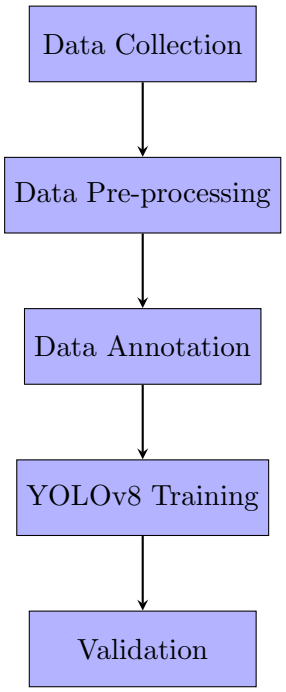


Figure 3.7: The training flowchart of the YOLOv8 anomaly detection model.

YOLOv8 Architecture

YOLOv8 differentiates itself from previous iterations by integrating advanced mechanisms like spatial attention, feature fusion, and context aggregation, contributing to its enhanced performance. These architectural innovations enable YOLOv8 to surpass its predecessors in speed and accuracy in certain applications.Gašparović et al. 2023b

The backbone of YOLOv8 is designed to capture a wide range of features from the input image. The neck uses a novel cross-stage partial network design to enhance feature propagation and reuse, which can be helpful for detecting anomalies in complex offshore images. The head of the network, which includes a series of convolutional and fully connected layers, benefits from a self-attention mechanism, allowing the model to focus on salient features that are indicative of anomalies.

For offshore inspection imagery, where anomalies could be subtle or manifest at various scales, YOLOv8's multi-scale prediction capability is particularly advantageous. It utilizes a Feature Pyramid Network (FPN) to integrate features from different levels of the network, enabling the detection of both small and large-scale anomalies. Advanced training techniques, such as adaptive learning rates and sophisticated data augmentation, ensure the model is robust and can generalize well to new, unseen offshore imagery.

Moreover, YOLOv8's adaptive use of mosaic augmentation optimizes training without compromising performance, an essential factor when dealing with the unique and varied characteristics of anomalies in such specialized imagery. These attributes make YOLOv8 particularly suited to the high precision and real-time demands of internal component inspection scenarios.

Training Strategy

The YOLOv8 model was trained to detect anomalies in offshore inspection imagery. Utilizing the 'yolov8s.pt' configuration, denoting the 'small' variant of the model optimized for speed and efficiency, the training was executed over 60 epochs with a batch size of 16, ensuring a balance between computational resource utilization and learning effectiveness.

Model Configuration The training strategy commenced with configuring the YOLOv8 'small' model (yolov8s.pt). This model variant was specifically chosen for its streamlined architecture, which is designed to provide a balance between speed and accuracy. Transfer learning was used, leveraging learned features from extensive datasets to improve training efficiency.

Training Preparation Preparation for training involved setting up the data processing pipeline as specified in 'data.yaml'. Images were resized to 640x640 pixels to ensure a uniform input size for the model while keeping the computational load manageable. The batch size was set to 16, and adaptive learning rate schedules were implemented, with a starting learning rate of 0.01. Patience was configured to 10 epochs to introduce an early stopping mechanism, preventing overfitting and promoting model generalization.

Training Process The model underwent training for 60 epochs, including real-time visualisation to closely monitor and adjust the training progress. Data augmentation was extensively used to enhance the model's robustness against the diverse anomalies encountered in offshore structures. The strategy also employed advanced techniques such as adaptive learning rates to adjust the training regime dynamically, maximizing the model's learning potential throughout the training epochs.

Validation

Object detection models are evaluated based on their ability to identify and locate objects within images accurately. For the YOLOv8 model, which is applied to detecting

anomalies, it is essential to use a set of metrics that can capture both the precision of classification and the accuracy of localization.

Precision and Recall: Precision measures the proportion of predicted positives that are true positives, essentially quantifying the model's accuracy in predicting anomalies. On the other hand, Recall quantifies the model's ability to detect all actual anomalies, reflecting the model's sensitivity to the presence of defects.

Intersection over Union (IoU): Intersection over Union (IoU) is a crucial metric for assessing the accuracy of an object's predicted location. It is determined by dividing the overlapping area between the predicted bounding box and the actual ground truth bounding box by the total area encompassed by both boxes. The performance of models like YOLOv8 is partially evaluated based on their capability to achieve high IoU scores, which signify precise detection and localization of objects.

Mean Average Precision (mAP): mAP provides an overall effectiveness measure of the model across multiple thresholds and classes. For YOLOv8, mAP is computed at different IoU levels, including mAP at IoU=0.5 (mAP@0.5) and mAP across a range from 0.50 to 0.95 (mAP@0.50-0.95). These metrics indicate the model's robustness across various overlap and detection difficulty degrees.

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a single score that balances the two. It is beneficial when seeking a model that must balance finding all anomalies (high recall) and maintaining high classification accuracy (high precision).

The YOLOv8 model's capability to detect anomalies is quantified through these metrics, offering insights into its practical deployment for automated inspection. High scores in these metrics indicate a model that can reliably assist in identifying defects, which is paramount in ensuring the integrity and safety of the inspected structures.

The performance of the YOLOv8 model in detecting anomalies was evaluated quantitatively using a test set of labelled images, which included known defects such as through-wall defects, coating loss, and pitting. The model's effectiveness was measured using previously defined metrics, focusing on precision, recall, mAP, and F1 score.

3.3.3 Stage 3 - Human in the Loop

The final stage integrates human validation into the loop, where engineers review and provide feedback on the model's predictions through an API. This feedback loop is essential for continuous improvement, enabling the retraining of models to enhance

precision and effectiveness. The platform also streamlines the data management process, significantly reducing manual classification work and expediting the inspection workflow (see Figure 3.8).

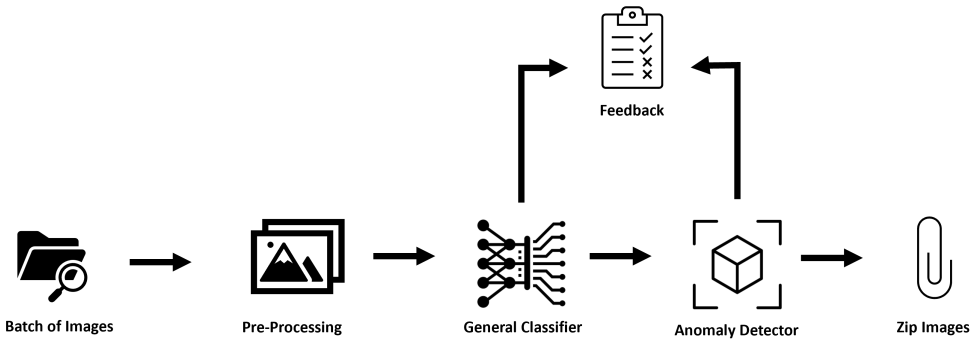


Figure 3.8: The workflow for API integration in the automated inspection system.

This structured approach, from data acquisition to model training and human integration, outlines a comprehensive system designed to augment the accuracy and efficiency of offshore visual inspections.

API Design

The API is structured into three phases, each building upon the previous to enhance functionality and user interaction.

Phase 1: Image Pre-processing The first stage involves uploading inspection images. Users can provide a link to a local directory or, by default, to a cloud storage server where the images reside. The API then downloads the image folder, validating the integrity and format of the images. These images are temporarily stored in the API's memory for efficient processing in subsequent stages.

A critical component of Stage 1 is applying Optical Character Recognition (OCR) to classify the images based on time stamps, as detailed in the "design-data" section. This classification aligns with the inspection process used by engineers, who record the time of anomaly detection. The API significantly reduces manual search efforts by organizing images by the minute they are taken.

Phase 2: DL Models In the second stage, the focus shifts to detailed image classification and anomaly detection. This stage integrates the General Classification Model (GCM) and an anomaly detection system. Images selected and labelled with elevation in Stage 1 are processed here.

Engineers can further annotate these images, explaining their selection and the nature of the content. The API supports a variety of annotations, including image features (automatically suggested by GCM for recognized objects like circumferential welds) and anomaly types. An object detection feature allows for more specific anomaly identification.

Phase 3: Human Feedback The final stage involves saving the annotated images, which can be downloaded in a zip file format. The naming convention for saved images includes the still number, elevation, feature, and anomaly type. Moreover, this stage retrieves annotations and model predictions in a CSV format. This data is crucial for analyzing the performance of our models and providing insights for future training and model improvement.

3.4 Conclusion

Our framework unfolds across three stages, each building upon the last to create a comprehensive inspection system. Initially, we focus on training DL models to distinguish between different types of weld images. We then advance to the second stage, refining our models to identify welds and accurately detect and classify various defects. The culmination of our design is the third stage, where we integrate these models into an API, crafting an interface that offers engineers a direct and simplified interaction with our advanced tools.

This tiered approach to design fosters a culture that values continuous development and leverages expert feedback to refine and enhance our system. Integrating the General Classification Model with the YOLOv8 anomaly detector represents a significant step towards a user-centred platform that evolves to meet the dynamic needs of its users.

Chapter 4

Implementation & Results

This chapter outlines developing and implementing a sophisticated two-stage classification system, pivotal in automating anomaly detection within engineering images, mainly focusing on welded structures. Initially, the chapter delves into creating a General Classification Model, employing advanced deep-learning architectures for initial defect identification. This progresses to a more focused anomaly detection model, which leverages state-of-the-art machine learning techniques to pinpoint and classify specific anomalies. The final stage introduces a Human-in-the-Loop API, integrating expert human feedback into the decision-making process and enhancing the model's accuracy and reliability in varied operational scenarios.

This chapter contributes significantly to the realization of Objectives 3 and 4. Objective 3, which revolves around developing a specialized machine-learning model for anomaly detection, is addressed through the careful construction and training of the system. Objective 4, focused on integrating human expertise, is actualized in the latter stage, where a collaborative approach is established between automated systems and human judgment. This integrated methodology not only underscores the chapter's alignment with the project's goals but also showcases a comprehensive approach to refining automated inspection systems in engineering.

4.1 Stage 1: General Classification Model

4.1.1 Data Collection

Welded components frequently contain inherent stresses, with the intensity of these stresses often equating to the material's yield strength. As detailed in Zhang's research, inherent tensile stresses adversely affect the structural soundness of engineering

structures. Y. Zhang et al. 2013a. Hence, when conducting remote visual inspections of caissons, a key focus is the evaluation of their welds. Caissons are commonly connected through circumferential welds (CWs), which join two cylindrical objects along their periphery. Due to their exposure to stresses from surface tides and ocean currents, CWs are prone to localized corrosion and fatigue, as discussed in the referenced study on welds Y. Zhang et al. 2013b (Figure 2.3).

Remotely inspecting circumferential welds (CWs) in caissons presents numerous challenges, particularly underwater environments. The images' quality is often compromised by fluctuating lighting conditions, material reflectivity, water movement, and water turbidity. These challenges increase the likelihood of errors and contribute to the time-intensive nature of the inspections. This is compounded by current manual inspection techniques being error-prone and time-consuming.

Circumferential welds, commonly called CWs, display a diversity in size, thickness, and colour, shaped by various factors. Notably, every CW is characterized by visible horizontal lines at the top and bottom, resulting from the changes that occur during the welding process. These alterations are significant, as they lead to a distinct difference in the microstructure and properties of the welded area compared to the original material. Despite the clear visibility of these horizontal lines to the human eye, identifying CWs in underwater environments is challenging. Factors such as low contrast, suspended particles in the water, and varying underwater lighting conditions contribute to the difficulty in spotting these welds in subsea sections.

A carefully selected team, composed of offshore and onshore engineers, including a senior inspection engineer with extensive experience in remote visual inspections, was formed. This team's task was to sort through an extensive collection of remote visual inspection records to ensure a diverse and representative sample of circumferential welds (CWs) from various geographical areas and caisson designs. A simple Pareto chart was created to display the distribution of these inspection jobs across different regions globally. The objective was to develop a model capable of handling a broad sample of CWs against varied backgrounds. The team's expertise was crucial in establishing clear-cut criteria for categorizing images into two distinct groups: 'cw' and 'non-cw', as indicated in the referenced figure (See Figure 4.1).

From the refined database, a collection of 9,100 images was gathered. These images, showcasing various inspection stills in multiple format sizes, were handpicked and categorized. The dataset was then divided into two distinct categories: ' cw' and 'non-cw'. The distribution of this data across the training, validation, and test sets is detailed in Table 4.1.

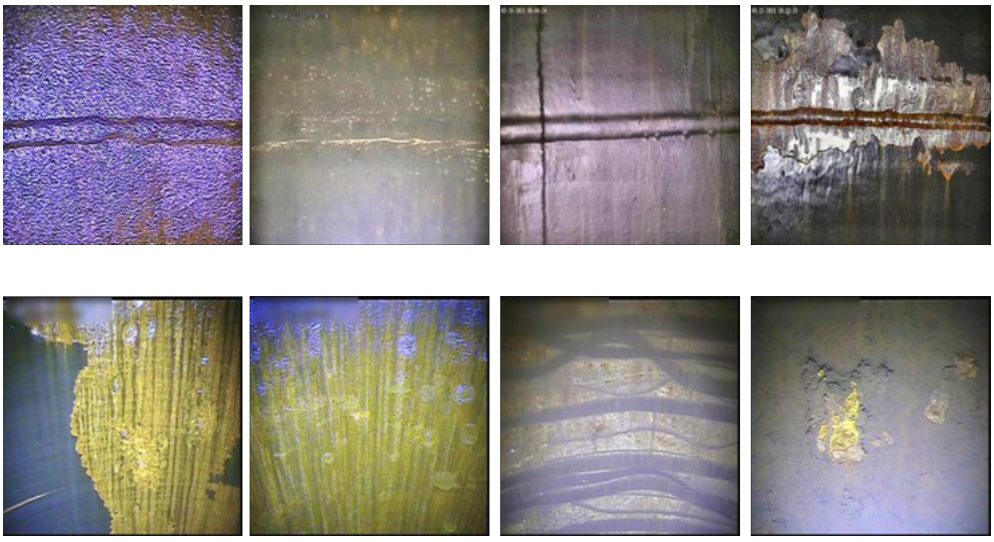


Figure 4.1: Circumferential weld (top) and non-circumferential weld (bottom)

Table 4.1: Dataset distribution

Label	**Training**	**Validation**	**Test**	**Total**
cw	3,185	910	455	4550
non-cw	3,185	910	455	4550

Note that all annotated data (stills and labels) have been checked for annotation correctness three times: one from the inspection technician who collected and reported the data, subsequently on-shore by the senior inspection engineer for the approval of the report, and finally during the manual extraction of the dataset itself by the offshore operations manager.

4.1.2 Data Pre-processing

In internal inspections, the quality of captured images is paramount, yet environmental adversities and variable lighting conditions frequently compromise it. These factors can significantly degrade the visual clarity, thus impeding accurate analysis. Prior studies, notably by Pirie et al.. Pirie and Moreno-Garcia 2021, have explored an array of filtering techniques to mitigate these issues, such as Contrast-Limited Adaptive Histogram Equalization (CLAHE), grayscale conversion, and inpainting, each designed to enhance image fidelity under challenging conditions.

Our tailored approach to image preprocessing incorporated these established methods, synergizing them into a custom filter meticulously calibrated to our specific dataset. This bespoke filter was systematically developed to accentuate the defining features of circumferential welds—particularly the top and bottom horizontal lines indicative of

the Heat-Affected Zone (HAZ)—while reducing glare and achieving a more consistent overall brightness.

The algorithm begins with converting images to grayscale, simplifying the data by reducing it to a single luminance channel, thus streamlining the subsequent enhancement process. We then apply a CLAHE filter, which adapts the image histogram in localized tiles to improve contrast without amplifying noise, a common pitfall of traditional histogram equalization.

In conjunction with CLAHE, we implement inpainting techniques to reconstruct areas of the image marred by glare or other distortions. This process employs advanced algorithms to fill these regions with plausible data inferred from the surrounding pixels, effectively 'healing' the visual data.

As shown in 4.2, our composite filter demonstrates a marked improvement in image quality, as evidenced by side-by-side comparisons with unfiltered images. The resulting enhanced images exhibit clear and discernible weld lines with minimal interference from reflections or uneven lighting, thus providing an optimal starting point for further processing by the General Classification Model and the YOLOv8 anomaly detector.

The intricate details of the custom filter algorithm and its implementation, which play a critical role in the preprocessing of inspection images, are comprehensively detailed in the annexe of this document A.1.

Furthermore, we standardized the size of all images to 224x224 pixels. This resizing is not merely for convenience but stems from the necessity of matching the input dimensions required by the architectures of our chosen models, EfficientNet B0 and the Vision Transformer (ViT). Both models are designed to work with images of this specific size, allowing them to effectively process and extract features from the data. By conforming to this dimensionality, we ensure that our models receive the data in an optimised format. This is crucial for maintaining the integrity of the feature extraction process and the subsequent accuracy of the classification and detection tasks. This uniformity in image size also facilitates more efficient batch processing during model training and inference, leading to improved computational performance.

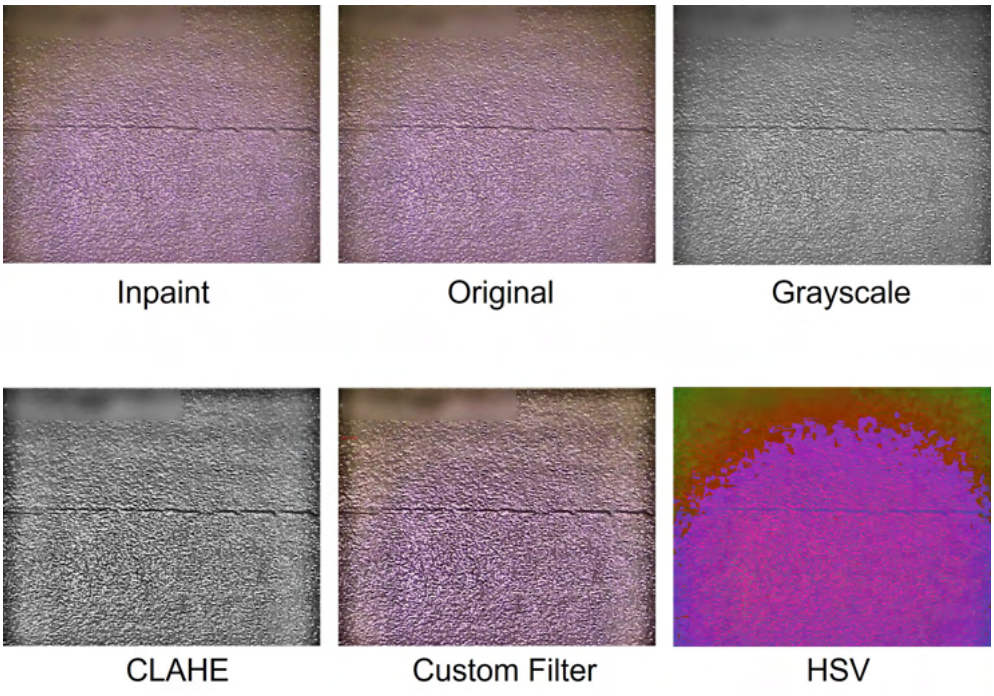


Figure 4.2: Comparison of a circumferential weld using different filters.

Having established the importance of data preprocessing and the standardization of image sizes for practical model training, we now focus on the intricacies of configuring and validating our chosen deep-learning models. The subsequent sections are dedicated to elucidating the processes involved in training the Visual Transformer (ViT) and EfficientNet models. Each model's unique architecture necessitates a tailored approach to training and validation, which is critical for achieving optimal performance in our classification tasks. We will delve into the specific hyperparameters selected, the rationale behind these choices, and the methods employed to ensure our models generalize well to new, unseen data. This level of detail is intended to provide clarity and insight into our machine-learning pipeline, illustrating how each decision is grounded in creating the most accurate and reliable classifier for the task at hand.

4.1.3 Training and Validation

Training and Validation - ViT model

The Visual Transformer (ViT) brings a novel approach to image classification by adopting the transformer architecture, renowned in natural language processing. This innovative model, particularly the 'vit-base-patch16-224' variant from Google, analyzes various segments of an image to identify the most critical features for accurate classification. The following outlines the steps undertaken to operationalize the ViT model,

specifically the ‘vit-base-patch16-224‘ version, for our classification objectives.

Environment Setup and Library Installation The initial phase of our project involved establishing a robust computational environment. Google Colab was selected for its provision of complimentary access to high-performance computing resources. We integrated our Google Drive with the Colab environment, a centralized repository for our image dataset.

After the environment setup, we installed the essential libraries necessary for our project. The Transformers library from Hugging Face, a comprehensive suite of pre-trained models and tools optimized for machine learning tasks, was particularly critical for working with the ViT model. We specifically utilized the ‘vit-base-patch16-224‘ model, which is pre-trained on a large corpus of images and fine-tuned for our specific dataset to achieve optimal performance in image classification tasks.

