UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Boris Gašparović

Automatic Pipeline Inspection Using Deep-Learning-Based Computer Vision

DOCTORAL THESIS

Rijeka 2026.

UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Boris Gašparović

Automatic Pipeline Inspection Using Deep-Learning-Based Computer Vision

DOCTORAL THESIS

Supervisor: Prof. Jonatan Lerga, PhD
Co-supervisor: Prof. Goran Mauša, PhD

Rijeka 2026.

SVEUÄŇILIÅăTE U RIJECI
TEHNIÄŇKI FAKULTET

Boris Gašparović

Automatska inspekcija cjevovoda korištenjem računalnog vida temeljenog na dubokom učenju

DOKTORSKI RAD

Mentor: prof. dr. sc. Jonatan Lerga
Komentor: prof. dr. sc. Goran Mauša

Rijeka 2026.

Doctoral thesis supervisor: Prof. Jonatan Lerga, PhD
Doctoral thesis co-supervisor: Prof. Goran Mauša, PhD

The doctoral thesis was defended on _____ at the University of Rijeka, Faculty of Engineering, Croatia, in front of the following Evaluation Committee:

1. - Committee Chair
2.
3.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Jonatan Lerga, and my co-supervisor, Prof. Goran Mauša, for their guidance, patience, and continued support throughout this research.

I thank my colleagues at the Faculty of Engineering and the Center for Artificial Intelligence and Cybersecurity, University of Rijeka, for the collaborative environment in which this work was carried out, and my co-authors Prof. Marina Ivašić-Kos and Josip Rukavina for their contributions to the research on which several chapters of this dissertation are based.

I am grateful to Vectrino d.o.o. for providing the pipeline inspection recordings on which this work is based. Without access to real inspection footage, neither the datasets nor the experiments presented here would have been possible.

This research was supported by the European Union Horizon 2020 project INNO2MARE, "Strengthening the capacity for excellence of Slovenian and Croatian innovation ecosystems to support the digital and green transitions of maritime regions" (grant agreement no. 101087348); and by the University of Rijeka projects uniri-iskusni-tehnic-23-11 and uniri-iskusni-tehnic-23-83.

Finally, I thank my family and friends for their patience and encouragement throughout my doctoral studies.

ABSTRACT

Pipeline infrastructure represents a critical component of modern urban, industrial, and environmental systems, requiring regular inspection to ensure operational reliability, public safety, and environmental protection. Traditional inspection procedures predominantly rely on manual analysis of video recordings acquired by closed-circuit television systems, remotely operated vehicles, and other inspection platforms. Such approaches are time-consuming, costly, and susceptible to subjective interpretation. Consequently, there is an increasing demand for automated inspection systems capable of accurately detecting and classifying structural defects in pipeline networks. This dissertation focuses on the development of advanced deep-learning-based computer vision methods for the automatic inspection of sewer, drainage, industrial, and underwater pipelines. The research encompasses the design of standardized datasets, defect annotation procedures based on the EN 13508-2 standard, image and video processing pipelines, object detection model optimization, and the development of efficient real-time inspection frameworks. Particular emphasis is placed on addressing challenges commonly encountered in practical inspection scenarios, including class imbalance, computational constraints, and the exploitation of temporal information available in inspection videos.

The proposed methodology investigates and compares multiple generations of YOLO object detection architectures and introduces novel optimization strategies aimed at improving the detection of rare and visually challenging defect classes. Various dataset balancing and augmentation techniques, including geometric and photometric transformations, MixUp, Mosaic, and Copy-Paste, are analyzed to enhance model generalization and robustness. The dissertation proposes a cascade YOLO framework in which a lightweight detector performs preliminary defect screening, while a more complex detector is selectively applied to potentially defective images. Experimental results demonstrate

that the proposed cascade approach substantially reduces computational requirements while maintaining detection accuracy comparable to conventional single-stage detectors.

To further improve inspection reliability, a dual-stream spatiotemporal framework is developed for video-based defect detection. The proposed architecture combines a YOLOv7V-based temporal aggregation stream with a PTSEFormer transformer-based temporal reasoning stream, enabling the integration of spatial and temporal information extracted from inspection videos. The resulting framework improves detection consistency, reduces false-positive detections, and increases robustness in challenging conditions characterized by motion blur, illumination variations, occlusions, and image degradation.

Experimental evaluation conducted on underwater and sewer pipeline datasets demonstrates that the proposed methods improve defect detection performance, particularly for underrepresented defect categories. The cascade framework reaches approximately 96 frames per second on the tested GPU platform for survey footage of the composition it is intended for, in which most frames contain no defect; its saving scales with that proportion, and Chapter 6. reports throughput across the range rather than as a single figure. Quantization to reduced numerical precision, combined with export to embedded-capable inference backends, further supports the framework's suitability for deployment on resource-constrained inspection hardware. The dual-stream framework reaches 18 frames per second, since its transformer-based stream is invoked selectively rather than on every frame. The obtained results confirm the effectiveness of combining advanced object detection architectures, class imbalance mitigation strategies, cascade processing, and spatiotemporal analysis for automated pipeline inspection, providing a basis for deployment in industrial and municipal inspection environments.

Keywords: Pipeline inspection, computer vision, object detection, class imbalance mitigation, video object detection, spatiotemporal analysis

PROŠIRENI SAŽETAK

Cjevovodna infrastruktura predstavlja ključnu komponentu suvremenih urbanih, industrijskih i okolišnih sustava te zahtijeva redovite inspekcije kako bi se osigurala operativna pouzdanost, sigurnost ljudi i zaštita okoliša. Tradicionalni postupci inspekcije uglavnom se temelje na ručnoj analizi videosnimki prikupljenih sustavima zatvorene televizije, daljinski upravljanim vozilima i drugim inspekcijskim platformama. Takvi pristupi zahtijevaju mnogo vremena i financijskih resursa te su podložni subjektivnim procjenama inspektora. Zbog toga postoji sve veća potreba za automatiziranim sustavima inspekcije koji mogu precizno detektirati i klasificirati strukturne nedostatke unutar cjevovodnih mreža.

Ova disertacija usmjerena je na razvoj naprednih metoda računalnog vida temeljenih na dubokom učenju za automatiziranu inspekciju kanalizacijskih, oborinskih, industrijskih i podvodnih cjevovoda. Istraživanje obuhvaća izradu standardiziranih skupova podataka, postupke anotacije označavanja temeljene na normi EN 13508-2, obradu slika i videozapisa, optimizaciju modela za detekciju objekata te razvoj učinkovitih okvira za inspekciju u stvarnom vremenu. Poseban naglasak stavljen je na rješavanje izazova koji se često pojavljuju u stvarnim inspekcijskim uvjetima, uključujući neuravnoteženost klasa, ograničenja računalnih resursa te iskorištavanje vremenskih informacija sadržanih u inspekcijskim videozapisima.

Predložena metodologija istražuje i uspoređuje više generacija YOLO arhitektura za detekciju objekata te uvodi nove strategije optimizacije usmjerene na poboljšanje detekcije rijetkih i vizualno zahtjevnih klasa označavanja. Analizirane su različite tehnike balansiranja skupova podataka i augmentacije, uključujući geometrijske i fotometrijske transformacije, MixUp, Mosaic i Copy-Paste, s ciljem povećanja sposobnosti generalizacije i robusnosti modela. Nadalje, u disertaciji se predlaže kaskadni YOLO ok-

vir u kojem lagani detektor provodi poÄĐetno prepoznavanje moguÄĞih oÅąteÄĞenja, dok se sloÅ¿eniji detektor selektivno primjenjuje samo na potencijalno oÅąteÄĞene slike. Eksperimentalni rezultati pokazuju da predloÅ¿eni kaskadni pristup znaÄĐajno smanjuje raÄĐunalne zahtjeve uz zadrÅ¿avanje toÄĐnosti detekcije usporedive s konvencionalnim jednostupanjskim detektorima.

Kako bi se dodatno poveÄĞala pouzdanost inspekcije, razvijen je dvostruki prostorno-vremenski (spatiotemporalni) okvir za detekciju oÅąteÄĞenja temeljem videozapisa. PredloÅ¿ena arhitektura kombinira vremenski agregacijski tok temeljen na modelu YOLOv7V s transformatorom PTSEFormer za vremensko zakljuÄĐivanje, omoguÄĞujuÄĞi integraciju prostornih i vremenskih informacija izdvojenih iz inspekcijskih videozapisa. RezultirajuÄĞi okvir poboljÅąava dosljednost detekcije, smanjuje broj laÅ¿no pozitivnih detekcija te poveÄĞava robusnost sustava u zahtjevnim uvjetima obilježenima zamuÄĞenjem uzrokovanim kretanjem, promjenama osvjetljenja, zaklonjenim objektima i degradacijom slike.

Sveobuhvatna eksperimentalna evaluacija provedena na skupovima podataka podvodnih i kanalizacijskih cjevovoda pokazuje da predloÅ¿ene metode poboljÅąavaju uspjeÅąnost detekcije oÅąteÄĞenja, osobito kod nedovoljno zastupljenih kategorija oÅąteÄĞenja. Kaskadni okvir postiÅ¿e pribliÅ¿no 96 sliÄĐica u sekundi na ispitanoj GPU platformi za inspekcijske snimke onakvog sastava za kakav je i namijenjen, u kojima veÄĞina sliÄĐica ne sadrÅ¿i oÅąteÄĞenje; njegova uÅąteda razmjerna je tom udjelu, pa se u poglavlju 6. propusnost iskazuje kroz cijeli raspon, a ne kao jedna vrijednost. Kvantizacija na smanjenu numeriÄĐku preciznost, zajedno s izvozom u zakljuÄĐne (inference) okvire prilagoÄŚene ugradbenim sustavima, dodatno podupire prikladnost okvira za primjenu na raÄĐunalno ograniÄĐenoj inspekcijskoj opremi. Dvostruki okvir postiÅ¿e 18 sliÄĐica u sekundi, buduÄĞi da se njegov transformatorski tok aktivira selektivno, a ne za svaku sliÄĐicu. Dobiveni rezultati potvrÄŚuju uÄĐinkovitost kombiniranja naprednih arhitektura za detekciju objekata, strategija ublaÅ¿avanja neuravnoteÅ¿enosti klasa, kaskadne obrade i prostorno-vremenske analize u automatiziranoj inspekciji cjevovoda te predstavljaju osnovu za primjenu u industrijskim i komunalnim inspekcijskim okruÅ¿enjima. **KljuÄĐne rijeÄĐi: Inspekcija cjevovoda, raÄĐunalni vid, detekcija objekata, ublaÅ¿avanje neuravnoteÅ¿enosti klasa, detekcija objekata**

u videozapisima, prostorno-vremenska analiza.

CONTENTS

Acknowledgements **I**

Abstract **III**

ProÅąireni saÅ¿etak **V**

1. Introduction **1**

1.1. Pipeline Infrastructure and Inspection Requirements 2

1.2. Conventional Pipeline Inspection Methods 3

1.3. Artificial Intelligence in Infrastructure Inspection 5

1.4. Deep Learning for Pipeline Defect Detection 6

1.5. Research Methodology 8

1.6. Structure of the Doctoral Dissertation 9

2. Review of existing studies **13**

2.1. Pipeline Inspection and Defect Assessment 13

2.2. Artificial Intelligence and Computer Vision for Infrastructure Inspection . . 15

2.2.1. Attention Mechanisms for Defect Detection 16

2.2.2. Segmentation and Transformer Architectures 17

2.3. Challenges in Deep Learning-Based Pipeline Inspection 18

2.4. Class Imbalance and Data Augmentation in Object Detection 19

2.5. Video-Based Defect Detection and Temporal Analysis 20

2.6. Cascade Detection Systems and Model Optimization 23

2.7. Summary and Research Gap 25

3. Datasets and Experimental Setup **29**

3.1. Introduction . . . 29

3.2. Dataset Overview . . . 30

3.3. Underwater Pipeline Dataset . . . 30

3.3.1. Data Acquisition . . . 30

3.3.2. Dataset Annotation . . . 32

3.3.3. Environmental Challenges . . . 32

3.3.4. Dataset Split . . . 32

3.4. EN 13508-2 Standard . . . 33

3.5. Sewer Inspection Dataset . . . 34

3.5.1. Dataset Development . . . 34

3.5.2. Annotation and Preprocessing . . . 34

3.6. Video Sequence Dataset for Spatiotemporal Detection . . . 36

3.7. Experimental Setup . . . 39

3.8. Chapter Summary . . . 41

4. Deep Learning Models for Pipeline Defect Detection **43**

4.1. Object Detection Architectures . . . 44

4.2. YOLO Architecture Overview . . . 45

4.2.1. YOLOv4 . . . 45

4.2.2. YOLOv5 . . . 45

4.2.3. YOLOv6 . . . 45

4.2.4. YOLOv7 . . . 46

4.2.5. YOLOv8 . . . 46

4.2.6. YOLOv9 . . . 46

4.2.7. YOLOv10 . . . 47

4.2.8. YOLOv11 . . . 47

4.2.9. YOLO26 . . . 47

4.2.10. Other YOLO-Based Architectures . . . 49

4.3. Detection pipeline . . . 50

4.3.1. Image Preprocessing . . . 50

4.3.2. Feature Extraction (Backbone) . . . 51

4.3.3. Feature Aggregation (Neck) . . . 52
4.3.4. Detection Head . . . 52
4.3.5. Post-Processing . . . 53
4.4. Attention Mechanisms . . . 55
4.4.1. Squeeze-and-Excitation Network (SE) . . . 56
4.4.2. Convolutional Block Attention Module (CBAM) . . . 57
4.4.3. Efficient Channel Attention (ECA) . . . 58
4.5. Loss Functions . . . 60
4.5.1. Intersection over Union (IoU) . . . 60
4.5.2. Generalized Intersection over Union (GIoU) . . . 62
4.5.3. Complete Intersection over Union (CIoU) . . . 63
4.5.4. Scylla Intersection over Union (SIoU) . . . 64
4.5.5. Combined Detection Loss . . . 65
4.6. Chapter Summary . . . 67

5. Class Imbalance Mitigation for Pipeline Inspection 69
5.1. Class Imbalance in Pipeline Inspection Datasets . . . 69
5.1.1. Distribution of Defect Classes . . . 70
5.1.2. Impact of Class Imbalance on Object Detection . . . 71
5.1.3. Challenges Associated with Rare Defects . . . 72
5.2. Data Augmentation Techniques . . . 73
5.2.1. Geometric Transformations . . . 74
5.2.2. Photometric Transformations . . . 74
5.2.3. MixUp Augmentation . . . 76
5.2.4. Mosaic Augmentation . . . 77
5.2.5. Copy-Paste Augmentation . . . 78
5.2.6. Synthetic Minority Over-sampling Technique (SMOTE) . . . 78
5.3. Dataset Balancing Strategies . . . 82
5.3.1. Oversampling Approaches . . . 84
5.3.2. Undersampling Approaches . . . 85
5.3.3. Hybrid Balancing Methods . . . 85
5.3.4. Class-Aware Sampling . . . 86

5.4. Loss Function Adaptations for Imbalanced Detection 87
5.4.1. Weighted Loss Functions 88
5.4.2. Focal Loss 89
5.4.3. Class-Balanced Loss 89
5.4.4. Comparative Analysis 90
5.5. Chapter Summary 90

6. Cascade YOLO Framework for Efficient Pipeline Inspection 93
6.1. Introduction 93
6.2. Motivation for Cascade Detection 94
6.3. Architecture of the Proposed Cascade Framework 94
6.3.1. Datasets Used for the Two Stages 95
6.3.2. Stage 1: Binary Defect Detection 96
6.3.3. Stage 2: Multi-Class Defect Detection 100
6.4. Cascaded Inference Workflow 102
6.5. Evaluation Protocol 103
6.6. Computational Complexity Analysis 104
6.6.1. Worked Example Using the Measured Operating Point 105
6.6.2. The Pass-Through Rate Is a Property of the Footage 106
6.7. Selection of Cascade Design Parameters 108
6.8. Failure Modes and Deployment Considerations 109
6.9. Chapter Summary 110

7. Dual-Stream Spatiotemporal Framework for Pipeline Defect Detection 111
7.1. Introduction 111
7.2. Limitations of Frame-Based Detection 112
7.3. Spatiotemporal Object Detection 113
7.3.1. Temporal Aggregation Module 113
7.3.2. Frame Sequence Construction 115
7.3.3. Spatial-Temporal Feature Fusion 116
7.4. YOLOv7V: Architecture and Implementation 116
7.4.1. Network Configuration 116
7.4.2. Rationale for the YOLOv7 Backbone 117

7.4.3. Training Configuration . . . 118
7.4.4. Limits of Convolutional Temporal Aggregation . . . 118
7.5. PTSEFormer-Based Temporal Reasoning Stream . . . 119
7.5.1. Temporal Feature Aggregation Module . . . 120
7.5.2. Spatial Transition Awareness Module . . . 121
7.5.3. Query Assembling Module . . . 122
7.5.4. Advantages of Transformer-Based Temporal Reasoning . . . 122
7.6. Proposed Dual-Stream Fusion Framework . . . 123
7.6.1. Framework Architecture . . . 124
7.6.2. Late Fusion Strategy . . . 124
7.6.3. Selective Invocation of the Transformer Stream . . . 124
7.6.4. Confidence-Based Decision Fusion . . . 125
7.6.5. Detection Workflow . . . 126
7.6.6. Implementation and Training . . . 127
7.6.7. Computational Cost of the Dual-Stream Design . . . 128
7.6.8. Evaluation Protocol for the Dual-Stream Framework . . . 129
7.6.9. Design Limitations . . . 130
7.7. Chapter Summary . . . 130

8. Experimental Results and Discussion 133
8.1. Introduction . . . 133
8.2. Evaluation of YOLO-Based Object Detection Architectures . . . 134
8.2.1. Performance Comparison of YOLO Architectures . . . 134
8.2.2. Underwater Pipeline Detection Results . . . 135
8.2.3. Sewer Defect Detection Results . . . 139
8.2.4. Modified Mosaic Augmentation and the YOLOv8–YOLOv11 Generations . . . 142
8.2.5. YOLO26 Compared with YOLOv11 . . . 144
8.2.6. Lightweight Architectural Modifications of YOLOv7 . . . 146
8.2.7. Capacity Additions to YOLOv7 . . . 148
8.2.8. Box-Regression Loss Variants . . . 149
8.2.9. Attention Modules . . . 150

8.2.10. Impact of Contextual Background Information 152
8.2.11. Impact of On-Screen Overlay Masking 153
8.2.12. Analysis of Detection Accuracy and Computational Efficiency . . . 154
8.3. Impact of Class Imbalance Mitigation 155
8.3.1. Effect of Data Augmentation Techniques 155
8.3.2. Evaluation of Dataset Balancing Strategies 158
8.3.3. Dataset Filtering and Negative Sampling 159
8.3.4. Evaluation of Loss Function Adaptations 160
8.3.5. Discussion of Rare Defect Detection Performance 164
8.4. Evaluation of the Cascade Detection Framework 166
8.4.1. Performance of the Binary Defect Detection Stage 166
8.4.2. Stage-2 Detection Performance 169
8.4.3. Comparison Between Single-Stage and Cascade Detection 170
8.4.4. Selection of the Gate Threshold 172
8.4.5. Dynamic Confidence, Score Fusion, and Tiling 175
8.4.6. Effect of Inference Precision (Quantization) 175
8.4.7. Computational Efficiency 176
8.5. Evaluation of Spatiotemporal Detection Approaches 177
8.5.1. Frame-Based Versus Video-Based Detection 177
8.5.2. Comparison of Individual Streams 178
8.6. Evaluation of the Proposed Dual-Stream Fusion Framework 179
8.6.1. Performance of the Dual-Stream Framework 179
8.6.2. Comparison with Individual Detection Streams 180
8.6.3. Comparison Against Newer Single-Frame YOLO Generations 181
8.6.4. Detection of Rare and Ambiguous Defects 182
8.6.5. Ablation: Temporal Alignment and Overlay Masking 183
8.6.6. Qualitative Detection Examples 184
8.7. Comparison Across Inspection Domains 185
8.8. Validation of Research Hypotheses 185
8.8.1. Hypothesis H1 185
8.8.2. Hypothesis H2 186
8.8.3. Hypothesis H3 186

8.8.4. Hypothesis H4 187
8.9. Summary of Results 187

9. Conclusion **189**
9.1. Summary of the Research 189
9.2. Validation of the Research Hypotheses 190
9.3. Scientific Contributions 191
9.4. Limitations 192
9.5. Future Work 192
9.6. Closing Remarks 194

Bibliography **197**

List of Figures **225**

List of Tables **229**

List of Abbreviations **235**

Appendix **239**

Curriculum Vitae **241**

List of Publications **243**

1. Chapter

INTRODUCTION

Pipeline networks carry water, wastewater, oil, and gas over distances that make manual, point-by-point inspection impractical [1]. They are also among the least visible parts of modern infrastructure. Because the pipes are buried or submerged, their condition can be established only through recorded imagery, sensor readings, or occasional direct access. This dissertation is concerned with the first of these, and specifically with how the recorded imagery is interpreted. Deterioration in such networks is gradual and largely silent. Corrosion, cracking, deformation, joint displacement, and material degradation accumulate over years until a failure draws attention, and by then the cost extends well beyond the repair itself to service disruption, environmental contamination, and, in water networks, to water that never reaches the consumer [1, 2, 3].

Pipeline inspection suits computer vision for two reasons. The data is already visual, since CCTV crawlers survey sewers and remotely operated vehicles (ROVs) survey submerged pipelines. A vision system therefore works from material that standard inspection practice produces anyway, and no new sensing equipment is required. The interpretation of that material is also the point at which the process slows down. A technician must review hours of footage frame by frame, which is tiring, time-consuming, and inconsistent between reviewers [4, 5, 6]. Deep-learning object detectors, and the YOLO (You Only Look Once) family in particular, can automate this step while keeping the balance between accuracy and speed that inspection in real time demands [7, 8, 9].

Transferring such detectors from benchmarks like MS COCO to pipeline footage is not straightforward, for three reasons. The first is structural class imbalance: deposits and

minor misalignments dominate any inspection dataset while breaks and structural cracks, which matter most for maintenance, appear rarely, so a model trained without correction learns the common conditions well and the critical ones poorly [10, 11]. The second is image quality – light attenuation, colour distortion, and backscatter underwater; uneven illumination, flowing water, sediment, and burned-in telemetry in sewers. The third is that inspection produces video, yet most pipelines process each frame independently and discard the temporal continuity a human inspector relies on when a single frame is ambiguous [12, 13, 14].

These three problems, class imbalance, environmental degradation, and frame independence, define the scope of this dissertation. They are addressed through modified YOLO architectures, a cascaded detection framework, and a dual-stream spatiotemporal fusion model, evaluated across two inspection domains, underwater pipeline and sewer CCTV footage, in three experimental dataset configurations.

One element of the approved research proposal falls outside that scope. The proposal framed the aim as identifying *and tracking the progression of* structural damage; this dissertation addresses detection and localisation only. Tracking damage progression requires repeat inspections of the same asset registered to a common spatial reference, and the available material consists of single-pass campaigns with no repeat coverage, so the question could not be posed against this data. Where tracking appears in what follows it is multi-object tracking within a single inspection pass, which is contrasted against the dual-stream design in Chapter 7. and is not adopted; longitudinal progression monitoring is identified as future work in Chapter 9.. The rest of this chapter describes pipeline infrastructure and what its inspection requires, reviews conventional inspection methods and where they fall short, introduces the role of deep learning in this field, and states the research hypotheses, contributions, methodology, and structure of the work.

1.1. Pipeline Infrastructure and Inspection Requirements

Water, wastewater, oil, and gas networks in developed countries typically extend for tens of thousands of kilometres, and offshore oil, gas, and desalination installations ac-

count for a comparable length of subsea pipeline [1, 3]. Much of this infrastructure was laid decades ago and now approaches or exceeds its design lifetime. Operators therefore manage risk mainly through repeated condition assessment over the service life of the asset, and no longer through inspection carried out once at the construction stage [1, 3].

The costs of undetected deterioration are asymmetric. A missed defect is far more expensive than an unnecessary inspection. Undetected leaks in water distribution networks compound over years into substantial non-revenue water losses, with consequences for water scarcity, public health, and the finances of the utility [15, 16]. Structural failures in sewer and drainage systems can cause flooding, groundwater contamination, and road subsidence, and failures in oil and gas transmission carry further environmental and safety risks [1]. This asymmetry is the argument for condition-based maintenance, in which inspection is used to find defects before they propagate instead of confirming damage after the fact. It is also why inspection capacity needs to scale with the size of the network and not with the number of trained inspectors available to review footage [3, 6, 17].

1.2. Conventional Pipeline Inspection Methods

Pipeline condition assessment has historically relied on a mix of direct visual inspection and non-destructive testing (NDT), each suited to a different stage of deterioration and a different degree of access.

Visual inspection remains the most widely deployed method. CCTV crawlers are pushed or driven through sewers and drains, and ROVs equipped with cameras survey submerged and offshore pipelines from several viewing angles [4, 5, 6]. Both platforms produce continuous video, which a trained technician then reviews and annotates with defect locations, types, and severity codes.

Other sensing techniques cover defects not reliably visible on the surface: acoustic sensors localise leaks from flow noise, pressure testing identifies sections outside their expected hydraulic range, and water-quality sampling flags contamination indicating an undetected breach. Where wall integrity itself is in question, non-destructive testing is applied directly – ultrasonic testing for wall thickness and internal flaws [18], infrared thermography for subsurface anomalies inferred from surface temperature gradients [19], and radiographic testing for composite and multi-layer walls [20]. Destructive methods

remain available where these are inconclusive, but take the pipeline out of service.

Robotic platforms such as instrumented "smart pigs" extend these techniques to longer stretches of pipe by carrying sensor payloads through the line under its own flow or under tow, combining several of the above modalities (typically visual, magnetic, and ultrasonic) with multivariate statistical analysis of the resulting signal to flag anomalies for follow-up [21]. Figure 1.1 summarises the resulting workflow common to CCTV- and ROV-based inspection: footage acquisition, manual or semi-manual review, defect coding, and reporting.

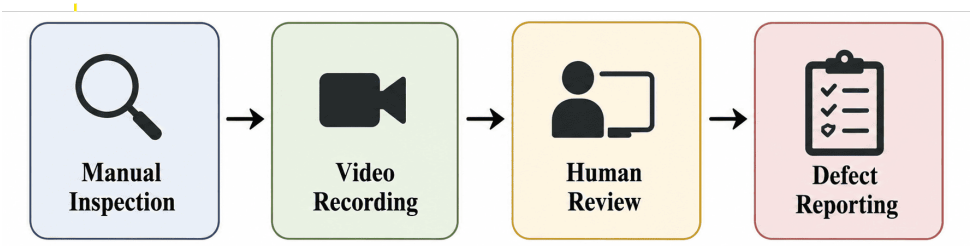


Figure 1.1: General workflow of conventional CCTV- and ROV-based pipeline inspection.

What limits these methods is not sensing capability but the rate at which their output can be interpreted. Robotic and remote platforms have largely solved the problem of reaching inaccessible pipe sections. The labour now sits in the step that follows: watching the recorded footage and making a judgement on every frame. A single inspection campaign can produce thousands of hours of video, and reviewing that volume by hand is slow, tiring, and inconsistent, since the outcome depends on the experience and attentiveness of the individual inspector [4, 6].

The footage itself makes the task harder. Figure 1.2 shows representative frames from both domains. Images recorded inside pipelines often suffer from poor illumination, motion blur, low contrast, and sediment obscuring the wall (Figure 1.2a). Underwater footage adds turbidity, light absorption, colour distortion, and suspended particles (Figure 1.2b) [22, 23]. Human attention degrades fastest under conditions of this kind, and they are the conditions under which the detection models in this dissertation are trained and evaluated.



(a) Sewer CCTV **(b)** Underwater ROV

Figure 1.2: Representative inspection imagery from the two datasets used in this dissertation: (a) sewer pipeline frames from the CCTV inspection dataset, showing uneven illumination, sediment and flowing water; (b) underwater pipeline frames from the PipeVision dataset, showing turbidity, light attenuation and colour distortion.

1.3. Artificial Intelligence in Infrastructure Inspection

The interpretation bottleneck described above – large volumes of inspection footage, reviewed manually and inconsistently – is the kind of problem that has been addressed elsewhere in infrastructure monitoring through machine learning, and increasingly through deep learning [24, 25, 26]. Instead of relying on hand-engineered features and rule-based defect criteria, convolutional neural networks (CNNs) learn discriminative visual representations directly from labelled examples, which has allowed them to generalise across the considerable variability in lighting, viewing angle, and defect appearance that rule-based systems struggle with [25, 26].

Within civil and industrial infrastructure, this shift is visible across several domains: crack and spalling detection in bridge decks, pothole and surface-crack detection in road pavements, rail and switch inspection, and faÃğade condition monitoring [17, 27]. Sewer and water pipeline inspection is a natural addition to this list, since CCTV and ROV surveys already generate the large, continuously growing image and video archives that deep learning models require for training [4, 5, 6].

The specific architectural family this dissertation builds on is object detection, which, unlike image classification, both localises and classifies defects within a frame, producing output (a bounding box and a class label) that maps directly onto the defect-coding workflow inspectors already use. Within object detection, single-stage architectures such as SSD and the YOLO family trade a small amount of accuracy for a large gain in

inference speed relative to two-stage detectors such as Faster R-CNN, which makes them the more practical choice for continuous video streams and for eventual deployment on the constrained hardware carried by inspection robots and ROVs [7, 8, 9, 28, 29].

Two open problems limit how directly this progress transfers to pipeline inspection specifically. The first is class imbalance: inspection datasets are dominated by common, low-severity observations, while the rare defects that matter most for maintenance decisions are the ones a model trained on raw class frequencies learns worst [10]. The second is the video-versus-image mismatch: pipeline inspection data is inherently sequential, yet most detection systems, including the baseline YOLO architectures evaluated in this dissertation, process each frame independently and discard the temporal continuity that would otherwise help confirm an uncertain detection or suppress a spurious one [12, 13, 14, 30]. Both problems are addressed directly in the later chapters of this dissertation, through data-level and loss-level interventions for imbalance (Chapter 5.) and through a dual-stream spatiotemporal model for the video-frame mismatch (Chapter 7.).

1.4. Deep Learning for Pipeline Defect Detection

Taken together, the preceding sections establish that the components required for automated pipeline inspection already exist individually – CCTV and ROV platforms that generate suitable imagery, and YOLO-family detectors capable of real-time localisation and classification – but that combining them for pipeline defect detection specifically surfaces three problems that are under-addressed in the existing literature.

The first is architectural: the YOLO family has progressed rapidly from YOLOv4 to YOLOv11 and YOLO26, each version shifting the accuracy–speed–resource trade-off, yet few studies compare these versions on pipeline inspection imagery instead of on general-purpose benchmarks. The second is computational: the most accurate models are too large for the hardware carried by inspection robots and ROVs, while lightweight models lose accuracy on small, low-contrast, or occluded defects [7, 9], and little work has explored cascaded detection – a fast binary pre-filter ahead of a heavier model – as a way to reconcile the two. The third, introduced above, is the treatment of inspection video as independent frames and not as a continuous signal, despite temporal reasoning being demonstrably useful for suppressing false positives and stabilising detections [12, 13, 14, 30, 31].

A further, more practical obstacle is methodological fragmentation: published studies on sewer and pipeline defect detection use different datasets, annotation schemes, augmentation strategies, and evaluation protocols, which makes their results difficult to compare and their best practices difficult to transfer between research groups [6, 17, 27]. Part of the contribution of this dissertation is therefore not only the models themselves, but standardised datasets and a common evaluation workflow – built around the EN 13508-2:2003 defect-coding standard for the sewer material – against which all model variants are trained and compared.

Motivated by these gaps, this dissertation investigates the development of deep-learning-based computer vision methods for automatic pipeline inspection, with particular emphasis on the application and optimisation of modern YOLO architectures, the development of a cascaded detection framework, the mitigation of class imbalance through data- and loss-level interventions, and the incorporation of temporal information for video-based defect detection.

The overall aim is a computer vision system capable of detecting and localising defects in pipeline images and video under real operating conditions, with sufficient accuracy on rare and visually difficult defect categories and at a computational cost that permits deployment on inspection hardware. This aim is pursued through five specific objectives:

1. To develop and evaluate YOLO-based models for the detection and localisation of defects in pipeline imagery.

2. To improve detection of rare and underrepresented defect classes through adapted augmentation methods and training strategies.

3. To develop and analyse cascaded detection architectures that balance detection accuracy against real-time computational efficiency.

4. To integrate spatiotemporal context from inspection video, improving detection stability and reducing false positives.

5. To assess the robustness and generalisation of the proposed models on real sewer and underwater pipeline datasets.

These objectives are formalised as the following four research hypotheses, referred to throughout the dissertation as H1–H4:

H1 Modified YOLO architectures (v4–v11) can achieve a balanced trade-off between inference speed, detection accuracy, and computational resource consumption, which is essential for deployment in real-world pipeline inspection environments.

H2 A cascaded approach with an initial binary defect-detection stage enables a significant acceleration of inspection video analysis while maintaining accuracy comparable to that of single-stage detection models.

H3 Augmenting imbalanced datasets can improve detection performance, particularly for underrepresented pipeline defect classes, and enhance model generalisation in both underwater and sewer inspection environments.

H4 Establishing a standardised workflow for data preparation, model training, and evaluation improves the robustness and reproducibility of deep learning models applied to real-world pipeline inspection scenarios.

Based on these hypotheses, the scientific contributions of this research are:

1. An annotated dataset for training models for the automatic inspection of pipeline infrastructure in sewer and underwater environments;
2. A cascaded deep learning model configuration for pipeline image analysis, enabling reduced computational complexity while maintaining high detection performance for real-time operation;
3. A computer vision system architecture for the automatic detection of pipeline infrastructure components and defects.

1.5. Research Methodology

The research reported in this dissertation was conducted in five phases, which together deliver the three scientific contributions above and map onto the chapters that follow.

The first phase, *data collection and preparation*, covers the acquisition, annotation, preprocessing, augmentation, and class-distribution analysis of the two pipeline inspection datasets used throughout this dissertation (Chapter 3.). This phase also included removal of on-screen telemetry (timestamps, distance and orientation indicators) from the

underwater and sewer footage, since these overlays are fixed in position and can otherwise be picked up as spurious visual features by the trained models.

The second phase, *baseline model analysis*, reviews existing object detection architectures and establishes reference models against which later, modified architectures are compared (Chapter 2. and Chapter 4.).

The third phase, *model development and optimisation*, adapts and tunes YOLO architectures (v4–v11) for pipeline defect detection, including loss-function modification and data-balancing strategies aimed specifically at underrepresented defect classes (Chapter 5.). Training was carried out primarily in PyTorch, with GPU acceleration and mixed-precision computation used to reduce training time.

The fourth phase, *cascaded and sequential model development*, extends single-stage detection in two directions: a cascaded architecture that filters non-defective frames with a lightweight binary model before applying a heavier multi-class detector (Chapter 6.), and a dual-stream spatiotemporal architecture that aggregates information across a sliding window of consecutive frames to stabilise detections in sewer and stormwater video (Chapter 7.).

The fifth phase, *evaluation and validation*, assesses all preceding contributions against a common set of metrics – precision, recall, mean Average Precision (mAP), and inference speed in frames per second (FPS) – together with a qualitative review of correct and incorrect detections, to identify the configuration that offers the most favourable balance of accuracy, speed, and computational cost for real-world deployment (Chapter 8.).

1.6. Structure of the Doctoral Dissertation

This dissertation is organised into nine chapters. This introductory chapter has motivated the problem of automated pipeline inspection, reviewed conventional inspection practice and its limitations, and stated the research hypotheses and contributions pursued in the remainder of the dissertation, together with the five-phase methodology through which they were pursued (Section 1.5.).

Chapter 2. reviews the existing literature on pipeline inspection and defect assessment, computer vision and deep learning for infrastructure monitoring, and the specific challenges – class imbalance, environmental degradation, and the frame-independence of

standard detectors – that motivate the research presented here.

Chapter 3. describes the data constructed for this research: the underwater pipeline dataset acquired via ROV, and the sewer inspection material annotated according to the EN 13508-2:2003 defect-coding standard, which is used at two extractions and therefore yields three experimental dataset configurations in total. It also sets out the shared experimental setup and hardware used for all subsequent training and evaluation.

Chapter 4. introduces the object detection architectures evaluated in this dissertation, centred on the YOLO family (YOLOv4 through YOLOv11 and YOLO26) and the generic detection pipeline of preprocessing, backbone, neck, head, and post-processing (Section 4.3.). It also reviews two families of modifications that define the design space these architectures are drawn from: channel and spatial attention modules (SE, CBAM, ECA) and IoU-based localization losses (IoU, GIoU, DIoU, CIoU, SIoU). Both families are tested experimentally in Chapter 8.: all three attention modules, and three alternatives to the CIoU default used by the baseline architecture.

Chapter 5. addresses class imbalance in the pipeline inspection datasets, evaluating data augmentation techniques (geometric and photometric transformation, MixUp, Mosaic, copy-paste), dataset balancing strategies, and loss function adaptations (focal loss, weighted loss, class-balanced loss) for improving detection of underrepresented defect classes. Synthetic minority oversampling (SMOTE) was additionally implemented and used to generate synthetic defect instances, but the resulting samples were assessed as physically implausible by an inspection professional and were excluded from all experiments; the assessment is reported in that chapter as a negative result.

Chapter 6. presents the proposed cascade YOLO framework, in which a lightweight binary defect-detection stage filters non-defective frames before a heavier multi-class detector is applied, and evaluates its computational complexity and accuracy trade-offs against single-stage detection.

Chapter 7. presents the proposed dual-stream spatiotemporal framework, combining a YOLO-based temporal aggregation stream with a PTSEFormer-based transformer reasoning stream through late, confidence-based fusion, to improve detection stability on video instead of isolated frames.

Chapter 8. reports the experimental results across all preceding chapters: baseline YOLO architecture comparison, the impact of class imbalance mitigation, the cascade

framework, and the spatiotemporal fusion framework, evaluated on both the underwater and sewer datasets and validated against the research hypotheses stated in this chapter.

Chapter 9. summarises the findings of the dissertation, discusses their implications for the deployment of automated inspection systems, and outlines limitations and directions for future research.

2. Chapter

REVIEW OF EXISTING STUDIES

This chapter reviews the literature on automated pipeline inspection using computer vision, covering the shift from manual visual assessment to deep-learning-based detection, developments in the YOLO family, augmentation techniques for class imbalance, cascade detection frameworks, and spatiotemporal approaches to video analysis. It establishes the foundation for the work that follows and identifies the research gaps that motivate it.

2.1. Pipeline Inspection and Defect Assessment

Aging pipeline infrastructure is increasingly affected by material deterioration, structural degradation, and growing failure rates, making condition assessment an essential component of infrastructure asset management [3]. Inspection technologies can be broadly categorized into visual and non-visual methods [2]. Visual inspection is commonly conducted using CCTV systems, remotely operated vehicles (ROVs), and autonomous underwater vehicles (AUVs), which provide image and video data for defect identification and condition assessment [32, 33]. Non-visual inspection techniques include acoustic sensing, electromagnetic testing, ultrasonic inspection, thermography, and radiographic methods, which are primarily used for detecting corrosion, wall-thickness loss, leaks, and other structural anomalies [18, 20, 34, 35].

CCTV inspection remains the most widely adopted method for sewer and drainage systems, with recordings reviewed by trained operators who assign condition codes according to established standards [36] – a process that is time-consuming and prone to

subjective interpretation at scale. Jung and Reiterer note that single-camera coverage is itself a limiting factor and propose a multi-sensor platform combining a camera array and LiDAR to improve capture quality ahead of any detection stage [37], a reminder that the sensing platform is the other half of inspection quality.

To ensure consistency in defect reporting, several standardized coding systems have been developed for pipeline condition assessment. One of the earliest and most influential frameworks was the WRc Sewer Condition Classification Manual, which established standardized procedures for recording and evaluating sewer defects and significantly influenced later standards used in infrastructure inspection [38]. The EN 13508-2 standard is among the most widely adopted frameworks for sewer and drainage inspection and provides a detailed classification system for structural, operational, and construction-related defects. Frequently reported defects include deformation, cracks, fractures, joint displacement, defective lining, deposits, encrustation, infiltration, water ingress, and obstructions [39]. Standardized defect coding facilitates condition assessment, maintenance prioritization, and the creation of annotated datasets for automated inspection systems.

Visual inspection of underwater and sewer pipelines presents several challenges. Underwater imagery is often degraded by light absorption, scattering, attenuation, turbidity, and color distortion, resulting in reduced contrast and visibility [40, 41]. Similarly, sewer inspection videos frequently contain occlusions, illumination variations, motion blur, sediment accumulation, and water interference that complicate defect identification. These environmental factors can substantially affect inspection accuracy and increase the likelihood of missed detections [42].

The operational stakes that justify this inspection effort are documented in the asset-management literature. Reviews of pipeline integrity management set out the maintenance frameworks within which inspection data is acted upon [43], studies of non-revenue water quantify the losses that undetected leakage accumulates in distribution networks [15], and public-health assessments link deteriorating distribution infrastructure to contamination risk [16]. These consequences are what make recall the operative metric for an automated inspection system, a point returned to throughout this dissertation.

Traditional computer-vision approaches have attempted to automate defect detection using handcrafted features, edge detection algorithms, thresholding methods, and image-processing techniques [33]. Morphological segmentation driven by edge detection, for

example, was applied to sewer CCTV imagery well before deep learning became practical in this domain [44]. Such methods often struggle to generalize across different inspection conditions and defect types. Recent advances in deep learning have enabled the development of data-driven inspection systems capable of automatically learning discriminative features directly from image and video data. In particular, convolutional neural networks and modern object detection architectures have demonstrated superior performance for defect localization and classification in complex inspection environments [45]. Consequently, deep learning-based computer vision methods have become the dominant research direction in automated pipeline inspection [46, 47].

Within sewer inspection specifically, a substantial body of work has developed along this line. Cheng and Wang applied Faster R-CNN to sewer CCTV images and reported reliable detection of cracks, deposits, infiltration, and root intrusion [48], and subsequently refined the approach for deployment on inspection footage [49] before extending it to pixel-level defect delineation by integrating a conditional random field with a convolutional backbone [50]. Yin et al. proposed an end-to-end framework covering detection through condition reporting [51], and Klusek and Szydlo addressed the deployment side of the same problem by adapting sewer defect models to embedded devices [52] – the constraint that motivates the cascade framework developed in Chapter 6..

Comparable progress is visible in adjacent infrastructure domains, where CNN-based crack and surface-defect detection has been evaluated on concrete structures and road pavements under varying illumination and acquisition conditions [53, 54, 55]. These studies share the difficulty central to pipeline inspection: defects are small, low-contrast, and easily confused with benign surface variation.

2.2. Artificial Intelligence and Computer Vision for Infrastructure Inspection

Deep learning has become the dominant approach to object detection because it learns discriminative features directly from large-scale visual data [45], and vision-based inspection systems now identify defects that would otherwise require manual assessment [4, 5]. Three tasks are relevant to pipeline inspection – image classification, object detection,

and semantic segmentation – and are discussed in turn below.

R-CNN introduced convolutional networks for localization and classification [56, 57], Fast R-CNN shared convolutional computation across region proposals [58], and Faster R-CNN replaced external proposal generation with a learned Region Proposal Network [28, 59], evaluated on benchmarks such as PASCAL VOC and ImageNet [60, 61]. The generality of the two-stage formulation is evident from the domains it transferred to with little modification, including face and pedestrian detection and aquatic behaviour recognition [62, 63, 64]. In parallel the YOLO family adopted single-stage detection, performing localization and classification simultaneously for real-time throughput [7, 8, 65], with YOLOv4 improving accuracy through enhanced backbones and feature aggregation [9].

Beyond object detection, convolutional neural network architectures such as AlexNet [25], VGGNet [66], GoogLeNet [67], and ResNet [26] have demonstrated strong performance in image classification tasks and have been widely adopted in automated defect inspection systems. For example, Kumar et al. applied CNN-based classification models to CCTV sewer imagery and achieved accuracies exceeding 85% for detecting cracks, deposits, and root intrusions [4], while Chow et al. demonstrated that ResNet50-based models can effectively detect defects under varying environmental conditions [5].

The YOLO family's balance of accuracy and efficiency has made it the most influential detection framework, and its evolution is well surveyed: Jiang et al. across application domains [68], Terven et al. from YOLOv1 to YOLOv8 [69], Hussain on feature extraction and deployment efficiency [70], and Ali and Zhang on emerging directions including transformer modules, attention, and lightweight deployment [71].

2.2.1. Attention Mechanisms for Defect Detection

A parallel line of work has focused on refining what a detection backbone attends to, without replacing the backbone itself. Squeeze-and-Excitation (SE) networks introduced channel-wise attention by learning to reweight feature channels according to their global importance, improving discriminative power at a modest computational cost [72]. The Convolutional Block Attention Module (CBAM) extended this idea by combining channel attention with spatial attention, allowing a network to determine not only which feature channels are informative but also which spatial regions of an image are most relevant [73].

Efficient Channel Attention (ECA) subsequently showed that the fully connected layers used in SE could be replaced with a lightweight one-dimensional convolution without loss of accuracy, making channel attention practical for real-time and embedded deployment [74].

These mechanisms have been adopted directly in sewer defect detection. Li et al. combined multiscale feature extraction with explicit attention to improve localization of small, low-contrast defects [75]; Wang et al. integrated CBAM into YOLOv5 alongside involution modules and knowledge distillation, improving detection of overlapping defects while reducing model size [76]; and Goharinezhad and Mirzaei combined a ResNet50–Swin Transformer classifier with a CBAM-enhanced YOLOv8 detector, reporting an 11-percentage-point mAP gain over the unmodified baseline [77]. Comparable enhancements are reported for YOLOv5 and YOLOv4 variants [78, 79]. Attention thus emerges as a consistently reported source of improvement for exactly the defect types this dissertation targets, and that is why Chapter 4. reviews SE, CBAM, and ECA and Section 8.2.9. tests all three on this dissertation's sewer data. The benefit largely does not transfer: the three modules move mAP@0.5 by at most half a percentage point, against interventions elsewhere worth three to five. That negative result is reported in full precisely because the positive reports above are so consistent.

2.2.2. Segmentation and Transformer Architectures

Semantic segmentation offers pixel-level localization of defects, which supports quantifying defect extent and condition grading more precisely than a bounding box. Fully Convolutional Networks introduced end-to-end segmentation by replacing fully connected layers with convolutional operations [80]; U-Net's encoder-decoder structure with skip connections became the most widely adopted variant [81]; and SegNet and DeepLab improved boundary delineation through efficient decoding and atrous convolutions [82, 83]. All three have been applied directly to sewer inspection.

Transformer architectures have since emerged as an alternative to convolutional backbones. Vision Transformer showed that self-attention alone can achieve competitive classification by modelling long-range relationships between image regions [84], and Swin Transformer added hierarchical representations and shifted-window attention for efficiency

[85]. In detection, DETR reformulated the task as direct set prediction, eliminating anchor boxes and non-maximum suppression [86], and Deformable DETR addressed its slow convergence and weak small-object performance by restricting attention to a sparse set of sampling points around each reference location [87] – the mechanism the transformer stream of Chapter 7. inherits.

Three transformer-based video object detectors are most relevant to this dissertation: PTSEFormer, TransVOD, and YOLOV. They are introduced here and discussed further in Section 2.5., which reviews spatiotemporal detection specifically. In brief, Wang et al. proposed PTSEFormer, which exploits temporal relationships between consecutive frames to improve detection consistency in video [14]; Zhou et al. introduced TransVOD, extending transformer-based detection to video sequences with improved robustness under motion blur and occlusion [13]; and Shi et al. proposed YOLOV, integrating temporal information directly into a YOLO architecture [88]. Together these represent a shift from frame-based analysis toward spatiotemporal inspection systems, which is the direction Chapter 7. builds on.

2.3. Challenges in Deep Learning-Based Pipeline Inspection

Three obstacles recur when these methods are applied to pipeline inspection. The first is image degradation: underwater imagery loses information to light absorption, scattering, attenuation, and colour distortion [40, 41, 42], which architectural and augmentation responses such as CSPNet [89] and CutMix [90] partially mitigate. The second is video instability: motion blur, occlusion, low-contrast defects, forward camera motion, and inconsistent illumination reduce detection stability across sequences [14], and the two architectural responses that recur in the literature are Flow-Guided Feature Aggregation, which aligns and aggregates features along optical-flow motion paths [12], and TransVOD, which models spatial and temporal relationships jointly in an end-to-end transformer [13]. The third is class imbalance, addressed in recent work through focal loss formulations that emphasise hard and infrequent examples [11, 91] and taken up directly in Chapter 5..

2.4. Class Imbalance and Data Augmentation in Object Detection

Class imbalance is listed as one challenge among several in Table 2.1, but its treatment in this dissertation (Chapter 5.) is substantial enough to warrant its own review here. Oksuz et al. provide a detailed taxonomy of imbalance problems specifically in object detection, distinguishing class imbalance from the related but distinct problems of scale imbalance, spatial imbalance, and objective imbalance, and cataloguing the loss-level, sampling-level, and architectural solutions proposed for each [98]. At the broader scale of visual recognition generally, Zhang et al. survey deep long-tailed learning and group existing solutions into three categories – class re-balancing, information augmentation, and module improvement – a taxonomy that maps directly onto the augmentation, dataset-balancing, and loss-function-adaptation structure of Chapter 5. [99].

Within the class re-balancing category, Focal Loss addresses foreground-background and class imbalance by down-weighting well-classified examples during training so that the loss is dominated by hard and rare examples [11]. Cui et al. refined this idea with Class-Balanced Loss, which reweights classes according to their effective number of samples instead of raw frequency, on the observation that additional samples from an already well-represented class contribute diminishing new information [91]. Fernández et al. provide a book-length treatment of the broader cost-sensitive and resampling literature that both of these approaches build on [100].

Within information augmentation, SMOTE was the first widely adopted method for generating synthetic minority examples by interpolating between neighbours in feature space instead of duplicating them [101]. For detection, MixUp interpolates pairs of images and labels [102] and Copy-Paste raises minority frequency by inserting annotated instances from one image into another [103], which Kisantal et al. found particularly effective for small objects – simultaneously under-represented and disproportionately hard to detect, a combination common in pipeline data [104]. Hybrid combinations of undersampling and SMOTE-based oversampling outperform either alone [105, 106], motivating the hybrid strategy evaluated in Chapter 5..

Within infrastructure inspection, Dang et al. combined hierarchical classification with

imbalance-aware training, reporting substantial minority-class recall gains [107], and Alshawi et al. proposed an Enhanced Feature Pyramid Network that places imbalance-aware choices at the architecture level instead of the loss or data level, reporting IoU improvements of 16–29% [108]. This body of work confirms that class imbalance here is well studied but that no single mitigation dominates – which is the comparison undertaken in Chapters 5. and 8..

2.5. Video-Based Defect Detection and Temporal Analysis

Video-based inspection offers richer contextual information than single-image analysis, and CCTV inspection generates volumes of footage that increasingly require automated handling [6]. Deep-learning detectors have shown promising results on underwater video, including fish detection and target recognition [22, 109], with YOLO-based adaptations improving accuracy and robustness in those conditions [23].

The underwater YOLO literature is broad enough to indicate which adaptations transfer. Comparative studies of YOLOv3 and YOLOv4 variants report that architectural changes aimed at small, low-contrast targets matter more than raw model capacity [110, 111, 112], with attention-based multi-scale fusion used to recover accuracy lost to turbidity and colour attenuation [113] and channel pruning pursuing the same goal from the efficiency side [114]. Applications outside infrastructure confirm that image degradation, not class count, is the dominant difficulty [115, 116], and Chen et al. identify degradation and the scarcity of large annotated underwater datasets as the field's two most persistent open challenges [117] – both applicable here. Within underwater infrastructure specifically, Andani and Ameri evaluated YOLO detectors for pipe defects in robot imagery [118], and da Silva et al. released a subsea leak-detection dataset with CNN benchmarks reaching macro F1 above 0.80 [119, 120].

Spatio-temporal fully convolutional networks have been proposed for semantic video segmentation, demonstrating the benefit of incorporating information from consecutive frames [34]. Most inspection approaches nonetheless still process frames independently and do not exploit temporal relationships explicitly.

The value of temporal context is not specific to infrastructure inspection. Motion cues and frame-to-frame continuity allow detectors to recover objects that are partially occluded, distorted, or weakly visible in any single frame, and this benefit has been demonstrated across autonomous driving [121, 122, 123], video surveillance [124, 125, 126, 127, 128], industrial inspection [129], and robotics [130, 131, 132]. The methodological paradigms developed in those domains are what the following paragraphs summarise, since they define the design space from which a pipeline inspection framework must choose.

