



UiT The Arctic University of Norway

Faculty of Engineering Science and Technology

Enhancing AI Systems through Representative Dataset, Transfer Learning, and Embedded Vision

Ravindra Rajaram Patil

A dissertation for the degree of Philosophiae Doctor

September 2023



Acknowledgement

I express my sincere gratitude and appreciation to my supervisors, Prof. Dr. Mohamad Y. Mustafa, and Prof. Dr. Rajnish Kaur Calay. Their continuous guidance, support and valuable insights have been instrumental in shaping this research work and bringing it to completion.

I also acknowledge the support of ADYPatil School of Engineering, Pune where I accomplished my experimental work and thank Dr. Saniya Ansari for her supervision in the initial stages of my research. I am grateful to SPRING, a Horizon 2020 EU-India Project for providing funding to my PhD studies through Grant No. GOI No. BT/IN/EU-WR/60/SP/2018 and No. 821423. The financial support from PEERS (UTF 2020/10131) for my research-stay at UiT- The Arctic University of Norway, Narvik, Norway is acknowledged.

I also recognise all my teachers from school and college, whose dedication and teachings played a crucial role for my educational growth and research interest.

I express my deepest gratitude to my parents for their unwavering support, hard work, guidance, values that shaped my character and achievements. I am indebted to my younger brother, Shailesh (Appa), for all his steady support and dedication. I extend my thanks to my dear grandmother and respected mother-in-law for their love and blessings. I also acknowledge the friendship of Kunal who always stands by me.

I am indebted to my beloved wife, Gayatri, for her constant support and understanding throughout this journey and my daughter, Reva for bringing immense joy in my life.

Above all, I am forever deeply grateful and humbled by the boundless love, blessings, and guidance I am receiving from my God, Shri Parabramha Sheelnath Ji Shri Gurumaharaj Ji. It is through his divine presence, and grace that I have found strength, direction, and purpose in every aspect of my life.

I dedicate my Ph.D. Thesis to my Shri Gurumaharaj Ji.

Thank you all.

Ravindra Rajaram Patil

Narvik, Norway

September 2023

Abstract

Artificial intelligence (AI) encompasses a range of techniques that enable machines to perceive, learn, and make intelligent decisions and it has emerged as transformative technology in many applications. This thesis presents the development of an AI model, focusing on the significance of the primary representative dataset and the effectiveness of transfer learning and fine-tuning techniques for model development. The research demonstrates the affirmative impact of methodical approaches on the accuracy, efficiency, and robustness of AI systems. Moreover, the application of the detection model is demonstrated in wastewater management i.e., for urban wastewater systems, thus underpinning the application of AI to real world scenarios.

The research approach followed in this work includes critical literature review, site surveys, intensive experimentations, and robust validation processes which allowed to identify and address existing gaps and limitations and helped to develop AI detection models for the selected application.

Deep neural networks, a prominent AI technique, chosen for developing AI model in this work has exceptional capabilities in handling complex tasks by learning from vast amounts of data. But the availability of high-quality and representative datasets to effectively train deep neural network models is critical. The comprehensive and diverse datasets provide effective training examples, reduce biases, and enhance the detection models' ability to handle complex inputs.

In the present case, the representative dataset was not available. Therefore, critical multiclass representative image dataset was generated in the laboratory with unparalleled authenticity using model sewer network and named as Sewer-Blockages Imagery Recognition Dataset (S-BIRD) which served as a benchmark for real-time detection and recognition models. The research also addressed the need for dataset curation, data integrity, and biases.

Using S-BIRD, deep neural object detection models were developed through transfer learning and fine-tuning. Inductive transfer learning technique used for development of models, improved convergence, training times, and performance on target detection tasks, enabling adaptation to different domains with minimal additional training. Transfer learning parameters were optimised for desired outcomes. The effectiveness of the developed model for detecting sewer blockages was evaluated by performance metrics. The model achieved high accuracy rate of 96.30% at an IoU of 0.5 in detecting different blockages validating efficacy of dataset and the applicability of the techniques used for developing the model.

AI detector trained on the S-BIRD dataset was then imported on advanced GPU-based single-board computer that formed an embedded vision-based automation system for the detecting sewer blockages. The output of the present research contributes to the advancement of AI and its application in wastewater management. The knowledge and findings acquired from this research form a strong foundation for future explorations and advancements in the AI field and facilitating its widespread implementation across various domains.

For future research work integration of AI techniques like semantic segmentation, instance segmentation and panoptic segmentation, can be investigated to reinforce detection tasks. To enhance model robustness, expansion of representative datasets coupled with continuous learning approaches is recommended. For further practical application of the outcome of the thesis, collaboration with industry will yield advancements in AI innovation.

Table of Contents

Acknowledgement	i
Abstract	ii
1 Introduction	1
1.1 Challenges in AI.....	1
1.2 Data Types in AI	2
1.3 AI in Computer Vision (CV) and Research Significance	3
1.4 Thesis Organization.....	4
2 Literature Review	5
2.1 Object Detection Models with Structural Insights	5
2.1.1 YOLO Series.....	9
2.2 Sources for Availability of Representative Data in AI	10
2.3 Computer Vision and AI Approach in Sewer Inspection.....	12
2.4 Evaluation of Previous Surveys	13
2.5 Existing Automated Systems	15
2.6 Existing Vision Methods in SOP	18
2.7 Research Gaps	20
2.8 Problem Statement & hypothesis	21
2.9 Aim and Objectives.....	22
3 Theoretical Background	23
3.1 Role of Machine learning Techniques	23
3.2 Modern Approaches to Computer Vision Techniques.....	27
3.2.1 ML and DL in Computer Vision.....	27
3.2.2 Object Detection.....	28
3.2.3 Embedded Vision Approach.....	30
3.3 Selected Models for Methodical Approach.....	31
3.4 Transfer learning and Fine-tuning.....	35
3.5 Role of Artificial Learning in understanding physical mechanisms and developing predictive models in Different Research Domains	36
3.6 Significant breakthroughs in AI	38
3.7 Summary: Leading to the Methodical Approach	40
4 Methodology and Case Study with Results.....	42
4.1 Methodology	42
4.1.1 Development of New Critical Multiclass Representative Image Dataset	42

4.1.1.1	Survey Details of Pune Municipal Corporation (PMC)	43
4.1.1.2	Why do we need to develop a New Representative Dataset?.....	44
4.1.2	Methodical Flow for New Dataset and Detection Model Training	46
4.2	Tools Utilized in S-BIRD Dataset Generation.....	47
4.2.1	Constructed Sewer Pipeline	47
4.2.2	Inspection Camera for Sewerage Systems.....	48
4.3	Image Data Collection for the Development of Novel S-BIRD	49
4.4	Detailed Analysis of Captured Frames.....	50
4.5	Preprocessing and Augmentation Techniques	53
4.6	Annotated Heatmap and Object Count Histogram.....	58
4.7	Development of Sewer Blockage Detection Models using Transfer Learning and Fine Tunning	60
4.7.1	Optimization and Training of YOLOX using newly developed S-BIRD dataset.....	60
4.7.2	Optimization and Training of TOLOv5 using newly developed S-BIRD dataset.....	63
4.8	Embedded Vision Approach with S-BIRD	71
4.9	Discussion	73
4.9.1	Discussion on Enhanced AI in Research Work	73
4.9.2	Discussion on Case Study in Wastewater Management.....	74
4.9.3	Comparative discussion on AI-Driven Approach and MOEAs.....	76
5	Conclusions and Further work.....	77
5.1	Conclusions	77
5.2	Recommendations for Further Work.....	78
	References.....	79
	Appendix 1	85
	List of Published Journal Papers.....	85
	Published Book Chapter	85
	Other Publications.....	86
	Appendix 2.....	87
	Creating an implementation of the applied methodology involves following these steps using self-developed programming codes	87
	(a) Implementation of Preprocessing and Augmentation from Scratch	87
	(b) Object count histogram and heatmap implementation.....	92
	(c) Development of Model-1 using YOLOX, and its Training and Evaluation method in Code Pieces.....	94
	i. Configuration structure for S-BIRD dataset from scratch.....	94

ii. Development of model for training and validation operation from scratch on corresponding dataset	96
iii. Implementation of model evaluation from scratch on corresponding dataset	99
iv. Implementation of Real-time Detection task using multi-threading on embedded platform in given Code.....	102
(d) Development of Model-2 using YOLOv5, and its Training and Evaluation method in Code Pieces	106
i. Development of C3, SPPF, and Conv actual layer types from scratch, and necessary adjustments based on specific developed dataset -	106
ii. Configuration structure for S-BIRD dataset from scratch –	110
iii. Loss computation during training	112
iv. Programming Development of model training operation from scratch on corresponding dataset	117
v. SGD programme for customization	120
vi. Implementation of Real-time Detection task using multi-threading on embedded platform in given Code.....	123
Research Article – Review of the State-of-the-art Sewer Monitoring and Maintenance Systems Pune Municipal Corporation-A Case Study	127
Research Article – S-BIRD: A Novel Critical Multi-Class Imagery Dataset for Sewer Monitoring and Maintenance Systems.....	137
Research Article – AI-Driven High-Precision Model for Blockage Detection in Urban Wastewater Systems	155

List of Tables

Table 1 Main structural parts of object detectors.....	7
Table 2 Important techniques to obtain detection results	14
Table 3 Limitations and comments for existing automated systems	17
Table 4 Distinctions between types of sewer robotic systems.....	17
Table 5 Indirect techniques for sewer inspection	18
Table 6 Tools for maintenance of sewerage system	20
Table 7 Key phrases in ML with significances.....	26
Table 8 Details of survey conducted at PMC. [39].....	43
Table 9 Common Causes and Consequences of Major Sewer Blockages.....	45
Table 10 Technical details of the utilized sewer camera	48
Table 11 Arithmetical details of captured frames.....	51
Table 12 Annotations for training data	53
Table 13 Computational details of training samples in S-BIRD after preprocessing and augmentation.....	57
Table 14 Key Training Parameters	61
Table 15 Timing analysis of the trained model	61
Table 16 Precision evaluation of the trained model.....	61
Table 17 Key Training Parameters	64
Table 18 Timing analysis of the trained model	65
Table 19 Precision evaluation of the trained model.....	66

List of Figures

Figure 1 AI Techniques and required data type.....	2
Figure 2 Structure of single-stage and two-stage object detectors	6
Figure 3 Sonar technique for sewer inspection.....	19
Figure 4 Light and Mirror technique for sewer inspection.....	19
Figure 5 Closed Circuit Television (CCTV) with step van for sewer inspection.....	19
Figure 6 Closed Circuit Television (CCTV) technique for sewer inspection.....	20
Figure 7 Significant types of the ML techniques.....	24
Figure 8 Crucial steps to implement the ML technique.....	27
Figure 9 Steps in applying ML techniques to computer vision tasks	28
Figure 10 Illustration of YOLOX Decoupled Head	32
Figure 11 Detailing of DarkNet-53 CNN	32
Figure 12 YOLOv5 Arithmetical Details	33
Figure 13 YOLOv5 Architectural Details.....	34
Figure 14 Decision-making workflow for the development of new dataset.....	42
Figure 15 S-BIRD dataset including major sewer blockages	45
Figure 16 Methodical Workflow with newly developed dataset.....	46
Figure 17 Illustration of the constructed sewer pipeline with (a) material and diameter details and (b) realistic design and internal environment.....	47
Figure 18 Watertight sewer camera employed for frame capture.....	48
Figure 19 Frames depicting tree root blockages in the S-BIRD dataset.....	49
Figure 20 Frames illustrating plastic blockages in the S-BIRD dataset	49
Figure 21 Frames displaying grease blockages in the S-BIRD dataset	50
Figure 22 Annotated illustrations depicting the balance of sewer blockage types.....	52
Figure 23 Heatmap visualization of annotation details for recorded images.....	52
Figure 24 Data balancing for each class	53
Figure 25 Distribution Graph of Aspect Ratios	54
Figure 26 Illustrative outcomes of common augmentation methods: (a) grayscale transformation, (b) salt and pepper noise, (c) arbitrary exposure variation.....	56
Figure 27 Visual outcomes of enhanced augmentation methods: (a) cutout and (b) mosaic ..	57
Figure 28 Annotation specifications for each class in the training dataset following image-level augmentation.....	58

Figure 29 Heatmap of annotations providing location details of all classes.....	59
Figure 30 Histogram depicting the number of objects for: (a) grease, (b) tree roots, (c) plastic, and (d) all categories	59
Figure 31 Detection Results of YOLOX-s for Sewer Block Classes in S-BIRD	62
Figure 32 Visual Illustrations of Precise Detection of Tree Roots, Plastic, and Grease Sewer Block Types	62
Figure 33 Results from training process – (a) at epoch 832 (b) at epoch 932	64
Figure 34 Detection Results of YOLOv5-s for Sewer Block Classes in S-BIRD	66
Figure 35 Confusion matrix for classes within dataset.....	66
Figure 36 The scatter graph for instances and associated labels	67
Figure 37 Correlations within the dataset of sewer blockage frames	67
Figure 38 Precision (P) vs Confidence (C) graph.....	68
Figure 39 Recall (R) vs Confidence (C) graph	68
Figure 40 Precision (P) vs Recall (R) graph	69
Figure 41 F1 score vs Confidence (C) graph	69
Figure 42 Training and validation losses of the detection model	70
Figure 43 Precision (P) vs Recall (R) graph for model trained without using exposure in dataset	70
Figure 44 Detection Results on some Google Source images	71
Figure 45 Embedded vision based system emphasizing AI detection with S-BIRD.....	72
Figure 46 Incorporation of Embedded Vision platform into the sewer automated system	72

1 Introduction

Artificial Intelligence (AI) is a concept that aims to create intelligent machines that have the ability to think and make intelligent decisions similar to humans. Since the advent of computers, programs have been used to solve problems in different fields such as engineering and business. However, finding correlations for making predictive models has always been the centre of empirical analysis. A milestone in AI technology was the historical Paper by *John McCarthy & Marvin Minsky* in 1956 [1], in which they discussed the potential areas of AI, including language processing, neural networks, automatic theorem proving, and learning machines. Nevertheless, the computational power was too small to do anything substantial and computers did not have enough storage nor fast command processing power to exhibit intelligence. Continuous development in computing power and storage capacity allowed the storage of huge amounts of data generated by digital transformation of real-world information (such as records of weather indicators, personal information, audio, videos, pictures, etc.), which is impossible to analyse by humans. AI uses algorithms that allow computer/machine to learn without being programmed explicitly. These algorithms analyse large datasets and create systems that can carry out tasks like human intelligence and cognitive capabilities, for example decision-making, recognising patterns, etc. There are various sub fields in AI due to the basis of algorithms such as neural network, machine learning, deep learning and many more. The scientific approach to AI involves formulating hypotheses, testing, and analysing data to enhance the autonomy and accuracy of intelligent systems.

Today we live in the age of “big data,” where vast amount of data is collected, which beyond the data processing capacity of humans. For this reason, the application of AI is making its way in various industries such as engineering, security, banking, marketing, and entertainment. The algorithms have not improved much, but the big data and massive computing are allowing AI to make progress into many more areas. Data is the key and plays a central role in AI. Data can be numerical data, text data and visual data. In terms of acquiring data, it can be obtained by observation (actual recording the happenings), or synthetic data generated by models.

Application of AI in engineering is similar to modelling, data acquisition, data preparation, simulation and test, and implementation. Like for traditional statistical analysis, a representative dataset is an essential requirement for the development of reliable AI models in real-world applications. It is also important to consider privacy, data ownership, ethical factors, bias mitigation, data quality, informed consent, and regulatory compliance to ensure responsible and ethical use of data in AI applications.

1.1 Challenges in AI

AI modelling poses several challenges when it comes to development of models for practical applications. These challenges can significantly impact the effectiveness and reliability of AI systems. Some challenges in the application of AI are listed below:

- **Data Availability and Quality:** In some applications acquiring datasets can be challenging due to privacy concerns, data access restrictions, or unstructured data formats. In addition, ensuring data quality, free from biases and inaccuracies, is crucial to prevent erroneous predictions or decisions.

- **Model Selection and Evaluation:** The selection of appropriate AI models for a given task is a complex decision. Researchers and practitioners face the challenge of identifying the most suitable model architecture and algorithms that can effectively handle the specific problem domain. Evaluating the performance of AI models in a reliable and consistent manner is critical, but often challenging due to the absence of universally accepted evaluation metrics.
- **Interpretability and Explainability:** AI models, particularly Deep Learning (DL) models, are often considered black boxes, making it difficult to understand the underlying decision-making process. This lack of interpretability may raise concerns regarding the trustworthiness and ethical implications of AI systems when used in certain fields where transparency and responsibility are critical.
- **Scalability and Resource Constraints:** AI models often require significant computational resources and time for training, especially when dealing with large datasets or complex tasks. Therefore, scaling up AI models to handle big data or real-time applications that require large computing resources may become an issue due to budget limitations.
- **Ethical and Legal Considerations:** AI models can potentially amplify biases present in the training data or make decisions that have discriminatory effects. It is an ongoing challenge to ensure fairness, transparency, and accountability in AI systems particularly in the applications where ethical and legal considerations are necessary.

1.2 Data Types in AI

Distinct AI techniques with essential representative data types are explained in below Fig.1.