Model Training Procedure Upon configuring our toolkit, we commenced training on the ViT model. The images were organized into batches using data loaders, a mechanism that systematically presents groups of images to the model, akin to an assembly line in a factory.

The training process involved guiding the model through a cycle of predictions and adjustments. The model attempted to identify patterns for each batch and compare its predictions to the actual labels. Discrepancies between the two triggered a learning process where a loss function quantified the model's prediction errors. An optimizer, functioning as the navigator for the learning process, directed the model to alter its parameters incrementally, thereby improving its predictive accuracy.

To ensure the model did not simply memorize the training images, we implemented dropout regularization, randomly concealing parts of the data, compelling the model to discern more general patterns. Additionally, early stopping criteria were implemented to terminate training if the model ceased to demonstrate improvement, thus preventing overfitting.

Model Evaluation Post-training, a rigorous evaluation was conducted using a validation set composed of images previously unseen by the model. The focus was not only on the loss metrics but also on the accuracy rate—the proportion of correct predictions made by the model.

Alertness to recurrent errors was paramount during this phase; a rise in validation loss or stagnation in accuracy improvement signalled the need to recalibrate the learning process.

Inference and Performance Visualization Satisfaction with the model's validation performance led to its application to a new set of images. We employed Matplotlib to visualize the model's predictions against the actual labels, which provided tangible insights into the model's classification strengths and weaknesses.

Further, we constructed a confusion matrix, a strategic tool that illustrates the model's predictive confusion—mislabeling images of one class as another, which is critical for understanding the model's limitations and areas for refinement.

Model Saving and Loading The culmination of the training process was the preservation of the model. The entire trained model was saved to disk, encompassing its structural configuration and learned parameters. This file can be reloaded for future evaluation, further training, or deployment, ensuring the continuity of our work.

In conclusion, the training and validation journey of the ViT model has underscored its potential as a formidable instrument for image classification. Its distinctive analytical approach to images promises to enhance our ability to make precise predictions.

Training and Validation - EfficientNet model

In this section, we detail the process of training and evaluating an EfficientNet-based image classification model. The approach leverages TensorFlow and Keras, utilizing hardware acceleration through GPUs/TPUs to facilitate the training and fine-tuning of the model. The methodology encompasses pre-processing and augmentation of the dataset, followed by a structured training procedure.

Pre-processing and Data Augmentation

Data preparation is crucial for effective model training. Our dataset, mounted from Google Drive, contains images that vary in size and require standardization. We utilized TensorFlow's `image_dataset_from_directory` to load the images in batches, categorizing them into 'cw' and 'non-cw' for the binary classification task. To accommodate the EfficientNet input requirements, we resized all images to 224x224 pixels.

To enhance the robustness and generalization ability of our model, we implemented several data augmentation techniques. Data augmentation helps in artificially expanding the dataset by creating modified versions of images, thereby preventing overfitting and improving the model's performance on unseen data. The following augmentation strategies were employed:

- **Random Flipping**: This technique involves randomly flipping the images horizontally or vertically. Flipping helps the model become invariant to the orientation of the images, which is particularly useful when defects might appear in various orientations.
- **Random Contrast Adjustment**: We applied random contrast adjustments with a factor of 0.2. This helps the model handle variations in lighting conditions by simulating different levels of image contrast.
- **Random Rotation**: Images were randomly rotated by up to 20%. This ensures the model can recognize patterns and defects regardless of their image orientation.
- **Random Zoom**: We used random zooming with height and width factors 0.2. This augmentation allows the model to focus on different parts of the image, enhancing its ability to detect features at various scales.

These augmentation techniques were chosen to simulate real-world variations and enhance the model's capability to generalize well across different inspection scenarios.

Training Procedure

With the data pre-processed and augmented, we constructed the model using the EfficientNet architecture with pre-trained ImageNet weights, modifying the top layer to suit our binary classification needs. The training procedure was conducted in two phases:

First Phase: We initially trained the model with the base layers frozen to retain the learned ImageNet features, focusing the learning on the new classification layer. The model was compiled with a binary cross-entropy loss function and accuracy metrics. This phase helps to leverage the general features learned from the large ImageNet dataset and adapt them to our specific task.

Second Phase: After the initial phase, we unfroze the top 20 layers and employed a smaller learning rate to fine-tune the model. This phase allows the model to adjust its more abstract representations for our specific task while minimizing the risk of overfitting. The learning rate was reduced to ensure gradual adjustments, preventing drastic changes that could disrupt the learned features.

To avoid overfitting during training, we implemented the following measures:

- **Dropout**: We used a dropout rate of 0.2 in the top layers to randomly deactivate neurons during training, which helps in preventing the model from becoming overly reliant on specific neurons.

- **Batch Normalization**: Applied after the global average pooling layer to normalize the output and stabilize the learning process.
- **Early Stopping**: Utilized callback functions to monitor validation performance and stop training when there was no improvement for a specified number of epochs, thus preventing overfitting.
- **Checkpoints**: Saved the best model weights during training based on validation accuracy to ensure the best-performing model was retained.

We also used an adaptive learning rate schedule to further optimize the training process. Initially, we set a higher learning rate to speed up convergence. In the second phase, we lowered the learning rate to fine-tune the model more carefully. This approach allows the model to make significant updates during the early stages of training and more precise adjustments later on.

The complete training process was designed to systematically adapt the EfficientNet model to our dataset, carefully balancing the retention of pre-trained knowledge with the acquisition of new, task-specific information.

4.2 Stage 2: Anomaly Detection Model

The development of an accurate image classification model hinges on the availability of a well-curated dataset. This section outlines the meticulous data collection and annotation process to compile a comprehensive set of images for training our object detection model.

4.2.1 Data Collection

The dataset comprised 4,038 images sourced from the blob storage backend of our application. These images were integral to the study as they captured a range of defects commonly encountered in our domain, such as coating loss, pitting, and through-wall defects (holes), alongside images that depicted no defects, serving as a contrast for the model to discern.

The data collection process involved offshore and onshore engineers working collaboratively to ensure the quality and relevance of the collected images. Offshore engineers were responsible for capturing images during inspections using high-resolution cameras and specialized inspection equipment. These engineers documented the conditions and identified potential defect areas, ensuring comprehensive coverage of the inspection sites. The captured images were then uploaded to the blob storage backend for further processing.

Onshore engineers played a critical role in curating and refining the dataset. They reviewed the uploaded images to ensure they met the required quality standards and represented the various defect types. Any images that did not meet these criteria were discarded. This collaborative approach ensured that the dataset was extensive and relevant, providing a solid foundation for model training.

The collected images were then divided into three distinct sets to facilitate a robust training regime and subsequent evaluation:

- **Training set**: Consisting of 2,826 images, this dataset formed the foundation for the model to learn the intricate features and variations of defects.
- **Validation set**: Containing 807 images, it was used to fine-tune the model parameters and to prevent overfitting during the training phase.
- **Test set**: Comprising 403 images, this set was pivotal for assessing the model's final performance, reflecting its ability to generalize and detect defects in unseen data.

4.2.2 Data Annotation

For labelling the images, we employed the Computer Vision Annotation Tool (CVAT), an open-source platform conducive to annotating digital images. CVAT is revered for its support in various machine learning tasks such as object detection, image classification, and segmentation, thereby serving as an ideal tool for our annotation needs.

The initial annotation was performed by a mechanical engineer with expertise in identifying and categorizing the types of defects present in the images. Each image was scrutinized, and labels were assigned to areas showcasing defects or lack thereof. Subsequently, a senior inspection engineer reviewed each labelled image to ensure the precision of the annotations. This two-tier annotation process not only reinforced the accuracy of the labels but also enriched the dataset with expert knowledge, an invaluable asset for training a reliable object detection model.

- **Annotation Process**: Each image underwent a systematic labelling process where defects were marked and categorized based on their characteristics. Images without defects were also labelled accordingly to provide the model with examples of both positive and negative instances.
- **Quality Assurance**: The annotations were subject to rigorous quality checks conducted by the senior inspection engineer, ensuring that the labels were consistent, accurate, and reflected the defects' true nature.

With the high-precision annotations complete, the dataset was well-prepared to serve as the foundation for training our model. This annotated dataset became the cornerstone of our training process, enabling the YOLOv8 model to effectively learn and differentiate between various defect types and undamaged surfaces. The rich and accurately labelled set of images provided the essential data needed for the model to develop a robust understanding of the inspection scenarios.

In summary, the data collection and annotation phases were critical in establishing a solid foundation for developing an object detection model that could meet our application's stringent requirements. The engineers' expertise in the labelling process played a significant role in ensuring the dataset's quality, which was pivotal in successfully training the YOLOv8 model.

4.2.3 Training and Validation

Implementing the YOLOv8 model for image classification entailed a series of systematic steps designed to leverage the capabilities of the Visual Transformer architecture. This process began with configuring our computational environment, which was instrumental in the seamless execution of the model training.

Environment Setup The foundation of our model training was laid by establishing a robust computational environment using Google Colab. This cloud-based platform provided us with the necessary computational resources, such as GPUs and TPUs, which are essential for the processing demands of DL models like YOLOv8.

Once the computational resources were allocated, we mounted our Google Drive to the environment. This step was akin to connecting an external drive to a computer, providing direct access to our dataset stored in the cloud.

Repository Setup Subsequently, we prepared the necessary directories and cloned the official YOLOv8 repository from Ultralytics. This repository contained the latest updates and algorithms required for the YOLOv8 model, ensuring we had access to cutting-edge object detection techniques.

Despite the pre-existence of the 'ultralytics' directory, indicative of a previous clone operation, this step was crucial to verify that the necessary files were in place for model training.

Custom Model Training The training of our custom YOLOv8 model was initiated after setting up the environment and ensuring the availability of the GPU resources. Before the commencement of training, we verified the GPU status using NVIDIA's

System Management Interface, which confirmed the availability and readiness of the GPU for training. Following this, we installed the required YOLOv8 packages and their dependencies. This included packages like 'ultralytics' and 'thop', which were necessary for the training process. This balance was especially crucial given the Tesla T4 GPU with 15102MiB memory, ensuring that the hardware was utilized efficiently. The table 4.2 shows an overview of the parameters from the yolov8 model was configured.

Training Execution The model was trained for 60 epochs, a sufficient duration for the model to converge and learn the necessary features from the data. A batch size of 16 was chosen. This size is large enough to provide a representative sample of the data and small enough to manage memory usage effectively. Each input image was resized to 640 pixels, a suitable compromise between retaining enough detail for accurate object detection and maintaining a manageable computational load. An initial learning rate of 0.01, coupled with a learning rate factor (lrf) of the same value, was set to enable a gradual and effective learning process. The model's momentum was set at 0.937, smoothing out model weights and biases updates.

Parameter	**Value**
Model Version	Ultralytics YOLOv8.0.154
GPU	Tesla T4, 15102MiB
Epochs	60
Patience	10
Batch Size	16
Image Size	640
Workers	8
Optimizer	auto
Close Mosaic	10
AMP	True
Overlap Mask	True
Mask Ratio	4
Validation	True
Split	val
IOU Threshold	0.7
Max Detections	300
Learning Rate (lr0)	0.01
Learning Rate Factor (lrf)	0.01
Momentum	0.937
Weight Decay	0.0005
Warmup Epochs	3.0
Warmup Momentum	0.8
Warmup Bias Learning Rate	0.1
Flip Left/Right	0.5
Mosaic	1.0

Table 4.2: Ultralytics YOLOv8 Model Training Configuration Parameters

A weight decay of 0.0005 was also applied to regularize the model and prevent overfitting. Warm-up epochs were set to 3.0 with a warm-up momentum of 0.8 and a warm-up bias learning rate of 0.1, which helped stabilize the learning process at the beginning of the training. Applying Mosaic augmentation in the final 10 epochs indicates a strategy where the model, after learning the basic features and patterns from the training data in its less augmented form, is further challenged with more complex and varied data representations towards the end of its training.

An automatic choice of optimizer by the model allowed for selecting the most effective

algorithm based on the training data and model architecture. Features like Automatic Mixed Precision (AMP) and an overlap mask with a ratio of 4 were employed to enhance training efficiency and effectiveness. Data augmentation techniques, such as flip left/right with a probability of 0.5 and mosaic augmentation, were used to improve the model's generalization capabilities.

Early Stopping and Model Optimization During the training, an early stopping mechanism was employed to monitor the model's performance. Training was halted if no improvement was observed in the validation metrics over a predetermined number of epochs. This strategy was pivotal in preventing overfitting and ensuring the model's generalizability. A 'patience' parameter of 10 epochs was set for early stopping. This means the training process will be halted if the model's performance on the validation dataset does not improve for ten consecutive epochs.

Model Validation and Inference Once the early stopping criterion was met, the model state at the point of highest validation accuracy was saved. This model is considered the most optimized version for deployment or further testing. Upon the completion of training, the model underwent a validation phase where it was tested against a set of unseen images to evaluate its detection performance. Validation was conducted on a separate dataset (as indicated by split=val) to ensure the model's performance was assessed consistently against unseen data. The model's predictions were then visualized and saved, providing qualitative and quantitative insights into its capabilities.

4.3 Stage 3: Human in the Loop

The implementation of the first stage, which involves uploading inspection images, is encapsulated within the API's home page interface.

The API home page, as depicted in Figure 4.3, serves as the primary interface for users, specifically engineers, to interact with the system. Central to this page is a prominent video tutorial to guide users through the API's functionalities. This tutorial is an integral part of the user experience, providing immediate assistance and instruction on using the API for image uploading and processing efficiently.

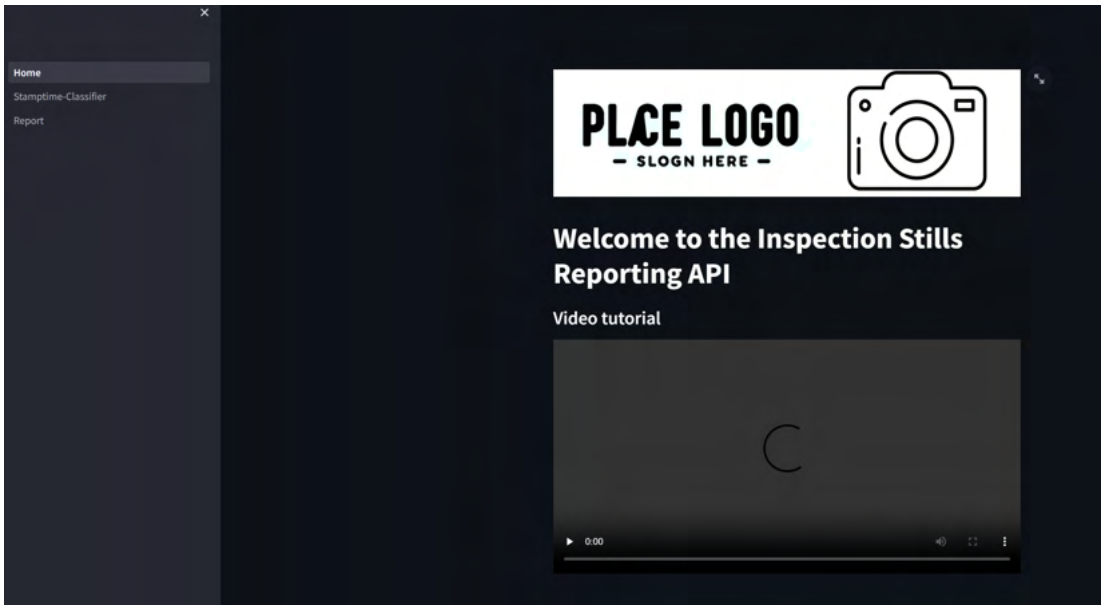


Figure 4.3: API Home Page

In the top left corner of the home page, users can find three distinct tabs: 'Home', 'Stamp Time Classifier', and 'Report'. These tabs represent the core components of the API's functionality.

This interface design ensures a user-friendly experience, allowing engineers to easily navigate through image processing and report generation stages, from the initial upload to zipping the selected images they will use for the final report.

4.3.1 Image Pre-processing

The first phase of the API's operation involves the engineer uploading images. This process is streamlined to ensure efficiency and ease of use. As illustrated in Figure **??**, the engineer is provided with a simple input field on the bar's left side. This field is specifically designated for the engineer to input the URL of the cloud storage server where the batch of images is located.

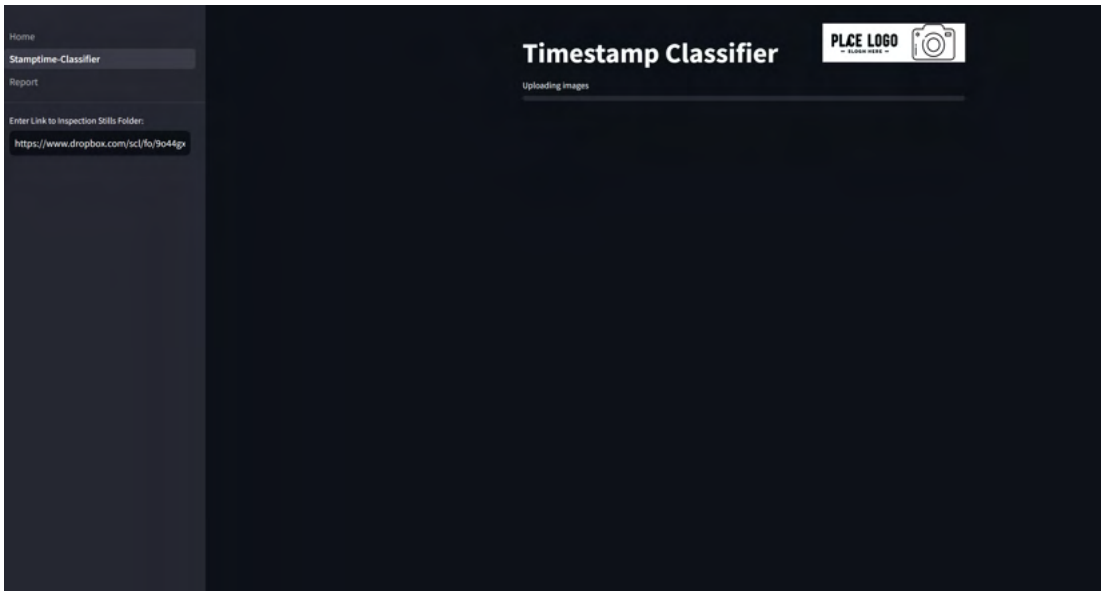


Figure 4.4: Image Processing

Upon entering the URL, the API's backend is triggered to download the content. This initial step is crucial as it ensures that all images are efficiently transferred and stored in the API's memory. This storage in the API's memory is an essential aspect of the design, enabling swift and practical application of the subsequent processing stage.

By focusing on this straightforward method of uploading images, the API significantly simplifies the initial phase of the inspection process. It allows engineers to initiate the analysis workflow quickly, setting the stage for more advanced processing and classification in the following stages.

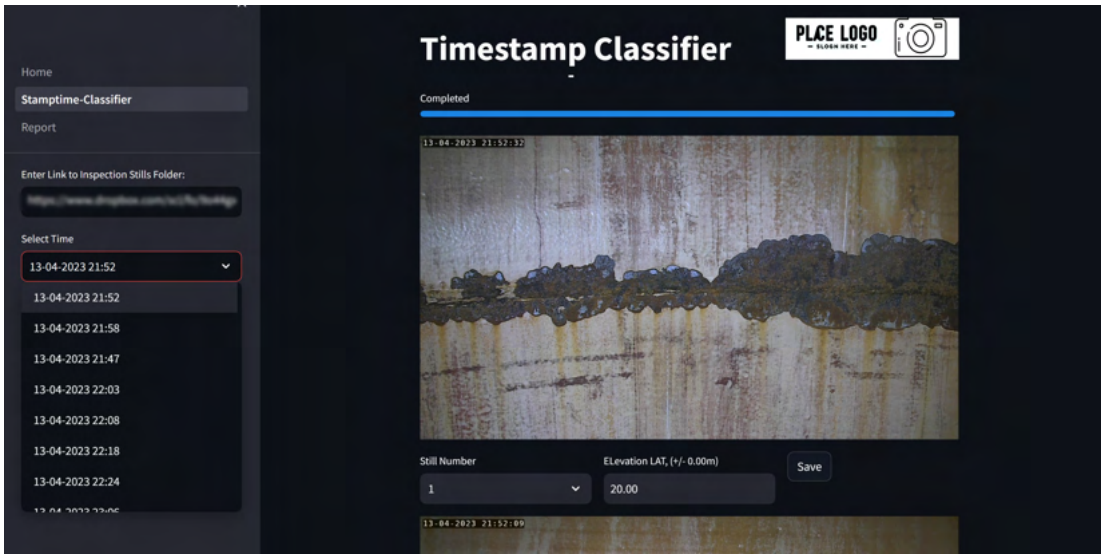


Figure 4.5: Selecting Inspection Stills

Figure 4.5 illustrates the interface engineers use for selecting and classifying inspection stills after they have been successfully uploaded and time-stamped. This stage is critical for organizing and preparing the images for further detailed analysis.