The simplest paradigm is spatiotemporal feature aggregation, in which features from several frames are combined to stabilise detections. YOLO-based extensions implement this by stacking or aligning neighbouring frames so that a spatial detector inherits temporal consistency [88, 124, 133, 134, 135]. Frame stacking works well for a fixed camera, but a moving platform such as a drone or an inspection robot introduces apparent motion that must be compensated explicitly before features from different frames describe the same physical location [136]. This constraint applies directly to pipeline inspection, where the camera advances continuously along the pipe.

Aggregation can also operate at a more abstract level than individual features. Sequence-level semantic pooling makes predictions informed by an entire clip instead of a local window [137], and adaptive weighting of surrounding frames improves robustness when some frames in the window are degraded [138]. Memory-based designs extend the effective temporal receptive field further, either by maintaining a global-local feature memory [139], by mining proposal relations across different videos [140], by aligning regions of interest across time [141], or by attention-based memory modules that retain long-range spatiotemporal state [142]. Post-processing methods that link detections into tubelets, such as sequence non-maximum suppression, achieve a related effect without modifying the detector itself [143].

A third family models inter-frame motion explicitly. Early approaches relied on optical-flow-based feature propagation to transfer information between adjacent frames and compensate for motion-induced appearance changes; Deep Feature Flow and Flow-Guided Feature Aggregation established that temporal feature reuse improves accuracy while reducing computation [12]. Unified frameworks that jointly encode appearance and motion improved the speed–accuracy balance further [144], while later designs learn where and how to sample temporal information instead of relying on a fixed flow estimate,

through learnable spatiotemporal sampling [145] or dual-level motion calibration at both pixel and instance level [146].

Transformer architectures subsequently replaced hand-crafted aggregation with learned attention. End-to-end spatial–temporal transformers learn temporal alignment and feature fusion directly, removing the need for optical flow or recurrent components [13, 147]. Related designs combine deformable alignment, semantic distillation, and proposal refinement to recover real-time throughput, including FastVOD-Net, TCNet, and dual selection networks [148, 149, 150]. Attention has also been extended from images to video classification: TimeSformer applies self-attention across both spatial and temporal dimensions [30], and Video Swin Transformer introduced hierarchical spatiotemporal representations that reached state-of-the-art results on several video benchmarks [31].

A final group pursues temporal consistency at minimal computational cost by reusing detector features across frames or attaching a lightweight regression tracker to an existing detector [151]. Surveys of multi-object tracking document both the maturity of these methods and their persistent difficulties with occlusion and abrupt appearance change [152]. Tracking maintains identity for objects already detected, so it does not recover a defect that the underlying detector never found in any frame, which is the specific failure this dissertation targets.

Sequential frames therefore provide motion cues and contextual information that help identify defects under occlusion, distortion, motion blur, and inconsistent illumination [14].

Transformer-based video object detection models have further advanced the exploitation of temporal context. Approaches such as TransVOD and PTSEFormer employ attention mechanisms to model long-range temporal dependencies and semantic consistency across video sequences, enabling improved detection of rare, ambiguous, or partially occluded objects [13, 14]. These methods demonstrate the advantages of combining spatial feature extraction with temporal reasoning, particularly in challenging inspection environments where individual frames may contain insufficient information for reliable defect identification.

Despite these advances, relatively few studies have investigated the application of spatiotemporal object detection frameworks specifically to sewer and underwater pipeline inspection. Most existing inspection systems continue to operate on individual frames

and therefore fail to fully exploit temporal consistency available in inspection videos.

The gap is partly one of setting. The video object detection literature surveyed above was developed largely for open or static scenes, whereas an enclosed pipe imposes conditions that appear together only rarely elsewhere: forward ego-motion along the optical axis, radial geometry in which apparent motion expands outward from the image centre, LED illumination that flickers and falls off sharply with distance, and visually repetitive wall texture that offers few stable features for alignment. Each degrades the frame-to-frame correspondence that every aggregation method above depends on. This combination motivates the development of inspection frameworks that integrate spatial and temporal information under precisely these constraints, to improve robustness, detection consistency, and performance on rare or visually ambiguous defect categories.

2.6. Cascade Detection Systems and Model Optimization

Recent research has focused on improving the trade-off between detection accuracy and computational efficiency. Feature aggregation methods such as PANet enhance information flow across network layers and improve localization performance [153]. Similarly, Spatial Pyramid Pooling (SPP) enables effective multi-scale feature extraction for objects of varying sizes [154]. Advanced architectures such as Scaled-YOLOv4 incorporate network scaling strategies to optimize performance across different computational constraints [155]. Backbone networks including ResNet and DenseNet have demonstrated strong feature-learning capabilities and are widely used in modern object detection frameworks [26, 156].

YOLO-based inspection systems have been enhanced through self-attention, vision transformers, optimized losses, and channel pruning. Chen et al. reported mAP above 93% for pipeline defect detection with a self-attention and transformer-augmented YOLOv5 [78]; Situ et al. showed channel pruning improves inference speed at maintained accuracy [27]; and Zhang et al. improved sewer detection by adding Spatial Pyramid Pooling to YOLOv4 [79, 157]. Lightweight sewer-specific variants target resource-constrained deployment while preserving real-time performance [158, 159, 160]. Two recent examples

illustrate that accuracy and efficiency are increasingly pursued together instead of traded off: Lu et al. gained 4.5 mAP points over their baseline at nearly 70 FPS using selective-kernel attention with bidirectional cascade fusion [161], and Sha et al. improved mAP while reducing parameter count and model size relative to YOLOv8n [162].

In parallel with architectural improvements, researchers have increasingly investigated cascaded inspection frameworks designed to address severe class imbalance and computational redundancy in inspection data. Such approaches employ a lightweight anomaly-detection or defect-presence module to rapidly filter non-defective samples before applying more computationally intensive localization and classification algorithms. Moradi et al. demonstrated the effectiveness of anomaly-based pre-filtering for sewer CCTV inspection, while subsequent studies extended similar concepts to infrastructure monitoring and industrial visual inspection tasks [163]. Recent work has shown that cascaded pipelines can substantially reduce computational requirements while maintaining high defect recall, making them particularly attractive for large-scale infrastructure inspection systems operating in highly imbalanced environments [164, 165, 166].

Recent studies have explored hybrid detection architectures that combine convolutional neural networks with transformer-based temporal reasoning modules. For example, dual-stream frameworks integrate fast YOLO-based detectors with transformer architectures capable of capturing long-range temporal dependencies, enabling improved performance for rare, ambiguous, and partially occluded defects. Late-fusion strategies further enhance detection reliability by combining predictions from complementary models while maintaining practical inference speeds [14]. Similar trends can be observed in recent video object detection approaches, where temporal feature aggregation, attention mechanisms, and transformer-based architectures have demonstrated substantial potential for improving detection consistency in sequential inspection data [12, 13, 31].

Beyond object detection, segmentation and transformer-based approaches have also gained attention for sewer inspection. PipeTransUNet combines convolutional and transformer features for semantic segmentation and severity assessment of sewer defects, demonstrating the benefits of integrating local and global contextual information [167]. At the same time, benchmark datasets such as Sewer-ML have facilitated standardized evaluation and comparative analysis of modern deep-learning methods for sewer condition assessment [168].

Two further paradigms have emerged that depart from the supervised-detector approach followed in this dissertation, and are worth noting precisely because of that contrast. Yin et al. proposed a weakly supervised object localization framework for sewer defect detection that requires only image-level labels instead of bounding-box annotations, trading some localization precision for a substantially reduced annotation burden [169]. Taormina and van der Werf evaluated eighteen general-purpose vision-language models in a zero-shot setting on multi-class sewer defect classification and found that the best proprietary model reached a macro-F1 of only 0.50, with all models failing on production errors and performing poorly on deformations [170] – evidence that, at least for now, task-specific supervised detectors of the kind investigated in this dissertation remain necessary without being superseded by general-purpose foundation models.

2.7. Summary and Research Gap

The literature reviewed above establishes the effectiveness of deep learning for automated pipeline inspection across classification, detection, and segmentation. Its limitations are also consistent: approaches tend to target specific defect categories, individual vision tasks, or highly specialised datasets, and the challenges of class imbalance, environmental variability, dataset availability, and temporal information remain only partially addressed. Figure 2.1 summarises that pattern of coverage across the five themes surveyed in this chapter, and locates the two gaps the following chapters address.

Two gaps follow specifically. First, little existing research aligns deep learning-based inspection with standardized assessment frameworks such as EN 13508-2:2003 [39], the step that would let a detector's output enter an existing inspection workflow unchanged. Second, although spatiotemporal information has been shown to improve video-based defect detection, few studies have integrated real-time detectors with transformer-based temporal reasoning for pipeline inspection specifically, so most systems continue to analyse frames independently and forgo the continuity that helps most on rare, low-contrast, and partially occluded defects [14]. These two gaps define the work reported in the chapters that follow.

Theme reviewed	Established in wider literature	Shown on pipeline inspection data	Gap addressed in this work
Detector architectures (YOLO family)	●	◑	–
Class imbalance mitigation	●	◑	–
Environmental image degradation	●	◑	–
Standardised coding (EN 13508-2)	◑	○	gap
Spatiotemporal video reasoning	●	○	gap

Coverage ● extensive ◑ partial ○ little or none

Figure 2.1: Coverage of the five themes surveyed in this chapter, distinguishing what the wider computer-vision literature has established from what has been demonstrated on pipeline inspection data specifically. The two rows marked as gaps are those where pipeline-specific work is largely absent: alignment of detector output with the EN 13508-2:2003 coding scheme, and the integration of a real-time detector with transformer-based temporal reasoning (Section 2.5.). Class imbalance is shown as partially addressed because the general techniques are well established (Section 2.4.) while their behaviour on long-tailed defect data is not; these are the gaps the remaining chapters address.

Table 2.1: Key challenges in deep learning-based pipeline inspection.

Challenge	**Description**	**Representative references**
Image degradation	Underwater images are affected by light absorption, scattering, attenuation, turbidity, low visibility, and color distortion, which reduce contrast and obscure defect features.	[40, 41, 42]
Video instability	Inspection videos often contain motion blur, occlusions, forward camera motion, low-contrast defects, and inconsistent illumination, reducing temporal detection stability.	[12, 13, 14]
Class imbalance	Critical defect categories are often rare compared with normal pipeline regions or common defects, causing models to underperform on minority classes.	[10, 11]
Limited datasets	Pipeline datasets are difficult to collect and annotate because inspection requires specialized equipment, expert knowledge, and standardized defect interpretation.	[46, 92, 93, 94, 95]
Annotation subjectivity	Defect coding and severity assessment can vary between inspectors, especially when defect boundaries are ambiguous or image quality is poor.	[34, 39]
Domain shift	Models trained on one inspection environment may perform poorly on different pipe materials, cameras, lighting conditions, water quality, or defect appearances.	[46, 96]
Computational constraints	Real-time inspection requires efficient models, but higher accuracy often requires larger architectures and greater computational resources.	[89, 97]

3. Chapter

DATASETS AND EXPERIMENTAL SETUP

3.1. Introduction

The performance and generalization of deep learning models depend largely on the quality, diversity, and representativeness of the data used for training and evaluation. In this dissertation, two independent datasets are used to investigate object detection performance in distinct pipeline inspection environments.

The first dataset contains underwater pipeline inspection images acquired by remotely operated vehicles (ROVs). Underwater imagery is affected by poor visibility, colour degradation, light attenuation, scattering, and suspended particles, all of which can reduce image quality and obscure object boundaries [22, 23]. These conditions make reliable object detection particularly challenging and require models that can cope with substantial visual variability. The dataset was previously introduced in [129, 171].

The second dataset was derived from CCTV recordings collected during sewer pipeline inspections. Sewer imagery presents a different set of challenges, including confined spaces, non-uniform illumination, water ingress, sediment accumulation, structural deterioration, and visual artifacts introduced by the inspection equipment [4, 5]. The annotated defect categories are based on the EN 13508-2:2003 standard [39].

The two datasets serve different evaluation purposes. The Underwater Pipeline Dataset is used to assess the detection of pipeline components and related infrastructure elements,

whereas the Sewer Inspection Dataset focuses on the detection and classification of structural and operational defects. Evaluating the models on both datasets makes it possible to examine their performance across two visually and operationally different inspection domains.

The sewer material is used at two different extractions, described in Section 3.6.. Both come from one annotation effort over the same 56 inspection videos: the image-based experiments use a quality-filtered selection of 13,000 frames, while the cascade and spatiotemporal experiments of Chapters 6. and 7. require the complete 22,427-frame stream in video order, because a gate and a temporal window are both defined over the continuous signal. They are one dataset filtered two ways, not two datasets.

3.2. Dataset Overview

Table 3.1 summarizes the main characteristics of the datasets used throughout this research.

Table 3.1: Summary of datasets used in this dissertation.

Characteristic	Underwater Dataset	Sewer Dataset
Application Domain	Subsea Inspection	Sewer Inspection
Image Source	ROV Cameras	CCTV Cameras
Number of Images	3,021	13,000
Number of Classes	5	12
Annotation Type	Bounding Boxes	Bounding Boxes
Annotation Format	YOLO	YOLO
Training Images	2,415	10,400
Validation Images	303	1,300
Testing Images	303	1,300
Main Challenge	Water Distortion	Defect Diversity

3.3. Underwater Pipeline Dataset

3.3.1. Data Acquisition

The underwater dataset was created from video recordings acquired during pipeline inspection missions using remotely operated vehicles equipped with high-resolution cameras. The recordings come from commercial inspection surveys carried out on operational

subsea pipelines, and were made available for this research by the company performing them. This provenance shapes the character of the data in ways that matter for the experiments that follow. Camera paths, lighting, and survey speed were dictated by inspection practice, no scene was staged or repeated, and the frequency with which each object class appears reflects what is actually present on the surveyed sections. The pipeline was recorded from three viewing perspectives and at different distances, resulting in variations in object size, orientation, visibility, and surrounding seabed conditions.

Frames were extracted from the recorded videos at a rate of two frames per second. This sampling rate was selected to retain changes in viewpoint and inspection conditions while limiting the number of highly similar consecutive frames.

The final dataset consists of 3,021 annotated images containing five object categories, as summarized in Table 3.2.

Table 3.2: Distribution of object categories in the Underwater Pipeline Dataset [172].

Class	**Number of Instances**
Pipeline	2,984
Concrete Weight	1,345
Concrete Mat	662
Leakage Point	191
Pipe Coupling	107
Total instances	5,289
Total annotated images	3,021

The per-class distribution mirrors the imbalance pattern seen in the sewer dataset described later in this chapter (Table 3.4) and analysed in Section 5.1.1.: the pipeline itself, present in nearly every frame, accounts for well over half of all annotated instances, while the two classes most directly indicative of an operational problem or a maintenance-relevant joint, leakage points and pipe couplings, are the least represented, with pipe couplings appearing barely a tenth as often as concrete weights. This is the same asymmetry motivating the class-imbalance mitigation techniques evaluated in Chapter 5.: the rarest classes in this dataset are also among the most operationally significant.

3.3.2. Dataset Annotation

All images were manually annotated using a YOLO Label [173]. Object instances were labelled using rectangular bounding boxes and exported in YOLO annotation format.

For each object instance, the following information was recorded:

- Class identifier
- Bounding box center coordinates
- Bounding box width
- Bounding box height

Coordinates were normalized to the image dimensions to maintain consistency across different image resolutions.

3.3.3. Environmental Challenges

Underwater imagery presents several unique challenges that influence object detection performance:

- Light attenuation
- Color distortion
- Backscatter noise
- Low contrast
- Motion blur
- Biofouling and marine growth

Representative samples showing these conditions are given in Figure 1.2b of Chapter 1..

3.3.4. Dataset Split

The dataset was divided according to an 80:10:10 ratio, giving 2,415 training, 303 validation, and 303 test images (Table 3.1). This partition is retained as it was defined in the source study [129].

3.4. EN 13508-2 Standard

The inspection and condition assessment of sewer and drainage pipelines require a standardized methodology to ensure consistency, repeatability, and comparability of inspection results. In Europe, one of the most widely adopted standards for the coding and classification of pipeline defects is EN 13508-2, entitled Investigation and Assessment of Drain and Sewer Systems Outside Buildings âĂŞ Part 2: Coding System for Visual Inspection [39]. The standard establishes a unified framework for describing structural and operational conditions observed during visual inspections of sewer and drainage infrastructure.

EN 13508-2 defines a detailed coding system that enables inspectors to systematically document observed defects, anomalies, and other relevant pipeline characteristics. The standard categorizes observations into structural defects, operational defects, construction features, and additional inspection observations. Structural defects include phenomena such as cracks, fractures, deformations, joint displacements, collapses, and corrosion, while operational defects encompass issues such as sediment deposits, root intrusion, infiltration, and blockages. Each observation is assigned a standardized code that describes its type, severity, position, and other relevant attributes.

A standardized coding system reduces subjectivity by providing explicit classification criteria, facilitates data exchange between operators, inspection companies, and regulatory authorities, and enables the construction of consistent datasets for automated inspection.

Within this research, EN 13508-2 [39] is the basis for annotating and categorizing defects in the sewer imagery and video. All defect classes used during model development are drawn from categories defined in the standard, and the annotation follows its terminology and classification principles. Aligning the label set with an internationally recognized standard is what lets the resulting models be evaluated against, and integrated into, the inspection workflows they are intended to support.

3.5. Sewer Inspection Dataset

3.5.1. Dataset Development

The dataset used in this research was created from CCTV recordings acquired during inspections of operational sewer pipelines at multiple locations. The recordings included pipes with different structural conditions, defect types, and operating characteristics. Frames were extracted from the videos at a rate of two frames per second, producing approximately 22,000 raw images (22,427 exactly).

Frames with insufficient visual information, substantial redundancy, or other quality issues were removed during preprocessing. Following filtering and annotation, the image-based dataset consisted of 13,000 images. Table 3.3 sets out the full derivation, since the same raw material supports two different extractions and the intermediate counts are referred to individually in later chapters.

Table 3.3: Derivation of the sewer material from the raw sampled frames. The final two rows partition the complete 22,427-frame stream and are the counts used by the sequence-based experiments of Chapters 6. and 7.; the 13,000-image row is the image-based dataset used in Chapters 5. and 8..

Stage	**Number of frames**
Raw sampled frames (2 fps)	≈22,000 (22,427)
Frames after quality filtering (image-based dataset)	13,000
Frames with at least one retained annotation	11,967
Frames annotated as defect-free (background)	10,460

The last two rows partition the complete stream exactly: 11,967 + 10,460 = 22,427. The 13,000-image row is a separate selection from the same material, retaining all annotated frames together with a smaller number of quality-checked background frames, and it is this selection that the per-image experiments of Chapters 5. and 8. use.

3.5.2. Annotation and Preprocessing

The images were manually annotated using rectangular bounding boxes. Defect categories were defined according to the EN 13508-2:2003 coding system for the visual inspection of drains and sewers. Twelve defect classes were included, listed with their instance counts in Table 3.4.

The number of annotated instances per class is given in Table 3.4. The distribution is severely long-tailed: pipe misalignment (BAJ) and deposits (BBC) together account for more than half of all annotations, while four codes – defective connection (BAH), vermin (BBH), obstructions (BBE), and break (BAC) – are each represented by fewer than one hundred instances. This distribution is the direct motivation for the class imbalance mitigation techniques evaluated in Chapter 5., and BBH and BBE in particular are the two classes on which the loss-function adaptations of that chapter produce their largest measured effect (Chapter 8.).

Table 3.4: Per-class instance counts in the 13,000-image, 12-class Sewer Inspection Dataset [174]. Counts marked * were recovered directly from the YOLO annotation files and are exact; because both extractions of the sewer material share the same 11,967 annotated frames (Table 3.5), they agree with Table 3.6 by construction. The three remaining codes are given as upper bounds: defective connection (BAH), vermin (BBH), and obstructions (BBE) are annotated more densely in this twelve-class pass than in the sequence extraction, where they carry one, zero, and zero instances respectively, so their counts here cannot be recovered from the sequence labels. Per-class results for all three are reported in Chapter 8..

Code	Description	Instances
BAA	Deformation	540*
BAB	Crack	548*
BAC	Break	50*
BAH	Defective Connection	<50
BAI	Intruding Sealing Material	228*
BAJ	Pipe Misalignment	5,363*
BAK	Defective Lining	2,211*
BBB	Wall Attachments	1,069*
BBC	Deposits	2,866*
BBF	Water Ingress	73*
BBH	Vermin	<50
BBE	Obstructions	<50

Before model training, visual overlays embedded in the CCTV recordings were removed where necessary. These included timestamps, distance indicators, and camera orientation markers. Figure 3.1 shows a representative frame before and after this masking step. The annotations were subsequently reviewed to identify missing, incorrectly positioned, or incorrectly classified bounding boxes. The effect of masking on detection accuracy is examined experimentally in Section 8.2.11. of Chapter 8., where the measured outcome on the sewer data proves less straightforward than the preprocessing rationale would suggest [175].

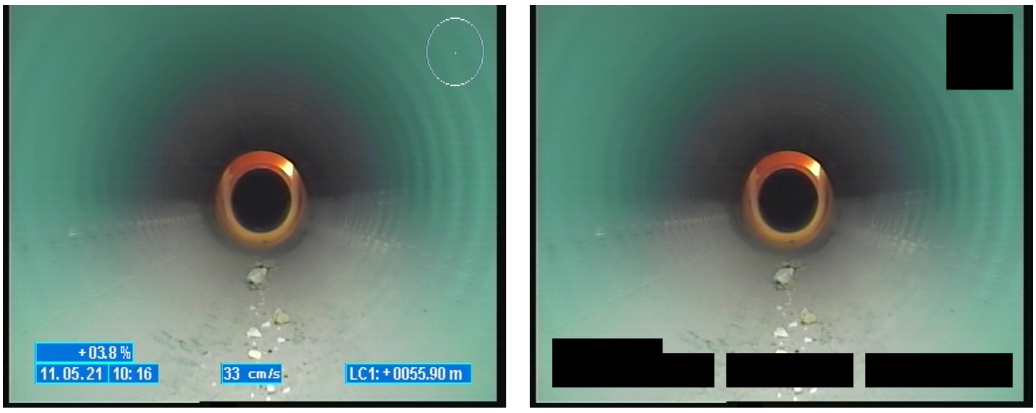


Figure 3.1: A sewer CCTV inspection frame (a) before and (b) after masking the fixed-position on-screen overlays: orientation indicator, timestamp, marching speed, and distance readout [175].

Class-balancing procedures and data augmentation were applied to the training data to reduce the effect of class imbalance and increase variation in the available training samples. Further details of the balancing and augmentation strategies are provided in the corresponding experimental sections.

The final dataset was partitioned using an 80:10:10 ratio, giving 10,400 training, 1,300 validation, and 1,300 test images (Table 3.1).

3.6. Video Sequence Dataset for Spatiotemporal Detection

The cascade and dual-stream spatiotemporal experiments in Chapters 6. and 7. draw on the same annotation effort as the Sewer Inspection Dataset above, but at a different extraction. The image-based experiments use the quality-filtered selection of 13,000 frames, which a per-image detector can consume in any order. A gate that must decide whether a frame is worth analysing, and a temporal window that must span consecutive observations, both require every intervening frame to be present, so those experiments use the full sampled stream of 22,427 frames in its original video order.

Table 3.5 sets out the relationship, and Table 3.3 above gives the derivation of both from the raw material. The full stream comprises 22,427 frames, of which 11,967 carry at least one bounding box and 10,460 are annotated as containing no defect. Ten of

the twelve EN 13508-2 codes are populated: vermin (BBH) and obstructions (BBE) are defined in the annotation scheme and appear in the twelve-class per-image experiments of Chapters 5. and 8., but have no instances in this material, and the sequence experiments therefore report ten classes. Of the ten that are populated, defective connection (BAH) has a single annotated instance and is therefore not evaluable in practice; it is reported as not applicable wherever per-class results for this extraction are given (Chapter 8.).

Because the two extractions differ in composition, results are not compared across them anywhere in this dissertation. A detector evaluated on defect-bearing frames alone faces a different problem from one evaluated on a stream that is 47% empty, and the second number is the lower of the two for reasons that have nothing to do with the model.

Table 3.5: The two extractions of the sewer material. Both derive from one annotation pass over the same 56 inspection videos; see Table 3.3 for the derivation from the raw sampled frames.

	Image-based	**Sequence-based**
Frames	13,000	22,427
with $\geq$1 annotation	11,967	11,967
annotated defect-free	1,033	10,460
Populated classes	12	10 (9 evaluable)
Frame order preserved	no	yes
Used in	Ch. 5., 8.	Ch. 6., 7.

The dataset covers ten of the twelve EN 13508-2:2003 codes (BAA, BAB, BAC, BAH, BAI, BAJ, BAK, BBB, BBC, BBF); BBH and BBE were not annotated in this pass and are evaluated only in the twelve-class dataset. Background frames carrying no annotated defect were deliberately retained, so the detector sees genuinely defect-free pipe wall during training, and that is why the instance count in Table 3.6 (12,949, computed directly from the YOLO label files) is smaller than the 22,427 frames.

The imbalance visible in Figure 3.2 is severe even by the standards of inspection data. Pipe misalignment (BAJ) accounts for more than 40% of all annotated instances, while defective connection (BAH) has a single labelled instance across the entire dataset. This ratio is more extreme than in the twelve-class Sewer Inspection Dataset (Section 5.1.1., Chapter 5.), and it explains why the class imbalance mitigation strategies of Chapter 5. were developed against the twelve-class dataset: a class represented by one instance offers no basis on which to measure whether a mitigation technique has helped.

Table 3.6: Per-class instance counts in the video sequence dataset (22,427 frames including non-target background frames, 10 classes).

Class	**Number of Instances**
BAA (Deformation)	540
BAB (Crack)	548
BAC (Break)	50
BAH (Defective Connection)	1
BAI (Intruding Sealing Material)	228
BAJ (Pipe Misalignment)	5,363
BAK (Defective Lining)	2,211
BBB (Wall Attachments)	1,069
BBC (Deposits)	2,866
BBF (Water Ingress)	73
Total	12,949

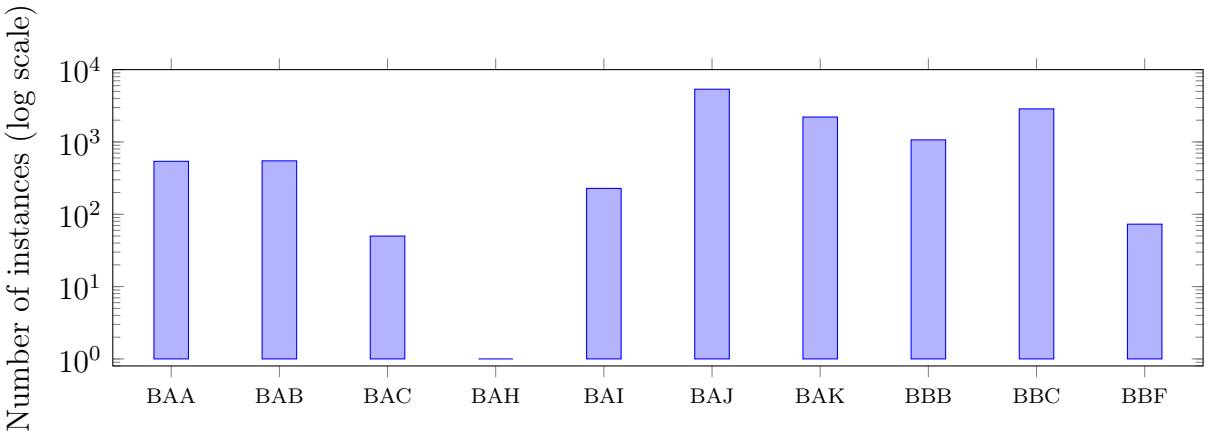


Figure 3.2: Class distribution of the video sequence dataset (Table 3.6), log-scaled.

Frames were sampled at two per second and grouped into five-frame temporal windows (two preceding and two following each reference frame) for the YOLOv7V and PT-SEFormer streams of Chapter 7., with on-screen overlays masked as before. The 22,427 frames were split approximately 80:10:10 into roughly 17,940 training, 2,240 validation, and 2,247 testing frames, at the video level so that frames from one inspection never appear in more than one subset. Because videos differ in length, the number assigned to a subset is not proportional to its frame count: the test subset comprises ten complete inspection videos, among the shorter recordings, together accounting for about 10% of frames. These ten are the units over which the paired significance test of Section 7.6.8. is computed.

3.7. Experimental Setup

Within each of the studies reported in Chapter 8., all models being compared were trained and evaluated on the same hardware, under the same software stack, and with the same data splits, so that the comparison inside a study is fair. Across studies the environment differs, because the work was carried out over several years and on the equipment available at the time. This is one of the reasons why absolute values from different studies are not compared against one another anywhere in this dissertation; only the ordering within a study is interpreted. The two platforms used are listed in Table 3.7.

Input images were resized to the required network resolution before training. Transfer learning from MS COCO weights and training from randomly initialised weights were both used, and which of the two applies is stated for each experiment: the underwater YOLOv4-family comparison used COCO-pretrained weights throughout (a YOLOv4 control trained from scratch reached 90.98% mAP against 94.21% with transfer learning), whereas the sewer YOLOv7 comparison trained every variant from scratch, the "s" suffix in Table 8.4 marking the scratch configurations.

Training hyperparameters were not held constant across all experiments. Within a study they were tuned per architecture, since the architectures differ in depth, batch behaviour, and optimizer sensitivity: the underwater study varied Darknet subdivisions (8–64) and learning rate (1×10^{-4} to 2.6×10^{-3}), and the sewer study varied epochs (250–350), learning rate (0.005–0.010), and momentum (0.900–0.937) between YOLOv7 variants. What was held constant within each study is the data split, the evaluation protocol, and the metric definitions, on which the comparisons in Chapter 8. rest.

Model performance was evaluated using the following metrics:

- Precision
- Recall
- F1-score
- Mean Average Precision (mAP@0.5)
- Mean Average Precision (mAP@0.5:0.95)

– Frames Per Second (FPS)

These metrics provide a detailed assessment of both detection accuracy and computational efficiency.

All experiments were run on GPU workstations at the Faculty of Engineering, University of Rijeka, on the two platforms detailed in Table 3.7. The YOLOv4-family underwater comparison used Platform A; all sewer, cascade, and spatiotemporal experiments used Platform B. The underwater YOLOv5–YOLOv8 comparison and the contextual-background experiment fall between the two and their platform was not recorded; since neither reports inference throughput, no figure in this dissertation is affected.

Deep learning models were implemented using Python and the PyTorch framework, selected for its maturity and its dominance in current object detection research; comparative evaluations of the major open-source deep learning frameworks report broadly similar training throughput, so the choice was driven by ecosystem support and not raw performance [176]. The earlier YOLOv4 experiments were run under the Darknet implementation in which that architecture was originally released [177], and the two-stage Faster R-CNN baseline was implemented using Detectron2 [178]. On the A6000 platform the software environment was Ubuntu 22.04 LTS with PyTorch 2.3, the Ultralytics library for the YOLOv8–YOLO26 generations, CUDA Toolkit 12.4, and cuDNN 9.0; quantized-inference benchmarking (Section 8.4.6.) additionally used OpenVINO 2025.1.

To ensure experimental reproducibility, all models were trained using fixed random seeds and identical software configurations. Python virtual environments were employed to maintain consistent package versions and dependency management throughout the research process.

Mixed-precision (FP16) training was used where appropriate to reduce GPU memory consumption and accelerate training while maintaining detection performance. Data loading and preprocessing operations were parallelized to maximize hardware utilization and minimize training bottlenecks.

Table 3.7 summarizes the hardware and software configuration used throughout the experimental evaluation.

Table 3.7: Hardware and software configuration of the two experimental platforms.

Component	Platform A (underwater)	Platform B (sewer, cascade, video)
CPU	Intel Xeon E5-2620 v4	AMD Ryzen Threadripper PRO 3995W
RAM	128 GB	256 GB
Primary GPU	NVIDIA RTX 2080 Ti (11 GB)	NVIDIA RTX A6000 (48 GB)
Secondary GPU	–	NVIDIA Quadro RTX 5000 (16 GB)
Operating System	Ubuntu	Ubuntu 22.04 LTS
CUDA / cuDNN	–	12.4 / 9.0
Framework	Darknet, Detectron2	PyTorch 2.3, Ultralytics
Used for	Table 8.1	sewer, cascade, and video experiments

3.8. Chapter Summary

Two bodies of data are used: the 3,021-image, five-class Underwater Pipeline Dataset acquired by ROV, and sewer CCTV material annotated under EN 13508-2:2003. The sewer material is one annotation effort over 56 videos used at two extractions (Tables 3.3 and 3.5): 13,000 quality-filtered frames for the per-image experiments of Chapters 5. and 8., and the full 22,427-frame stream with its 10,460 defect-free frames for the cascade and spatiotemporal experiments requiring frame continuity. Every subsequent table and figure states which extraction it draws on, and no comparison is made across them.

4. Chapter

DEEP LEARNING MODELS FOR PIPELINE DEFECT DETECTION

The application of deep learning has significantly advanced the field of automated visual inspection, enabling the development of systems capable of detecting and classifying structural defects with a level of accuracy previously unattainable using traditional image processing techniques [17, 26, 27]. In pipeline inspection, deep learning models have demonstrated considerable potential for addressing the limitations of manual assessment by automatically analyzing large volumes of inspection imagery and identifying defects such as cracks, corrosion, deformation, joint displacement, and deposits [4, 5, 6, 129]. The ability of these models to learn complex visual patterns directly from data has made them particularly well suited for operation in challenging inspection environments characterized by varying illumination conditions, image degradation, occlusions, and substantial intra-class variability.

Among the numerous deep learning approaches proposed in the literature, object detection architectures have emerged as the dominant paradigm for automated pipeline defect detection [6, 17, 27]. Unlike image classification methods, which determine only the presence or absence of a defect, object detection models simultaneously localize and classify defects within an image, producing output that maps onto inspection and maintenance workflows. Which architecture to select remains a difficult question, however, since designs differ in accuracy, computational complexity, memory requirement, and robustness to environmental variability, and this chapter therefore sets out the architectures

evaluated in this dissertation and the design space from which their modifications are drawn.

4.1. Object Detection Architectures

Modern object detection algorithms are generally categorized into two-stage and one-stage detectors. Two-stage detectors such as Faster R-CNN [28] first generate candidate object regions and subsequently perform classification and localization on the proposals, which yields high accuracy at the cost of substantial computation and longer inference. One-stage detectors instead perform localization and classification simultaneously within a single network, reducing complexity enough to permit real-time operation [7, 8, 9]. Both share a common workflow: feature extraction from the input image, followed by localization and classification, producing bounding boxes and class labels (Figure 4.1).

Among one-stage detectors, the YOLO (You Only Look Once) family has become the most widely adopted framework [7], with successive versions improving feature extraction, multi-scale representation, and training strategy [8, 9, 179, 180, 181]. This suits pipeline inspection, where systems operate on continuous image streams acquired by mobile platforms and real-time processing matters as much as accuracy [27, 129, 171]. The modular design of these architectures also accommodates the attention modules, feature-pyramid changes, and loss-function substitutions investigated later in this dissertation [27, 78, 79, 157].

This research focuses primarily on the evaluation and optimization of modern YOLO architectures, ranging from YOLOv4 [9] to YOLOv11 [182]. These models were selected due to their widespread adoption within the computer vision community, demonstrated effectiveness in industrial inspection applications, and suitability for real-time deployment. The following sections provide an overview of the considered YOLO architectures, their key design characteristics, and the modifications investigated to improve defect detection performance in underwater and sewer pipeline environments.

4.2. YOLO Architecture Overview

Since Redmon et al. introduced the original architecture [7], successive YOLO generations have advanced feature extraction backbones, multi-scale fusion strategies, anchor design, loss functions, and training procedures [9, 69, 155, 179, 183], progressively improving small-object detection and efficiency on constrained hardware. The models considered in this research span YOLOv4 to YOLOv11, with YOLO26 evaluated separately once it became available.

4.2.1. YOLOv4

YOLOv4 [9] combines a CSPDarknet53 backbone, a Path Aggregation Network (PANet) neck, and a YOLOv3-based detection head, together with the training and inference optimizations its authors group as the "Bag of Freebies" and "Bag of Specials": Mosaic augmentation, Self-Adversarial Training, Cross-Stage Partial connections, Mish activation, and CIoU loss. Its primary contribution was demonstrating that state-of-the-art accuracy could be reached on conventional GPU hardware, and it has since been applied to structural defect, corrosion, and crack detection in infrastructure settings [79, 129]. Its weakness is very small objects and complex defect patterns under degraded conditions, but it remains the baseline on which several later versions were built [155].

4.2.2. YOLOv5

YOLOv5 [184] was released by Ultralytics in 2020 and was the first of the family implemented entirely in PyTorch [185], which is largely why it became the common baseline for defect detection research. It combines CSP-based feature extraction [89] with Path Aggregation Networks for multi-scale fusion [153] and Mosaic and MixUp augmentation [102], and is published in a range of scaled variants from embedded-sized to accuracy-optimized.

4.2.3. YOLOv6

YOLOv6 [186] was introduced by Meituan as an industrially oriented framework optimized for deployment efficiency. It pairs an EfficientRep backbone with a Rep-PAN ag-

gregation structure and decoupled detection heads, and its hardware-aware design choices make it attractive for embedded, robotic, and edge platforms operating under strict computational constraints [171, 186, 187].

4.2.4. YOLOv7

YOLOv7 introduced Extended Efficient Layer Aggregation Networks (E-ELAN), model scaling strategies, and trainable bag-of-freebies techniques that improve gradient propagation without substantially increasing computational cost [179]. It reached state-of-the-art accuracy–speed trade-offs at release [69], has been adopted for infrastructure inspection [175, 188], and its aggregation structure is the insertion point for the temporal extensions used in Chapter 7. [88]. It is therefore the base detector for most of the experimental work reported here.

4.2.5. YOLOv8

YOLOv8 [183] was the first release in the family to abandon anchor-based detection, which removes a set of hyperparameters from training and improves performance on small and irregularly shaped objects [69]. It pairs a CSP-inspired backbone with an enhanced fusion network and decoupled heads that optimize classification and localization separately. Its versatility across detection, segmentation, pose estimation, and tracking within one implementation is largely why it became a standard baseline in defect detection research [162, 189].

4.2.6. YOLOv9

YOLOv9 addresses the information degradation that occurs as data propagate through a deep network, introducing Programmable Gradient Information (PGI) to preserve gradient flow and combining it with the Generalized Efficient Layer Aggregation Network (GELAN) [180]. The reported benefit is concentrated on small objects, low-contrast features, and cluttered backgrounds. For pipeline inspection this is of particular interest, since many defect categories are precisely low-contrast and small in extent [190, 191] – the failure mode PGI targets.

4.2.7. YOLOv10

YOLOv10 introduces a consistent dual-assignment training strategy that removes the need for non-maximum suppression at deployment, alongside optimized feature extraction modules and refined detection heads [181]. Removing NMS from the inference path is what separates its efficiency gain from the other architectures in this chapter: the latency reduction comes from eliminating a post-processing step entirely and not from a faster backbone or a leaner neck, which is particularly relevant for continuous video streams and resource-constrained inspection hardware [192, 193].

4.2.8. YOLOv11

YOLOv11 refines backbone design and feature aggregation for more effective extraction across spatial scales, improving detection of small, partially occluded, and low-contrast objects while retaining deployability from workstation GPUs down to embedded platforms [182, 194]. Being the most recent architecture with a full generation of independent validation behind it when this dissertation's experiments were designed, YOLOv11 – and not the newer but less-validated YOLO26 (Section 4.2.9.) – serves as the single-stage baseline against which the cascade framework of Chapter 6. is compared [195, 196, 197].

4.2.9. YOLO26

YOLO26 is the most recent evolution of the family [198]. It targets deployment efficiency and inference latency on resource-constrained hardware, adopting an end-to-end framework that produces final predictions directly instead of through Non-Maximum Suppression, and adding training improvements aimed at convergence, small-object detection, and feature aggregation.

YOLO26 arrived after the experimental framework of this dissertation had been designed around the YOLOv4 to YOLOv11 range, which is also the range covered by most published pipeline inspection studies and therefore the range that permits direct comparison with prior work. It was nonetheless evaluated once it became available, on both sewer datasets used here. Neither evaluation showed it displacing the earlier generations: on the twelve-class Sewer Inspection Dataset it trailed YOLOv11 on precision, mAP@0.5,

Table 4.1: Comparison of YOLO architectures considered in this research.

Model	**Year**	**Backbone**	**Detection Type**	**Main Characteristics**
YOLOv4 [9]	2020	CSPDarknet53	Anchor-based	Introduced Bag of Freebies and Bag of Specials, PANet feature aggregation, Mish activation, and CIoU loss. Designed to achieve a balance between speed and accuracy.
YOLOv5 [184]	2020	CSP-based Backbone	Anchor-based	PyTorch implementation with scalable model variants (n, s, m, l, x), improved training pipeline, and efficient deployment.
YOLOv6 [186]	2022	EfficientRep	Anchor-free	Industrial-oriented architecture optimized for deployment efficiency, featuring Rep-PAN and hardware-aware design.
YOLOv7 [179]	2022	E-ELAN	Anchor-based	Introduced Extended Efficient Layer Aggregation Networks (E-ELAN) and trainable bag-of-freebies for improved trainability and accuracy.
YOLOv8 [183]	2023	CSP-based Backbone	Anchor-free	Anchor-free detection architecture with decoupled detection heads, improved feature fusion, and support for multiple vision tasks.
YOLOv9 [180]	2024	GELAN	Anchor-free	Introduced Programmable Gradient Information (PGI) and Generalized Efficient Layer Aggregation Network (GELAN) for enhanced feature learning.
YOLOv10 [181]	2024	Optimized Lightweight Backbone	End-to-End	NMS-free detection framework using a dual-assignment training strategy for reduced inference latency and improved efficiency.
YOLOv11 [182]	2024	Enhanced Feature Extraction Network	Anchor-free	Improved multi-scale feature representation, computational efficiency, and detection performance across diverse deployment scenarios. Unlike YOLOv10, it retains NMS post-processing.

and F1-score (Section 8.2.5.), and on the video sequence dataset it led the single-frame generations by a small margin while remaining well behind the spatiotemporal framework of Chapter 7. (Table 8.20). Both results are consistent with the wider finding of this dissertation that benchmark gains in the YOLO family transfer only weakly to pipeline inspection imagery.

4.2.10. Other YOLO-Based Architectures

Several alternative YOLO-based frameworks exist outside the main evolutionary branch: YOLO-NAS, which applies Neural Architecture Search to identify efficient configurations [199]; Gold-YOLO, which improves feature fusion and semantic exchange across layers [200]; PP-YOLOE, a collection of training and architectural optimizations that add no inference cost [201]; and DAMO-YOLO [202].

These architectures were not included in the experimental evaluation. They do not represent the main evolutionary branch of the YOLO family and are less frequently employed in the pipeline inspection literature, whereas YOLOv4–YOLOv11 are the versions against which published inspection studies report results [69]. Restricting the comparison to that range is what lets the results of Chapter 8. to be read against prior work, and the alternatives remain a direction for future research.

4.3. Detection pipeline

Although YOLO variants differ in architectural detail, they share a common pipeline: image preprocessing, feature extraction, feature aggregation, detection, and post-processing [7, 9, 179, 180]. Figure 4.1 illustrates it. The backbone extracts hierarchical feature representations from the input image, the neck aggregates multi-scale information, and the detection head generates localization and classification predictions at several feature scales.

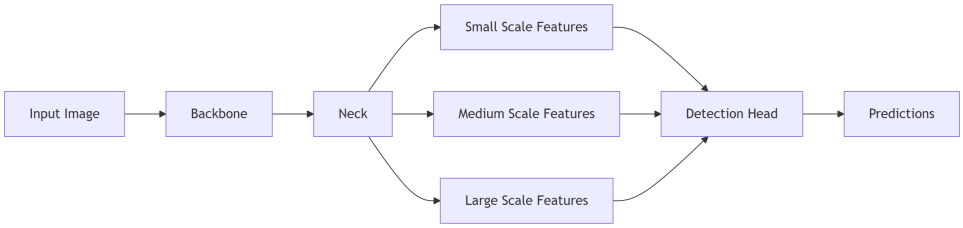


Figure 4.1: General processing pipeline of modern YOLO-based object detectors.

In the context of pipeline inspection, the detection pipeline must operate robustly in challenging scenarios characterized by varying illumination, motion blur, image degradation, underwater distortions, and highly imbalanced object distributions [4, 5, 129]. Understanding how each pipeline component behaves under these conditions, instead of treating the detector as a black box, is what informs the architecture and loss-function choices made in the remaining chapters of this dissertation. The following sections describe the major components of the object detection pipeline employed by modern YOLO architectures.

4.3.1. Image Preprocessing

Preprocessing prepares input images for the network and standardizes characteristics that otherwise vary widely across raw inspection footage: resolution, illumination, contrast, and image quality. Input images are resized to the fixed resolution the architecture expects, which enables efficient batch processing, and pixel values are normalized to a predefined range for numerical stability [25]. Colour-space conversion, histogram equalization, denoising, and contrast enhancement may be applied additionally, depending on the inspected environment.

Preprocessing is complemented by data augmentation, summarized in Table 4.2. Rotation, scaling, flipping, brightness adjustment, Mosaic [9], MixUp [102], and Copy-Paste [103] increase dataset diversity and improve generalization [203], which matters particularly in pipeline inspection because annotated defect samples are expensive to obtain.

Table 4.2: Common preprocessing and augmentation techniques used in object detection

Technique	**Purpose**
Resizing	Fixed network input dimensions
Normalization	Numerical stability and faster convergence
Histogram Equalization	Contrast enhancement
Rotation and Flipping	Orientation invariance
Scaling	Multi-scale robustness
Brightness Adjustment	Illumination robustness
Mosaic	Improved small-object detection
MixUp	Regularization and generalization
Copy-Paste	Minority-class enhancement

4.3.2. Feature Extraction (Backbone)

The backbone transforms raw pixel information into hierarchical feature representations: early layers learn edges, corners, and textures, deeper layers capture semantic information about object shape, structure, and context [26]. Figure 4.2 illustrates the process.

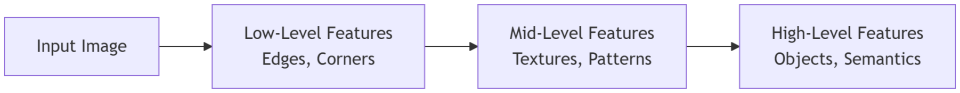


Figure 4.2: Hierarchical Feature Extraction in CNN Backbones

Modern YOLO architectures employ specialized backbone networks designed to maximize feature extraction efficiency while maintaining manageable computational requirements. Examples include CSPDarknet53 in YOLOv4 [9], EfficientRep in YOLOv6 [186],

E-ELAN in YOLOv7 [179], and various CSP-inspired architectures in later YOLO versions [180, 181]. These backbones incorporate residual connections, feature reuse mechanisms, and efficient convolutional operations to improve gradient flow, representation learning, and computational efficiency [26].

Backbones are commonly pre-trained on large-scale datasets and fine-tuned for the target domain, which improves convergence since every later pipeline stage operates on the features this network produces. Extraction quality matters more here than in general-purpose detection, because pipeline defects occupy small image regions with low contrast under poor illumination, motion blur, turbidity, and sediment. This is why the backbone comparison across YOLO generations in Chapter 8. is reported separately for the underwater and sewer datasets.

4.3.3. Feature Aggregation (Neck)

Backbone features originate at different spatial resolutions and semantic levels, and the neck aggregates them so that the detector can handle objects of varying size (centre of Figure 4.1). Common implementations combine low-level spatial with high-level semantic information hierarchically: Feature Pyramid Networks [204], Path Aggregation Networks [153], Bi-directional Feature Pyramid Networks [205], and the enhanced mechanisms of recent YOLO versions [9, 179].

Successive YOLO versions have introduced increasingly sophisticated aggregation mechanisms for this reason: PANet in YOLOv4, enhanced fusion in YOLOv7, and further-optimized modules in later variants [9, 179, 180]. Multi-scale fusion matters in infrastructure inspection because defect size varies so widely. Small cracks, joint defects, and localized corrosion need detailed spatial information, whereas large structural deformations benefit from broader context, and in underwater and sewer footage the same defect appears at different scales depending on camera distance.

4.3.4. Detection Head

The detection head represents the final prediction stage of the object detection network. Using the aggregated feature maps produced by the neck, the detection head performs object localization and classification simultaneously. For each potential object

instance, the network predicts the location of a bounding box, the corresponding object class, and a confidence score indicating the likelihood of a valid detection (rightmost part of Figure 4.1). The outputs generated by the detection head are subsequently processed to produce the final object detections presented to the user [7, 8].

Earlier YOLO architectures used anchor-based detection mechanisms, where predefined anchor boxes served as reference templates during localization [8, 9]. More recent architectures, including YOLOv8 and subsequent versions, employ anchor-free detection strategies that simplify training and reduce dependence on manually selected anchor configurations. Many modern architectures use decoupled detection heads that separate classification and localization tasks, allowing each task to be optimized independently and improving overall detection performance [181, 183].

Recent developments have further enhanced detection heads through the introduction of advanced label assignment strategies, improved bounding-box regression mechanisms, and end-to-end detection pipelines. These improvements aim to increase localization accuracy, reduce false-positive detections, and improve computational efficiency while maintaining real-time performance [180, 181].

The quality of the detection head directly affects both localization accuracy and classification performance. In pipeline inspection applications, accurate localization is essential for identifying the precise position and extent of a defect, while reliable classification is what turns a detection into an actionable maintenance record instead of an unlabelled bounding box.

4.3.5. Post-Processing

Dense prediction produces multiple overlapping detections of the same object, so post-processing is required to remove redundant predictions and retain only the most reliable ones [7, 28].

Traditionally, Non-Maximum Suppression (NMS) has been the most widely used post-processing technique in object detection. NMS selects the highest-confidence prediction among overlapping detections and suppresses neighboring predictions whose overlap exceeds a predefined threshold [206]. This process substantially reduces duplicate detections while preserving localization accuracy and has become a standard component of most

object detection frameworks, including many generations of the YOLO family [9, 179]. Variants such as Soft-NMS have also been proposed to mitigate the risk of suppressing valid detections in crowded scenes by reducing confidence scores instead of removing overlapping predictions outright [207].

In practical object detection systems, post-processing is typically controlled through confidence-score and Intersection-over-Union (IoU) thresholds. Confidence thresholds are used to filter out low-probability predictions, while IoU thresholds determine the degree of overlap permitted between neighboring detections before suppression occurs. The selection of these thresholds can substantially influence precision, recall, and overall detection performance, particularly when detecting small objects, visually similar targets, or defects with ambiguous boundaries [208, 209].

Recent object detection architectures have explored end-to-end detection frameworks that eliminate the need for explicit post-processing operations. Transformer-based detectors such as DETR demonstrated that object detection can be formulated as a direct set-prediction problem, thereby removing the requirement for NMS during inference [86]. More recently, architectures such as YOLOv10 and YOLO26 have adopted end-to-end detection strategies that directly generate final predictions without conventional post-processing procedures [181, 198]. Such approaches aim to reduce inference latency, simplify deployment pipelines, and improve computational efficiency while maintaining competitive detection performance.

Post-processing nonetheless remains a component of most practical systems, and its configuration carries real consequences here. Pipeline inspection datasets frequently contain visually similar defects in close spatial proximity – cracks, deposits, corrosion patterns, joint defects, and deformations – so suppression must be balanced against the risk of discarding a valid, closely spaced instance, particularly given that the resulting defect reports drive maintenance decisions [4, 5, 129]. The challenge is more pronounced in video, where temporal consistency between consecutive frames also bears on reliability and is treated separately in Chapter 7..

4.4. Attention Mechanisms

Attention mechanisms assign greater weight to informative features and reduce the influence of irrelevant or noisy responses, recalibrating feature maps along the channel dimension, the spatial dimension, or both [72, 73, 74, 210]. This matters in pipeline inspection because defects are often small, partially occluded, or hard to separate from the surrounding surface under poor illumination, motion blur, turbidity, or deposits, conditions in which standard convolutional features may not discriminate defect from background [4, 5, 129]. This section covers the three mechanisms most often used for this purpose – Squeeze-and-Excitation (SE) [72], the Convolutional Block Attention Module (CBAM) [73], and Efficient Channel Attention (ECA) [74] – whose structures are illustrated in Figure 4.3 and which have all been incorporated into YOLO-based architectures [27, 78, 79].

All three are evaluated experimentally on the twelve-class Sewer Inspection Dataset in Section 8.2.9., where none is found to change detection performance materially, although they redistribute precision and recall in the directions their mechanisms predict and ECA emerges as the best of the three. SE additionally enters the experimental work indirectly, through the squeeze-and-excitation blocks internal to the MobileNetV3-Large backbone selected for the cascade gate (Section 6.3.2.).