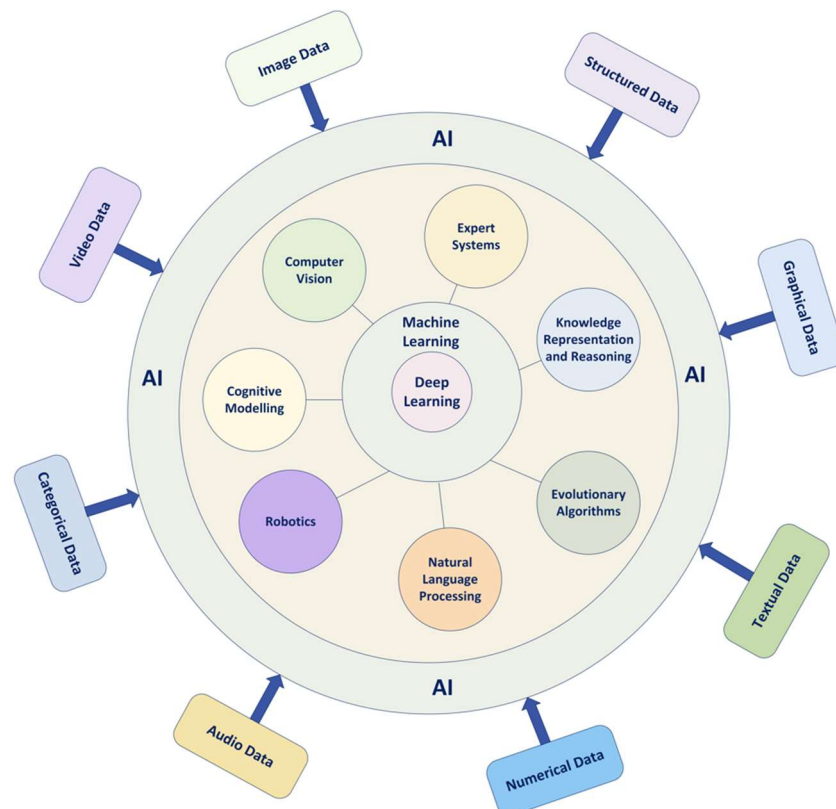


Figure 1 AI Techniques and required data type

- **Numerical Data:** It consists of numerical values and is one of the most common types of data used in AI. It includes continuous variables such as temperature, time, or sensor

readings. Numerical data can be processed using mathematical and statistical techniques, and it forms the basis for many machine learning algorithms.

- **Image Data:** This data consists of visual information in the form of pixels. It is commonly used in computer vision tasks such as object detection, image classification, and image generation. Deep learning (DL) algorithms, especially convolutional neural networks (CNNs), are widely employed to process and extract features from image data.
- **Video Data:** It consists of a sequence of frames, each containing visual information. It can include surveillance footage, movies, or videos captured from cameras. Video processing and analysis techniques are used to extract information, detect events, or recognize objects and actions in videos.
- **Textual Data:** It comprises unstructured text, such as documents, articles, emails, or social media posts. Natural Language Processing (NLP) techniques are used to analyse and extract meaningful information from text, enabling tasks like sentiment analysis, text classification, and language translation etc.
- **Audio Data:** This data represents sound waves and is used in various applications such as speech recognition, music analysis, and sound classification. Audio data analysis and interpretation make use of various techniques like signal processing and DL. The recurrent neural networks (RNNs) are widely employed to analyse and interpret audio data.
- **Temporal Data:** It involves sequences or time-series data points collected over time. Examples include stock market prices, sensor data, or weather patterns. Temporal data analysis often employs techniques like time-series analysis, RNNs, or Long Short Term Memory (LSTM) networks to capture patterns and make predictions.
- **Graphical Data:** It represents entities and their relationships, commonly visualized as nodes connected by edges. It is used in social network analysis, recommendation systems, and network analysis. Graph neural networks (GNNs) and graph-based algorithms are employed to analyse and extract insights from graphical data.

The choice of input data depends on the specific AI task, application domain, and the nature of the problem being solved. AI techniques employ a wide range of methods and approaches; namely machine learning (ML), deep learning (DL), Cognitive Modelling, and Evolutionary Algorithms, Computer Vision (CV), etc.

1.3 AI in Computer Vision (CV) and Research Significance

The computer vision enables machines to understand and interpret visual information from images or videos. It involves tasks such as object recognition, image classification, image segmentation, and object tracking. The computer vision algorithms include feature extraction, pattern recognition, and deep neural networks to extract meaningful information from visual data.

There are various fields that employ AI image recognition, ranging from recognising fruits and vegetables for labelling the produce to defence and healthcare. Image recognition systems are used to analyse visual data more efficiently, faster, and more accurately. Detection of blockages in the sewerage systems is one such application. Maintaining sewerage systems is a critical operational challenge for water and wastewater utilities. Identifying the type of blockage and predicting blockage formations in sewer pipes and pumping stations early so that required measures are taken before the blockage develops a service failure. In places where same network is used for storm water, heavy rainfall raises high levels within the sewer network due to additional water runoff entering the sewer system, that may trigger hundreds of alarms. The

volume of these alarms during wet weather periods can be unavoidable for operational and maintenance teams.

Particularly in the developed world, smart water and wastewater networks are at the forefront of investment plans for authorities as a step towards circular economy. With technological advancements it is possible to gather more information to allow water companies to implement AI for better management. Autonomous robots appear to have great potential for inspecting difficult to access water pipe networks [2]. A report on Robotic Autonomous Systems (RAS) by TWENTY65, emphasises the importance of sewer monitoring in the practical world [3].

Developing countries like India, where traditionally human scavenging was used for cleaning blocked pipes, have started to use mechanical systems and robotic scavengers. These automated methods of maintaining sewers critically employ AI techniques for improving the performance of detection of blockages and planning their removal. This thesis focuses on developing and implementing AI techniques to detect blockages and select appropriate unblocking techniques. This work is a part of an EU-India collaborative project Horizon 2020 SPRING, which focuses on developing wastewater management technologies. The case used for developing and implementing AI techniques is the sewerage system within Pune municipality (India). The research work includes developing a new representative image dataset and AI model training through transfer learning followed by fine-tuning techniques to improve the model's performance and effectiveness for detecting different types of blockages in the sewerage network.

1.4 Thesis Organization

The work presented in thesis is organised as follows

Chapter 1 introduces the concept of AI, challenges in AI modelling, input data types, application fields of AI, research significance, and thesis organization.

Chapter 2 presents literature review leading to problem statement and justification of the objectives. It conducts a review of existing literature for examining different approaches in AI and computer vision. Research gaps and limitations in the current methods are discussed leading problem statement. Hypothesis of the research and objectives are stated in this chapter.

Chapter 3 provides theoretical background for distinct AI techniques, modern computer vision approaches, deep neural networks for methodical approach, artificial learning, and crucial advances in AI.

Chapter 4 gives details about applied methodology in the research work. It progresses by presenting a case study, conducting theoretical and mathematical analyses, elucidating the development of a representative dataset, providing intricate arithmetic details, describing the experimentation and validation processes, elaborating on the creation of detection models using AI techniques, and presenting the corresponding results and discussions.

Chapter 5 summarises the whole research with conclusions and provides recommendation for further work.

This organizational structure ensures a logical and coherent progression through the thesis, guiding the reader from the foundational concepts to the culmination of the research outcomes and their implications.

2 Literature Review

This chapter provides a detailed literature review related to the work presented in the thesis. The purpose is to discuss the state-of-the-art techniques and methods which have been considered for developing the methodology used in the research work.

2.1 Object Detection Models with Structural Insights

Over the years, various approaches have been developed to solve object detection problems and advance relevant algorithms. Here, a brief overview of some of the major techniques that have significantly impacted the field of object detection is provided. It will also offer valuable insights into the structural aspects of detection models, enabling a comprehensive understanding.

- **Evolution of Object Detection Algorithms:** Object detection algorithms have undergone significant evolution over the years, driven by advancements in machine learning and computer vision. Key contributions and approaches include:
 - **Traditional Approaches:** Earlier object detection algorithms relied on handcrafted features and classical machine learning techniques. These methods used feature extraction techniques like Histogram of Oriented Gradients (HOG) and Haar-like features, combined with classifiers such as Support Vector Machines (SVM) or AdaBoost.
 - **Sliding Window Approaches:** Sliding window-based methods scanned the image at multiple scales and positions, applying a classifier to each window to determine if an object is present. This approach had limitations in terms of computational efficiency and accuracy due to exhaustive search over all possible windows.
 - **Region Proposal Approaches:** The introduction of region proposal methods, such as Selective Search and EdgeBoxes, improved efficiency by generating a set of potential object regions instead of exhaustive search. These methods reduced the number of windows to be evaluated, improving both speed and accuracy.
 - **Deep Learning Approaches:** The advent of deep learning revolutionized object detection. R-CNN, Faster R-CNN, YOLO, SSD, RetinaNet, etc.
- **Strengths and Limitations of Object Detection Algorithms:** When comparing different object detection algorithms, several factors need to be considered:
 - a) **Accuracy:** Accuracy measures how well the algorithm can correctly detect and classify objects. Deep learning-based algorithms, especially those using CNNs, have shown superior accuracy compared to traditional methods.
 - b) **Speed:** It is crucial for real-time applications. Traditional sliding window approaches were slower due to exhaustive search, while region proposal-based methods improved speed. Deep learning-based approaches like Faster R-CNN, YOLO, RetinaNet, etc further enhanced speed and efficiency.
 - c) **Robustness:** It refers to the algorithm's ability to handle various environmental conditions, such as changes in lighting, occlusions, and object deformations. Deep learning algorithms trained on large datasets have demonstrated improved robustness compared to traditional methods.
 - d) **Scalability:** Scalability relates to an algorithm's performance as the number of objects or complexity of the scene increases. Traditional methods often struggled with

scalability due to the large search space. Deep learning algorithms, especially those with region proposal networks, have shown better scalability.

- e) **Training Data Requirements:** Deep learning-based algorithms typically require large, labelled datasets for training, which can be a limitation in certain domains where labelled data is scarce or expensive to obtain.
- f) **Computational Resources:** Deep learning-based algorithms, particularly those with deep neural networks, require substantial computational resources during training and inference. This can be a limitation in resource-constrained environments.
- g) **Generalization:** Generalization refers to an algorithm's ability to perform well on unseen data. Deep learning algorithms trained on diverse datasets tend to exhibit better generalization, although overfitting can still occur if not properly regularized.
- h) **Interpretability:** Deep learning algorithms often lack interpretability compared to traditional methods. Understanding the decision-making process and explaining why a certain detection occurred can be challenging with complex neural networks.

Considering these factors, deep learning-based approaches have emerged as the state-of-the-art in object detection due to their balance between accuracy and speed. However, the choice of algorithm depends on the specific application requirements and constraints.

The advent of deep learning, particularly convolutional neural networks (CNNs), has revolutionized object detection. Deep learning-based detectors have shown remarkable performance improvements, leveraging large-scale datasets and powerful network architectures. They can automatically learn discriminative features and effectively handle complex visual patterns. These object detectors typically comprise two main components: a pretrained backbone that extracts features from input frames, and a head that utilizes these feature maps to estimate object classes and bounding boxes. In recent object detection models, an additional component known as the neck has been introduced. The neck consists of a few layers positioned between the backbone and the head, responsible for aggregating feature maps from different stages. Figure 2 [4] provides an illustration of the architectures of single-stage detectors like SSD and YOLO, which consist of a backbone and a densely predicted head. On the other hand, two-stage object detectors like Faster R-CNN and R-FCN include a backbone and a head with both dense and sparse predictions. Sparse and dense predictions refer to how object detectors make predictions at different spatial locations within an image.

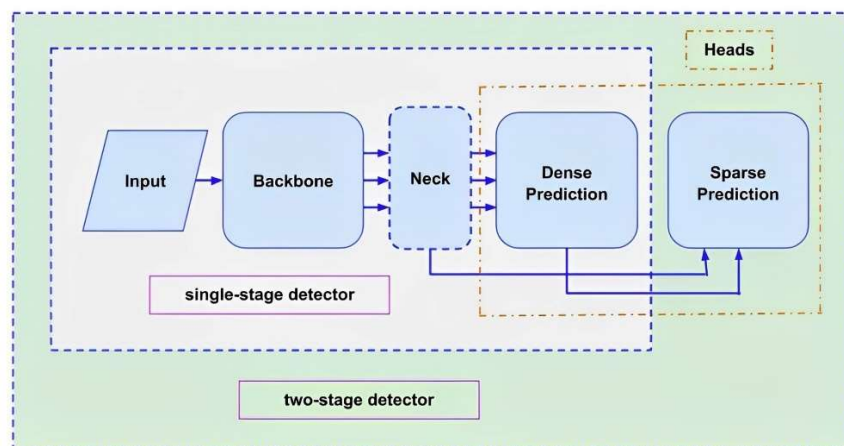


Figure 2 Structure of single-stage and two-stage object detectors

Dense predictions involve making predictions for every spatial location or grid cell in the input image. This means that the detector generates class probabilities and bounding box coordinates for multiple objects at each location. The dense predictions can be achieved by using techniques like fully convolutional networks (FCNs) or sliding window approaches. Sparse predictions, on the other hand, involve making predictions for a subset of selected regions or anchor boxes within an image. Instead of estimating object properties for every location, these detectors focus on a smaller set of regions or anchor boxes that are likely to contain objects. The regions or anchor boxes are determined through techniques like region proposal methods (e.g., selective search) or Region Proposal Networks (RPNs). Sparse predictions are common in methods like keypoint detection or landmark localization, where the focus is on specific points of interest. Both dense and sparse prediction strategies have their advantages and limitations. Dense predictions offer fine-grained object localization and can capture small objects effectively. However, they may introduce a large number of false positives due to the high-resolution output. Sparse predictions, on the other hand, focus on selected regions, which can reduce false positives and computational overhead. But they may struggle with detecting small objects or objects at different scales. The choice between dense and sparse predictions depends on the specific requirements of the application, including factors like speed, accuracy, and the size and diversity of the objects being detected.

However, single-stage detectors are favoured in real-time embedded applications due to their faster inference times compared to two-stage detectors. These object detectors integrated into automated systems play a crucial role in various fields. Table 1 [4] illustrates the components that adhere to the structural framework of object detector models

Table 1 Main structural parts of object detectors

Structural Parts		Details
Input		multi-scaled frames, frames, frame patches
Backbones		Darknet53, CSPDarknet-53, ResNet-152, ResNet-50, ResNet-10, Inception-ResNet-V2, GoogLeNet, DetNet-59, CBN, VGG16, ThunderNet, ViT, EfficientNet-B0/B7, etc.
Neck		FPN, Bi-FPN, PAN, SFAM, etc.
Heads	Dense	SqueezeDet, YOLO, SSD, DetectNet, RetinaNet, CenterNet, MatrixNet, etc.
	Sparse	R-FCN, Faster R-CNN, Mask R-CNN, Cascade R-CNN, etc.

Object detection algorithms employ various strategies, such as feature extraction, region proposal generation, classification, regression, and post-processing, to accurately detect and localize objects in images or videos. Region-based and anchor-based detectors are approaches within object detection that primarily deal with how objects are localized and matched within an image. Region-based detectors divide the object detection task into two stages: region proposal generation and object classification. They generate a set of candidate regions (bounding boxes) within an image using methods like selective search or region proposal

networks (RPNs). The regions are then classified to determine if they contain an object or not. This approach, used in method like R-CNN, allows for accurate localization but can be computationally expensive. Whereas, anchor-based detectors, such as SSD and Faster R-CNN, use predefined anchor boxes (also known as priors) at various scales and aspect ratios. These anchor boxes serve as reference templates to match objects present in the image. The detectors predict offsets and class probabilities for each anchor box to determine the final bounding box predictions. This approach allows for handling objects of different sizes and aspect ratios efficiently. Anchor-free detectors, like CornerNet and CenterNet, do not rely on predefined anchor boxes. Instead, they directly predict the bounding box coordinates and class probabilities without the need for anchor box matching. This simplifies the detection process and can be more suitable for objects with diverse scales and aspect ratios.

Girshick et al. [5], introduced the Region-based Convolutional Neural Networks (R-CNN) framework, which revolutionized the field of object detection and semantic segmentation. R-CNN presents the idea of using region proposals to select a set of potential object locations in an image, followed by applying a convolutional neural network (CNN) to classify and refine those regions. It achieved promising results but was computationally expensive. Further, *Girshick* improved R-CNN object detection framework and presented ‘Fast R-CNN’ [6]. The enhancement was done by proposing a unified architecture that shared the computation of the CNN across different region proposals, resulting in faster processing. It also introduced a region of interest (RoI) pooling layer to extract fixed-size features from the region proposals. Fast R-CNN demonstrated the benefits of shared feature extraction and end-to-end training for object detection, paving the way for further advancements in the field. *Ren et al.* [7], initiated Faster R-CNN which addressed the drawbacks of the previous methods by introducing a Region Proposal Network (RPN) that shared convolutional layers with the detection network. This allowed for end-to-end training and significantly improved the speed and accuracy of object detection. *Redmon et al.* [8], came up with YOLO (You Only Look Once) detection framework which actually revolutionized object detection by introducing a single-stage detection algorithm that jointly predicted class probabilities and bounding box coordinates using a single pass of the neural network. This resulted in real-time performance, but it faced challenges with smaller object detection. It was evaluated on the PASCAL VOC and COCO datasets and achieved competitive results compared to existing state-of-the-art methods. *Liu et al.* [9], produced SSD (Single Shot MultiBox Detector) which aimed to improve the speed and accuracy of object detection by utilizing a series of convolutional feature maps at different scales to detect objects of various sizes. It combined the benefits of both region proposal methods and dense prediction techniques. It employed default anchor boxes and predicted offsets and class probabilities for each anchor, enabling efficient and accurate detection. *Lin et al.* [10], presented ‘RetinaNet’ framework. It introduced the focal loss, which addressed the problem of extreme class imbalance during training in dense object detection. It assigned higher weights to challenging examples and down-weighted easy examples to improve the model's performance, making it particularly effective for detecting objects at different scales. *Kaiwen Duan et al.* [11], conferred ‘CenterNet’, a keypoint-based object detection framework that utilizes triplet keypoints for accurate and efficient object localization. It employs a fully convolutional network architecture, often based on popular backbone networks such as Hourglass or ResNet, for feature extraction. The network predicts heatmaps for object centres and offset vectors to locate the bounding boxes around each centre point. It achieves competitive accuracy in object detection tasks and performs well across various object scales

and occlusion scenarios. *Hei Law and Jia Deng* presented ‘CornerNet’ that employs a deep neural network architecture based on Hourglass modules for object detection. It detects objects by predicting the top-left and bottom-right corners of their bounding boxes as paired keypoints [12]. This representation enables precise localization and better handling of object scale and aspect ratio variations. It is a two-stage architecture that includes a keypoint estimation network and a refinement network. The keypoint estimation network predicts corner heatmaps, and the refinement network refines the corner locations. Here, pooling mechanism aggregates information from the corner keypoints to enhance the localization accuracy and robustness.