Once the images are uploaded and classified by their timestamp, the engineer is presented with a user-friendly interface that allows for efficient browsing and selection of the stills. The key feature of this interface is the ability to visualize all the uploaded images in a scrollable format. This design ensures that engineers can quickly and easily review the entire set of images without navigating through multiple pages or complex menus.

Additionally, on the left-hand side of the interface, there is a dropdown menu that enables the engineer to select images based on the exact minute they were taken. This functionality significantly speeds up the classification task by allowing the engineer to filter and view images from specific time intervals, thereby streamlining the selection process.

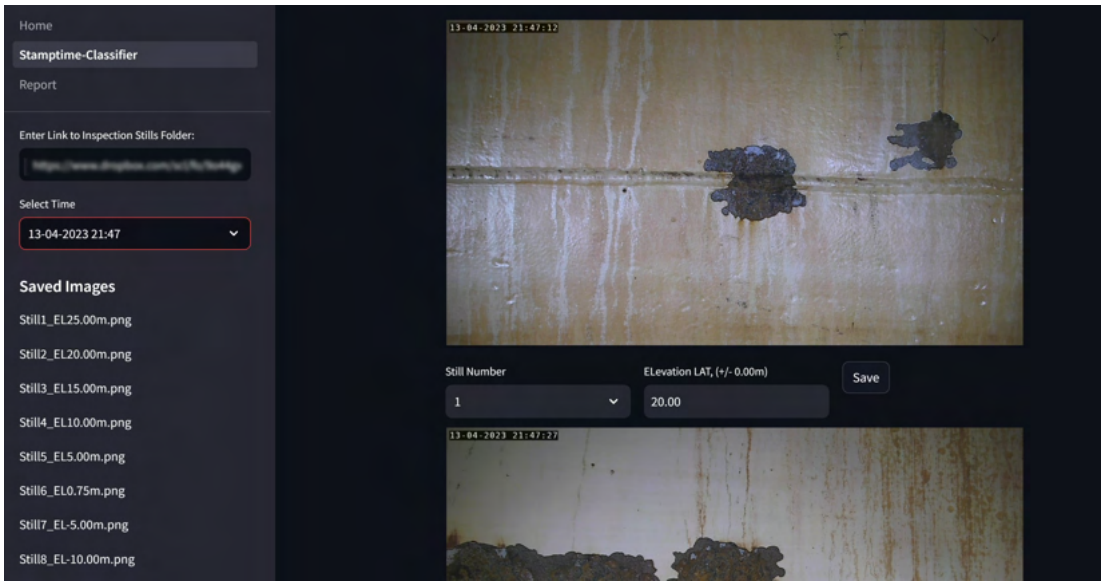


Figure 4.6: Saved Images

Once the relevant images are identified, the engineer can select them and assign two critical pieces of information: the still number and the elevation at which each image was taken during the offshore visual inspection. This labelling format, as shown in Figure 4.6, is essential for subsequent stages of the analysis, as it provides context and categorization for each image, facilitating more accurate and efficient processing and reporting.

4.3.2 DL Models

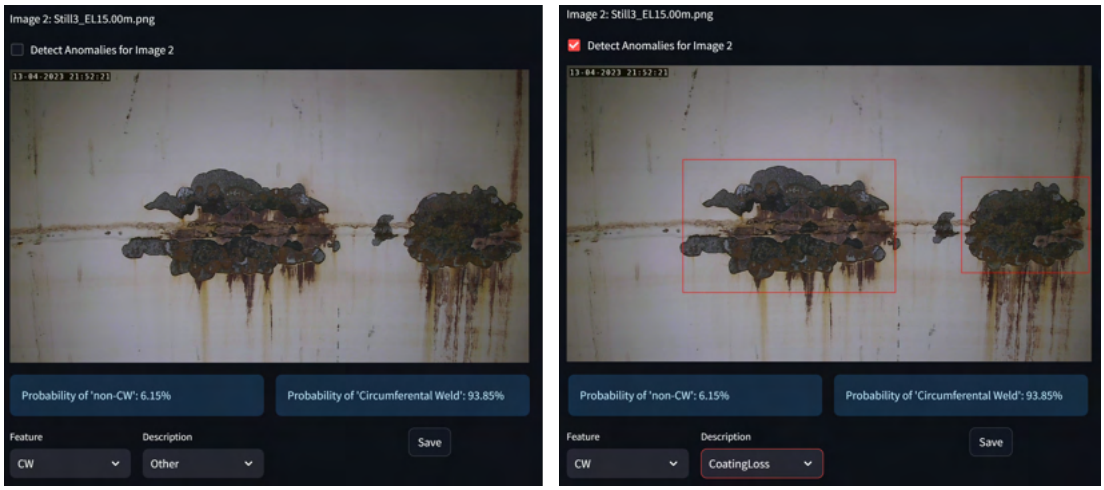


Figure 4.7: Enable Anomaly Detection Model

Figure 4.7 demonstrates the interface where the General Classification Model (GCM) is applied to the images selected in the previous phase. This stage is pivotal in automating part of the classification process by utilizing the capabilities of the GCM.

When the GCM detects a circumferential weld (CW) in an image, it automatically selects the 'CW' option from a dropdown menu. This feature is designed to streamline the classification task by pre-populating the tags based on the model's predictions, thus reducing manual input requirements.

As shown in the figure, the model selects the appropriate tag and displays the probability of the inspection still containing a circumferential weld. This probability metric is displayed below the image, offering immediate feedback on the model's confidence in its classification. This information is crucial for engineers as it provides a quantifiable measure of the model's assessment, aiding decision-making.

Additionally, the interface allows the user to activate the anomaly detection model. This feature allows for a more detailed examination of the stills, where anomalies can be visualized and further analyzed. The user interface, as depicted in the figure, is intuitively designed to facilitate easy activation and interaction with the anomaly detection model.

Integrating the GCM and the anomaly detection model in this interface significantly enhances the efficiency and accuracy of the image classification process. It empowers the engineers to identify key features and anomalies in the inspection images quickly, optimizing the workflow for generating detailed and informative reports.

4.3.3 Human Feedback

This figure 4.8 presents the user interface of an engineering image annotation tool. On the left-hand side of the application, the interface displays a list of images annotated by the engineers. Each image in this list is accompanied by its respective label, which the engineer has assigned during the annotation process.

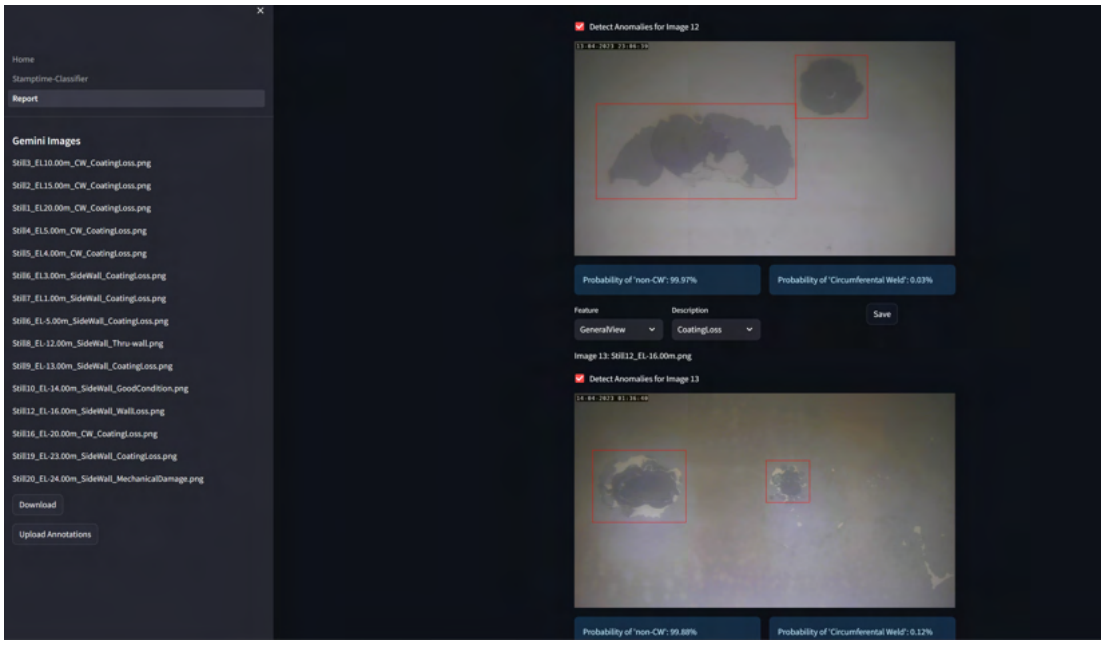
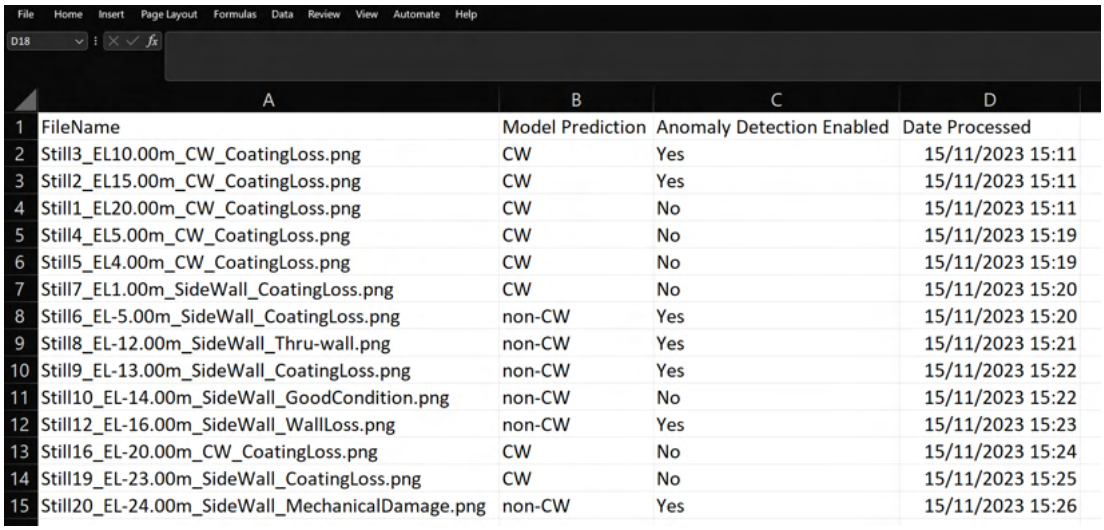


Figure 4.8: The interface showcases the 'Save Images' and 'Upload Annotations' functionalities.

There are two primary buttons available in this interface. The first button, labelled "Save images," serves a key function. Upon clicking this button, the application compiles all the annotated images into a zip file. This zip file is then made available for download, allowing the engineer to conveniently download the entire folder of annotated images. This feature is handy for engineers, allowing them to quickly transfer these images to their company's data management platform for further use or storage.

The second button, labelled "Upload annotations," offers another crucial functionality. When clicked, this button facilitates the export of the image annotations into a CSV (Comma-Separated Values) file format. The naming convention for this file is based on a data timestamp, ensuring that each annotation export is uniquely identified and can be easily traced back to its respective annotation session.



	A	B	C	D
1	FileName	Model Prediction	Anomaly Detection Enabled	Date Processed
2	Still3_EL10.00m_CW_CoatingLoss.png	CW	Yes	15/11/2023 15:11
3	Still2_EL15.00m_CW_CoatingLoss.png	CW	Yes	15/11/2023 15:11
4	Still1_EL20.00m_CW_CoatingLoss.png	CW	No	15/11/2023 15:11
5	Still4_EL5.00m_CW_CoatingLoss.png	CW	No	15/11/2023 15:19
6	Still5_EL4.00m_CW_CoatingLoss.png	CW	No	15/11/2023 15:19
7	Still7_EL1.00m_SideWall_CoatingLoss.png	CW	No	15/11/2023 15:20
8	Still6_EL-5.00m_SideWall_CoatingLoss.png	non-CW	Yes	15/11/2023 15:20
9	Still8_EL-12.00m_SideWall_Thru-wall.png	non-CW	Yes	15/11/2023 15:21
10	Still9_EL-13.00m_SideWall_CoatingLoss.png	non-CW	Yes	15/11/2023 15:22
11	Still10_EL-14.00m_SideWall_GoodCondition.png	non-CW	No	15/11/2023 15:22
12	Still12_EL-16.00m_SideWall_WallLoss.png	non-CW	Yes	15/11/2023 15:23
13	Still16_EL-20.00m_CW_CoatingLoss.png	CW	No	15/11/2023 15:24
14	Still19_EL-23.00m_SideWall_CoatingLoss.png	CW	No	15/11/2023 15:25
15	Still20_EL-24.00m_SideWall_MechanicalDamage.png	non-CW	Yes	15/11/2023 15:26

Figure 4.9: Annotated Image Data CSV File.

This figure illustrates a CSV file displayed in a spreadsheet application, representing the annotations made by engineers on various images. The CSV file is organized into four columns:

- **FileName**: Lists the names of image files, following a naming convention that includes an identifier (e.g., "Still3"), elevation information (e.g., "EL 10.00m"), and a description of the image content (e.g., "CW_Coating Loss").
- **Model Prediction**: Contains predictions made by a model, with entries such as "CW" (possibly indicating a specific condition or category) and "non-CW" (indicating a different condition or category).
- **Anomaly Detection Enabled**: Indicates whether anomaly detection was enabled for each image, with entries being "Yes" or "No".
- **Date Processed**: Shows the timestamp for each image's processing in the format "day/month/year hour: minute".

The file records the work completed, including automated model predictions and the status of anomaly detection for each image. This CSV can be downloaded for local use or uploaded to cloud storage services like Azure for further analysis, sharing, or archiving as part of the company's data management practices.

4.4 Conclusion

The "Implementation" chapter has systematically articulated the multifaceted approach to address the challenge of classifying and detecting anomalies within the visual inspection of offshore components, specifically circumferential welds (CWs) in caissons. This chapter has elucidated the rigorous process of data collection, which harnessed a significant volume of remote visual inspection data, ensuring a dataset that encapsulates the diverse manifestations of weld defects and non-defect scenarios. The meticulous data annotation, led by domain experts, imbued the dataset with high precision, providing a solid foundation for the subsequent training of advanced DL models.

The General Classification Model (GCM) training was underpinned by state-of-the-art architectures such as the Visual Transformer (ViT) and EfficientNet, selected for their proven efficacy in image classification tasks. This chapter detailed the comprehensive training and validation methodologies, highlighting the significance of environmental setup, data pre-processing, and augmentation techniques that collectively contributed to the robustness of the models.

The implementation of dropout regularization, early stopping, and other strategic measures underscored our commitment to developing models that perform with high accuracy and exhibit the capacity to generalize well beyond the training data. The chapter has also touched upon the importance of model evaluation, setting the stage for an in-depth analysis in the subsequent "Evaluation" chapter.

The "Evaluation" chapter will build on the foundation to critically assess the implemented models' performance and present a cohesive understanding of their capabilities and potential for practical deployment.

Chapter 5

Evaluation

This chapter comprehensively evaluates the DL models developed for offshore inspection tasks. The focus here is to critically assess the performance of our General Classification Model and the Anomaly Detection Model, both of which are pivotal in the automated analysis of inspection images. This assessment is not merely a quantitative exercise in metrics but a deeper exploration into how effectively these models meet the practical requirements of the offshore inspection domain.

5.1 Stage 1: General Classification Model

This section delves into the meticulous evaluation of our General Classification Model, specifically focusing on the Vision Transformer (ViT) and EfficientNet models. This evaluation is critical in determining the efficacy of these models in accurately classifying images relevant to offshore inspections, specifically in distinguishing between 'cw' (circumferential weld) and 'non-cw' classes. Our approach encompasses a detailed analysis of various performance metrics across training epochs, emphasizing the progression in model performance and its ability to generalize from the training data. We also explore the models' responses to different surface conditions through test images, assessing their real-world applicability and robustness. This comprehensive evaluation aims not only to gauge the accuracy and reliability of the models but also to uncover potential issues, such as overfitting, thereby ensuring that the models are well-suited for practical deployment in the demanding field of offshore structural assessments.

5.1.1 ViT Model

In the model "vit-base-patch-16-224" evaluation chapter, a comprehensive analysis of the validation results reveals a significant progression in model performance over the

training epochs. The detected classes are binary, categorized as 'cw' and 'non-cw', indicating a binary classification task. Figure 5.1 illustrates the model's training and validation metrics. On the left, the training loss decreased steadily, indicative of the model's improving ability to generalize from the training data. On the right, the validation accuracy achieved high scores early in the training process. This combined visualization underscores the model's potential efficacy, although further testing is warranted to confirm its generalizability and robustness.

Figure 5.1: Left: Training Loss of the ViT model; Right: Validation accuracy of the ViT model

Initially, at epoch 0, the model exhibited a substantial variation in training loss, starting at a higher loss of 0.7300 and dramatically decreasing to as low as 0.0116 by the end. Despite the fluctuating loss within this initial phase, the validation accuracy swiftly reached a perfect score of 1.00 after a few iterations, except for two instances where it slightly dipped to 0.90. This rapid achievement of high validation accuracy suggests an overly optimistic scenario, possibly due to overfitting or a simple validation task.

As the training progressed to epochs 1 and 2, the training loss consistently decreased, indicating an improvement in the model's ability to generalize from the training data. This is corroborated by a stable validation accuracy, which consistently remained at 1.00 or near-perfect scores, except for occasional decreases to 0.90. The persistence of high validation accuracy throughout these epochs suggests that the model has effectively learned the distinguishing features of the 'cw' and 'non-cw' classes.

However, the initial fluctuations in training loss and the quick attainment of high validation accuracy raise questions about the complexity of the validation set and the model's capacity to generalize to unseen data. It would be advisable to evaluate the model further on a more challenging and diverse test set to assess its real-world applicability.

Additionally, the learning rate, batch size, and other hyperparameters should be fine-tuned to ensure that the model is not overfitting and is indeed capturing the underlying patterns in the data. Regularization techniques and cross-validation could also be

employed to enhance the robustness of the model's performance metrics.

In conclusion, while the "vit-base-patch-16-224" model displays promising validation accuracy early in training, careful consideration must be given to potential overfitting, and further validation is recommended to ensure the model's efficacy in practical scenarios.

In our analysis of the "vit-base-patch-16-224" model's performance, a set of images was subjected to the model to assess its predictive capabilities in a real-world scenario. As shown in Figure 5.2, the images represent various surface conditions our model is tasked to classify. These may include multiple states of wear, corrosion, or other material conditions critical for maintenance and safety assessments in industrial settings.

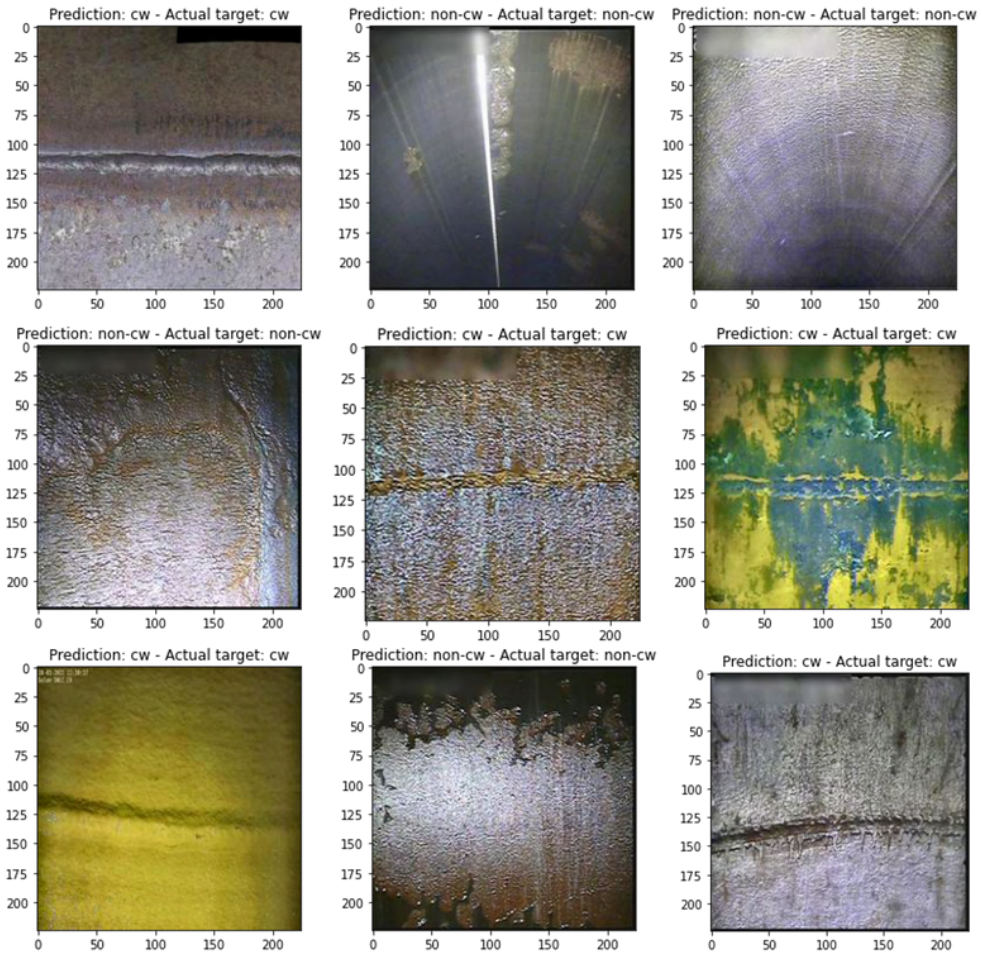


Figure 5.2: Sample test set prediction of ViT

The model effectively identifies vital characteristics of circumferential welds ('cw'). For example, it accurately classifies an image with significant rust in the top-left as 'cw',

differentiating structural joinery from material degradation. Despite reflective surfaces suggesting smoothness, it correctly identifies the second and third images in the top row as 'non-cw', demonstrating its ability to discern despite camera angle variations.