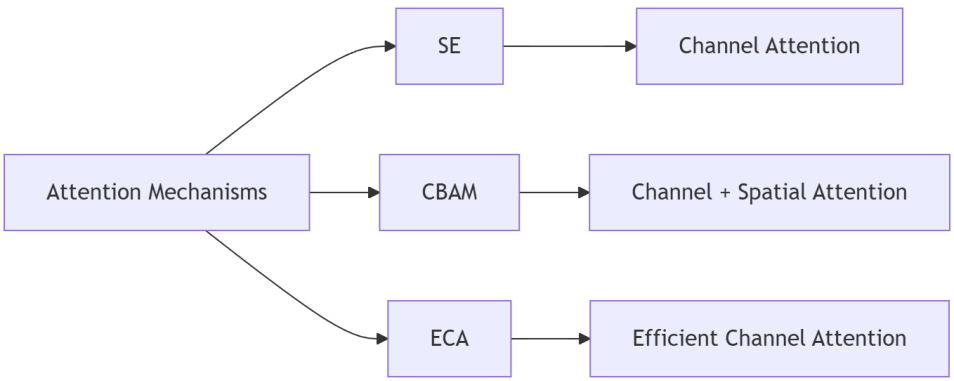


Figure 4.3: Overview of channel and spatial attention mechanisms (SE, CBAM, ECA) applied to a convolutional feature map.

4.4.1. Squeeze-and-Excitation Network (SE)

The Squeeze-and-Excitation (SE) module [72] is a channel attention mechanism that learns channel-wise weights instead of treating every channel as equally important, on the observation that individual feature channels contribute differently to a recognition task. It consists of two operations (Figure 4.3a). The squeeze stage applies global average pooling to aggregate each channel's spatial information into a compact descriptor; the excitation stage uses a lightweight fully connected network to learn channel importance weights, which then recalibrate the original feature maps by channel-wise multiplication.

Let $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ denote an input feature map with height H, width W, and C channels. The squeeze operation generates a channel descriptor by applying global average pooling:

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_c(i, j) \tag{4.1}$$

where z_c represents the aggregated response of channel c. The excitation operation then learns channel weights using two fully connected layers and nonlinear activation functions:

$$\mathbf{s} = \sigma(W_2 \delta(W_1 \mathbf{z})) \tag{4.2}$$

where $\delta(\cdot)$ denotes the ReLU activation function, $\sigma(\cdot)$ denotes the sigmoid activation function, and W_1 and W_2 are learnable weight matrices. The resulting vector $\mathbf{s}$ contains channel importance weights that are applied to recalibrate the original feature maps.

Strengthening informative channels improves discriminative power, which in detection tasks tends to help most where objects are visually subtle or occupy a small portion of the image [72].

SE modules have been incorporated into numerous object detection architectures, including several YOLO-based models, to enhance feature extraction and improve the detection of small or visually ambiguous targets. Because SE recalibrates channels only, it adds relatively few parameters for the accuracy it provides, which makes it a common first attention module to try when a detector needs to better separate small, low-contrast defects from the background without materially increasing model size [69, 211, 212, 213].

4.4.2. Convolutional Block Attention Module (CBAM)

The Convolutional Block Attention Module (CBAM), introduced by Woo et al. [73], combines channel and spatial attention within a single module. While channel attention determines which feature channels contain the most relevant information, spatial attention identifies where this information is located within the feature map. This allows CBAM to refine features along both dimensions and distinguish target objects from surrounding background content.

As illustrated in Figure 4.3b, CBAM applies channel attention followed by spatial attention. The channel attention module uses global average pooling and global max pooling to summarize the input feature map. The resulting descriptors are processed by a shared multilayer perceptron to estimate channel importance weights, which are then applied to the original feature map.

The channel-refined output then passes to the spatial attention module, where average and maximum pooling along the channel dimension produce spatial descriptors that a convolutional layer turns into a spatial attention map emphasising informative image regions.

Let $\mathbf{F} \in \mathbb{R}^{H \times W \times C}$ denote an input feature map. The channel attention map $M_c(\mathbf{F})$ is applied to obtain the channel-refined feature representation:

$$\mathbf{F}' = M_c(\mathbf{F}) \otimes \mathbf{F} \tag{4.3}$$

where $\otimes$ denotes element-wise multiplication. The resulting feature map is subsequently processed by the spatial attention module:

$$\mathbf{F}'' = M_s(\mathbf{F}') \otimes \mathbf{F}' \tag{4.4}$$

where $M_s(\mathbf{F}')$ represents the spatial attention map. The final output feature map $\mathbf{F}''$ contains feature responses that have been refined both channel-wise and spatially, enabling improved representation of relevant visual information [73].

The combination of channel and spatial attention allows CBAM to provide a more detailed feature refinement strategy than channel-only approaches such as SE, at the cost of the additional spatial-attention convolution. This trade-off matters most for defects

that are low-contrast and also spatially small or embedded in cluttered backgrounds, such as joint displacements seen against biofouling or cracks seen against sediment, where knowing *where* to look matters as much as knowing *which channels* to trust [73, 211, 213].

4.4.3. Efficient Channel Attention (ECA)

Efficient Channel Attention (ECA) [74] was motivated by the observation that the dimensionality-reduction bottleneck SE relies on – the W_1, W_2 pair in its excitation step – can itself hinder the learning of channel dependencies. ECA therefore eliminates both the reduction and the fully connected layers, capturing local cross-channel interactions with a lightweight one-dimensional convolution instead. This preserves richer channel representations at substantially fewer parameters; the information flow is shown in Figure 4.3c.

Let $\mathbf{z} \in \mathbb{R}^C$ denote the channel descriptor obtained through global average pooling. Instead of employing fully connected layers, ECA applies a one-dimensional convolution operation to model local cross-channel interactions:

$$\mathbf{s} = \sigma(\text{Conv1D}_k(\mathbf{z})) \tag{4.5}$$

where Conv1D_k denotes a one-dimensional convolution with kernel size k and $\sigma(\cdot)$ represents the sigmoid activation function. The resulting attention weights are subsequently applied to recalibrate the original feature maps through channel-wise multiplication. This formulation enables efficient learning of channel relationships while maintaining a lightweight architecture [74].

ECA also selects its kernel size adaptively, as a function of the channel dimension instead of a fixed value, so that the module captures appropriate local interactions for feature maps of different sizes [74]. ECA therefore sits at a specific point in the effectiveness/efficiency trade-off relative to the other two modules: fewer parameters than SE because it drops the fully connected bottleneck, and lower overhead than CBAM because it never computes a spatial-attention map at all. On structural grounds this makes it the module best suited to onboard deployment on ROVs and embedded inspection hardware [74, 129, 213]. Table 4.3 summarises the three mechanisms side by side.

The measurements in Section 8.2.9. qualify this comparison. ECA gives the best

Table 4.3: Comparison of attention mechanisms commonly used in object detection architectures.

Module	Attention Type	Additional Parameters	Computational Cost	Main Characteristics
SE [72]	Channel	Medium	Medium	Models channel dependencies through squeeze and excitation operations. Improves feature discrimination but introduces additional fully connected layers.
CBAM [73]	Channel + Spatial	High	High	Sequentially applies channel and spatial attention. Provides detailed feature refinement and improved localization of relevant image regions.
ECA [74]	Channel	Low	Low	Uses lightweight one-dimensional convolutions to model channel interactions without dimensionality reduction. Maintains strong performance with minimal computational overhead.

mAP@0.5 of the three on the sewer dataset while remaining the cheapest, but its margin over the unmodified baseline is half a percentage point, so the choice between the three modules matters considerably less on this task than the decision to spend the effort elsewhere.

4.5. Loss Functions

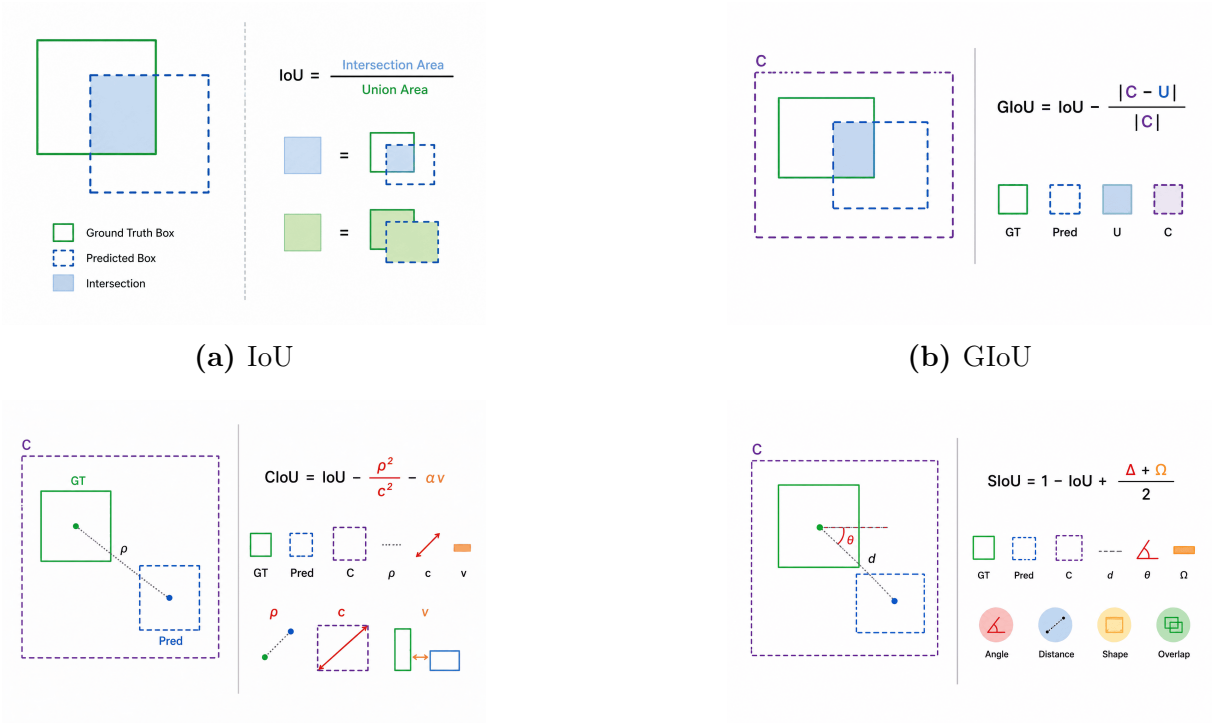
Detection performance depends not only on architecture but on the optimization objective, defined through loss functions that quantify the discrepancy between predictions and ground-truth annotations [214].

In object detection the overall loss combines a localization term, which measures the accuracy of predicted bounding boxes, a classification term, which evaluates class predictions, and a confidence or objectness term [7, 28]. Localization was historically performed with coordinate-based regression losses such as Mean Squared Error (MSE) and Smooth L1, which correlate weakly with the evaluation metrics used in detection benchmarks, particularly when predicted and ground-truth boxes overlap poorly [28]. The family of Intersection over Union (IoU)-based losses was developed to address this by optimizing box overlap directly, and now spans IoU, Generalized IoU (GIoU), Distance IoU (DIoU), Complete IoU (CIoU), and Scylla IoU (SIoU), each adding a geometric constraint the previous formulation lacked [215, 216, 217].

Classification losses matter equally. Early detectors used cross-entropy; recent architectures use Binary Cross-Entropy (BCE), Focal Loss, Quality Focal Loss, and VariFocal Loss to address class imbalance and learn from difficult examples [11, 218]. Both choices carry more weight in pipeline inspection than in general-purpose detection, because defect datasets combine severe class imbalance with small, low-contrast targets seen under poor illumination and background clutter [4, 129].

4.5.1. Intersection over Union (IoU)

Intersection over Union (IoU) quantifies the overlap between a predicted bounding box and its ground-truth annotation as the ratio of their intersection area to their union area, and is the standard evaluation metric in benchmarks such as PASCAL VOC and



(a) IoU **(b)** GIoU

(c) CIoU **(d)** SIoU

Figure 4.4: Geometry of the IoU-based localization losses: (a) the intersection and union areas from which IoU is computed; (b) the smallest enclosing box C supplying the GIoU penalty; (c) the centre-point distance and aspect-ratio terms added by CIoU; (d) the angle, distance, and shape components of SIoU.

MS COCO [208, 209].

The IoU metric is defined as:

$$IoU = \frac{A_{intersection}}{A_{union}} \tag{4.6}$$

where $A_{intersection}$ denotes the area shared by both bounding boxes and A_{union} represents the total area covered by either box. The IoU value ranges between 0 and 1, where a value of 0 indicates no overlap and a value of 1 corresponds to perfect alignment between the predicted and ground-truth bounding boxes. Figure 4.4a illustrates the overlap geometry on which this ratio is computed.

Detections are conventionally counted as correct when IoU exceeds a threshold such as 0.5 or 0.75. Used directly as a loss, however, IoU has a decisive weakness: when two boxes do not overlap it evaluates to zero and yields no gradient, so the optimizer cannot determine how the predicted box should move. It also ignores centre distance, aspect-ratio difference, and any geometric relationship between non-overlapping boxes [215].

These limitations motivated the development of more advanced IoU-based loss formulations, including Generalized Intersection over Union (GIoU), Distance Intersection over Union (DIoU), Complete Intersection over Union (CIoU), and Scylla Intersection over Union (SIoU), each introducing additional geometric constraints that improve optimization behaviour and localization accuracy [215, 216, 217]. IoU itself is rarely used as the primary localization loss in contemporary detectors, but it remains the conceptual foundation on which those formulations are built.

4.5.2. Generalized Intersection over Union (GIoU)

Generalized Intersection over Union (GIoU) [215] addresses the zero-gradient case by adding a geometric penalty derived from the smallest enclosing box containing both the predicted and ground-truth boxes, which lets the loss encode their relative positioning even when they do not intersect.

The GIoU metric is defined as:

$$GIoU = IoU - \frac{|C - U|}{|C|} \tag{4.7}$$

where C denotes the smallest enclosing box covering both the predicted and ground-truth bounding boxes, and U represents their union. The second term measures the normalized area outside the union but inside the enclosing box and acts as a penalty for poor localization.

Figure 4.4b shows the enclosing box C from which this penalty is derived.

Unlike standard IoU, GIoU produces informative gradients even when the predicted and ground-truth bounding boxes do not intersect, so optimization can continue guiding the predicted box toward its target throughout training. It also remains scale-invariant, and that is what allowed it to replace IoU as a localization loss without further modification [215].

Its limitation is that the penalty term is derived from the enclosing box area instead of directly considering the distance between box centres or differences in aspect ratio. Optimization may therefore remain slow where bounding boxes overlap but are poorly aligned, which motivated the subsequent formulations described below [216].

4.5.3. Complete Intersection over Union (CIoU)

Complete Intersection over Union (CIoU) [216] extends these formulations with information about object positioning and shape, on the observation that two boxes with identical IoU can still differ substantially in centre position and aspect ratio, so overlap alone is an incomplete objective.

The CIoU loss consists of three complementary components:

– overlap area between bounding boxes,

– Euclidean distance between box centers,

– aspect ratio consistency.

The corresponding formulation is expressed as:

$$CIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2} - \alpha v \tag{4.8}$$

where $\rho(b, b^{gt})$ denotes the Euclidean distance between the centers of the predicted bounding box b and the ground-truth bounding box b^{gt}, c represents the diagonal length

of the smallest enclosing box covering both bounding boxes, v measures aspect ratio consistency, and α is a weighting factor controlling the influence of the aspect-ratio term [216].

The distance and aspect-ratio terms are illustrated in Figure 4.4c.

The distance term encourages the predicted bounding box to move directly toward the target location, while the aspect-ratio term promotes geometric consistency between the predicted and ground-truth object shapes. Jointly optimizing overlap, distance, and shape yields faster convergence and more accurate localization than earlier formulations, particularly for elongated or irregularly shaped objects [216], and CIoU is consequently the default localization loss of multiple YOLO generations including YOLOv4 [9].

CIoU nonetheless relies entirely on geometric relationships between boxes and does not model directional information during optimization, which motivated Scylla Intersection over Union (SIoU) [217].

4.5.4. Scylla Intersection over Union (SIoU)

Scylla Intersection over Union (SIoU) [217] adds directional information. Conventional IoU-based losses do not guide the direction in which a predicted box should move, so optimization may explore unnecessary regions of the search space before converging; SIoU introduces an angle-aware strategy that steers the box toward its target along a more efficient path.

The SIoU formulation consists of four complementary components:

- angle cost,
- distance cost,
- shape cost,
- overlap cost.

The overall SIoU loss can be expressed as:

$$L_{SIoU} = 1 - IoU + \frac{\Delta + \Omega}{2} \tag{4.9}$$

where Δ represents the distance cost and Ω denotes the shape cost. The angle cost influences the computation of the distance term and guides the optimization trajectory toward the target location [217].

Figure 4.4d shows the angle, distance, and shape components together.

The angle cost is the principal innovation. By considering the angle between the predicted and ground-truth bounding boxes, the optimization process reduces the search space and accelerates convergence, which improves localization accuracy particularly during the early stages of training. SIoU is the only loss reviewed here that models optimization direction and not overlap, distance, and shape alone, and this is the property most relevant to the elongated, irregularly shaped defects – cracks, joint displacements – that motivate this dissertation's interest in IoU variants [129, 217].

4.5.5. Combined Detection Loss

Modern YOLO architectures employ a composite loss function that integrates localization, classification, and confidence components into a unified optimization objective. The total detection loss can generally be expressed as:

$$L_{total} = \lambda_{box} L_{box} + \lambda_{cls} L_{cls} + \lambda_{obj} L_{obj} \tag{4.10}$$

where L_{box} represents the localization loss, L_{cls} corresponds to the classification loss, and L_{obj} denotes the objectness or confidence loss. The weighting coefficients λ_{box}, λ_{cls}, and λ_{obj} control the relative importance of each component during optimization.

The selection of an appropriate loss formulation is central to achieving accurate and robust object detection, and both components of the composite objective are investigated experimentally in this dissertation. The classification term L_{cls} is treated in Chapter 5. and Section 8.3., which evaluate focal, weighted, and class-balanced formulations for their effect on rare defect classes. The localization term L_{box} is treated in Section 8.2.8., which substitutes GIoU, DIoU, and SIoU for the CIoU default of YOLOv7 and finds the four losses to order themselves as the geometric argument predicts, though across a span of well under one percentage point of mAP@0.5. Table 4.4 summarises the properties on which that ordering rests.

Table 4.4: Comparison of IoU-based localization loss functions.

Loss Function	**Considers**	**Optimization Quality**	**Main Characteristics**
IoU	Overlap Area	Low	Measures overlap between predicted and ground-truth bounding boxes. Fails to provide meaningful gradients when boxes do not overlap.
GIoU	Overlap + Enclosing Area	Medium	Introduces a penalty based on the smallest enclosing box, enabling optimization even when no overlap exists.
DIoU	Overlap + Distance	Medium	Adds a normalised centre-point distance term, giving a direct optimisation direction and faster convergence than GIoU; the basis on which CIoU builds.
CIoU	Overlap + Distance + Aspect Ratio	High	Considers center-point distance and aspect ratio consistency, resulting in faster convergence and improved localization accuracy.
SIoU	Overlap + Distance + Shape + Angle	Very High	Introduces angle-aware optimization to reduce regression complexity and improve convergence efficiency, particularly for small or irregular objects.

4.6. Chapter Summary

This chapter set out the detection machinery the remainder of the dissertation builds on: the YOLO family from YOLOv4 to the most recent releases, collected architecturally in Table 4.1, and the generic detection pipeline decomposed into preprocessing, feature extraction, aggregation, detection head, and post-processing, so that later modifications can be located precisely within it.

Two families of modification received closer attention because both define the design space the later chapters select from. Attention modules (SE, CBAM, ECA) sharpen feature representations at modest cost; all three are tested in Section 8.2.9. and none changes performance materially, though ECA is the best of them. IoU-based localization losses progressively add geometric constraints that matter most when boxes overlap poorly, the common case for small, low-contrast defects; Section 8.2.8. finds SIoU 0.3 points ahead of the CIoU default. Both families therefore produce measurable but second-order effects. The loss-level intervention with a first-order effect acts on the classification term, and is the subject of the next chapter.

Chapter 5. takes up the first of the domain-specific obstacles identified in Chapter 1., namely the severe class imbalance of inspection datasets.

5. Chapter

CLASS IMBALANCE MITIGATION FOR PIPELINE INSPECTION

In practical inspection environments some defect categories occur far less frequently than others, producing skewed datasets that degrade training and detection performance. This chapter examines the effect of that imbalance and the three families of mitigation evaluated in this dissertation: data augmentation, dataset balancing, and loss function adaptation.

5.1. Class Imbalance in Pipeline Inspection Datasets

Class imbalance arises when the number of training samples differs substantially between object categories, biasing the learning process toward frequently occurring classes [11, 219]. It is unavoidable in pipeline inspection. Benchmark computer vision datasets are curated toward balanced class distributions, whereas inspection datasets are collected during routine maintenance operations, so their defect distributions reflect the actual condition of the infrastructure, not the requirements of the learning algorithm. Common anomalies such as deposits, sediment accumulation, corrosion products, and fouling recur throughout inspection recordings, while cracks, fractures, joint displacements, intrusions, and collapses – the most critical maintenance concerns – are comparatively rare [4, 5]. Rare defects also cannot simply be collected in greater numbers, since they occur infrequently in operational networks, so the resulting datasets are reliably long-tailed

[109, 129, 220].

The 13,000-image, twelve-class Sewer Inspection Dataset used throughout this chapter (Table 3.4) illustrates the problem. Occurrence frequencies vary by two orders of magnitude, and vermin (BBH) and obstructions (BBE) are both the least represented and, as Chapter 8. shows, the classes benefiting most from the loss adaptations investigated below. The same asymmetry appears in the underwater dataset, where the pipeline itself dominates and supporting components are rare. All experiments in this chapter use the twelve-class dataset, not the ten-class video sequence extraction of Section 3.6..

From an object detection perspective, severe class imbalance affects both classification and localization performance. During optimization, majority classes contribute disproportionately to the overall loss function, causing the model to allocate more representational capacity toward frequently occurring categories [11, 219]. While this often results in strong overall accuracy and mean average precision (mAP), detection performance for minority classes frequently remains unsatisfactory. In practical infrastructure inspection applications, such behavior is undesirable because the failure to detect a rare but critical defect may have significantly greater consequences than the incorrect classification of a common anomaly.

Numerous mitigation approaches have been proposed, including data augmentation, oversampling and undersampling strategies, class-aware sampling, synthetic data generation using generative adversarial networks [221], and specialized loss functions. The following sections examine the characteristics of class imbalance within the datasets considered in this dissertation and review the principal techniques employed to improve detection of underrepresented defect categories.

5.1.1. Distribution of Defect Classes

The imbalance ratio, the relationship between the most and least frequent classes, provides a quantitative measure of how skewed a dataset is, and high ratios correlate with increased false-negative rates on minority classes. Figure 5.1 shows the distribution for the sewer dataset. The disparities it exposes are what the augmentation, sampling, and loss-function techniques evaluated in the rest of this chapter are intended to address.

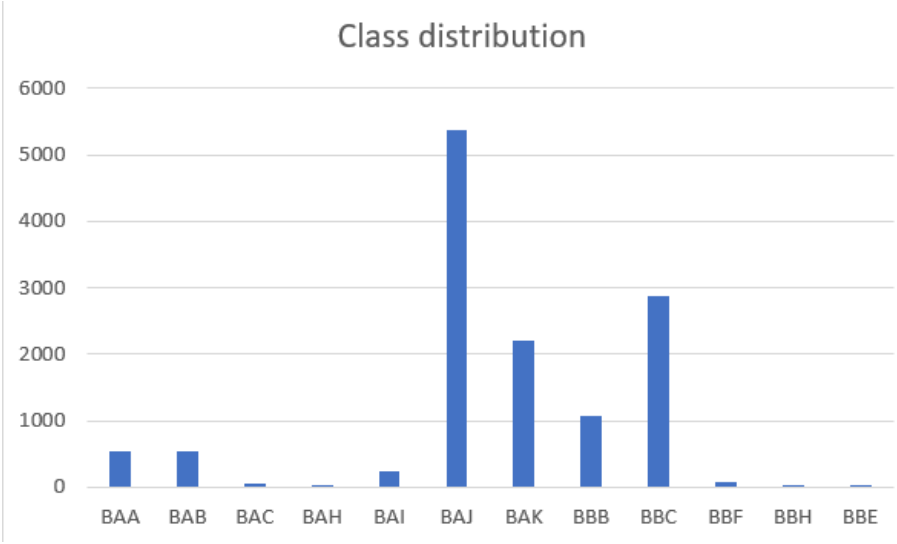


Figure 5.1: Class distribution of the sewer inspection dataset, illustrating the long-tail imbalance between frequently occurring and rare defect categories.

5.1.2. Impact of Class Imbalance on Object Detection

Because majority classes contribute a substantially larger proportion of training instances, they dominate the gradient updates used to optimize network parameters, and the model allocates a greater portion of its representational capacity to them [11, 219]. The result is an apparent contradiction between aggregate and class-specific performance: a model may reach high overall precision, recall, and mAP while detecting rare defects poorly. A detector optimized for dominant classes can therefore appear successful by aggregate metrics while failing the inspection requirement that motivated it.

The effect is not confined to classification. Insufficient exposure to minority-class examples also limits the geometric representations the model can learn for them, so bounding-box predictions for rare classes show reduced localization precision and greater sensitivity to illumination changes, image degradation, occlusion, and background clutter – conditions that are routine in underwater and sewer inspection [4, 129]. Detectors additionally become more confident on majority classes, so confidence thresholding and Non-Maximum Suppression suppress minority-class detections preferentially at post-processing, compounding the problem at inference time as well as during training.

For these reasons class-wise evaluation is recommended over aggregate metrics alone when assessing detectors trained on long-tailed data [220], and the mitigation strategies

examined in the following sections are evaluated in Chapter 8. on per-class as well as aggregate measures.

5.1.3. Challenges Associated with Rare Defects

Rare defects compound three difficulties that are individually manageable. The first is sample scarcity: with only a handful of annotated instances, a network cannot capture the full variability of a defect class, so its performance becomes highly sensitive to changes in appearance, scale, illumination, and viewing angle [214, 219, 220]. That variability is itself unusually wide here. Structural cracks differ in length, width, orientation, and severity; joint displacements may be partially obscured by deposits, corrosion products, or sediment; and underwater defects are further affected by scattering, attenuation, colour distortion, and biofouling [109, 129]. Intra-class variance is therefore high precisely where the sample count is lowest.

The second is object size. Fine cracks, localized fractures, water ingress points, and minor deformations occupy only a small number of pixels, so the detector must solve class imbalance and small-object detection simultaneously, two problems each known to degrade performance on their own [11, 104].

The third is annotation uncertainty. Few examples mean less opportunity to establish consistent labelling criteria for a class, so rare categories carry both more label noise and greater sensitivity to each individual error.

None of this would matter much if rare classes were also unimportant, but the reverse holds: structural fractures, major deformations, joint separations, and collapses are simultaneously the rarest categories and the ones that drive maintenance decisions [4, 5]. This is also why aggregate metrics mislead here, and why class-specific precision, recall, and average precision, together with rare-class recall and rare-class AP, are the measures used to assess mitigation techniques in Chapter 8. [208, 209].

Figure 5.2 summarizes how this problem compounds across the training pipeline. A rare defect contributes only a handful of annotated samples, which limits the feature representations the detector can learn for that class; the resulting model then produces disproportionately more false negatives than true positives for that category, and the defect is missed during inspection despite being present in the frame.

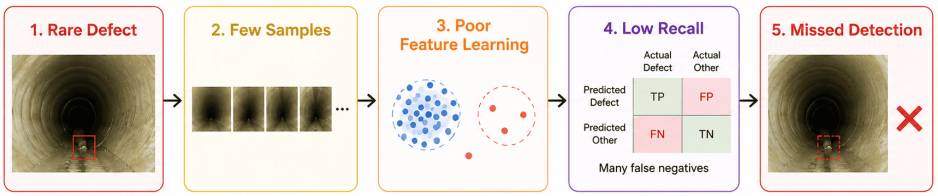


Figure 5.2: From rarity to missed detection: a rare defect class contributes few training samples, leading to poor feature learning, low recall, and an increased false-negative rate at inference time.

Addressing the challenges associated with rare defects requires strategies capable of increasing the effective representation of minority classes during training while preserving the overall characteristics of the dataset. Data augmentation, synthetic sample generation, class-aware sampling, oversampling techniques, and adaptive loss functions represent promising approaches for improving the detection of rare defects and enhancing the practical applicability of automated pipeline inspection systems. The following sections examine these approaches in detail and evaluate their suitability for imbalanced pipeline inspection datasets.

5.2. Data Augmentation Techniques

Data augmentation increases the diversity and quantity of training samples by generating modified versions of existing data while preserving their semantic content, which reduces overfitting and improves generalization [203].

Augmentation matters more in pipeline inspection than in general-purpose detection for two reasons. Acquiring and annotating inspection data is expensive, since campaigns require specialized equipment, trained personnel, and manual coding, so the dataset cannot simply be enlarged when a class is underrepresented [4, 129]. And the variations that augmentation simulates – changes in illumination, viewing angle, turbidity, sediment, reflection, motion blur, and object scale – are exactly the ones that degrade detection in this domain, so the synthetic samples are realistic as well as diverse [9, 104].

Augmentation techniques divide into conventional image transformations, which modify existing images geometrically or photometrically, and sample synthesis approaches, which generate new examples by combining information from several images or by con-

structing minority-class samples outright [203]. This research evaluates geometric and photometric transformations together with three synthesis techniques: MixUp [102], Mosaic [9], and Copy-Paste [103]. The Synthetic Minority Over-sampling Technique (SMOTE) [101] is described alongside them: it was implemented and run on this dissertation's data, but its output was assessed as unsuitable for training by an inspection professional and excluded from every experiment, for the reasons set out in Section 5.2.6..

5.2.1. Geometric Transformations

Geometric transformations modify the spatial arrangement of image content while preserving the semantic characteristics of the depicted objects, and include rotation, scaling, translation, flipping, cropping, shearing, and perspective modification [203]. Presenting objects under different spatial configurations teaches the detector to recognize features independently of their exact position, orientation, or scale, which reduces overfitting [222]. In object detection the corresponding bounding-box coordinates must be transformed alongside the image, which modern frameworks handle automatically.

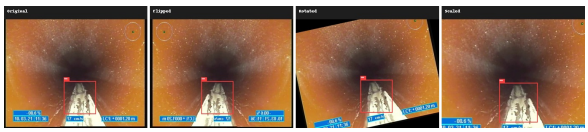
These transformations suit pipeline inspection because inspection cameras mounted on crawler robots, CCTV systems, and ROVs capture defects from continuously varying perspectives as they advance through the pipe [4, 129]. They also partially alleviate class imbalance, since rotated, translated, or scaled versions of a rare defect provide additional examples for categories represented by only a handful of annotated instances.

Geometric augmentation is not free of risk. An aggressive rotation or crop can push a defect partially out of frame, or distort a crack's orientation in a way that no longer matches how that defect forms in a pipe wall, so rotation range, crop ratio, and shear angle need bounds reflecting plausible camera motion, not the widest range the library allows [9, 203]. Figure 5.3a illustrates the transformations applied here.

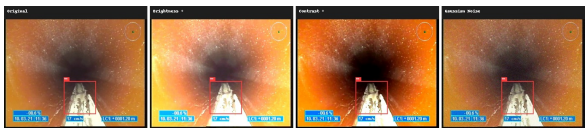
5.2.2. Photometric Transformations

Photometric transformations adjust brightness, contrast, saturation, hue, sharpness, colour balance, and noise without changing the spatial structure of the depicted objects, improving robustness to variations in imaging conditions [203].

Exposing the detector to such variations encourages invariant feature representations



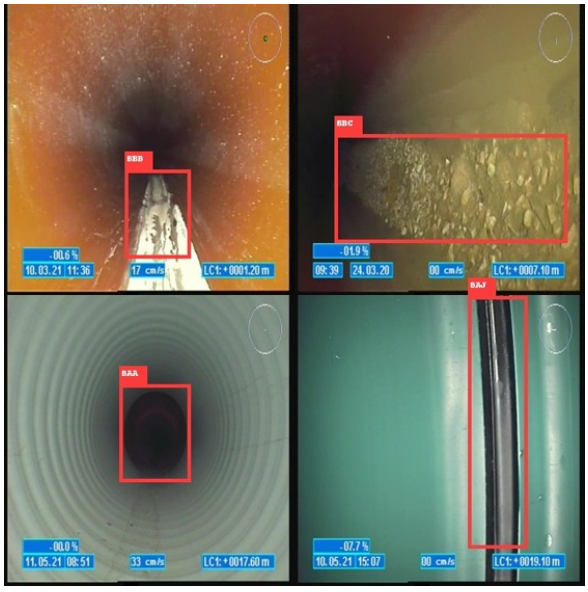
(a) Geometric



(b) Photometric



(c) MixUp



(d) Mosaic



(e) Copy-Paste

Figure 5.3: Augmentation techniques applied to real annotated frames from the sewer inspection dataset. (a) horizontal flip, rotation, and scaling of a BBB (wall attachments) frame, with bounding boxes transformed accordingly; (b) brightness increase, contrast increase, and additive Gaussian noise on the same frame; (c) two frames (BBB and BBC) linearly interpolated at $\lambda = 0.5$; (d) four frames (BBB, BBC, BAA, BAJ) tiled into one composite; (e) a BAK (defective lining) instance extracted from its source frame, rescaled, and inserted into a frame already containing a BAA (deformation) annotation.

[222], and this matters more here than in most domains. Sewer imagery suffers from non-uniform illumination, shadows, reflections, moisture, deposits, and CCTV sensor noise; underwater imagery adds light attenuation, scattering, colour distortion, suspended particles, and turbidity [109, 129]. The same defect can therefore look substantially different from one run to the next, and controlled photometric variation during training teaches the model to key on defect-related features instead of the environmental artefacts that happen to accompany them.

Common photometric transformations include brightness adjustment, contrast modification, colour jittering, gamma correction, Gaussian noise injection, blurring, and histogram-based enhancement, and are usually combined with geometric transformations and synthesis methods instead of being applied alone [9, 203]. Figure 5.3b illustrates those applied here.

5.2.3. MixUp Augmentation

MixUp [102] generates new training samples by linearly interpolating pairs of images and their labels, creating synthetic examples that lie between existing samples in feature space.

Given two training samples (x_i, y_i) and (x_j, y_j), a new synthetic sample is generated according to:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \tag{5.1}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{5.2}$$

where $\lambda \in [0, 1]$ is sampled from a Beta distribution and controls the contribution of each sample to the resulting image and label representation [102].

Exposing the network to intermediate representations between classes encourages smoother decision boundaries, reduces overconfidence, and improves robustness to label noise [9, 102]. MixUp is, however, the least targeted at class imbalance of the three synthesis techniques evaluated here: it raises sample diversity globally instead of boosting minority-class exposure selectively, and that is why Mosaic and Copy-Paste produce

larger gains on the rarest categories.

Its drawback is specific to localization. Blending two frames at $\lambda \approx 0.5$ leaves both sets of defect edges semi-transparent, so a bounding box is drawn against a worse edge than either source image offered. This is why MixUp is used here alongside Mosaic and Copy-Paste [9]. Figure 5.3c shows an example.

5.2.4. Mosaic Augmentation

Mosaic augmentation, introduced with YOLOv4 [9], combines four independent training images into a single composite, so the detector observes multiple object instances, backgrounds, and contexts in one training iteration.

Because objects from the four source images are resized and repositioned within the composite, the detector sees a broader range of object scales and becomes less dependent on any particular background. This makes Mosaic especially effective for small objects [104], and it suits this dissertation's data because small object size and class rarity co-occur here: several of the rarest sewer codes also occupy a small fraction of the frame. A secondary benefit is reduced reliance on large batch sizes, since each composite carries information from four training samples [9].

The behaviour of Mosaic on deliberately unbalanced detection datasets, including a modified variant of the standard tiling scheme, was examined by the author in a separate comparative study of YOLOv7 against the YOLOv8–YOLOv11 generations [174]. The augmentation ablation in Chapter 8. extends that work to the twelve-class dataset used here.

Mosaic's failure mode is contextual, not geometric: a pipeline component tiled beside three unrelated inspection scenes can end up next to a background or lighting condition it would never actually appear beside, and aggressive per-tile scaling can shrink a defect below the resolution at which it was annotated as visible. Both are reasons Mosaic is applied here with bounded scaling parameters, not the full range some implementations default to. Figure 5.3d illustrates the technique.

5.2.5. Copy-Paste Augmentation

Copy-Paste [103] operates at object level, extracting annotated instances from one image and inserting them into another, which gives direct control over how often each class is represented.

This makes Copy-Paste the most direct answer to class imbalance of the three synthesis techniques. Minority-class objects can be selectively duplicated and inserted into multiple training images, raising their contribution to gradient updates without altering the distribution of majority classes [103]. Where Mosaic increases object frequency only as a side effect of tiling, Copy-Paste controls which classes are boosted and by how much. It also preserves object realism, since the pasted instances are real annotated defects from actual inspections, not synthesised content.

Its failure mode is the mirror image of that strength. Because the technique duplicates specific instances, relying on a small pool of source crops for the rarest codes (BBH, BBE; Section 5.1.1.) risks the network memorizing those few appearances instead of a general representation of the class, however many background frames each is pasted into. Figure 5.3e illustrates the technique.

5.2.6. Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE [101] increases minority-class representation by generating synthetic instances instead of duplicating existing ones. It identifies the nearest neighbours of a selected minority sample and creates new samples along the line segments connecting them, expanding the minority distribution while preserving the structure of the feature space.

Its advantage over conventional oversampling is that duplication presents identical samples repeatedly, which encourages memorization, whereas synthetic samples carry slight variations and encourage more robust decision boundaries [101]. Variants adapted to image-based learning have since been proposed and applied to imbalanced classification, anomaly detection, and object recognition where minority examples are scarce [100], which is the situation of rare structural defects in inspection data.

Applying SMOTE directly to raw pixels does not transfer cleanly from its tabular-data origins, though: interpolating two images pixel-by-pixel does not produce a third, semantically valid defect image the way interpolating two feature vectors produces a third valid

feature vector, since image content lacks the tabular data's per-column independence. It also cannot generate the bounding-box geometry that an object detector requires, because there is no meaningful interpolation between two boxes drawn on different pipe walls.

These are theoretical objections, and they were not treated as sufficient grounds for excluding the method. A SMOTE implementation adapted to object detection was therefore built and run on the sewer dataset before any decision was taken.

Generation and Expert Assessment

SMOTE was therefore applied at the level of the defect instance, where the algorithm's assumptions can be satisfied. Each annotated instance of a target class was cropped and resized to a common resolution to give a fixed-length vector; nearest neighbours were computed in that space; and the standard update $x_{new} = x_i + \lambda(x_j - x_i)$, $\lambda \sim U(0, 1)$, was applied between a seed and one of its k neighbours, so that the two parents are visually alike and the interpolant stays nearer the class manifold. Each synthetic patch was composited into a real defect-free frame with a feathered elliptical mask and per-channel illumination matching, and a YOLO bounding box written.

Water ingress (BBF) was chosen as the target class, being rare enough to motivate synthesis (73 annotated instances, Table 3.4) while retaining enough parents for meaningful nearest-neighbour selection. Representative outputs are shown in Figure 5.4.

The generated samples were reviewed by an inspection professional experienced in EN 13508-2 defect coding, and were judged unsuitable for use as training data. The assessment did not turn on image quality in the photographic sense: the compositing shows no visible seams and the illumination is consistent with the destination frame. It turned on hydraulic and geometric plausibility, and three specific objections were raised.

The first, and decisive, objection is that water ingress requires a physical entry path. Infiltration occurs through a crack, an open joint, a defective connection, or a hole in the pipe wall; it cannot originate from intact material. The synthetic samples place a wet, encrusted texture onto an unbroken wall surface with no such path present, which is a configuration that does not occur in a real pipe. An inspector coding these frames would not record BBF, because the visual evidence that defines the code is absent even though its surface appearance is present.

The second objection concerns perspective. Sewer inspection footage is captured along

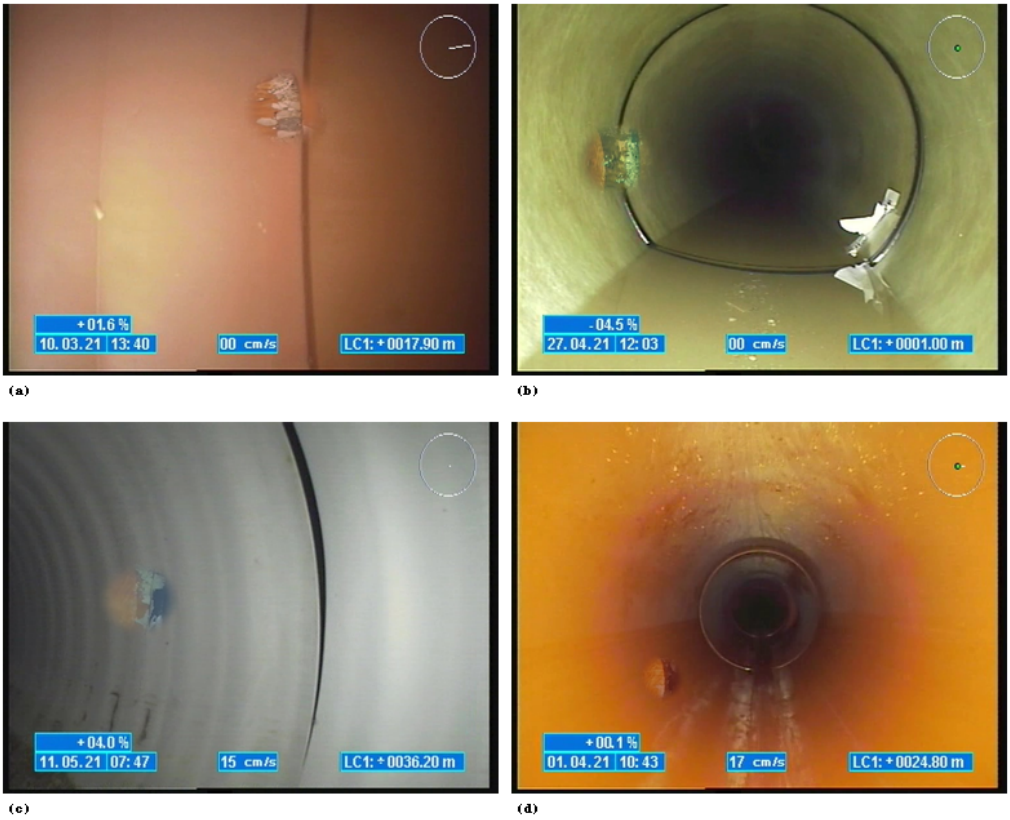


Figure 5.4: Synthetic BBF (water ingress) instances generated by instance-level SMOTE and composited into real inspection frames. These samples were assessed by an experienced inspection professional and judged unsuitable for training; they were not used in any experiment reported in this dissertation.

the pipe axis, so apparent object size falls sharply with distance down the bore. The compositing procedure places patches at a scale sampled independently of depth, producing instances that are too large for their position in several frames, and in one case an instance positioned within the bore instead of on the wall.

The third objection concerns the relationship between a defect and its surroundings. Real water ingress is accompanied by correlated evidence: staining below the entry point, deposit accumulation along the flow path, and often discolouration of the surrounding wall. A pasted patch carries the appearance of the defect but none of this context, and the inspector observed that the absence of a wet trail beneath an apparent ingress point is itself a strong signal that the observation is not genuine.

The judgement was therefore that these samples would teach a detector to associate the BBF code with a surface texture instead of with the structural condition the code denotes. Training on them risks a model that fires on wet-looking wall patches and misses genuine infiltration at joints, which is the opposite of what the class is intended to capture, and it would do so while appearing to improve the aggregate metrics that the additional samples inflate.

SMOTE is therefore not applied anywhere in this dissertation. This is a negative result, not an omission: the theoretical objection that pixel-space interpolation does not preserve semantic validity was borne out in practice, and by a domain criterion, not a statistical one. It also marks where the successful techniques differ. Copy-Paste relocates a *real* defect instance with its immediate context, so the sample remains a genuine observation in an unusual place; SMOTE constructs an instance never observed at all, and for a class defined by a physical mechanism that distinction proves decisive.

A feature- or embedding-space variant remains a plausible route worth testing, since interpolating detector features instead of pixels restores the validity of the interpolation argument and would not produce images subject to the objections above. That variant is identified as future work in Chapter 9.. The generation code used here is retained so that the experiment can be repeated on other classes or with a different generator.

5.3. Dataset Balancing Strategies

While data augmentation increases the diversity and quantity of available training samples, it represents only one component of a broader class imbalance mitigation framework. In many practical applications, augmentation alone is insufficient to fully compensate for severe disparities in class representation. Consequently, additional balancing strategies are often required to ensure that deep learning models receive adequate exposure to minority classes during training [100, 219].

Dataset balancing strategies aim to reduce disparities in class representation and improve the learning of underrepresented categories. In object detection tasks, these approaches seek to mitigate the tendency of optimization procedures to favor majority classes, thereby improving the detection performance of rare objects. Effective balancing is particularly important in pipeline inspection applications, where the natural occurrence of defects frequently results in highly skewed class distributions. Without appropriate balancing procedures, object detection models may become biased toward frequently occurring defect categories, resulting in poor detection performance for rare but operationally significant defects [4, 11].

The objective of dataset balancing is not necessarily to create perfectly uniform class distributions, but rather to ensure that minority classes contribute sufficiently to the learning process. Achieving this objective requires careful consideration of the trade-off between increasing minority-class representation and preserving the statistical characteristics of the original dataset. Excessive balancing may introduce unrealistic distributions or increase the risk of overfitting, whereas insufficient balancing may fail to adequately address class bias.

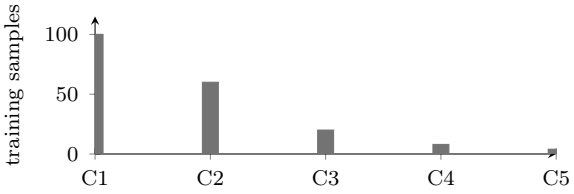
Several balancing strategies have been proposed within the machine learning literature, ranging from simple resampling methods to sophisticated adaptive sampling techniques [101, 219]. These approaches can generally be categorized into four groups:

- **Oversampling**, which increases the representation of minority classes by duplicating or generating additional samples;
- **Undersampling**, which reduces the number of majority-class samples to achieve a more balanced distribution;

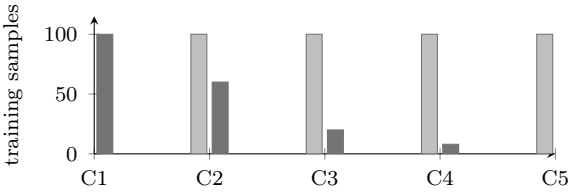
– **Hybrid balancing methods**, which combine oversampling and undersampling strategies to balance classes while minimizing their individual limitations;

– **Class-aware sampling**, which dynamically adjusts sampling probabilities during training to ensure improved exposure to minority classes.

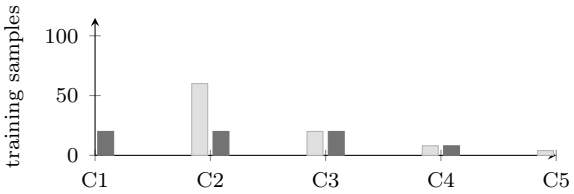
Figure 5.5 contrasts the four strategies schematically against a common long-tailed starting distribution, and the paragraphs that follow examine each in turn.



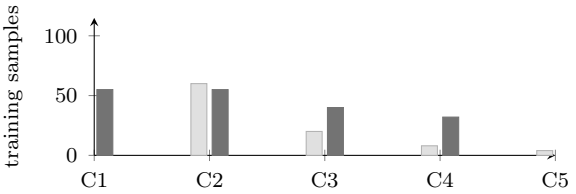
(a) Original long-tailed distribution



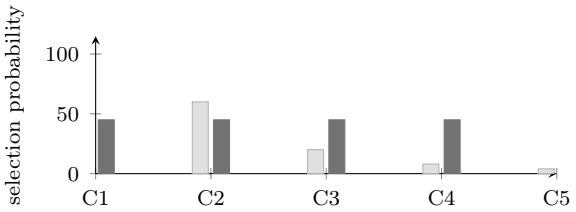
(b) Oversampling: minority classes duplicated up to the majority level



(c) Undersampling: majority classes reduced, discarding data



(d) Hybrid balancing: moderate oversampling combined with moderate undersampling



(e) Class-aware sampling: dataset unchanged, selection probability equalised

Figure 5.5: Schematic comparison of the four balancing strategies described in this section, on a five-class example ordered from most to least frequent, with panel (a) giving the common starting distribution. Light bars show the original distribution, dark bars the distribution the optimiser effectively sees. Oversampling (b) raises minority counts by duplication and enlarges the dataset; undersampling (c) equalises by discarding majority-class data; hybrid balancing (d) moves both ends part of the way, avoiding the worst of either; class-aware sampling (e) leaves the stored dataset untouched and equalises the probability with which each class is drawn into a batch, and that is why it avoids both the information loss of (c) and the duplication risk of (b). The counts shown are illustrative and chosen for legibility; they are not measured values from either inspection dataset, whose actual distributions are given in Chapter 3..

5.3.1. Oversampling Approaches

Oversampling is one of the most widely used strategies for addressing class imbalance in machine learning and deep learning applications. The primary objective of oversampling is to increase the representation of minority classes within the training dataset, thereby ensuring that these classes contribute more frequently to the optimization process [101, 219]. By increasing the number of minority-class samples, oversampling reduces the dominance of majority classes and encourages the learning of more balanced feature representations.

The simplest form of oversampling involves randomly duplicating existing minority-class samples until a more balanced class distribution is achieved. Although straightforward to implement, random oversampling presents several limitations. Since identical samples are repeatedly presented during training, the model may become overly specialized to the duplicated examples, increasing the risk of overfitting and reducing generalization performance on previously unseen data [219].

To address these limitations, modern oversampling approaches frequently employ synthetic sample generation techniques. Methods such as Synthetic Minority Over-sampling Technique (SMOTE), MixUp, Mosaic, and Copy-Paste augmentation generate additional minority-class samples by introducing controlled variations or combining information from existing examples [9, 101, 102, 103]. These techniques increase minority-class representation while simultaneously preserving sample diversity, thereby reducing the risk of overfitting associated with simple duplication.

In object detection, oversampling can act at image level, increasing how often images containing minority-class objects are drawn, or at object level, increasing the occurrence of specific defect instances within training samples [11]. It is the simplest of the four strategies compared here, requiring no removal of existing data and no change to the sampling logic, only additional minority-class examples, and it is therefore treated as the reference against which the other three are read [4, 129].

Excessive oversampling nonetheless distorts the original data distribution and raises computational cost, and where the generated samples carry insufficient diversity the detector still develops overly specialized representations. Oversampling is consequently often combined with augmentation instead of applied alone.

5.3.2. Undersampling Approaches

Undersampling is a dataset balancing strategy that seeks to reduce class imbalance by decreasing the number of samples belonging to majority classes. Instead of increasing minority-class representation, undersampling attempts to achieve a more balanced class distribution by selectively removing examples from overrepresented categories [219, 223]. As a result, majority classes exert less influence during optimization, allowing minority classes to contribute more effectively to the learning process.

The simplest form of undersampling involves randomly removing majority-class samples until a desired class distribution is achieved. Although straightforward to implement, random undersampling may discard potentially informative training examples and reduce the diversity of the retained data. To mitigate this limitation, more sophisticated approaches have been proposed, including cluster-based undersampling, prototype selection methods, and informed sample removal strategies that attempt to preserve the most representative majority-class instances while eliminating redundant samples [219].

Reducing the proportion of majority-class examples rebalances gradient updates and shortens training, but it does so by discarding data outright. Majority-class samples in inspection datasets are not interchangeable duplicates: deposits, corrosion, and sediment accumulation appear under a wide range of illumination, scale, viewpoint, and environmental conditions, and removing them removes that variation along with the redundancy the technique targeted [223]. The cost is compounded by how expensive the data was to acquire in the first place, given the specialized equipment, trained personnel, and annotation effort each inspection campaign requires [4, 129]. Undersampling is therefore applied cautiously here, generally in combination with oversampling or augmentation.

5.3.3. Hybrid Balancing Methods

Hybrid balancing methods combine oversampling and undersampling strategies to exploit the advantages of both approaches while mitigating their individual limitations [219, 223]. Instead of relying exclusively on a single balancing technique, hybrid methods seek to simultaneously increase minority-class representation while reducing the dominance of majority classes. The objective is to achieve a more balanced class distribution without introducing excessive duplication, overfitting, or substantial information loss.

A typical hybrid balancing framework involves moderate oversampling of minority classes combined with selective undersampling of highly represented categories. By carefully adjusting class frequencies, these methods can improve minority-class representation while preserving the diversity and informational content of the original dataset. Compared with pure oversampling or undersampling approaches, hybrid methods often provide a better balance between dataset size, computational efficiency, and predictive performance.