2.1.1 YOLO Series

The YOLO (You Only Look Once) series of object detection models have made significant contributions to the field of computer vision. Here is an explanation of the YOLO models along with the key papers associated with each version:

Joseph Redmon and Ali Farhadi introduced YOLOv2 and YOLO9000 which comprise several improvements to the original YOLO [13]. They include the use of anchor boxes for better handling of object scales and aspect ratios, multi-scale training and testing, and incorporating unified object detection and classification on a large-scale dataset (COCO) along with ImageNet. The hierarchical classification approach and dataset combination contribute to improved accuracy and scalability, making YOLO9000 a significant advancement in the YOLO series of models. Further, they came up with YOLOv3 by introducing a few key modifications such as the Darknet-53 architecture, feature pyramid network (FPN), and multiple detection scales [14]. It achieved better performance and accuracy compared to the previous versions through architectural improvements and training techniques. Darknet-53 consists of 53 convolutional layers. This deeper network enables better feature extraction and representation compared to the shallower networks used in previous YOLO versions. A feature pyramid network allows to capture objects at different scales and improve detection performance on small objects. YOLOv3 detects objects at three different scales and this multiple detection scales approach allows the model to handle objects with varying scales and aspect ratios more effectively. *Alexey Bochkovskiy et al.* [15], initiated YOLOv4 model which aimed to optimize both speed and accuracy by introducing several architectural improvements, including CSPDarknet53 as the backbone to enhance information flow and improve performance, PANet (Path Aggregation Network) as the neck to help the model for capturing features at different scales by aggregating information from multiple levels of the feature pyramid, and various optimization techniques such as Mish activation function, CIOU loss, etc. It achieved state-of-the-art performance on multiple object detection benchmarks. The YOLOv5 and YOLOX models, which are discussed in detail in Chapter 3 of the theoretical background, have been considered for a methodical approach.

Each iteration of the YOLO series introduced novel techniques and architectural enhancements to improve object detection accuracy, efficiency, and speed. These models have been widely adopted in research and practical applications due to their competitive and real-time detection performance across different datasets.

The development of novel methodical architectures, fusion with other computer vision tasks, transfer learning, and the discovery and development of new datasets with evaluation metrics are driving progress in the field of AI, including object detection. It continues to be an active area of research, with ongoing efforts to enhance detection performance. So, the advances in

this are paving the way for the deployment of intelligent systems in various domains, enabling machines to interact and understand the visual world around them.

2.2 Sources for Availability of Representative Data in AI

Sources of representative data can vary depending on the specific application or domain. Here are some common sources where representative data may be available:

- Publicly available datasets: Numerous organizations and research institutions make their datasets publicly available for AI research. These datasets cover a wide range of domains such as image recognition, natural language processing, and healthcare. Examples include ImageNet, COCO, and MNIST.
- Open data initiatives: Governments and public institutions often release datasets related to demographics, transportation, weather, and more. These datasets can be valuable sources of representative data for AI applications.
- Web scraping: The internet contains vast amounts of data that can be scraped and used for AI training. However, it is important to respect the terms of service and legal guidelines when scraping data from websites.
- Data discovery platforms: Online platforms exist that facilitate the exchange of data, where individuals or organizations can buy or sell datasets. These platforms often cover diverse domains and can provide access to representative data.
- Academic research papers: Research papers often provide datasets used in experiments or evaluations. Many papers include links to download the data or provide instructions on how to access it. Platforms like arXiv, IEEE Xplore, and ACM Digital Library are good resources for finding research papers.
- Crowdsourcing: Platforms like Amazon Mechanical Turk or specialized crowdsourcing platforms allow researchers to collect data by outsourcing tasks to human workers. This method can be employed to gather labelled or annotated data for training AI models.
- Data collection initiatives: Organizations sometimes conduct data collection initiatives specifically aimed at creating representative datasets. They may employ various methods, such as surveys, crowdsourcing, or partnerships with data providers, to collect comprehensive and diverse data.
- Data augmentation techniques: In some cases, representative data can be generated or expanded using data augmentation techniques. These techniques involve applying transformations or modifications to existing data to create additional representative examples.
- Data collaboration: Collaborations among researchers, industry professionals, and data scientists can lead to the pooling of data resources, allowing access to larger and more representative datasets.

While representative data plays a crucial role in AI, it is essential to address various aspects such as privacy, data ownership, and ethical considerations when sourcing and utilizing data for AI applications.

Gebru et al. [16], introduced the concept of datasheets for datasets, which provide a structured framework for documenting critical information about datasets, including their collection process, potential biases, and limitations. It highlights the importance of representative data to avoid biased and unfair AI systems. *Bhardwaj et al.* [17], presented DataHub, a platform for dataset management and collaboration. This work discusses the features of DataHub and how

it enables dataset search, versioning, and sharing among data scientists and researchers. It also evaluates multiple dataset search platforms based on various criteria such as dataset coverage, metadata quality, and search performance. The paper underscores the importance of dataset search in supporting AI research. *Umbrich et al.* [18], focussed on the evaluation and evolution of open data portals, which are online platforms that provide access to datasets from various sources. They proposed a quality assessment framework for open data portals that comprises a set of metrics to evaluate various aspects of the portals, such as data availability, freshness, relevance, and usability. The contribution is about the understanding of open data portals and offers practical guidance for their improvement to better serve the needs of data users and the broader community. *Koesten et al.* [19], focussed on the concept of data summarization and its importance in understanding and utilizing datasets effectively. They explored different dimensions and aspects that are relevant to users when working with datasets. The studies cover topics such as data availability, provenance, quality, statistics, and schema information. It provides insights into the information needs of users and presents guidelines for designing effective dataset summaries, with the aim of improving data comprehension, decision-making, and collaboration in various domains.

AI plays a crucial role in enhancing computer vision capabilities by employing intelligent algorithms to extract valuable digital statistics from images and videos. This augmentation enables automated systems based on embedded platforms to possess greater vision power and intelligence. To achieve advanced results, it is imperative to have a large quantity of appropriate and labelled data for training AI's Deep Neural Object Detection Models. In the realm of AI, a dataset refers to a collection of significant and distinctive details within a particular field. These datasets are utilized for training AI models with specific objectives, including clustering, segmentation, regression, classification, and detection. Various types of data can be found, such as images, time series, numerical data, graphs, text, and so on. It is essential to recognize that the performance of a detection model heavily relies on the quality of the dataset used for training. Even the best detection model will yield inferior results if trained on a poor dataset. On the other hand, a poorly performing detection model can benefit from a highly featured and high-quality dataset. At the core of single-stage or two-stage object detectors lies a classifier responsible for identifying the intended object classes. It becomes evident that the performance and accuracy of any detection model are determined by the quality of the input imagery dataset. Therefore, having a comprehensive, diverse, and accurately labelled dataset significantly contributes to the effectiveness of object detection models.

Obtaining a relevant dataset for training AI models and achieving accurate results is a crucial requirement and a significant focus of research in relevant communities. This involves acquiring or collecting data, appropriately labelling the data, and improving the available data or models [20]. Many funding agencies have embraced an open-access research strategy, resulting in the availability of large datasets from various fields on different platforms. Data can be obtained from data-sharing platforms like Kaggle datasets [21], DataHub [17], Mendeley Data [22], and data-searching platforms like IEEE DataPort [24], Google Dataset Search [23], and others. Despite challenges in data discovery, researchers can succeed in obtaining the necessary dataset [25]. In 2011, the difficulties in accessing and tracing open data were acknowledged, leading to the regulation of data publishing movements in Europe [26]. Six barriers to obtaining and tracing open data were identified, including limited information about data existence and accessibility, uncertainty regarding data ownership by government

authorities, ambiguity concerning terms of reuse, data cost and its sensitivity, complex licensing procedures and high fees, specific reuse contracts with professional members, and restrictions on recycling for state-owned companies.

Notably, data acquisition involves various functions such as searching, augmenting, and generating data as needed. In our case, the dataset is not only generated due to unavailability but also undergoes individual preprocessing, augmentation, and labelling for classification and detection tasks. The dataset can be created manually or through automated techniques, and synthetic data is used to fill in any missing parts. For optimal learning models, standardized or benchmark datasets are preferred, and transfer learning techniques can be applied using representative datasets. Transfer learning in computer vision refers to the process of leveraging knowledge from a pre-trained model on a large dataset and applying it to a new task with limited labelled data. It involves using the learned features and representations from the pre-trained model as a starting point for the new task, allowing the model to benefit from the general visual knowledge gained during pre-training. Fine-tuning, on the other hand, involves further training the pre-trained model on the new task-specific dataset. By updating the model's parameters using the task-specific data, it adapts the learned representations to the nuances and characteristics of the new task, improving its performance and generalization, and helping avoid overfitting. In computer vision problems, a digital imagery dataset with object class details is divided into a training set, validation set, and testing set. These sets are then used as input for AI models to facilitate training, evaluation, and testing, respectively. Cross-validation techniques such as holdout, k-fold, and bootstrap can be employed to ensure the selection of the most suitable model during the training process. These techniques help in avoiding bias in the dataset or training model and ensure relevant results.

2.3 Computer Vision and AI Approach in Sewer Inspection

This discussion focuses on examining the influence and constraints of notable contributions in the realms of computer vision and AI, aiming to define the boundaries and possibilities within these domains.

Kumar and Abraham introduced a framework that utilized Deep Convolutional Neural Networks (CNN) to classify various issues, such as cracks, root intrusions, and deposits in CCTV frames of sewer pipelines [27]. Their study involved training and evaluating the CNNs using a dataset of 12,000 frames from more than 200 sewer pipelines. However, it is important to note that their work focused on static frames rather than real-time navigation, and they primarily classified faults without providing information about their specific location (localization). *Cheng and Wang* proposed an automated approach centered around fast R-CNN for fault detection in sewer pipes [28]. They trained a detection model using a dataset of 3,000 sewer pipe images extracted from CCTV inspection videos. The accuracy and computational cost of the model were analysed using metrics such as mean accuracy (MAP), training time, missing rate, and detection speed. Similar to the previous study, this work primarily focused on analysing standing frames rather than real-time frames and encountered some misclassification issues for cracks during the experiments.

Gutiérrez-Mondragón et al. developed a training technique for a convolutional neural network aimed at detecting levels of obstruction in sewer pipes [29]. They trained their model using significant frames extracted from a CCTV video database. Additionally, they integrated the Layerwise Relevance Propagation explainability technique to gain insights into the neural

networks' behaviour and performance in relational tasks. The authors predicted that their proposed system could offer high accuracy, speed, and consistency for real-time sewer inspection. However, it is worth mentioning that this work considers the degree of blockage in the drain but does not provide information regarding the specific type and location of the blockage.

Halfawy and Hengmeechai proposed a systematic algorithm combining HOG (Histogram of Oriented Gradient) and SVM (Support Vector Machine) to detect tree root intrusion faults in conventional CCTV monitoring videos [30]. The algorithm consisted of two steps: (a) segmenting the frames to extract regions of interest (ROIs) indicating defect regions, and (b) applying an SVM classifier trained with HOG features to classify the ROIs. It should be noted that this approach only considered static frames and did not account for large datasets or video sequences. *Yin et al.* developed a framework for real-time automatic fault detection in sewer pipes using a CNN-based YOLOv3 object detector [31]. Their model was trained on a dataset of 4056 frames, including six classes of defects such as holes, breaks, cracks, deposits, fractures, and roots. The framework also incorporated construction feature detection. However, it is worth mentioning that this model has not been tested in a real-time sewer pipe scenario and may require further improvements in performance.

Moradi et al. introduced an automated method that utilized computer vision techniques for the inspection and condition assessment of sewer pipelines [32]. The process involved identifying a region of interest (ROI) containing sewer defects and then classifying the frames. They used Hidden Markov Models (HMMs) to extract sewer frames from CCTV videos and employed CNNs for defect detection and classification. *Kumar et al.* evaluated deep learning-based frameworks such as YOLO, SSD, and Faster R-CNN for speed and accuracy in detecting and localizing root infiltration and deposits in CCTV sewer frames [33]. They trained and tested their models using a collection of 3800 annotated frames. The faster R-CNN model achieved the highest accuracy in defect detection, although it had the slowest processing speed per frame. The YOLOv3 model had slightly lower accuracy but a processing speed almost twice as fast as the faster R-CNN. The SSD model exhibited the lowest accuracy but the highest processing speed per image. However, it is important to note that the dataset used for training and testing in this study was relatively small, which may have limited the achievement of desired results.

2.4 Evaluation of Previous Surveys

A comprehensive examination of image-based automation in closed-circuit television (CCTV) and sewer scanner and evaluation technology (SSET) is presented by reviewing 25 years of sewer inspection research [34]. This survey conducted by *Haurum and Moeslund*, examines pipeline algorithms and datasets, along with the protocols used in sewer inspection. The survey investigates various aspects of automated sewer pipeline inspection, including frame acquisition, pre-processing, detection and segmentation, feature description, classification, and temporal filtering. The survey suggests the creation of free and publicly available datasets for each release, accompanied by open-source code and standardized evaluation metrics.

Another review by *Moradi et al.* focuses on recent sewer inspection technologies utilizing computer vision and machine learning techniques [35]. The review compares the advantages and disadvantages of different methods through evaluation. It thoroughly investigates image representation, image pre-processing, and learning techniques for sewer pipe fault detection. The review recommends the use of a standard CCTV camera, effective hardware with high

specifications, and standardized datasets with robust algorithms. *Liu and Kleiner* present sewer pipe inspection and evaluation techniques, discussing augmented reality, smart pipes, and intelligent robots [36]. They assess the functionality of these technologies and their relevance to real-world applications. The importance of CCTV and laser scanning techniques is also emphasized.

Tur and Garthwaite analyse available robotic tools and identify unresolved issues in the successful implementation of sewer inspection systems [37]. They shed light on various automated systems, sensing techniques, SSET, and CCTV techniques. The authors suggest that automated systems should be programmed for specific tasks to reduce costs and minimize energy consumption. They also highlight the need for advanced artificial visual processing techniques, deep learning algorithms, and supervised/unsupervised algorithms in fault detection and classification. In another review conducted by *Czimmermann et al.*, the focus is on fault detection and classification using advanced artificial visual processing techniques, deep learning algorithms, supervised and unsupervised algorithms [38]. The authors note that challenges such as insufficient test samples, inconsistent databases, and a lack of solid algorithms hinder the implementation of ideal sewer inspection systems.

Overall, these surveys and reviews provide a detailed analysis of computer vision and AI based automation in CCTV and sewer inspection technologies. They offer insights into the strengths and weaknesses of different methods, recommend best practices for hardware and datasets, and highlight the importance of advanced techniques and standardized evaluation metrics in achieving effective sewer inspection systems.

Previous research studies have often focused on various common issues that arise in sewer systems, including breaks, tree root infiltration, holes, cracks, deposits, fractures, and obstacles. However, the most significant problem encountered is blockages in sewers, which occur due to the accumulation of various types of waste such as sludge, rocks, toilet waste, plastic, tree roots, leaves, grease, and foreign objects. These blockages pose a major challenge in maintaining the functionality of sewer networks.

In Table 2, the techniques applied to obtain detection results are listed due to their significance and relevance in previous research endeavours [39]. It enhances the transparency of the study by explicitly referencing and acknowledging the techniques that contributed to the obtained results.

Table 2 Important techniques to obtain detection results

Conventional Algorithms in Computer Vision for pre-processing and detection task	<ul style="list-style-type: none"> • Geometric transformations • Thresholding • Morphological operations • Noise removing • Image stitching, mosaicking, and unwrapping • Colour spaces • Image enhancement and filtering
Learning and Classification Techniques in Machine Learning (ML)	<ul style="list-style-type: none"> • Decision Trees • Random Forests • k-means • Logistic Regression

	<ul style="list-style-type: none"> • SVM • k-NN • Naïve Bayes
Object detection models in Deep Learning (DL)	<ul style="list-style-type: none"> • Faster-RCNN • SSD VVG • Tiny YOLOv2 • YOLOv3
Deep Learning based Classification models	<ul style="list-style-type: none"> • GoogleNet • MobileNet v2 • AlexNet • CaffeNet • SqueezeNet • ZFNet 512 • DenseNet 121 • ResNet – 18v1, ResNet – 50v1 • CNN Mnist • ShuffleNet
Deep Neural Network (DNN) Models for Segmentation	<ul style="list-style-type: none"> • ResNet 101_DUC_HDC • Mask R-CNN • ENet • FCN

2.5 Existing Automated Systems

In this passage, the discussion focuses on various existing automated systems and highlight their features, limitations, and potential improvements. The aim is to provide a comprehensive overview of the advancements made in this area for AI and computer vision and draw conclusions regarding the state of the art.

The first system discussed is PIRAT (Pipe Inspection Real-Time Assessment Technique), which was evolved by *Kirkham et al.* [40]. PIRAT is a semi-autonomous tethered system that uses interpretation techniques to assess physical data. It employs a three-dimensional model for classifying and detecting damages. However, the system has certain limitations. It requires a human operator to manually detect and mark the damaged areas in the images, making it less efficient. Additionally, the proposed algorithm is a decade old, suggesting that it may lack some of the more recent advancements in the field. Next, *Kuntz et al.* [41] developed KARO (KAnalRoboter), another tethered, semi-autonomous sewer inspection device. KARO features self-correcting tilting poses and wheel slippage. It utilizes 3D optical sensors and microwave sensors to detect damages such as cracks, bends, and blockages. However, this system heavily relies on sensors, and the onboard hardware is not as advanced as desired. The main control unit is located at a distant place, which can introduce communication delays and potential issues.

Kirchner and Hertzberg introduced KURT (Canal-Untersuchungs-Robot-Testplattform) in [42]. KURT1, a part of this system, focuses on autonomous navigation in dry sewer networks. It classifies pipe junction types and has the potential for mapping sewer landmarks. KURT2, on the other hand, incorporates sensory platforms such as optional bumpers, odometry sensors, an inclinometer, obstacle detection, and ultrasound distance measurement using an infrared

transducer. While KURT demonstrates autonomous navigation capabilities, it may lack some advanced computer vision techniques. *Rome et al.* [43] presented the MAKRO (Mehrsegmentiger Autonomer KAnalRoboter) robot, which utilizes an ultrasound range sensor to detect obstructions in sewer pipes. It also incorporates collision avoidance, landmark detection, and speed control functions. However, the authors note that the system lacks efficient use of computer vision techniques, which could potentially enhance its capabilities. *Nassiraei et al.* [44] developed the KANTARO system, which features an intelligent modular architecture with implicated sensors and mechanisms. It employs a small-sized smart 2D laser scanner to detect directional markings, while a fisheye camera evaluates pipe condition and detects defects. The system demonstrates a promising combination of sensor technologies.