In the middle row, the rightmost image poses a challenge with its complex textures of corrosion and potential weld seams, yet the model skillfully detects rust streaks and classifies them as 'cw'. The central and leftmost images, depicting wall loss patterns, are also accurately classified.

The bottom row showcases the model's precision; it identifies a faint yellowish linear indentation in the leftmost image as 'cw'. Under reflective-light conditions, the centre and rightmost images, with more apparent 'cw' indicators, are correctly classified.

Overall, the model's high accuracy in distinguishing 'cw' from 'non-cw' is evident. However, its handling of ambiguous features highlights the need for refinement. It could enhance its performance by incorporating a wider variety of training data to cover the entire weld appearance spectrum. Additionally, implementing a confidence scoring system would provide greater insight into its predictive certainty, aiding in interpreting classifications across various scenarios.

5.1.2 EfficientNet

The learning dynamics of the EfficientNet B0 model are depicted in Figure 5.3. The left graph within the figure presents the training loss, showing a steady decrease as the model learns from the data. The right graph displays the validation accuracy, which increases correspondingly, indicating the model's improving predictive accuracy.

Figure 5.3: Left: Training Loss of the EfficientNetB0 model; Right: Validation accuracy of the EfficientNetB0 model

The "Start Fine Tuning" marker indicates a strategic shift in the training approach. From this point, the model maintains a relatively stable validation accuracy, plateauing slightly above the training accuracy. This period of fine-tuning is critical as it allows the model to refine its parameters on a more granular level, potentially improving its performance on the validation set.

Moving to the loss graph, training and validation loss decrease steadily, which is expected during the initial phase of model training. The decline in loss indicates that the model is learning to minimize the error in its predictions. Like the accuracy graph, the validation loss mirrors the training loss closely until the fine-tuning phase begins. After the "Start Fine Tuning" marker, the losses converge, and the model achieves a low and stable loss, which is an ideal outcome of fine-tuning.

The convergence of training and validation loss at a low level and high training and validation accuracy that does not diverge significantly suggest that the model is not memorizing the training data. Instead, it is learning generalizable patterns, which is the hallmark of a well-trained neural network.

However, a detailed analysis would also consider the complexity of the dataset, the diversity of the validation set, and whether the validation metrics continue to hold when the model is exposed to an utterly unseen test set. Moreover, if the validation accuracy plateaus or starts to decrease. In contrast, the training accuracy continues to increase; this might indicate overfitting, which should be addressed with techniques such as early stopping, regularization, or further data augmentation.

The EfficientNet B0 model exhibits promising training behaviour with practical learning and fine-tuning phases. The close tracking of training and validation metrics suggests good generalization. However, the ultimate test of the model's performance will be its ability to maintain this accuracy on an independent test dataset.

The classification efficacy of the EfficientNet B0 model is exemplified in Figure 5.4, where the model discerns images containing a circumferential weld ('cw') from those without ('non-cw'). The photos are annotated with both the predicted and actual labels, enabling a direct assessment of the model's accuracy.

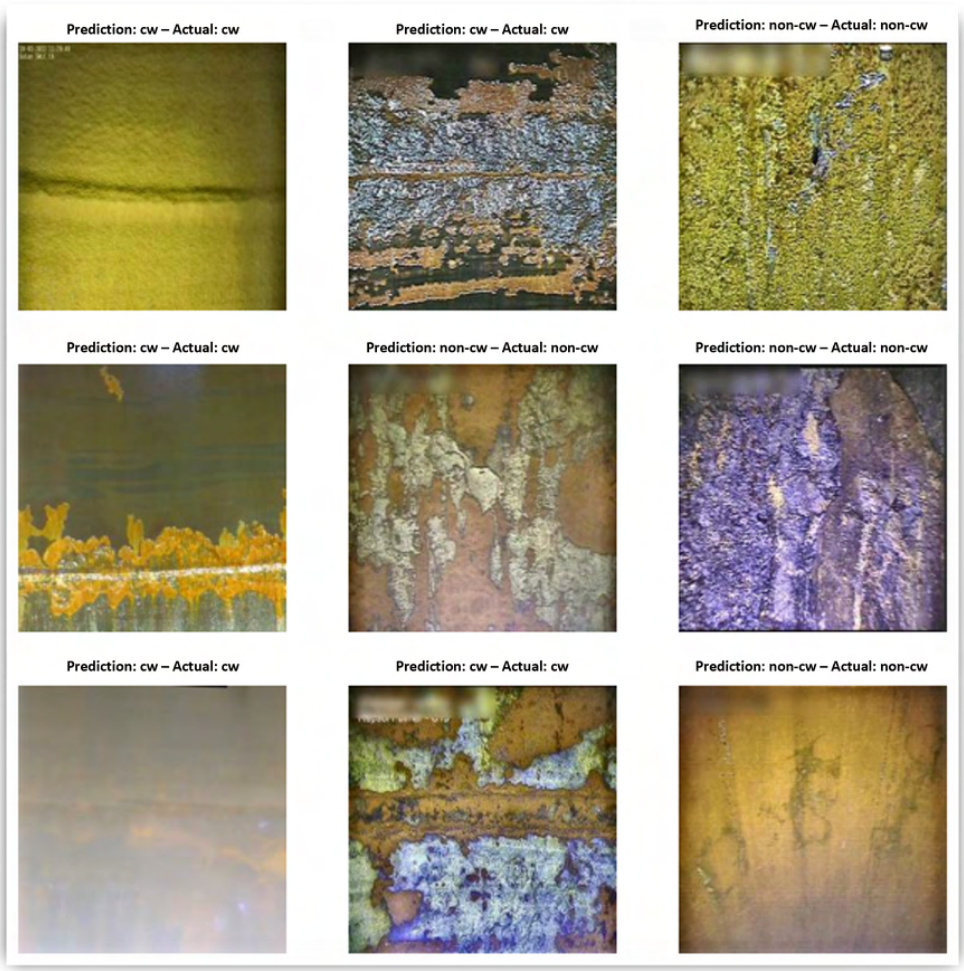


Figure 5.4: Sample test set prediction of EfficientNetB0

The top row of Figure 5.4 showcases the model's precision in identifying 'cw' conditions, even with appearance and texture variances that could confound a less robust classifier. The accurate prediction of a 'non-cw' situation in the third image underscores the model's nuanced understanding of the defining characteristics of circumferential welds.

Continuing to the second row, the model sustains its high-performance level, correctly labelling 'cw' and 'non-cw' conditions. This row further demonstrates the model's generalization capabilities, crucial for practical applications where lighting conditions and surface textures vary significantly.

The third row reinforces the model's consistent performance with correct 'cw' identifications despite the subtler visual cues present in these particular images. The model's correct 'non-cw' classification in the last image highlights its reliability and suggests a high level of sophistication in feature extraction.

Collectively, the predictions displayed in Figure 5.4 illustrate the EfficientNet B0 model's potential as a tool for automated identification of circumferential welds. Its demonstrated ability to distinguish between 'cw' and 'non-cw' conditions with high accuracy is promising. However, further testing on an extensive and varied dataset is recommended to confirm these findings and to establish the model's robustness across a broader range of real-world scenarios.

5.1.3 Results Analysis

This section aims to dissect and contrast the performance of two sophisticated machine learning models—the EfficientNetB0 and the Vision Transformer (ViT). A meticulous examination of each model's capabilities is conducted, utilizing a comprehensive suite of evaluation metrics to measure their proficiency in classifying images that contain circumferential welds (CW).

In this comparative study, we draw upon the confusion matrices of both models, which delineate the true positives, true negatives, false positives, and false negatives encountered during the classification process. The matrices offer a granular view of the models' performances, revealing their strengths and weaknesses in distinguishing between CW-containing and non-CW images. This analysis is paramount, as it directly informs the choice of the most suitable model for deployment in real-world settings, where accuracy and reliability are non-negotiable.

For clarity and to facilitate direct comparison, we collate the key performance metrics for both models in Table 5.1. These metrics, which include Sensitivity, Specificity, Precision, and others, indicate the models' classification accuracy and predictive power. Mainly, they highlight how each model fares in correctly identifying CW images, avoiding false alarms, and maintaining an overall balance between various classification errors.

The confusion matrix is a powerful tool for evaluating the performance of classification models, particularly in binary classification tasks like ours. It provides a visual representation of the accuracy of the predictions made by the model by comparing the predicted class labels with the actual class labels. For our models, the confusion matrix helps us understand how well the models distinguish between images containing circumferential welds (CW) and those without (non-CW).

In the context of our study, where each model assessed a total of 910 images, the confusion matrix is instrumental in quantifying the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These metrics are crucial as they directly impact other essential performance measures such as precision, recall, and

F1-score.

The true positives and true negatives represent the images correctly identified by the model as CW and non-CW, respectively. These are the ideal outcomes we aim for. False positives occur when the model incorrectly labels a non-CW image as CW, which could lead to unnecessary scrutiny or maintenance actions. Conversely, false negatives represent a more critical error, where the model misses a CW image, potentially overlooking a weld that requires attention.

A balanced ratio of these four outcomes indicates a well-performing classifier, as seen in the confusion matrices for the EfficientNetB0 and Vision Transformer (ViT) models. In our evaluation, we strive for a high number of true positives and true negatives while minimizing the false positives and false negatives, ensuring the reliability and utility of the models in practical scenarios.

Moreover, the confusion matrix allows us to calculate the sensitivity (recall) and specificity of the models—key metrics that provide insights into the models' ability to detect CW images (sensitivity) and their capability to ignore non-CW images (specificity) correctly.

By examining the confusion matrices, detailed in Figure 5.5, we gain valuable feedback on the models' classification prowess, informing further optimization and deployment decisions in our automated inspection system.

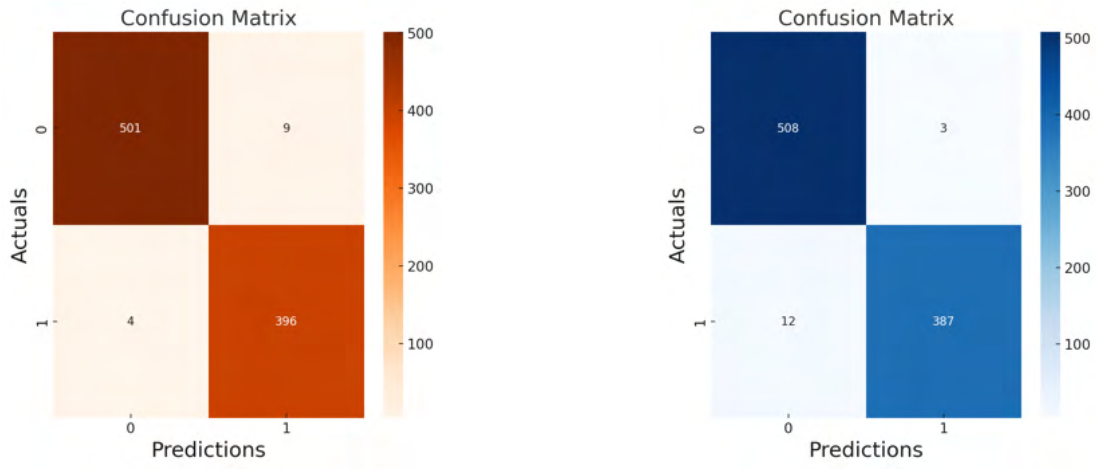


Figure 5.5: Left: Confusion Matrix of the EfficientNetB0 model; Right: Confusion Matrix of the ViT model

The comprehensive evaluation of the binary classification models, EfficientNetB0 and Vision Transformer (ViT), with base patch size 16 and image size 224, reveals significant insights into their performance. Table 5.1 encapsulates the quantitative metrics, and the

ensuing analysis draws comparisons to understand their efficacy in image classification tasks.

Sensitivity, or the True positive rate, indicates the model's ability to identify positive instances correctly. The ViT model outperforms EfficientNetB0 with a sensitivity score of 0.9699 compared to 0.9348, suggesting that ViT is more adept at detecting True circumferential welds within the dataset.

Specificity measures the True negative rate, reflecting the model's capability to recognize negatives accurately. Again, ViT demonstrates superior performance, with a specificity of 0.9941 over EfficientNetB0's 0.9393, indicating fewer false alarms during classification.

Precision, or the positive predictive value, assesses the model's accuracy in predicting positive labels. ViT achieves an impressive precision of 0.9923, significantly higher than EfficientNetB0's 0.9233, indicating that it is usually correct when ViT predicts a weld.

The Negative Predictive Value (NPV) complements precision by reflecting the model's accuracy in predicting negative labels. ViT's NPV stands at 0.9769, superior to EfficientNetB0's 0.9486, indicating ViT's higher reliability in classifying non-weld images.

The False Positive Rate (FPR) and False Discovery Rate (FDR) are critical in scenarios where the cost of false alarms is high. ViT's lower FPR and FDR of 0.0059 and 0.0077, respectively, compared to EfficientNetB0's 0.0607 and 0.0767, signify a model less likely to identify non-welds as welds mistakenly.

Conversely, the False Negative Rate (FNR) indicates missed positive detections. The lower FNR of ViT at 0.0301, against EfficientNetB0's 0.0652, underscores its reduced likelihood of missing actual welds in the classification process.

Accuracy is the overall correctness of the model, and here ViT notably prevails with a score of 0.9835 against EfficientNetB0's 0.9374, marking it as the more accurate model overall.

The F1 Score is a balanced measure that considers both precision and sensitivity. ViT's F1 Score of 0.9810 is considerably higher than EfficientNetB0's 0.9290, suggesting a more harmonious balance between accuracy and sensitivity in the ViT model.

Lastly, the Matthews Correlation Coefficient (MCC), which provides a balanced measure of the quality of binary classifications, strongly favours vit-base-patch16-224 with a score of 0.9666 over EfficientNetB0's 0.8730, indicating a higher correlation between observed and predicted classifications.

In summary, the comparative analysis underscores the vit-base-patch16-224 model's robustness and reliability in classifying images for the presence of circumferential welds. Its consistent outperformance across almost all metrics suggests that it is a more suitable candidate for deployment in practical applications within the domain of weld inspection. This section presents a side-by-side comparison of the two models across several standard metrics.

Table 5.1: Model Evaluation Metrics

Measure	**EfficientNetB0**	**vit-base-patch16-224**
Sensitivity	**0.9900**	0.9699
Specificity	0.9824	**0.9941**
Precision	0.9778	**0.9923**
Negative Predictive Value	**0.9921**	0.9769
False Positive Rate	0.0176	**0.0059**
False Discovery Rate	0.0222	**0.0077**
False Negative Rate	0.0100	**0.0301**
Accuracy	**0.9857**	0.9835
F1 Score	**0.9839**	0.9810
Matthews Correlation Coefficient	**0.9711**	0.9666

5.1.4 Discussion

Comparative Evaluation of EfficientNetB0 and Vision Transformer (ViT)
In the comparative analysis of the EfficientNetB0 and Vision Transformer (ViT) models, distinct characteristics emerge, revealing each model's strengths and potential limitations in image classification tasks involving circumferential welds (CW).

Model Size and Efficiency: A crucial aspect of this comparison lies in the model size, where EfficientNetB0, with a significantly smaller footprint of 139MB, contrasts with ViT's 337MB. This difference is not just a matter of storage efficiency but also impacts the deployment feasibility in resource-constrained environments. EfficientNetB0's compact size makes it more adaptable for integration into systems with limited memory and processing power, such as handheld devices or embedded systems used in industrial inspections.

Training Time Considerations: Another critical factor is the training time. EfficientNetB0 required approximately 1.5 hours for training, substantially longer than ViT's swift 10-minute training period. While this might seem a drawback for EfficientNetB0 at first glance, it's essential to consider the context of model training. The longer training time could be attributed to EfficientNetB0's architecture, which is designed

to balance efficiency and accuracy meticulously. In scenarios where model training is a one-off or infrequent task, the longer training time of EfficientNetB0 may be a reasonable trade-off for its benefits in terms of deployment efficiency and model size.

Performance Metrics Analysis: When considering key performance metrics like sensitivity, specificity, and accuracy, both models demonstrate commendable proficiency. However, the slightly lower performance of EfficientNetB0 in some metrics should be weighed against its size and training time advantages. The trade-offs between these models become a matter of prioritizing either performance or efficiency, depending on the specific requirements of the deployment environment.

Favoring EfficientNetB0 for Resource-Constrained Environments
Given the constraints of model size and deployment context in resource-limited settings, EfficientNetB0 is a favourable choice. Its smaller size and satisfactory performance metrics position it as a more versatile model for practical applications, particularly where model agility and adaptability are paramount.

Model Deployment and Practicality: In real-world scenarios where models need to be deployed on devices at the edge, such as remote inspection vehicles, EfficientNetB0's compactness and efficiency offer tangible benefits. These advantages facilitate easier integration and lower computational demands, which are crucial for continuous, real-time operations in industrial settings.

Balancing Accuracy and Efficiency: While the training time is notably longer for EfficientNetB0, this aspect becomes less significant when considering a model's lifecycle in practical applications. The initial investment in training time is often offset by the benefits gained during the deployment phase, especially in lower computational requirements and greater adaptability to diverse operating conditions.

Concluding Remarks
In conclusion, while ViT demonstrates impressive speed in training and high performance in specific metrics, the EfficientNetB0 model's balance of size, efficiency, and reasonably high accuracy makes it a compelling choice for practical applications in environments where model size and computational efficiency are critical. This analysis underscores the importance of considering model characteristics, including size, training time, and performance metrics, to make informed decisions tailored to specific operational needs and constraints. Future research might explore optimizing EfficientNetB0 further, enhancing its accuracy without significantly impacting its size or efficiency, thereby solidifying its suitability for real-world industrial applications.

5.2 Stage 2: Anomaly Detection Model

The YOLOv8 model achieved a precision peak at a confidence threshold of 0.903, as depicted in Figure 5.6, indicating a high level of accuracy in the predictions made by the model. The recall showed a high value at lower confidence levels, suggesting that the model has a solid capability to detect the presence of anomalies. However, the recall decreased with an increased confidence threshold, a common occurrence in precision-recall trade-offs.