Recent studies have demonstrated the effectiveness of hybrid balancing strategies in highly imbalanced learning problems. Newaz and Haq [105] proposed a hybrid sampling framework that combines Neighborhood Cleaning Rule (NCL), random undersampling, and SMOTE-based oversampling, achieving superior classification performance compared with individual balancing methods. Similarly, Vairetti et al. [106] introduced the SMOTENN framework, which integrates synthetic minority oversampling with intelligent undersampling to simultaneously enhance minority-class representation and eliminate redundant majority-class samples. Their results demonstrated that hybrid balancing approaches can outperform conventional resampling techniques while preserving the overall structure of the dataset.

Hybrid balancing is by construction a direct answer to the trade-off described above: applying both mechanisms in moderation removes the duplication risk of pure oversampling and the information loss of pure undersampling, addressing sample scarcity and class dominance at once [4, 129]. In practice it is frequently combined with augmentation techniques such as MixUp, Mosaic, and Copy-Paste, which increase the diversity of the retained minority examples and not merely their count [101, 102, 103].

5.3.4. Class-Aware Sampling

Class-aware sampling represents a dynamic balancing strategy that modifies the probability of selecting training samples during the learning process. Unlike oversampling and undersampling approaches, which directly alter the dataset composition, class-aware sampling adjusts the frequency with which samples are presented to the model during training [91, 220]. The primary objective is to ensure that minority classes contribute more consistently to parameter updates while preserving the original dataset structure.

In a typical class-aware sampling framework, higher sampling probabilities are assigned

to underrepresented classes, whereas frequently occurring categories are sampled less often. As a result, training batches contain a more balanced distribution of classes despite the underlying dataset remaining unchanged. This strategy enables minority-class samples to influence the optimization process more effectively without requiring duplication, synthetic sample generation, or removal of existing training data.

Class-aware sampling is the only strategy in this chapter that touches neither the dataset's contents nor its size. All available samples remain accessible throughout training, so it avoids both the information loss of undersampling and the duplication risk of oversampling, at the cost of being the least straightforward of the four to implement correctly. Cui et al. [91] showed that rebalancing based on effective sample numbers substantially improves performance on underrepresented classes, and Liu et al. [220] demonstrated the value of adaptive balancing for long-tail visual recognition more generally.

The effectiveness of all four strategies is evaluated experimentally in Chapter 8..

5.4. Loss Function Adaptations for Imbalanced Detection

Although data augmentation and dataset balancing techniques can substantially improve the representation of minority classes, they do not completely eliminate the effects of class imbalance during model optimization. Even when balancing strategies are applied, object detection models may continue to exhibit a bias toward majority classes due to the disproportionate contribution of frequent classes to the overall training loss. Consequently, considerable research has focused on modifying loss functions to improve the learning process for underrepresented classes [11, 91].

Loss function adaptation is an algorithm-level approach: unlike augmentation and sampling, which modify the training data, adaptive losses leave the dataset untouched and change what the optimizer is penalized for [219]. They are therefore evaluated separately in this dissertation (Chapter 8.) and not as a substitute for balancing. The need for them is acute in object detection specifically, since detectors process a large number of candidate locations of which most correspond to background, creating an imbalance between positive

and negative samples on top of the imbalance between classes [11].

The following sections describe the three adaptations evaluated here – weighted loss, focal loss, and class-balanced loss – which respectively weight by raw class frequency, by per-example difficulty, and by effective sample number [91, 220].

5.4.1. Weighted Loss Functions

Weighted loss functions represent one of the earliest and most widely adopted approaches for addressing class imbalance in machine learning and deep learning applications [219]. The fundamental idea is to assign different importance weights to different classes based on their frequency within the training dataset. Minority classes receive larger weights, thereby increasing their contribution to the overall loss, while majority classes receive smaller weights to prevent them from dominating the optimization process.

The motivation is that standard classification losses implicitly assume balanced class distributions, so on skewed data the minority classes contribute only marginally to parameter updates [100, 219].

For a weighted classification loss, the objective can be expressed as:

$$L_{weighted} = -\sum_{i=1}^{C} w_i y_i \log(\hat{y}_i) \tag{5.3}$$

where C denotes the number of classes, w_i represents the weight assigned to class i, y_i is the ground-truth label, and $\hat{y}_i$ is the predicted probability.

The approach belongs to cost-sensitive learning, in which classes are assigned different misclassification costs [224], with weights commonly derived from inverse class frequency [91]. It is the most direct of the three adaptations compared here: each class's contribution is multiplied by a fixed, frequency-derived factor, with no notion of example difficulty or diminishing information gain – the two refinements that Focal Loss and Class-Balanced Loss add below.

Its performance depends strongly on the choice of weights. Excessive weighting destabilizes optimization and raises false-positive rates, while insufficient weighting fails to compensate for the imbalance at all, and it is this sensitivity that motivated the more advanced formulations [11, 91].

5.4.2. Focal Loss

Focal Loss was introduced by Lin et al. [11] to address the extreme imbalance between easy and difficult training examples frequently encountered in dense object detection tasks. Traditional cross-entropy loss treats all training samples equally, causing optimization to become dominated by large numbers of easily classified background examples. As a result, the contribution of difficult samples and minority classes may become insufficient for effective learning. Focal Loss addresses this limitation by dynamically reducing the influence of well-classified examples and concentrating the optimization process on hard or misclassified samples.

The Focal Loss formulation is defined as:

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{5.4}$$

where p_t denotes the predicted probability of the correct class, α controls class weighting, and γ is the focusing parameter that determines the degree to which easy examples are down-weighted.

The modulating factor $(1 - p_t)^{\gamma}$ is the key innovation. As the predicted probability of a correctly classified sample approaches one, that sample's contribution to the loss falls sharply, while difficult, low-confidence samples retain their influence [11]. Focal Loss therefore accounts for sample difficulty as well as class frequency, and that is what separates it from the weighted formulation above and makes it effective where large numbers of easy majority-class examples dominate training [11, 91].

It is the loss adaptation with the strongest evidence behind it in this dissertation: on the twelve-class Sewer Inspection Dataset it raised average recall on the two rarest classes, BBH and BBE, from 0.596 to 0.863, the largest measured rare-class improvement of any single technique reported here (Chapter 8.).

5.4.3. Class-Balanced Loss

While weighted loss functions rely directly on class frequencies, Class-Balanced Loss incorporates the concept of the effective number of samples, which was introduced by Cui et al. [91]. The underlying motivation is that additional samples from highly represented

classes contribute progressively less new information than samples belonging to minority classes – the tenth image of a common deposit adds far less than the tenth image of a rare structural crack – so weighting purely by raw class frequency overstates how much a majority class actually needs and understates how much a minority class does.

The effective number of samples is defined as:

$$E_n = \frac{1 - \beta^n}{1 - \beta} \tag{5.5}$$

where n denotes the number of samples belonging to a particular class and β is a hyperparameter close to one.

Class weights are derived from the inverse of the effective sample count. Where Weighted Loss scales with raw class counts, this scaling avoids assigning disproportionately – and for the rarest classes, unstably – large weights to categories with only a handful of training examples, while still compensating for the imbalance [91]. Cui et al. further showed that effective-sample weighting composes with both cross-entropy and Focal Loss, and later long-tail work confirmed that reweighting class contributions during optimization can matter as much as rebalancing the dataset itself [220].

5.4.4. Comparative Analysis

The three adaptations act through different mechanisms and trade off differently. Weighted Loss is simple to implement but sensitive to the choice of weighting factors; Focal Loss targets difficult examples and performs strongly on highly imbalanced detection tasks; Class-Balanced Loss is the more theoretically grounded of the two frequency-aware formulations. Table 5.1 summarises their operating principles, and Chapter 8. evaluates their effect on overall and rare-class detection performance.

5.5. Chapter Summary

This chapter examined class imbalance as a property of pipeline inspection data: the distribution is long-tailed in both datasets, and the categories carrying the greatest operational significance are consistently among the least represented, which makes overall accuracy a misleading measure of usefulness.

Table 5.1: Comparison of loss function adaptations for imbalanced object detection.

Loss Function	**Primary Mechanism**	**Complexity**	**Main Advantage**	**Limitation**
Weighted Loss	Class weighting	Low	Simple implementation and direct compensation for class imbalance	Sensitive to weight selection and may produce unstable training
Focal Loss	Hard example mining	Medium	Focuses learning on difficult and minority-class samples	Requires tuning of α and γ parameters
Class-Balanced Loss	Effective sample weighting	Medium	Accounts for diminishing information gain from majority classes	Additional hyperparameter and increased implementation complexity

Three families of mitigation were reviewed, each addressing the same deficit of minority-class gradient signal at a different stage of the training pipeline: augmentation in image space, sampling at the batch level, and loss adaptation during optimisation. Synthetic minority oversampling was also implemented, but its output failed expert review and was excluded (Section 5.2.6.), which locates the boundary of the first family – techniques that relocate or transform observed defects proved usable, while one synthesising unobserved defects did not. All are evaluated in Chapter 8..

6. Chapter

CASCADE YOLO FRAMEWORK FOR EFFICIENT PIPELINE INSPECTION

6.1. Introduction

Real-world inspection datasets differ from conventional object detection benchmarks in one respect that matters for deployment: in sewer and underwater inspection video the large majority of acquired frames contain no visible defect, and only a small subset requires detailed analysis [163]. Applying a computationally intensive multi-class detector uniformly to every frame therefore spends most of the inference budget on frames with nothing to detect, which scales badly with data volume and can preclude deployment on embedded platforms [159, 225].

This chapter investigates a two-stage cascade framework that addresses this inefficiency directly. A lightweight, high-recall first stage filters out defect-free frames. That stage is a supervised binary defect-presence classifier trained on labelled defective and defect-free frames, used as an anomaly gate; it is referred to as the anomaly gate throughout, and the term should not be read as classical anomaly detection, which trains predominantly on normal samples only. Only the frames it flags as potentially defective are passed to a full multi-class YOLO-based detector for precise localization and classification. The expensive component of the pipeline is thereby moved off the majority of frames, reducing computational redundancy while preserving detection coverage.

The remainder of this chapter reviews prior cascade-detection work (Section 6.2.),

presents the proposed two-stage architecture (Section 6.3.), describes the datasets and training procedure used for each stage, formalises the cascaded inference workflow and its evaluation protocol, and analyses the computational complexity of the resulting system. Experimental results on real-world sewer inspection data are reported in Chapter 8..

6.2. Motivation for Cascade Detection

Cascade-style filtering has proven effective wherever a rare, informative event must be found within a large volume of uninformative data, including face detection, anomaly detection, surveillance, and industrial quality control [226, 227]. A related body of work motivates using an anomaly detector for the first stage specifically: patch-based feature modelling [228], memory-bank nearest-neighbour methods [229], synthetic self-supervision [230], and normalising-flow and student–teacher approaches [231, 232] all learn primarily from normal data, which suits settings where defects are rare and visually diverse [165, 166]. The framework proposed here adopts the operational role of these methods – a fast gate ahead of a heavier analysis stage – while implementing the gate as a supervised binary classifier, since annotated defective frames are available from the same campaigns used to train the detector.

The primary objectives of the proposed cascade framework are:

- Reduce computational redundancy caused by defect-free frames.
- Increase processing speed for large-scale inspection datasets.
- Maintain high defect detection recall.
- Reduce the computational burden of multi-class detection.
- Enable deployment on resource-constrained inspection platforms.
- Improve the scalability of automated inspection systems.

6.3. Architecture of the Proposed Cascade Framework

The architecture consists of two stages. The first performs binary defect-presence classification and acts as an anomaly-detection gate. The second performs localization and

classification of individual defect categories using a multi-class object detector. Figure 6.1 gives a schematic overview.

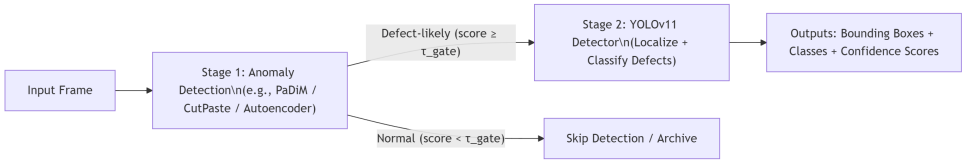


Figure 6.1: Proposed cascade detection framework.

Each acquired frame is first analysed by the lightweight anomaly detection module. Frames classified as defect-free are dropped; the rest are forwarded to the second stage, which predicts defect category, confidence score, and spatial location. Separating anomaly detection from localization concentrates computation on informative data.

6.3.1. Datasets Used for the Two Stages

Both stages were developed on the video sequence dataset introduced in Chapter 3. (Section 3.6.), assembled from 56 CCTV inspection videos recorded by a remotely operated vehicle in real sewer pipelines under varying lighting and flow conditions. Frames were sampled at 2 frames per second and annotated according to the EN 13508-2:2003 coding system [39] using the YOLO-Label annotation tool [173], with each annotation stored as a class identifier and normalised bounding-box coordinates (x, y, w, h).

The two stages solve different tasks and therefore require differently constructed training sets, summarised in Table 6.1. The Stage-2 detector is trained on the annotated frames themselves, across the ten EN 13508-2 defect categories. The Stage-1 classifier is trained on a binary relabelling of the same footage: every frame carrying at least one annotated bounding box is assigned the *defective* label, giving 11,967 positives, and the 10,460 frames annotated as containing no defect form the *non-defective* class. Deriving both stages from a single body of footage ensures the gate learns the same visual domain as the detector it protects.

The Stage-1 binary classification dataset was constructed by retaining all 10,460 available defect-free frames and randomly undersampling the defective class to the same number of frames. The resulting balanced dataset therefore contained 20,920 frames in total, comprising 10,460 defective and 10,460 defect-free examples.

Table 6.1: Dataset composition for the two stages of the cascade framework [233].

Stage	**Task**	**Composition**
Stage 1	Binary defect-presence classification	20,920 frames 10,460 defective + 10,460 background (positives undersampled from 11,967) Train / val / test: ≈80 / 10 / 10, by video
Stage 2	Multi-class defect detection	22,427 frames 10 EN 13508-2 defect categories Train / val / test: ≈80 / 10 / 10, by video

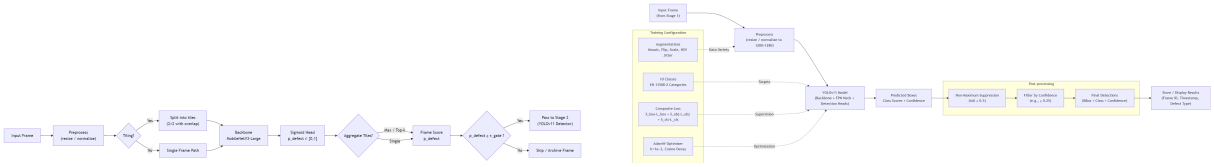
Partitioning was performed at the video level for both stages: all frames extracted from a given inspection video were assigned entirely to the training, validation, or test subset. Because consecutive frames sampled at 2 frames per second overlap heavily in content, a frame-level random split would place near-duplicate images on both sides of the partition and inflate measured performance. Video-level partitioning removes this source of temporal leakage. Undersampling of the Stage-1 defective class was performed only after the video-level train, validation, and test partition had been defined, ensuring that frames originating from the same inspection video did not appear in more than one subset.

6.3.2. Stage 1: Binary Defect Detection

The first stage is a lightweight filtering mechanism that distinguishes defective from non-defective inspection frames. Because the task is binary, substantially simpler models suffice than those required for multi-class object detection. The objective of this stage is reliable detection of potentially anomalous frames, not precise defect identification: the module operates as a high-recall gate whose purpose is to preserve as many defective frames as possible for subsequent analysis. Figure 6.2 sets the two stages side by side; panel 6.2a shows this stage's internal structure and decision flow.

Backbone architecture

Each candidate backbone terminates in global average pooling followed by a single sigmoid-activated output neuron, producing a frame-level defect probability $p_{\text{defect}} \in [0, 1]$. Three alternative architectures were trained and evaluated for this role alongside the con-



(a) Stage 1: anomaly gate **(b)** Stage 2: multi-class detection

Figure 6.2: Internal structure of the two cascade stages [233]: (a) the Stage-1 anomaly detection module and its decision flow; (b) the Stage-2 pipeline, from preprocessing through YOLOv11 inference and non-maximum suppression to structured defect output.

figuration ultimately adopted: EfficientNet-B0 at 256–384 px input [234], ConvNeXt-Tiny at 256–384 px [235], and DeiT-Tiny/TinyViT at 224–320 px [236], against MobileNetV3-Large at 224–320 px [237] trained with Class-Balanced Focal Loss [11, 91].

MobileNetV3-Large was adopted for three reasons specific to this stage's role. Its depthwise-separable convolutions and squeeze-and-excitation blocks give it a smaller parameter count and FLOP budget than the ConvNeXt and EfficientNet alternatives, which matters because Stage 1 runs on every acquired frame, whereas Stage 2 runs only on the fraction that passes the gate. Any inefficiency in Stage 1 is therefore paid in full on the whole data stream. MobileNetV3 was also designed for mobile and embedded deployment, matching the eventual target hardware for this framework. Finally, pairing it with Class-Balanced Focal Loss addresses frame-level imbalance in the loss function itself instead of relying on the backbone to compensate for a training signal dominated by defect-free examples. Comparative results across all backbones actually trained are reported in Table 8.15 (Chapter 8.).

To improve sensitivity to small defects, the module optionally applies centre-crop tiling, typically 2×2 with 25–50% overlap, and aggregates tile scores by maximum or top-k pooling. Under tiled training each tile inherits the frame label if any bounding box intersects it, following the multiple-instance formulation in which a frame is positive when any of its tiles is positive [238].

Training objective

Let $y \in \{0, 1\}$ denote the frame label, with $y = 1$ for defective, and let $\hat{p} \in [0, 1]$ be the predicted probability that the frame contains a defect. Given the severe imbalance between defective and defect-free frames, the classifier is trained with the Class-Balanced

Focal Loss [91], which combines focal re-weighting with effective-number scaling. The class weight is

$$w_y = \frac{1-\beta}{1-\beta^{n_y}}, \qquad \beta \in [0.9,\, 0.999), \tag{6.1}$$

where n_y is the number of training samples in class y, and the loss is

$$\begin{aligned} \mathcal{L}_{\text{CB-Focal}} = &- w_y\, y\, (1-\hat{p})^{\gamma}\, \log(\hat{p}) \\ &- w_{1-y}\, (1-y)\, \hat{p}^{\,\gamma}\, \log(1-\hat{p}), \end{aligned} \qquad \gamma \in [1, 2], \tag{6.2}$$

with w_{1-y} defined analogously. In Equation 6.2 the weighting term w_y of Equation 6.1 increases the contribution of minority defective samples, while the focusing term $(1-\hat{p})^{\gamma}$ emphasises hard examples. Values of $\beta = 0.999$ and $\gamma = 2.0$ gave stable convergence.

Asymmetric Loss (ASL) [239] was evaluated as an alternative objective. ASL down-weights easy negatives more aggressively through separate focusing parameters for the positive and negative terms together with a probability margin:

$$\begin{aligned} \mathcal{L}_{\text{ASL}} = &- y\, (1-\hat{p})^{\gamma_+}\, \log(\hat{p}) \\ &- (1-y)\, \hat{p}_m^{\,\gamma_-}\, \log(1-\hat{p}_m), \end{aligned} \tag{6.3}$$

$$\hat{p}_m = \max(\hat{p} - m,\, 0), \qquad \gamma_+ \in [0, 1],\ \gamma_- \in [2, 5],\ m \in [0, 0.1], \tag{6.4}$$

with typical settings $\gamma_+ = 0.0$, $\gamma_- = 4.0$, and $m = 0.05$ in Equations 6.3 and 6.4. Both objectives reduce the dominance of non-defective frames and allow the gate to reach high recall under heavy imbalance; their measured effect on gate quality and end-to-end throughput is compared in Table 8.15 (Chapter 8.).

Sampling and augmentation

Positive frames are oversampled by a factor of four to eight, and hard-negative mining replays frames that produce a high p_{defect} despite containing no labelled boxes. Augmentation comprises random rotation of $\pm 10°$, horizontal flipping, brightness and contrast

jitter, and mild Gaussian noise. Colour shifts are bounded so that the characteristic pipe-wall texture is preserved, since that texture is the main cue separating a sound wall from a degraded one.

Threshold selection

Minimising false negatives is the critical requirement of this stage. A defective frame incorrectly classified as defect-free is discarded before reaching the second stage, producing an unrecoverable detection error, since Stage 2 never revisits a frame that Stage 1 has dropped. The classifier is therefore optimised with emphasis on recall over precision.

Instead of the default 0.5 decision threshold, the operating threshold τ_{gate} is selected on a held-out validation split by taking the point on the precision–recall curve that meets a minimum target recall, reported in Table 8.15 as the gate recall, and reading off the corresponding false-positive and pass-through rates afterwards. Fixing precision or false-positive rate first would risk silently discarding rare defects, whereas fixing recall first guarantees a known bound on missed detections at the cost of a measured amount of additional Stage 2 computation. A threshold of $\tau_{\text{gate}} = 0.35$ was adopted; Section 6.7. discusses the sensitivity of the complete system to this choice. The additional false-positive frames forwarded under this setting are filtered by the object detector, which contributes a correct true negative when it finds nothing in a genuinely defect-free frame.

Temporal smoothing

Applying a per-frame threshold to a continuous video stream produces flicker: frames near the decision boundary alternate between passing and being dropped even when the underlying scene changes little. In the inspection footage used here, defects typically persist for three to twelve consecutive frames at the 2 frames per second sampling rate, so isolated single-frame decisions are usually unreliable. Two mechanisms exploit this persistence. A sliding window of W frames smooths p_{defect} by maximum or exponential moving average, and hysteresis replaces the single threshold with a pair $\tau_{\text{on}} > \tau_{\text{off}}$: once the gate opens it remains open until the smoothed probability falls below τ_{off}. Both suppress isolated false negatives without lowering the nominal threshold across the whole stream.

Both mechanisms are part of the framework as specified, not of the configuration that produced the measured results. The values of W, τ_{on}, and τ_{off} were not recorded in the source experiment, and none of the cascade results reported in Chapter 8. is obtained with smoothing enabled: every figure quoted there comes from the per-frame threshold τ_{gate} applied without hysteresis. The two mechanisms are therefore described here qualitatively, and quantifying their effect is left as outstanding work. The same applies to the optional thresholds τ_{high} and τ_{tile} of Algorithm 1 below: the refinements they gate are evaluated in Chapter 8. as on-or-off configurations, and the threshold values at which they were triggered are not recoverable from the reported results.

Training configuration

The Stage-1 classifier was trained with the Adam optimizer at a learning rate of 1×10^{-4} and weight decay of 1×10^{-5}, a batch size of 64, for 50 epochs, with early stopping monitored on validation recall and F1-score instead of on training loss alone, consistent with recall being the metric this stage is optimised for [233]. Mixed-precision arithmetic was used throughout, and input resolution was matched to the backbone: 224–320 px for MobileNetV3-Large and DeiT-Tiny, 256–384 px for EfficientNet and ConvNeXt.

6.3.3. Stage 2: Multi-Class Defect Detection

Where the first stage is constrained by computational efficiency, the second prioritises accuracy and localization precision. Panel 6.2b of Figure 6.2 shows its sequence of preprocessing, detection, post-processing, and structured output generation.

Because only a subset of frames reaches this stage, a larger and more demanding architecture can be used without a proportional increase in overall processing time.

In the implementation evaluated here the detector is YOLOv11 [182, 240], configured for the ten EN 13508-2 defect categories of the Stage-2 dataset. Input frames are resized to 1280×1280 pixels. YOLOv11 is anchor-free, so no anchor priors are fitted; the resolution choice is the adaptation to the scale distribution of pipeline defects, which is skewed towards small objects relative to general-purpose detection benchmarks, and it is complemented at inference by the optional tiling described in Section 6.4..

Training minimises a composite objective with three terms: box regression, distribu-

tion focal loss, and classification. YOLOv11 is anchor-free and NMS-based and so has no separate objectness branch; the second term is therefore the distribution focal loss, and not the objectness loss carried by anchor-based YOLO generations:

$$\mathcal{L}_{\text{YOLO}} = \lambda_{\text{box}}\,\mathcal{L}_{\text{box}} + \lambda_{\text{dfl}}\,\mathcal{L}_{\text{dfl}} + \lambda_{\text{cls}}\,\mathcal{L}_{\text{cls}}, \tag{6.5}$$

with components

$$\begin{aligned} \mathcal{L}_{\text{box}} &= 1 - \text{IoU}(b_i, \hat{b}_i), \\ \mathcal{L}_{\text{dfl}} &= \text{DFL}(\mathbf{d}_i, \hat{\mathbf{d}}_i), \\ \mathcal{L}_{\text{cls}} &= \text{BCE}(c_i, \hat{c}_i), \end{aligned} \tag{6.6}$$

where IoU denotes intersection over union between predicted and ground-truth boxes, DFL is the distribution focal loss over the discretised box-offset distribution $\mathbf{d}_i$, and c_i the defect class label. The reference configuration sets λ_{box}=7.5, λ_{dfl}=1.5, and λ_{cls}=0.5. The localization term of Equation 6.6 is the general IoU-based form discussed in Section 4.5. of Chapter 4., from which the CIoU and SIoU variants reviewed there are derived.

The detector was trained with the AdamW optimizer at an initial learning rate of 1×10^{-3} under a cosine annealing schedule, a batch size of 16, for 100 epochs, with early stopping on validation mAP [233]. This schedule is longer and more conservative than Stage 1's, reflecting the second stage's role as the accuracy-critical component of the cascade. Augmentation during training comprised mosaic blending, random scaling, horizontal flipping, and HSV jittering.

At inference each forwarded frame is normalised and resized before detection. Non-maximum suppression with an IoU threshold of 0.5 removes redundant predictions, and the surviving detections, comprising location, size, class, and confidence, are stored together with the frame timestamp for inspection reporting.

6.4. Cascaded Inference Workflow

At deployment, frames are processed sequentially through the two stages. The Stage-1 classifier produces an image-level probability p_{defect}; frames scoring at or above τ_{gate} are forwarded to Stage 2, where the detector performs localization and multi-class labelling; and post-processing yields the final detection set. Algorithm 1 states the procedure, including three optional refinements described below.

Algorithm 1 Cascaded inference at deployment

Require: Video stream $\{I_t\}$; gate threshold τ_{gate}; base confidence τ_{conf}; NMS threshold τ_{IoU}; optional high-prior threshold τ_{high} and tiling threshold τ_{tile}
Ensure: Detections $\mathcal{D}_t = \{(b_i, c_i, s_i)\}$ for each frame t

function ProcessFrame(I_t)
  $p \leftarrow \text{Stage1Classifier}(I_t)$
  if $p < \tau_{\text{gate}}$ **then**
    return $\varnothing$ ▷ skip frame
  if $p \geq \tau_{\text{high}}$ **then** ▷ dynamic confidence
    $\tau \leftarrow \max(0.10,\ 0.5\,\tau_{\text{conf}})$
  else
    $\tau \leftarrow \tau_{\text{conf}}$
  if $p \geq \tau_{\text{tile}}$ **then** ▷ tiling on demand
    $\mathcal{T} \leftarrow \text{Tile}(I_t)$
    $\mathcal{D} \leftarrow \text{DeTileMerge}\big(\bigcup_{T \in \mathcal{T}} \text{Detector}(T, \tau)\big)$
  else
    $\mathcal{D} \leftarrow \text{Detector}(I_t, \tau)$
  for all $d = (b, c, s) \in \mathcal{D}$ **do** ▷ score fusion with gate prior
    $s' \leftarrow s \cdot p$
  $\mathcal{D} \leftarrow \text{NMS}(\mathcal{D},\ \tau_{\text{IoU}})$
  $\mathcal{D} \leftarrow \{d \in \mathcal{D} \mid s' \geq \tau\}$
  $\mathcal{D} \leftarrow \text{TemporalUpdate}(\mathcal{D}, t)$ ▷ smoothing and hysteresis
  return $\mathcal{D}$
for all frames I_t in the stream **do**
  output ProcessFrame(I_t)

The three optional refinements exploit information the gate has already computed. Dynamic confidence lowers the detector's confidence threshold on frames the gate scored highly, on the reasoning that a strong defect prior justifies accepting weaker detection evidence. Tiling on demand subdivides only high-scoring frames, restricting the cost of tiled inference to frames likely to contain the small defects that tiling helps recover. Score fusion multiplies each detection confidence by the gate probability, so that detections on

marginal frames are ranked below equally confident detections on frames the gate scored strongly. Their measured contributions are reported in Section 6.7. and Chapter 8..

6.5. Evaluation Protocol

Because the two stages solve different tasks, evaluation is performed separately for each and then jointly for the complete pipeline.

Stage 1 is assessed as a binary classifier using precision, recall, F1-score, and the area under the receiver operating characteristic curve. Precision and recall follow the standard definitions in Equations 6.7 and 6.8, with F1 their harmonic mean:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{6.7}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{6.8}$$

$$\text{F1} = 2\,\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6.9}$$

The F1-score of Equation 6.9 is the summary statistic used to compare gate configurations in Chapter 8., since it penalises a configuration that reaches high recall only by forwarding most of the stream. AUROC is obtained by sweeping τ_{gate} across $[0, 1]$ and integrating the resulting curve, which characterises the gate independently of any particular operating point [241, 242]. Recall at a fixed false-positive rate is reported alongside it, since that is the quantity the deployed threshold is actually chosen to satisfy.

Stage 2 is evaluated using mean Average Precision at an IoU threshold of 0.5 and averaged over the range 0.5:0.95, following the COCO convention, together with per-class precision–recall curves. These metrics are shared with the detection experiments elsewhere in this dissertation, which allows the cascade to be compared against the single-stage baselines of Chapter 8..

For the complete pipeline, end-to-end recall is the fraction of all true defects recovered by the system as a whole, and the end-to-end negative-frame false-alarm rate is measured over all non-defective frames. It is a frame-level quantity: the denominator is the set of defect-free frames, a frame counts once regardless of how many spurious boxes it contains, and frames discarded by the gate are counted as negatives. It is therefore not comparable

with a detection-level false-positive rate:

$$\text{Recall}_{\text{cascade}} = \frac{\text{detected defects}}{\text{all true defects}}, \qquad \text{FPR}_{\text{cascade}} = \frac{\text{false alarms}}{\text{all non-defective frames}}. \tag{6.10}$$

Both quantities of Equation 6.10 are computed over the entire input stream, so that defects discarded by the gate are counted as misses. Throughput is reported as end-to-end frames per second measured over all input frames, defective and non-defective alike, and latency is averaged over 1,000 frames on the same GPU used for inference. Each reported result is the mean of three independent runs.

6.6. Computational Complexity Analysis

This section formalises the computational saving motivated above. Let T_1 denote the average inference time of the first-stage anomaly detector and T_2 that of the second-stage object detector. If a fraction p of frames is forwarded to the second stage, the average processing time per frame is

$$T_{cascade} = T_1 + pT_2, \tag{6.11}$$

where $0 \leq p \leq 1$. The pass-through rate is

$$p = \frac{N_{forwarded}}{N_{total}}, \tag{6.12}$$

with $N_{forwarded}$ the number of frames forwarded and N_{total} the number processed. The pass-through rate is the single most important characteristic of the framework, since it determines how much data requires expensive multi-class detection. Minimising it alone is insufficient, because defective frames rejected by the first stage never reach the detector; the gate is therefore held at a high-recall operating point even though this forwards additional non-defective frames.

For a conventional single-stage detector every frame must be processed by the detection network, giving $T_{single} = T_2$, and the relative saving is

$$S = 1 - \frac{T_{cascade}}{T_{single}}. \tag{6.13}$$

As the pass-through rate falls, the benefit grows, which makes the architecture attractive for real-time inspection, long-duration video analysis, and deployment on embedded platforms.

6.6.1. Worked Example Using the Measured Operating Point

The framework was benchmarked under two protocols, which produce different absolute rates from the same architecture. They differ in one respect, the Stage-2 input resolution. The *comparison protocol* (Table 8.17) measures the cascade against a single-stage YOLOv11 baseline at the 1280×1280 resolution specified in Section 6.3.3., giving 30.2 FPS single-stage and 96.0 FPS cascaded. The *FP32 ablation protocol* (Tables 8.15 and 8.18) measures end-to-end throughput over the full input stream at 640×640, giving 91 FPS single-stage and 308 FPS cascaded. The ratio between the two Stage-2 latencies is 3.01× (33.11 against 11.00 ms) for a fourfold difference in pixel count, which is the sub-linear scaling expected once fixed per-frame overheads and the better device utilisation of the larger input are accounted for. The speedup factor is consequently close under both protocols (3.18× and 3.39×); only the absolute rates differ, and figures from the two are never compared against each other.

Substituting the comparison protocol into the expression above: a pass-through rate of $p = 0.15$, $T_{single} = T_2 \approx 33.11$ ms/frame, and $T_{cascade} \approx 10.42$ ms/frame. Solving Equation 6.11 for the Stage-1 contribution gives

$$T_1 = T_{cascade} - pT_2 \approx 10.42 - (0.15)(33.11) \approx 5.45 \text{ ms}. \tag{6.14}$$

Substituting the same values into Equation 6.13 gives $S \approx 0.685$, a 68.5% reduction in average per-frame processing time. The FP32 ablation protocol gives a closely comparable saving from its own pair of rates, $1 - 91/308 \approx 0.705$, which is the same result expressed through the second protocol and is not an independent one.

6.6.2. The Pass-Through Rate Is a Property of the Footage

The saving computed above depends on the pass-through rate p, and p is not a property of the cascade. It is set jointly by the gate's operating point, which is fixed, and by the proportion of the incoming stream that is genuinely defect-free, which varies from one body of footage to the next. This subsection makes that dependence explicit, because it determines how far the measured speedup transfers to a deployment and because the annotated dataset of Section 6.3.1. is an unusually poor guide to it.

For a gate with defect-frame retention r and false-positive rate f, applied to a stream of which a fraction e carries no defect,

$$p \;=\; (1-e)\,r \;+\; e\,f. \tag{6.15}$$

The adopted gate measures $r = 0.983$ and $f = 0.08$ at $\tau_{\text{gate}} = 0.35$ (Table 8.18). Holding those fixed and varying only e gives Table 6.2.

Table 6.2: Pass-through rate and end-to-end throughput as a function of how empty the input stream is, with the gate held at its measured operating point ($r = 0.983$, $f = 0.08$; Table 8.18) and the FP32 stage latencies of Section 6.6.1. ($T_1 = 1.60$ ms, $T_2 = 11.00$ ms, 640×640). Only the footage changes across the rows. The first and fifth rows are measured directly and are the two streams of Table 6.3; the remaining rows are computed from Equation 6.15.

Defect-free fraction e	**Pass-through** p	**FPS**	**Speedup**
0.466 (annotated extraction, measured)	0.562	129	1.41×
0.70	0.351	183	2.01×
0.80	0.261	224	2.46×
0.85	0.215	252	2.77×
0.920 (survey footage, measured)	0.150	307	3.38×
0.95	0.125	336	3.69×

Two things follow. The first is that the cascade becomes more useful the emptier the footage, which is the expected behaviour and the reason the design was adopted: the saving comes from frames that never reach the detector, so a stream with more of them yields more of it.

The second concerns this dissertation's own numbers, and both ends of the range were measured directly rather than inferred. Two benchmark runs were logged under the FP32 ablation protocol, identical in gate configuration, stage latencies and threshold,

and differing only in the footage they were run over. The first covers 20,000 frames of continuous survey footage of the kind the framework is intended for; the second covers the complete 22,427-frame annotated extraction of Section 6.3.1., which was assembled for training and is defect-dense by construction. Table 6.3 gives the gate's frame-level confusion matrix for each.

Table 6.3: Gate behaviour measured directly on two streams under the FP32 ablation protocol (640×640 Stage-2 input, $T_1 = 1.60$ ms, $T_2 = 11.00$ ms, $\tau_{\text{gate}} = 0.35$). Counts are frame-level: a frame is positive when it carries at least one annotated bounding box. Both streams are labelled under the same EN 13508-2 annotation effort and differ in composition, not in gate configuration.

	Survey footage	**Annotated extraction**
Total frames N	20,000	22,427
defect-bearing N_+	1,600	11,967
defect-free N_-	18,400	10,460
Forwarded, defect-bearing (TP)	1,573	11,764
Dropped, defect-bearing (FN)	27	203
Forwarded, defect-free (FP)	1,435	837
Dropped, defect-free (TN)	16,965	9,623
Gate retention r	0.983	0.983
Gate false-positive rate f	0.078	0.080
Defect-free fraction e	0.920	0.466
Pass-through p	0.150	0.562
Cascade throughput	307 FPS	129 FPS
Single-stage throughput	91 FPS	91 FPS
Speedup	3.38×	1.41×

The two runs bracket the deployment range. On survey footage 18,400 of 20,000 frames carry no defect, the gate forwards 3,008, and the cascade reaches 307 FPS against 91 FPS single-stage. On the annotated extraction 10,460 of 22,427 frames are defect-free, the gate forwards 12,601, and the same gate and detector reach 129 FPS. The 307 FPS of the first run reproduces the 308 FPS reported for this configuration in Tables 8.15 and 8.18 to within a third of a percent. The accuracy figures reported in Chapter 8. were computed on the second stream, the throughput figures on the first, and the two are therefore not measurements of one body of footage.

Equation 6.15 predicts both outcomes. Substituting each run's measured e, r and f returns $p = 0.150$ and $p = 0.562$ against measured pass-through rates of 0.150 and 0.562. The gate also characterises almost identically on the two streams despite compositions

differing by a factor of two, at $r = 0.983$, $f = 0.078$ on survey footage against $r = 0.983$, $f = 0.080$ on the annotated extraction. The relationship in Equation 6.15 is therefore not a reconstruction of the reported rates but a model validated at both ends of the interval the intermediate rows of Table 6.2 interpolate across.

The practical consequence is the reason throughput is reported across a range instead of as a single figure: which rate a reader should expect is determined by their own footage. The 1.41× of a defect-dense benchmark and the 3.38× of continuous survey footage are both correct for their own input, and neither is a property of the architecture alone. The ordering between configurations, on which every ablation in Chapter 8. rests, is unaffected in either case, since all of them were measured under one protocol against one another.

Under the FP32 ablation protocol the Stage-1 classifier was timed directly, at approximately 625 FPS ($T_1 = 1.60$ ms) against a Stage-2 latency of 11.00 ms, which reproduces the measured cascade rate on survey footage: $1.60 + 0.150 \times 11.00 \approx 3.25$ ms, i.e. 307 FPS. The corresponding Stage-1 figure under the comparison protocol, $T_1 \approx 5.45$ ms, was not timed but solved for from Equation 6.11, and is the one quantity in this chapter obtained that way. Under either protocol the gate is roughly six to seven times faster than the detector it filters for ($33.11/5.45 \approx 6.1$ and $11.00/1.60 \approx 6.9$), consistent with the parameter-count and FLOP rationale for selecting MobileNetV3-Large in Section 6.3.2..

6.7. Selection of Cascade Design Parameters

Three design parameters govern the accuracy–throughput balance of the complete system, and each was selected empirically.

The gate threshold τ_{gate} controls the trade-off directly. Lowering it raises Stage-1 recall and the pass-through rate together, so more defects survive the gate but more computation is spent downstream; raising it recovers throughput at the cost of defects discarded irrecoverably. Sweeping τ_{gate} across the range 0.25 to 0.55 shows end-to-end recall falling monotonically as throughput rises, and $\tau_{\text{gate}} = 0.35$ was adopted as the point at which end-to-end recall remains at 0.97 while the pass-through rate falls to 15% on survey footage (Section 6.6.2.). Because a missed defect cannot be recovered downstream while wasted computation only costs time, the threshold was deliberately placed on the high-recall side of the point that would maximise raw throughput.

The choice of Stage-1 loss function matters for the same reason. Class-Balanced Focal Loss and Asymmetric Loss both address the dominance of defect-free frames, and the ablation in Table 8.15 shows them trading recall against pass-through rate: ASL yields a lower pass-through rate and a faster complete cascade, while CB-Focal retains higher recall at the gate. CB-Focal was adopted for the deployed configuration, consistent with the recall-first principle applied to the threshold.

The optional refinements in Algorithm 1 were evaluated on the same basis. Dynamic confidence and score fusion each improve end-to-end recall by exploiting the gate probability the system has already computed, at a modest throughput cost, while tiling on demand produces the largest recall gain and the largest slowdown, since it multiplies detector invocations on exactly the frames most likely to require them. These measurements are reported in Chapter 8..

6.8. Failure Modes and Deployment Considerations

The cascade design concentrates its unrecoverable risk in a single place: a false negative at Stage 1. Because Stage 2 never sees a frame the gate has classified as defect-free, any defect missed by the gate is missed by the system as a whole. A false negative inside a single-stage detector at least occurs within a system where every frame was examined by the full-capacity model. This asymmetry is why the operating point is chosen by fixing a recall floor first (Section 6.3.2.): a cascade tuned for balanced accuracy would trade a small and hard-to-observe increase in missed defects for a pass-through reduction that is easy to see and easy to over-value during development.

The failure cases observed in practice cluster around a small number of visual conditions. Underexposed footage, heavy occlusion, glare from the inspection lamp, and camera shake account for most Stage-1 false positives, since each produces local intensity structure resembling the texture discontinuities that signal a defect. The same conditions also produce the residual false negatives, which is the more serious direction; temporal smoothing and hysteresis (Section 6.3.2.) exist primarily to recover defects lost to transient artefacts of this kind.

A second consideration is resource footprint, not raw compute. Both models must be resident in memory, or cycled in and out, even though they never execute simultaneously

on the same frame – a real constraint on the embedded platforms this framework targets. This is a deliberate trade against a single larger multi-task model: two independently updatable models are easier to retrain when only one part of the pipeline changes, such as adding a defect class to Stage 2 without retouching the gate. Validating the throughput gains on embedded hardware instead of workstation GPUs remains future work (Chapter 9.).

6.9. Chapter Summary

This chapter presented a two-stage cascade addressing the computational asymmetry of inspection video, in which most acquired frames contain no defect: a lightweight binary classifier acts as a high-recall gate (Section 6.3.2.), and only the frames it forwards reach the multi-class detector (Section 6.3.3.), both trained on the extractions described in Section 6.3.1.. Every design decision follows from one constraint, that a defect discarded by the gate cannot be recovered downstream; Section 6.8. sets out where that constraint concentrates the residual risk. The operating threshold is therefore set by fixing a recall floor first and accepting the resulting pass-through rate, the loss counteracts the dominance of defect-free frames, and temporal smoothing with hysteresis protects against transient misclassification.

The saving this buys is quantified analytically in Section 6.6. and measured in Chapter 8.; because it scales with how much of the incoming stream is defect-free, Section 6.6.2. reports it across that range and measures both ends of the range directly. The gate-threshold sweep behind $\tau_{\text{gate}} = 0.35$ is reported in Section 8.4.4., the detector's per-class behaviour on the forwarded frames in Section 8.4.2., and the three optional inference-time refinements in Section 8.4.5.. Chapter 7. turns to the second structural limitation of frame-based inspection, that consecutive frames are treated as independent observations.

7. Chapter

DUAL-STREAM SPATIOTEMPORAL FRAMEWORK FOR PIPELINE DEFECT DETECTION

7.1. Introduction

Chapter 6. reduced the cost of inspecting video by declining to process most of it. This chapter addresses the complementary problem: what is lost by processing the frames that do reach the detector in isolation from one another. Most existing inspection frameworks, including every configuration evaluated so far in this dissertation, remain fundamentally frame-based, treating each image independently and relying exclusively on spatial information [4, 5, 129].

Such an approach is often insufficient for real-world inspection environments. Pipeline inspection videos frequently contain motion blur, illumination variations, turbidity, occlusions, sediment deposits, water reflections, and other visual disturbances that may obscure defect appearance. As a result, defects that are difficult to recognize in a single frame may become visible only when information from multiple consecutive frames is considered. Ignoring temporal context can therefore lead to unstable predictions, missed detections, and reduced robustness in challenging inspection conditions.

Video object detection addresses these limitations by exploiting spatiotemporal information contained within image sequences. Instead of analyzing each frame independently,

video-based approaches aggregate information across neighbouring frames to improve feature representation and detection consistency. Recent studies have demonstrated that temporal feature aggregation, motion-aware alignment, and transformer-based temporal reasoning can substantially improve object detection performance, particularly for small, occluded, or visually ambiguous objects.

This chapter presents a dual-stream spatiotemporal framework for automated pipeline defect detection, integrating a YOLOv7V-based temporal aggregation stream with a PTSEFormer-based transformer stream and combining their predictions through late fusion. The remainder reviews spatiotemporal detection methods, describes the two streams and the fusion strategy, and states the design limitations of the result; Chapter 8. evaluates it on real sewer inspection video.

7.2. Limitations of Frame-Based Detection

Most deep learning-based pipeline inspection systems perform object detection independently on individual frames extracted from inspection videos. Although modern object detectors such as YOLO, SSD, and Faster R-CNN have demonstrated strong performance in infrastructure inspection applications, they rely exclusively on spatial information and do not exploit the temporal relationships naturally present within video sequences [7, 28, 29]. Consequently, valuable information contained in neighbouring frames remains unused during the detection process.

This limitation becomes particularly important in real-world inspection environments, where image quality is frequently degraded by motion blur, illumination variations, reflections, sediment accumulation, occlusions, turbidity, and other environmental factors. Under such conditions, defects may be only partially visible within a single frame or may temporarily disappear from view. As a result, object detectors operating on individual images can produce missed detections or unstable predictions [12, 13].

Frame-based approaches also suffer from temporal inconsistency. Because each image is processed independently, small variations in object appearance or camera position produce fluctuations in confidence score and box location across consecutive frames, manifesting as intermittent detections and localization instability. The problem is most pronounced for the small, visually ambiguous defects that dominate this domain – cracks,

localized corrosion, joint defects, minor deformations – which occupy few pixels in any one frame but persist across many. The same continuity would allow a detector to separate genuine defects from transient artefacts such as reflections, shadows, and suspended particles, which resemble defects in a single frame but do not persist [14, 88].

7.3. Spatiotemporal Object Detection

Spatiotemporal detection exploits the temporal coherence of video: consecutive frames overlap in content, so information that is ambiguous, occluded, or degraded in one frame can be recovered from its neighbours [12, 13]. Early approaches propagated features along optical flow, as in Flow-Guided Feature Aggregation [12], which is effective but requires expensive motion estimation and struggles under complex motion.

Two families have since displaced it. Convolution-based methods aggregate features across neighbouring frames while preserving real-time throughput, YOLOV and YOLOv7V being the representative examples [88]. Transformer-based methods such as TransVOD and PTSEFormer instead use self-attention to model long-range dependencies and semantic relationships across the sequence, improving detection under occlusion, motion blur, and complex scene dynamics [13, 14]. Their strengths are complementary: convolutional aggregation is efficient but limited to a local temporal window, while transformers reason over longer ranges at substantially higher cost. That complementarity is the motivation for the hybrid framework developed in this chapter.

7.3.1. Temporal Aggregation Module

The temporal aggregation module represents the key component that differentiates YOLOv7V from conventional frame-based object detectors. Its objective is to integrate information from multiple consecutive frames and generate feature representations that simultaneously encode spatial appearance and temporal context [12, 88]. Instead of relying exclusively on the current frame, the detector processes a short temporal window of neighboring observations: features extracted from adjacent frames are aligned and aggregated with the reference frame's own features to produce an enriched spatiotemporal representation, exploiting visual information that may be degraded, partially occluded,

or entirely missing within the reference frame alone.

Figure 7.1 presents the resulting process. Spatial features are first extracted from consecutive frames using the YOLO backbone network; the temporal alignment module then establishes correspondences between neighboring observations before feature aggregation generates the unified representation used for final object detection.

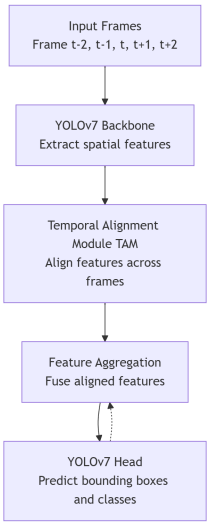


Figure 7.1: Conceptual overview of the YOLOv7V temporal aggregation process. Features extracted from neighboring frames are aligned and aggregated before object detection.

Formally, let F_t denote the feature representation extracted from the reference frame at time t, and let $F_{t-k}, \ldots, F_{t+k}$ denote feature representations extracted from the k neighboring frames on either side. Temporal aggregation is expressed as:

$$F_{agg} = A(F_{t-k}, \ldots, F_t, \ldots, F_{t+k}) \tag{7.1}$$

where $A(\cdot)$ denotes the temporal aggregation function and F_{agg} the resulting spatiotemporal feature representation. For the five-frame sequence employed in this framework ($k = 2$, Section 7.3.2.), this is:

$$F_{agg} = A(F_{t-2}, F_{t-1}, F_t, F_{t+1}, F_{t+2}) \tag{7.2}$$

Temporal aggregation is most beneficial precisely when object appearance varies across consecutive frames – defects partially obscured by reflections, motion blur, or transient environmental conditions rarely suffer the same degradation identically across an entire

sequence, so neighboring frames typically contain complementary information that compensates for what is missing in any single frame. This has two measurable consequences for pipeline inspection specifically: it reduces false-negative detections on defects that occupy only a small, easily-degraded portion of the frame, and it stabilizes confidence scores and bounding-box locations across the sequence, reducing the intermittent detections that plague frame-independent baselines (Section 8.5.).

7.3.2. Frame Sequence Construction

To enable temporal feature aggregation, inspection videos are divided into overlapping frame sequences centered around a reference frame. Instead of analyzing individual images independently, the detector processes a temporal window consisting of multiple consecutive observations. The central frame serves as the primary detection target, while neighboring frames provide contextual information used to improve feature extraction and localization accuracy.

In this framework, a sequence of five consecutive frames is employed, consisting of the reference frame together with two preceding and two subsequent observations. This configuration provides sufficient temporal context while maintaining practical computational requirements. The resulting input sequence can be expressed as:

$$\{I_{t-2}, I_{t-1}, I_t, I_{t+1}, I_{t+2}\} \tag{7.3}$$

where I_t denotes the reference frame and the remaining frames provide temporal context. Figure 7.2 shows a five-frame window extracted at this sampling rate. Consecutive frames overlap substantially in content while differing in camera distance and viewpoint, the latter visible in the on-screen distance readout. This overlap is the redundancy that the temporal aggregation module and the PTSEFormer stream (Section 7.5.) are designed to exploit.

The use of overlapping sequences ensures that each frame contributes to multiple temporal windows, improving detection stability and reducing sensitivity to short-term image degradation. This strategy enables the detector to exploit temporal continuity while preserving the dense frame coverage required for infrastructure inspection applications.

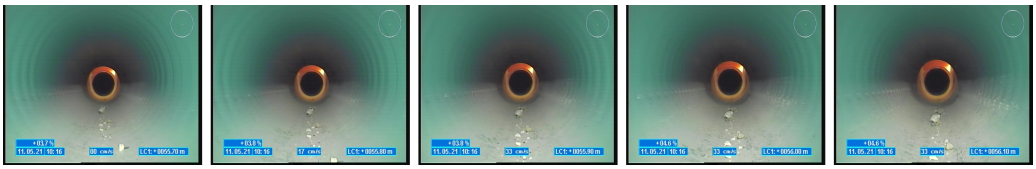


Figure 7.2: A five-frame temporal window $\{I_{t-2}, \ldots, I_{t+2}\}$ sampled at 2 frames per second from ROV inspection footage [243].

7.3.3. Spatial-Temporal Feature Fusion

The aggregated representations are then fused into a unified spatiotemporal feature map combining spatial detail on defect appearance with temporal continuity across neighbouring observations, and forwarded to the YOLO detection heads for localization and classification. The fused representation carries more context than single-frame features, and that is what makes the detector more robust to illumination change, motion blur, and partial occlusion while retaining the real-time characteristics of a one-stage architecture.

7.4. YOLOv7V: Architecture and Implementation

7.4.1. Network Configuration

YOLOv7V extends the YOLOv7 detector [179], whose backbone, neck, and head structure is summarised in Figure 7.3, with a temporal aggregation mechanism modelled on the design introduced in YOLOV [88].

The input layer is widened to accept 15 channels, formed by stacking the five consecutive RGB frames of the temporal window described in Section 7.3.2.. A temporal alignment module (TAM) estimates motion offsets between the key frame and each of its four support frames, and these offsets are used to spatially align features before aggregation. Alignment is necessary because the inspection camera advances continuously along the pipe: without it, features extracted from I_{t-2} and I_{t+2} describe different physical positions on the pipe wall, and aggregating them directly would blur the defect signature. The aggregated features are passed to a standard YOLOv7 detection head for bounding-box regression and classification. The contribution of the alignment module is isolated experimentally in the component ablation reported in Chapter 8..