Alejo et al. [45] presented the SIAR (Sewer Inspection Autonomous Robot) system, capable of detecting critical structural defects in pipelines. SIAR employs real-time 3D structure reconstruction techniques and collects environmental water or gas samples for analysis purposes. It utilizes RGB-D sensors and an impressive wireless transmission network. This system showcases advanced capabilities in real-time data collection and analysis. *Abidin* [46] introduced an in-pipe robot that uses an ultrasonic sensor to detect differences in diameter, indicating the presence of a blockage if the diameter is small. It can clean soft and medium clogs and operates at distances of less than 30 mm. However, it should be noted that this is a basic laboratory-scale experiment, and further development is required for practical implementation. *BhrtyArtana*, as described by *Vaani et al.* [47], is a system designed to detect corrosion, cracks, and obstacles in turbine mechanisms. It utilizes a camera to capture real-time frames and a proximity sensor to identify obstacles. When an obstacle is detected, the system employs a turbine mechanism to cut and clear the obstruction.

Gobinath and Malathi developed a relatively expensive robotic machine [48] equipped with a robot arm capable of moving in different angles, from left to right and top to bottom. This machine is specifically designed for sewer cleaning purposes. It incorporates a SewerSnort gas board to detect toxic gases and an LCD display to visualize the cleaning process. *Prasad and Karthikeyan* designed a robot [49] to clean and eliminate obstacles in large sewer pipes. Obstructions are detected using an ultrasonic sensor, and a drilling technique is employed to remove them. A MATLAB tool is utilized to observe wireless camera videos and frames. *Abro* introduced an autonomous system called SewerBot [50], which employs digital image processing to detect defects in sewerage pipelines. The system uses gradient and segmentation techniques with the assistance of wireless cameras to identify sewer pipe blockages. However, the algorithm and performance of the system presented by Abro were found to be subpar for practical implementation.

In conclusion, several automated sewer inspection systems have been developed with varying levels of autonomy and capabilities. These systems utilize different sensors, techniques, and algorithms to detect damages, blockages, and other structural defects in sewer pipes. While some systems require human intervention for certain tasks, others aim to achieve full autonomy. However, there are limitations and areas for improvement in terms of efficiency, use of computer vision techniques, practical implementation, and system performance. These limitations and comments for the existing automated systems are summarized in Table 3.

Table 3 Limitations and comments for existing automated systems

Automated Systems	Ref. No.	Features	Limitations and Comments
KARO	[41]	Tethered, 3D optical sensors	Possible to work through the acquired sensory data information and depend on the reliability of the human operators
MAKRO	[43]	Ultrasound range sensor, collision avoidance	Absence of effective Computer vision technique and not able to navigate inside bending pipes
SIAR	[45]	3D structure reconstruction, RGB-D sensors	Innovative system for inspection and sample collection purposes but not capable of corrective action.
PIRAT	[40]	Semi-autonomous, 3D models for damage classification	Depends on reliability of human operator and lack of onboard control routine
KURT	[42]	Autonomous navigation, sensory platforms	System used entirely sensors and it was affected by ecological attributes
KANTARO	[44]	Modular architecture, fisheye camera	Defects detection software had lower precision rate and lack of systematic approach to improve logically
Machine Robot	[48]	Robot arm, SewerSnort gas board	Expensive and requires adaption to techniques for system development.
In-pipe Robot	[46]	Ultrasonic sensor	Elementary system and not convenient for the practical world
Sewerbot	[50]	gradient and segmentation	Need for efficiency improvement and reduction of poor techniques for practical development.
BhrtyArtana	[47]	proximity sensor and camera for detection	Need of methodical approach capable of being applied to real fault detection and cleaning situations
MATLAB Based Robot	[49]	Ultrasonic sensor, drilling technique	High improvement needs in applied computer vision method

The following table, Table 4, provides a distinction between various types of sewer robotic assemblies.

Table 4 Distinctions between types of sewer robotic systems

Full autonomy	Semi autonomy	No autonomy
The reliability of the evaluation depends on the intelligence of the system	The reliability of the evaluation depends on both the intelligence of the system and human operator.	The reliability of the evaluation depends on human operator
Not reliable in small diameter pipes	Recommendable in lesser diameter pipes	Acceptable in lesser diameter pipes

Un-tethered	Might be tethered or un-tethered	Tethered
Brings all mandatory resources onboard	All mandatory resources may be brought onboard, or the control unit might be located at a remote location	Control unit is located at remote location
Absolute intelligence for self-navigation	Teleoperated with some degree of self-intelligence	Fully teleoperated
Includes many sensors and intricated navigation structure	Includes average sensors with navigation structure	Includes less sensors and operated only by a human

2.6 Existing Vision Methods in SOP

Sewers are essential underground structures that are crucial for managing sewage in a city or town. They provide a network of pipes and channels through which wastewater flows to treatment plants or disposal sites. However, one of the major concerns in sewerage systems is the occurrence of blockages in sewer pipes, which can be caused by a variety of factors, both natural and human-made. Dealing with these blockages requires significant manpower and resources. Traditionally, in India manual cleaning methods were employed, but they pose serious risks to the workers' health and safety. Therefore, Government of India (GOI) introduced a standardized measure in August 2021 to eliminate manual scavenging and promote safer alternatives.

Even prior to this, the GOI had taken proactive steps to prevent hazardous and improper cleaning of drains and septic tanks, aiming to avoid accidents at all and ensure the well-being of workers. As per the presented Standard Operating Procedure (SOP) by The Ministry of Housing and Urban Affairs, India, some vision based indirect inspection technologies have been detailed in below given Table 5 [51].

Table 5 Indirect techniques for sewer inspection

Sr. No	Feasible Attributes			Technique
	Situation of Sewer	Composite for Sewer	Measurements of Sewer	
(1)	Unfilled	Altering	Diverse Measurement	CCTV
(2)	Entirely conducting	Altering	Diverse Measurement	Sonar Technique
(3)	Unfilled	Altering	prepared for 300 mm	Light and Mirror Technique

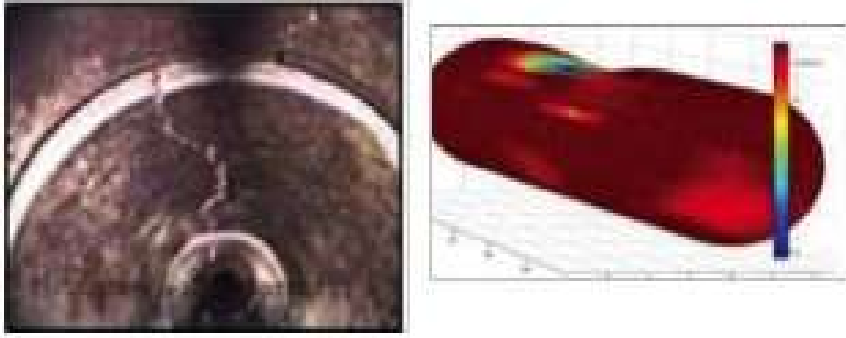


Figure 3 Sonar technique for sewer inspection

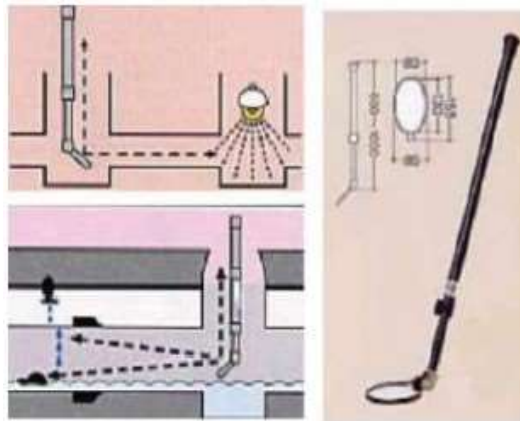


Figure 4 Light and Mirror technique for sewer inspection



Step Van



Camera Hoist Monitor Cable Reel System Tool Box Pulley/Guide Equipment Water Supply Reel

Figure 5 Closed Circuit Television (CCTV) with step van for sewer inspection

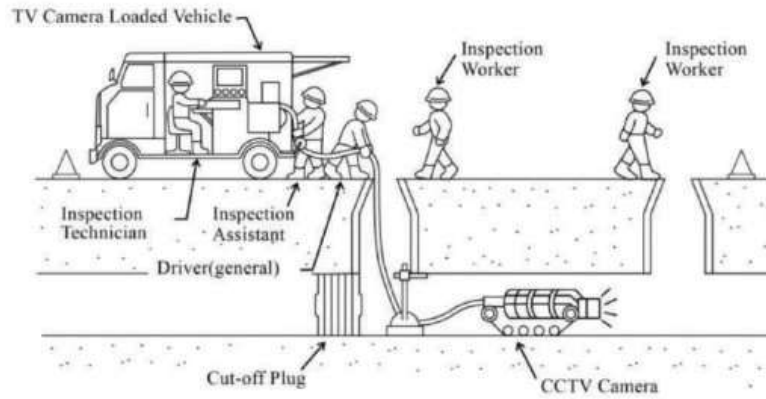


Figure 6 Closed Circuit Television (CCTV) technique for sewer inspection

Figures 3,4,5,6 illustrate indirect inspection functions for sewer systems [51], [52]. Whereas visual inspection by concerned authority is known as direct inspection. Tools utilised for maintenance of sewerage system are given in Table 6 [39], [51].

Table 6 Tools for maintenance of sewerage system

Sewer Maintenance Tools			
Automated Executions	Rodding Machine with Flexible Sewer Rods	Labour-intensive Executions	A collected wood board - Scraper
	Speedy cleaners (Jetting Machines)		Sectional Rods for Sewer
	Bucket Machine		
	Dredger (Clamshell)		
	Hydraulically Driven Tactics		
Gully Emptier		Cloth Ball and Manila Rope	

2.7 Research Gaps

After conducting a thorough review of existing literature, relating to AI detection techniques, computer vision approaches, sources of representative data availability and sewer inspection systems, the following research gaps have been identified:

- Inefficient utilization of computer vision algorithms with on-board processing: Existing detection techniques do not efficiently utilize computer vision algorithms that can process data on-board. There is a need to optimize these algorithms and adapt them for practical implementation.
- Lack of focus on sewer clogging issues: The majority of research in sewer inspection has primarily focused on detecting damages and clearing soft and medium clogs. However, little attention has been given to the problem of sewer clogging caused by debris accumulation. This indicates a research gap in the development of robust algorithms and automated systems capable of both real-time detection and removal of obstructions in sewer pipes.

- Lack of standardized dataset for sewer obstructions: Currently, there is no standardized dataset available that specifically addresses the problem of sewer obstructions. Moreover, issues related to personal liability, copyright, and privacy restrict the accessibility of existing datasets. Having an open and accessible research dataset would be beneficial for the research community to contribute and enhance the AI field more broadly.
- Opportunity for algorithmic model integration: The identified research gaps present an opportunity to develop an algorithmic model that combines computer vision and AI approaches. This model can be integrated with existing or newer automated systems used for inspecting and cleaning sewer systems. By leveraging these revolutionary techniques, more effective and efficient sewer inspection and maintenance processes can be achieved.
- Absence of accessible source code and evaluation metrics: In the research field of AI and computer vision focussing on sewer inspection, the availability of accessible source code for published work is limited. This hampers the replication and further development of existing algorithms. Additionally, the lack of approved evaluation metrics makes it challenging to compare and assess the performance of different approaches. Addressing these issues would promote transparency, reproducibility, and collaboration within the research community.
- Need to enhance Learning Strategy of AI Models: There is a need to improve the learning strategy of AI models in detection fields such as sewer inspection. This can be achieved through the use of representative data, transfer learning, and fine-tuning techniques. By incorporating these approaches, the performance parameters of AI models can be increased, making them more suitable for practical deployment.

In summary, the literature review identified several areas for improvement in the current research that involves AI techniques application to sewerage maintenance. These areas include the underutilization of computer vision algorithms, the lack of focus on sewer clogging, the potential for integrating algorithmic models, the absence of a standardized dataset, the limited availability of accessible source code and evaluation metrics, and the necessity to improve the learning strategy of AI models.

2.8 Problem Statement & hypothesis

The automated systems are capable of navigating and operating in hazardous, odorous, and sludgy areas. In order to develop advanced robotic solutions, AI techniques can be used which will allow inspection and cleaning of sewer systems. Obstructions in drains, displacement of joints, cracks, encroachment of tree roots are main reasons for deterioration of sewers that lead to sewage spills, endangering the environment, and causing public health problems. However, existing methods lack the assurance required for comprehensive sewer inspection and cleaning. In view of this, the problem statement is to develop AI-powered solution for sewer inspection and maintenance. The hypothesis suggests that the development and utilization of representative image datasets coupled with AI detection model can enhance the precision and efficacy of sewer blockage detection for removal with automated system. These detection model can offer efficient and cost-effective maintenance of real-world urban sewer systems.

2.9 Aim and Objectives

The aim of the research work presented in this thesis is to develop a new representative image dataset of sewerage blockages and develop AI model for their detection by training through transfer learning and fine-tuning techniques, with the goal of improving the model's performance and effectiveness for real-world applications.

The following research objectives are formulated to address the research gap and achieve the aim of this thesis work.

- To investigate AI techniques, including Machine Learning (ML) and Deep Learning (DL), and the structure of Deep Neural Object Detection Models.
- To develop a new representative image dataset, and analyse its strength, performance, consistency, and viability for real-time applications.
- To develop AI detection models using transfer learning and fine-tuning techniques on the new dataset, aiming to achieve a high precision rate for a specific application.
- To specify a methodical approach for system development based on embedded vision and integrate the trained detection model into an embedded processor for a certain real-time application.

In addition to above objectives, for the sewer maintaining applications specific objectives are achieved.

- To review existing automated systems and applied techniques used for sewer monitoring and maintenance purposes.
- To identify the constraints in existing AI and computer vision techniques for sewer inspection and cleaning in order to devise efficient solutions to overcome.
- To investigate distinct types of sewer pipe blockages and creating a new imagery dataset of sewer blockages caused by grease, plastic, and tree roots.
- To develop detection models by transfer learning and fine tuning with modifications using representative dataset for identification and localization of sewer blockages with high precision rate.
- To import trained detection model in embedded processor for real-time application and it can be added into existing or newly developed sewer automated system.

3 Theoretical Background

This chapter provides the theoretical foundation necessary for a comprehensive understanding of the concepts explored in the research work.

3.1 Role of Machine learning Techniques

AI has the ability to make decisions like humans and has standard rules encoded in the style computer programs. Machine Learning (ML), an inherent branch of Artificial Intelligence (AI), which is one of the most leading technologies in the current scenario. ML techniques encompass a wide range of algorithms and approaches used to enable computers to learn from data and make predictions or decisions. The action and reaction of big data in ML can be interchanged to attain maximum scalability, efficiency, and adaptability. Figure 7 [53] shows classification details of ML techniques.

Arthur Samuel, an American pioneer in the field of AI and computer gaming, earliest presented the phrase ML in 1959 and delineated it as, “it gives computers the ability to learn without being explicitly programmed.” Later, *Tom Mitchell* in 1997 specified ML as, “A computer program is said to learn from experience E concerning some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

The following are significant types of ML techniques.

- **Supervised Learning:** In supervised learning, the algorithm is trained on labelled data, where each data point has a corresponding target or output label. The algorithm learns to map input features to the desired output based on the provided examples. Popular supervised learning algorithms include decision trees, support vector machines (SVM), and neural networks.
- **Unsupervised Learning:** Unsupervised learning involves training models on unlabelled data, where the algorithm aims to discover patterns or relationships in the data without any specific target variable. Clustering algorithms, such as k-means clustering and hierarchical clustering, are common unsupervised learning techniques. Dimensionality reduction techniques like principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE) are also used for unsupervised learning.
- **Reinforcement Learning:** Reinforcement learning focuses on training an agent to interact with an environment and learn optimal actions to maximize a reward signal. The agent learns through trial and error, receiving feedback in the form of rewards or penalties. Reinforcement learning techniques are commonly used in areas such as robotics, game playing, and autonomous systems.

Further, Deep Learning (DL) is a subset of machine learning that leverages artificial neural networks with multiple layers to learn complex patterns and representations from data. Deep Neural Networks apply subtractive computation at numerous levels to perform human-like tasks [54]. DL has revolutionized fields like computer vision, natural language processing, speech recognition, etc. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are commonly used deep learning architectures.

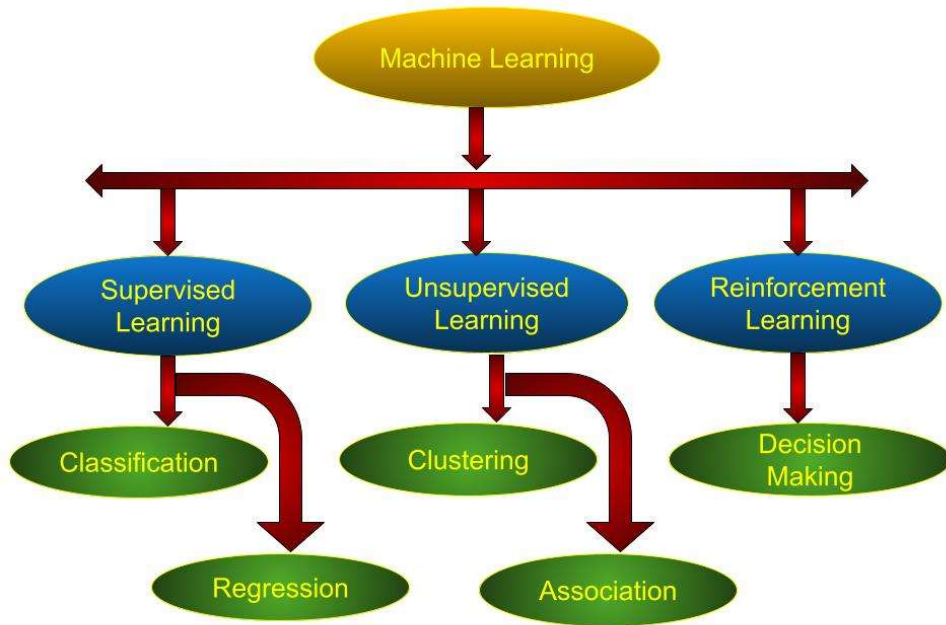


Figure 7 Significant types of the ML techniques

- CNNs are primarily used for image and video processing tasks. They employ convolutional layers to extract local features and pooling layers to downsample and aggregate information. CNNs have demonstrated exceptional performance in tasks such as image classification, object detection, and image segmentation. In general CNN equation can be expressed as follows:

$$\mathbf{y} = f\left(\sum_{i=1}^N \mathbf{W}_i * \mathbf{x}_i + \mathbf{b}\right) \quad (3.1)$$

Where,

$f(\cdot)$ is the activation function applied element-wise to the sum of convolutions.