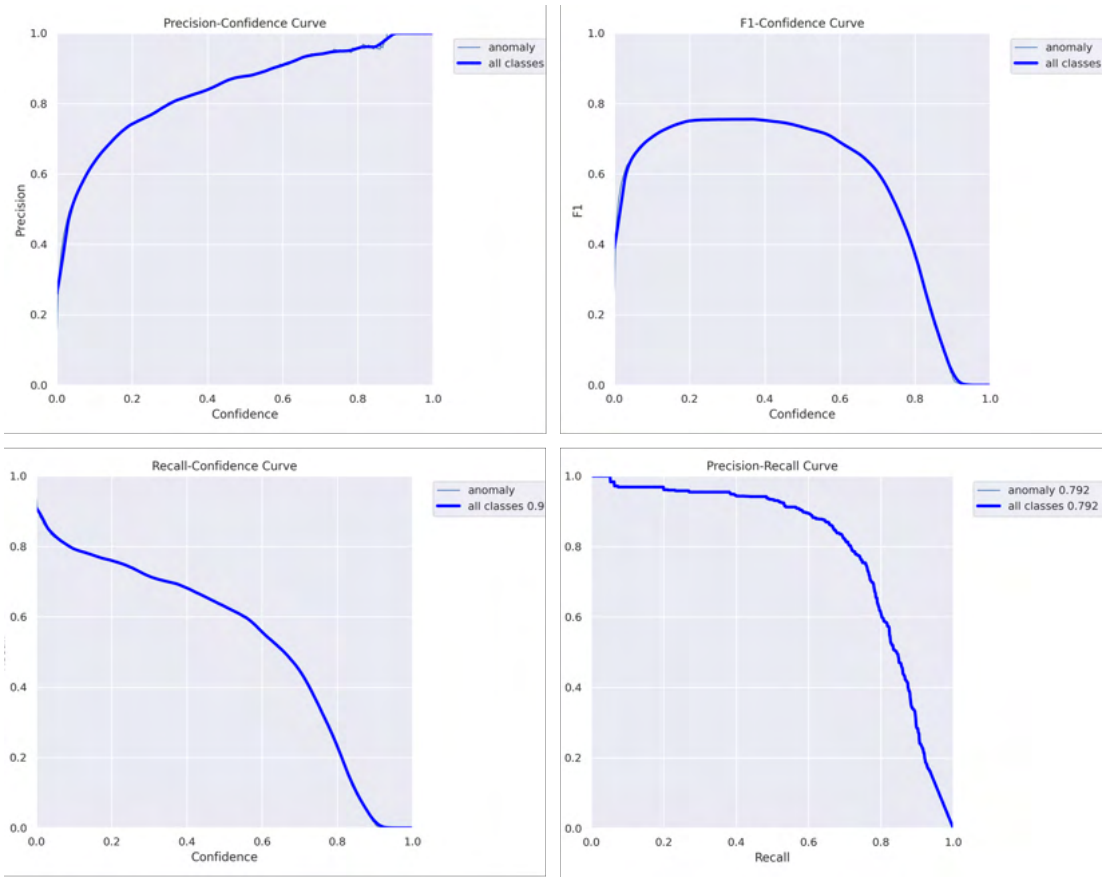


Figure 5.6: Performance evaluation plots including Precision-Confidence, F1-Confidence, Precision-Recall, and Recall-Confidence curves for the YOLO model.

The model's F1 score, which balances precision and recall, peaked at 0.76 for a confidence threshold of approximately 0.36, signifying an optimal balance at this threshold level. The mAP for the model at an IoU threshold of 0.5 was 0.792, demonstrating robust detection across all classes. The mAP at varying IoU thresholds from 0.50 to 0.95 (mAP@0.50-0.95) was also calculated to ensure a comprehensive evaluation across different levels of detection difficulty, with the model achieving a respectable score.

5.2.1 Analysis of Detection Metrics Over Epochs

The evolution of detection metrics throughout training provides critical insights into the model's learning trajectory and eventual proficiency in anomaly detection. As depicted in Figure 5.7, the metrics of precision, recall, and mean average precision (mAP) at varying intersections over union (IoU) thresholds are charted across epochs. Precision, which gauges the model's accuracy in predicting true anomalies, demonstrates a trend of stabilization after initial fluctuations. This behaviour reflects the model's increasing discernment in correctly classifying anomalies as it learns discriminative features over time.

Recall, indicative of the model's sensitivity in identifying all present anomalies, exhibits a gradual ascent to an equilibrium point. This ascent suggests an initial phase of expansive learning where the model can capture a broader spectrum of anomalies. The plateau in recall underlines a state of balance where the model maintains a consistent detection rate for anomalies within the test set.

The mAP metrics are an aggregate measure of the model's precision and recall across multiple IoU thresholds, offering a comprehensive view of performance. The mAP at IoU=0.5 offers a lenient threshold for overlap between predicted and actual bounding boxes. In contrast, the mAP at IoU=0.50-0.95 imposes a more stringent set of criteria, evaluating the model's precision at finer gradations of detection accuracy. The observed stability in mAP scores across epochs underscores the model's robustness and capability to generalize well to unseen data, maintaining a steady level of performance even as the criteria for correct detections are tightened.

The trends observed in these metrics are symbolic of a well-tuned training process. The stabilization of precision and recall points to a model that has effectively converged, achieving a balance between identifying true anomalies and minimizing false positives. The consistent mAP scores across strict IoU thresholds affirm the model's reliability and precision in localizing anomalies. This is vital for practical applications where the exact delineation of defects is crucial for subsequent analysis and decision-making.

5.2.2 Learning Rate and Loss Analysis

The optimization strategy for DL models significantly impacts their learning efficiency and final performance. For the YOLOv8 model, an adaptive learning rate schedule was implemented to optimize the detection of anomalies in inspection images. As illustrated in Figure 5.8, the learning rate for parameter group 0 (PG0) started at a higher magnitude and experienced a rapid decline within the initial epochs. This approach, often called learning rate annealing, exploits the benefits of a more significant learning

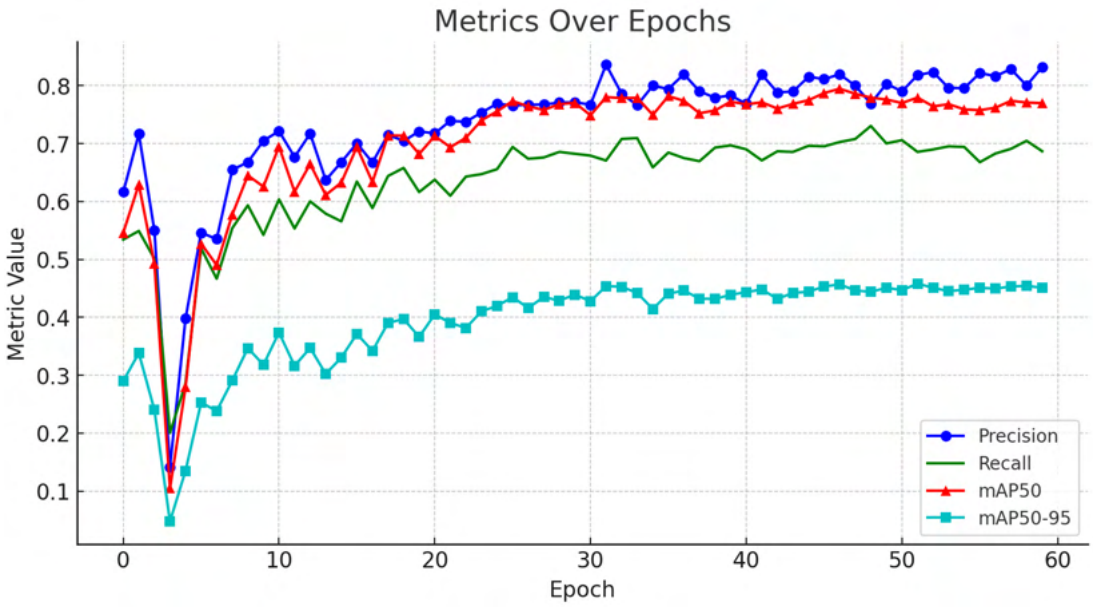


Figure 5.7: Metrics over epochs for the YOLO model, illustrating the precision, recall, mAP at IoU=0.5, and mAP at IoU=0.50-0.95.

rate—navigating the parameter space more broadly to escape local minima—before refining the model parameters as the loss landscape becomes smoother.

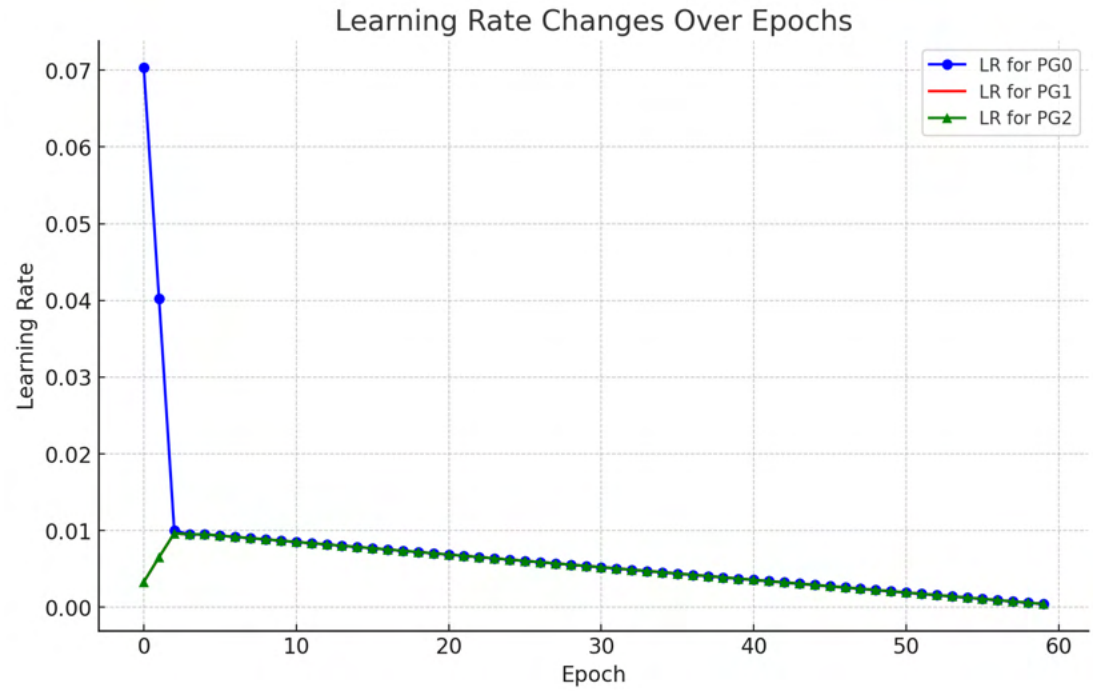


Figure 5.8: Learning rate changes over epochs for different parameter groups, indicating the optimization dynamics of the YOLO model during training.

Following the steep descent, the learning rates for all parameter groups entered a subtle, steady decay phase. This gradual reduction aligns with the principles of fine-tuning, where more minor updates are made to hone in on optimal parameters without overshooting. The continuous, albeit marginal, adjustments suggest employing a decay function or a scheduler like exponential decay, where the learning rate decreases by a multiplicative factor each epoch or a more sophisticated method that adjusts the rate based on validation performance.

The interplay between learning rate and loss is evident when examining the training and validation losses in Figures 5.9 and 5.10. Initially, a pronounced drop in all three loss components—box, classification, and directional field—signifies the model's rapid learning and adaptation. Subsequently, the losses plateau, reflecting the diminishing returns of learning as the model approaches an optimal state. The consistency between the training and validation loss trends indicates that the model is not overfitting to the training data, maintaining its ability to generalize to unseen data, which is crucial for its application in practical scenarios.

Moreover, the directional field loss, representing the model's understanding of object orientations and boundaries, decreases in lockstep with the classification loss. This synchronized reduction underscores the model's concurrent learning of object features and spatial relations, a testament to its comprehensive learning capability.

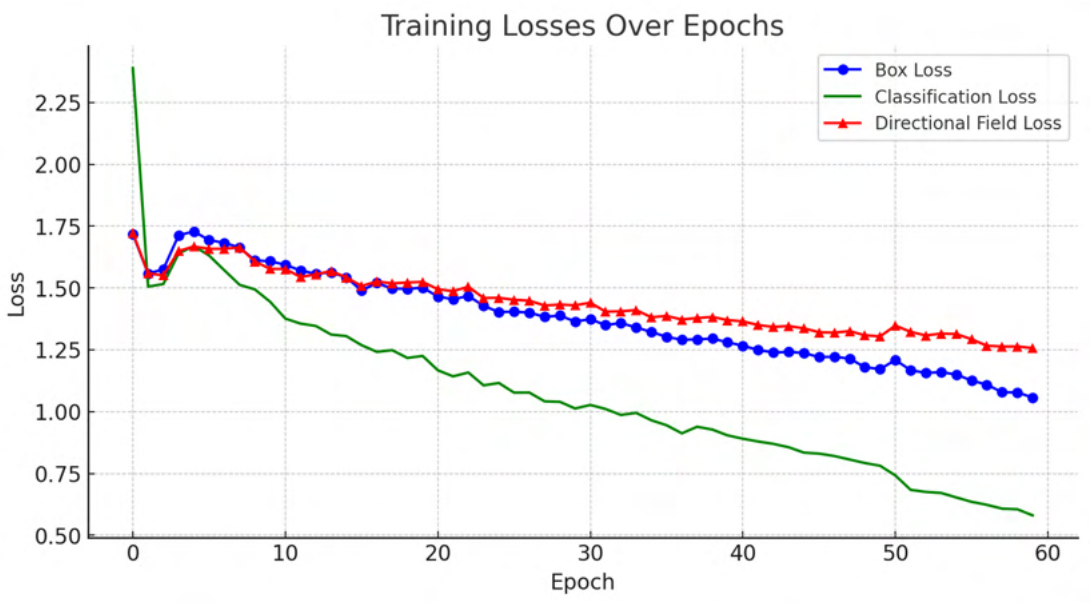


Figure 5.9: Training losses over epochs, displaying the box, classification, and directional field losses indicative of the learning progression.

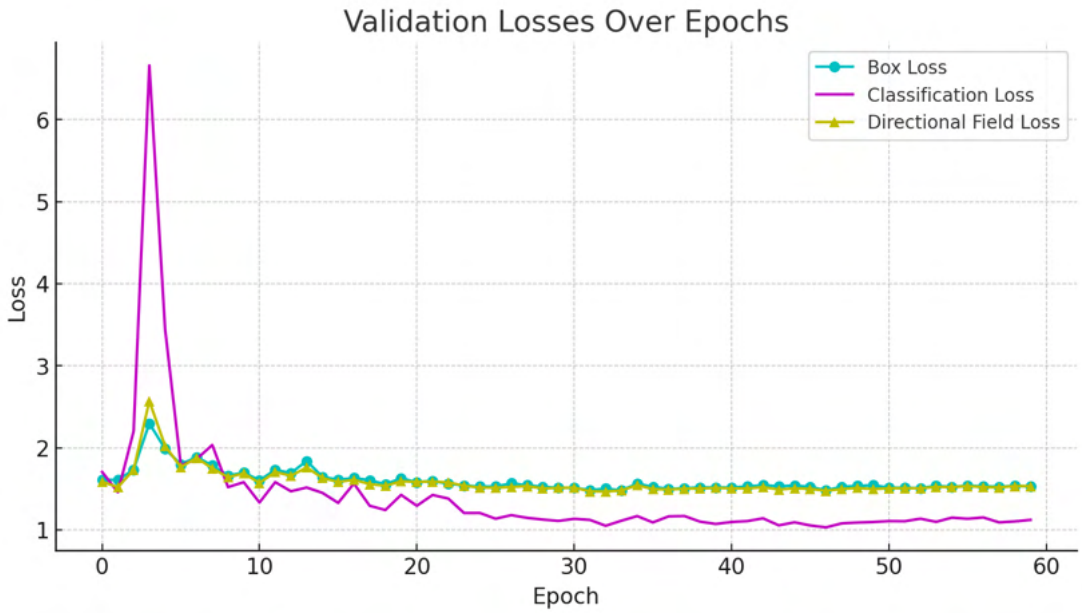


Figure 5.10: Validation losses over epochs for the YOLO model, which aid in understanding the model's generalization performance.

5.2.3 Detection Examples

Visual inspection of the model's predictions on test images provides further insight into its performance. As shown in Figure 5.11, the model successfully detects various anomalies confidently. The bounding boxes closely align with the anomalies, and the confidence scores correlate well with the type and severity of the defects. This visual evidence supports the model's quantitative metrics, showcasing its practical utility in automated inspection tasks.

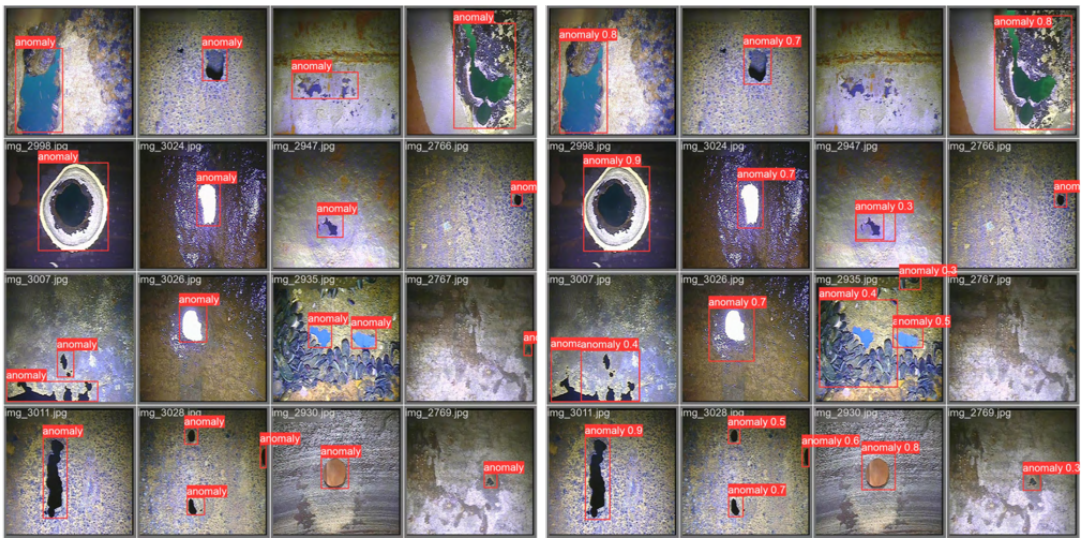


Figure 5.11: Example of actual(left) vs predicted (right) anomalies by the YOLOv8 model, showcasing the model's ability to detect various anomalies with corresponding confidence scores.

5.2.4 Overall Assessment

The quantitative results underscore the YOLOv8 model's capability to serve as an effective tool for anomaly detection. The high precision and mAP scores indicate an accurate and reliable model. In contrast, the high initial recall suggests that the model can be adjusted to prioritize detecting all possible anomalies, an essential requirement for inspection tasks where missing a defect could have severe implications.

These results, combined with the qualitative analysis, affirm the model's applicability in real-world scenarios, particularly in domains where the accurate detection of anomalies is critical to safety and operational integrity.

5.3 Stage 3: Human in the Loop

In this evolving phase of our project, where engineers currently utilise the entire API, a comprehensive evaluation of its effectiveness and efficiency is still underway. While the primary scope of this work was centred around the development and streamlining of the integration of DL model frameworks and data preprocessing to expedite remote visual inspections, the ongoing use of the API by engineers provides invaluable insights for future assessments.

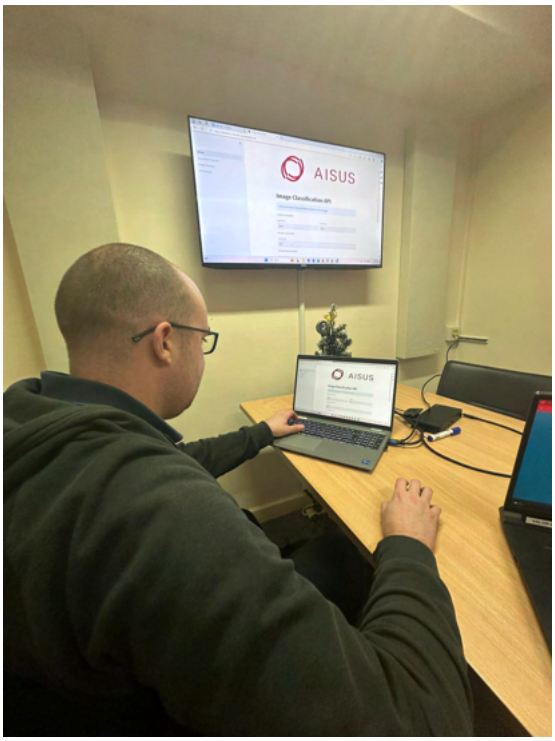


Figure 5.12: User Interaction with API platform

5.3.1 User Feedback

The preliminary observations indicate that the tool has reduced the time engineers spend selecting the appropriate images for reporting by an estimated 20-30%. This significant reduction in time is a testament to the effectiveness of the API's user-friendly design and the integration of advanced DL technologies. However, as the tool is still in its early stages of deployment, a complete evaluation encompassing all aspects of its functionality and user experience is yet to be conducted.

The future evaluation will focus on various critical aspects:

1. User Experience and Efficiency: Assessing how the API's design and workflow optimization contribute to the overall user experience for engineers. This will include evaluating the ease of uploading and processing images, the effectiveness of the tutorial and guidance provided, and the overall usability of the interface.

2. Accuracy and Reliability of DL Models: Understanding how the integrated DL models perform in real-world scenarios, including their accuracy in classifying and detecting anomalies and their adaptability based on engineer feedback.

3. Impact of Human Feedback: Investigating the role of human-in-the-loop feedback in refining and enhancing the model's performance over time. This will

also involve assessing the system's learning curve as it adapts to new data and annotations provided by the engineers.