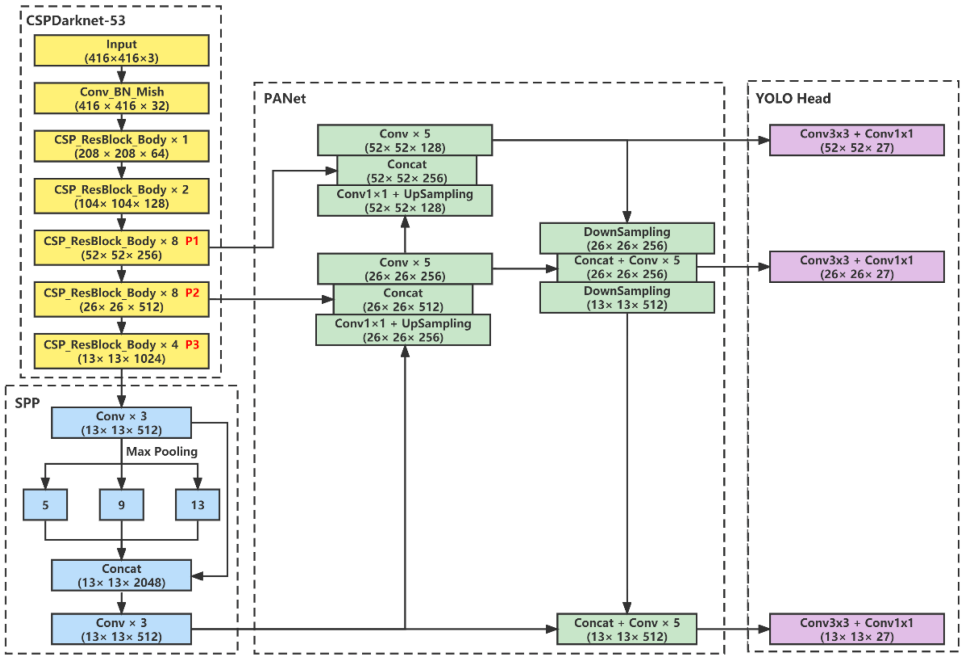


Figure 7.3: YOLOv7 architecture used as the backbone of the YOLOv7V stream.

7.4.2. Rationale for the YOLOv7 Backbone

YOLOv7 was selected as the base detector despite the availability of later releases, for reasons that the experiments in Chapter 8. subsequently tested instead of assumed. YOLOv7 was widely adopted in both research and industry at the time the spatiotemporal experiments were conducted, and its accuracy relative to later releases is not uniformly inferior: the author's own comparative evaluation found YOLOv7 outperforming YOLOv8 on pipeline inspection imagery [171], and the large-scale ODverse33 benchmark reported that newer YOLO releases do not consistently surpass YOLOv7 across diverse detection tasks [244]. Its feature aggregation structure also provides a convenient insertion point for the temporal alignment and aggregation stages described above.

Because this reasoning rests on evidence that predates the most recent YOLO generations, the assumption was tested directly: Chapter 8. evaluates YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLO26 on the same dataset under identical conditions, and compares them against both the YOLOv7V stream and the complete fusion framework.

7.4.3. Training Configuration

Focal loss [11] was applied alongside the objectness and classification terms to improve learning on underrepresented defect classes, down-weighting easy negatives so that the gradient signal is dominated by hard and infrequent examples. This is the same intervention analysed in detail in Chapter 5., applied here within a video detector.

The model was trained from scratch using stochastic gradient descent with a learning rate of 0.005, a momentum coefficient of 0.937, and a batch size of 16, for 300 epochs with learning-rate warmup followed by cosine decay [243]. The momentum value follows the default adopted across the YOLOv5 and YOLOv7 training pipelines [245]. Input frames were resized to 640×640 pixels and normalised to the $[0, 1]$ range. Training required approximately 28 hours on the hardware described in Chapter 3..

7.4.4. Limits of Convolutional Temporal Aggregation

The benefits described above are bounded by what a convolutional aggregation function $A(\cdot)$ with a fixed, local temporal window can represent. YOLOv7V combines information from the $k = 2$ neighboring frames on each side, but has no mechanism for reasoning about dependencies beyond that window, and its alignment step is a learned local correlation instead of an explicit model of long-range semantic consistency. Large inter-frame displacements and weak-contrast features are the two conditions under which this design degrades most noticeably, both of which occur regularly in sewer footage when the inspection platform accelerates or when a defect is visible only under oblique illumination.

Section 7.5. introduces PTSEFormer, a transformer-based stream designed to address this gap, and Section 7.6. describes how the two streams are combined.

7.5. PTSEFormer-Based Temporal Reasoning Stream

The second stream addresses the limitation identified above: a convolutional aggregation function with a fixed local window cannot model long-range temporal dependencies, yet defect evidence in inspection video is often distributed across observations separated by more frames than that window spans. It employs PTSEFormer (Progressive Temporal-Spatial Enhanced Transformer), which extends the DETR paradigm by integrating temporal and spatial information from neighbouring frames progressively instead of in a single step, through dedicated modules for temporal aggregation and spatial transition modelling [14].

The distinction between single-step and progressive aggregation is illustrated in Figure 7.4. Conventional video detectors correlate the target frame against its context frames in one operation, so a degraded or misaligned reference frame contributes to the fused representation with the same weight as a clean one. PTSEFormer instead builds the representation in stages, allowing temporal and spatial information to be introduced separately and re-anchored to the target frame between steps.

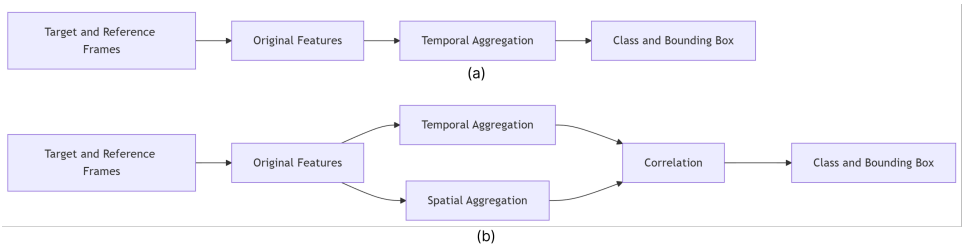


Figure 7.4: Feature aggregation strategies: (a) conventional single-step aggregation, (b) the progressive aggregation used by PTSEFormer [14].

The architecture is built on the Detection Transformer (DETR) framework [86]. Target and context frames pass through a shared encoder backbone, typically a ResNet followed by a transformer encoder, and the resulting frame-level features are refined by three principal components: the Temporal Feature Aggregation Module (TFAM), the Spatial Transition Awareness Module (STAM), and the Query Assembling Module (QAM). Together these capture temporal dependencies, model object motion and spatial transitions, and generate context-aware object queries for transformer-based detection. Figure 7.5 gives an overview of the complete pipeline.

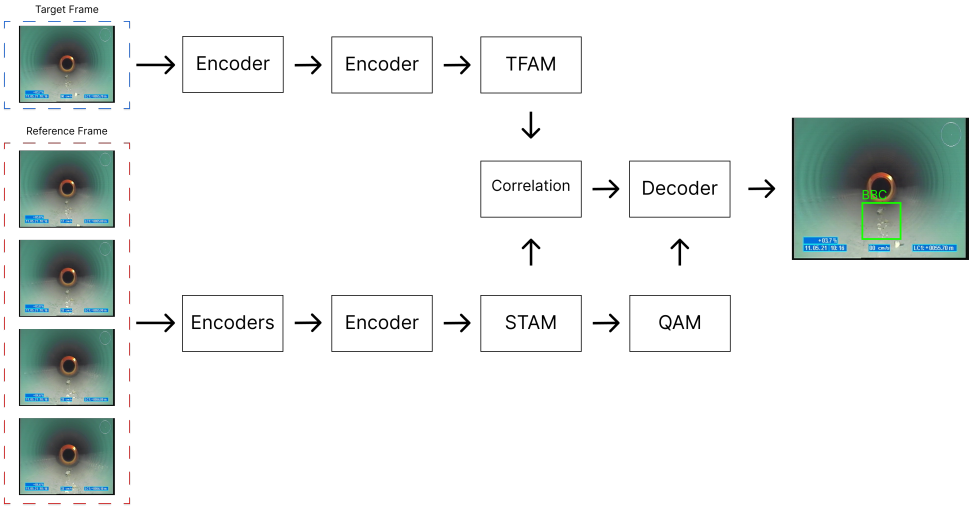


Figure 7.5: PTSEFormer pipeline.

7.5.1. Temporal Feature Aggregation Module

The Temporal Feature Aggregation Module (TFAM) is responsible for introducing temporal information from neighboring frames into the feature representation of the target frame. The underlying assumption is that objects observed in adjacent frames provide complementary visual information that can improve recognition under challenging imaging conditions such as motion blur, occlusion, or low contrast.

Given a target-frame feature representation M_t and a set of context-frame features M_t^c, TFAM generates a temporal memory representation h_t according to:

$$h_t = C(M_t, M_t^c) \tag{7.4}$$

where $C(\cdot)$ denotes a transformer-based correlation operation [14].

The correlation operation is defined as:

$$C(Q, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V + Q \tag{7.5}$$

where Q, K, and V represent query, key, and value embeddings, respectively.

PTSEFormer models temporal relationships using self-attention mechanisms. The attention operation can be expressed as:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{7.6}$$

where Q, K, and V denote query, key, and value feature embeddings, respectively, and d represents the feature dimensionality.

By attending to context-frame features, TFAM allows the detector to recover information that may be missing or degraded within the current frame. This capability is particularly valuable in sewer inspection videos, where defects may be only partially visible due to reflections, sediment accumulation, water ingress, or temporary occlusions.

7.5.2. Spatial Transition Awareness Module

While TFAM focuses on temporal consistency, PTSEFormer additionally introduces a Spatial Transition Awareness Module (STAM) designed to model spatial relationships between the target frame and neighboring observations. The motivation behind STAM is that objects and defects continuously change position as the inspection platform moves through the pipeline. Consequently, robust detection requires not only temporal consistency but also awareness of spatial transitions occurring between frames.

STAM estimates the spatial correspondence between the target-frame representation and each neighboring frame by learning transition-aware feature mappings. Unlike conventional temporal aggregation methods that focus primarily on appearance similarity, STAM explicitly models positional changes and object movement across the video sequence [14].

To improve feature fusion, PTSEFormer introduces a Gated Correlation mechanism that balances contributions from current-frame and context-frame features. The operation can be expressed as:

$$C_g = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V + M \odot Q + (1 - M) \odot V \tag{7.7}$$

where M represents a learned gating function controlling the relative influence of current and context features [14].

This mechanism improves robustness against low-quality reference frames and prevents the model from becoming overly dependent on noisy temporal observations.

7.5.3. Query Assembling Module

A distinguishing characteristic of PTSEFormer is the Query Assembling Module (QAM), which generates context-aware object queries for transformer decoding. In conventional DETR-based detectors, object queries are fixed learnable embeddings that remain unchanged during inference. While effective for static image detection, fixed queries are less suitable for video sequences where object appearance and location continuously evolve.

To address this limitation, QAM dynamically assembles object queries using information extracted from neighboring frames. Instead of relying solely on static embeddings, the module combines query information with contextual frame features through a lightweight transformer decoder. The resulting query representation incorporates temporal information describing object appearance, position, and motion patterns throughout the sequence [14].

The assembled query representation can be expressed as:

$$Q = [Q_p, \{SD(Q_p, M_{t+i})\}] \tag{7.8}$$

where Q_p denotes the original object query and $SD(\cdot)$ represents the shallow decoder used for query refinement [14].

By incorporating temporal context directly into query generation, QAM improves detection consistency and enables more reliable localization of small, occluded, and temporally ambiguous defects.

7.5.4. Advantages of Transformer-Based Temporal Reasoning

Two properties make this complementary to the YOLOv7V stream. Self-attention permits direct interaction between all frames in the sequence, so long-range dependencies can be modelled across the whole sequence, and the representation does not depend on a local receptive field, so context distributed across the whole segment remains available [14]. Where YOLOv7V provides efficient real-time aggregation, PTSEFormer contributes stronger temporal reasoning and better detection of rare, occluded, and visually ambiguous defects.

7.6. Proposed Dual-Stream Fusion Framework

Although both YOLOv7V and PTSEFormer improve upon conventional frame-based object detection, they address different aspects of the video object detection problem. YOLOv7V emphasizes computational efficiency and real-time processing through convolution-based temporal aggregation, whereas PTSEFormer focuses on robust temporal reasoning using transformer-based attention mechanisms. Experimental observations indicate that YOLOv7V performs particularly well for frequently occurring and visually distinct defect classes, while PTSEFormer exhibits superior robustness when detecting rare, partially occluded, or temporally ambiguous defects.

Motivated by these complementary characteristics, a dual-stream detection framework is proposed. YOLOv7V processes every video sequence as the primary stream; PTSEFormer is invoked selectively, only when the YOLOv7V result is unavailable, uncertain, or concerns a rare class, as formalised by the routing criterion of Equation 7.12, and its output is then combined with YOLOv7V's through decision-level fusion. Instead of merging feature representations during inference, each invoked model independently generates defect detections that are integrated only after the prediction stage. This design preserves the strengths of both architectures while maintaining implementation flexibility and modularity.

An overview of the proposed dual-stream fusion framework is shown in Figure 7.6.

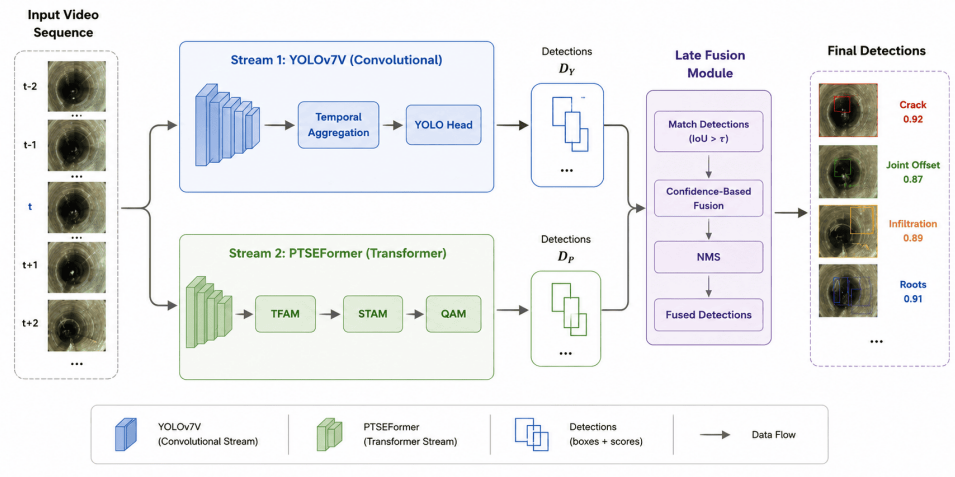


Figure 7.6: Proposed dual-stream fusion framework combining the YOLOv7V temporal aggregation stream and the PTSEFormer temporal reasoning stream through decision-level late fusion.

7.6.1. Framework Architecture

Input video is divided into fixed temporal windows containing the target frame and its neighbours, and every window is processed by YOLOv7V (Section 7.4.) for efficient short-range temporal aggregation. PTSEFormer's transformer-based reasoning over longer ranges and spatial transitions is invoked on a window only when YOLOv7V's result triggers the routing criterion of Equation 7.12; whichever streams run produce bounding-box coordinates, class labels, and confidence scores, which the fusion module then matches and consolidates into a single prediction set.

7.6.2. Late Fusion Strategy

Late fusion operates on the final predictions of each detector and not on intermediate feature representations, which lets each stream retain its specialized behaviour and keeps the two independently trainable. Let

$$D_Y = \{d_Y\} \tag{7.9}$$

denote the set of detections generated by YOLOv7V and

$$D_P = \{d_P\} \tag{7.10}$$

denote the detections generated by PTSEFormer.

Each detection is represented as:

$$d = (b, c, s) \tag{7.11}$$

where b denotes the predicted bounding box, c represents the defect class label, and s corresponds to the associated confidence score. [243]

7.6.3. Selective Invocation of the Transformer Stream

YOLOv7V is evaluated on every temporal window; PTSEFormer is not. Whether the transformer stream is invoked on a given window W must therefore be decided from the YOLOv7V output alone, since that is the only evidence available at the point the decision

is taken. Let τ_{route} denote a confidence threshold on the primary stream and C_{rare} the set of underrepresented defect classes. The routing indicator is

$$\rho(W) = \begin{cases} 1, & \text{if } D_Y = \varnothing \ \text{ or } \ \max\limits_{d \in D_Y} s_Y < \tau_{route} \ \text{ or } \ \exists\, d \in D_Y : c \in C_{rare} \\ 0, & \text{otherwise,} \end{cases} \tag{7.12}$$

where $\rho(W) = 1$ means PTSEFormer is executed on W and $\rho(W) = 0$ that it is not. The three disjuncts correspond to the three conditions under which the primary stream is judged insufficient on its own: it returned nothing, it returned nothing it was confident about, or it returned a class whose rarity is the case the transformer stream was added to cover.

Equation 7.12 governs *whether* the second stream executes; Equation 7.15 below governs *which* detection survives once it has. Keeping the two separate is a necessity: the fusion rule is defined over a matched pair (d_Y, d_P) and so presupposes that d_P has already been computed, which means it cannot also serve as the criterion deciding whether to compute it. Where $\rho(W) = 0$ no fusion takes place and the YOLOv7V detections stand as the output for that window.

The routing criterion follows the framework as implemented in [243].

7.6.4. Confidence-Based Decision Fusion

Candidate detections produced by the two streams are matched using the Intersection over Union (IoU) criterion. For a pair of detections generated by YOLOv7V and PTSEFormer, the overlap is calculated as:

$$IoU = \frac{|b_Y \cap b_P|}{|b_Y \cup b_P|} \tag{7.13}$$

where b_Y and b_P denote the bounding boxes predicted by YOLOv7V and PTSEFormer, respectively. [243]

If the overlap exceeds a predefined threshold τ,

$$IoU > \tau \tag{7.14}$$

the detections are considered to correspond to the same defect instance and are subsequently fused.

Let s_Y and s_P denote the confidence scores produced by YOLOv7V and PTSEFormer for the matched pair, and let C_{rare} denote the set of underrepresented defect classes. The final prediction is then selected, not averaged:

$$d^* = \begin{cases} d_P, & \text{if } s_P > s_Y \text{ or } c \in C_{rare} \\ d_Y, & \text{otherwise} \end{cases} \tag{7.15}$$

The surviving detection retains its own confidence score. A weighted alternative, in which the two scores are combined as $C_{fusion} = w_1 s_Y + w_2 s_P$ with $w_1 + w_2 = 1$, was considered and rejected: averaging two confidences that were produced by differently calibrated models yields a score that is not comparable with either, which would complicate the confidence thresholding applied downstream. Equation 7.15 is therefore the rule used throughout, and it is the rule whose results are reported in Chapter 8..

This strategy explicitly prioritizes PTSEFormer predictions for rare defect categories because transformer-based temporal reasoning demonstrated superior performance for minority classes and temporally ambiguous defects. Conversely, YOLOv7V predictions are retained for high-confidence detections belonging to frequently occurring classes, thereby preserving real-time detection efficiency while improving robustness for difficult defect categories.

7.6.5. Detection Workflow

The complete detection workflow of the proposed framework can be summarized as follows:

1. Acquisition of inspection video sequences.
2. Video preprocessing and temporal window generation.
3. Processing by YOLOv7V on every window; evaluation of the routing criterion (Equation 7.12) and selective processing by PTSEFormer where it triggers.
4. Generation of independent defect predictions.

5. Matching of overlapping detections.

6. Confidence-based fusion of predictions.

7. Removal of redundant detections using Non-Maximum Suppression.

8. Generation of final defect localization and classification results.

This workflow enables the framework to exploit both local temporal aggregation and global temporal reasoning while maintaining a modular architecture suitable for practical deployment in automated pipeline inspection systems.

7.6.6. Implementation and Training

Both streams were trained on the video sequence dataset described in Chapter 3. (Section 3.6.), using the same video-level partitioning and the same five-frame temporal windows, so that the two sets of detections fused at inference derive from identical inputs. Preprocessing followed the masking procedure described in Chapter 3.: fixed pixel regions carrying the timestamp, speed, distance, and orientation overlays were set to zero using the Pillow imaging library [246] before frames were resized to 640×640 and normalised.

Training the transformer stream is substantially more expensive than training the convolutional one. PTSEFormer required approximately 72 hours against YOLOv7V's 28 hours on the hardware described in Chapter 3. [243]. The same asymmetry is expected at inference: measured single-stream throughput is 12 FPS for PTSEFormer against 30 FPS for YOLOv7V, equivalent to per-frame latencies of roughly 83 ms and 33 ms respectively.

The fused configuration reaches 18 FPS [243]. This is above the 12 FPS PTSEFormer achieves standalone because PTSEFormer is not executed on every window: it is invoked selectively, per the routing criterion of Equation 7.12, so only the subset of windows that trigger it pays its cost.

The three reported rates are mutually consistent under this reading, and taken together they fix the invocation rate implicitly. Writing r for the fraction of windows on which the transformer stream runs, and executing the two streams sequentially on one device,

$$T_{fused} \;=\; T_Y + r\,T_P + \varepsilon, \tag{7.16}$$

where $T_Y = 1/30 \approx 33.3$ ms and $T_P = 1/12 \approx 83.3$ ms are the measured per-frame costs of the two streams and ε is the cost of matching and fusing the detections. The measured 18 FPS corresponds to $T_{fused} \approx 55.6$ ms, so with ε neglected

$$r \;=\; \frac{T_{fused} - T_Y}{T_P} \;\approx\; \frac{55.6 - 33.3}{83.3} \;\approx\; 0.27. \tag{7.17}$$

The two limiting cases of Equation 7.16 bound the result. At $r = 0$ the framework degenerates to YOLOv7V alone at 30 FPS. At $r = 1$, with both streams run on every window, it would fall to $1/(33.3 + 83.3) \approx 8.6$ FPS, which is below PTSEFormer's own standalone rate because the convolutional stream's cost would then be paid on top of it. A fused rate of 18 FPS is therefore unreachable if both branches always execute, and is reached at an invocation rate of roughly one window in four. Any non-zero fusion overhead ε lowers the estimate of r further, so 0.27 is an upper bound.

Equation 7.17 therefore fixes the rate at which the framework invokes its transformer stream, and establishes that the measured throughput is arithmetically consistent with selective invocation and arithmetically inconsistent with unconditional execution of both streams, which is the claim the framework's design rests on.

7.6.7. Computational Cost of the Dual-Stream Design

Running a second detector on the frames that trigger it adds cost relative to a single-stream design, and this cost is a deliberate part of the design instead of an incidental overhead. A tracking-based pipeline such as YOLO combined with DeepSORT [247] offers an instructive comparison: detection runs once per frame and a lightweight association step maintains object identities, adding little beyond the cost of the baseline detector. The framework proposed here instead runs YOLOv7V on every frame and PTSEFormer selectively, fusing their outputs where both are available.

The two designs optimise for different objectives. Tracking maintains identity for objects that have already been detected, so a defect missed by the detector in every frame remains missed. The dual-stream design targets exactly that failure: recovering rare, subtle, or occluded defects that a single detector does not find at all. In inspection settings where a missed structural defect carries a far higher cost than additional computation, that trade is worth making. The model-size consequences are reported in Chapter 8.,

where the fused framework's parameter count and FLOP budget are compared against each individual stream.

The relevant throughput target is modest: inspection platforms advance slowly, and the footage analysed here was sampled at 2 frames per second, well below the fused framework's 18 FPS, so the design carries substantial headroom over the acquisition rate it must keep up with.

7.6.8. Evaluation Protocol for the Dual-Stream Framework

The two streams and their fusion are evaluated with the precision, recall, average precision, and mean average precision definitions applied throughout this dissertation, computed identically for all configurations so that the fusion result can be attributed to the fusion itself.

Because the motivating claim is that fusion improves detection of underrepresented defect classes specifically, overall mAP alone is insufficient: a model can raise mAP while continuing to fail on rare categories that contribute little to the average. Per-class average precision is therefore reported for selected representative and challenging categories, chosen to span the range from structurally distinct, high-frequency defects to visually ambiguous ones. Categories whose test-split populations are too small to yield a stable per-class value are pooled and reported as an aggregate instead of individually. This exposes where the streams differ and makes it possible to check that any aggregate gain is distributed across the class range and not concentrated on the categories that were already well handled.

Statistical significance is assessed with a two-sided paired t-test performed on per-video mAP@0.5 values from the ten held-out inspection videos that constitute the test subset ($n = 10$, df $= 9$; Chapter 3., Section 3.6.). Because the test set comprises complete inspection videos, mAP@0.5 is computed separately for each of them, giving paired measurements for the two streams and the fused framework across the same set of videos. Pairing at the video level instead of the frame level respects the dependence between frames drawn from a single inspection, which would otherwise inflate the effective sample size. The outcome of this test is reported in Chapter 8..

7.6.9. Design Limitations

Three limitations of the proposed framework follow directly from its design and are stated here instead of being deferred to the conclusion.

The routing and fusion rules are both fixed. Selection between the two streams' detections depends on a hand-set IoU threshold τ and a predefined set of rare classes C_{rare}, and the decision to invoke the second stream at all depends on a hand-set τ_{route}; all three are chosen in advance. This keeps the rules interpretable and cheap to evaluate, at the cost of an operating point that cannot adapt to footage whose class distribution differs from the training data.

Motion compensation is implicit. The temporal alignment module estimates offsets between frames but is not specialised to the forward ego-motion characteristic of an inspection platform advancing along a pipe, where apparent motion is dominated by radial expansion from the image centre and not the translation typical of general video.

Evaluation is confined to a single acquisition setting. All experiments use footage from one inspection platform annotated under one protocol, so the framework's robustness to different camera systems, lighting rigs, and annotation conventions remains untested. Cross-dataset validation on independently collected sewer footage is identified as future work in Chapter 9..

7.7. Chapter Summary

This chapter developed a dual-stream spatiotemporal framework, starting from the observation that inspection footage is a temporally continuous signal while conventional detectors discard that continuity. The two streams are complementary: YOLOv7V aggregates features across a short aligned window at real-time throughput, suiting frequent and visually distinct classes, while PTSEFormer models longer-range dependencies through progressive temporal and spatial attention, suiting rare, occluded, and low-contrast defects a local convolutional window cannot resolve. A routing rule decides from the primary stream's output alone whether the transformer stream is invoked at all, and a late-fusion rule then combines the outputs of whichever streams ran by IoU matching, preferring the transformer stream for rare classes and low-confidence cases.

The cost of the design was stated, not minimised (Section 7.6.7.): PTSEFormer's selective invocation keeps the workload below that of running two detectors on every frame, but both the routing threshold τ_{route} of Equation 7.12 (Section 7.6.3.) and the fusion rule of Equation 7.15 are fixed. Both are acceptable in a setting where a missed structural defect is far more expensive than additional computation, and both are set out in full in Section 7.6.9. and revisited in Chapter 9.. The resulting 18 FPS throughput is reported in Chapter 8. alongside its accuracy.

8. Chapter

EXPERIMENTAL RESULTS AND DISCUSSION

8.1. Introduction

This chapter reports the experimental results obtained for every method developed in the preceding chapters and evaluates them against the research hypotheses stated in Chapter 1..

The presentation follows the order in which the methods were introduced. Section 8.2. establishes baseline detection performance by comparing YOLO generations on the underwater and sewer datasets, and examines two dataset-level factors specific to inspection imagery: contextual background content and on-screen telemetry overlays. Section 8.3. evaluates the class imbalance mitigation strategies of Chapter 5., covering augmentation techniques, dataset balancing strategies, and loss function adaptations, with particular attention to their effect on rare defect classes. Section 8.4. evaluates the cascade framework of Chapter 6. against a single-stage baseline on both accuracy and throughput. Section 8.5. evaluates the individual spatiotemporal streams, and Section 8.6. the complete dual-stream fusion framework of Chapter 7.. Section 8.8. then assesses each hypothesis against the evidence assembled, and Section 8.9. summarises the findings.

All experiments use the hardware and software configuration described in Chapter 3.. Detection results are drawn from a common metric set (precision, recall, mAP@0.5, mAP@0.5:0.95, and frames per second), but not every metric was recorded for every study:

mAP@0.5:0.95 is unavailable for the sampling and loss-function experiments, rare-class recall for the augmentation ablation, and rare-class AP for the loss-function comparison. Missing entries are shown as dashes in the tables concerned, and comparisons in the text are made only on metrics both configurations report. Two inspection datasets are used, represented through three experimental dataset configurations, and every result below states which configuration it draws on: the 3,021-image underwater pipeline dataset, the 13,000-image twelve-class Sewer Inspection Dataset, and the 22,427-frame ten-class video sequence dataset. The latter two are two extractions of a single sewer annotation effort, not two independent datasets, and absolute values are therefore not comparable between them (Chapter 3.).

8.2. Evaluation of YOLO-Based Object Detection Architectures

8.2.1. Performance Comparison of YOLO Architectures

The evaluation is organised by dataset, not by architecture, because the two datasets were not covered by the same set of models. On the underwater dataset the comparison spans the YOLOv4 family and a Faster R-CNN baseline (Table 8.1), then the YOLOv5–YOLOv8 generations (Table 8.3). On the sewer dataset it spans thirteen YOLOv7 variants (Table 8.4), the YOLOv8–YOLOv11 generations together with modified Mosaic augmentation (Section 8.2.4., Table 8.6), and YOLO26 against YOLOv11 (Table 8.7). Within each table all models were trained and evaluated under one protocol; between tables the protocols differ, and no comparison is drawn across them. Performance is assessed on precision, recall, and mAP@0.5 throughout, with F1-score reported where the source study recorded it (Table 8.7) and inference speed and training cost reported separately in Tables 8.2 and 8.5. The architectural properties of these models are summarised in Table 4.1 of Chapter 4.; the tables below report their measured detection performance on this dissertation's datasets. One baseline named in the approved research proposal, MobileNet-SSD, is not evaluated here. The single-shot detector comparison was consolidated onto the YOLO family, which shares the same one-stage design while being actively maintained across the generations this dissertation tracks, and Faster R-CNN

and EfficientDet-D0 were retained as the two-stage and alternative one-stage references respectively. The MobileNet family still enters the experimental work, as MobileNetV3-Large is the backbone adopted for the cascade gate (Section 6.3.2.); what is untested is MobileNet-SSD as a complete detection baseline.

8.2.2. Underwater Pipeline Detection Results

The underwater dataset was used to evaluate the capability of YOLO-based detectors to identify pipeline components and anomalies under challenging environmental conditions. Underwater imagery is characterized by reduced visibility, light attenuation, scattering effects, color distortion, and varying levels of turbidity, all of which increase the difficulty of automated object detection.

The dataset was extracted from more than 15,000 raw frames captured during ROV inspection videos, of which 3,021 images were annotated across five object categories: pipeline, leakage point, concrete weight, concrete mat, and pipe coupling (Chapter 3., Table 3.2) [172].

Table 8.1: Detection performance of evaluated architectures on the underwater pipeline dataset [129]. The five per-class columns report detection accuracy at an IoU threshold of 0.5, defined as the proportion of predicted boxes that are correct; the final column reports mAP@0.5 over all five classes. The @16 and @64 suffixes denote the Darknet subdivision setting used for YOLOv4-Tiny, not a model size.

Architecture	Pipeline	Leakage Point	Concrete Weight	Concrete Mat	Pipe Coupling	mAP (%)
YOLOv4	98.53	81.83	94.62	96.00	96.40	**94.21**
YOLOv4-Tiny@64	54.03	66.78	76.07	83.09	79.39	72.97
YOLOv4-Tiny@16	86.84	81.27	96.05	98.78	**97.46**	92.43
CSP-YOLOv4	91.04	**84.37**	**97.85**	**98.80**	85.03	93.97
YOLOv4@ResNet50	78.05	34.92	84.70	95.48	89.38	79.50
YOLOv4@DenseNet201	91.26	63.83	90.69	97.92	95.65	88.40
Faster R-CNN	65.48	45.99	72.37	80.45	76.10	70.49

YOLOv4 achieves the highest overall mAP among the evaluated models at 94.21%, though not the best per-class result: CSP-YOLOv4 leads on leakage point, concrete weight, and concrete mat, while YOLOv4-Tiny@16 leads on pipe coupling. No single architecture dominates every class.

One-stage detectors outperform the Faster R-CNN baseline across every class. Leakage point is the hardest category for every model, plausibly because it is small and visually indistinct against the pipe surface, whereas larger infrastructure components – pipelines, concrete mats, concrete weights – are detected consistently well across architectures.

All models in this comparison were initialised from MS COCO weights; a YOLOv4 control trained from randomly initialised weights on the same data reached 90.98% mAP against the 94.21% obtained with transfer learning, confirming that pretraining was worth its cost on this dataset. This is the opposite of what was later observed on the sewer dataset (Section 8.2.3.), where training from scratch outperformed transfer learning; the difference is discussed in Section 8.2.12..

Table 8.1 reports accuracy alone; Table 8.2 adds the other half of the trade-off, reporting inference throughput together with training cost and convergence behaviour [129].

Table 8.2: Inference throughput, training time, and convergence behaviour of the evaluated architectures on the underwater pipeline dataset [129]. Faster R-CNN's training time and convergence iteration were not recorded; the two YOLOv4-Tiny subdivision settings share one FPS figure since subdivision affects training memory, not inference cost.

Architecture	**FPS**	**Training time (h)**	**Convergence iter.**
YOLOv4	25	23.7	7,500
YOLOv4-Tiny@16	**38**	**3.2**	8,000
YOLOv4-Tiny@64	**38**	6.4	8,500
CSP-YOLOv4	15	–	–
YOLOv4@ResNet50	17	58.8	37,500
YOLOv4@DenseNet201	16	65.1	37,500
Faster R-CNN	4	–	–

Read together, Tables 8.1 and 8.2 show that YOLOv4's leading mAP is not the fastest configuration: YOLOv4-Tiny is roughly 1.5× faster (38 against 25 FPS) while giving up 21.2 points of mAP at 64 subdivisions, though the 16-subdivision variant recovers most of that gap at 92.43%. CSP-YOLOv4 and the two modified-backbone variants are both slower and less accurate than plain YOLOv4, so neither substitute backbone paid for its added depth. Faster R-CNN, at 70.49% mAP and 4 FPS against YOLOv4-Tiny's 38, is the clearest evidence that two-stage detectors are not competitive here on either axis.

The training-cost columns of Table 8.2 reinforce the conclusion the accuracy and throughput figures already established: the modified-backbone variants are not a favourable trade. YOLOv4@ResNet50 and YOLOv4@DenseNet201 both required roughly 18–20× the training time of the best-performing YOLOv4-Tiny@16 configuration and needed five times as many iterations to converge (37,500 vs. 7,500–8,500), yet neither exceeded plain YOLOv4's mAP (Table 8.1: 79.50% and 88.40% respectively, against YOLOv4's 94.21%). YOLOv4-Tiny@16 is the standout on this axis specifically: at 3.2 hours it trained roughly

7× faster than the full YOLOv4 model (23.7 hours) while reaching 92.43% mAP, within 1.8 percentage points of YOLOv4's result – the smallest training-time-to-accuracy cost of any architecture in this comparison, consistent with its already-noted status as the strongest efficiency-oriented choice on the inference side as well.

The convergence behaviour summarised numerically in Table 8.2 is shown directly in Figure 8.1, which plots the training loss curves for the YOLOv4 reference model alongside YOLOv5 and YOLOv7 [129]. All three converge without the loss oscillation or divergence that would indicate an unstable learning rate, and the plateau positions correspond to the convergence iterations reported above.

Precision–recall and F1–confidence curves for the two strongest architectures are shown in Figure 8.2. The precision–recall curves confirm that the high mAP values in Table 8.3 reflect performance sustained across the operating range instead of a single favourable threshold, and the F1–confidence curves identify the confidence settings at which each model is best balanced, which is the practical quantity for configuring a deployed detector.

Table 8.3: Performance comparison of YOLO generations on the underwater pipeline dataset [171]. Values are as reported in the source study; the YOLOv4 row is the reference model carried over from Table 8.1.

Architecture	Variant	Precision	Recall	mAP (%)
YOLOv4	(reference)	0.901	0.951	94.21
YOLOv5	YOLOv5l6	0.901	0.951	**97.10**
YOLOv6	YOLOv6l6	0.778	0.857	95.30
YOLOv7	YOLOv7-e6e	0.885	0.955	96.30
YOLOv8	YOLOv8x	0.840	**0.969**	95.10

YOLOv5 achieved the highest overall detection performance with a mAP of 97.10%, while YOLOv8 produced the highest recall value of 0.969, indicating superior capability for detecting a larger proportion of underwater objects. YOLOv7 achieved a balanced performance, obtaining a recall of 0.955 and a mAP of 96.30%. In contrast, YOLOv6 produced the lowest precision (0.778) and recall (0.857) of the five, although its mAP of 95.30% still exceeds that of the YOLOv4 reference model, which indicates that the weakness lies in the operating point at which precision and recall were read, not in the ability to localise the objects at all. Confidence calibration is one possible explanation; threshold selection, class averaging, and differences in the reporting conventions of the implementation used are others that this experiment does not separate.

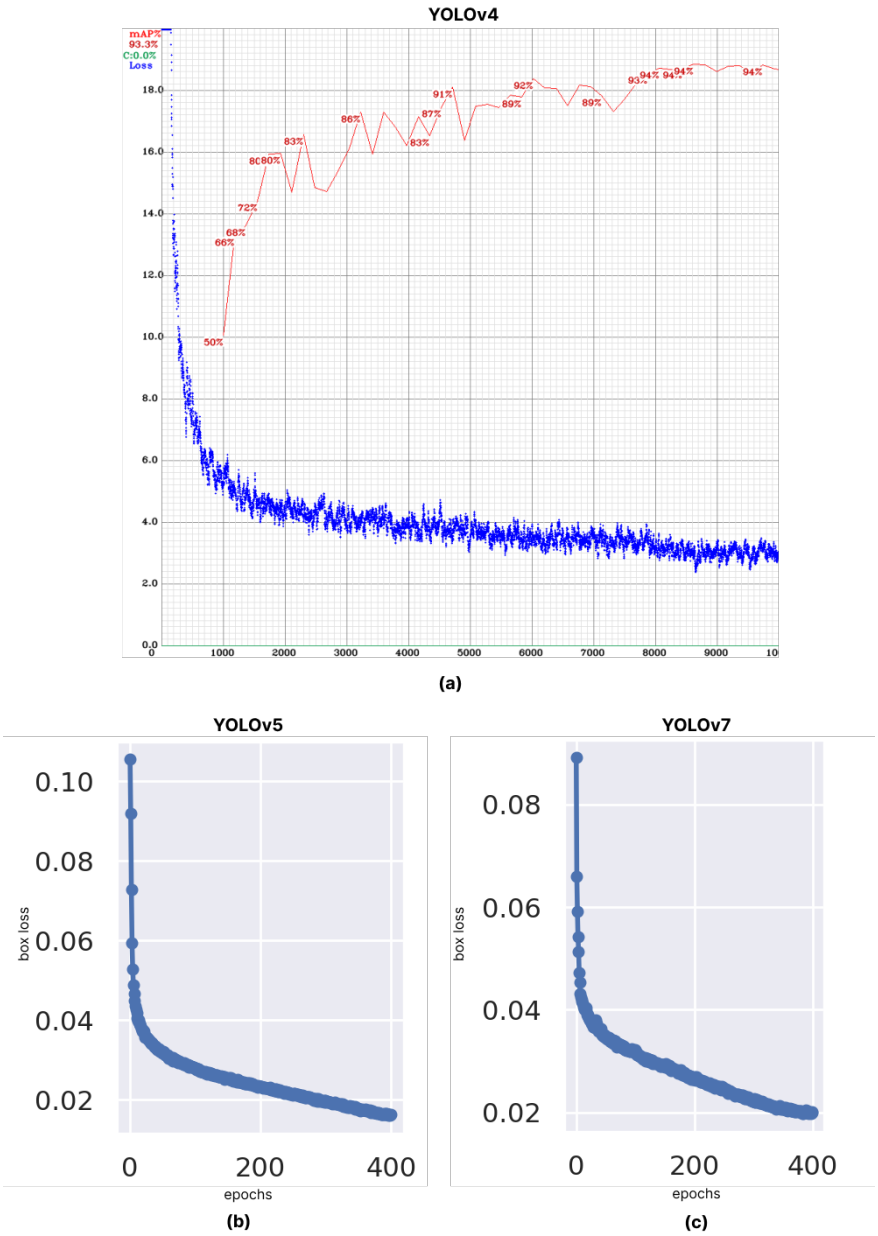
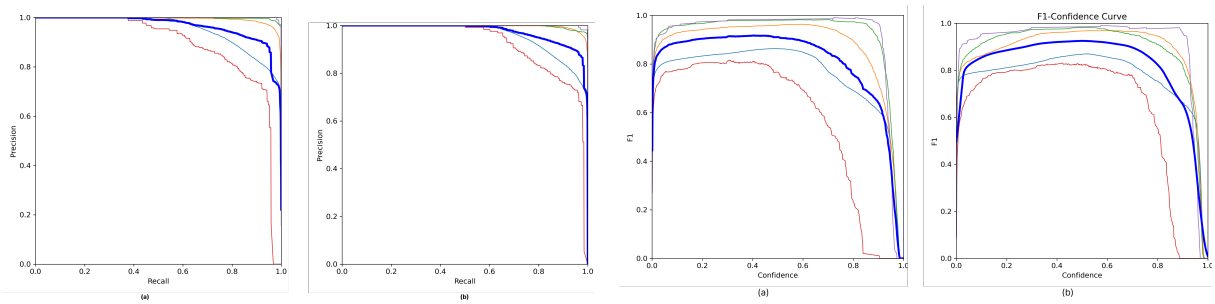


Figure 8.1: Training loss on the underwater pipeline dataset: (a) loss and mAP for the YOLOv4 reference model, (b) loss for YOLOv5, (c) loss for YOLOv7 [129].



(a) Precision–recall **(b)** F1–confidence

Figure 8.2: Precision–recall and F1–confidence curves on the underwater pipeline dataset for YOLOv7 and YOLOv5 [129].

These results indicate that improvements introduced in newer YOLO generations do not necessarily guarantee superior performance for underwater inspection tasks. Instead, the effectiveness of a particular architecture depends on the characteristics of the dataset and inspection environment. Overall, YOLOv5 and YOLOv7 demonstrated the most favorable balance between detection accuracy and robustness, establishing a strong baseline for subsequent investigations involving class imbalance mitigation, cascade detection, and spatiotemporal video analysis.

8.2.3. Sewer Defect Detection Results

The 13,000-image Sewer Inspection Dataset (Chapter 3.) was used to evaluate the performance of YOLO-based object detection architectures for identifying structural and operational defects defined according to the EN 13508-2 standard. Compared with the underwater dataset, sewer inspection imagery presents a more challenging detection scenario due to the larger number of defect categories, significant visual variability, and severe class imbalance. The dataset includes all twelve defect categories used in this dissertation, namely BAA, BAB, BAC, BAH, BAI, BAJ, BAK, BBB, BBC, BBF, BBH, and BBE; this is distinct from the separate 22,427-frame, 10-class video sequence dataset used for the cascade and spatiotemporal experiments in Chapters 6. and 7. (Section 3.6.).

Among the standard YOLOv7 variants, YOLOv7s achieved the highest precision of 0.955 and a mAP value of 0.912. Larger architectures such as YOLOv7-D6 and YOLOv7-D6s produced comparable results, achieving mAP values of 0.905 and 0.910, respectively, but these differences are small relative to their substantially greater computational requirements.

The table also isolates the effect of the training regime, since each architecture appears in both a COCO-pretrained and a from-scratch configuration. Training from scratch was better in every paired comparison but one: YOLOv7s over YOLOv7 (0.912 vs. 0.901), YOLOv7-xs over YOLOv7-x (0.832 vs. 0.826), YOLOv7-W6s over YOLOv7-W6 (0.874 vs. 0.853), YOLOv7-D6s over YOLOv7-D6 (0.910 vs. 0.905), and YOLOv7-E6Es over YOLOv7-E6E (0.904 vs. 0.875); only YOLOv7-E6s fell marginally below YOLOv7-E6 (0.890 vs. 0.894). Transfer learning did reduce training time by roughly 5%, but on this dataset it cost accuracy. This is the reverse of the underwater result reported in

Table 8.4: Performance comparison of YOLOv7 variants on the twelve-class Sewer Inspection Dataset [175]. The suffix *s* denotes training from randomly initialised weights ("from scratch"); rows without it were initialised from MS COCO weights. The suffix is therefore a training regime, not a model size.

Architecture	Precision	Recall	mAP
YOLOv7	0.939	0.880	0.901
YOLOv7s	**0.955**	0.894	0.912
YOLOv7-x	0.848	0.792	0.826
YOLOv7-xs	0.877	0.812	0.832
YOLOv7-W6	0.824	0.820	0.853
YOLOv7-W6s	0.866	0.815	0.874
YOLOv7-E6	0.879	0.845	0.894
YOLOv7-E6s	0.898	0.885	0.890
YOLOv7-D6	0.906	0.879	0.905
YOLOv7-D6s	0.915	0.889	0.910
YOLOv7-E6E	0.927	0.888	0.875
YOLOv7-E6Es	0.916	0.901	0.904
YOLOv7-ModifiedLoss	0.903	**0.931**	**0.936**

Section 8.2., where COCO pretraining was worth 3.2 percentage points of mAP, and the contrast is taken up in Section 8.2.12..

The modified YOLOv7 model incorporating focal loss achieved the highest recall of 0.931 and the highest mAP of 0.936 in the table, exceeding the best standard variant (YOLOv7s, 0.912) by 2.4 percentage points. This is the single largest gain produced by any intervention on this dataset, and it is produced entirely by a change to the training objective: the architecture, the data, and the splits are identical to the YOLOv7s row above it.

The aggregate figure in fact understates the effect, for a reason visible only in the per-class breakdown of Table 8.14. Focal loss redistributes capacity from classes the baseline already handled well to those it did not. Averaged over the twelve classes with equal weight the baseline scores 0.852 mAP against the focal model's 0.936; measured at dataset level, where frequent classes dominate, the baseline scores 0.912. The two conventions disagree by six points on the baseline precisely because it is weak on rare classes, and agree at 0.936 for the focal model because its performance is no longer strongly frequency-dependent. That convergence is itself evidence the imbalance was mitigated, not relabelled.

Figure 8.3 summarises the mAP of all thirteen variants graphically.

The accuracy figures above should be read alongside what each variant cost to train.

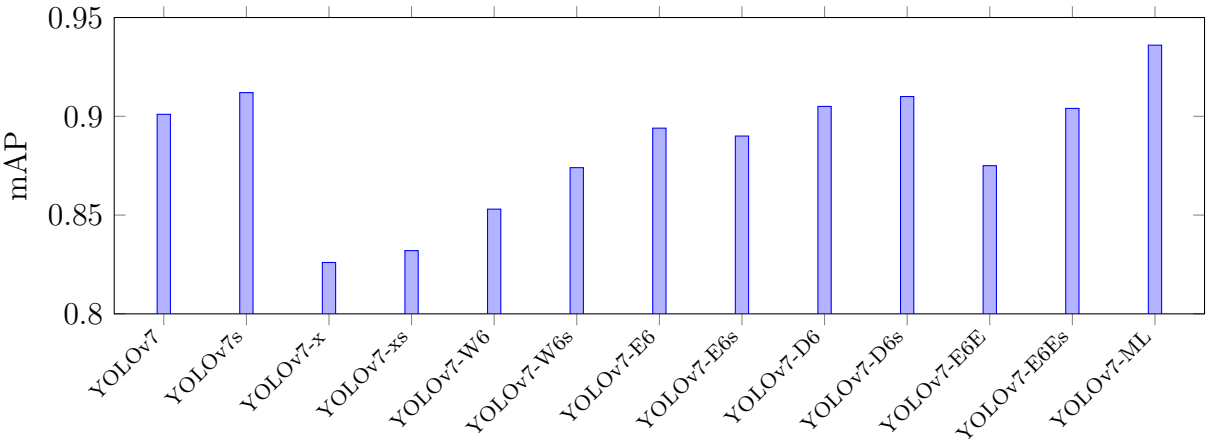


Figure 8.3: mAP comparison across YOLOv7 variants on the sewer inspection dataset (Table 8.4). YOLOv7-ML denotes the focal-loss-adapted YOLOv7-ModifiedLoss configuration.

Table 8.5 reports wall-clock training time for all thirteen configurations [175]. The ordering is informative: the best-performing standard variant, YOLOv7s, trained in 22.4 hours, whereas the heaviest configurations (YOLOv7-E6E, YOLOv7-E6Es) took 26.1 and 26.8 hours for lower mAP. The focal-loss variant is the outlier, at 34.9 hours, which is 56% more than the YOLOv7s baseline it improves on. Focal loss is therefore not a free intervention: its rare-class gain, quantified in Section 8.3., is paid for in training time even though inference cost is unchanged.

Table 8.5: Wall-clock training time for the YOLOv7 variants on the twelve-class Sewer Inspection Dataset [175].

Architecture	**Training time (h)**
YOLOv7	20.4
YOLOv7s	22.4
YOLOv7-x	**15.7**
YOLOv7-xs	19.1
YOLOv7-W6	19.9
YOLOv7-W6s	20.4
YOLOv7-E6	19.6
YOLOv7-E6s	22.7
YOLOv7-D6	24.8
YOLOv7-D6s	25.4
YOLOv7-E6E	26.1
YOLOv7-E6Es	26.8
YOLOv7-ModifiedLoss	34.9

Figure 8.4 shows the training loss curves for the best-performing standard variant

(YOLOv7s) and one of the larger configurations (YOLOv7-D6s) [175]. Both converge smoothly, and the near-identical final loss of the two models is consistent with the small mAP difference between them in Table 8.4: the additional capacity of the D6s configuration did not translate into a better fit on this dataset.

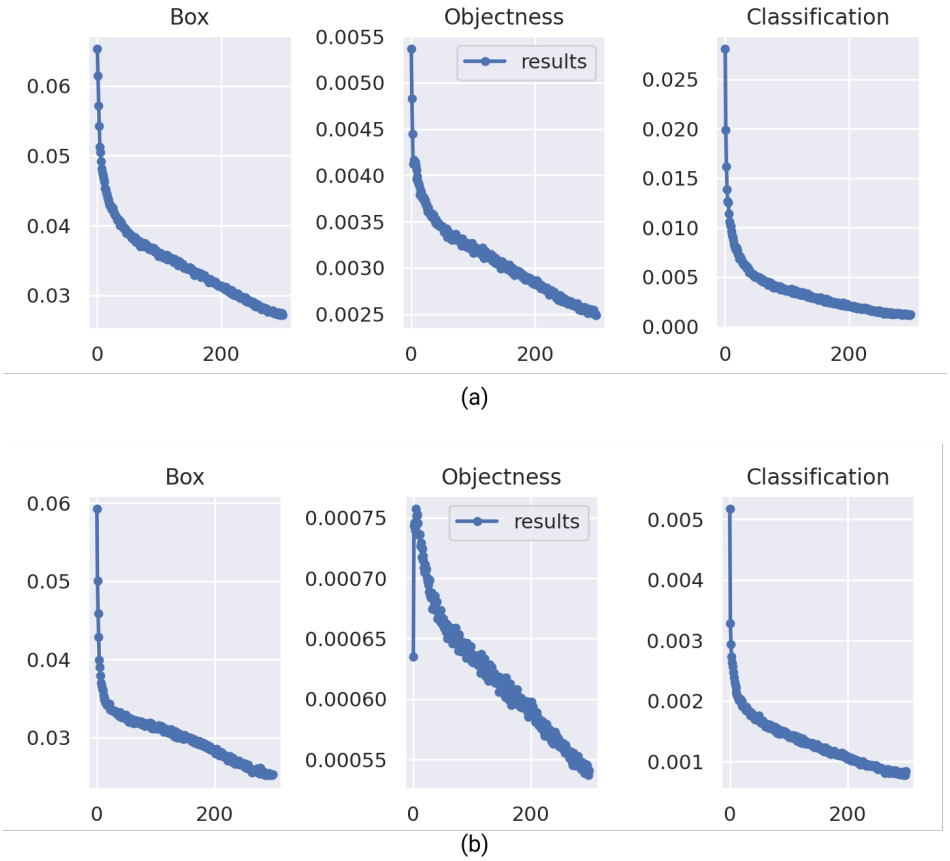
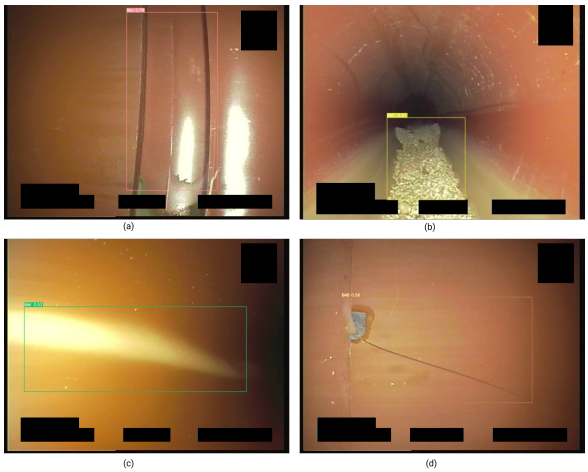


Figure 8.4: Training loss on the sewer inspection dataset: (a) YOLOv7s, (b) YOLOv7-D6s [175].