N is the number of input channels.

\mathbf{W}_i represents the i -th set of learnable convolutional filters (also called kernels or weights).

$*$ denotes the convolution operation.

\mathbf{x}_i represents the i -th input feature map or activation map.

\mathbf{b} is the bias term applied to each convolutional filter.

\mathbf{y} represents the output feature map or activation map of the CNN.

The convolution operation involves sliding each filter over the input feature map, computing element-wise multiplications between the filter weights and the corresponding input values, summing up the results, and applying the activation function. This process generates the output feature map \mathbf{y} . The sizes of the filters, input feature maps, and output feature maps determine

the dimensions of the convolutional layers in the CNN. The specific architecture and layer configurations of a CNN can vary based on the problem domain and design choices.

The CNN equation for processing an image can be broken down into the following steps:

- **Input:** Consider a colour image with dimensions $H \times W \times C$, where H represents the height, W represents the width, and C represents the number of channels (usually 3 for RGB images).
- **Convolutional Layer:** The convolutional layer applies a set of filters to the input image. Each filter has dimensions $K \times K \times C$, where K is the size of the filter (often 3×3 or 5×5). The convolution operation involves sliding the filters over the input image, computing the element-wise multiplication between the filter weights and the corresponding pixels in the receptive field, and summing up the results. This produces a set of feature maps. The output of a single convolutional layer can be computed as follows:

$$\begin{aligned} output[i, j, k] \\ = activation(\text{sum}(\text{input_patch} * \text{filter}[k]) \\ + bias[k]) \end{aligned} \quad (3.2)$$

Here, $output[i, j, k]$ represents the value of the k -th feature map at position (i, j) in the output, $input_patch$ is the receptive field from the input image corresponding to the filter position, $filter[k]$ represents the k -th filter, $bias[k]$ is the bias term for the k -th feature map, and $activation$ is the activation function applied element-wise to the summed result.

- **Pooling Layer:** The pooling layer reduces the spatial dimensions of the feature maps, aiming to capture the most salient information. Common pooling operations include max pooling and average pooling. A pooling operation with a pool size of P and stride of S can be defined as follows:

$$\begin{aligned} output[i, j, k] \\ = pool_function(\text{input}[i * S : i * S + P, j * S : j * S \\ + P, k]) \end{aligned} \quad (3.3)$$

Here, $output[i, j, k]$ represents the value of the k -th pooled feature at position (i, j) in the output, $input[i * S : i * S + P, j * S : j * S + P, k]$ represents the pooling region from the k -th feature map, and $pool_function$ is the pooling function applied to the pooling region.

- **Fully Connected Layers:** After the convolutional and pooling layers, the resulting feature maps are often flattened into a 1-dimensional vector. This vector is then fed into one or more fully connected layers, which perform high-level feature extraction and map the learned features to the desired output classes or predictions. The fully connected layers can be represented as:

$$output = activation(\text{dot_product}(\text{input}, \text{weights}) + bias) \quad (3.4)$$

Here, $input$ represents the flattened feature vector, $weights$ represent the weight matrix connecting the input to the fully connected layer, $bias$ represents the bias term, $dot_product$ denotes the dot product operation, and $activation$ is the activation function applied element-wise to the summed result.

- Output: The output of the last fully connected layer represents the predicted class probabilities or regression values, depending on the task being performed.

In practice, CNNs often have multiple convolutional layers with different filter sizes and strides, non-linear activation functions, regularization techniques, and complex architectures, such as residual connections or attention mechanisms, to improve performance on various image-related tasks.

- RNNs are designed to handle sequential and temporal data. They utilize recurrent connections to capture dependencies between elements in a sequence. RNNs are commonly employed in tasks such as natural language processing, speech recognition, and time series analysis. Also, there are some other DL algorithms such as Generative Adversarial Networks (GANs), autoencoders, transformers.
- LSTMs are a type of RNN that mitigate the vanishing gradient problem and can retain information over long sequences. LSTMs are particularly effective in modelling and generating sequential data and have been successful in tasks like speech recognition, language translation, and handwriting recognition.
- GANs consist of a generator network and a discriminator network that compete against each other. The generator aims to generate realistic samples, while the discriminator aims to distinguish between real and generated samples. GANs have been widely used for tasks such as image synthesis, style transfer, and data augmentation.
- Autoencoders are unsupervised learning models that aim to reconstruct their input data. They consist of an encoder network that maps the input data to a lower-dimensional latent space and a decoder network that reconstructs the input from the latent representation. Autoencoders are used for tasks such as dimensionality reduction, anomaly detection, and denoising.
- Transformers have gained prominence in natural language processing tasks. They utilize self-attention mechanisms to capture global dependencies and learn contextual representations of words or tokens. Transformers have demonstrated state-of-the-art performance in tasks like machine translation, text summarization, and language modelling.

Overall, the choice of technique depends on the nature of the problem, available data, computational resources, and desired results. Machine learning techniques continue to evolve and advance, enabling computers to learn and make predictions in increasingly complex situations.

The following Table 7 informs about key phrases in ML and their respective meanings.

Table 7 Key phrases in ML with significances

Phrases	Significance
Model	Trained by employing ML algorithm to produce outputs
Algorithm	Bunch of rules along with computational techniques to gain profound details
Training Data	Consist of features, patterns, and key trends
Validation Data	To evaluate model performance during training

Testing Data	To assess the accuracy of the trained model
Predictor Variable	A data trait to predict the outcome
Response Variable	A trait of the output variable and Predictor Variable should envisage it

To implement the ML technique illustrated in Figure 8, one needs to follow these steps:

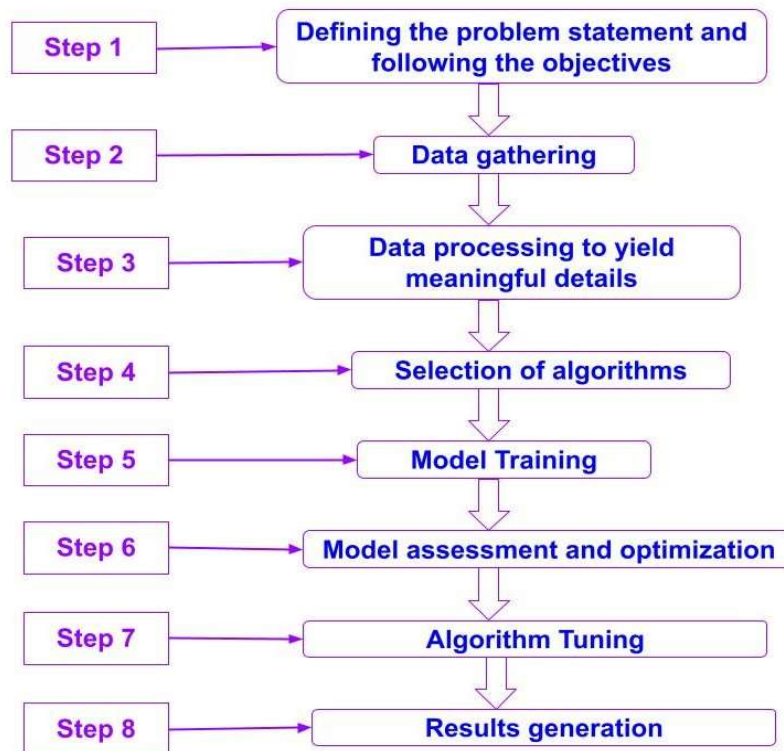


Figure 8 Crucial steps to implement the ML technique

3.2 Modern Approaches to Computer Vision Techniques

Computer vision techniques that are intelligent algorithms to extract deep feature details from images and videos. The primary goal of computer vision is to enable machines to analyse and interpret visual data, recognize objects, understand scenes, and extract relevant information. This encompasses a wide range of tasks, including image classification, object detection, image segmentation, pose estimation, image generation, and video analysis.

3.2.1 ML and DL in Computer Vision

The computer vision field has its own traditional algorithms and a large area of it is untouched by AI techniques. Conventional algorithms in computer vision may provide acceptable results for low imagery data but these algorithms may not perform well with large datasets i.e., produce saturated results. At this point, using artificial intelligence (AI), i.e., machine learning and deep learning techniques with computer vision provide excellent results with large datasets and also enhance performance properties. Figure 9(a) shows a systematic approach of ML with computer vision while a deep neural network approach is considered in Figure 9(b). In advanced ML techniques, both feature extraction and learning are automated. Deep learning has had a significant impact on the field of computer vision. Convolutional Neural Networks (CNNs) have become the backbone of many computer vision systems, enabling highly accurate

image recognition and object detection. CNNs learn hierarchical representations of visual data by stacking multiple convolutional layers, which capture increasingly complex features. Overall, it plays a crucial role in enabling machines to perceive and understand the visual world, bridging the gap between humans and machines in terms of visual understanding and interpretation.

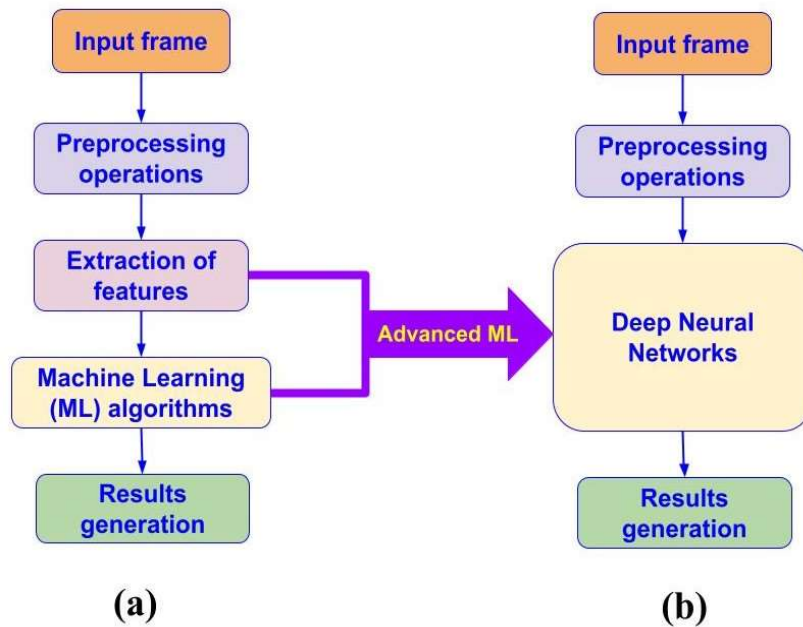


Figure 9 Steps in applying ML techniques to computer vision tasks

3.2.2 Object Detection

Object detection techniques of computer vision detect the occurrence of objects in an image or video with bounding boxes and identify their classes. It has two method types such as single-stage which works for inference speed and real-time use and two-stage which works for model performance i.e., detection accuracy [55]. The single-stage detectors remove the process of region of interest (ROI) extraction and moves for classification and regression whereas two-stage detectors extract ROI and then apply classification and regression. Classification and localization accuracy and inference speed are two important metrics for object detectors.

Object detection techniques have advanced significantly with the rise of deep learning and convolutional neural networks (CNNs). Here is a high-level overview of the typical process involved in object detection:

- **Input Image:** The object detection algorithm takes an image or a video frame as input.
- **Feature Extraction:** A CNN is employed to extract features from the input image. This is typically done by passing the image through multiple convolutional and pooling layers to generate a feature map. CNN learns hierarchical representations that capture visual patterns and discriminative features from the input data.
- **Region Proposal:** The feature map is used to generate a set of potential object locations, often referred to as region proposals. This step helps narrow down the search space and improve efficiency. Various methods are used for region proposal generation, such as selective search, region proposal networks (RPNs), or anchor-based approaches.

- **Classification:** The extracted features are then used to classify each region into specific object classes or background. This is typically done using classifiers, such as support vector machines (SVMs) or softmax classifiers, which are trained on labelled data to recognize different object categories.
- **Localization:** In addition to classifying objects, the algorithm also localizes them by predicting the bounding boxes that tightly enclose the detected objects. This can be done using regression techniques, where the algorithm learns to estimate the coordinates of the bounding box corners.
- **Post-processing:** To refine the object detections, post-processing steps are performed. These steps may involve filtering out overlapping or low-confidence detections, applying non-maximum suppression to keep the most confident detections, or incorporating contextual information to improve accuracy.

The deep learning architectures such as Faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector), have greatly improved object detection systems. These models have achieved impressive performance on a wide range of object detection tasks and are widely used in applications like autonomous driving, surveillance, object recognition, augmented reality, maintenance practices, etc. Overall, object detection plays a vital role in many computer vision applications, enabling machines to understand and interact with visual data by detecting and localizing objects of interest within images or videos.

The following terms and equations are essential for evaluating the performance of object detection models. They provide insights into the model's ability to detect objects accurately and balance precision and recall trade-offs.

- **True Positive (TP):** The model correctly predicts the presence of an object when it actually exists in the image.
- **True Negative (TN):** The model correctly predicts the absence of an object when there is no object in the image.
- **False Positive (FP):** The model incorrectly predicts the presence of an object when there is no object in the image (false alarm).
- **False Negative (FN):** The model incorrectly predicts the absence of an object when an object is present in the image (missed detection).
- **Accuracy:** The proportion of correctly classified objects (both positives and negatives) to the total number of predictions made by the model.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (3.5)$$

- **Precision:** The proportion of correctly classified positive predictions (TP) to the total number of positive predictions made by the model.

$$Precision = TP / (TP + FP) \quad (3.6)$$

- **Recall (also known as Sensitivity or True Positive Rate):** The proportion of correctly classified positive predictions (TP) to the total number of actual positive instances in the dataset.

$$Recall = TP / (TP + FN) \quad (3.7)$$

- Average Precision (AP): A measure of how well the model ranks the predicted bounding boxes for different object classes. It is calculated by computing the precision-recall curve for each class and then computing the average precision.

$$AP = \frac{\sum(\text{Precision at each recall point})}{\text{Number of recall points}} \quad (3.8)$$

- Mean Average Precision (mAP): The average of the AP values across all object classes in the dataset. It is commonly used as an evaluation metric for object detection models.

$$mAP = \frac{\sum(AP \text{ for each class})}{\text{Number of classes}} \quad (3.9)$$

- F1 score: The harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall.

$$F1 \text{ score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (3.10)$$

- Precision-Recall (PR) Curve: A graph that represents the trade-off between precision and recall for different classification thresholds. The x-axis represents the recall, and the y-axis represents the precision. The curve shows how precision changes as the recall threshold varies.
- Intersection over Union (IOU): A measure of overlap between the predicted bounding box and the ground truth bounding box. It is commonly used to evaluate the accuracy of object detection algorithms.

$$IOU = \frac{\text{Area of Intersection}}{\text{Area of Union}} \quad (3.11)$$

3.2.3 Embedded Vision Approach

Embedded vision systems leverage computer vision algorithms to analyse visual data captured by embedded cameras or sensors. These algorithms process and interpret the visual information to extract meaningful insights, make decisions, or trigger actions [56], [57]. Common tasks in embedded vision include object detection, recognition, tracking, image segmentation, and scene understanding. Here, the term embedded refers to an embedded system which is any microprocessor-based computing hardware system and vision refers to computer vision techniques. One of the key challenges in embedded vision is the limited computational resources and power constraints of embedded devices. To overcome these limitations, specialized hardware accelerators, such as GPUs (Graphics Processing Units), FPGAs (Field-Programmable Gate Arrays), and dedicated vision processing units (VPUs), are often used to perform computationally intensive tasks efficiently. These hardware accelerators enable real-time processing of visual data on resource-constrained devices. Embedded vision finds applications in various domains, including autonomous vehicles, robotics, smart surveillance, augmented reality, healthcare monitoring, and industrial automation. Researchers and engineers in the field of embedded vision continuously develop novel algorithms, architectures, and optimization techniques to improve the efficiency and accuracy of visual processing on embedded devices. This includes advancements in deep learning models, compression techniques, and real-time processing algorithms tailored for embedded systems.

3.3 Selected Models for Methodical Approach

Here, two YOLO models selected as YOLOX and YOLOv5 for methodical approach are described below in detail.

- YOLOX – It is a single-stage detection model which functioned well on multiple object detection benchmark datasets, including COCO, PASCAL VOC, and Open Images [58]. YOLOX comprises three crucial facets such as an anchor-free approach for precise bounding box detection, a decoupled head for efficient classification and regression tasks, and advanced label allocation tactics like SimOTA. The Darknet53 is a CNN and as a backbone which involves 1×1 convolutions, residual connections, and 3×3 convolutions as shown in Figure 11 [14]. The anchor-free design utilises a center-based approach for each pixel's detection mechanism. This approach selects a single positive instance per pixel and estimates four distances (left, top, right, bottom) from the positive instance to the image borders. As a result, YOLOX uses a single 4D vector to encode the location of the bounding box for every foreground pixel. The decoupled head enables better optimization and scalability by separating the two tasks. It also allows for the addition of multiple detection heads with varying feature scales, resulting in improved object detection across different object sizes. The head architecture includes a 1×1 convolutional layer that effectively reduces the channel dimension. It is then followed by two parallel branches, each consisting of two 3×3 convolutional layers as shown in Figure 10. SimOTA is a Simplified Optimal Transport Assignment, redesigned strategy for target assignment during training. It improves average precision without increasing training cost. It estimates the number of positive anchors for each ground truth based on IoU values, considering factors like size, scale, and occlusion. SimOTA reduces the number of iterations significantly (training time reduces), leading to improved performance i.e., enhancing the accuracy of the model. The loss function is computed for optimize the model for accurate class predictions, bounding box regression, and objectness scoring.

$$Loss = class_loss + reg_weight * (reg_loss + iou_loss) \quad (3.12)$$

Here, *class_loss* is the Binary Cross Entropy (BCE) loss between the predicted class probabilities and the ground truth class labels. The *reg_loss* is the regression loss, which is optimized using Generic Intersection over Union (GIoU) to measure the accuracy of bounding box predictions. *iou_loss* is the objectness loss, which uses BCE to optimize the objectness predictions based on the IoU values. The *reg_weight* parameter is a scaling factor that determines the relative importance of the regression loss compared to the other losses in the model.