4. Time Efficiency: Quantitatively measure the time saved in the inspection process due to implementing the API and identify areas where further efficiency can be gained.

5.3.2 Areas of Opportunity

As the full-scale evaluation of the API is still pending, this section will serve as a preliminary acknowledgement of the potential limitations and challenges that might arise. These could include technical issues, user interface improvements, the need for more robust data handling, and ensuring the reliability of the models under varied inspection conditions. We aim to address these challenges in future work, continually refining and enhancing the API based on real-world usage and feedback.

In summary, while the current implementation of the API has shown promising results in terms of time efficiency, a detailed evaluation and refinement process is planned for the near future. This will ensure that the tool maintains its efficacy in speeding up remote visual inspections and evolves to meet the changing needs and challenges of the field.

5.4 Conclusion

The comprehensive evaluation of the General Classification Model, encompassing both the Vision Transformer (ViT) and EfficientNet models, was a crucial aspect of this project to enhance the efficacy of offshore inspection processes. The critical comparison points between these models were multifaceted, focusing on model size, training time, accuracy, and efficiency in real-world scenarios.

Comparison Between Vision Transformer (ViT) and EfficientNet:

1. Model Size and Efficiency: EfficientNet emerged as a more compact model with a smaller footprint of 139MB compared to ViT's 337MB. This smaller size makes EfficientNet particularly suitable for deployment in environments with limited computational resources, such as handheld devices or embedded systems used in offshore inspections.

2. Training Time: EfficientNet required approximately 1.5 hours, substantially longer than ViT's 10-minute training period. However, given the context in which these models are deployed, the longer training time of EfficientNet can be a reasonable trade-off considering its benefits in terms of deployment efficiency and

model size.

3. Performance Metrics: In terms of performance metrics like sensitivity, specificity, and accuracy, both models demonstrated high proficiency. EfficientNet, despite its slightly lower performance in some metrics, was favoured due to its balance between model size and efficiency.

4. Practical Deployment: EfficientNet's balance of size, efficiency, and reasonably high accuracy made it a more pragmatic choice for practical applications, especially in environments where model size and computational efficiency are crucial considerations.

Evaluation of the Anomaly Detection Model (YOLOv8):

The evaluation of YOLOv8 focused on its precision, recall, and mean average precision (mAP) at various intersections over union (IoU) thresholds. The model demonstrated high precision and mAP scores, indicating its accuracy and reliability in anomaly detection. The balance between precision and recall, as evidenced by the model's F1 score, indicated its capability to effectively detect anomalies in inspection images. The learning rate and loss analysis further substantiated the model's comprehensive learning capability and ability to generalize well to unseen data.

Integration into the Company Pipeline:

Integrating these models into the company's pipeline was a pivotal step. Engineers now interact directly with the models via a user-friendly API, streamlining the inspection process. This integration has facilitated a more efficient workflow and opened continuous feedback and improvement channels.

API Design and Workflow: The API's design ensures an intuitive user experience for engineers, enabling easy navigation through image processing and report generation stages.

Human-in-the-Loop Feedback: The system's design incorporates human feedback, allowing engineers to provide inputs that can be used to refine and enhance the model's performance continually. This aspect is crucial for adapting the system to real-world variations and complexities.

Time Efficiency and User Experience: Preliminary observations indicate a significant reduction in engineers' time selecting images for reporting. The ongoing use of the API provides invaluable insights for future assessments and refinements.

In conclusion, the choice of EfficientNet as the preferred model was driven by its balanced attributes suitable for the constrained environments typical in offshore inspections. The evaluation of YOLOv8 highlighted its robustness in anomaly detection. Integrating these models into the company's pipeline, facilitated through a thoughtfully designed API, represents a significant advancement in automating and optimizing offshore inspection tasks. The future evaluation will further enhance this integration, ensuring the system's adaptability and efficiency in real-world applications.

Chapter 6

Conclusion & Future Directions

6.1 Summary of Findings

This dissertation has significantly advanced automated visual inspection systems by developing, integrating, and evaluating DL models tailored for the energy sector, particularly in offshore asset inspections. The following highlights encapsulate the project's objectives and achievements:

In-Depth Literature Review: The review underscored the transformative impact of DL and transfer learning in industrial inspections, spotlighting technologies like Vision Transformer, EfficientNet, and YOLO architectures. This review informed the research and identified gaps and potential for innovation.

Implementation of a Novel-DL Framework : A General Classification Model was developed using state-of-the-art architectures. The ADM, specifically YOLOv8, was designed for precise anomaly detection. These models demonstrated high proficiency in classifying and detecting anomalies in weld images and offshore components.

Human-in-the-Loop API Integration: A significant achievement was integrating these models into a user-friendly API. This integration facilitated direct interaction with the models, streamlining the inspection process and incorporating human feedback, which is crucial for the continual refinement of the models.

Model Performance Evaluation: The evaluation phase thoroughly compared the Vision Transformer and EfficientNet on various parameters like model size, efficiency, and training time. EfficientNet was chosen for its balance between size and efficiency, suitable for constrained offshore environments. YOLOv8 excelled in precision, recall, and mean average precision, indicating its reliability in anomaly detection.

Practical Implementation and Impact: The practical application of these models in offshore asset inspections marked a significant shift towards more efficient, accurate, and automated processes. The integration into the company's pipeline and API design led to a more efficient workflow and reduced the time and resources required for inspections.

The dissertation's outcomes have fulfilled its objectives and set a benchmark in the field, showcasing a systematic approach that blends expert knowledge, cutting-edge technology, and meticulous planning. The resulting models and systems demonstrate a potentially revolutionary impact on the reliability and efficiency of remote visual inspections in structural engineering.

6.2 Contributions to the Field

This dissertation makes significant contributions to offshore engineering and data science by innovatively merging advanced DL technologies with practical industrial applications. The impact of this project on safety, efficiency, and reliability in offshore inspections is profound and multi-dimensional, setting new benchmarks in these critical areas.

Advancing Offshore Engineering Practices: The integration of the General Classification Model and the Anomaly Detection Model, particularly the implementation of YOLOv8, represents a substantial leap in the capabilities of offshore inspection technologies. These models, underpinned by state-of-the-art architectures like the Vision Transformer and EfficientNet, elevate the accuracy and precision of defect detection in challenging offshore environments. This advancement directly translates to enhanced safety in offshore operations, as more accurate and reliable inspections can prevent potential hazards caused by undetected faults.

Innovations in Data Science Applications: The project showcases the potential of data science in industrial applications, specifically in the processing and analysis of complex visual data. The use of DL for image classification and anomaly detection in an industrial context exemplifies how theoretical data science principles can be effectively applied to solve real-world problems. This cross-disciplinary approach not only enriches the field of data science but also opens up new avenues for research and development.

Enhancing Safety in Offshore Inspections: One of the most significant impacts of this dissertation is the improvement of safety standards in offshore inspections. By automating the detection of faults and anomalies in offshore structures, the risk of human error is significantly reduced. This automation ensures a more consistent and thorough inspection process, which is crucial in maintaining the integrity and safety of

offshore facilities.

Improving Efficiency and Reducing Costs: Implementing these models through a human-in-the-loop API optimizes the inspection process, making it more efficient and less time-consuming. This efficiency is a game-changer for offshore engineering, where traditional inspection methods are often slow and labour-intensive. Reducing time and labour directly translates to cost savings, making it an economically attractive solution for the industry.

Reliability and Future Readiness: The dissertation demonstrates the reliability of automated systems in critical inspection tasks. By rigorously evaluating the models against various performance metrics, this research ensures that the systems are effective in current scenarios and adaptable to future challenges. This reliability and adaptability aspect is crucial for offshore inspection technologies' long-term sustainability.

In summary, this dissertation contributes to offshore engineering and data science by showcasing how advanced technological solutions can be effectively employed to enhance safety, efficiency, and reliability in offshore inspections. The project addresses current challenges in the industry and lays a robust foundation for future innovations.

6.3 Future Directions

While marking a significant advance in applying DL to offshore inspections, this dissertation has encountered several challenges that open avenues for future research and enhancements.

Enhancing Data Collection and Model Robustness: One of the primary challenges faced was data diversity and volume limitation. To address this, future research should focus on expanding the dataset to include a wider variety of defect types, especially rare or underrepresented ones. Innovative data collection strategies, such as synthetic data generation and advanced augmentation techniques, could help overcome current limitations, improving the models' robustness and generalization capabilities.

Balancing Model Complexity and Computational Resources: The advanced nature of the DL architectures used posed significant demands on computational resources. Investigating alternative architectures and emerging technologies could lead to more efficient models that balance computational efficiency with performance. Future work could explore newer neural network versions, hybrid models, and advancements in edge computing to enhance on-site processing capabilities.

Adapting to Real-world Conditions and Scalability: Translating the models from controlled environments to real-world offshore settings revealed challenges due

to environmental variability. Extending the models' applicability to various offshore assets and conditions is vital. This involves scaling the models, ensuring adaptability to different inspection types, and maintaining performance in dynamic operational environments.

Integration, User Acceptance, and Ethical Considerations: Integrating the models into existing workflows and achieving user acceptance remains an ongoing process. Future efforts should continue to focus on developing user-friendly interfaces and addressing resistance to technological change. Maintaining ethical considerations in automated systems is crucial for gaining trust and acceptance.

Long-term Performance Evaluation: Continuous and comprehensive evaluation of the models in real-world scenarios is essential. Longitudinal studies assessing the models' long-term performance under various operational conditions will be crucial to identify areas for improvement and ensure reliability.

In summary, the future direction of this research involves refining the existing framework and exploring new methodologies and technologies. The aim is to continually enhance the safety, efficiency, and reliability of offshore inspections through innovative deep-learning applications, ensuring they are adaptable, user-friendly, and meet the sector's ethical standards.

Bibliography

Alharam, Aysha et al. (2020). "Real Time AI-Based Pipeline Inspection using Drone for Oil and Gas Industries in Bahrain". In: *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pp. 1–5. DOI: 10.1109/3ICT51146.2020.9312021.

Ali, Sayyed Bashar et al. (2020). "Wall Crack Detection Using Transfer Learning-based CNN Models". In: *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1–7. DOI: 10.1109/INDICON49873.2020.9342392.

Altabey, Wael A. et al. (2022). "A Deep Learning-Based Approach for Pipeline Cracks Monitoring". In: *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pp. 1–6. DOI: 10.1109/ICECCME55909.2022.9987998.

Atha, Deegan J and Mohammad R Jahanshahi (2018). "Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection". In: *Structural Health Monitoring* 17.5, pp. 1110–1128. DOI: 10.1177/1475921717737051. URL: https://doi.org/10.1177/1475921717737051.

Bastian, Blossom Treesa et al. (2019). "Visual inspection and characterization of external corrosion in pipelines using deep neural network". In: *NDT E International* 107, p. 102134. ISSN: 0963-8695. DOI: https://doi.org/10.1016/j.ndteint.2019.102134. URL: https://www.sciencedirect.com/science/article/pii/S096386951930060X.

Bhavani, Nallamilli P.G. et al. (2022). "Real-Time Inspection in Detection Magnetic Flux Leakage by Deep Learning Integrated with Concentrating Non-Destructive Principle and Electromagnetic Induction". In: *IEEE Instrumentation Measurement Magazine* 25.7, pp. 48–54. DOI: 10.1109/MIM.2022.9908257.

Chen, Pengchao et al. (2023). "A cascaded deep learning approach for detecting pipeline defects via pretrained YOLOv5 and ViT models based on MFL data". In: *Mechanical Systems and Signal Processing* 206, p. 110919. DOI: 10.1016/j.ymssp.2023.110919. URL: https://doi.org/10.1016/j.ymssp.2023.110919.

De Masi, Giulia et al. (May 2015). "Machine learning approach to corrosion assessment in subsea pipelines". In: *OCEANS 2015 - Genova*, pp. 1–6. DOI: 10.1109/OCEANS-Genova.2015.7271592.

de Moura, Nájla Vilar Aires et al. (2022). "Deep-water oil-spill monitoring and recurrence analysis in the Brazilian territory using Sentinel-1 time series and deep learning". In: *International Journal of Applied Earth Observation and Geoinformation* 107, p. 102695. ISSN: 1569-8432. DOI: https://doi.org/10.1016/j.jag.2022.102695. URL: https://www.sciencedirect.com/science/article/pii/S0303243422000216.

De Tomi, G. et al. (2014). "Challenges for the inspection of Pre-Salt ultra-deep offshore production facilities". In: *2014 Oceans - St. John's*, pp. 1–7. DOI: 10.1109/OCEANS.2014.7003301.

Dias, A. et al. (2022). "Unmanned Aerial Vehicle for Wind-Turbine Inspection. Next Step: Offshore". In: *OCEANS 2022, Hampton Roads*, pp. 1–6. DOI: 10.1109/OCEANS47191.2022.9977308.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929. arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

Fu, Guizhong et al. (2019). "A deep-learning-based approach for fast and robust steel surface defects classification". In: *Optics and Lasers in Engineering* 121, pp. 397–405. ISSN: 0143-8166. DOI: https://doi.org/10.1016/j.optlaseng.2019.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0143816619301678.

Gašparović, Boris et al. (2023a). "Evaluating YOLOV5, YOLOV6, YOLOV7, and YOLOV8 in Underwater Environment: Is There Real Improvement?" In: *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–4. DOI: 10.23919/SpliTech58164.2023.10193505.

– (2023b). "Evaluating YOLOV5, YOLOV6, YOLOV7, and YOLOV8 in Underwater Environment: Is There Real Improvement?" In: *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–4. DOI: 10.23919/SpliTech58164.2023.10193505.

Health and Safety Executive (2009). "HSE". In: *Structural Integrity Management Framework for Fixed Jacket Structures*. URL: https://www.hse.gov.uk/research/rrpdf/rr684.pdf.

Hoang, Nhat-Duc and Tran Duc (June 2019). "Image Processing Based Detection of Pipe Corrosion Using Texture Analysis and Metaheuristic-Optimized Machine Learning Approach". In: *Computational Intelligence and Neuroscience* In Press. DOI: 10.1155/2019/8097213.

Huang, Hong-wei, Qing-tong Li, and Dong-ming Zhang (2018). "Deep learning based image recognition for crack and leakage defects of metro shield tunnel". In: *Tunnelling and Underground Space Technology* 77, pp. 166–176. ISSN: 0886-7798. DOI: https://doi.org/10.1016/j.tust.2018.04.002. URL: https://www.sciencedirect.com/science/article/pii/S0886779817310258.

Kolesnikov, Alexander, Lucas Beyer, Xiaohua Zhai, et al. (2019). "Large Scale Learning of General Visual Representations for Transfer". In: *CoRR* abs/1912.11370. arXiv: 1912.11370. URL: http://arxiv.org/abs/1912.11370.

Komijani, A. et al. (2022). "Multi-label Classification of Steel Surface Defects Using Transfer Learning and Vision Transformer". In: *2022 13th International Conference on Information and Knowledge Technology (IKT)*, pp. 1–5. DOI: 10.1109/IKT57960.2022.10039038.

Langenkämper, Daniel et al. (2020). "Efficient visual monitoring of offshore windmill installations with online image annotation and deep learning computer vision". In: *Global Oceans 2020: Singapore – U.S. Gulf Coast*, pp. 1–6. DOI: 10.1109/IEEECONF38699.2020.9389305.

Lin, Chia-Yu, Yan-Hung Chou, and Yun-Chiao Cheng (2023). "A Deep Learning-based General Defect Detection Framework for Automated Optical Inspection". In: *2023 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pp. 332–337. DOI: 10.1109/IAICT59002.2023.10205799.

Liu, Yang et al. (2023). "NDT Method for Weld Defects Based on FMPVit Transformer Model". In: *IEEE Access* 11, pp. 61390–61400. DOI: 10.1109/ACCESS.2023.3283589.

Lu, Kai-Liang (2021). *Evaluation and Comparison of Deep Learning Methods for Pavement Crack Identification with Visual Images*. arXiv: 2112.10390 [cs.CV].

Estimating Wall Loss Risk Distributions Using Machine Learning and Geospatial Analytics (June 2020). Vol. All Days. NACE-2020-14640. eprint: https://onepetro.org/NACECORR/proceedings-pdf/CORR20/All-CORR20/NACE-2020-14640/2247055/nace-2020-14640.pdf.

Medak, Duje et al. (2022). "DefectDet: A deep learning architecture for detection of defects with extreme aspect ratios in ultrasonic images". In: *Neurocomputing* 473, pp. 107–115. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.12.008. URL: https://www.sciencedirect.com/science/article/pii/S0925231221018464.

Mohamed, Abduljalil, Mohamed Salah Hamdi, and Sofiène Tahar (2015). "A Machine Learning Approach for Big Data in Oil and Gas Pipelines". In: *2015 3rd International Conference on Future Internet of Things and Cloud*, pp. 585–590. DOI: 10.1109/FiCloud.2015.54.

Mohamed, Yasser S. et al. (2019). "Steel crack depth estimation based on 2D images using artificial neural networks". In: *Alexandria Engineering Journal* 58.4, pp. 1167–1174. ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2019.10.001. URL: https://www.sciencedirect.com/science/article/pii/S1110016819301048.

Moreno-Garcia, Carlos Francisco and Eyad Elyan (2019). "Digitisation of Assets from the Oil Gas Industry: Challenges and Opportunities". In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Vol. 7, pp. 2–5. DOI: 10.1109/ICDARW.2019.60122.

O'Byrne, Michael et al. (2018). "Semantic Segmentation of Underwater Imagery Using Deep Networks Trained on Synthetic Imagery". In: *Journal of Marine Science and Engineering* 6.3. ISSN: 2077-1312. DOI: 10.3390/jmse6030093. URL: https://www.mdpi.com/2077-1312/6/3/93.

Ortiz, Alberto et al. (2016). "Vision-based corrosion detection assisted by a micro-aerial vehicle in a vessel inspection application". In: *Sensors* 16.12. ISSN: 1424-8220. DOI: 10.3390/s16122118. URL: https://www.mdpi.com/1424-8220/16/12/2118.

Oyama, Akira et al. (July 2021). "Detection of rust from images in pipes using deep learning". In: *2021 18th International Conference on Ubiquitous Robots (UR)*, pp. 476–479. DOI: 10.1109/UR52253.2021.9494700.

Perez, Husein, Joseph Tah, and Amir Mosavi (Aug. 2019). *Deep learning for detecting building defects using convolutional neural networks*. DOI: 10.20944/preprints201908.0068.v1.

Pirie, Craig and Carlos Francisco Moreno-Garcia (Aug. 2021). "Image pre-processing and segmentation for real-time subsea corrosion inspection." In: *22nd Engineering Applications of Neural Networks Conference (EANN2021)*. Ed. by Lazaros Iliadis et al. Springer, pp. 220–231. ISBN: 9783030805678. DOI: 10.1007/978-3-030-80568-5_19. URL: https://rgu-repository.worktribe.com/output/1369876.

Prasad, Jayashree Rajesh, Ayushi Parikh, and Hridya K Prasanth (2023). "Exploration of Deep Learning Based Underwater Image Processing Techniques". In: *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1222–1225.

Redmon, Joseph et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

Regulator, Offshore Safety Directive (2021). "Offshore Safety Directive Regulator". In: *Caisson Structural Integrity*. URL: https://www.hse.gov.uk/offshore/infosheets/is5-2019.pdf.

Ren, Ruoxu, Terence Hung, and Kay Chen Tan (2018). "A Generic Deep-Learning-Based Approach for Automated Surface Inspection". In: *IEEE Transactions on Cybernetics* 48.3, pp. 929–940. DOI: 10.1109/TCYB.2017.2668395.

Sarker, Iqbal H. (2021). "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions". In: *SN Computer Science* 2.420. DOI: 10.1007/s42979-021-00815-1. URL: https://doi.org/10.1007/s42979-021-00815-1.

Smith, Amos, Jeremy Coffelt, and Kai Lingemann (2022). "A Deep Learning Framework for Semantic Segmentation of Underwater Environments". In: *OCEANS 2022, Hampton Roads*, pp. 1–7. DOI: 10.1109/OCEANS47191.2022.9977212.

Soares, Luciane et al. (2021). "A Visual Inspection Proposal to Identify Corrosion Levels in Marine Vessels Using a Deep Neural Network". In: *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, pp. 222–227. DOI: 10.1109/LARS/SBR/WRE54079.2021.9605400.

Stamoulakatos, Anastasios, Javier Cardona, Chris Mccaig, et al. (Jan. 2020). "Automatic Annotation of Subsea Pipelines Using Deep Learning". In: *Sensors* 20. DOI: 10.3390/s20030674.

Sudevan, Vidya, Amit Shukla, and Hamad Karki (2018). "Current and Future Research Focus on Inspection of Vertical Structures in Oil and Gas Industry". In: *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 144–149.

Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 6105–6114.

Waqar, Asad, Idris Othman, Nasir Shafiq, et al. (2023). "Applications of AI in oil and gas projects towards sustainable development: a systematic literature review". In: *Artificial Intelligence Review* 56, pp. 12771–12798. DOI: 10.1007/s10462-023-10467-7. URL: https://doi.org/10.1007/s10462-023-10467-7.

Wu, Xiaojun et al. (2021). "Deep Learning-Based Generic Automatic Surface Defect Inspection (ASDI) With Pixelwise Segmentation". In: *IEEE Transactions on Instrumentation and Measurement* 70, pp. 1–10. DOI: 10.1109/TIM.2020.3026801.

Xia, Yiyu et al. (2018). "A Deep Learning Based Image Recognition and Processing Model for Electric Equipment Inspection". In: *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1–6. DOI: 10.1109/EI2.2018.8582593.

Yan, Y. et al. (2020). “A Deep Learning-Based Ultrasonic Pattern Recognition Method for Inspecting Girth Weld Cracking of Gas Pipeline”. In: *IEEE Sensors Journal* 20.14, pp. 7997–8006. DOI: 10.1109/JSEN.2020.2982680.

Yu, Tang et al. (2022). “Intelligent Detection Method of Forgings Defects Detection Based on Improved EfficientNet and Memetic Algorithm”. In: *IEEE Access* 10, pp. 79553–79563. DOI: 10.1109/ACCESS.2022.3193676.

Zhang, Yanhui et al. (June 2013a). “Measurement and Modelling of Residual Stresses in Offshore Circumferential Welds”. In: vol. 3. DOI: 10.1115/OMAE2013-10234.

– (June 2013b). “Measurement and Modelling of Residual Stresses in Offshore Circumferential Welds”. In: vol. 3. DOI: 10.1115/OMAE2013-10234.

Zhu, Haixing, Weimin Ge, and Zhenzhong Liu (2019). “Deep Learning-Based Classification of Weld Surface Defects”. In: *Applied Sciences* 9.16. ISSN: 2076-3417. DOI: 10.3390/app9163312. URL: https://www.mdpi.com/2076-3417/9/16/3312.

Appendix A

Annex

A.1 Custom Filter

```
import cv2
import numpy as np
import time
import glob

# Create a CLAHE object (Contrast Limited Adaptive Histogram Equalization)
clahefilter = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(16,16))

# Initialize a counter for the image files
cont = 0
for filename in glob.glob("images/*.jpg"): # Iterate over jpg files in the 'clean' ↩
    directory

    # Read the image
    img = cv2.imread(filename)

    # Convert the image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Define the range for glare in HSV color space
    GLARE_MIN = np.array([0, 0, 50], np.uint8)
    GLARE_MAX = np.array([0, 0, 225], np.uint8)

    # Convert the image to HSV color space
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # Create a mask for the glare
    frame_threshed = cv2.inRange(hsv_img, GLARE_MIN, GLARE_MAX)

```

```
    # Apply inpainting to the original image using the glare mask
    result = cv2.inpaint(img, frame_threshed, 0.1, cv2.INPAINT_TELEA)

    # Apply CLAHE to the grayscale image
    clahecorrecttedFrame = clahefilter.apply(gray)

    # Convert the image to LAB color space for color enhancement
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    lab_planes = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    lab_planes[0] = clahe.apply(lab_planes[0])
    lab = cv2.merge(lab_planes)
    clahe_bgr = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

    # Convert the enhanced color image to grayscale
    grayimg1 = cv2.cvtColor(clahe_bgr, cv2.COLOR_BGR2GRAY)
    # Create a mask for the bright areas in the enhanced image
    mask2 = cv2.threshold(grayimg1, 220, 255, cv2.THRESH_BINARY)[1]
    # Apply inpainting to the original image using the new mask
    result2 = cv2.inpaint(img, mask2, 0.1, cv2.INPAINT_TELEA)

    # Combine the previous inpainting result with CLAHE
    lab1 = cv2.cvtColor(result, cv2.COLOR_BGR2LAB)
    lab_planes1 = cv2.split(lab1)
    clahe1 = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    lab_planes1[0] = clahe1.apply(lab_planes1[0])
    lab1 = cv2.merge(lab_planes1)
    clahe_bgr1 = cv2.cvtColor(lab1, cv2.COLOR_LAB2BGR)

    # Write the final image to a file with a modified name
    cv2.imwrite(filename[0:5] + '_' + str(cont) + '.jpg', clahe_bgr1)
    cont += 1
```

Listing A.1: Image enhancement process using OpenCV.

A.2 General Classifier

A.2.1 Visual Transformer (ViT)

The Visual Transformer (ViT) represents a class of models that employ the transformer architecture, which has been highly successful in natural language processing, to the domain of image classification. This section outlines the steps taken to train and evaluate a ViT model, starting with the setup of the environment and installation of necessary libraries.

Environment Setup and Library Installation

Before training the ViT model, it is necessary to mount the Google Drive to access the dataset and change the working directory to the folder containing the dataset. Additionally, relevant libraries, particularly Hugging Face's Transformers library, need to be installed as it provides the implementation of the ViT model.

```
from google.colab import drive
drive.mount("/content/gdrive")
```

Listing A.2: Mounting the Google Drive to access the dataset.

```
%cd /content/gdrive/MyDrive/Work/image_classification/dataset_TW
```

Listing A.3: Changing the current working directory to the dataset directory.

```
!pip install -q git+https://github.com/huggingface/transformers
```

Listing A.4: Installing the Hugging Face Transformers library.

This setup is a prerequisite for further actions such as data preprocessing, model training, and evaluation. Ensuring that the correct libraries are installed is crucial for the seamless functioning of the model training code that follows.

Model Training Procedure

Once the environment is configured and the necessary libraries are in place, we commence with the training of the ViT model. The dataset is loaded into data loaders, which handle batching, shuffling, and providing data to the model during the training loop. The code block below demonstrates the initialization of data loaders and the training loop where the model parameters are updated in each epoch.

```
import torch.utils.data as data
from torch.autograd import Variable
import numpy as np

# Detect the available number of classes based on the dataset
print("Number of train samples: ", len(train_ds))
print("Number of test samples: ", len(valid_ds))
print("Detected Classes are: ", valid_ds.class_to_idx)

# Data loaders for batching, shuffling, and loading data in parallel
train_loader = data.DataLoader(train_ds, batch_size=BATCH_SIZE, shuffle=True,
    num_workers=2)
test_loader = data.DataLoader(valid_ds, batch_size=BATCH_SIZE, shuffle=True,
    num_workers=2)

```

```
# Training loop
for epoch in range(EPOCHS):
    model.train()
    for step, (x, y) in enumerate(train_loader):
        # Preprocessing of input batch and forward pass
        # ...

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Listing A.5: Training the Visual Transformer model.

In this code segment, the model is set to training mode. Each batch of data is passed through the model, the loss is calculated, and the gradients are propagated back through the model (backpropagation). The optimizer then updates the model's weights. This process is repeated for a specified number of epochs or until a certain condition, such as early stopping, is met.

This training procedure is iterative and can be adjusted based on the performance observed during the validation phase, which helps in fine-tuning the model to achieve better accuracy on the dataset.

Model Evaluation

Post-training, it is critical to evaluate the ViT model's performance to ensure it has learned to classify images accurately. The model evaluation is done using the validation set, where we calculate the loss and accuracy to understand the model's effectiveness. The following code details the evaluation loop where the model is set to evaluation mode, thus disabling any training-specific operations like dropout.

```
# Model evaluation on validation set
model.eval()
val_loss = 0
accuracy = 0
num_samples = 0
with torch.no_grad():
    for step, (x, y) in enumerate(test_loader):
        # Forward pass and loss computation
        # ...

        # Update total validation loss
        val_loss += loss.item() * x.size(0)
        # Calculate accuracy
        # ...

```

```
val_loss /= num_samples
accuracy /= num_samples
print(f'Validation loss: {val_loss:.4f}, Validation accuracy: {accuracy:.4f}')
```

Listing A.6: Evaluation of the Visual Transformer model on the validation set.

This validation loop traverses through the validation dataset and aggregates the losses and correct predictions to compute the average loss and accuracy over all validation samples. These metrics provide an indication of how well the model will perform on unseen data.

Moreover, this section could include any early stopping mechanisms used to halt the training process if the model ceases to improve, thus preventing overfitting and saving computational resources.

```
# Early stopping condition
if val_loss < min_val_loss:
    min_val_loss = val_loss
    patience_counter = 0
else:
    patience_counter += 1
    if patience_counter >= EARLY_STOPPING_PATIENCE:
        print("Early stopping triggered. Stopping training.")
        break
```

Listing A.7: Implementing early stopping based on validation loss.

Early stopping is an approach used to terminate the training process if the model's performance on the validation set does not improve for a specified number of epochs, referred to as the "patience" period. This is an effective method to avoid overfitting and is especially useful when training deep learning models.

Inference and Performance Visualization

Following the model's evaluation, it is insightful to conduct inference on individual test images. This step is critical for visually assessing the model's predictive capability. The code below demonstrates how to load a batch of test images, perform inference to obtain the model's predictions, and then visualize these alongside the true labels for comparison.

```
# Inference on test images
import matplotlib.pyplot as plt

# Load a batch of test images
eval_loader = data.DataLoader(test_ds, batch_size=EVAL_BATCH, shuffle=True)
```

```
with torch.no_grad():
    # Obtain a batch of test images and labels
    inputs, targets = next(iter(eval_loader))
    # ...

    # Generate predictions for the batch
    predictions = model(inputs)
    # ...

    # Visualize the test images and the model's predictions
    for i in range(EVAL_BATCH):
        plt.subplot(1, EVAL_BATCH, i + 1)
        plt.imshow(inputs[i].permute(1, 2, 0).numpy())
        plt.title(f'Predicted: {predicted_labels[i]}, Actual: {target_labels[i]}')
        plt.axis('off')
    plt.show()
```

Listing A.8: Performing inference on test images and visualizing predictions.

This visualization provides an immediate qualitative assessment of the model's performance on the test set. Such visual feedback is invaluable for understanding the model's behavior, including any systematic errors it may be making.

Additionally, it is beneficial to compute and visualize quantitative metrics such as the confusion matrix to summarize the performance of the model in a format that is easily interpretable.

```
from sklearn.metrics import classification_report, confusion_matrix
from mlxtend.plotting import plot_confusion_matrix

# Compute the confusion matrix
conf_matrix = confusion_matrix(target_labels, predicted_labels)

# Plot the confusion matrix
fig, ax = plot_confusion_matrix(conf_mat=conf_matrix, figsize=(6, 6), cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

Listing A.9: Generating a confusion matrix to visualize the model's performance.

The confusion matrix and classification report provide a comprehensive overview of the model's performance across all classes, highlighting the true positive, false positive, true negative, and false negative predictions. These tools are essential for fine-tuning

the model and can guide further improvements to the training process or model architecture.

Model Persistence

Preserving the trained model allows us to deploy the model in different environments or continue development at a later time without retraining from scratch. The code snippet below outlines the process of saving the entire ViT model to disk. This includes the model architecture, weights, and training configuration, enabling an exact replica of the model to be reloaded.

```
# Save the entire model to a file
torch.save(model.state_dict(), 'ViT_model.pth')

print("Model saved successfully!")
```

Listing A.10: Saving the trained Visual Transformer model to disk.

The model is saved using PyTorch's native 'save' function, which serializes the model to a file. The saved model file can then be reloaded using PyTorch's 'load' function and the model's 'load-state-dict' method, restoring the saved model's state.

```
# Load the model for inference or further training
model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load('ViT_model.pth'))
model.eval()

print("Model loaded successfully for inference!")
```

Listing A.11: Loading the saved Visual Transformer model.

This process is particularly important for operationalizing the model in production environments or for conducting further experiments and research. It ensures that the model's exact configuration and learned knowledge are preserved and can be easily shared or deployed.

A.2.2 EfficientNet Model

The following sections describe the implementation details of the EfficientNet model used for image classification. The process begins with the configuration of the computing environment to utilize TensorFlow's capabilities fully, followed by the data preprocessing steps necessary for training such a model effectively.

Environment Setup

The training of deep learning models can be significantly accelerated by leveraging specialized hardware such as TPUs (Tensor Processing Units) or GPUs (Graphics Processing Units). The code snippet below illustrates the setup of a TPU client within the TensorFlow framework. This setup is essential for utilizing the TPU's computing power efficiently, as it configures the TPU to the correct version of TensorFlow that we are using.

```
from cloud_tpu_client import Client
c = Client()
c.configure_tpu_version(tf.__version__, restart_type="always")
```

Listing A.12: Configuring TensorFlow to use a TPU environment

Following the TPU configuration, we establish TensorFlow's distribution strategy. This strategy allows for a model to be trained on multiple TPU cores simultaneously, which is crucial for handling large datasets and complex models like EfficientNet. If a TPU is not available, the strategy defaults to using available CPUs or GPUs, ensuring that the training process can still proceed.

```
import tensorflow as tf

try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver.connect()
    print("Device:", tpu.master())
    strategy = tf.distribute.TPUStrategy(tpu)
except ValueError:
    print("Not connected to a TPU runtime. Using CPU/GPU strategy")
    strategy = tf.distribute.MirroredStrategy()
```

Listing A.13: Setting up TensorFlow's distribution strategy

Data Access

Training a model like EfficientNet requires access to a substantial amount of data. When working in cloud-based notebooks such as Google Colab, it's common to store and retrieve training data from Google Drive. The code below demonstrates how to mount a Google Drive in Colab and change the directory to the location where the dataset is stored. This step ensures that our training environment has direct access to the data, allowing for efficient reading and processing of the dataset required for training the model.

```
from google.colab import drive
drive.mount("/content/gdrive")
```

Listing A.14: Mounting Google Drive to access the dataset

Once the drive is mounted, we navigate to the specific directory within the drive where the dataset resides. This is typically a folder that contains subdirectories for training, validation, and testing data.

```
%cd /content/gdrive/MyDrive/ds_small
```

Listing A.15: Changing the current working directory to the dataset directory

Loading and Preprocessing the Dataset

With the training environment set up and the data access established, the next step is to load the dataset. This involves specifying the paths to the data, and utilizing TensorFlow's utilities to preprocess the images to be suitable for feeding into the EfficientNet model. The following code demonstrates the initialization of the dataset variables and the use of TensorFlow's `image_dataset_from_directory` to efficiently load the data.

```
from tensorflow import keras

batch_size = 64

# specify the path to the train and test folders
train_dir = "train"
val_dir = "val"
test_dir = "test"

# Loading the dataset
ds_train = keras.preprocessing.image_dataset_from_directory(
    train_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(224, 224),
    batch_size=batch_size
)

ds_val = keras.preprocessing.image_dataset_from_directory(
    val_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(224, 224),
    batch_size=batch_size
)
```

```
28
29 # Check labels
30 print(list(ds_train.as_numpy_iterator())[0][1])
```

Listing A.16: Loading the dataset using TensorFlow's image-dataset-from-directory

This segment of the code is vital as it not only loads the data but also splits it into training and validation sets, which is necessary for evaluating the model's performance during training and making adjustments if needed. The batch size is defined, and the image size is set to match the input size expected by the EfficientNet model. Additionally, a seed for reproducibility is specified to ensure that the dataset splits are consistent across different runs.

The last line of the code outputs the labels of the first batch of the training dataset, which serves as a quick check to ensure that the data is loaded correctly and that the labels are as expected.

This careful loading and checking of the dataset set the stage for an effective training process, ensuring that the model has the right data in the right format for learning.

Model Initialization and Training

The core of our model training involves initializing the EfficientNet model with pretrained ImageNet weights and setting it up for transfer learning. The model is adapted for our specific binary classification task by adding a custom top layer. The following code illustrates the process of building the model, compiling it with the necessary loss function and metrics, and then training it with our dataset.

```
1  from tensorflow.keras.applications import EfficientNetB0
2  from tensorflow.keras import layers
3  import tensorflow as tf
4
5  def build_model(num_classes):
6      inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
7      x = img_augmentation(inputs)
8      model = EfficientNetB0(include_top=False, input_tensor=x, weights="imagenet")
9
10     # Freeze the pretrained weights
11     model.trainable = False
12
13     # Rebuild top
14     x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
15     x = layers.BatchNormalization()(x)
16
17     top_dropout_rate = 0.2
```

```
    x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
    outputs = layers.Dense(num_classes, activation="sigmoid", name="pred")(x)

    # Compile the model
    model = tf.keras.Model(inputs, outputs, name="EfficientNet")
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
    model.compile(
        optimizer=optimizer, loss="binary_crossentropy", metrics=["accuracy"]
    )
    return model
```

Listing A.17: Building and compiling the EfficientNet model for binary classification.

With the model built and compiled, the next snippet of code demonstrates the training process within the scope of TensorFlow's distribution strategy. This is critical for leveraging the TPU or GPU setup configured earlier.

```
NUM_CLASSES = 1 # Binary classification
IMG_SIZE = 224 # Size of the input images

with strategy.scope():
    model = build_model(num_classes=NUM_CLASSES)

epochs = 25
hist = model.fit(
    ds_train,
    epochs=epochs,
    validation_data=ds_val,
    verbose=2
)
```

Listing A.18: Training the model using TensorFlow's distribution strategy.

The strategy's scope ensures that the model is distributed across available devices, which can significantly speed up the training process. The model is trained for a specified number of epochs, with the training and validation datasets defined earlier.

This process encapsulates the first phase of the model's training, where the base layers with the ImageNet weights are frozen to ensure that the learned features are retained during the initial adaptation to our dataset. The model's performance is tracked through accuracy and loss metrics, providing insight into the training process and allowing for early detection of issues such as overfitting or underfitting.

Training Visualization and Model Persistence

After the training process, it is important to visualize the training and validation metrics to understand the model's learning behavior over time. The following code segment plots the accuracy and loss for both training and validation sets, providing a visual representation of the model's performance across epochs.

```
import matplotlib.pyplot as plt

initial_epochs = epochs
history = hist

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([0,1.0])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

Listing A.19: Plotting training and validation accuracy and loss.

This visualization is essential for identifying patterns such as overfitting, where the model performs well on the training data but poorly on the validation data. Adjustments to the model, such as early stopping or changes in the learning rate, can be made based on insights from these plots.

Upon satisfactory training and validation results, the model is saved to disk. This allows us to preserve the state of the model after training, enabling us to reload it later for evaluation or further fine-tuning without the need to retrain from scratch.

```
# Save the fine-tuned model
model.save('efficientnet_fine_tuned.h5')

# The model can later be loaded using:
# model = tf.keras.models.load_model('efficientnet_fine_tuned.h5')
```

Listing A.20: Saving the trained model for later use.

Saving the model is straightforward in TensorFlow and Keras, and the saved model includes both the architecture and the learned weights. This step concludes the training and visualization phase, leading to the next stage of our workflow, which involves model evaluation and performance analysis on the test dataset.

Model Evaluation and Performance Analysis

The trained model's performance is quantitatively evaluated on the test dataset to ensure it generalizes well to new, unseen data. The following code demonstrates how to load the test dataset, evaluate the model's accuracy, and print the test accuracy. Furthermore, it includes the steps to visualize the predictions alongside the actual images for a qualitative assessment.

```
from tensorflow.keras.models import load_model

# Load the fine-tuned model
model = load_model('efficientnet_fine_tuned.h5')

# Evaluate on the test dataset
loss, accuracy = model.evaluate(ds_test)
print('Test accuracy:', accuracy)
```

Listing A.21: Evaluating the model's performance on the test dataset.

The above evaluation provides a straightforward accuracy metric, but for a more in-depth analysis, we plot some of the test images with their predicted labels. This visual inspection can offer insights into the types of errors the model is making and whether there are any patterns in the misclassifications.

```
# Retrieve a batch of images from the test set
image_batch, label_batch = ds_test.as_numpy_iterator().next()
predictions = model.predict_on_batch(image_batch).flatten()

# Apply a sigmoid since our model returns logits
predictions = tf.nn.sigmoid(predictions)
predictions = tf.where(predictions < 0.65, 0, 1)

```

```
print('Predictions:\n', predictions.numpy())
print('Labels:\n', label_batch)

# Plot some test images with predictions
plt.figure(figsize=(10, 10))
for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(image_batch[i].astype("uint8"))
    plt.title(class_names[predictions[i]])
    plt.axis("off")
```

Listing A.22: Displaying test images with their predicted labels.

Finally, for a comprehensive analysis, we perform a classification report which includes precision, recall, and F1-score metrics, and we visualize the confusion matrix. These metrics provide a more nuanced view of the model's performance beyond simple accuracy.