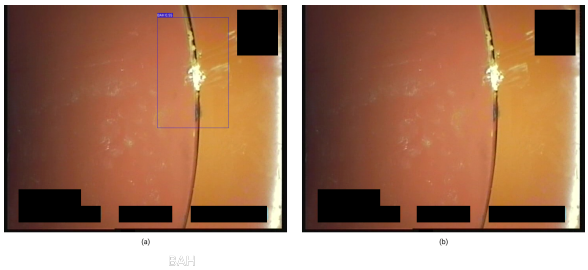
Representative detections produced by the trained detector are shown in Figure 8.5 8.5a, covering four defect categories that differ substantially in appearance: pipe misalignment (BAJ), deposits (BBC), defective lining (BAK), and cracks (BAB).

8.2.4. Modified Mosaic Augmentation and the YOLOv8–YOLOv11 Generations

The comparison above is confined to the YOLOv7 family. A separate study on the same twelve-class dataset extended it in two directions at once: forward through the YOLOv8–YOLOv11 generations, and sideways to a modified form of Mosaic augmentation



(a) Baseline detections



(b) Focal loss vs. baseline

Figure 8.5: Qualitative detections on the sewer inspection dataset [175]: (a) representative detections across four defect categories that differ substantially in appearance – pipe misalignment (BAJ), deposits (BBC), defective lining (BAK) and cracks (BAB); (b) the focal-loss-adapted model against the unmodified baseline on the same frames, where the adapted model recovers instances the baseline misses or scores lower.

designed for this data specifically [174]. Standard Mosaic tiles four images into one by resizing *and cropping* them. The modified variant resizes without cropping, preserving each source image whole and padding with black pixels to maintain the aspect ratio. The motivation is directly tied to the class distribution of Table 3.4: cropping is most likely to destroy exactly the small, rare defect instances that the augmentation is meant to expose the detector to more often. Results are given in Table 8.6.

Table 8.6: Modified Mosaic augmentation and the YOLOv8–YOLOv11 generations on the twelve-class Sewer Inspection Dataset [174].

Architecture	Precision	Recall	mAP@0.5
YOLOv7 (standard Mosaic)	0.913	0.857	0.898
YOLOv7@Modified Mosaic	0.929	0.889	0.912
YOLOv8	0.905	0.871	0.902
YOLOv9	0.938	0.910	0.921
YOLOv10	0.949	0.922	0.929
YOLOv11	**0.952**	**0.932**	**0.937**

Two findings follow. First, the augmentation change alone is worth 1.4 percentage points of mAP@0.5 (0.898 to 0.912) and 3.2 points of recall (0.857 to 0.889), at no inference cost whatever, since Mosaic is applied only during training. That places a purely data-level modification of YOLOv7 above the unmodified YOLOv8 (0.902), which is the clearest

single demonstration in this dissertation that how a detector is trained on inspection data can matter more than which generation it belongs to.

Second, the newer generations do continue to improve on this dataset, monotonically from YOLOv8 (0.902) through YOLOv11 (0.937), which is a different picture from the underwater result in Table 8.3 where no such ordering held. The sewer dataset has twelve classes against the underwater dataset's five and a far more severe class imbalance, so it rewards the improvements in feature aggregation and label assignment that distinguish the later generations, whereas the underwater task is close to saturated for all of them. Architecture ranking is therefore dataset-specific in a way that has a concrete explanation.

8.2.5. YOLO26 Compared with YOLOv11

YOLO26 was released as this research was being completed, and its arrival raised an obvious question for a dissertation that had already committed to earlier generations. The release is built around end-to-end, NMS-free inference and a reduced deployment footprint, both of which are attractive for inspection hardware. Whether those changes also improve defect detection on sewer imagery is a separate matter, and one that can only be settled by measurement. YOLO26 was therefore trained and evaluated on the twelve-class Sewer Inspection Dataset under the same protocol as the YOLOv11 baseline.

Table 8.7 reports the outcome alongside the YOLOv7 baseline used throughout this section. The three rows come from two separate training campaigns run at different times, so the YOLOv11 figures here differ slightly from those in Table 8.6, which came from the augmentation study; the comparison that carries weight is YOLO26 against the YOLOv11 run it was benchmarked alongside, under one protocol.

Table 8.7: YOLOv7, YOLOv11, and YOLO26 on the twelve-class Sewer Inspection Dataset. Values are averaged over repeated runs. Entries marked † were not measured directly but derived algebraically from the other reported metrics.

Model	Precision	Recall	F1-score	mAP@0.5
YOLOv7	0.939	0.880	0.909	0.901
YOLOv11	**0.962**	**0.901**	**0.931**	**0.902**
YOLO26	0.919	0.890†	0.905	0.871

† Derived from the measured precision and F1-score, not evaluated directly.

YOLOv11 improved on the YOLOv7 baseline across the board, though by margins that are modest and uneven: 2.3 percentage points of precision and 2.1 of recall, but only

0.1 percentage points of mAP@0.5. The last figure means the two architectures locate defects about equally well and differ mainly in how confidently they classify what they have found.

YOLO26 then reversed most of that gain. Against YOLOv11 it gave up 4.3 percentage points of precision, 3.1 of mAP@0.5, and 2.6 of F1-score, leaving it below even the YOLOv7 baseline on precision and mAP@0.5. Precision carries the most practical weight among these three: lower precision means more false-positive detections, and every spurious detection consumes operator review time in a workflow whose whole purpose is to reduce manual effort. The reduced F1-score confirms that no compensating recall gain offset this, and the drop in mAP@0.5 points to weaker localization as well as weaker classification.

The likely explanation lies in what the newer architecture was designed to optimise. YOLO26 targets efficient, NMS-free deployment, and those changes do nothing in particular for the conditions that make sewer footage hard: low contrast between a defect and the surrounding pipe wall, boundaries blurred by camera motion, and backgrounds cluttered by sediment, water ingress, and uneven artificial lighting. An architecture tuned for inference efficiency on general-purpose benchmarks has no reason to handle those conditions better than its predecessor, and on this dataset it did not.

The finding repeats the pattern established across the generations compared earlier in this section. Release order is a poor guide to performance on inspection data, and on the twelve-class dataset examined here YOLOv11 remains the more suitable choice.

That qualification matters, because the ordering reverses on the other extraction of the same sewer material. Evaluated on the 22,427-frame video sequence dataset, YOLO26 is the strongest of the single-frame generations and leads YOLOv11 on precision, recall, and mAP@0.5 (Table 8.20). The two extractions differ in class count, in the proportion of defect-free frames, and in whether frame order is preserved, so the reversal is not a contradiction; it is a further instance of the same lesson, that a ranking established on one preparation of a dataset does not carry to another preparation of the same footage, let alone to a different domain.

8.2.6. Lightweight Architectural Modifications of YOLOv7

Comparing published architectures answers only half of Hypothesis H1, which concerns modified architectures and the balance they strike between accuracy and resource consumption. The other half requires changing an architecture deliberately and measuring what the change costs. Nine structural modifications were therefore applied to the YOLOv7 baseline and evaluated on the twelve-class Sewer Inspection Dataset under the training protocol of Chapter 3.. Each modification was applied in isolation so that its effect could be attributed unambiguously.

The modifications fall into three groups. The first reduces the size of the feature extractor, by narrowing channel counts, removing layers, doing both together, or replacing standard convolutions with depthwise separable ones. The second thins the feature aggregation path, either by removing ELAN blocks or by narrowing the SPPCSPC module. The third alters the detection head itself, by removing detection scales or by deleting the branch responsible for large or small objects. Results are given in Table 8.8.

Table 8.8: Effect of fourteen structural modifications applied individually to the YOLOv7 baseline, twelve-class Sewer Inspection Dataset. The upper block of nine removes capacity (Section 8.2.6.); the lower block of five adds it (Section 8.2.7.), each configuration trained three times from different random seeds ($n = 3$) with means reported. Bold marks the best value in each column across the whole table.

Model variant	**Precision**	**Recall**	**mAP@0.5**
YOLOv7 baseline	0.939	0.880	0.901
Reduced width	0.936	0.873	0.895
Reduced depth	0.934	0.869	0.891
Reduced width and depth	0.932	0.865	0.887
Depthwise convolutions	0.928	0.861	0.884
Fewer ELAN blocks	0.931	0.867	0.889
Fewer detection scales	0.935	0.852	0.878
Removed large-object head	0.937	0.874	0.895
Removed small-object head	0.941	0.831	0.864
Reduced SPPCSPC channels	0.935	0.874	0.895
Add one ELAN block	**0.942**	0.888	0.907
Add several ELAN blocks	0.940	0.891	0.908
Add small-object head	0.936	**0.902**	**0.913**
Add large-object head	0.940	0.884	0.904
Add detection scale	0.938	0.898	0.911

None of the nine reductions in the upper block improved on the baseline, which is

the expected outcome when capacity is removed from a network that was not overparameterised for the task. (The five additions in the lower block do improve on it, and are treated separately in Section 8.2.7..) What matters for deployment is the size of each penalty, and these differ by a factor of six across the nine variants.

Three modifications cost almost nothing. Removing the large-object detection head, narrowing the SPPCSPC module, and reducing width each gave up 0.6 percentage points of mAP@0.5. The first of these is the most informative. Sewer defects occupy a small share of the frame, large-object predictions contribute little on this data, and the branch dedicated to them can be deleted with a recall cost of 0.6 percentage points. For a system targeting embedded hardware this is close to free capacity.

The opposite case is removing the small-object head, which produced the largest loss in the table: recall fell from 0.880 to 0.831 and mAP@0.5 from 0.901 to 0.864. That variant also recorded the highest precision of any reduction, 0.941, which is not a contradiction but a direct consequence of what was removed. With the small-object branch gone the detector no longer attempts the predictions it was least certain about, so the detections it does make are more often correct while a larger number of genuine defects go unreported. Precision alone is therefore a misleading summary of this variant, and in an inspection setting the trade is a poor one, since an unreported defect is the expensive failure.

Reducing the number of detection scales produced the second-largest recall loss, 2.8 percentage points, for reasons that follow from the same argument: fewer scales means coarser coverage of the object-size range that defects actually span. Depthwise convolutions were the most costly of the feature-extractor modifications at 1.7 percentage points of mAP@0.5, consistent with their weaker mixing of information across channels.

Taken together, the table supports the part of H1 concerning resource consumption in a specific and usable way. Architectural cost can be removed from YOLOv7 for this task, but where it is removed matters more than how much. Modifications targeting the large-object pathway and the width of the aggregation module are nearly free, while anything that reduces the network's ability to resolve small objects is expensive.

8.2.7. Capacity Additions to YOLOv7

Removing capacity establishes which parts of the architecture the task depends on. It does not establish whether the baseline is short of capacity anywhere, and that is a separate question with a separate answer. Five additions were therefore trained and evaluated under the same protocol, each applied individually to the same YOLOv7 baseline, and each trained three times from different random seeds. Run-to-run spread is small relative to the differences between variants: the strongest configuration, the additional small-object head, gave 0.936 ± 0.003 precision, 0.902 ± 0.006 recall, and 0.913 ± 0.005 mAP@0.5 across its three seeds, so the ordering below is not an artefact of a single favourable initialisation.

Every addition improved on the baseline, so the baseline was not saturated. The improvements are small, however, and none reaches the magnitude of the losses caused by the more damaging removals in Table 8.8. Capacity is worth more when it is present than additional capacity is worth when added, which is the asymmetry that makes lightweight variants attractive for this task.

Read alongside the removal study, the two tables agree on where the architecture is sensitive, and the agreement is quantitative. The small-object head is the clearest case: removing it cost 3.7 percentage points of mAP@0.5, the largest penalty recorded, and adding a second one produced the largest gain, 1.2 points, together with the highest recall of any configuration tested (0.902). Adding a detection scale, which serves a related purpose, was the second most effective change at 1.0 points. The dataset explains both results: sewer defects are predominantly small and medium in extent, so the pathways that resolve them carry most of the detection burden.

The large-object head behaves as the mirror image. Removing it cost 0.6 percentage points, among the cheapest reductions available, and adding a second one returned only 0.3. On this data the large-object pathway is close to redundant in both directions, which is a useful thing to know when trimming a model for embedded deployment: it is the first component to remove and the last worth adding.

The two ELAN variants show diminishing returns clearly. One additional block reached 0.907 mAP@0.5 and several additional blocks reached 0.908, a difference of one tenth of a percentage point that is smaller than the seed-to-seed spread reported above.

Depth in the aggregation path is not the constraint on this task, and the extra parameters that several blocks introduce buy nothing measurable over one.

The pattern across both tables is consistent. Every modification that helps or hurts materially acts on the network's ability to resolve *small* objects – the small-object head, the number of detection scales – while modifications to depth, width, and the large-object pathway are close to neutral in both directions. Architectural effort on this task is worth spending on scale coverage at the small end and essentially nowhere else.

One limitation applies to both studies and should be stated plainly. Parameter counts, FLOPs, and inference throughput were not recorded for the fourteen variants, so the accuracy side of the trade-off is quantified here while the efficiency side rests on the structural argument for each modification. Completing that measurement is identified as future work in Chapter 9..

8.2.8. Box-Regression Loss Variants

Section 4.5. traced the development of IoU-based localization losses, each adding a geometric constraint the previous formulation lacked. YOLOv7 uses CIoU by default, so the baseline of Table 8.4 is already a CIoU configuration. Three alternatives were substituted for it, changing nothing else in the training recipe, to establish whether the geometric refinements that distinguish them matter on pipeline defect data. Results are given in Table 8.9.

Table 8.9: Second-order architectural and localization-loss modifications on the twelve-class Sewer Inspection Dataset, each applied in isolation to the YOLOv7 baseline. CIoU is the YOLOv7 default and therefore coincides with the baseline row. F1 is computed from the reported precision and recall.

Variant	**Precision**	**Recall**	**mAP@0.5**	**F1**
YOLOv7 baseline (CIoU, no attention)	0.939	0.880	0.901	0.909
(a) Attention modules				
+ SE	**0.947**	0.863	0.904	0.903
+ ECA	0.944	0.876	**0.906**	0.909
+ CBAM	0.938	**0.887**	0.898	**0.912**
(b) Box-regression loss				
GIoU	0.936	0.876	0.898	0.905
DIoU	0.938	0.879	0.900	0.908
SIoU	0.938	0.884	0.904	0.910

The four losses span 0.6 percentage points of mAP@0.5 in total, and they order themselves exactly as the theory in Section 4.5. predicts. GIoU, which penalises only the area of the enclosing box not covered by the union, is weakest at 0.898. DIoU, which replaces that with an explicit centre-point distance, recovers to 0.900. CIoU, which adds an aspect-ratio term on top of the distance term, reaches 0.901. SIoU, which additionally models the *direction* in which the predicted box should move, is best at 0.904.

That the ordering reproduces the chronological development of these losses is not guaranteed: a refinement motivated by general-purpose benchmarks need not help on a domain where object statistics differ sharply. Here each successive geometric constraint does carry a small benefit, which suggests the constraints are capturing something real about box regression instead of fitting the benchmarks they were developed against.

SIoU's advantage is concentrated in recall (0.884 against CIoU's 0.880), not precision, which is consistent with its mechanism. The angle term shortens the path a predicted box travels during optimisation by giving it a direction as well as a distance, and the boxes that benefit most from a shorter path are those that start furthest from their target – typically the small, low-contrast instances the detector is least confident about. Recovering those raises recall without changing how the detector treats predictions it was already making confidently.

The practical significance is nonetheless limited. Switching from CIoU to SIoU is free at inference, so it is worth taking, but 0.3 percentage points is an order of magnitude below what the classification loss (3.5) or the augmentation scheme (3.3) deliver against the same baseline. The result belongs with the attention finding below: refinements to how the detector regresses and weights features it has already extracted produce small, second-order gains, while the substantial improvements come from changing what it is shown and what it is asked to optimise.

8.2.9. Attention Modules

Chapter 4. reviewed channel and spatial attention as the most frequently reported architectural route to better detection of small, low-contrast targets, and Chapter 2. noted that gains demonstrated on general benchmarks need not transfer to inspection imagery. All three modules were inserted at the same point in the YOLOv7 backbone

and trained under the protocol of Section 8.2.3., changing nothing else. Results are given in Table 8.9. Each configuration was trained once, without repeated seeds, and the total spread across the three modules is below one percentage point of mAP@0.5; the values below are therefore point estimates, and the ordering between them is not established as statistically reliable.

The three modules separate cleanly by type. The two channel-only modules, SE and ECA, both move the detector toward precision: SE by 0.8 percentage points and ECA by 0.5, each at the cost of recall. The one module that adds a spatial term, CBAM, moves in the opposite direction, gaining 0.7 points of recall and giving up 0.3 of mAP@0.5. Channel recalibration sharpens the evidence the network already has and makes it more selective; a spatial attention map highlights candidate regions and makes the detector more willing to propose them. Attention does not lift the precision–recall trade-off encountered throughout this chapter, it selects a position on it, in the same way the loss functions of Table 8.12 do.

ECA obtained the highest point estimate of the three on mAP@0.5 at 0.906, half a point above the baseline and marginally above SE, while matching the baseline F1 exactly. Its margin over SE is well inside single-run variation and is not claimed as a reliable ranking. The direction is nonetheless consistent with the argument ECA was designed around and set out in Section 4.4.3.: the dimensionality-reduction bottleneck in SE's excitation step is not merely a parameter saving but a mild information loss, and removing it in favour of a one-dimensional convolution recovers a little accuracy while also being the cheapest of the three to run. Where an attention module must be chosen for this task, ECA is therefore a defensible default, chosen on cost as much as on an accuracy margin this experiment cannot resolve.

The size of these effects limits the conclusion. The largest movement is 0.5 percentage points of mAP@0.5, against 3.5 from the loss function and 3.3 from the augmentation scheme measured against the same baseline on the same dataset. Repeated seeds were not run for these three configurations, and half a point is within the range seed variation alone could produce. Attention modules are therefore approximately performance-neutral here: they redistribute precision and recall predictably, but supply no accuracy the baseline was not already capable of.

This negative result is worth stating plainly because it runs against the weight of

the sewer-inspection literature surveyed in Section 2.2.1., in which attention modules are among the most frequently reported sources of improvement. A plausible explanation for the discrepancy is the nature of the imagery. Sewer defects are defined by texture over a scene whose global geometry is nearly constant from frame to frame, so there is little spatial saliency for an attention map to discover: the informative region is most of the pipe wall, most of the time. The interventions that did move performance in this dissertation all changed what the network was shown or asked to optimise, not how it weights what it has already extracted.

8.2.10. Impact of Contextual Background Information

In addition to architectural design choices, object detection performance can also be influenced by the amount of contextual information available during training. While object detectors are typically trained using annotations that include surrounding image content, it remains unclear whether the presence of background information contributes positively to defect recognition performance. To investigate this question, an additional experiment was conducted using two alternative annotation strategies, following the protocol the author established for sewage pipeline inspection in [188].

This experiment uses a narrower subset than the rest of this section: 11,510 images covering only the four most frequent EN 13508-2 codes (BAJ, BAK, BBB, BBC), split 9,208 / 1,151 / 1,151 [188]. Restricting to the four common classes isolates the effect of background context from the confounding effect of class scarcity, which is the subject of Section 8.3.. Because the label set and image count differ, the absolute values in Table 8.10 are not comparable with those in Table 8.4; only the difference between the two rows is interpreted.

The first approach used the original training images containing both the target object and its surrounding context. The second approach removed the surrounding background entirely: each annotated object was cropped out, rescaled so that its larger dimension met a fixed size, and padded to a uniform 480×480 frame, so that the whole image is the bounding box. Both datasets were subsequently used to train identical YOLOv7 models from scratch under the same experimental conditions, and both were evaluated on the test split of the original, context-preserving dataset.

Table 8.10: Influence of contextual background information on YOLOv7 detection performance, four-class 11,510-image subset [188].

Model	Precision	Recall	mAP
YOLOv7	0.957	0.920	**0.930**
YOLOv7@NoBackground	**0.962**	0.910	0.921

Removing the background costs more than it gains: precision rises marginally, from 0.957 to 0.962, but recall falls from 0.920 to 0.910 and mAP from 0.930 to 0.921.

The drop implies that pipe geometry, surrounding material, illumination, and the spatial relationship between infrastructure elements function as detection cues, not noise: cropping them out simplifies the image but removes information the network was using to distinguish defects from visually similar structures. Object appearance alone is therefore not sufficient for this task – dataset design and annotation strategy affect detector performance measurably, and should be considered alongside architectural choices.

8.2.11. Impact of On-Screen Overlay Masking

A related preprocessing question, distinct from the background-cropping experiment above, is whether masking the fixed-position on-screen overlays (timestamp, marching speed, and distance/orientation indicators) that CCTV inspection systems burn directly into the video frame actually helps detection, or is simply good preprocessing hygiene with no measurable effect. [175] tested this directly on the sewer dataset by training the same YOLOv7 configuration on three otherwise-identical versions of the same footage: with the overlay fully masked (Custom-AF), with only the text portion masked but the orientation indicator left in place (Custom-TF), and with neither masked (Custom-TO). Results are given in Table 8.11.

Table 8.11: Effect of on-screen overlay masking on YOLOv7 detection performance, sewer dataset [175].

Dataset variant	Precision	Recall	mAP
Custom-AF (text + orientation masked)	0.939	0.880	0.901
Custom-TF (orientation indicator only)	0.921	0.891	0.913
Custom-TO (no masking)	**0.951**	0.855	**0.922**

The result runs counter to the expectation that removing an irrelevant fixed-position overlay should only help: full masking (Custom-AF) produced the *lowest* mAP of the

three (0.901), while leaving both overlays in place (Custom-TO) produced the highest (0.922) and the highest precision (0.951), at the lowest recall (0.855). Two explanations are possible and were not distinguished experimentally [175]: the distance readout may act as an incidental but genuinely useful spatial feature, or the difference may be ordinary training variance. The dual-stream ablation on the video sequence dataset (Table 8.22) points the other way by an equally negligible margin, masking raising mAP by 0.001. The two experiments bracket zero. Masking is therefore retained here to prevent a fixed-position artifact being learned as a spurious feature (Chapter 3.), not because it reliably improves accuracy.

That failure mode is an instance of a well-documented phenomenon. Lapuschkin et al. showed that classifiers frequently achieve strong benchmark scores by exploiting incidental correlations in the data instead of the intended object evidence, a behaviour they term "Clever Hans" prediction [248]. A burned-in distance readout that advances monotonically as the camera travels along the pipe is such a correlate: it carries genuine information about position without carrying any information about defect condition. The masking decision in this dissertation is therefore a guard against a known generalisation risk, and the ambiguous measured effect reported above does not remove the justification for it.

8.2.12. Analysis of Detection Accuracy and Computational Efficiency

Two findings cut across the experiments above. The first is that architectural improvements do not translate into proportional accuracy gains, and that which architecture wins depends on the dataset: YOLOv5 led on the underwater data while YOLOv7 variants led on the sewer data, and on the twelve-class sewer extraction YOLO26 trailed YOLOv11 despite being the later release, though it leads it on the video sequence extraction of the same footage. The second is that dataset preparation matters at least as much as architecture selection. The contextual background experiment showed that preserving surrounding scene content raises recall and mAP over training on isolated object crops, and the modified Mosaic study placed a purely data-level change to YOLOv7 above an unmodified YOLOv8.

The two datasets also disagree about transfer learning, informatively and not contra-

dictorily. COCO pretraining was worth 3.2 percentage points of mAP on the underwater data (94.21% against 90.98%), while training from scratch was better in five of six paired comparisons on the sewer data (Table 8.4). The likely explanation is domain overlap with the COCO distribution. Underwater imagery contains large, coherent objects with intact silhouettes – a pipeline, a concrete mat – for which generic low- and mid-level features transfer. Sewer defect classes are texture-defined, not shape-defined: a crack, a deposit, or a defective lining is a deviation in surface appearance within a scene whose global geometry is constant. Features learned on natural images offer little there, and the pretrained initialisation appears to act as a constraint the network must unlearn. Transfer learning should therefore be treated as a per-domain hypothesis to be tested, particularly where target classes are textural.

8.3. Impact of Class Imbalance Mitigation

This section evaluates the three families of imbalance mitigation reviewed in Chapter 5. – data augmentation, dataset balancing, and imbalance-aware loss functions – on the twelve-class Sewer Inspection Dataset. All three are measured against the same YOLOv7 baseline, dataset, and splits, so their results can be read against one another as well as within themselves.

8.3.1. Effect of Data Augmentation Techniques

The augmentation methods evaluated here are geometric transformations, photometric transformations, MixUp, Mosaic, and Copy-Paste (described in Chapter 5.). Each configuration trains an otherwise-identical model with one technique applied in isolation; Table 8.12 reports the resulting Precision, Recall, mAP@0.5, mAP@0.5:0.95, and Rare-Class AP, the last averaged over the dataset's least-represented defect categories. F1-scores quoted in the discussion below are computed from the reported precision and recall, not tabulated separately.

Every technique improved on the baseline in recall, mAP@0.5, and Rare-Class AP, and every technique cost precision to do it. The consistency of that pattern identifies the mechanism the five techniques share. Augmentation increases the variety of appearances

Table 8.12: Class imbalance mitigation on the twelve-class Sewer Inspection Dataset. All three families share the YOLOv7 baseline of Table 8.4, the same data and the same splits; incomplete metric recording and potentially unequal tuning effort across families nonetheless limit strict cross-family ranking. Rare-class metrics are averaged over the least-represented defect categories and are computed from the per-class breakdown of Table 8.14, whose baseline column is the from-scratch YOLOv7s configuration; the aggregate columns of the baseline row are the COCO-initialised YOLOv7. A dash marks a metric not recorded for that family.

Configuration	**Prec.**	**Rec.**	**mAP@0.5**	**mAP@.5:.95**	**RC-Rec.**	**RC-AP**
Baseline						
YOLOv7 baseline	**0.939**	0.880	0.901	0.631	0.596	0.491
(a) Data augmentation						
Geometric	0.936	0.898	0.914	0.644	–	0.521
Photometric	0.931	0.910	0.923	0.653	–	0.541
MixUp	0.934	0.888	0.908	0.627	–	0.506
Mosaic	0.923	0.918	0.931	0.661	–	0.566
Copy-Paste	0.914	**0.923**	0.934	**0.664**	–	**0.591**
(b) Dataset balancing						
Oversampling	0.924	0.900	0.912	–	0.684	0.536
Undersampling	0.916	0.891	0.897	–	0.652	0.519
Hybrid sampling	0.921	0.907	0.916	–	0.712	0.552
Class-aware sampling	0.918	0.914	0.921	–	**0.741**	0.571
(c) Loss function adaptation						
Weighted Loss	0.925	0.899	0.914	–	–	–
Class-Balanced Loss	0.916	0.912	0.924	–	–	–
Focal Loss	0.903	**0.931**	**0.936**	–	0.863	–

the detector treats as belonging to a class, which recovers defects that would otherwise be missed while also admitting predictions that turn out to be wrong. The techniques differ in how far they push along that trade, not in which direction they push.

The ordering along the trade is systematic. The techniques that intervene most aggressively in the composition of the training image sit at the far end of it: precision falls from 0.939 at the baseline to 0.923 for Mosaic and 0.914 for Copy-Paste, the two techniques that recompose the image outright, while the transformations that leave image composition intact (geometric, photometric) and the blending technique that leaves object positions intact (MixUp) cost between 0.003 and 0.008. F1-score computed from the tabulated precision and recall balances the two, and peaks in the middle of the range at 0.920 for both Photometric and Mosaic, falling marginally for Copy-Paste (0.918) despite its higher recall. A configuration should therefore be chosen according to which error is more costly in deployment, and for pipeline inspection that is the missed defect.

The ordering within that trade is explained by how much novelty each technique adds to this particular material. Geometric transforms gain little (0.914 mAP@0.5) because inspection footage already contains substantial viewpoint variation, the camera advancing and rotating continuously within the pipe. Photometric augmentation does better (0.923) because non-uniform lighting, reflection off wet surfaces, and haze from suspended sediment are the primary causes of missed detections in sewer footage, so simulating them addresses the dominant failure mode. MixUp gains least of all (0.908) and is the only configuration to fall below the baseline on any metric, dropping mAP@0.5:0.95 from 0.631 to 0.627: linear blending leaves both sets of defect edges semi-transparent, which is tolerable for the coarse localization mAP@0.5 rewards and harmful for the tight localization mAP@0.5:0.95 requires.

The two techniques that recompose the image outright do best. Mosaic (0.931 mAP@0.5, 0.566 rare-class AP) exposes the detector to more objects, backgrounds, and apparent scales per training sample, addressing small object size and class scarcity together, though it improves the sampling of everything at once instead of concentrating where the need is greatest. Copy-Paste leads on overall mAP@0.5 (0.934), mAP@0.5:0.95 (0.664), recall (0.923), and rare-class AP (0.591, ten points over the baseline), because it controls directly which rare instances are duplicated and how often. Its precision cost is also the largest, and identifies the technique's limit: pasted instances are composited into frames whose

lighting, scale, and surface texture were not produced with that instance in mind, and the resulting mismatch generates detections that correspond to no real defect. Placement realism, more than the choice or frequency of the pasted class, governs how much of the rare-class gain survives as usable precision.

8.3.2. Evaluation of Dataset Balancing Strategies

To further address the class imbalance problem present in the sewer inspection dataset, four dataset balancing strategies were evaluated: random oversampling (rare-class images repeated more frequently), random undersampling (majority-class images removed), hybrid balancing (moderate oversampling combined with moderate undersampling), and class-aware sampling (a class is selected first, then an image containing that class is drawn, instead of drawing images uniformly). Unlike data augmentation, which increases dataset diversity through image transformations, these strategies directly modify the class distribution observed during training without altering the images themselves. Table 8.12 reports Precision, Recall, and mAP@0.5 together with two class-specific metrics, Rare-Class Recall and Rare-Class AP, both averaged over the least-represented defect categories.

This comparison shares the YOLOv7 baseline, dataset, and splits used for the augmentation ablation above and the loss-function comparison below, so all three families of mitigation are measured against one reference and can be read against one another directly.

The four strategies order themselves consistently: class-aware sampling leads on every metric except precision, followed by hybrid, then oversampling, with undersampling last. Each is examined in turn below.

The baseline row is the reference, not a strategy: images are drawn at their natural frequency, so frequent classes dominate every minibatch as they dominate the raw data, giving the weakest rare-class recall (0.596) and AP (0.491) alongside the highest precision (0.939).

Random oversampling raises rare-class recall to 0.684 and AP to 0.536 for 1.5 points of precision, the smallest return of the three strategies that increase minority exposure, because it creates no new visual information. Random undersampling is the only strictly dominated configuration, costing the most precision for less rare-class recall than oversam-

pling and the only mAP@0.5 below baseline. The explanation is specific to video-derived data: majority-class frames are not interchangeable duplicates but carry the variation in lighting, camera distance, pipe material, and clutter the detector must discriminate against, and because consecutive frames at two per second are already correlated, the dataset has less genuine redundancy to give up than its size suggests.

Hybrid sampling exceeds both parents instead of landing between them (0.712 rare-class recall, 0.916 mAP@0.5), which indicates the two mechanisms offset each other's failure modes: moderate oversampling raises minority exposure without repeating any instance often enough to overfit it, while moderate undersampling reduces majority dominance without cutting into informative variation. Class-aware sampling is strongest on every metric except precision (0.741 rare-class recall, 0.571 AP, 0.921 mAP@0.5) because it acts on the quantity that matters. Image-level oversampling controls how often an *image* is drawn, and those images usually contain frequent classes too, so majority exposure rises alongside minority exposure; selecting the class first makes per-class exposure a directly controlled quantity. It is also the only strategy that alters neither the contents nor the size of the dataset.

Because both studies share the baseline of Table 8.4, the two families can be compared directly. On rare-class AP the best augmentation technique reaches 0.591 against the best sampling strategy's 0.571, and on mAP@0.5, 0.934 against 0.921 – comparably effective. Augmentation is marginally ahead because it adds new visual variability where sampling can only change how often existing pixels are shown, which matters most for the rarest classes, where the pool of distinct real examples is smallest. That is also why the two are plausibly complementary: sampling determines how often rare classes appear, augmentation how much they vary when they do.

8.3.3. Dataset Filtering and Negative Sampling

Beyond the four balancing strategies above, two further dataset-level interventions were evaluated on the sewer dataset and reported in [175]: removing the rarest defect classes below a fixed instance-count threshold, and adding real, unannotated "non-target" background frames to the training set. Both modify the training distribution directly instead of through image-level augmentation, and both are distinct from the oversampling,

undersampling, hybrid, and class-aware sampling strategies discussed above.

Table 8.13: Two dataset-level interventions on the sewer dataset (YOLOv7s) [175]. The upper block removes defect classes falling below an instance-count threshold; the final row adds 13,000 real background images containing no annotated defect to the four-class filtered training set. The sewer annotation effort itself yields 10,460 defect-free frames (Table 3.3), so the background images used here must include material drawn from outside it; the source study does not record their provenance.

Dataset Version	Precision	Recall	mAP	Classes
Custom-AF (no filtering)	0.939	0.880	0.901	12
Custom-AF@500	0.950	0.913	0.928	6
Custom-AF@1000	0.957	0.920	0.930	4
Custom-AF@1500	**0.966**	0.910	0.933	3
Custom-AF@1000 + 13,000 non-target	0.923	**0.935**	**0.947**	4

Table 8.13 shows every aggregate metric improving monotonically as rarer classes are dropped, from 0.901 mAP across all twelve classes to 0.933 across three. The metric improves specifically *because* the classes the detector struggled with are no longer scored, not because the remaining classes became easier. A 0.901-mAP model that still detects all twelve codes, including operationally critical ones such as BAA and BAB, is of more practical use than a 0.933-mAP model no longer evaluated on rare structural defects [175]. The result is included as a cautionary counterpoint to Tables 8.12 and 8.6: an mAP gain is evidence of progress on class imbalance only if the number and identity of the scored classes has not also changed.

The final row evaluates a different intervention: adding 13,000 real background images to the four-class filtered training set. Recall and mAP both improved (0.920 to 0.935; 0.930 to 0.947) at a small precision cost, consistent with the detector learning a sharper decision boundary from exposure to genuinely defect-free backgrounds as well as to frames already containing an annotated instance. The mechanism differs from every augmentation technique in Table 8.12: those increase the diversity or frequency of positive examples, whereas negative sampling increases the diversity of true negatives, and the gain confirms the two are not redundant.

8.3.4. Evaluation of Loss Function Adaptations

Although data augmentation and dataset balancing strategies improved minority-class representation, class imbalance remained present during optimization. To further improve

the detection of rare defects, three loss function adaptations were investigated: inverse-frequency weighting, class-balanced loss, and focal loss. All three were trained against the same YOLOv7 baseline, on the same twelve-class dataset and the same splits, changing only the classification term of the composite objective in Equation 6.5. Results are given in Table 8.12.

Moving down the rows, precision falls monotonically (0.939 to 0.903) while recall (0.880 to 0.931) and mAP@0.5 (0.901 to 0.936) rise monotonically. No configuration improves both sides. The three adaptations do not represent three different ideas about how to handle imbalance so much as three positions along a single trade-off, ordered by how aggressively each reweights the objective away from frequent, easily-classified examples.

The exchange rate is also nearly constant. Weighted Loss buys 1.3 percentage points of mAP for 1.4 of precision, Class-Balanced Loss 2.3 for 2.3, and Focal Loss 3.5 for 3.6 – ratios of 0.93, 1.00, and 0.97. The recall-to-precision exchange is equally stable at 1.36, 1.39, and 1.42. The three mechanisms differ substantially in construction: Weighted Loss scales by raw inverse frequency, Class-Balanced Loss by effective sample number, and Focal Loss by per-example difficulty, not by class at all. That three such different formulations land on a similar exchange rate suggests that, under this specific training configuration, the three loss adaptations produced comparable precision–mAP trade-offs. Repeated runs and additional architectures would be required to determine whether the pattern reflects a property of the data or is specific to this setup.

The practical consequence is that the choice between them is a choice of operating point, not of method. For sewer inspection, where a missed structural defect costs far more than a false alarm an operator dismisses in a second, the correct position on that curve is the far end, and Focal Loss is the configuration that reaches it. Where precision matters more – an automated system that files maintenance orders without review, say – Weighted Loss reaches a milder point on the same curve with two-thirds less precision loss.

Focal Loss additionally has the per-class evidence behind it that the other two do not. Its effect on the rarest categories, examined in detail below, is a rise in averaged BBH and BBE recall from 0.596 to 0.863 (Table 8.14). Equivalent per-class breakdowns were not recorded for Weighted and Class-Balanced Loss, so their aggregate gains cannot be attributed to rare classes with the same confidence, even though the direction of their

movement along the shared curve makes that the natural reading.

Class-balanced loss also appears elsewhere in this dissertation in a different role. The Stage-1 anomaly classifier of the cascade framework is trained with Class-Balanced Focal Loss, and the backbone and loss ablation in Section 8.4. reports its effect on binary defect-presence classification [233]. That evaluation concerns a different task and metric set and is kept separate from the multi-class comparison here.

Weighted Loss

Weighted loss raised recall from 0.880 to 0.899 and mAP@0.5 from 0.901 to 0.914 while precision fell from 0.939 to 0.925. It is the mildest of the three adaptations, and its position at the near end of the shared curve follows from how it derives its weights: raw inverse frequency is a fixed per-class multiplier fitted once from the training distribution, so it responds to how rare a class is but not to how difficult. Where scarcity and difficulty coincide the weighting is well targeted; where they do not, it spends gradient on classes that were never the problem. The other two formulations differ precisely in decoupling weight from raw count.

Focal Loss

Focal Loss produced the largest measured improvement in rare-defect detection of any technique evaluated in this dissertation. Against the shared baseline it raised recall from 0.880 to 0.931 and mAP@0.5 from 0.901 to 0.936 at a precision cost of 3.6 percentage points, the furthest movement along the trade-off curve of the three adaptations. Averaged over the twelve classes with equal weight, the view that exposes rare-class behaviour, the same comparison reads 0.828 to 0.903 precision, 0.876 to 0.931 recall, and 0.852 to 0.936 mAP.

The effect is concentrated where the mechanism predicts. Down-weighting easily classified majority-class examples shifts gradient away from classes the baseline already detects reliably and toward those it does not: recall for BBH (vermin) rose from 0.647 to 0.884 and for BBE (obstructions) from 0.544 to 0.842, roughly 27 percentage points on the two classes the baseline handled worst, while BBC, BAI, and BBB each moved by less than 0.01. The redistribution has a cost side, visible in BAJ falling from 0.951 to 0.922 and

BAA more noticeably from 0.940 to 0.869, and the technique is therefore reported here as a trade. Since several of the operationally most significant sewer defect categories are also the rarest, that trade is the favourable one for inspection.

The full per-class breakdown is given in Table 8.14, and Figure 8.6 shows the recall column graphically.

Table 8.14: Per-class performance of the baseline YOLOv7s and the focal-loss-adapted YOLOv7-ModifiedLoss on the twelve-class Sewer Inspection Dataset [175]. The final row is the macro average over classes.

	YOLOv7s (baseline CE)			**YOLOv7-ModifiedLoss (focal)**		
Class	P	R	mAP	P	R	mAP
BAA	0.915	**0.940**	0.964	**0.929**	0.869	**0.968**
BAB	0.934	0.969	**0.981**	**0.941**	**0.974**	0.980
BAC	0.982	0.923	0.990	**0.998**	**0.995**	**0.997**
BAH	0.603	0.755	0.684	**0.800**	**0.877**	**0.862**
BAI	**0.971**	**0.961**	**0.965**	0.970	0.960	0.964
BAJ	**0.950**	**0.951**	**0.949**	0.941	0.922	0.940
BAK	0.920	**0.970**	**0.956**	**0.921**	0.966	0.942
BBB	**0.929**	**0.964**	**0.947**	0.915	0.961	0.931
BBC	**0.965**	**0.989**	**0.980**	0.960	0.980	0.969
BBF	0.670	0.895	**0.956**	**0.860**	**0.941**	0.947
BBH	0.512	0.647	0.478	**0.742**	**0.884**	**0.884**
BBE	0.578	0.544	0.380	**0.860**	**0.842**	**0.842**
Macro avg.	0.828	0.876	0.852	**0.903**	**0.931**	**0.936**

The table makes the redistribution explicit. Measured on mAP, focal loss wins on five of the twelve classes and loses on seven, but the wins and losses are of very different magnitudes. The three largest gains are on the three worst-performing classes in the baseline: BBE (mAP 0.380 to 0.842), BBH (0.478 to 0.884), and BAH (0.684 to 0.862). The losses are all on classes the baseline already handled well and are all under 0.02 mAP, with the single exception of BAA's recall. In an inspection setting that asymmetry is the right one: 46 percentage points of mAP recovered on obstructions is worth more than one or two points conceded on deposits, because the deposits will be detected in the next frame anyway and the obstruction may not be.

The effect is visible qualitatively as well. Figure 8.5b compares detections from the focal-loss-adapted model against the unmodified baseline on the same frames, where the adapted model recovers defect instances the baseline either misses or assigns lower confidence.

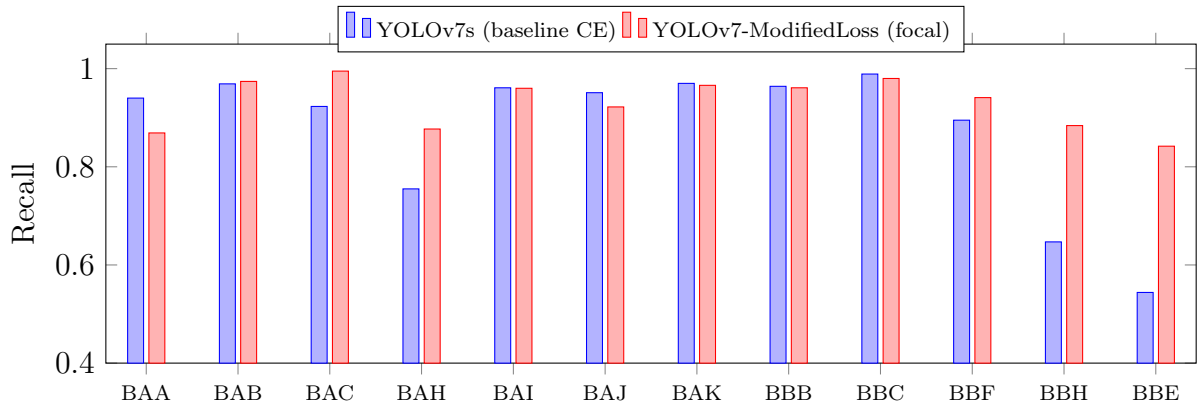


Figure 8.6: Per-class recall on the sewer dataset, baseline cross-entropy vs. focal-loss-adapted YOLOv7 [175]. The largest gains fall on the two rarest classes, BBH and BBE.

Class-Balanced Loss

Class-balanced loss weights each class by its effective number of samples instead of raw inverse frequency, raising recall to 0.912 and mAP@0.5 to 0.924 for 2.3 points of precision, between the other two on every metric. The construction predicts that intermediate position. Raw inverse frequency treats a class appearing ten times less often as needing ten times the weight, over-correcting for classes whose scarcity does not make them proportionally harder to learn; effective-number weighting saturates as counts grow, so it pushes further without the instability an unbounded weight would introduce on classes with a handful of instances. It stops short of focal loss because it still assigns one weight per class and cannot distinguish a hard example from an easy one within a category.

The three rows therefore trace a progression in what the weighting is allowed to depend on – raw count, then effective count, then per-example difficulty – and each loosening of that constraint moves the model further along the same curve.

8.3.5. Discussion of Rare Defect Detection Performance

Because all three families share the baseline of Table 8.4, they can be ranked against one another, and the ranking is close. On overall mAP@0.5 the best of each family reaches 0.936 (Focal Loss), 0.934 (Copy-Paste), and 0.921 (class-aware sampling). On rare-class performance the loss-level intervention is clearest, raising averaged BBH and BBE recall from 0.596 to 0.863 against class-aware sampling's 0.741 on the same metric. No family is ineffective and none dominates by a wide margin.

Three interventions acting at different stages of the pipeline – what the training images contain, how often each class is drawn, and how errors are weighted in the objective – arrive within 1.5 percentage points of mAP@0.5 of one another. Each is capable of supplying most of the available minority-class gradient signal on its own, which supports H3 (Chapter 1.) and simultaneously suggests they are substitutes: a system needs one of them, not necessarily all three.

Whether they compound is the open question this dissertation does not resolve. Combining the best of each family could plausibly add little, since all three address the same deficit and the first application of any of them captures most of the headroom. It could equally exceed each alone, since they act on different quantities: augmentation on how much the rare classes vary, sampling on how often they appear, and the loss on how heavily their errors count. Testing that combination directly is a natural next step (Chapter 9.).

8.4. Evaluation of the Cascade Detection Framework

The proposed cascade framework was evaluated on the 22,427-frame, 10-class video sequence dataset (Chapter 3., Section 3.6.), since its binary pre-filtering stage operates on the same continuous frame stream used by the spatiotemporal experiments later in this chapter, and not on the pre-filtered, per-image 13,000-image Sewer Inspection Dataset used in Chapter 5.. The evaluation focused on two key objectives. First, the effectiveness of the binary anomaly detection stage was analyzed to determine its ability to distinguish defective from non-defective frames. Second, the overall performance of the complete cascade framework was compared with a conventional single-stage object detection approach. Particular emphasis was placed on recall, computational efficiency, inference speed, and the reduction of unnecessary processing associated with defect-free inspection frames.

8.4.1. Performance of the Binary Defect Detection Stage

The first stage of the cascade framework serves as a lightweight anomaly detector responsible for distinguishing defective and non-defective inspection frames. Since any frame incorrectly classified as defect-free would be excluded from subsequent processing, recall represents the most important evaluation metric. The objective of the first stage is therefore to maximize defect preservation while maintaining low computational complexity.

MobileNetV3-Large gave the best result on every metric, and its margin over the next-best backbone is wide on precision, 0.062 over EfficientNet-B0, while narrowing to 0.018 in recall and 0.002 in AUROC, both of those over ConvNeXt-Tiny. Classification quality alone would therefore be a thin basis for the choice, and it is settled on end-to-end throughput instead.

The wider pattern in the table matters more than the ranking itself. Additional capacity brought no benefit on this task: ConvNeXt-Tiny is substantially larger than MobileNetV3-Large and did not surpass it on any metric. Binary defect presence appears to be a sufficiently coarse discrimination that a compact backbone can perform it as well as a heavy one, and that is the premise the cascade design rests on.

A measurement note applies to Table 8.15 above and to every subsequent cascade

Table 8.15: Stage-1 gate: classification performance of the four backbone candidates on the held-out test set, and the loss-function ablation on the adopted backbone [233]. Gate recall is the fraction of defect-bearing *frames* the gate forwards at its adopted operating point, and is the quantity the gate is selected on; it is distinct from the end-to-end cascade recall of Table 8.17, which counts defect *instances*. Pass-through rate and end-to-end throughput were measured only for the configurations carried into the loss ablation, under the FP32 protocol of Section 6.6.1., and over the 20,000-frame survey stream of Table 6.3 ($e = 0.920$) rather than the annotated extraction. Precision, recall, F1 and AUROC are computed at the default 0.5 decision threshold; gate recall is read at the deployed τ_{gate}, where the corresponding Stage-1 false-positive rate is given per threshold in Table 8.18. The adopted configuration additionally reaches an accuracy of 0.966 on the balanced held-out split. Dashes are unmeasured, not zero.

Configuration	Prec.	Rec.	F1	AUROC	Gate rec.	Pass-thr.	E2E FPS
MobileNetV3-L + CB-Focal	**0.952**	**0.978**	**0.965**	**0.985**	**0.983**	15.0%	308
EfficientNet-B0 + CB-Focal	0.89	0.95	0.92	0.981	0.978	17.0%	243
ConvNeXt-Tiny + CB-Focal	0.88	0.96	0.92	0.983	–	–	–
DeiT-Tiny + CB-Focal	0.87	0.94	0.90	0.979	–	–	–
MobileNetV3-L + ASL	–	–	–	0.984	0.976	**12.0%**	**346**
MobileNetV3-L + BCE	–	–	–	0.976	0.965	19.0%	224

ablation in this section. Their throughput figures come from the FP32 end-to-end benchmark described in Section 6.6.1., which runs Stage 2 at 640×640 and measured a 91 FPS single-stage baseline, and not from the comparison protocol, which runs it at the 1280×1280 of Section 6.3.3. and produced the 30.2 and 96.0 FPS results in Table 8.17. The two sets of absolute numbers are therefore never read against each other; what these ablations establish is the ranking between configurations under one consistent protocol.

Recall at the adopted $\tau_{\text{gate}} = 0.35$ varies slightly across the ablations below because each isolates a different Stage-2 setting: the gate-threshold sweep and the headline comparison of Table 8.17 use the cascade's default Stage-2 configuration (0.970); part (b) below fixes Stage-2 to a static confidence of 0.25 specifically to isolate that setting (0.957); and the precision ablation of part (c) uses its own FP32 reference configuration (0.965). The three are not repeated measurements of one configuration and are not compared against each other.

Figure 8.7 shows the ROC and precision–recall curves for the EfficientNet-B0 configuration, which ranks second of the four backbones in Table 8.15 on precision and third on AUROC at 0.981. The curves illustrate the shape of the trade-off from which the gate's operating point is read: the steep initial rise and extended high-recall plateau are

what allow a high-recall, low-false-positive operating point to be selected without a large sacrifice in precision (Section 6.3.2.).

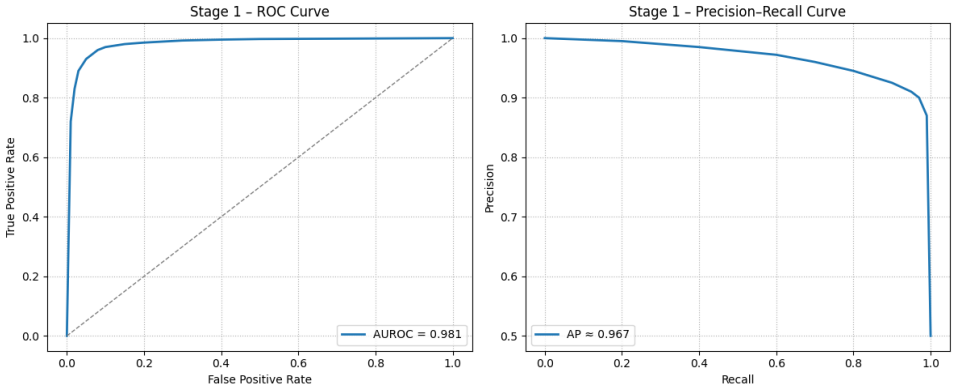


Figure 8.7: Stage-1 ROC curve (AUROC = 0.981) and precision–recall curve (AP ≈ 0.967) for the EfficientNet-B0 + CB-Focal configuration [233].

DeiT-Tiny produced the weakest result on every metric, consistent with vision transformers requiring more data, longer training, or distillation to match convolutional inductive biases on a modest, repetitive, and imbalanced dataset. Pass-through rate and end-to-end throughput were measured only for the two configurations carried into the loss ablation (Table 8.15), so ConvNeXt-Tiny and DeiT-Tiny are compared on classification quality alone.

Table 8.15 confirms this ordering translates into end-to-end cascade throughput as well as classification quality. Under an FP32 benchmark that measured a 91 FPS single-stage YOLOv11 baseline and a Stage-1-classifier-alone rate of approximately 625 FPS for MobileNetV3-L, the complete cascade reaches 308 FPS – consistent with the additive relation $T_{cascade} = T_1 + pT_2$ (Equation 6.11) given a 1.6 ms Stage-1 latency, a 15% pass-through rate, and a roughly 11 ms Stage-2 latency ($1.6 + 0.15 \times 11.0 \approx 3.25$ ms, i.e. 308 FPS).

Replacing MobileNetV3-L with EfficientNet-B0 is worse on both terms of that expression at once: it raises the pass-through rate from 15.0% to 17.0%, so more frames reach the detector, and it is itself slower per frame. The net effect is a markedly slower complete cascade (243 against 308 FPS). This is the measured form of the point made in Section 6.3.2.: because Stage 1 runs on every acquired frame while Stage 2 runs only on the fraction that passes, a gate's own latency is paid in full on the whole stream and cannot be recovered by selectivity downstream.

The loss-function rows hold the backbone fixed at MobileNetV3-L and vary only the training objective, which isolates the loss as a factor in system throughput. ASL trades a small amount of gate retention (0.976 against 0.983) for a lower pass-through rate and the fastest cascade of the four configurations (346 FPS), while plain BCE – without either CB-Focal's effective-number re-weighting or ASL's asymmetric down-weighting of easy negatives – produces both the lowest retention (0.965) and the slowest cascade (224 FPS). A gate trained with an imbalance-unaware objective is thus doubly penalised: it misses more defects *and* forwards more frames, because its decision boundary sits in a worse place, not merely at a different point on a good one. MobileNetV3-L with Class-Balanced Focal Loss is therefore the Stage-1 configuration adopted throughout this dissertation.