It is a versatile detection framework that offers different version sizes to accommodate varying requirements. The YOLOX-nano has 0.91 million (M) parameters and performs well with a test image size of 416 pixels in both width and height. On the other hand, YOLOX-tiny utilizes 5.06 M parameters and is optimized for the same test image size. For more demanding tasks, YOLOX provides the YOLOX-small which was selected in our case for embedded vision purpose, YOLOX-medium, and YOLOX-large, which have 9 M, 25.3 M, and 54.2 M parameters respectively, and are designed to work with a test image size of 640. Lastly, the YOLOX-large version boasts 99.1 million parameters and is suitable for processing test image sizes of 640 or 800.

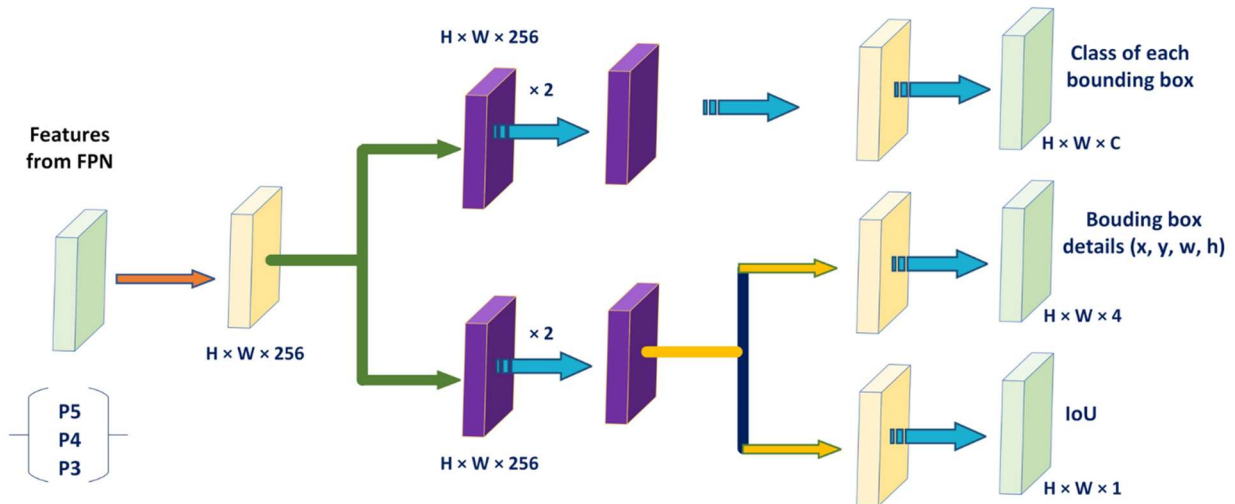


Figure 10 Illustration of YOLOX Decoupled Head

	Type	Filters	Size	Output	
	Convolutional	32	3 × 3	256 × 256	
	Convolutional	64	3 × 3 / 2	128 × 128	
1x	Convolutional	32	1 × 1	128 × 128	
	Convolutional	64	3 × 3		
	Residual				
2x	Convolutional	128	3 × 3 / 2	64 × 64	
	Convolutional	64	1 × 1		
	Convolutional	128	3 × 3		
8x	Residual			64 × 64	
	Convolutional	256	3 × 3 / 2		32 × 32
	Convolutional	128	1 × 1		
8x	Convolutional	256	3 × 3	32 × 32	
	Residual				16 × 16
	Convolutional	512	3 × 3 / 2		
8x	Convolutional	256	1 × 1	16 × 16	
	Convolutional	512	3 × 3		8 × 8
	Residual				
4x	Convolutional	1024	3 × 3 / 2	8 × 8	
	Convolutional	512	1 × 1		8 × 8
	Convolutional	1024	3 × 3		
	Residual				
	Avgpool		Global		
	Connected		1000		
	Softmax				

Figure 11 Detailing of DarkNet-53 CNN

- YOLOv5 – It is based on PyTorch framework, having .yaml configuration file and targets on a simplified architecture, model scaling, and transfer learning for various object detection tasks. The architecture comprises CSP Darknet-53 backbone to extract essential features from input images. It is a modified version of the Darknet-53, incorporating Cross Stage Partial (CSP) connections to improve information flow and feature representation. A neck employs Path Aggregation Network (PAN) to create feature pyramids for effective object scaling and generalization. A head design is same as that of YOLOv3 and v4 and is responsible for the final detection step, using anchor boxes to generate output vectors with

class probabilities, objectness scores, and bounding boxes (center_x, center_y, height, width) [59]. To update the model parameters during training, loss is computed as follows.

$$\begin{aligned} Loss &= BCE(classes) + BCE(objectness) + CIoU(location) \\ &= \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \end{aligned} \quad (3.13)$$

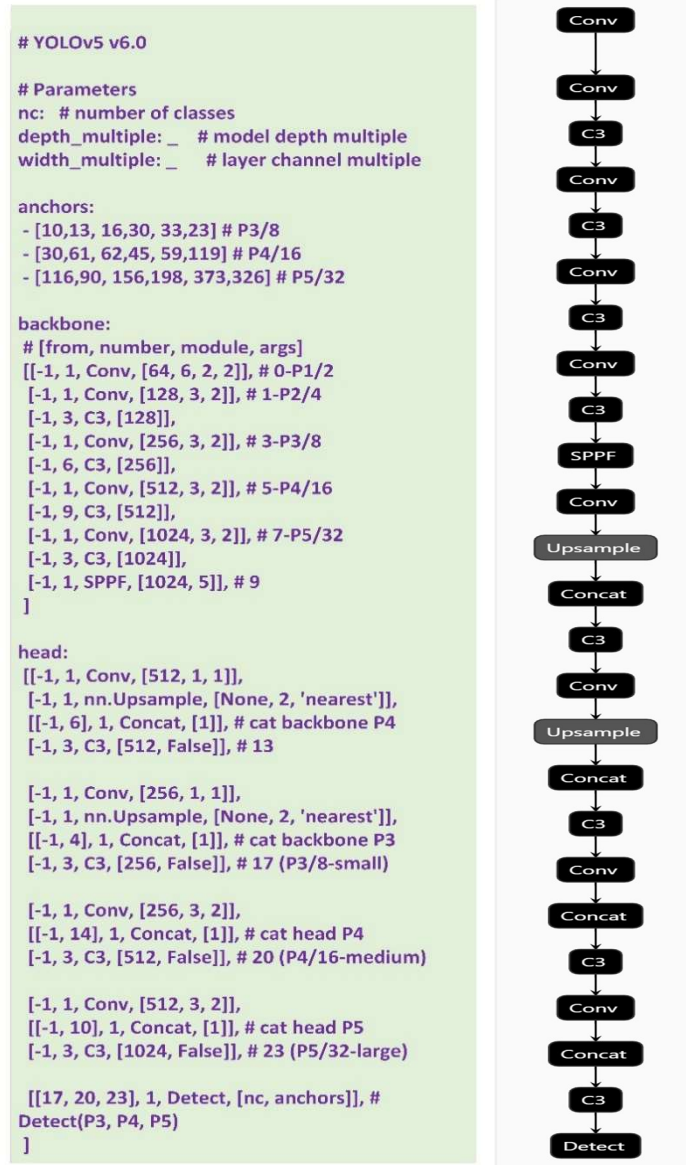


Figure 12 YOLOv5 Arithmetical Details

The arithmetical details in the model architecture have been given in Figure 12. In above equation, BCE (classes) represents the Binary Cross Entropy loss for the predicted classes, BCE (objectness) represents the Binary Cross Entropy loss for the objectness scores, and CIoU (location) represents the Complete Intersection over Union loss for the bounding box locations. It uses autoanchor to automatically verify and generate the anchor boxes based on the distribution of bounding boxes in the custom dataset with K-means clustering and genetic learning algorithm. This ensures better alignment between the model and the objects it needs to detect. Activation functions such as SiLU, or the Sigmoid Linear Unit

also known as swish, combines the sigmoid and linear functions to capture complex features in hidden layers.

$$silu(x) = x * \sigma(x) \quad (3.14)$$

Here, $\sigma(x)$ is the logistic sigmoid.

Its powerful gradients enable faster and more stable training. The sigmoid activation function is used in the output layer for binary classification tasks. It comprises different versions of sizes (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x) to accommodate different resource constraints and performance requirements. YOLOv5n is a lightweight architecture for edge devices, weighing less than 2.5MB in INT8 format and 4MB in FP32 format. YOLOv5s is a small version optimized for CPU inference as selected in our case for mobile deployment, while YOLOv5m strikes a balance between speed and accuracy with 21.2 M parameters. YOLOv5l is designed for detecting smaller objects, featuring 46.5 M parameters. Finally, YOLOv5x is the largest version, offering the highest mean average precision (mAP) but with 86.7 M parameters and slower inference speed.

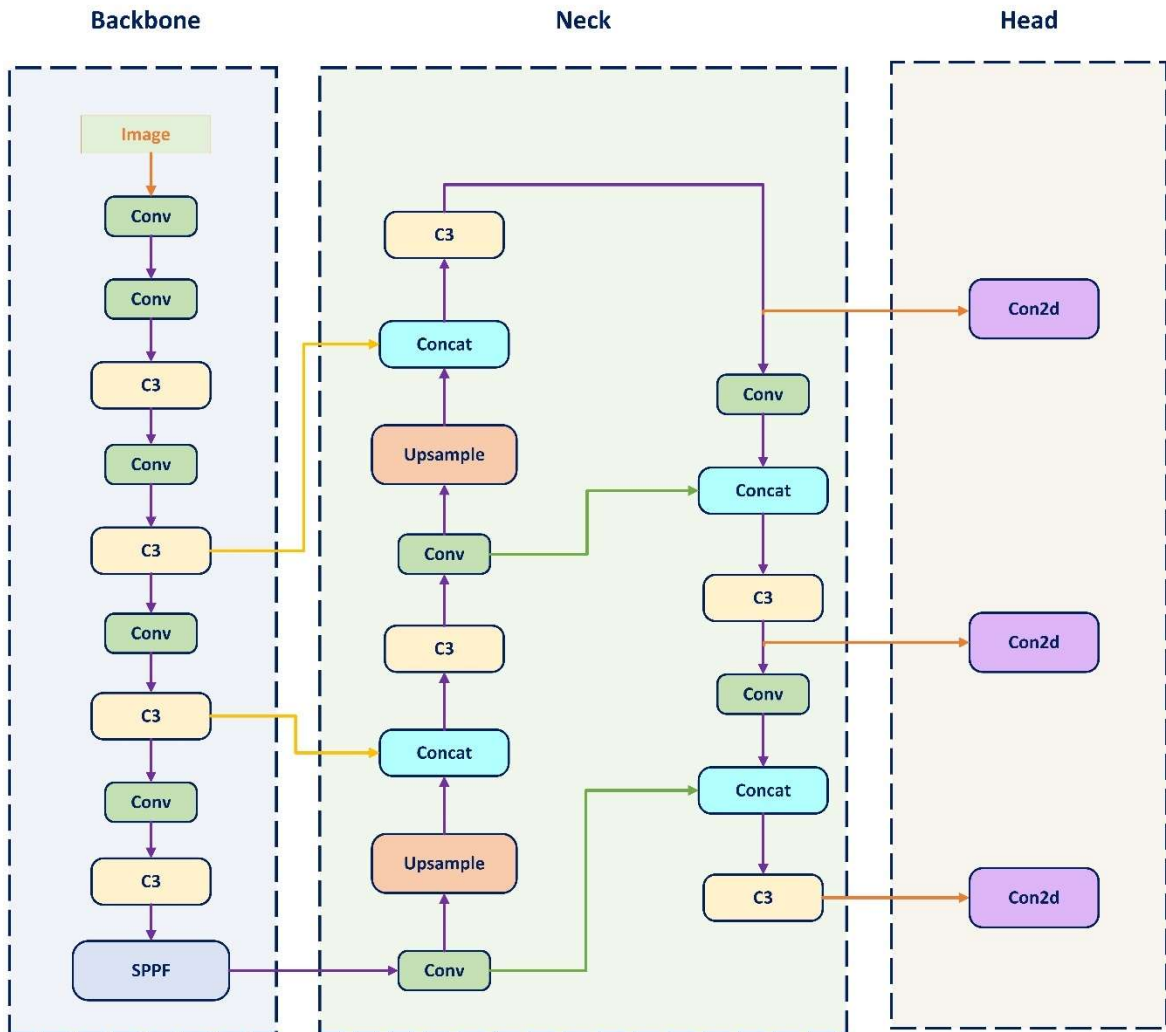


Figure 13 YOLOv5 Architectural Details

The architecture illustrated in both Figure 13 employs a combination of Convolutional (Conv) and C3 layers within the backbone to extract relevant features from input images. These features are subsequently merged at different hierarchical levels utilizing Conv, Upsample, Concat, and C3 layers within the head of the model. Facilitating the object detection process is a dedicated Detect layer, which utilizes anchor boxes and the designated class count for accurate identification. Notably, the C3 (CSP-3) blocks within the architecture consist of two parallel convolutional layers each. The first layer compresses input features through a bottleneck, while the second layer directly produces features. The resultant feature streams are concatenated, further processed through pooling and convolutional layers, and benefit from skip connections and attention mechanisms present in the C3 blocks to enhance information flow and diminish the impact of noise. This comprehensive architecture focuses on precise object detection across varying scales present within the input image.

3.4 Transfer learning and Fine-tuning

Transfer learning and fine-tuning are both essential techniques in the field of AI, particularly in deep learning. Here is an explanation of their prominence in the AI:

- Transfer Learning - It refers to the process of leveraging knowledge gained from one task or domain and applying it to another related task or domain. It involves using a pre-trained model that has been trained on a large-scale dataset and reusing its learned representations or features for a new task. The pre-trained model serves as a starting point, and its knowledge is transferred to the target task, typically by using the pre-trained model as a feature extractor or initializing the weights of a new model. It is very important for the following reasons:
 - a) Data efficiency - Transfer learning enables models to learn from smaller labelled datasets by leveraging the knowledge learned from large-scale datasets. This is particularly useful when labelled data is scarce or expensive to acquire. By using transfer learning, models can effectively extract relevant features from limited data, preventing overfitting and improving generalization.
 - b) Improved performance - Pre-trained models, such as those trained on large-scale datasets like ImageNet, have learned general features that are useful across various tasks. By utilizing these pre-trained models as a starting point, transfer learning allows the model to benefit from the previously learned representations, resulting in improved performance on the target task.
 - c) Reduced training time - Training deep neural networks from scratch on large-scale datasets can be computationally expensive and time-consuming. Transfer learning reduces training time significantly by utilizing pre-trained models as initial starting points. Instead of training the entire model, only specific layers or parts of the model are fine-tuned on the target task, accelerating the training process.
 - d) Domain adaptation - Transfer learning is beneficial when the source and target domains have different characteristics. By transferring knowledge from a source domain to a target domain, models can adapt to new data distributions, bridging the gap between the two domains and improving generalization performance.
- Fine-tuning - It is a specific step in the transfer learning process. Once the pre-trained model is utilized as a feature extractor or initialized, fine-tuning involves further training the model on the target task-specific dataset. During fine-tuning, the parameters of the pre-

trained model are updated by backpropagating gradients through the added task-specific layers. This process allows the model to adapt and optimize its performance for the target task. Fine-tuning is important for the following reasons:

- a) Task-specific adaptation - Fine-tuning allows the model to adapt to the intricacies and specific requirements of the target task. By updating the pre-trained model's parameters, it can learn task-specific patterns and optimize its performance on the specific problem.
- b) Model customization - Fine-tuning allows practitioners to customize and tailor pre-trained models to suit their specific needs. By modifying or extending the architecture of the pre-trained model and fine-tuning it on the target task, researchers and developers can create models that are optimized for their specific application, thereby improving performance and efficiency.
- c) Preserving learned representations - While fine-tuning task-specific layers, the pre-trained model's early layers, often referred to as feature extractors, are typically kept frozen. This ensures that the valuable general features learned from the source domain are preserved and effectively utilized during training. Fine-tuning only modifies the parameters of the later layers, which are more specialized for the target task.

Yosinski et al. [60] investigated the transferability of features learned in deep neural networks (DNNs) and their effectiveness in transfer learning. They found that initial DNN layers learn more generic and transferable features, while deeper layers become task-specific. Similarity between the source and target tasks influences feature transferability, with better transfer observed for similar visual or semantic concepts. The size of the target task dataset and the capacity of the pre-trained model also impact transfer learning performance. *Oquab et al.* [61] proposed a method using mid-level representations in CNNs for transfer learning across tasks with different label spaces. Their approach involves pre-training on a large-scale dataset for the source task and fine-tuning on the target task with a smaller labelled dataset. *Long et al.* [62] introduced Deep Adaptation Networks (DANs) for transfer learning in computer vision tasks, achieving improved performance by aligning features from the source and target domains. *He et al.* [63] presented ResNet architecture, emphasizing the importance of transfer learning and fine-tuning in image recognition tasks. Their approach, with pre-trained weights and fine-tuning, outperformed traditional networks on various benchmarks such as ImageNet, CIFAR-10, and COCO.

In essence, transfer learning sets up the initial knowledge transfer, and fine-tuning fine-tunes the model to fit the target task by updating its parameters. Together, these techniques can significantly improve model performance for accuracy, data efficiency, reduction in training time, and facilitate adaptation to new domains or tasks. Their importance is evident across diverse applications, including computer vision, natural language processing, recommendation systems, and speech recognition.

3.5 Role of Artificial Learning in understanding physical mechanisms and developing predictive models in Different Research Domains

Artificial learning involves training algorithms on large datasets to recognize patterns, relationships, and structures within the data. In the context of understanding physical mechanisms, artificial learning can be applied in various scientific and engineering domains as given below.

- **Engineering and Manufacturing:** Artificial learning facilitates the development of predictive models for engineering applications. It can be used to analyse sensor data, monitor equipment performance, optimize processes, and predict failure or maintenance needs. By learning from historical data and real-time measurements, machines can provide valuable insights for improving efficiency, quality, and safety in manufacturing and engineering domains.
- **Physics and Natural Sciences:** In physics, artificial learning techniques are employed to understand complex physical systems, such as quantum mechanics, particle physics, and astrophysics. By training models on experimental data or simulations, researchers can uncover hidden patterns and relationships, enabling a deeper understanding of fundamental physical processes. It helps to model the behaviour of particles, understand quantum phenomena, or predict the properties of materials.
- **Biology:** Artificial learning can assist in deciphering genetic data, analysing protein structures, understanding biological processes, and predicting drug interactions. It can help identify disease patterns, classify different cell types, or optimize drug discovery processes.
- **Environmental Science:** Artificial learning can be employed to model and predict climate patterns, analyse satellite imagery for land cover classification, or assess the impact of pollution on ecosystems. It aids in understanding complex environmental interactions and developing more accurate predictive models.