```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# Make predictions for the entire test dataset
predictions = []
labels = []
for image_batch, label_batch in ds_test.as_numpy_iterator():
    pred_batch = model.predict_on_batch(image_batch).flatten()
    pred_batch = tf.nn.sigmoid(pred_batch)
    pred_batch = tf.where(pred_batch < 0.65, 0, 1)
    predictions.extend(pred_batch.numpy())
    labels.extend(label_batch)

# Obtain recall, precision, and F1-score
print(classification_report(labels, predictions))

# Calculate the confusion matrix
conf_matrix = confusion_matrix(labels, predictions)

# Plot the confusion matrix
plt.figure(figsize=(10, 10))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap=plt.cm.Blues)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Listing A.23: Generating a classification report and visualizing the confusion matrix.

These steps complete the model's evaluation, providing both quantitative metrics and qualitative visuals to thoroughly understand the model's performance. The generated insights are crucial for determining if the model is ready for deployment or if further refinement is needed.

A.3 Anomaly Detection Model

As we ventured into training a custom YOLOv8 model for object detection tasks, our initial setup involved configuring the Google Colab environment to harness the computational power of TPUs and GPUs. The following code snippets and their respective captions illustrate the step-by-step procedure adopted in our study.

Mounting the Google Drive To access the dataset required for training, we integrated Google Drive with the Colab environment. This process is akin to connecting an external storage device to a computer, allowing for seamless data retrieval during the training process.

```
from google.colab import drive
drive.mount("/content/gdrive")
```

Listing A.24: Mounting Google Drive in Colab for data access.

Setting Up the Working Directory After establishing the connection to our data repository, we navigated to the specific directory within Google Drive that contained our dataset. This ensured that all file operations would occur in the correct location.

```
%cd /content/gdrive/MyDrive
```

Listing A.25: Navigating to the dataset directory in Google Drive.

Cloning the Official YOLOv8 Repository To utilize the latest YOLOv8 algorithms, we cloned the official Ultralytics repository. This repository provided us with the necessary codebase to execute our object detection models.

```
import os
if not os.path.isdir("yolo"):
    os.makedirs('yolo')
%cd yolo
!git clone https://github.com/ultralytics/ultralytics.git
```

Listing A.26: Cloning the YOLOv8 repository from Ultralytics.

Training the Anomaly Model With the preliminary setup complete, we began training our custom YOLOv8 model. This involved verifying the available GPU with NVIDIA's System Management Interface (SMI) and then proceeding with the installation of the YOLOv8 package and its dependencies.

```
!nvidia-smi
!pip install ultralytics
!pip install thop
```

Listing A.27: Checking GPU availability and installing YOLOv8 dependencies.

Upon successful installation, we initiated the model training procedure. Our command specified the task, mode, model, data, epochs, image size, visualization, augmentation, batch size, and patience, which are crucial hyperparameters and settings for effective model training.

```
!yolo task=detect mode=train model=yolov8s.pt data=data.yaml epochs=60 imgsz=640 ↩
    visualize=True augment=True batch=16 patience=10
```

Listing A.28: Initiating the training of the YOLOv8 model with specified parameters.

Early Stopping and Validation Throughout the training process, we monitored the model's performance and implemented early stopping to prevent overfitting. The training was halted if no improvement was observed over a defined number of epochs, ensuring that the model generalized well to new data.

```
# Early stopping was triggered based on the predefined patience parameter.
```

Listing A.29: Implementing early stopping during model training.

Model Evaluation Post-training, the model was subjected to a thorough evaluation using the validation dataset. This phase was critical to assess the model's detection capabilities and involved saving the results in a JSON format for further analysis.

```
!yolo task=detect mode=val model=runs/detect/train5/weights/best.pt data=data.yaml ↩
    save_json=True
```

Listing A.30: Evaluating the YOLOv8 model on the validation dataset.

Inference and Result Compilation The trained model was then utilized for inference on new data, demonstrating its predictive performance. The predictions were compiled into a structured format using a pandas DataFrame, which was subsequently saved to a CSV file for record-keeping and analysis.

```
!yolo task=detect mode=predict model=runs/detect/train5/weights/best.pt conf=0.3 ↩
    source=/content/gdrive/MyDrive/yolo/datasets/test/images save=True save_txt=True ↩
     save_conf=True save_crop=True
```

Listing A.31: Conducting inference with the trained model and compiling results.

In summary, the training and evaluation of the YOLOv8 model were meticulously executed, following a structured approach to optimize the model's performance for our object detection tasks.

To obtain the Confussion matrix, we used https://onlineconfusionmatrix.com/

A.4 API Integration

A.4.1 Home Page

This code snippet demonstrates the setup of a python-based web application for an "Inspection Stills Reporting API" with a focus on the home page. The code performs the following tasks:

1. Sets the page title and icon for the web application.
2. Hides the Streamlit menu and footer for a cleaner interface.
3. Displays a logo image with dynamic column width to fit the content.
4. Sets the title as "Welcome to the Inspection Stills Reporting API."
5. Adds a subheader for a video tutorial section.

The code is structured to create a welcoming and user-friendly interface for the API's home page.

```
import streamlit as st

# Set page title and icon
st.set_page_config(
    page_title="Image Classification",
    page_icon=":camera:",
)

# Hide Streamlit menu and footer
hide_streamlit_style = """
<style>
#MainMenu {visibility: hidden;}
footer {visibility: hidden;}
```

```
</style>
"""
st.markdown(hide_streamlit_style, unsafe_allow_html=True)

# Display a logo image
logo_image = "media/Logo_nobackground.png"
st.image(logo_image, use_column_width=True)

# Set the title
st.title("Welcome to the Inspection Stills Reporting API")

# Add a subheader for video tutorial
st.subheader("Video tutorial")
st.video("Inserted video tutorial")
```

Listing A.32: Streamlit setup and UI elements

The code aims to create an aesthetically pleasing and informative home page for the Inspection Stills Reporting API using Streamlit.

A.4.2 Image Processing and Timestamp Classifier

This code snippet is an image processing and timestamp classification application. It utilizes various libraries and functions to process and classify images. The main functionalities include:

1. Importing necessary libraries, including Streamlit, PyTesseract, OpenCV, PIL, and others.
2. Configuring the Tesseract OCR engine path.
3. Defining functions to modify Dropbox URLs, fetch images from Dropbox, and create a ZIP file.
4. Loading and processing image files to extract timestamps and other information.
5. Handling user interactions through a Streamlit web interface.

The code is designed to process inspection still images, extract timestamps, and provide an interface for users to interact with the processed data.

Configuration and Library Imports

This section of the code snippet initializes the configuration and imports necessary libraries for the image processing and timestamp classification application. It sets the

path for the Tesseract OCR engine and imports libraries for image manipulation, text recognition, and file handling.

```
import streamlit as st
import pytesseract
import cv2
import numpy as np
import re
from PIL import Image
from collections import defaultdict
import io
import base64
import zipfile
import requests
from io import BytesIO

# Indicate tesseract.exe directory path
pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'
```

Listing A.33: Configuration and Library Imports

Functions and Classes

This class represents a custom file-like object that extends the BytesIO class. It is used to encapsulate file content along with its name, size, and type (assumed to be 'image/jpeg' for simplicity).

```
class FileLikeObject(BytesIO):
    def __init__(self, name, content):
        super().__init__(content)
        self.name = name
        self.size = len(content)
        # Assuming all images are JPEG for simplicity; adjust as needed.
        self.type = 'image/jpeg'
```

Listing A.34: Custom FileLikeObject Class

This function fetches images from a provided Dropbox URL, processes them, and returns a list of custom FileLikeObject instances containing image data. It also updates a progress bar to show the upload progress.

```
def fetch_images_from_dropbox(url, progress_bar):
    try:
        response = requests.get(url)
        response.raise_for_status()

        with zipfile.ZipFile(BytesIO(response.content)) as zip_file:
```

```
            file_like_objects = []
            total_files = len(zip_file.namelist())
            processed = 0
            for file in zip_file.namelist():
                if file.endswith('.jpg') or file.endswith('.png'):
                    img_data = zip_file.read(file)
                    file_like_object = FileLikeObject(file, img_data)
                    file_like_objects.append(file_like_object)

                processed += 1
                progress_bar.progress(processed / total_files, text="Uploading images ↩
                    ")
            return file_like_objects
    except requests.exceptions.RequestException as e:
        st.error(f"Error fetching file: {e}")
        return []
    except zipfile.BadZipFile:
        st.error("The downloaded file is not a valid ZIP file.")
        return []
```

Listing A.35: Function to Fetch and Process Images from Dropbox

This function creates a ZIP file containing saved image data. It takes a list of tuples, where each tuple contains a file name and image data, and generates a ZIP file with the specified content.

```
def create_zip_file(saved_images_data):
    zip_buffer = io.BytesIO()
    with zipfile.ZipFile(zip_buffer, 'a', zipfile.ZIP_DEFLATED, False) as zip_file:
        for file_name, img_data in saved_images_data:
            zip_file.writestr(file_name, img_data)
    return zip_buffer.getvalue()
```

Listing A.36: Function to Create ZIP File from Saved Images

This function loads an image from its byte data and returns it as a PIL Image object. It also returns the original byte data.

```
def load_image(file_bytes):
    try:
        img = Image.open(io.BytesIO(file_bytes)).convert('RGB')
        return img, file_bytes
    except Exception as e:
        raise e
```

Listing A.37: Function to Load Image from Byte Data

Time-stamp Classifier

This section defines the main application function responsible for timestamp classification and image processing. It initializes the user interface, session state variables, and handles the user input for processing inspection stills.

```
# Initialize session state variables if they don't exist
if 'last_processed_url' not in st.session_state:
    st.session_state.last_processed_url = ''

if 'times_by_minute' not in st.session_state:
    st.session_state.times_by_minute = {}

if 'timestamps' not in st.session_state:
    st.session_state.timestamps = {}

if 'unrecognized_images' not in st.session_state:
    st.session_state.unrecognized_images = []

if 'uploaded_files' not in st.session_state:
    st.session_state.uploaded_files = []

if 'saved_images' not in st.session_state:
    st.session_state.saved_images = []

if 'image_details' not in st.session_state:
    st.session_state.image_details = []
```

Listing A.38: Initializing Session State Variables

In this part of the code, session state variables are initialized. These variables are used to store and manage application data across different user interactions. If these session state variables don't exist, they are created with default values.

```
# Dropbox URL input
dropbox_url = st.sidebar.text_input("Enter Link to Inspection Stills Folder: ")
# Initialize saved_images and image_details in session_state if they don't exist

if dropbox_url and dropbox_url != st.session_state.last_processed_url:

    st.session_state.last_processed_url = dropbox_url
    direct_url = modify_dropbox_url(dropbox_url)
    # Initialize progress bar for fetching images
    fetch_progress_bar = st.progress(0, text="Uploading images")
    st.session_state.uploaded_files = fetch_images_from_dropbox(direct_url, ↩
        fetch_progress_bar)
    fetch_progress_bar.progress(0.5, text="Recognizing text in images")
    # Process the images and retrieve the results
```

```
    st.session_state.timestamps, st.session_state.times_by_minute, st.session_state.unrecognized_images = process_files(st.session_state.uploaded_files, fetch_progress_bar)
    fetch_progress_bar.progress(1.0, text ='Completed')
else:
    st.info("At the left-hand side panel, Please enter the URL path to the folder containing the inspection stills")
```

Listing A.39: Handling Dropbox URL Input

This part of the code handles user input for the URL to the path folder images. It displays a text input field in the sidebar where the user can enter the link to the inspection stills folder. It checks if the entered URL is different from the last processed URL to avoid redundant processing. The progress bar is updated during image upload and OCR processing.

```
# Dropdown for selecting timestamp
if st.session_state.times_by_minute:
    if 'selected_minute' not in st.session_state:
        st.session_state.selected_minute = list(st.session_state.times_by_minute.keys())[0]

    # Create the selectbox and link it directly to the session state variable
    st.sidebar.selectbox("Select Time", list(st.session_state.times_by_minute.keys()), key='selected_minute')

    selected_timestamps = st.session_state.times_by_minute.get(st.session_state.selected_minute, [])
    for timestamp in selected_timestamps:
       for idx, file_bytes in enumerate(st.session_state.timestamps.get(timestamp, [])):
            img = Image.open(io.BytesIO(file_bytes))
            st.image(img)

            # Create columns for each input field
            col1,col2,col3 = st.columns(3)

            # Dropdown for Still number
            still_options = list(range(1, 51))
            still_number = col1.selectbox("Still Number", still_options, key=f"still_{timestamp}")

            # Text input for Elevation
            elevation = col2.text_input("Elevation LAT, (+/- 0.00m)", key=f"elevation_{timestamp}")

            # Save button in a new row
```

```
            if col3.button("Save", key=f"btn_save_{timestamp}_{idx}"):
                if still_number and elevation:
                    file_name = f"Still{still_number}_EL{elevation}m.png
                    st.session_state.saved_images.append((file_name, file_bytes))
                    st.session_state.image_details.append(file_name)
```

Listing A.40: Selecting Timestamps and Processing Images

A dropdown is created to allow the user to select a timestamp. It checks if there are timestamps available in the session state and initializes the selected timestamp if it doesn't exist. It displays images corresponding to the selected timestamp, provides input fields for selecting a still number and entering elevation, and allows the user to save the images. The saved images and their details are stored in session state variables.

```

if st.session_state.saved_images:
    st.sidebar.header("Saved Images")
    for image_name in st.session_state.image_details:
        st.sidebar.write(image_name)
        if st.sidebar.button("Download"):
            zip_data = create_zip_file(st.session_state.saved_images)
            b64_zip = base64.b64encode(zip_data).decode()
            st.sidebar.markdown(
            f'<a href="data:application/zip;base64,{b64_zip}" download="saved_images. ↩
                zip">Download saved images as ZIP</a>',unsafe_allow_html=True)
```

Listing A.41: Downloading Saved Images

This part of the code provides a download button for the saved images. If there are saved images available in the session state, it displays them in the sidebar and allows the user to download them as a ZIP file.

A.4.3 General Classifier Model and Anomaly detection

Imported Packages

This section lists the packages and libraries imported in the code. These packages provide essential functionality for various tasks within the application.

```
import streamlit as st
from PIL import Image
import io
from io import BytesIO
import base64
import zipfile
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.efficientnet import preprocess_input
from tensorflow.keras.applications.efficientnet import decode_predictions
import tensorflow as tf
from ultralytics import YOLO
from PIL import Image, ImageDraw
import numpy as np
import cv2
```

Listing A.42: Imported Packages

Functions

This section defines several functions used in the application. These functions perform various tasks such as image preprocessing, creating ZIP files, and decoding predictions.

```
def preprocess_image(img, target_size):
    img = img.resize(target_size)
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    return preprocess_input(img_array)
```

Listing A.43: Image Pre-processing for GCM

This function preprocesses the image to feed into the GCM ("EfficientNetB0") by resizing it to a specified target size, converting it to an array, and expanding the dimensions to match the model input shape.

```
def create_zip_file(saved_images_data):
    zip_buffer = io.BytesIO()
    with zipfile.ZipFile(zip_buffer, 'a', zipfile.ZIP_DEFLATED, False) as zip_file:
        for file_name, img_data in saved_images_data:
            zip_file.writestr(file_name, img_data)
    return zip_buffer.getvalue()
```

Listing A.44: ZIP image files function

This function creates a ZIP file containing saved images and their data.

```
class_names = ['Circumferental Weld', 'non-CW']
```

Listing A.45: List with model's labels

This variable holds a list of class names used for decoding predictions. In this case, it represents two classes: 'Circumferential Weld' and 'non-CW'.

```
def decode_predictions(preds, top=1):
    results = []
```

```
    for pred in preds:
        top_indices = pred.argsort()[-top:][::-1]
        result = [class_names[i] for i in top_indices]
        results.append(result)
    return results
```

Listing A.46: Decode GCM predictions

This function decodes the predictions generated by a model and returns the top class names.

Loading models

```
# Load the GCM
model_path = 'models/model.h5' # Update this to the path of your model
model_effnet = load_model(model_path)

# Load the ADM
model_path = 'models/best.pt'
model_yolo = YOLO(model_path)
```

Listing A.47: Upload Deep Learning models

Both models, the GCM and the yolov8 are uploaded.

```
# Check if the variable exists in the session state before accessing it
if 'gemini_details' not in st.session_state:
    st.session_state.gemini_details = []


if 'gemini_images' not in st.session_state:
    st.session_state.gemini_images = []
```

Listing A.48: Declare session state variables

Checks there are existing images in the session state variables. If not, they are initialized as empty lists. This is likely for storing and managing user session data like image details and selections.

```
if 'saved_images' in st.session_state:
    for idx, (image_name, image_binary) in enumerate(st.session_state.saved_images):
        st.write(f"Image {idx+1}: {image_name}")

        image_bytes_io = BytesIO(image_binary)
        image = Image.open(image_bytes_io)

        # Checkbox to enable/disable anomaly detection
```

```
9         detect_anomalies = st.checkbox(f"Detect Anomalies for Image {idx+1}", key=f" ↩
              anomaly_detection_{idx}")
10
11        if detect_anomalies:
12            # Perform YOLOv8 inference
13            yolo_results = model_yolo(image)
14            draw = ImageDraw.Draw(image)
15
16            for result in yolo_results:
17                boxes = result.boxes
18                for box in boxes.xyxy:
19                    x1, y1, x2, y2 = box.tolist()
20                    draw.rectangle([x1, y1, x2, y2], outline="red", width=2)
21
22        # Display the image with bounding boxes
23        st.image(image, use_column_width=True)
24
25        # Preprocess the image and predict
26        processed_image = preprocess_image(image, target_size=(224, 224)) #  ↩
              EfficientNetB0 default size
27        predictions = model_effnet.predict(processed_image)[0] # Assuming single  ↩
              image prediction
28        probability = tf.nn.sigmoid(predictions[0]) # If the model outputs a single  ↩
              logit
29        col1, col2 = st.columns(2)
30
31        with col1:
32            st.info(f"Probability of '{class_names[1]}': {probability * 100:.2f}%")
33        with col2:
34            st.info(f"Probability of '{class_names[0]}': {(1 - probability) * 100:.2f ↩
                  }%")
```

Listing A.49: Predictions

This code displays saved images from the session state. Each image is given an option to detect anomalies using the YOLO model. If selected, the model detects objects in the image, and their bounding boxes are drawn. Each image is also processed and fed into the GCM for classification. The probability of the image belonging to a particular class (e.g., 'Circumferential Weld' or 'non-CW') is displayed.

```
1 col3, col4, col5, col6 = st.columns(4)
2
3         feature_options = ['GeneralView', 'SideWall', 'CW', 'SeamWeld', 'Termination' ↩
              , 'RetentionBars', 'Splashzone', 'MechanicalConnection', 'SwageLiner', ' ↩
              Ext Wrap', 'Other','select_option']
4         default_feature = 'CW' if (1 - probability) > probability else 'select_option ↩
              '
```

```
        default_index = feature_options.index(default_feature)
        feature = col3.selectbox("Feature", feature_options, index=default_index, key =f"feature_{idx}")
        # feature = col1.selectbox("Feature", feature_options, key=f"feature_{idx}")

        anomaly_options = ['WallLoss', 'CoatingLoss', 'Pitting', 'Debris', 'Corrosion', 'Crack', 'Thru-wall', 'Weldroot', 'Abrasion', 'Fretting', 'MIC', 'MechanicalDamage', 'GoodCondition', 'Other']
        anomaly = col4.selectbox("Description", anomaly_options, key=f"anomaly_{idx}")

        severity_options = ['High', 'Moderate', 'Low', "NA"]
        severity = col5.selectbox("Severity", severity_options, key=f"severity_{idx}")

        # Save button in a new row
        if col6.button("Save", key=f"btn_save_{image_name}{idx}"):
            if feature and anomaly and severity:
                file_name = f"{image_name[:-4]}_{feature}_{anomaly}_{severity}.png"
                st.session_state.gemini_images.append((file_name, image_binary))
                st.session_state.gemini_details.append(file_name)
```

Listing A.50: Image Annotations

This code, provides options for the user to select features, descriptions, and severity of any detected issues or anomalies in the image. There is a save button for each image to save these annotations.

```
# Download button for the saved images
if st.session_state.gemini_images:
    st.sidebar.header("Gemini Images")
    for image_name in st.session_state.gemini_details:
        st.sidebar.write(image_name)
    if st.sidebar.button("Download"):
        zip_data = create_zip_file(st.session_state.gemini_images)
        b64_zip = base64.b64encode(zip_data).decode()
        st.sidebar.markdown(
            f'<a href="data:application/zip;base64,{b64_zip}" download="gemini_images.zip">Download gemini images as ZIP</a>',
            unsafe_allow_html=True)
```

Listing A.51: Saving and Downloading Annotated Images

If the save button is clicked, the annotated image details are added to the session state. There is also an option to download all saved and annotated images as a ZIP file.