8.4.2. Stage-2 Detection Performance

Table 8.16 reports the Stage-2 detector's per-class performance on the frames forwarded by the gate, across the ten EN 13508-2 categories of the video sequence dataset.

Table 8.16: Stage-2 YOLOv11 per-class detection performance on the video sequence dataset [233]. Instance counts are the dataset totals from Table 3.6. BAH is reported as not applicable: with a single annotated instance in the entire dataset and a video-level split, it contributes no evaluable test instances, so no average precision can be computed for it. The mean is therefore taken over the nine scored categories.

Code	Class	Instances	AP@0.5	AP@0.5:0.95
BAA	Deformation	540	0.90	0.70
BAB	Crack	548	**0.94**	**0.77**
BAC	Break	50	0.92	0.74
BAH	Defective Connection	1	N/A	N/A
BAI	Intruding Sealing Material	228	0.85	0.62
BAJ	Pipe Misalignment	5,363	0.90	0.71
BAK	Defective Lining	2,211	0.86	0.64
BBB	Wall Attachments	1,069	0.86	0.63
BBC	Deposits	2,866	0.85	0.61
BBF	Water Ingress	73	0.87	0.66
Mean (mAP), 9 scored classes			**0.883**	**0.676**

One category cannot be scored at all. Defective connection (BAH) carries a single annotated instance across the whole video sequence dataset (Table 3.6), and because the split is performed at the video level that instance falls entirely within one subset. It therefore contributes either zero or one test instance, neither of which supports a

meaningful average precision, and it is reported as not applicable instead of assigned a number the data cannot support.

The support convention applied here, and to the per-class results elsewhere in this dissertation, is the following. A class contributing fewer than five instances to the test split yields an average precision that is arithmetically computable but is not reported as a class AP, because a single additional detection or miss moves the value across a large part of its range and the result carries no information about the detector. Between five and nine test instances a value is reported only with an explicit warning that its support is very low. Ten or more is treated as the minimum at which a per-class figure may be read without that qualification. BAH falls in the first band with at most one test instance, so no value is given for it. The source manuscript for this experiment [233] does report an average precision for BAH; it is not carried over here, because at that level of support the quantity is not interpretable as a measure of detection performance. All means in this table are computed over the remaining nine categories.

Across those nine, the spread is narrow, from 0.85 to 0.94 AP@0.5, which is a different picture from the twelve-class results of Section 8.3.. Two factors account for this. The two rarest codes of the full scheme, BBH and BBE, are absent from this annotation pass entirely, so the hardest cases are not being scored. And the gate has already removed the defect-free frames on which a detector would otherwise accumulate false positives, which lifts precision uniformly. The strongest classes are the structurally distinct ones – cracks (0.94) and breaks (0.92) – while the texture-defined categories of deposits (0.85) and intruding sealing material (0.85) remain hardest, consistent with the pattern throughout this chapter.

Two further categories should be read with caution for the same reason in weaker form. Break (BAC) and water ingress (BBF) carry 50 and 73 annotated instances respectively across the whole dataset, so their test-split populations are small and the corresponding AP values are correspondingly less stable than those of the high-frequency codes.

8.4.3. Comparison Between Single-Stage and Cascade Detection

To evaluate the effectiveness of the proposed framework, the complete cascade architecture was compared with a conventional single-stage object detection system based

on YOLOv11. Both approaches were evaluated using identical inspection datasets and performance metrics.

Table 8.17: Comparison between conventional single-stage detection, two external baselines, and the proposed cascade framework [233]. NFAR is the negative-frame false-alarm rate defined in Section 6.5.: a frame-level quantity over defect-free frames, not comparable with a detection-level false-positive rate. The Recall column is the end-to-end instance-level $\text{Recall}_{\text{cascade}}$ of Equation 6.10, computed over the entire stream. The YOLOv8 and YOLOv11 rows are therefore not comparable with the rows of the same name in Table 8.20, which were measured in a separate campaign under a different protocol; the paragraph following this table sets out the difference.

Method	mAP@0.5	Recall	NFAR	FPS
EfficientDet-D0	0.82	0.90	0.11	24.6
YOLOv8	0.87	0.94	0.09	31.4
YOLOv11 (all frames)	**0.89**	**0.98**	0.08	30.2
Cascade Framework	**0.89**	0.97	**0.04**	**96.0**

Recall falls from 0.980 to 0.970, mAP@0.5 holds at 0.89, and the negative-frame false-alarm rate halves from 0.08 to 0.04. The unchanged mAP is the central result. A gate that discards frames cannot change how the detector behaves on the frames it keeps, so the only accuracy the cascade can lose is whatever sits in the frames the gate wrongly discarded – one percentage point of recall, on this evidence. The halved false-alarm rate is the same mechanism read from the other side: frames the detector would have fired on spuriously never reach it.

The two external baselines place this in context. EfficientDet-D0 is both less accurate (0.82 mAP@0.5) and slower (24.6 FPS) than any YOLO configuration tested, and YOLOv8 sits between it and YOLOv11. The cascade's advantage over all three is therefore not a matter of using a stronger detector – it uses the same YOLOv11 as the single-stage row – but of not running that detector on frames that contain nothing.

More importantly, the cascade framework increased processing speed from 30.2 FPS to 96.0 FPS, a more than threefold increase in throughput, which confirms that filtering defect-free frames prior to detection substantially reduces computational redundancy without materially affecting accuracy.

Two rows of this table carry names that reappear in Table 8.20 with different values, and the two sets of figures are not measurements of the same quantity. This table comes from the cascade study [233] and Table 8.20 from the dual-stream study [243]. The two

campaigns were run separately on the same video sequence dataset, each under its own protocol and its own detector operating point, and their recall columns differ in definition as well as in setting. Recall here is the end-to-end instance-level $\text{Recall}_{\text{cascade}}$ prescribed for whole-pipeline evaluation in Section 6.5., computed over the entire input stream with every instance inside a discarded frame counted as a miss, which is why YOLOv11 reads 0.980; Table 8.20 reports each detector's own recall at its confidence threshold, where the same architecture reads 0.863. Throughput separates for the same reason: single-stage YOLOv11 measures 30.2 FPS under the comparison protocol used here, 41 FPS under the dual-stream study's protocol, and 91 FPS under the FP32 ablation protocol of Section 6.6.1. – three inference configurations of one architecture on the one GPU platform of Table 3.7. No comparison is therefore drawn between the two tables anywhere in this dissertation. Each is read internally, and the cascade's one-percentage-point recall cost and 3.18× speedup are both comparisons against the single-stage row measured alongside it in this table.

Figure 8.8 shows representative test frames processed by the standalone single-stage detector and by the complete cascade. On the frames shown, which are frames Stage 1 correctly forwards, the two configurations produce the same bounding boxes, class labels, and confidence scores. The accuracy difference recorded in Table 8.17 therefore arises from the small number of frames on which the gate and the single-stage detector disagree, and not from any systematic difference in how the two configurations handle frames they both process.

8.4.4. Selection of the Gate Threshold

The operating point of the gate is the single parameter that governs the accuracy–throughput balance of the whole system, and Section 6.7. argued that it should be selected by fixing a recall floor. Table 8.18 gives the sweep on which that choice was made.

End-to-end recall falls monotonically as the threshold rises, from 0.982 to 0.945, while throughput climbs from 263 to 370 FPS. The relationship is not linear in the useful direction: moving from 0.35 to 0.55 buys 20% more throughput at the cost of 2.5 percentage points of recall, whereas moving from 0.35 down to 0.25 buys 1.2 points of recall for a 15% throughput loss. Since a defect discarded at Stage 1 cannot be recovered downstream

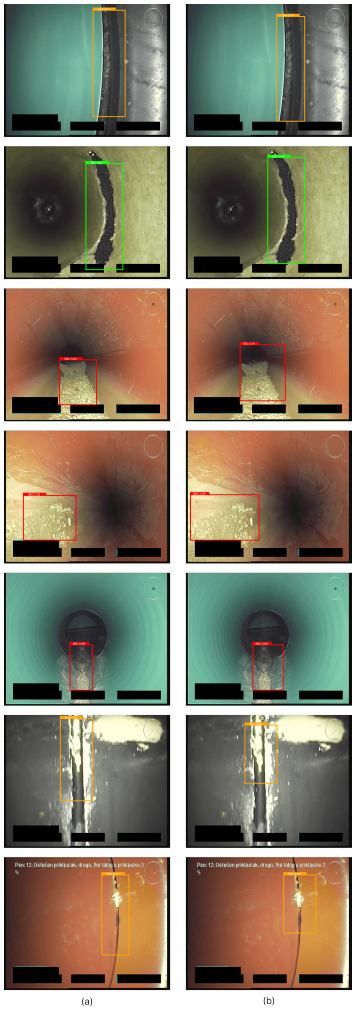


Figure 8.8: Representative defect detections on test frames: (a) single-stage detector, (b) complete cascade [233].

Table 8.18: Cascade ablations under the FP32 end-to-end protocol of Section 6.6.1. [233]: (a) the gate-threshold sweep from which $\tau_{\text{gate}} = 0.35$ was selected; (b) the optional Stage-2 refinements, measured against a fixed static Stage-2 confidence of 0.25; (c) reduced numerical precision applied to that same reference configuration. Stage-1 recall and Stage-1 FPR characterise the gate alone and are frame-level; $\text{Recall}_{\text{cascade}}$ is the end-to-end instance-level figure of Equation 6.10, so the former bounds the latter without being a strict multiple of it. Blocks (b) and (c) hold the gate fixed at the adopted threshold, hence the repeated Stage-1 values. Absolute rates are not comparable with the 30.2 and 96.0 FPS of Table 8.17, which used the comparison protocol. Bold marks the best value in each column within each block; the adopted gate threshold is labelled, not bolded.

Configuration	**mAP@0.5**	**Stage-1 recall**	**Stage-1 FPR**	**$\text{Recall}_{\text{cascade}}$**	**Pass-through**	**E2E FPS**
(a) Gate threshold τ_{gate}						
0.25	–	**0.991**	0.14	**0.982**	20%	263
0.35 (adopted)	–	0.983	0.08	0.970	15%	308
0.45	–	0.973	0.05	0.958	12%	346
0.55	–	0.962	**0.04**	0.945	**10%**	**370**
(b) Optional Stage-2 refinements, $\tau_{\text{gate}} = 0.35$						
Static confidence = 0.25	0.889	0.983	0.08	0.957	15%	308
Dynamic confidence via p_{defect}	**0.900**	0.983	0.08	0.971	15%	291
Score fusion $s' = s \cdot p_{\text{defect}}$	0.898	0.983	0.08	0.973	15%	291
Tile-on-demand (2×2)	0.889	0.983	0.08	**0.982**	15%	228
(c) Inference precision						
FP32	0.889	0.983	0.08	0.965	15%	308
FP16	0.888	0.983	0.08	0.963	15%	448
INT8	0.884	0.983	0.08	0.961	15%	**606**

while wasted computation only costs time, $\tau_{\text{gate}} = 0.35$ was adopted as the point at which end-to-end recall remains at 0.97 with the pass-through rate down to 15%.

8.4.5. Dynamic Confidence, Score Fusion, and Tiling

The three optional refinements of Algorithm 1 all exploit the gate probability p_{defect} that the system has already computed. Their measured contributions are given in Table 8.18.

Dynamic confidence and score fusion behave almost identically, each recovering 1.4–1.6 percentage points of recall for a 6% throughput cost, which is a favourable trade given that both reuse a quantity already computed. Dynamic confidence additionally lifts mAP@0.5 by 1.1 points, since lowering the detection threshold on frames with a strong defect prior admits correct detections that would otherwise fall below it.

Tiling on demand is the outlier in both directions. It produces the largest recall gain in the table, 2.5 percentage points to 0.982, and the largest slowdown, from 308 to 228 FPS, because it multiplies detector invocations on precisely the frames most likely to need them. Notably it does not improve mAP@0.5, which stays at 0.889: tiling recovers small defects the detector was missing entirely, which lifts recall while leaving the localisation of ones it already found where it was. For a deployment where a missed defect is the expensive failure, that is the more valuable of the two effects even though the aggregate accuracy metric does not register it.

8.4.6. Effect of Inference Precision (Quantization)

Beyond the cascade architecture itself, both stages were additionally evaluated under reduced numerical precision to assess suitability for embedded and resource-constrained deployment [233]. Table 8.18 reports FP32, FP16, and INT8 configurations of the same MobileNetV3-L + CB-Focal cascade, quantized using OpenVINO, on the same FP32-benchmark protocol as Table 8.15 above (hence the 308 FPS FP32 baseline, consistent with that table but not with the 96.0 FPS figure in Table 8.17).

Quantization to FP16 raises throughput by roughly 45% relative to FP32 (308 to 448 FPS) at a mAP cost of only 0.001 and a recall cost of 0.002; INT8 pushes throughput to 606 FPS, very nearly double the FP32 rate, at a cumulative cost of 0.005 mAP and 0.004

recall. Neither reduction in accuracy is large relative to the throughput gained, which is the practical case for deploying the quantized cascade on the embedded or resource-constrained hardware carried by inspection robots and ROVs (Section 6.3.2.) instead of the workstation-class GPU used for the rest of this dissertation's experiments. Both stages were exported to ONNX and optimized for the TensorRT and OpenVINO inference backends that support this class of edge hardware [233], and the INT8 throughput gain compounds with the cascade's own gain over single-stage detection — 3.39× under this FP32 ablation protocol (91 to 308 FPS), and 3.18× under the comparison protocol of Table 8.17, the two protocols agreeing on the ratio while differing in absolute rate (Section 6.6.1.) — since one saving reduces how many frames reach the detector and the other reduces the cost of each frame that does. Together they substantially widen the margin by which real-time inspection is achievable on hardware with a fraction of the RTX A6000's compute budget used to obtain these measurements.

8.4.7. Computational Efficiency

At the adopted operating point $\tau_{\text{gate}} = 0.35$ the gate forwarded 3,008 of 20,000 frames of continuous survey footage to the object detection stage, a measured pass-through rate of 15.0% (Table 6.3), so the expensive detector was applied to only a small subset of the acquired data. Substituting that pass-through rate into the complexity model of Chapter 6. (Equation 6.11) reproduces the measured throughput, as set out in the worked example of Section 6.6.1.. On that basis the gate filtered out 85% of frames before they reached the detector, giving the 30.2 to 96.0 FPS improvement of Table 8.17: a speedup factor of 3.18×, equivalent to a 68.5% reduction in average per-frame processing cost.

That pass-through rate belongs to the footage rather than to the cascade, and Section 6.6.2. sets out the dependence and measures it at both ends of its range. The survey stream over which the timing benchmark was run is 92.0% defect-free; the annotated extraction on which the accuracy figures were computed is 46.6% defect-free, having been assembled for training, and the same gate and detector reach 129 FPS on it against 307 FPS on survey footage. Both figures are correct for their own input; neither is a property of the architecture on its own, and Table 6.2 gives the range between them.

The 0.970 recall reported for that configuration is an *end-to-end* figure for the complete

two-stage system, and not the gate's own retention. The two are measured over different denominators. The gate's 0.983 (Table 8.18) is the fraction of defect-bearing *frames* it forwards, whereas 0.970 is the fraction of annotated defect *instances* the complete system recovers, counted over the entire stream so that instances inside discarded frames are misses. Gate retention therefore bounds the end-to-end figure without fixing it exactly, and the margin between 0.983 and 0.970 is what Stage 2 loses on the frames it does receive.

These savings matter most in the deployment scenarios the framework targets: long inspection videos and resource-constrained hardware, where reducing the volume of data reaching the detector is the only thing that makes continuous analysis tractable.

8.5. Evaluation of Spatiotemporal Detection Approaches

This section evaluates the effectiveness of spatiotemporal object detection for automated pipeline inspection. Unlike conventional frame-based detectors that analyze images independently, the investigated approaches exploit temporal information available in video sequences. The objective is to determine whether temporal context improves detection accuracy, increases robustness to challenging imaging conditions, and enhances the recognition of rare or visually ambiguous defects.

8.5.1. Frame-Based Versus Video-Based Detection

To quantify the benefit of temporal information, the performance of conventional frame-based object detection was compared with video-based detection approaches on the 22,427-frame, 10-class video sequence dataset (Chapter 3., Section 3.6.) extracted from 56 CCTV-captured sewer inspection videos, split approximately 80:10:10 at the video level and labelled under EN 13508-2:2003. The frame-based detector processes each image independently, whereas video-based approaches exploit temporal relationships between consecutive observations.

Both video-based streams improve on the conventional frame-based YOLOv7 baseline, consistent with the mechanism described in Chapter 7.: aggregating information from

neighbouring frames helps most for small defects, partially occluded objects, and frames affected by motion blur or illumination changes, where a single frame alone provides insufficient evidence. YOLOv7V's temporal aggregation raises mAP from 0.901 to 0.918 while adding only a small precision gain (0.923 to 0.945); PTSEFormer's transformer-based reasoning raises mAP further, to 0.935, and produces the larger of the two recall gains (0.858 to 0.902), consistent with its design goal of recovering objects that are only weakly visible in the reference frame.

8.5.2. Comparison of Individual Streams

The two streams behave as their designs predict, and their measured detection performance is given in Table 8.20. YOLOv7V's temporal aggregation improves robustness to temporary visibility loss, motion blur, and partial occlusion, with its strongest results on structurally distinct, high-frequency classes such as cracks (BAB) and deformation (BAA). At 30 FPS it is barely slower than the 30.2 FPS frame-based YOLOv7 baseline (Table 8.22), so that gain costs essentially nothing in throughput.

PTSEFormer produces the strongest single-stream mAP (0.935) and recall (0.902) of any configuration in this chapter, consistent with its design goal of modelling long-range temporal relationships for defects affected by occlusion, degradation, and appearance variability. The cost is substantial: 12 FPS, 72M parameters, and 210 GFLOPs against YOLOv7V's 30 FPS, 45M, and 120 GFLOPs (Table 8.19), with training taking approximately 72 hours against 28. That asymmetry is the trade-off motivating the late-fusion design of Section 8.6..

Table 8.19: Runtime and efficiency comparison of the two streams and their fusion [243]. PTSEFormer is invoked selectively rather than on every frame (Section 7.6.6.), so the fusion throughput is not bounded by its 12 FPS standalone rate. Parameters and model size are the resident totals, since both networks must be held in memory whether or not both execute. FLOPs are reported both ways: 330 G is the cost when both streams run on a frame, and 177 G the per-frame average at the invocation rate $r \approx 0.27$ (Equation 7.17), which is the figure consistent with the measured 18 FPS.

Model	FPS	Params (M)	FLOPs (G)	Size (MB)
YOLOv7V	30	45	120	180
PTSEFormer	12	72	210	290
Fusion (YOLOv7V + PTSEFormer)	18	117	330 / 177‡	470

‡ Both streams executing / per-frame average at $r \approx 0.27$.

The comparison confirms the complementary characteristics anticipated by the architectural design: YOLOv7V offers higher precision (0.945 vs. 0.932) and more than double the throughput (30 vs. 12 FPS) at less than half the parameter count (45M vs. 72M), while PTSEFormer achieves higher recall (0.902 vs. 0.872) and mAP (0.935 vs. 0.918) by exploiting longer-range temporal dependencies at substantially higher computational cost. In wall-clock terms, YOLOv7V required approximately 28 hours to train and 33 ms per frame at inference, against 72 hours and roughly 83 ms per frame for PTSEFormer.

These findings confirm the hypothesis that convolutional temporal aggregation and transformer-based temporal reasoning provide complementary strengths, motivating the late-fusion framework evaluated in Section 8.6..

8.6. Evaluation of the Proposed Dual-Stream Fusion Framework

The final stage of this research investigates the proposed dual-stream fusion framework that combines the complementary strengths of YOLOv7V and PTSEFormer. While YOLOv7V provides efficient temporal feature aggregation and real-time processing capabilities, PTSEFormer contributes advanced transformer-based temporal reasoning and long-range dependency modeling. By integrating both streams through a late-fusion strategy, the proposed framework aims to improve detection accuracy, increase robustness to challenging inspection conditions, and enhance the recognition of rare pipeline defects.

The proposed framework was evaluated on the sewer inspection dataset against each individual detection stream, following the protocol of Section 7.6.8..

8.6.1. Performance of the Dual-Stream Framework

The fused framework reaches 0.957 precision, 0.921 recall, 0.953 mAP@0.5, and an F1-score of 0.939 (Table 8.20). It outperforms both individual streams on every accuracy metric: precision improves from YOLOv7V's 0.945 and PTSEFormer's 0.932 to 0.957, recall from 0.872/0.902 to 0.921, and mAP from 0.918/0.935 to 0.953. This confirms that the framework successfully combines the strengths of convolutional and transformer-based spatiotemporal detection instead of simply averaging them – fusion mAP exceeds

both inputs, not just the weaker one. The dual-stream framework introduces additional computational cost, though not the full cost of running both branches on every frame: PTSEFormer is invoked selectively (Section 7.6.6.), and the fused configuration reaches 18 FPS [243].

The paired t-test specified in Section 7.6.8. confirms the improvement against both streams: against YOLOv7V the mean paired difference is 0.034, $t(9) = 3.12$, $p = 0.012$; against PTSEFormer it is 0.018, $t(9) = 2.48$, $p = 0.035$.

The two comparisons differ in the way the design predicts. The margin over YOLOv7V is roughly twice that over PTSEFormer and is the more securely established of the two, which follows from the streams' respective roles: PTSEFormer already supplies most of the temporal reasoning the fused system benefits from, so fusion adds less on top of it than on top of the convolutional stream. The smaller margin over PTSEFormer nonetheless remains significant at the conventional threshold, which is the result that matters, since PTSEFormer alone is the stronger of the two baselines and the one a reader would otherwise ask why not simply deploy.

8.6.2. Comparison with Individual Detection Streams

Table 8.20 sets the fused framework against each individual stream, and one feature of that comparison needs drawing out, because it is not what a selection rule normally produces.

Equation 7.15 does not average the two streams' confidences. It picks one detection and discards the other. A rule of that kind is usually bounded by its inputs on any single metric: if it always chose YOLOv7V it would score YOLOv7V's precision, if it always chose PTSEFormer it would score PTSEFormer's recall, and any mixture of the two should land between them. What the table shows instead is a fused configuration ahead of the better parent on both axes at once – 0.957 precision against YOLOv7V's 0.945, the higher of the two, and 0.921 recall against PTSEFormer's 0.902, again the higher of the two.

That can only happen if the two streams are wrong about different instances. Where both detect the same defect the rule keeps the more confident box and nothing is gained; the gain comes from the frames where one stream produces a detection the other misses

entirely, and the rule keeps whichever one fired. PTSEFormer contributes most of those recoveries on the visually ambiguous classes, which Table 8.21 shows directly, while YOLOv7V's sharper confidence calibration on the frequent classes keeps the rule from promoting weak transformer detections in their place. Error complementarity is what the fusion exploits.

It also explains why the margin over PTSEFormer is the smaller of the two. PTSEFormer already recovers most of what the convolutional stream misses, so adding it to YOLOv7V closes a wider gap than adding YOLOv7V to it. The two streams therefore learn different but complementary temporal representations, and combining them at the decision level yields a gain that the significance test above shows is not an artifact of a single favourable test split.

8.6.3. Comparison Against Newer Single-Frame YOLO Generations

A natural question is whether the dual-stream framework's advantage simply reflects PTSEFormer and YOLOv7V being newer or more heavily engineered than the frame-based YOLOv7 baseline in Table 8.20, instead of temporal fusion specifically. To rule this out, the fused framework was additionally compared against single-frame YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLO26, all trained and evaluated under identical conditions on the same sewer dataset.

Every single-frame generation tested, including YOLO26, clusters within a narrow 0.901–0.919 mAP@0.5 band. The fused framework's 0.953 exceeds the best of them by 3.4 percentage points, and by more on the stricter mAP@[0.5:.95] metric (0.748 against 0.664). Since the newest single-frame architecture still trails a fusion built on an older one, the improvement is attributable to exploiting temporal structure. The trade-off is throughput: single-frame models reach 38–48 FPS against the fused framework's 18, since PTSEFormer's selective invocation keeps the fusion off the 12 FPS floor its standalone rate would otherwise impose. This is why the cascade and fusion frameworks are presented as complementary – the cascade addresses the volume problem, fusion the per-frame accuracy problem once a frame is worth analysing.

Table 8.20: Frame-based baseline, single-frame YOLO generations, the two spatiotemporal streams, and their fusion, all evaluated on the video sequence dataset [243]. Precision, recall and mAP are each detector's own values at its operating point under this study's protocol, and throughput is measured under the same. The YOLOv8 and YOLOv11 rows are consequently not comparable with the rows of the same name in Table 8.17, which come from the separate cascade campaign and report end-to-end instance-level recall; see the discussion following that table.

Model	Precision	Recall	mAP@0.5	mAP@[0.5:.95]	FPS
YOLOv7 (frame-based)	0.923	0.858	0.901	–	30.2
YOLOv8	0.927	0.856	0.907	0.641	42
YOLOv9	0.921	0.861	0.901	0.632	38
YOLOv10	0.934	0.864	0.912	0.648	**48**
YOLOv11	0.933	0.863	0.908	0.643	41
YOLO26	0.941	0.873	0.919	0.664	39
YOLOv7V	0.945	0.872	0.918	0.645	30
PTSEFormer	0.932	0.902	0.935	0.712	12
Dual-Stream Fusion	**0.957**	**0.921**	**0.953**	**0.748**	18

8.6.4. Detection of Rare and Ambiguous Defects

The framework's central objective was improving recognition of rare and visually ambiguous defect categories, which are the cases where one stream frequently produces stronger evidence than the other and the fusion rule has something to choose between.

Table 8.21: Per-class Average Precision (AP@0.5) of YOLOv7V, PTSEFormer, and the fused framework [243].

Class	YOLOv7V	PTSEFormer	Fusion
Crack (BAB)	0.944	0.936	**0.951**
Deformation (BAA)	0.931	0.928	**0.940**
Intruding Sealing Material (BAI)	0.825	0.891	**0.910**
Minor defects (pooled)	0.841	0.877	**0.892**

The final row pools the remaining low-severity codes, which individually carry too few test instances for a stable per-class AP; it is reported as an aggregate for that reason and is not an EN 13508-2 category in its own right.

The pattern across all four classes supports the framework's design intent: YOLOv7V is competitive with or ahead of PTSEFormer on structurally distinct, high-frequency classes (Crack, Deformation), where the gap to Fusion is small (0.007–0.009 AP). For intruding sealing material and the pooled minor defects, categories that are visually ambiguous and not simply rare, PTSEFormer alone already outperforms YOLOv7V by 0.066

and 0.036 AP respectively, and Fusion extends that lead further (0.910 and 0.892 AP), confirming that transformer-based temporal reasoning is doing the work the hypothesis predicted, specifically on the classes it was intended for.

8.6.5. Ablation: Temporal Alignment and Overlay Masking

To isolate the contribution of individual design choices, a component-wise ablation was conducted, progressively adding temporal alignment (TAM) and overlay masking (removal of on-screen timestamps and orientation indicators, Chapter 3.) to the fusion pipeline.

Table 8.22: Ablation of model components: effect of fusion, temporal alignment (TAM), and overlay masking [243].

Configuration	mAP@0.5	Precision	Recall	FPS
YOLOv7 (image only)	0.901	0.923	0.858	30.2
YOLOv7V + TAM	0.918	0.945	0.872	30
PTSEFormer	0.935	0.932	0.902	12
Fusion w/o TAM	0.942	0.957	0.908	19.2
Fusion w/ TAM	0.952	**0.962**	0.916	18.9
Fusion w/ TAM + overlay masking	**0.953**	0.957	**0.921**	**18**

Two design choices are each shown to contribute independently of fusion itself. First, adding temporal alignment (TAM) to the fusion pipeline improves mAP from 0.942 to 0.952 and recall from 0.908 to 0.916, confirming that explicitly compensating for motion between frames adds value beyond what feature aggregation and decision-level fusion already provide. This is the larger of the two effects and the one that justifies the alignment module's cost.

Second, masking the on-screen overlays raises mAP@0.5 marginally further, from 0.952 to 0.953, and recall from 0.916 to 0.921, at a small precision cost (0.962 to 0.957). The size of this effect should not be overstated: 0.001 mAP is well within the range that ordinary training variance could produce, and the corresponding experiment on the twelve-class Sewer Inspection Dataset (Section 8.2.11.) found masking to be mildly *harmful* on that data. What the two results jointly support is a weaker claim than "masking improves accuracy": overlay masking is approximately accuracy-neutral, and it is retained throughout this dissertation as a guard against a known generalisation risk (Section 8.2.11.), and not because it reliably raises the metric.

8.6.6. Qualitative Detection Examples

Figure 8.9 shows annotated ground-truth frames from the sewer inspection dataset. Three visual conditions visible in these frames motivate the preprocessing and temporal-fusion choices described in this chapter: the on-screen telemetry overlays present before masking (Section 8.2.11.), the variation in illumination between frames, and the small size of most defects relative to the frame [243].

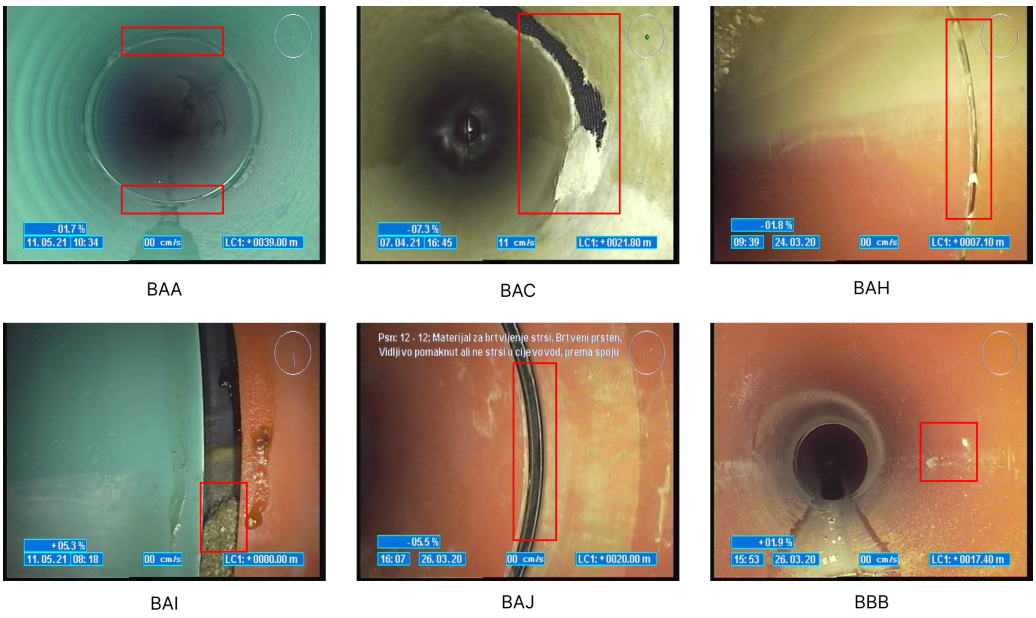


Figure 8.9: Annotated ground-truth examples from the sewer inspection dataset, labelled by EN 13508-2:2003 code: deformation (BAA), break (BAC), defective connection (BAH), intruding sealing material (BAI), pipe misalignment (BAJ), and wall attachments (BBB) [243].

The frames in Figure 8.9 illustrate the conditions under which the two streams differ most. Defects occupying a small fraction of the frame, or partially obscured by water, sediment, or glare, are the cases where single-frame evidence is weakest and where the per-class results in Table 8.21 show the widest gap between the convolutional and transformer streams.

The quantitative evaluation carries the main weight of the argument. Table 8.20 shows the dual-stream framework improving mAP@0.5 from 0.918 and 0.935 for the individual streams to 0.953, a gain significant against both (Section 8.6.). This supports the design rationale developed in Chapter 7.: integrating complementary spatiotemporal representations improves robustness, rare-defect recognition, and localization beyond either

representation alone.

8.7. Comparison Across Inspection Domains

The two domains fail differently, and the difference is instructive. On the underwater dataset the limiting factor is image quality, not class count: with five categories, most of them large and visually distinct, accuracy was high across every architecture, and the variance between models came almost entirely from the leakage-point category, whose small size and low distinctiveness made it hardest regardless of architecture. On the sewer dataset the limiting factor is the number and imbalance of defect classes, not raw image quality, and that is why the loss-function and data-level imbalance mitigations of Chapter 5. produced measurable gains there and would be expected to matter less on the underwater dataset's five-class label set, whose imbalance is confined to two small categories instead of spread across a long tail.

No single failure mode therefore dominates automated pipeline inspection, and a deployment covering both environments would need to combine the image-quality-robust architecture choices validated on the underwater data with the imbalance mitigation validated on the sewer data. Each model here was trained and tested within its own domain; cross-domain transfer without retraining was not attempted and is identified as future work in Chapter 9..

8.8. Validation of Research Hypotheses

8.8.1. Hypothesis H1

H1 states that modified YOLO architectures (v4–v11) can achieve a balanced trade-off between inference speed, detection accuracy, and computational resource consumption. This is supported, but with a qualification. Tables 8.3 and 8.4 show that no single generation dominates both datasets. YOLOv5 performed best on the underwater data, ahead of YOLOv7, YOLOv6, YOLOv8 and the YOLOv4 reference in that order, while YOLOv7 variants performed best on the sewer data. The hypothesis therefore holds in the sense that a well-chosen YOLO variant does balance speed and accuracy. Which

variant achieves that balance, however, depends on the dataset and cannot be assumed in advance.

8.8.2. Hypothesis H2

H2 states that a cascaded approach with binary defect detection accelerates inspection video analysis while maintaining accuracy comparable to single-stage detection. This is strongly supported: Table 8.17 shows a 3.18× throughput increase (30.2 to 96.0 FPS) with recall decreasing by only one percentage point (0.980 to 0.970) relative to a single-stage YOLOv11 baseline. The cascade also halved the negative-frame false-alarm rate, from 0.08 to 0.04, while preserving high recall. This is the most completely evidenced of the four hypotheses.

8.8.3. Hypothesis H3

H3 states that augmenting imbalanced datasets improves detection performance for underrepresented classes and enhances generalization in both inspection environments. All three families of imbalance mitigation technique discussed in Chapter 5. were evaluated experimentally, and all three support H3.

At the loss level, all three adaptations were measured against a common YOLOv7 baseline (Table 8.12) and all three improved recall and mAP@0.5 monotonically with the aggressiveness of their reweighting: Weighted Loss to 0.899/0.914, Class-Balanced Loss to 0.912/0.924, and Focal Loss to 0.931/0.936, against the baseline's 0.880/0.901. Focal Loss additionally improved averaged rare-class recall on BBH and BBE from 0.596 to 0.863, with per-class mAP on those two categories rising from 0.478 to 0.884 and from 0.380 to 0.842 (Table 8.14), which confirms that the aggregate gain is concentrated where H3 predicts it.

At the data level, the per-technique augmentation ablation (Table 8.12) showed every one of five augmentation techniques improving Rare-Class AP over the unaugmented YOLOv7 baseline, from 0.491 to between 0.506 and 0.591, with Copy-Paste producing both the largest rare-class gain (0.591) and the highest overall mAP@0.5 (0.934). Independently, the modified Mosaic study (Table 8.6) showed an augmentation change alone worth 1.4 percentage points of mAP@0.5 and 3.2 of recall on the same twelve-class dataset.

At the sampling level, oversampling, undersampling, hybrid, and class-aware sampling all improved rare-class recall over the same baseline, from 0.596 to between 0.652 and 0.741, with class-aware sampling strongest and hybrid second (Table 8.12).

Consistency across the three families is the strongest part of the case for H3. All twelve configurations tested across the three families improved at least one minority-class metric over the common baseline, though some reduced overall precision, strict-IoU mAP, or overall mAP in doing so: undersampling lowered overall mAP from 0.901 to 0.897, MixUp lowered mAP@0.5:0.95 from 0.631 to 0.627, and every augmentation technique cost precision. The best of each family lands within 1.5 percentage points of mAP@0.5 of the others. It is not one favourable technique but three independent mechanisms, acting at three different stages of the training pipeline, converging on comparable gains.

8.8.4. Hypothesis H4

H4 states that a standardized workflow for data preparation, training, and evaluation improves robustness and reproducibility. This hypothesis is inherently harder to validate numerically, since it concerns methodology. The evidence offered in this dissertation is that adopting EN 13508-2:2003 for the sewer dataset and applying one shared training/evaluation protocol (Chapter 3.) allowed every architecture and mitigation technique in Chapters 4.–7. to be compared on a common basis, which is a precondition for the comparisons underlying H1–H3 to be meaningful at all. H4 is therefore supported as a methodological enabler instead of through a standalone metric.

8.9. Summary of Results

The best baseline configurations were YOLOv5 on the underwater dataset (97.10% mAP) and YOLOv7-ModifiedLoss on the sewer dataset (0.936 mAP, driven by focal loss, not architecture). The cascade framework moved the operating point from 30.2 FPS at 0.980 recall to 96.0 FPS at 0.970. The dual-stream framework establishes a higher-accuracy operating point, raising mAP@0.5 to 0.953 and exceeding every single-frame YOLO generation tested including YOLO26, at 18 FPS.

Four findings recur. First, architecture ranking does not transfer between the two

domains, so it must be validated per deployment domain. Second, class imbalance mitigation works from more than one direction and no family dominates: augmentation, sampling, and loss adaptation each close most of the minority-class gap independently, and whether they compound remains open. Third, the cascade reduces the computational cost of continuous inspection by roughly 68% for one percentage point of recall, which is the clearest actionable finding here for real-time deployment. Fourth, temporal information improves accuracy beyond what a newer detector alone provides, at 18 FPS, since PTSEFormer's selective invocation under the routing criterion of Equation 7.12 keeps the fused framework off the 12 FPS floor its standalone rate would otherwise impose (Section 7.6.6.).

These results support the three scientific contributions claimed in Chapter 1., which Chapter 9. states in full. The annotated datasets enabled every comparison reported here, and the dataset-level experiments establish how that annotation should be prepared and evaluated: the overlay-masking and contextual-background comparisons, the class-filtering result of Table 8.13, which shows that an mAP gain obtained by dropping rare classes from the evaluation is not progress, and the negative-sampling result in the same table. The cascaded configuration is supported by the 3.18× speedup with recall preserved to within one percentage point, and the broader system architecture by the stage-by-stage progression from baseline comparison through imbalance mitigation and cascading to spatiotemporal fusion.

Two items remain open: the cross-dataset generalisation study is limited by the number of independent inspection campaigns available, and neither the cascade nor the fusion framework has been validated on embedded hardware. Both are addressed in Chapter 9..

9. Chapter

CONCLUSION

9.1. Summary of the Research

This dissertation investigated the automation of pipeline defect detection in two visually and operationally distinct environments, submerged pipelines surveyed by ROV and sewer networks surveyed by CCTV, using deep-learning-based object detection. The work was motivated by three problems identified in Chapter 1. that prevent a direct transfer of general-purpose object detectors to this domain: severe class imbalance between common and operationally critical defects, environmental degradation specific to underwater and sewer imagery, and the treatment of inherently sequential inspection video as a stream of independent frames.

Custom data was built across two inspection domains to support the investigation: an underwater pipeline dataset of ROV-acquired imagery, and sewer inspection material annotated according to the EN 13508-2:2003 defect-coding standard, the latter used at two extractions so that three experimental dataset configurations result in total (Chapter 3.). These, together with a shared training and evaluation pipeline, were used across the baseline architecture comparison (Chapter 4.), the class imbalance mitigation study (Chapter 5.), the cascade detection framework (Chapter 6.), and the dual-stream spatiotemporal framework (Chapter 7.).

9.2. Validation of the Research Hypotheses

Section 8.8. of Chapter 8. assesses each hypothesis against the full experimental evidence. This section states the outcomes and what follows from them.

H1 – Modified YOLO architectures can balance speed, accuracy, and resource consumption. Supported, with the qualification that newer architectures did not uniformly outperform older ones. On the underwater dataset there is no monotonic improvement with release order; on the sewer dataset the later generations do improve monotonically from YOLOv8 to YOLOv11, but YOLO26 reverses the trend and a purely data-level change to YOLOv7 lifts it above the unmodified YOLOv8 at no inference cost. The sewer task has twelve classes and severe imbalance where the underwater task has five well-separated ones and is close to saturated for every architecture, which explains why generation matters on one dataset and not the other. Architecture selection should therefore be driven by measured performance on the target dataset, and how a detector is trained can matter more than which generation it belongs to.

H2 – A cascaded binary-then-multiclass approach accelerates inspection while preserving accuracy. This hypothesis received the strongest support of the four. The gate forwarded only 15% of frames while retaining 98.3% of defect-bearing frames at its adopted operating point, raising throughput from 30.2 to 96.0 FPS against a single-stage YOLOv11 baseline with mAP@0.5 unchanged and end-to-end recall falling by one percentage point, from 0.980 to 0.970. INT8 quantization compounds this with a further near-doubling of throughput at a cost of 0.005 mAP.

H3 – Augmenting imbalanced datasets improves detection of underrepresented classes. Supported: all twelve configurations tested across the three mitigation families improved at least one minority-class metric, although several did so at a cost to overall precision or strict-IoU mAP. The loss-function results additionally support a sharper claim than H3 itself makes. Measured against one baseline, the three adaptations fall in strict order on every metric, gaining recall and mAP while losing precision at a nearly constant exchange rate of roughly one point of precision per point of mAP. Three formulations that differ substantially in construction – weighting by raw count, by effective count, and by per-example difficulty – converge on the same rate. The implication is that the achievable trade-off on this data is a property of the data itself, and that the

choice of loss determines only where along that trade-off a system operates. For pipeline inspection, where a missed structural defect is the expensive failure, that choice should be made at the high-recall end.

H4 – A standardised workflow improves robustness and reproducibility. Adopting EN 13508-2:2003 as the annotation standard for the sewer dataset, and applying an identical training and evaluation protocol across both datasets, allowed every architecture and mitigation technique in this dissertation to be compared on a common basis. This is a qualitative validation, but it is the precondition on which the comparisons underlying H1–H3 rest.

Beyond the four numbered hypotheses, the dual-stream spatiotemporal framework of Chapter 7. produced the clearest single accuracy result of the dissertation. Fusing YOLOv7V and PTSEFormer raised mAP@0.5 from 0.918 and 0.935 for the individual streams to 0.953, exceeding every single-frame YOLO generation tested up to and including YOLO26. A two-sided paired t-test on per-video mAP@0.5 values from the ten held-out inspection videos ($n = 10$, df $= 9$) confirmed significance against both streams ($p = 0.012$ and $p = 0.035$). This supports the premise stated in Chapter 1. that treating inspection video as a temporal signal yields accuracy gains that architecture upgrades alone do not.

9.3. Scientific Contributions

In line with the contributions stated in Chapter 1., this research produced three outcomes.

The first is an annotated dataset for training models for the automatic inspection of pipeline infrastructure in sewer and underwater environments, built under the EN 13508-2:2003 defect-coding scheme for the sewer material (Chapter 3.).

The second is a cascaded deep learning model configuration for pipeline image analysis that reduces the computational cost of real-time inspection while keeping recall within roughly one percentage point of a full single-stage detector.

The third is a computer vision system architecture for the automatic detection of pipeline infrastructure components and defects, spanning baseline detection, imbalance mitigation, cascading, and spatiotemporal fusion across two independent inspection do-

mains, with the spatiotemporal fusion component measurably outperforming every single-frame baseline evaluated.

9.4. Limitations

Two limitations of the present work should be acknowledged.

First, the cascade's accuracy and throughput results are obtained on differently composed streams. The accuracy figures come from the annotated 22,427-frame extraction, of which 11,967 frames carry a box; the throughput figures come from 20,000 frames of continuous survey footage, of which 1,600 do. Both streams are logged and their compositions measured rather than inferred, and Equation 6.15 is validated at each end of the interval between them (Section 6.6.2., Table 6.3), so throughput is reported across that range and no claim rests on an undescribed benchmark. What remains is that the two sets of figures are not measurements of one body of footage, and that a single quantity is still solved for rather than timed: the Stage-1 latency under the comparison protocol, $T_1 \approx 5.45$ ms, is obtained from Equation 6.11 instead of measured directly, the corresponding figure under the FP32 protocol being the one that was timed.

Second, several results reported in Chapter 8. derive from manuscripts that were submitted or in preparation at the time of writing. Where a source manuscript is internally inconsistent, the discrepancy is flagged in Chapter 8..

9.5. Future Work

The limitations above motivate several directions for further research, presented here in roughly descending order of priority.

The most immediate is to characterise the routing behaviour of the dual-stream framework across a wider range of operating points, under a controlled latency benchmark that specifies the hardware configuration, the batch size, the input resolution, the numerical precision, and whether preprocessing is included in the measurement. Mapping the invocation rate r of the selectively triggered PTSEFormer branch against the routing threshold τ_{route} that produces it would let the criterion be tuned to a target throughput, turning the operating point into a design parameter instead of a consequence of the threshold chosen.

A second and readily answerable question is whether the strongest technique from each imbalance-mitigation family, namely Copy-Paste or modified Mosaic augmentation, Focal or Class-Balanced Loss, and class-aware sampling, compounds when the techniques are combined. Each closes most of the minority-class gap on its own and all three now sit on a common baseline, so the combination is the obvious remaining question and can be answered directly with the pipeline already developed here.

A related direction follows from the negative SMOTE result of Section 5.2.6.. Pixel-space interpolation produced samples that an inspection professional rejected as hydraulically implausible, because a synthesised defect carries the appearance of a condition without the physical evidence that defines it. Interpolating in a learned feature space instead, or using a generative model conditioned on defect type and pipe geometry, would avoid producing images subject to that objection. Any such work should be validated the same way: by expert assessment of the generated samples before they are trained on, and not by the aggregate metric improvement that additional samples of any quality tend to produce.

Validating the cascade and fusion frameworks on embedded or edge hardware representative of deployed ROVs and CCTV crawlers would confirm whether the throughput gains measured on workstation GPUs transfer to field conditions. The quantization results reported in Chapter 8. suggest this is achievable, but they were measured on the same workstation hardware.

Extending both datasets with additional inspection campaigns, pipeline materials, and camera hardware would test the limits of the models' generalisation. Where dataset expansion is not feasible, domain adaptation or synthetic-to-real transfer offers a lower-cost route to covering new inspection environments.

Such an extension is also the precondition for longitudinal work. Monitoring how a given defect develops between inspections, the progression tracking set aside in Chapter 1., requires repeat passes over the same asset registered to a common spatial reference, so that a detection in one campaign can be matched to the same physical location in the next. Acquiring even a small number of repeat-inspection sequences would make severity trending possible, which is ultimately what determines maintenance scheduling and is the form in which inspection results are most useful to an operator.

For the dual-stream framework specifically, replacing the fixed IoU-threshold fusion rule with adaptive, learnable weighting would allow the framework to adjust its operating

point to footage whose class distribution differs from the training data. Motion compensation tailored to ROV camera behaviour, and cross-dataset validation on independently collected sewer footage, are natural companions to that work.

The architecture study in Chapter 8. also invites a follow-up: the removal and addition experiments reported there quantified accuracy but not cost, since parameter counts, FLOPs, and inference throughput were not recorded for the fourteen variants. Establishing that missing half would turn the study into a full accuracy–efficiency characterisation and allow a deployment configuration to be selected on evidence instead of on structural reasoning.

The same measurement gap applies to the attention and box-regression studies above: repeating those configurations across several random seeds would separate the orderings within each family from seed variance instead of merely reporting them. This is inexpensive, since both studies are single-configuration changes on a pipeline already built.

The finding those two studies share is the most transferable negative result in this dissertation. Against the same baseline and dataset, attention modules moved mAP@0.5 by at most 0.5 percentage points and box-regression losses by at most 0.3, while augmentation moved it by 3.3 and the classification loss by 3.5. Each intervention improved on its own reference configuration and the absolute gains are not directly ranked across differing tasks, but the qualitative asymmetry holds: the interventions that mattered changed what the network was shown or asked to optimise, while those changing how it weights and regresses already-extracted features did not. For a domain in which defects are textural and scene geometry is nearly constant, that asymmetry is worth carrying into future designs instead of rediscovering.

Given the pace of change in the YOLO family observed during this research, spanning YOLOv4 through YOLOv11 with YOLO26 appearing as the work concluded, periodically re-benchmarking new architectures against the datasets and frameworks developed here would help keep the accuracy–speed trade-off current.

9.6. Closing Remarks

Taken together, the results of this dissertation indicate that automated, deep-learning-based pipeline inspection is practical for both underwater and sewer environments, pro-

vided that the class imbalance inherent to defect data and the sequential nature of inspection video are treated as first-class design constraints. The cascaded detection framework in particular demonstrates that a substantial share of the computational cost of continuous video inspection can be avoided without a meaningful loss of recall, which is the property most directly relevant to deploying such a system alongside a human inspector.

BIBLIOGRAPHY

[1] Z. Liu and Y. Kleiner, "State of the art review of inspection technologies for condition assessment of water pipes," *Measurement*, vol. 46, no. 1, pp. 1–15, 2013.

[2] R. Wirahadikusumah, D. M. Abraham, T. Iseley, and R. K. Prasanth, "Assessment technologies for sewer system rehabilitation," *Automation in Construction*, vol. 7, no. 4, pp. 259–270, 1998.

[3] Y. Kleiner and B. Rajani, "Comprehensive review of structural deterioration of water mains: statistical models," *Urban Water*, vol. 3, no. 3, pp. 131–150, 2001.

[4] S. S. Kumar, D. M. Abraham, M. R. Jahanshahi, T. Iseley, and J. Starr, "Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks," *Automation in Construction*, vol. 91, pp. 273–283, Jul. 2018.

[5] J. K. Chow, Z. Su, J. Wu, Z. Li, P. S. Tan, K.-f. Liu, X. Mao, and Y.-H. Wang, "Artificial intelligence-empowered pipeline forimage-based inspection of concrete structures," *Automation in Construction*, vol. 120, p. 103372, Dec. 2020.

[6] Y. Li, H. Wang, L. M. Dang, H.-K. Song, and H. Moon, "Vision-based defect inspection and condition assessment for sewer pipes: A comprehensive survey," *Sensors*, vol. 22, no. 7, p. 2722, Apr. 2022.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[10] J. Johnson and T. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, 2019.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[12] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 408–417.

[13] Q. Zhou, X. Li, L. He, Y. Yang, G. Cheng, Y. Tong, L. Ma, and D. Tao, "Transvod: End-to-end video object detection with spatial-temporal transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7853–7869, 2022.

[14] H. Wang, J. Tang, X. Liu, S. Guan, R. Xie, and L. Song, "Ptseformer: Progressive temporal-spatial enhanced transformer towards video object detection," in *European conference on computer vision*. Springer, 2022, pp. 732–747.

[15] H. Motiee, E. McBean, and A. Motiei, "Estimating physical unaccounted for water (ufw) in distribution networks using simulation models and gis," *Urban Water Journal*, vol. 4, no. 1, pp. 43–52, Mar. 2007.

[16] D. Vacs Renwick, A. Heinrich, R. Weisman, H. Arvanaghi, and K. Rotert, "Potential public health impacts of deteriorating distribution system infrastructure," *Journal AWWA*, vol. 111, no. 2, pp. 42–53, Feb. 2019.

[17] L. M. Dang, H. Wang, Y. Li, T. N. Nguyen, and H. Moon, "Defecttr: End-to-end defect detection for sewage networks using a transformer," *Construction and Building Materials*, vol. 325, p. 126584, Mar. 2022.

[18] X. Zang, Z.-D. Xu, H. Lu, C. Zhu, and Z. Zhang, "Ultrasonic guided wave techniques

and applications in pipeline defect detection: A review," *International Journal of Pressure Vessels and Piping*, vol. 206, p. 105033, Dec. 2023.

[19] C. Fan, F. Sun, and L. Yang, "Investigation on nondestructive evaluation of pipelines using infrared thermography," in *2005 Joint 30th International Conference on Infrared and Millimeter Waves and 13th International Conference on Terahertz Electronics*, vol. 2, 2005, pp. 339–340 vol. 2.

[20] W. Silva, R. Lopes, U. Zscherpel, D. Meinel, and U. Ewert, "X-ray imaging techniques for inspection of composite pipelines," *Micron*, vol. 145, p. 103033, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096843282100024X

[21] M. RUIZ, L. MUJICA, M. QUINTERO, S. QUINTERO, and J. FLOREZ, "In-line inspection of pipelines by using a smart pig (ition) and multivariate statistical analysis," in *Structural Health Monitoring 2015*, ser. SHM. Destech Publications, 2015.

[22] W. Xu and S. Matzner, "Underwater fish detection using deep learning for water power applications," in *2018 International conference on computational science and computational intelligence (CSCI)*. IEEE, 2018, pp. 313–318.

[23] K. Raza and S. Hong, "Fast and accurate fish detection design with improved yolo-v3 model and transfer learning," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, pp. 7–16, 2020.

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

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.