Overall, artificial learning plays a pivotal role in understanding physical mechanisms and developing predictive models across various domains. It enables machines to learn from data, discover patterns, and make accurate predictions or decisions. By harnessing the power of artificial learning, researchers and practitioners can gain valuable insights, optimize processes, and make informed choices in diverse real-world applications.

I. Demir et al. [64], introduced the DeepGlobe challenge dataset which consists of high-resolution satellite images covering various regions of the Earth. It includes labelled ground truth data for tasks such as land cover classification, road extraction, and building delineation. The dataset enables participants to develop and evaluate their deep learning models on real-world scenarios. This work reveals the potential of AI in analysing Earth's satellite images for a wide range of applications. *Maziar Raissi et al.* [65], instituted physics-informed neural networks (PINNs), a framework that combines physics-based models with neural networks to solve forward and inverse problems involving nonlinear partial differential equations (PDEs). It demonstrates how artificial learning can be leveraged to learn the underlying physical mechanisms and make predictions based on limited or noisy data. *Feng et al.* [66], presented a methodology for structural damage detection using deep CNNs and transfer learning. The collection images of different types and degrees of structural damage were done for dataset development. They applied pre-processing and annotation operations on the images, labelled them as either damaged or undamaged. The CNN model was trained using this dataset to learn the patterns and features associated with structural damage. This work contributes to the advancement of AI-based approaches for structural health monitoring and maintenance in the field of civil engineering. *Biamonte et al.* [67], explored the intersection of quantum computing and machine learning. They discussed the use of quantum algorithms, such as quantum support vector machines, quantum clustering, and quantum neural networks, to tackle various machine learning tasks. They described how these algorithms can leverage quantum properties, such as superposition and entanglement, to perform computations in parallel and potentially provide

speedup over classical counterparts. *Moen et al.* [68], presented an approach for the development and implementation of several deep learning i.e., CNNs models tailored for different cellular image analysis tasks. They discuss the architecture and training procedures of these models, which include strategies such as data augmentation, transfer learning, and assembling. The models had been trained on large-scale datasets, providing a diverse range of microscopy cellular image examples. It also highlights future directions and opportunities for integrating deep learning with other imaging techniques and multi-modal data analysis. *Butler et al.* [69] considered the application of machine learning techniques such as support vector machines, neural networks, and random forests for the prediction of material properties, identification of novel materials, and designing of molecules with specific functionalities. It highlights the use of various data types, such as crystal structures, molecular fingerprints, and experimental measurements, to train machine learning models. *Hino et al.* [70], provided details about specific machine learning models in decision-making for sustainable environmental management, which includes air quality prediction, water quality assessment, species identification, weather forecasting, climate change modelling, and other environmental parameters. These algorithms are capable to analyse large amounts of environmental data collected from sensors, satellite imagery, and other sources. *Florian Shroff et al.* [71], presented FaceNet, a deep learning model that learns compact representations of face images, known as face embeddings, and maps them into a multidimensional space, where similar faces are close to each other for further face recognition and clustering tasks. They also proposed a triplet loss function that encourages the network to learn embeddings with small intra-class variance and large inter-class variance, enabling accurate and robust face recognition. It has a wide application area in advanced face recognition and verification systems, biometric authentication, and surveillance applications.

The general equation that represents the fundamental concept of AI can be given as

$$Y = f(X, \theta) \quad (3.15)$$

Where,

X - indicates the input data or features given to the AI algorithm

Y - denotes the output or prediction generated by the AI algorithm

θ - signifies the parameters or weights of the AI model

f - stands for the function or algorithm that maps the input data to the output predictions.

The equation signifies that the output Y is a function of the input X and the model parameters θ . The function f represents the learning algorithm or model architecture that transforms the input data using the learned parameters to produce the desired output. Here, the function f will vary depending on the AI technique being used such as deep learning i.e., complex neural network architecture with multiple layers and activation functions, support vector machines, decision trees, etc.

3.6 Significant breakthroughs in AI

Many experiments have been done since the inception of AI to its modern state. Some of the impressive advances are listed as follows.

Vapnik presented a comprehensive and rigorous treatment of the theoretical underpinnings of statistical learning [72]. It highlights the importance of understanding the generalization

properties of learning algorithms and presents key concepts such as empirical risk, true risk, VC dimension, and structural risk minimization. The support vector machines (SVMs), a powerful learning algorithm introduced by Vapnik, and his colleagues is based on the principle of finding an optimal hyperplane that separates the data into different classes with the maximum margin. This paper has had a significant impact on the development of machine learning algorithms and has contributed to advancing the field of AI.

LeCun et al. introduced the use of convolutional neural networks for document recognition tasks and demonstrated their effectiveness on the MNIST dataset, which is a widely used benchmark dataset in the field of machine learning and consists of a large number of grayscale images of handwritten digits [73]. The presented CNN architecture i.e., LeNet-5, included multiple layers of convolutional filters, pooling layers for subsampling, and fully connected layers for classification which has the ability to automatically learn features and capture spatial hierarchies present in the input images. It played a pivotal role in advancing the field of computer vision and contributed to the broader adoption of deep learning techniques in AI research.

Hinton and Salakhutdinov demonstrated the effectiveness of unsupervised pretraining and deep belief networks on several benchmark datasets, including handwritten digit recognition and object recognition [74]. It showed that by using unsupervised pretraining, deep neural networks could achieve better generalization performance, especially when the labelled training data was limited. Once the unsupervised pretraining was complete, the entire network was fine-tuned using supervised learning, such as backpropagation, to optimize it for the specific task at hand. The unsupervised pretraining served as an effective initialization step that helped the network escape local optima and facilitated faster convergence during the fine-tuning phase. It showed the potential of these techniques through empirical results and significantly influenced the field of deep learning.

The research work presented by *Alex Krizhevsky et al.*, is highly influential in the field of computer vision and marked a significant breakthrough in image classification using deep convolutional neural networks (CNNs) [75]. They proposed a deep CNN architecture called AlexNet and trained it on a large dataset of labelled images from the ImageNet database. AlexNet achieved a top-5 error rate of 15.3% in the ILSVRC 2012 competition, significantly outperforming other methods and surpassing human-level performance in image classification tasks.

Tsung-Yi Lin et al., provided the MS COCO dataset which serves as an important reference for researchers and practitioners in the computer vision community [76]. This work was published in the European Conference on Computer Vision (ECCV) in 2014. It provides an in-depth description of dataset, its creation process, annotations, and its significance as a benchmark for evaluating and advancing computer vision algorithms.

The "Attention is All You Need" paper has had a significant impact on the field of modern AI and NLP, contributing to the development of state-of-the-art models in machine translation, language generation, and other language-related tasks. The Transformer model introduced in this work has become the de facto standard for various NLP tasks, surpassing previous approaches in terms of performance and efficiency [77]. This novel architecture relies solely on attention mechanisms, without using recurrent neural networks (RNNs) or convolutional neural networks (CNNs) commonly used in sequence modelling. The self-attention mechanism

allows the model to capture dependencies between words or tokens in a sequence and enables the model to attend to various parts of the input sequence when generating each output token, making it more effective at capturing long-range dependencies compared to traditional sequential models.

Goodfellow et al. [78], introduced the concept of Generative Adversarial Networks (GANs), that consists of two neural networks, a generator, and a discriminator, which are trained in a competitive manner. The key idea of GANs is to generate synthetic data that is indistinguishable from real data by learning from a training dataset. The generator network takes random input noise and generates synthetic data samples, while the discriminator network tries to differentiate between real and generated data. The networks are trained in a two-player minimax game, where the generator aims to fool the discriminator, and the discriminator tries to correctly identify the real data from the generated data. Through this adversarial training process, the generator network gradually improves its ability to generate realistic data, while the discriminator network becomes more adept at distinguishing real from fake data. It has a tremendous impact on various domains, including image synthesis, text generation, and data augmentation. GANs have been used to generate realistic images, create deepfakes, enhance low-resolution images, etc.

Brundage et al., conferred a comprehensive exploration of the potential risks and challenges associated with the malicious use of AI. It raises awareness about the ethical and security implications of AI technologies and provides valuable insights into the forecasting, prevention, and mitigation of these risks [79]. The authors identify three primary areas where the malicious use of AI could have significant consequences: digital security, physical security, and political security. They explore various scenarios and potential applications where AI could be exploited for harmful purposes, such as automated hacking, social engineering, autonomous weapons, and AI-driven disinformation campaigns. The work serves as a foundation for further research and policy discussions regarding the responsible development and deployment of AI systems to ensure the beneficial use of this transformative technology.

3.7 Summary: Leading to the Methodical Approach

In this, the included sections highlight the significance of machine learning techniques, including supervised, unsupervised, and reinforcement learning, in AI and computer vision. Deep learning, specifically CNNs have greatly impacted computer vision tasks like image recognition and object detection. Object detection involves various steps such as feature extraction, region proposal generation, classification, and localization. Embedded vision systems leverage computer vision algorithms and specialized hardware accelerators to process visual data in resource-constrained environments. ML and DL techniques continue to advance the field of computer vision, enabling machines to effectively analyse and understand visual information.

The section specifically focuses on the selection of YOLOX and YOLOv5 models, describing their features and architectures. YOLOX employs an anchor-free approach, decoupled head, and advanced label allocation tactics, while YOLOv5 is based on the PyTorch framework and emphasizes a simplified architecture, model scaling, and transfer learning. It utilizes a CSP Darknet-53 backbone and a PAN neck for effective feature extraction and scaling. Both models employ different loss functions to optimize class predictions, bounding box regression, and objectness scoring.

Furthermore, the selection of these models was also based on their performance on benchmark datasets, their versatility in accommodating varying requirements, and their availability of different model sizes. The YOLOX model provides different sizes ranging from YOLOX-nano to YOLOX-large, while the YOLOv5 model offers sizes from YOLOv5n to YOLOv5x.

Additionally, the concepts of transfer learning and fine-tuning are introduced, highlighting their importance in improving data efficiency, performance, and adaptability. The relevance of these techniques is supported by research findings and their successful application in the domain.

The chosen methodology in the next chapter will delve deeper into the implementation and performance evaluation of AI algorithms, providing a comprehensive understanding of their application in computer vision tasks.

4 Methodology and Case Study with Results

The AI model for object detection is developed in this thesis. As already mentioned, the application field is chosen as detecting blockages in sewers.

4.1 Methodology

A comprehensive explanation of the research methodology employed in this study is provided in the following sections.

4.1.1 Development of New Critical Multiclass Representative Image Dataset

Figure 14 depicts the workflow involved in dataset decision-making, illustrating the comprehensive procedure from requirement generation to model training [4]. The subsequent subsections succinctly elaborate on the significance and necessity of developing a novel dataset, which is based on authentic facts, meticulous surveys, insightful observations, and thorough analysis.

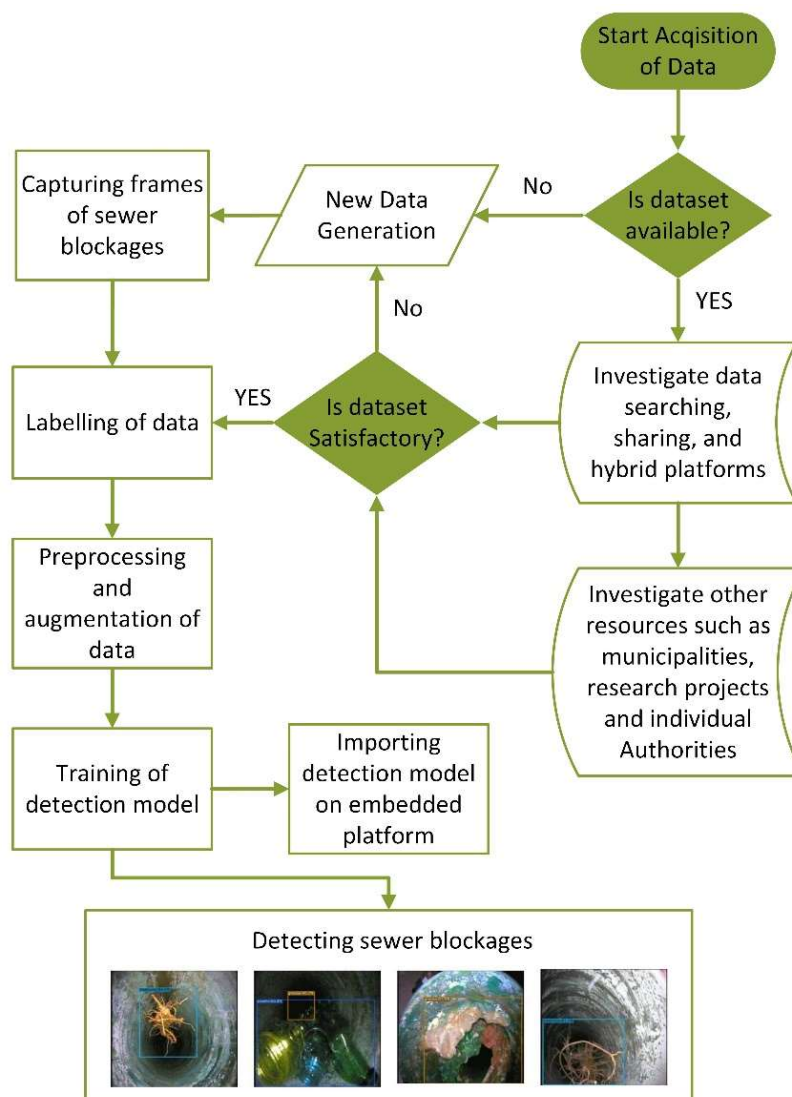


Figure 14 Decision-making workflow for the development of new dataset

In this, recognizing the indispensability of data, we embarked on a comprehensive investigation into its availability. We meticulously scrutinized various avenues, including data searching methodologies, sharing protocols, and hybrid platforms. Our efforts extended to engaging with diverse stakeholders such as authorities, municipal corporations, and the open research community. Despite these endeavors, certain constraints and confidentiality issues posed challenges in accessing secondary data. This scenario prompted us to navigate an alternative route, involving the creation of a primary dataset to attain further objectives.

4.1.1.1 Survey Details of Pune Municipal Corporation (PMC)

To gain a comprehensive understanding of the research landscape, a survey was conducted in Pune, India, a representative mid-size city in a developing country. The sewerage system in the city was designed in 1928, to accommodate a capacity of 31.8 million litres per day (MLD) to cater to a population of 0.26 million. However, as of 2020, the city's population has surged to 7.4 million, resulting in significant strain on the existing infrastructure. Presently, the city has 11 Sewage Treatment Plants (STPs) with a total capacity of treating 396 MLD.

Within SPRING project with the support of DYPatil, Engineering college and in collaboration with the Pune Municipal Corporation (PMC), a comprehensive city survey was conducted to evaluate the available sewage treatment techniques and identify associated challenges. The findings of the survey, along with insights from specific cases of cleaning works, are presented in Table 8 [39].

Based on information obtained through official sources, it was gathered that the primary objective of sewerage maintenance activities is to minimize drainage blockages per unit length. Generally, external mechanical systems are employed for cleaning purposes, incurring substantial costs. Although PMC endeavours to adhere to government directives for regular sewer inspection and maintenance, budgetary constraints have resulted in a lack of appropriate techniques and inadequate equipment for this purpose.

Table 8 Details of survey conducted at PMC. [39]

Terms	Details
Sewer Line	2167 kilometre
Sewer Pipe Diameter	Ranges from 100 mm to 1800 mm
Total Chambers (manhole)	2187
Sewer Pipe Material	<ul style="list-style-type: none"> • RCC • High-density polyethylene (HDPE) • bid-iron! • PVC
Distance Between Chambers	10 to 15 meters
Sewer Net pressure	1 to 4
Sewer Cleaning Techniques	<ul style="list-style-type: none"> • Suction Cum Jetting Machine with a Recycler • Suction Cum Jetting Machine • Jetting Machine
Total Generated Sewage	744 MLD

Intermediate pump stations (IPS)	6
Sewage Treatment Plants (STPs)	9
Main Sewer Lines	<ul style="list-style-type: none"> • River side • Below road • Canal side
Cleaning Tools	Charges/Shift (8 hours shift)
<ul style="list-style-type: none"> • Suction Cum Jetting Machine 	6400 INR
<ul style="list-style-type: none"> • Suction Cum Jetting Machine with a Recycler 	37000 INR
<ul style="list-style-type: none"> • Jetting Machine 	5360 INR

4.1.1.2 Why do we need to develop a New Representative Dataset?

A comprehensive review of the literature on computer vision applications and automated systems in sewer inspection work reveals that sewer blockages are difficult to detect. The existing algorithms and automated systems for real-time detection and cleaning of sewer blockages are found to be unreliable and lacking robustness. This problem of maintaining the sewers is further aggravated if there is a single sewer line for sewage and stormwater.

To address the real-time detection and identification of sewer blockages using AI model, it is essential to have a standardized dataset. Despite extensive efforts to gather relevant data from open literature and reaching out to various authorities and municipalities, no suitable datasets for real-time sewer blockage detection could be obtained. The noxious, unhygienic, and malodorous environment of sewers poses a significant hurdle in capturing images for dataset generation. It is worth pointing out that individual obligations, copyright, or confidentiality issues related to prior works are also accountable for the inaccessibility of datasets.

Clogging in drains is mainly initiated by the presence of grease, plastic, and tree roots as detailed in Table 9. However, there are additional components within sewage that mix with black water, making them challenging to identify. These components are generally considered as black sewer blockages and are represented as black grease in the dataset. Altogether, grease, plastic, and tree root imagery data have been considered as mentioned above in the representative dataset, named as S-BIRD (Sewer-Blockages Imagery Recognition Dataset), which is employed for learning of object detection models to detect and identify sewer barriers in real-time.

Figure 15 illustrates the concept of creating the S-BIRD dataset, which incorporates imagery data of grease, plastic, and tree roots [4].

The absence of a standardized matrix for implemented algorithms poses a significant challenge in practical development. However, the AI models trained on the S-BIRD dataset provide a valuable benchmark for assessing the localization results in real-time scenarios. By utilizing this dataset, researchers and developers can evaluate the performance of their implemented algorithms in real-world situations, making it a crucial resource in the field. This research case study aims to utilize new techniques in computer vision and AI technologies to optimize the

performance and effectiveness of sewer robotic systems by improving the efficiency of sewer blockage removal because blind systems may lack the same level of competence as vision-based automated systems, resulting in more efficient and reliable sewer maintenance processes.