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

[27] Z. Situ, S. Teng, W. Feng, Q. Zhong, G. Chen, J. Su, and Q. Zhou, "A transfer learning-based yolo network for sewer defect detection in comparison to classic object detection methods," *Developments in the Built Environment*, vol. 15, p. 100191, Oct. 2023.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, vol. 9905. Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://springer.com

[30] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *International Conference on Machine Learning*, 2021, pp. 813–824.

[31] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video swin transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3202–3211.

[32] M. Jacobi and D. Karimanzira, "Underwater pipeline and cable inspection using autonomous underwater vehicles," in *2013 MTS/IEEE OCEANS-Bergen*. IEEE, 2013, pp. 1–6.

[33] M. Narimani, S. Nazem, and M. Loueipour, "Robotics vision-based system for an underwater pipeline and cable tracker," in *Oceans 2009-Europe*. IEEE, 2009, pp. 1–6.

[34] M. Fayyaz, M. H. Saffar, M. Sabokrou, M. Fathy, and R. Klette, "STFCN: spatio-temporal FCN for semantic video segmentation," *CoRR*, vol. abs/1608.05971, 2016. [Online]. Available: http://arxiv.org/abs/1608.05971

[35] Y. Liu, S. A. Moghaddas, S. Shi, Y. Huang, J. Kong, and Y. Bao, "Review on applications of computer vision techniques for pipeline inspection," *Measurement*,

vol. 252, p. 117370, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224125007298

[36] R. Fenner, "Approaches to sewer maintenance: a review," *Urban Water*, vol. 2, no. 4, pp. 343–356, 2000, sewer Systems and Processes. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1462075800000650

[37] J. T. Jung and A. Reiterer, "Improving sewer damage inspection: Development of a deep learning integration concept for a multi-sensor system," *Sensors*, vol. 24, no. 23, p. 7786, 2024.

[38] WRc, *Sewer Condition Classification Manual*. Water Research Centre, 2001.

[39] *Investigation and Assessment of Drain and Sewer Systems Outside Buildings - Part 2: Visual Inspection Coding System*, European Committee for Standardization (CEN) Std., 2003, eN 13508-2:2003.

[40] R. Schettini and S. Corchs, "Underwater image processing: State of the art of restoration and image enhancement methods," *EURASIP Journal on Advances in Signal Processing*, 2010.

[41] H. Lu, Y. Li, Y. Zhang, M. Chen, S. Serikawa, and H. Kim, "Underwater optical image processing: a comprehensive review," *Mobile networks and applications*, vol. 22, no. 6, pp. 1204–1211, 2017.

[42] P. M. Uplavikar, Z. Wu, and Z. Wang, "All-in-one underwater image enhancement using domain-adversarial learning." in *CVPR Workshops*, 2019, pp. 1–8.

[43] H. A. Kishawy and H. A. Gabbar, "Review of pipeline integrity management practices," *International Journal of Pressure Vessels and Piping*, vol. 87, no. 7, pp. 373–380, Jul. 2010.

[44] T.-C. Su, M.-D. Yang, T.-C. Wu, and J.-Y. Lin, "Morphological segmentation based on edge detection for sewer pipe defects on cctv images," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13 094–13 114, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417411006439

[45] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[46] X. Luo, Y. Zhao, M. Li, H. Zhou, F. Wang, Y. Du, W. Jiang, J. Zhu, and Y. Zhao, "Underground pipeline inspection and monitoring in mountainous areas: A state-of-the-art review and future perspectives," *Tunnelling and Underground Space Technology*, vol. 164, p. 106783, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0886779825004213

[47] Y. Liu and Y. Bao, "Review on automated condition assessment of pipelines with machine learning," *Advanced Engineering Informatics*, vol. 53, p. 101687, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034622001471

[48] J. C. Cheng and M. Wang, "Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques," *Automation in Construction*, vol. 95, pp. 155–171, Nov. 2018.

[49] M. Wang and J. C. P. Cheng, "Development and improvement of deep learning based automated defect detection for sewer pipe inspection using faster r-cnn," in *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 171–192.

[50] ——, "A unified convolutional neural network integrated with conditional random field for pipe defect segmentation," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 2, pp. 162–177, Jul. 2019.

[51] X. Yin, Y. Chen, A. Bouferguene, H. Zaman, M. Al-Hussein, and L. Kurach, "A deep learning-based framework for an automated defect detection system for sewer pipes," *Automation in Construction*, vol. 109, p. 102967, Jan. 2020.

[52] M. Klusek and T. Szydlo, "Supporting the process of sewer pipes inspection using machine learning on embedded devices," in *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 347–360.

[53] L. Ali, F. Alnajjar, H. A. Jassmi, M. Gocho, W. Khan, and M. A. Serhani, "Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures," *Sensors*, vol. 21, no. 5, p. 1688, Mar. 2021.

[54] K. Chaiyasarn, A. Buatik, H. Mohamad, M. Zhou, S. Kongsilp, and N. Poovarodom, "Integrated pixel-level cnn-fcn crack detection via photogrammetric 3d texture mapping of concrete structures," *Automation in Construction*, vol. 140, p. 104388, Aug. 2022.

[55] T. Lee, Y. Yoon, C. Chun, and S. Ryu, "Cnn-based road-surface crack detection model that responds to brightness changes," *Electronics*, vol. 10, no. 12, p. 1402, Jun. 2021.

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

[57] ——, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

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

[59] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.

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

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

[62] H. Jiang and E. Learned-Miller, "Face detection with the faster r-cnn," in *2017 12th*

IEEE international conference on automatic face & gesture recognition (FG 2017). IEEE, 2017, pp. 650–657.

[63] X. Zhao, W. Li, Y. Zhang, T. A. Gulliver, S. Chang, and Z. Feng, "A faster rcnn-based pedestrian detection system," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2016, pp. 1–5.

[64] F. Lei, H. Zhu, F. Tang, and X. Wang, "Drowning behavior detection in swimming pool based on deep learning," *Signal, Image and Video Processing*, pp. 1–8, 2022.

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

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

[67] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[68] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo algorithm developments," *Procedia computer science*, vol. 199, pp. 1066–1073, 2022.

[69] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.

[70] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, 2023.

[71] M. L. Ali and Z. Zhang, "The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection," *Computers*, vol. 13, no. 12, p. 336, 2024.

[72] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.

[73] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *ECCV*, 2018.

[74] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "Eca-net: Efficient channel attention for deep convolutional neural networks," in *CVPR*, 2020.

[75] Y. Li, H. Wang, L. M. Dang, H.-K. Song, and H. Moon, "Attention-guided multiscale neural network for defect detection in sewer pipelines," *Computer-Aided Civil and Infrastructure Engineering*, vol. 38, no. 15, pp. 2163–2179, 2023.

[76] T. Wang, Y. Li, Y. Zhai, W. Wang, and R. Huang, "A sewer pipeline defect detection method based on improved yolov5," *Processes*, vol. 11, no. 8, p. 2508, 2023.

[77] S. Goharinezhad, A. Hadi, and S. Mirzaei, "Automated defect classification and localization in sewer pipelines using hybrid resnet50–swin transformer and modified yolov8 on cctv inspection images," *Scientific Reports*, 2025.

[78] K. Chen, H. Li, C. Li, X. Zhao, S. Wu, Y. Duan, and J. Wang, "An automatic defect detection system for petrochemical pipeline based on cycle-gan and yolo v5," *Sensors*, vol. 22, no. 20, p. 7907, Oct. 2022.

[79] J. Zhang, X. Liu, X. Zhang, Z. Xi, and S. Wang, "Automatic detection method of sewer pipe defects using deep learning techniques," *Applied Sciences*, vol. 13, no. 7, p. 4589, Apr. 2023.

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

[81] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

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

[83] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *Proceedings of the European Conference on Computer Vision*, pp. 801–818, 2018.

[84] A. Dosovitskiy, L. Beyer, A. Kolesnikov *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[85] Z. Liu, Y. Lin, Y. Cao *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[86] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020, pp. 213–229.

[87] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations*, 2021.

[88] Y. Shi, N. Wang, and X. Guo, "YOLOV: Making still image object detectors great at video object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 2, 2023, pp. 2254–2262.

[89] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[90] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6032.

[91] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019, pp. 9268–9277.

[92] L. R. Marnet, S. Grasshof, Y. Brodskiy, and A. Wasowski, "Marinapipe dataset: Underwater pipeline rgb images for segmentation," https://github.com/remaro-network/MarinaPipe-dataset, 2024, dataset, accessed June 2026.

[93] B. M. Santiago, "Subsea inpainting dataset: Removing overlays from subsea inspection videos," https://www.kaggle.com/datasets/brunomsantiago/subsea-inpainting-dataset, 2021, kaggle dataset, accessed June 2026.

[94] VideoPipe Authors, "Cctv-pipe dataset: Industrial video dataset for urban pipe defect localization," https://videopipe.github.io/cctvpipe/index.html, 2023, videoPipe project, accessed June 2026.

[95] X. Du, Y. Sun, Y. Song, L. Dong, and X. Zhao, "Marine_pulse dataset: Side-scan sonar images for submarine pipeline detection," https://zenodo.org/records/7922705, 2023, zenodo dataset, accessed June 2026.

[96] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017. [Online]. Available: https://arxiv.org/abs/1702.05374

[97] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7310–7311.

[98] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance problems in object detection: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3388–3415, 2020.

[99] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 795–10 816, 2023.

[100] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Springer, 2018.

[101] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[102] H. Zhang, M. Cisse, Y. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[103] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2918–2928.

[104] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho, "Augmentation for small object detection," in *ICCV Workshops*, 2019.

[105] A. Newaz and F. S. Haq, "A novel hybrid sampling framework for imbalanced learning," *arXiv preprint arXiv:2208.09619*, 2022.

[106] C. Vairetti, J. L. Assadi, and S. Maldonado, "Efficient hybrid oversampling and intelligent undersampling for imbalanced big data classification," *Expert Systems with Applications*, vol. 246, p. 123149, 2023.

[107] L. M. Dang, S. Kyeong, Y. Li, H. Wang, T. N. Nguyen, and H. Moon, "Deep learning-based sewer defect classification for highly imbalanced dataset," *Computers & Industrial Engineering*, vol. 161, p. 107630, 2021.

[108] R. Alshawi, M. M. Ferdaus, M. Abdelguerfi, K. Niles, K. Pathak, and S. Sloan, "Imbalance-aware culvert-sewer defect segmentation using an enhanced feature pyramid network," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2025, arXiv:2408.10181.

[109] M. Sung, S.-C. Yu, and Y. Girdhar, "Vision based real-time fish detection using convolutional neural network," in *OCEANS 2017-Aberdeen*. IEEE, 2017, pp. 1–6.

[110] M. S. Asyraf, I. S. Isa, M. I. F. Marzuki, S. N. Sulaiman, and C. C. Hung, "Cnn-based yolov3 comparison for underwater object detection," *Journal of Electrical and Electronic Systems Research (JEESR)*, vol. 18, pp. 30–37, 2021.

[111] M. S. A. B. Rosli, I. S. Isa, M. I. F. Maruzuki, S. N. Sulaiman, and I. Ahmad, "Underwater animal detection using yolov4," in *2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 2021, pp. 158–163.

[112] L. Chen, M. Zheng, S. Duan, W. Luo, and L. Yao, "Underwater target recognition based on improved yolov4 neural network," *Electronics*, vol. 10, no. 14, p. 1634, 2021.

[113] M. Zhang, S. Xu, W. Song, Q. He, and Q. Wei, "Lightweight underwater object detection based on yolo v4 and multi-scale attentional feature fusion," *Remote Sensing*, vol. 13, no. 22, p. 4706, 2021.

[114] M. Tian, X. Li, S. Kong, J. Yu *et al.*, "Pruning-based yolov4 algorithm for underwater gabage detection," in *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021, pp. 4008–4013.

[115] X. Hu, Y. Liu, Z. Zhao, J. Liu, X. Yang, C. Sun, S. Chen, B. Li, and C. Zhou, "Real-time detection of uneaten feed pellets in underwater images for aquaculture using an improved yolo-v4 network," *Computers and Electronics in Agriculture*, vol. 185, p. 106135, 2021.

[116] M. Moniruzzaman, S. M. S. Islam, P. Lavery, and M. Bennamoun, "Faster r-cnn based deep learning for seagrass detection from underwater digital images," in *2019 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2019, pp. 1–7.

[117] L. Chen, Y. Huang, J. Dong, Q. Xu, S. Kwong, H. Lu, H. Lu, and C. Li, "Underwater object detection in the era of artificial intelligence: Current, challenge, and future," *arXiv preprint arXiv:2410.05577*, 2024.

[118] M. T. Andani and F. Ameri, "Automated pipe defect identification in underwater robot imagery with deep learning," *Journal of Marine Science and Application*, vol. 25, pp. 197–215, 2026.

[119] V. F. da Silva, T. A. Netto, and B. A. Ribeiro, "Deep learning approaches for fault detection in subsea oil and gas pipelines: A focus on leak detection using visual data," *Journal of Marine Science and Engineering*, vol. 13, no. 9, p. 1683, 2025.

[120] X. Zhao, X. Wang, and Z. Du, "Research on detection method for the leakage of underwater pipeline by yolov3," in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2020, pp. 637–642.

[121] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[122] J. Yin, J. Shen, X. Gao, D. J. Crandall, and R. Yang, "Graph neural network and spatiotemporal transformer attention for 3D video object detection from point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 8, pp. 9822–9835, 2021.

[123] J. Wang, H. Gang, S. Ancha, Y.-T. Chen, and D. Held, "Semi-supervised 3D object detection via temporal graph neural networks," in *2021 International conference on 3D Vision (3DV)*. IEEE, 2021, pp. 413–422.

[124] C. W. Corsel, M. van Lier, L. Kampmeijer, N. Boehrer, and E. M. Bakker, "Exploiting temporal context for tiny object detection," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 79–89.

[125] A. Mhalla, T. Chateau, and N. E. B. Amara, "Spatio-temporal object detection by deep learning: Video-interlacing to improve multi-object tracking," *Image and Vision Computing*, vol. 88, pp. 120–131, 2019.

[126] J. Xu, Y. Cao, Z. Zhang, and H. Hu, "Spatial-temporal relation networks for multi-object tracking," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3988–3998.

[127] I. Vrsalovic, J. Lerga, and M. Ivasic-Kos, "A system for real-time detection of abandoned luggage," *Sensors*, vol. 25, no. 9, p. 2872, 2025.

[128] N. Lopac, K. Severinski, T. Krljan, and J. Lerga, "Computer vision-based real-time vessel detection and tracking in marinas," in *17th Baška GNSS Conference: Global Navigation Satellite Systems and Green Navigation and Smart Systems*, 2025.

[129] B. Gašparović, J. Lerga, G. Mauša, and M. Ivašić-Kos, "Deep learning approach for objects detection in underwater pipeline images," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2146853, 2022.

[130] R. Palma-Amestoy, P. Guerrero, J. Ruiz-del Solar, and C. Garretón, "Bayesian spatiotemporal context integration sources in robot vision systems," in *Robot Soccer World Cup*. Springer, 2008, pp. 212–224.

[131] R. Palma-Amestoy, J. Ruiz-Del-Solar, J. M. Yánez, and P. Guerrero, "Spatiotemporal context integration in robot vision," *International Journal of Humanoid Robotics*, vol. 7, no. 03, pp. 357–377, 2010.

[132] R. Gayathri, V. Uma, R. Rajakumar, R. V. Priya, and R. Vedhapriyavadhana, "Spatio-Temporal and Semantic Enhanced Context-Aware Navigation for Indoor Robots," *IEEE Access*, 2025.

[133] M. C. van Leeuwen, E. P. Fokkinga, W. Huizinga, J. Baan, and F. G. Heslinga, "Toward Versatile Small Object Detection with Temporal-YOLOv8," *Sensors*, vol. 24, no. 22, p. 7387, 2024.

[134] N. Alzahrani, O. Bchir, and M. M. B. Ismail, "YOLO-Act: Unified Spatiotemporal Detection of Human Actions Across Multi-Frame Sequences," *Sensors*, vol. 25, no. 10, p. 3013, 2025.

[135] H. Luo, L. Huang, H. Shen, Y. Li, C. Huang, and X. Wang, "Object detection in video with spatial-temporal context aggregation," *arXiv preprint arXiv:1907.04988*, 2019.

[136] V. Reilly, H. Idrees, and M. Shah, "Detection and tracking of large number of targets in wide area surveillance," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*. Springer, 2010, pp. 186–199.

[137] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9217–9225.

[138] F. He, N. Gao, Q. Li, S. Du, X. Zhao, and K. Huang, "Temporal context enhanced feature aggregation for video object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 941–10 948.

[139] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 337–10 346.

[140] M. Han, Y. Wang, X. Chang, and Y. Qiao, "Mining inter-video proposal relations for video object detection," in *European conference on computer vision*. Springer, 2020, pp. 431–446.

[141] T. Gong, K. Chen, X. Wang, Q. Chu, F. Zhu, D. Lin, N. Yu, and H. Feng, "Temporal roi align for video object recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, 2021, pp. 1442–1450.

[142] Z. Chang, X. Zhang, S. Wang, S. Ma, and W. Gao, "Stam: A spatiotemporal attention based memory for video prediction," *IEEE Transactions on Multimedia*, vol. 25, pp. 2354–2367, 2023.

[143] W. Han, P. Khorrami, T. Paine, and T. Huang, "Seq-nms for video object detection," in *arXiv preprint arXiv:1602.08465*, 2016.

[144] X. Zhu, J. Dai, L. Yuan, and Y. Wei, "Towards high performance video object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7210–7218.

[145] Z. Jiang, Y. Liu, C. Yang, J. Liu, P. Gao, Q. Zhang, S. Xiang, and C. Pan, "Learning where to focus for efficient video object detection," in *European conference on computer vision*. Springer, 2020, pp. 18–34.

[146] S. Wang, Y. Zhou, J. Yan, and Z. Deng, "Fully motion-aware network for video

object detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 542–557.

[147] L. He, Q. Zhou, X. Li, L. Niu, G. Cheng, X. Li, W. Liu, Y. Tong, L. Ma, and L. Zhang, "End-to-end video object detection with spatial-temporal transformers," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1507–1516.

[148] Q. Qi, X. Wang, T. Hou, Y. Yan, and H. Wang, "Fastvod-net: A real-time and high-accuracy video object detector," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 926–20 942, 2022.

[149] Q. Qi, T. Hou, Y. Yan, Y. Lu, and H. Wang, "Tcnet: A novel triple-cooperative network for video object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 8, pp. 3649–3662, 2023.

[150] T. Hou, Q. Qi, Y. Lu, K. Du, and H. Wang, "Dual selection network for video object detection," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 1–6.

[151] Y. Lyu, M. Y. Yang, G. Vosselman, and G.-S. Xia, "Plug & play convolutional regression tracker for video object detection," *arXiv preprint arXiv:2003.00981*, 2020.

[152] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial intelligence*, vol. 293, p. 103448, 2021.

[153] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[154] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[155] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 029–13 038.

[156] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.

[157] Y. Tan, R. Cai, J. Li, P. Chen, and M. Wang, "Automatic detection of sewer defects based on improved you only look once algorithm," *Automation in Construction*, vol. 131, p. 103912, Nov. 2021.

[158] X. Zhao, N. Xiao, Z. Cai, and S. Xin, "Yolov5-sewer: Lightweight sewer defect detection model," *Applied Sciences*, vol. 14, no. 5, p. 1869, 2024.

[159] D. B. Ha, B. Schalter, L. White, and J. Köhler, "Automatic defect detection in sewer network using deep learning based object detector," *arXiv preprint arXiv:2404.06219*, 2024.

[160] M. OÄ§urlu, B. Bayram, B. Kulavuz, and T. BakÄśrman, "Deep learning for automated sewer defect detection: Benchmarking yolo and rt-detr on the istanbul dataset," *Applied Sciences*, vol. 15, no. 20, 2025. [Online]. Available: https://www.mdpi.com/2076-3417/15/20/11096

[161] J. Lu, W. Song, Y. Zhang, X. Yin, and S. Zhao, "Real-time defect detection in underground sewage pipelines using an improved yolov5 model," *Automation in Construction*, vol. 173, p. 106068, 2025.

[162] R. Sha, Z. Zhang, X. Cui, and Q. Mu, "A lightweight cross-scale feature fusion model based on yolov8 for defect detection in sewer pipeline," *PLOS ONE*, 2025.

[163] S. Moradi, T. Zayed, F. Nasiri, and F. Golkhoo, "Automated anomaly detection and localization in sewer inspection videos using proportional data modeling and deep learning-based text recognition," *Journal of Infrastructure Systems*, vol. 26, no. 3, p. 04020018, 2020.

[164] K. Tian, S. Yang, L. Chen, K. Wang, Y. Ke, Z. Miao, J. Hu, C. Sun, and X. Zhang, "Ddspnet: A two-stage defect detection and severity prediction model for industrial products," *Applied Intelligence*, vol. 55, no. 15, p. 1024, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s10489-025-06924-1

[165] V. Shukla, A. Shukla, S. S. K. Prakash, and S. Shukla, "A systematic survey: Role of deep learning-based image anomaly detection in industrial inspection contexts," *Frontiers in Robotics and AI*, vol. 12, p. 1554196, 2025.

[166] Z. Li, Y. Yan, X. Wang, Y. Ge, and L. Meng, "A survey of deep learning for industrial visual anomaly detection," *Artificial Intelligence Review*, vol. 58, no. 9, p. 279, 2025.

[167] X. Li, X. Li, B. Han, S. Wang, and K. Chen, "Application of efficientnet and yolov5 model in submarine pipeline inspection and a new decision-making system," *Water*, vol. 15, no. 19, p. 3386, Sep. 2023.

[168] J. B. Haurum and T. B. Moeslund, "Sewer-ml: A multi-label sewer defect classification dataset and benchmark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[169] J. Yin, X. Yin, M. Pan, and L. Li, "Scalable and transparent automated sewer defect detection using weakly supervised object localization," *Automation in Construction*, vol. 174, p. 106152, 2025.

[170] R. Taormina and J. A. van der Werf, "Multi-class sewer defect detection with vision-language models," *Cambridge Prisms: Water*, vol. 4, 2026.

[171] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Evaluating yolov5, yolov6, yolov7, and yolov8 in underwater environment: Is there real improvement?" in *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2023, pp. 1–4.

[172] ——, "Pipevision: Underwater pipeline inspection dataset for object detection and automated analysis," 2026, author's manuscript.

[173] Y. Kwon, "Yolo_Label: GUI for marking bounded boxes of objects," Dec. 2019. [Online]. Available: https://github.com/developer0hye/Yolo_Label

[174] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Comparative analysis of YOLOv7 with modified mosaic augmentation against YOLOv8-11 for object detection in unbalanced datasets," in *2025 10th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2025.

[175] B. Gašparović, G. Mauša, M. Ivašić-Kos, and J. Lerga, "Sewage inspection and assessment automatization: Deep learning approaches in line with en 13508-2:2003 standard," 2023, submitted to Wiley.

[176] A. Shatnawi, G. Al-Bdour, R. Al-Qurran, and M. Al-Ayyoub, "A comparative study of open source deep learning frameworks," in *2018 9th international conference on information and communication systems (icics)*. IEEE, 2018, pp. 72–77.

[177] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[178] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[179] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 7464–7475.

[180] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.

[181] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.

[182] Ultralytics, "Ultralytics yolo11 documentation," 2024. [Online]. Available: https://docs.ultralytics.com

[183] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," GitHub repository, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[184] ——, "Yolov5 by ultralytics," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[185] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-

performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[186] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, H. Wei, M. Ke, Q. Zhang, M. Cheng *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[187] C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, and X. Chu, "Yolov6 v3.0: A full-scale reloading," *arXiv preprint arXiv:2301.05586*, 2023.

[188] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Contextual background impact on YOLOv7 object detection in sewage pipeline inspection," in *2025 10th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE, 2025.

[189] X. Song, S. Cao, J. Zhang, and Z. Hou, "Steel surface defect detection algorithm based on yolov8," *Electronics*, vol. 13, no. 5, p. 988, 2024.

[190] C. Chen, H. Lee, and M. Chen, "Steel surface defect detection method based on improved yolov9," *Scientific Reports*, vol. 15, p. 25098, 2025.

[191] P.-H. Chou, C.-C. Wang, and W.-L. Mao, "Yolo-based defect detection for metal sheets," *arXiv preprint arXiv:2509.25659*, 2025.

[192] H. Haoyan, T. Jinwu, W. Haibin, and L. Xinyun, "Ead-yolov10: Lightweight steel surface defect detection algorithm research based on yolov10 improvement," *IEEE Access*, vol. 13, pp. 55 382–55 397, 2025.

[193] L. Zhang, Z. Wang, Y. Ma, and G. Li, "Steel surface defect detection algorithm based on improved yolov10," *Scientific Reports*, vol. 15, no. 1, p. 32827, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-16725-8

[194] L. Zhang and J. Wu, "Steel surface defect detection method based on improved yolov11," in *2025 China Automation Congress (CAC)*, 2025, pp. 3987–3992.

[195] F. Wang *et al.*, "Yolo-lsdi: An enhanced algorithm for steel surface defect identification," *Electronics*, vol. 14, no. 13, p. 2576, 2025.

[196] C. Zhao *et al.*, "An improved yolov11 for defect detection on steel pipe inner walls," in *Proceedings of SPIE*, vol. 14070, 2026.

[197] J. Meng, L. Guo, W. Hao, and D. K. Jain, "A surface defect detection method for electronic products based on improved yolov11," *PLOS ONE*, vol. 20, no. 10, p. e0334333, 2025.

[198] R. Sapkota, R. H. Cheppally, A. Sharda, and M. Karkee, "Yolo26: Key architectural enhancements and performance benchmarking for real-time object detection," *arXiv preprint arXiv:2509.25164*, 2025.

[199] S. Aharon, Louis-Dupont, Ofri Masad, and et al., "Super-gradients," 2021. [Online]. Available: https://zenodo.org/records/7789328

[200] C. Wang, W. He, Y. Nie, J. Guo, C. Liu, K. Han, and Y. Wang, "Gold-yolo: Efficient object detector via gather-and-distribute mechanism," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[201] S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, and B. Lai, "Pp-yoloe: An evolved version of yolo," *arXiv preprint arXiv:2203.16250*, 2022.

[202] X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun, "Damo-yolo: A report on real-time object detection design," *arXiv preprint arXiv:2211.15444*, 2023.

[203] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[204] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2117–2125.

[205] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[206] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *International Conference on Pattern Recognition*, 2006, pp. 850–855.

[207] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Improving object detection with one line of code," *CoRR*, vol. abs/1704.04503, 2017. [Online]. Available: http://arxiv.org/abs/1704.04503

[208] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

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

[210] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[211] A. S. Khan, A. Rahman, and A. J. Moshayedi, "Pipeline surface defect detection using yolov11 with attention mechanisms: A comparative study of sa, lka, and cbam approaches," *Journal of Robotics Research*, vol. 2, no. 2, pp. 31–40, 2025.

[212] D. Ma, N. Wang, H. Fang, W. Chen, B. Li, and K. Zhai, "Attention-optimized 3d segmentation and reconstruction system for sewer pipelines employing multi-view images," *Computer-Aided Civil and Infrastructure Engineering*, 2025.

[213] X. Wang, G. Xue, S. Huang, and Y. Liu, "Underwater object detection algorithm based on adding channel and spatial fusion attention mechanism," *Journal of Marine Science and Engineering*, vol. 11, no. 6, p. 1116, 2023.

[214] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[215] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *CVPR*, 2019.

[216] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," *AAAI*, vol. 34, no. 07, pp. 12 993–13 000, 2020.

[217] Z. Gevorgyan, "Siou loss: More powerful learning for bounding box regression," *arXiv preprint arXiv:2205.12740*, 2022.

[218] H. Zhang, Y. Wang, F. Dayoub, and N. Sunderhauf, "Varifocalnet: An iou-aware dense object detector," in *CVPR*, 2021.

[219] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[220] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," *CVPR*, 2019.

[221] I. G. et al, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020.

[222] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[223] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of Artificial Intelligence Research*, vol. 61, pp. 863–905, 2018.

[224] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[225] W. Li, Y. Peng, M. Zhang, L. Ding, H. Hu, and L. Shen, "Deep model fusion: A survey," *arXiv preprint arXiv:2309.15698*, 2023. [Online]. Available: https://arxiv.org/abs/2309.15698

[226] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[227] G. Wu, X. Zhang, Z. Li, Z. Chen, J. Liang, J. Yang, and X. Li, "Cascade prompt learning for vision-language model adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2024, pp. 294–310. [Online]. Available: https://dl.acm.org/doi/10.1007/978-3-031-72973-7_18

[228] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "PaDiM: A patch distribution modeling framework for anomaly detection and localization," in *Pattern Recogni-*

tion. ICPR International Workshops and Challenges, ser. Lecture Notes in Computer Science. Springer International Publishing, 2021, pp. 475–489.

[229] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 14 298–14 308.

[230] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "CutPaste: Self-supervised learning for anomaly detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[231] J. Yu, Y. Zheng, X. Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "FastFlow: Unsupervised anomaly detection and localization via 2d normalizing flows," *arXiv preprint arXiv:2111.07677*, 2021.

[232] G. Wang, S. Han, E. Ding, and D. Huang, "Student-teacher feature pyramid matching for anomaly detection," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2021.

[233] B. Gašparović, G. Mauša, and J. Lerga, "Cascaded anomaly-detection and yolov11 architecture for real-time sewer inspection," 2026, author's manuscript.

[234] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 6105–6114.

[235] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[236] K. Wu, J. Zhang, H. Peng, M. Liu, B. Xiao, J. Fu, and L. Yuan, "TinyViT: Fast pretraining distillation for small vision transformers," in *Computer Vision – ECCV 2022*, ser. Lecture Notes in Computer Science. Springer Nature Switzerland, 2022.

[237] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3,"

in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1314–1324.

[238] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6479–6488.

[239] T. Ridnik, E. Ben-Baruch, N. Zamir, A. Noy, I. Friedman, M. Protter, and L. Zelnik-Manor, "Asymmetric loss for multi-label classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[240] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024. [Online]. Available: https://arxiv.org/abs/2410.17725

[241] J. Li, "Area under the ROC curve has the most consistent evaluation for binary classification," *PLOS ONE*, vol. 19, no. 12, p. e0316019, 2024.

[242] M. B. A. McDermott, H. Zhang, L. H. Hansen, G. Angelotti, and J. Gallifant, "A closer look at AUROC and AUPRC under class imbalance," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024.

[243] B. Gašparović *et al.*, "A dual-stream network combining yolov7v and ptseformer for enhancing sewer inspection," 2026, submitted to Applied Computing and Informatics.

[244] T. Jiang and Y. Zhong, "Odverse33: Is the new yolo version always better? a multi domain benchmark from yolo v5 to v11," *arXiv preprint arXiv:2502.14314*, 2025.

[245] glenn jocher, "What hyperparams do i need to tune when i want to continue a previous training?" https://github.com/ultralytics/yolov5/issues/9257, Sep. 2022, gitHub issue.

[246] A. Clark and Contributors, "Pillow (pil fork) documentation," https://pillow.readthedocs.io/, 2015, version 10.1.0.

[247] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[248] S. Lapuschkin, S. WÃď ldchen, A. Binder, G. Montavon, W. Samek, and K.-R. MÃijller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Communications*, vol. 10, no. 1, Mar. 2019.

LIST OF FIGURES

1.1 General workflow of conventional CCTV- and ROV-based pipeline inspection. 4
1.2 Representative inspection imagery from the two datasets used in this dissertation: (a) sewer pipeline frames from the CCTV inspection dataset, showing uneven illumination, sediment and flowing water; (b) underwater pipeline frames from the PipeVision dataset, showing turbidity, light attenuation and colour distortion. 5

2.1 Coverage of the five reviewed themes and the two research gaps 26

3.1 A sewer CCTV inspection frame (a) before and (b) after masking the fixed-position on-screen overlays: orientation indicator, timestamp, marching speed, and distance readout [175]. 36
3.2 Class distribution of the video sequence dataset (Table 3.6), log-scaled. . . 38

4.1 General processing pipeline of modern YOLO-based object detectors. . . . 50
4.2 Hierarchical Feature Extraction in CNN Backbones 51
4.3 Overview of channel and spatial attention mechanisms (SE, CBAM, ECA) applied to a convolutional feature map. 55
4.4 Geometry of the IoU-based localization losses: (a) the intersection and union areas from which IoU is computed; (b) the smallest enclosing box C supplying the GIoU penalty; (c) the centre-point distance and aspect-ratio terms added by CIoU; (d) the angle, distance, and shape components of SIoU. 61

5.1 Class distribution of the sewer inspection dataset, illustrating the long-tail imbalance between frequently occurring and rare defect categories. 71

5.2 From rarity to missed detection: a rare defect class contributes few training samples, leading to poor feature learning, low recall, and an increased false-negative rate at inference time. 73

5.3 Augmentation techniques applied to real annotated frames from the sewer inspection dataset. (a) horizontal flip, rotation, and scaling of a BBB (wall attachments) frame, with bounding boxes transformed accordingly; (b) brightness increase, contrast increase, and additive Gaussian noise on the same frame; (c) two frames (BBB and BBC) linearly interpolated at $\lambda = 0.5$; (d) four frames (BBB, BBC, BAA, BAJ) tiled into one composite; (e) a BAK (defective lining) instance extracted from its source frame, rescaled, and inserted into a frame already containing a BAA (deformation) annotation. 75

5.4 Synthetic BBF (water ingress) instances generated by instance-level SMOTE and composited into real inspection frames. These samples were assessed by an experienced inspection professional and judged unsuitable for training; they were not used in any experiment reported in this dissertation. . . 80

5.5 Schematic comparison of the four dataset balancing strategies 83

6.1 Proposed cascade detection framework. 95

6.2 Internal structure of the two cascade stages [233]: (a) the Stage-1 anomaly detection module and its decision flow; (b) the Stage-2 pipeline, from preprocessing through YOLOv11 inference and non-maximum suppression to structured defect output. 97

7.1 Conceptual overview of the YOLOv7V temporal aggregation process. Features extracted from neighboring frames are aligned and aggregated before object detection. 114

7.2 A five-frame temporal window $\{I_{t-2}, \ldots, I_{t+2}\}$ sampled at 2 frames per second from ROV inspection footage [243]. 116

7.3 YOLOv7 architecture used as the backbone of the YOLOv7V stream. . . . 117

7.4 Feature aggregation strategies: (a) conventional single-step aggregation, (b) the progressive aggregation used by PTSEFormer [14]. 119

7.5 PTSEFormer pipeline. 120

7.6 Proposed dual-stream fusion framework combining the YOLOv7V temporal aggregation stream and the PTSEFormer temporal reasoning stream through decision-level late fusion. 123

8.1 Training loss on the underwater pipeline dataset: (a) loss and mAP for the YOLOv4 reference model, (b) loss for YOLOv5, (c) loss for YOLOv7 [129]. 138
8.2 Precision–recall and F1–confidence curves on the underwater pipeline dataset for YOLOv7 and YOLOv5 [129]. 138
8.3 mAP comparison across YOLOv7 variants on the sewer inspection dataset (Table 8.4). YOLOv7-ML denotes the focal-loss-adapted YOLOv7-ModifiedLoss configuration. 141
8.4 Training loss on the sewer inspection dataset: (a) YOLOv7s, (b) YOLOv7-D6s [175]. 142
8.5 Qualitative detections on the sewer inspection dataset [175]: (a) representative detections across four defect categories that differ substantially in appearance – pipe misalignment (BAJ), deposits (BBC), defective lining (BAK) and cracks (BAB); (b) the focal-loss-adapted model against the unmodified baseline on the same frames, where the adapted model recovers instances the baseline misses or scores lower. 143
8.6 Per-class recall on the sewer dataset, baseline cross-entropy vs. focal-loss-adapted YOLOv7 [175]. The largest gains fall on the two rarest classes, BBH and BBE. 164
8.7 Stage-1 ROC curve (AUROC = 0.981) and precision–recall curve (AP $\approx$ 0.967) for the EfficientNet-B0 + CB-Focal configuration [233]. 168
8.8 Representative defect detections on test frames: (a) single-stage detector, (b) complete cascade [233]. 173
8.9 Annotated ground-truth examples from the sewer inspection dataset, labelled by EN 13508-2:2003 code: deformation (BAA), break (BAC), defective connection (BAH), intruding sealing material (BAI), pipe misalignment (BAJ), and wall attachments (BBB) [243]. 184

LIST OF TABLES

2.1 Key challenges in deep learning-based pipeline inspection. 27

3.1 Summary of datasets used in this dissertation. 30
3.2 Distribution of object categories in the Underwater Pipeline Dataset [172]. 31
3.3 Derivation of the sewer material from the raw sampled frames. The final two rows partition the complete 22,427-frame stream and are the counts used by the sequence-based experiments of Chapters 6. and 7.; the 13,000-image row is the image-based dataset used in Chapters 5. and 8.. 34
3.4 Per-class instance counts in the 12-class Sewer Inspection Dataset 35
3.5 The two extractions of the sewer material. Both derive from one annotation pass over the same 56 inspection videos; see Table 3.3 for the derivation from the raw sampled frames. 37
3.6 Per-class instance counts in the video sequence dataset (22,427 frames including non-target background frames, 10 classes). 38
3.7 Hardware and software configuration of the two experimental platforms. . . 41

4.1 Comparison of YOLO architectures considered in this research. 48
4.2 Common preprocessing and augmentation techniques used in object detection 51
4.3 Comparison of attention mechanisms commonly used in object detection architectures. 59
4.4 Comparison of IoU-based localization loss functions. 66

5.1 Comparison of loss function adaptations for imbalanced object detection. . 91

6.1 Dataset composition for the two stages of the cascade framework [233]. . . 96

6.2 Pass-through rate and end-to-end throughput as a function of how empty the input stream is, with the gate held at its measured operating point ($r = 0.983$, $f = 0.08$; Table 8.18) and the FP32 stage latencies of Section 6.6.1. ($T_1 = 1.60$ ms, $T_2 = 11.00$ ms, 640×640). Only the footage changes across the rows. The first and fifth rows are measured directly and are the two streams of Table 6.3; the remaining rows are computed from Equation 6.15. 106

6.3 Gate behaviour measured directly on two streams under the FP32 ablation protocol (640×640 Stage-2 input, $T_1 = 1.60$ ms, $T_2 = 11.00$ ms, $\tau_{\text{gate}} = 0.35$). Counts are frame-level: a frame is positive when it carries at least one annotated bounding box. Both streams are labelled under the same EN 13508-2 annotation effort and differ in composition, not in gate configuration. 107

8.1 Detection performance of evaluated architectures on the underwater pipeline dataset 135

8.2 Inference throughput, training time, and convergence behaviour of the evaluated architectures on the underwater pipeline dataset [129]. Faster R-CNN's training time and convergence iteration were not recorded; the two YOLOv4-Tiny subdivision settings share one FPS figure since subdivision affects training memory, not inference cost. 136

8.3 Performance comparison of YOLO generations on the underwater pipeline dataset [171]. Values are as reported in the source study; the YOLOv4 row is the reference model carried over from Table 8.1. 137

8.4 Performance comparison of YOLOv7 variants on the sewer inspection dataset 140

8.5 Wall-clock training time for the YOLOv7 variants on the twelve-class Sewer Inspection Dataset [175]. 141

8.6 Modified Mosaic augmentation and the YOLOv8–YOLOv11 generations on the twelve-class Sewer Inspection Dataset [174]. 143

8.7 YOLOv7, YOLOv11, and YOLO26 on the twelve-class Sewer Inspection Dataset. Values are averaged over repeated runs. Entries marked † were not measured directly but derived algebraically from the other reported metrics. 144

8.8 Effect of fourteen structural modifications applied individually to the YOLOv7 baseline, twelve-class Sewer Inspection Dataset. The upper block of nine removes capacity (Section 8.2.6.); the lower block of five adds it (Section 8.2.7.), each configuration trained three times from different random seeds ($n = 3$) with means reported. Bold marks the best value in each column across the whole table. 146

8.9 Second-order architectural and localization-loss modifications on the twelve-class Sewer Inspection Dataset, each applied in isolation to the YOLOv7 baseline. CIoU is the YOLOv7 default and therefore coincides with the baseline row. F1 is computed from the reported precision and recall. 149

8.10 Influence of contextual background information on YOLOv7 detection performance, four-class 11,510-image subset [188]. 153

8.11 Effect of on-screen overlay masking on YOLOv7 detection performance, sewer dataset [175]. 153

8.12 Class imbalance mitigation on the twelve-class Sewer Inspection Dataset. All three families share the YOLOv7 baseline of Table 8.4, the same data and the same splits; incomplete metric recording and potentially unequal tuning effort across families nonetheless limit strict cross-family ranking. Rare-class metrics are averaged over the least-represented defect categories and are computed from the per-class breakdown of Table 8.14, whose baseline column is the from-scratch YOLOv7s configuration; the aggregate columns of the baseline row are the COCO-initialised YOLOv7. A dash marks a metric not recorded for that family. 156

8.13 Two dataset-level interventions on the sewer dataset (YOLOv7s) [175]. The upper block removes defect classes falling below an instance-count threshold; the final row adds 13,000 real background images containing no annotated defect to the four-class filtered training set. The sewer annotation effort itself yields 10,460 defect-free frames (Table 3.3), so the background images used here must include material drawn from outside it; the source study does not record their provenance. 160

8.14 Per-class performance of the baseline YOLOv7s and the focal-loss-adapted YOLOv7-ModifiedLoss on the twelve-class Sewer Inspection Dataset [175]. The final row is the macro average over classes. 163

8.15 Stage-1 gate: classification performance of the four backbone candidates on the held-out test set, and the loss-function ablation on the adopted backbone [233]. Gate recall is the fraction of defect-bearing *frames* the gate forwards at its adopted operating point, and is the quantity the gate is selected on; it is distinct from the end-to-end cascade recall of Table 8.17, which counts defect *instances*. Pass-through rate and end-to-end throughput were measured only for the configurations carried into the loss ablation, under the FP32 protocol of Section 6.6.1., and over the 20,000-frame survey stream of Table 6.3 ($e = 0.920$) rather than the annotated extraction. Precision, recall, F1 and AUROC are computed at the default 0.5 decision threshold; gate recall is read at the deployed τ_{gate}, where the corresponding Stage-1 false-positive rate is given per threshold in Table 8.18. The adopted configuration additionally reaches an accuracy of 0.966 on the balanced held-out split. Dashes are unmeasured, not zero. 167

8.16 Stage-2 YOLOv11 per-class detection performance on the video sequence dataset [233]. Instance counts are the dataset totals from Table 3.6. BAH is reported as not applicable: with a single annotated instance in the entire dataset and a video-level split, it contributes no evaluable test instances, so no average precision can be computed for it. The mean is therefore taken over the nine scored categories. 169

8.17 Comparison between conventional single-stage detection, two external baselines, and the proposed cascade framework [233]. NFAR is the negative-frame false-alarm rate defined in Section 6.5.: a frame-level quantity over defect-free frames, not comparable with a detection-level false-positive rate. The Recall column is the end-to-end instance-level $\text{Recall}_{\text{cascade}}$ of Equation 6.10, computed over the entire stream. The YOLOv8 and YOLOv11 rows are therefore not comparable with the rows of the same name in Table 8.20, which were measured in a separate campaign under a different protocol; the paragraph following this table sets out the difference. 171

8.18 Cascade ablations under the FP32 end-to-end protocol of Section 6.6.1. [233]: (a) the gate-threshold sweep from which $\tau_{\text{gate}} = 0.35$ was selected; (b) the optional Stage-2 refinements, measured against a fixed static Stage-2 confidence of 0.25; (c) reduced numerical precision applied to that same reference configuration. Stage-1 recall and Stage-1 FPR characterise the gate alone and are frame-level; $\text{Recall}_{\text{cascade}}$ is the end-to-end instance-level figure of Equation 6.10, so the former bounds the latter without being a strict multiple of it. Blocks (b) and (c) hold the gate fixed at the adopted threshold, hence the repeated Stage-1 values. Absolute rates are not comparable with the 30.2 and 96.0 FPS of Table 8.17, which used the comparison protocol. Bold marks the best value in each column within each block; the adopted gate threshold is labelled, not bolded. 174

8.19 Runtime and efficiency comparison of the two streams and their fusion [243]. PTSEFormer is invoked selectively rather than on every frame (Section 7.6.6.), so the fusion throughput is not bounded by its 12 FPS standalone rate. Parameters and model size are the resident totals, since both networks must be held in memory whether or not both execute. FLOPs are reported both ways: 330 G is the cost when both streams run on a frame, and 177 G the per-frame average at the invocation rate $r \approx 0.27$ (Equation 7.17), which is the figure consistent with the measured 18 FPS. . 178

8.20 Frame-based baseline, single-frame YOLO generations, the two spatiotemporal streams, and their fusion, all evaluated on the video sequence dataset [243]. Precision, recall and mAP are each detector's own values at its operating point under this study's protocol, and throughput is measured under the same. The YOLOv8 and YOLOv11 rows are consequently not comparable with the rows of the same name in Table 8.17, which come from the separate cascade campaign and report end-to-end instance-level recall; see the discussion following that table. 182

8.21 Per-class Average Precision (AP@0.5) of YOLOv7V, PTSEFormer, and the fused framework [243]. 182

8.22 Ablation of model components: effect of fusion, temporal alignment (TAM), and overlay masking [243]. 183

LIST OF ABBREVIATIONS

AdamW	Adam optimizer with decoupled Weight decay
AP	Average Precision
ASL	Asymmetric Loss
AUROC	Area Under the Receiver Operating Characteristic curve
BCE	Binary Cross-Entropy
CB-Focal	Class-Balanced Focal (Loss)
CBAM	Convolutional Block Attention Module
CCTV	Closed-Circuit Television
CE	Cross-Entropy
CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
COCO	Common Objects in Context (dataset)
CPU	Central Processing Unit
CSP	Cross-Stage-Partial
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network library
DeiT	Data-efficient image Transformer
DenseNet	Densely Connected Convolutional Network
DETR	DEtection TRansformer
DFL	Distribution Focal Loss
DIoU	Distance Intersection over Union
E2E	End-to-End
ECA	Efficient Channel Attention
E-ELAN	Extended Efficient Layer Aggregation Network

ELAN	Efficient Layer Aggregation Network
EN 13508-2	European Standard 13508-2 (sewer/drain visual inspection coding)
FLOPs / GFLOPs	Floating-Point Operations / Giga-FLOPs
FN	False Negative
FP	False Positive
FP16 / FP32	16-bit / 32-bit Floating-Point (half / single precision)
FPR	False-Positive Rate
FPS	Frames Per Second
GELAN	Generalized Efficient Layer Aggregation Network
GIoU	Generalized Intersection over Union
GPU	Graphics Processing Unit
HSV	Hue–Saturation–Value (colour space)
INT8	8-bit Integer (quantization)
IoU	Intersection over Union
LiDAR	Light Detection And Ranging
LTS	Long-Term Support (Ubuntu release)
mAP	mean Average Precision
MSE	Mean Squared Error
NAS	Neural Architecture Search
NCL	Neighborhood Cleaning Rule
NDT	Non-Destructive Testing
NFAR	Negative-Frame False-Alarm Rate
NMS	Non-Maximum Suppression
PANet	Path Aggregation Network
PGI	Programmable Gradient Information
PTSEFormer	Progressive Temporal-Spatial Enhanced transFormer
QAM	Query Assembling Module
RAM	Random Access Memory
RC-AP	Rare-Class Average Precision
RC-Rec	Rare-Class Recall
R-CNN	Region-based Convolutional Neural Network

ReLU	Rectified Linear Unit
ResNet	Residual Network
ROC	Receiver Operating Characteristic
ROV	Remotely Operated Vehicle
SE	Squeeze-and-Excitation
SIoU	SCYLLA Intersection over Union
SMOTE	Synthetic Minority Over-sampling Technique
SPP	Spatial Pyramid Pooling
SPPCSPC	Spatial Pyramid Pooling with Cross-Stage-Partial Connections
SSD	Single Shot MultiBox Detector
STAM	Spatial Transition Awareness Module
TAM	Temporal Alignment Module
TFAM	Temporal Feature Aggregation Module
TP	True Positive
WRc	Water Research Centre
YOLO	You Only Look Once
YOLOv7V	YOLOv7 with Video (temporal aggregation) stream

APPENDIX

CURRICULUM VITAE

Boris Gašparović was born on 31 March 1997 in Ogulin, Croatia. He enrolled in undergraduate studies in Computer Science at the Faculty of Engineering, University of Rijeka, and graduated in 2018 with a thesis on methods for fundamental frequency estimation in speech. He continued his education at the same faculty, completing graduate studies in 2020 with a thesis on the extraction of roads from satellite imagery.

He is employed at the Faculty of Engineering, University of Rijeka, where he carries out his research within the Center for Artificial Intelligence and Cybersecurity. His work concerns computer vision and deep learning for the automated visual inspection of infrastructure, and in particular object detection in degraded imaging conditions, the treatment of class imbalance in industrial defect datasets, and the use of temporal information in inspection video. The research presented in this dissertation was conducted in collaboration with Prof. Jonatan Lerga, Prof. Goran Mauša, and Prof. Marina Ivašić-Kos.

His doctoral research was supported by the European Union Horizon 2020 project INNO2MARE (grant agreement no. 101087348), by the University of Rijeka projects uniri-iskusni-tehnic-23-11 and uniri-iskusni-tehnic-23-83, and by the project "Istraživanje novih pristupa procjene stanja podvodnih betonskih struktura korištenjem strojnog učenja". In the course of this work he assembled and annotated two pipeline inspection datasets: one covering underwater pipelines surveyed by remotely operated vehicle, and one covering sewer networks surveyed by CCTV and coded according to the EN 13508-2:2003 standard.

Alongside the work presented here, he has contributed to applied machine learning research in medical imaging and in sports analytics.

LIST OF PUBLICATIONS

[1] ***Publications related to the dissertation***

[2] B. Gašparović, J. Lerga, G. Mauša, and M. Ivašić-Kos, "Deep Learning Approach for Objects Detection in Underwater Pipeline Images," *Applied Artificial Intelligence*, vol. 36, no. 1, art. 2146853, 2022. doi: 10.1080/08839514.2022.2146853.

[3] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Evaluating YOLOv5, YOLOv6, YOLOv7, and YOLOv8 in Underwater Environment: Is There Real Improvement?," in *Proceedings of the 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, Split, Croatia, 2023, pp. 1–4. doi: 10.23919/SpliTech58164.2023.10193505.

[4] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Comparative Analysis of YOLOv7 with Modified Mosaic Augmentation Against YOLOv8-11 for Object Detection in Unbalanced Datasets," in *Proceedings of the 2025 10th International Conference on Smart and Sustainable Technologies (SpliTech)*, 2025.

[5] B. Gašparović, G. Mauša, J. Rukavina, and J. Lerga, "Contextual Background Impact on YOLOv7 Object Detection in Sewage Pipeline Inspection," in *Proceedings of the 2025 10th International Conference on Smart and Sustainable Technologies (SpliTech)*, 2025.

[6] B. Gašparović, G. Mauša, J. Rukavina, I. Otulić, and J. Lerga, "Survey and Best Practices in Annotating Underwater Pipeline Imagery for Deep Learning," in *Proceedings of the International Conference on Smart and Sustainable Technologies (SpliTech)*, 2026.

[7] B. Gašparović, "Inspekcija podvodnih cjevovoda korištenjem umjetne inteligencije," *Infcon*, 2023.

[8] ***Manuscripts in publication***

[9] B. Gašparović, G. Mauša, M. Ivašić-Kos, and J. Lerga, "Sewage Inspection and Assessment Automatization: Deep Learning Approaches in Line with the EN 13508-2:2003 Standard," in publication.

[10] B. Gašparović, G. Mauša, and J. Lerga, "Cascaded Anomaly-Detection and YOLOv11 Architecture for Real-Time Sewer Inspection," in publication.

[11] B. Gašparović, G. Mauša, and J. Lerga, "A Dual-Stream Network Combining YOLOv7V and PTSEFormer for Enhancing Sewer Inspection," in publication.

[12] ***Other publications***

[13] B. Gašparović, K. Lenac, J. Lerga, T. Zibar Belačić, and V. Katić, "The Reproducibility of a Facial Scanner Using Automated Tracing Landmarks," *European Journal of Orthodontics*, 2023.

[14] B. Gašparović, L. Morelato, K. Lenac, G. Mauša, A. Zhurov, and V. Katić, "Comparing Direct Measurements and Three-Dimensional (3D) Scans for Evaluating Facial Soft Tissue," *Sensors*, 2023.

[15] A. Skoki, B. Gašparović, S. Ivić, J. Lerga, and I. Štajduhar, "Building Individual Player Performance Profiles According to Pre-Game Expectations and Goal Difference in Soccer," *Sensors*, 2024.

[16] D. Šimac Pavičić, A. Sviličić, B. Gašparović, L. Šimunović, S. Crnković, et al., "Effect of the Functional Appliances on Skeletal, Dentoalveolar, and Facial Soft Tissue Characteristics," *Applied Sciences*, 2025.