Figure 15 S-BIRD dataset including major sewer blockages

Table 9 Common Causes and Consequences of Major Sewer Blockages

Obstruction Type	Causes	Impact and Issues
Treeroots	Tree roots infiltrating sewer lines	<ul style="list-style-type: none"> causes physical obstruction, leading to blockages. roots seek out moisture and can grow into pipes, causing cracks and blockages. lead to sewage backups and potential pipe damage.
Plastic	plastic waste such as single use plastic, transparent and multi-coloured bottles, containers, medical waste, bags, etc.	<ul style="list-style-type: none"> accumulation of plastic debris leads to gradual blockages. plastics become entangled with other debris, exacerbating blockages. contributes to sewage overflows and environmental issues.
Grease	accumulated grease and fat deposits such as cooking oil and fats, dairy-based fats, industrial grease, shortening, hydrogenated oil, etc.	<ul style="list-style-type: none"> grease solidify in pipes and accumulate over time, causing blockages. leads to reduced flow capacity and causes backups and overflows. attract other debris, further exacerbating the blockage.

4.1.2 Methodical Flow for New Dataset and Detection Model Training

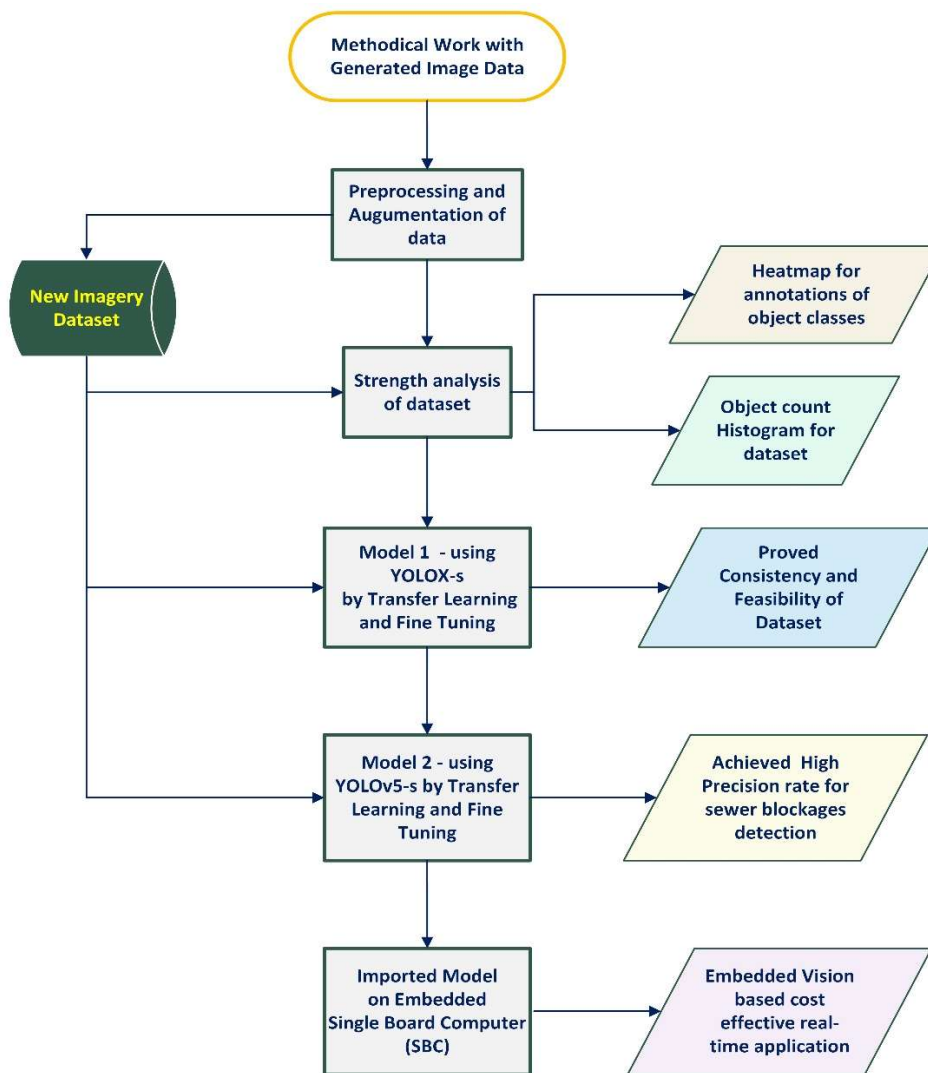


Figure 16 Methodical Workflow with newly developed dataset

Figure 16 presents a systematic workflow that outlines the key steps and techniques employed for development of sewer blockage detection models using the newly developed representative dataset and transfer learning. The workflow encompasses various essential techniques and their implications, highlighting the practical relevance of the research findings.

To begin, frames capturing sewer blockages were collected, and preprocessing and augmentation operations were performed to generate critical instances suitable for training purposes. Heatmap and object count histogram analyses were conducted to evaluate the strength of each object class within the newly developed dataset. This dataset, named the S-BIRD dataset, was specifically designed to identify common sewer blockages, and demonstrated its effectiveness in training robust detection models.

The application oriented model development process involves utilizing the YOLOX-s and YOLOv5-s architectures, incorporating transfer learning and fine-tuning techniques with optimized parameters and data augmentation. The evaluation of the developed models is done for the confirmation of their exceptional accuracy in detecting sewer blockages and demonstrated the consistency and feasibility of the employed dataset.

Furthermore, the methodical flow emphasizes the integration of embedded vision techniques for real-time applications, underscoring the practical implications of the research findings in wastewater management. Various tools and techniques, including OpenCV (Open Source Computer Vision and Machine Learning Library), Python programming, the PyTorch framework, machine learning libraries, high-performance Nvidia GPU workstations, Single Board Computers (SBC), and the Linux operating system, were utilized to facilitate the development of model training programs and the creation of a robust and efficient embedded vision platform.

By following this methodical flow, the research successfully develops detection models trained on the newly created representative primary dataset, for showcasing their accuracy and feasibility in identifying sewer blockages. Additionally, the integration of embedded vision techniques highlighted the practicality of the research findings, contributing to advancements in wastewater management.

4.2 Tools Utilized in S-BIRD Dataset Generation

Below, a comprehensive explanation with significance of the tools utilized in the development of the S-BIRD dataset is presented, which serves a crucial role in the practice.

4.2.1 Constructed Sewer Pipeline

The sewer network simulation and dataset generation work were done at laboratory of DY Patil School of Engineering, Pune one of the project partners in SPRING. A simulated sewer network was constructed using PVC pipelines with a diameter of 200 mm as shown in Figure 17 (a), similar which are used in residential sewers. The purpose of this simulated network was to mimic a real sewer environment while eliminating the noxious atmosphere and stench as illustrated in Figure 17 (b) (pure negative samples). The resulting sewer pipeline, as depicted in Figure 17 [4], closely resembled an actual sewer system.

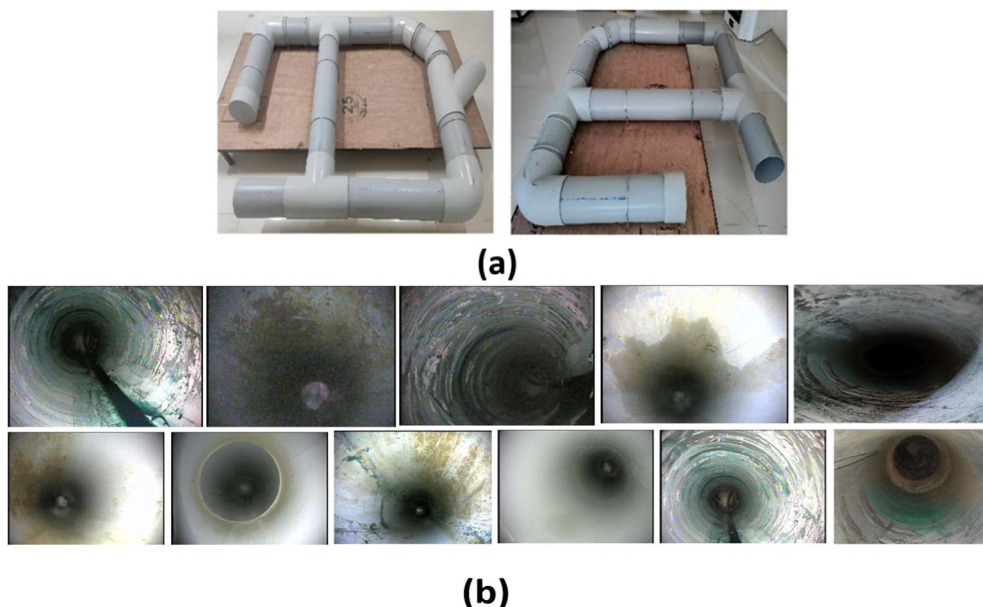


Figure 17 Illustration of the constructed sewer pipeline with (a) material and diameter details and (b) realistic design and internal environment

In order to ensure authenticity, the simulated sewer network replicated the various types of blockages such as tree roots, plastics, and grease, that naturally occur in real sewer systems, as

discussed in section 4.3. All relevant information and characteristics of these blockages were incorporated into the simulated network. This ensured that the detection model trained using the S-BIRD dataset would be capable of functioning effectively in practical situations.

4.2.2 Inspection Camera for Sewerage Systems

The watertight sewer camera, illustrated in Figure 18 [4], played a crucial role in capturing real-time frames of sewer barriers such as grease, plastics, and tree roots. This camera possessed specific features and characteristics, as outlined in Table 10.



Figure 18 Watertight sewer camera employed for frame capture

The dataset generation process incorporated a highly advanced sewer camera with a compact dimension of 23 mm × 120 mm. This camera was equipped with 12 modifiable white LEDs, enabling it to adapt to varying lighting conditions by adjusting the brightness levels. Its exceptional waterproofing grade of IP68 provided reliable protection against water infiltration, which is of utmost importance when operating in sewer environments. Furthermore, the camera boasted a wide vision angle of 140 degrees, facilitating comprehensive coverage during inspections.

Table 10 Technical details of the utilized sewer camera

Attributes	Specifications
Illumination source	12 adjustable white LEDs
Camera dimension	Camera dimension 23 mm × 120 mm
Vision angle (Field of view)	140 degrees
Waterproof grade	IP68

Utilizing this sophisticated sewer camera was instrumental in enhancing the S-BIRD dataset. It enabled the capture of real-time frames from diverse angles, allowing for the desired aspect ratio and accommodating different lighting conditions. Moreover, it accurately documented the various obstacles encountered within sewer systems. These captured frames serve as invaluable training data for the detection model, guaranteeing its efficacy when confronted with similar real-world scenarios.

4.3 Image Data Collection for the Development of Novel S-BIRD

The dataset of sewer blockages comprises a comprehensive collection of carefully captured images taken under diverse lighting conditions and from various angles within the simulated sewer network. These images offer essential insights and features required for detection and recognition tasks. Detailed descriptions of the captured blockage scenarios are provided in the following paragraphs.

Figure 19 [4] presents a selection of frames displaying blockages caused by tree roots, providing a glimpse into the diversity of occurrences encountered in the dataset. These images offer valuable insights into the presence and characteristics of tree root blockages within sewer pipes, contributing to the dataset's authenticity and relevance.

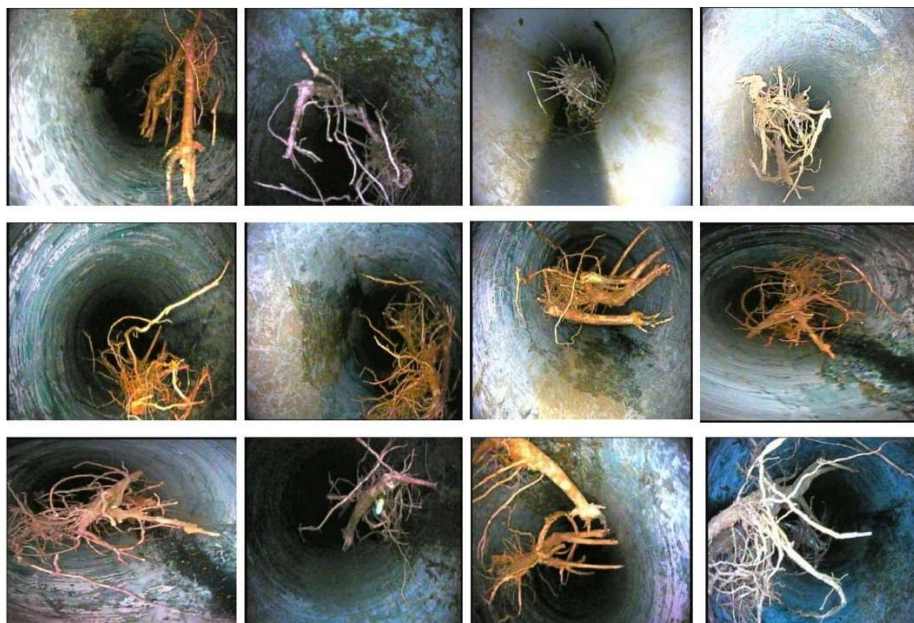


Figure 19 Frames depicting tree root blockages in the S-BIRD dataset



Figure 20 Frames illustrating plastic blockages in the S-BIRD dataset

In Figure 20 [4], the dataset captures images of blockages made up of different-coloured plastics, which are crucial for obtaining key information relevant to identification tasks. The inclusion of diverse plastic colours adds complexity to the dataset, enhancing its realism and practicality in training detection models.

Within the dark mass of sewage, additional elements such as plastic bags or debris may be present. However, due to their mixture with black water and grease, they often appear predominantly blackish in colour, posing challenges for visual identification. Nonetheless, the dataset effectively captures this characteristic, enriching the variety of blockage scenarios encountered in real-world sewer systems.

Figure 21 [4] displays frames depicting grease blockages, capturing a wide range of colours and diverse information. Grease blockages originate from various sources, including domestic households and both high- and low-density production plants that generate significant amounts of chemical and processed waste. The inclusion of such instances in the dataset enhances its authenticity and reflects the complexity of real sewer systems.

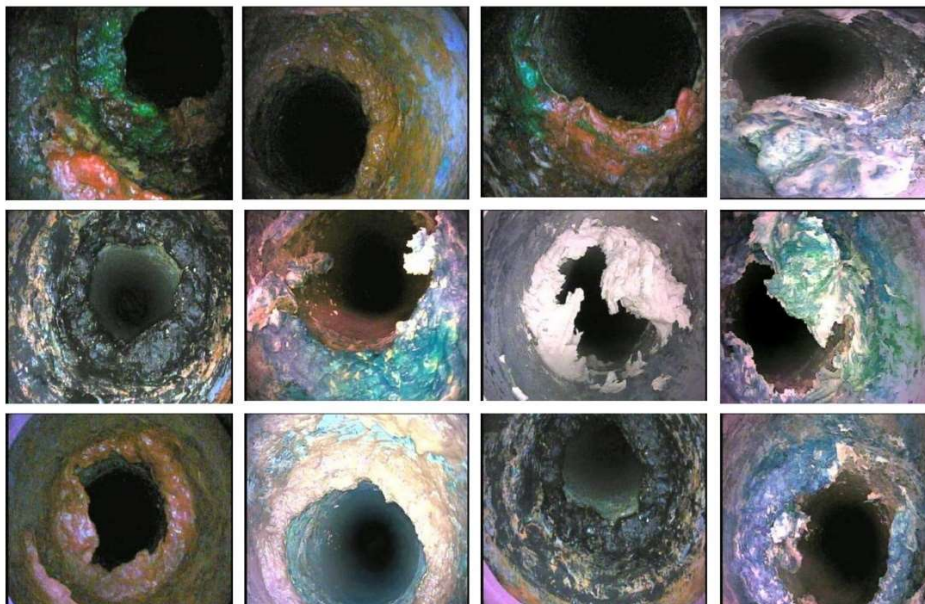


Figure 21 Frames displaying grease blockages in the S-BIRD dataset

Through meticulous collection and inclusion of these diverse blockage scenarios, the dataset provides a comprehensive and representative collection of images essential for training robust detection and recognition models in sewer systems.

4.4 Detailed Analysis of Captured Frames

The captured frames are accompanied by comprehensive arithmetic details, which are presented in Table 11 for further implementation. Annotating the objects in each captured frame required meticulous efforts, ensuring high skill and accuracy without any labelling errors. These annotations provide vital information regarding the location, specifically the center coordinates (center x, center y), width, height, and class of objects present in each frame of the S-BIRD dataset. This is essential information for subsequent computations and analysis. To ensure consistency and facilitate further computations, all these parameters are normalized

based on the original frame's width and height. Normalization is performed to ensure that the values range from 0 to 1, irrespective of the original image size.

Mathematically, the normalized parameters are computed as follows:

- Normalized Center Coordinates:

$$\begin{aligned} x_center_norm &= x_center / frame_width, \\ y_center_norm &= y_center / frame_height \end{aligned} \quad (4.1)$$

- Normalized Width and Height:

$$w_norm = w / frame_width, \quad h_norm = h / frame_height \quad (4.2)$$

For example, let's examine annotations for three classes: 0 for plastic, 1 for grease, and 2 for tree roots. For Class (0), representing plastic, the annotation includes the center coordinates (center x = 0.8389423076923077, center y = 0.25841346153846156), width (0.3173076923076923), and height (0.5168269230769231).

The annotated data provides valuable training examples for machine learning models, allowing them to learn and recognize objects of interest in images.

Table 11 Arithmetical details of captured frames.

Captured frames	Object Class (Sewer Blockage Type)	Acquired Frames
	Tree roots	2295
	Plastic	2392
	Grease	2353
Total frames	7040	
Annotations	10,233 (Average = 1.5 per frame)	
Average frame size	0.08 Megapixels	
Mean frame ratio	352 × 240 (wide)	
Angle of diagonal	0.598 radian = 34.3°	
Length of diagonal	426 pixels	
Aspect ratio Class	1.467:1	
Pixel density	9 pixels/mm or 230 pixels/inch	

To visualize the class balance in terms of annotations, Figure 22 displays the total number of annotations for each sewer blockage type: 4131 for grease, 3471 for tree roots, and 2631 for plastic.

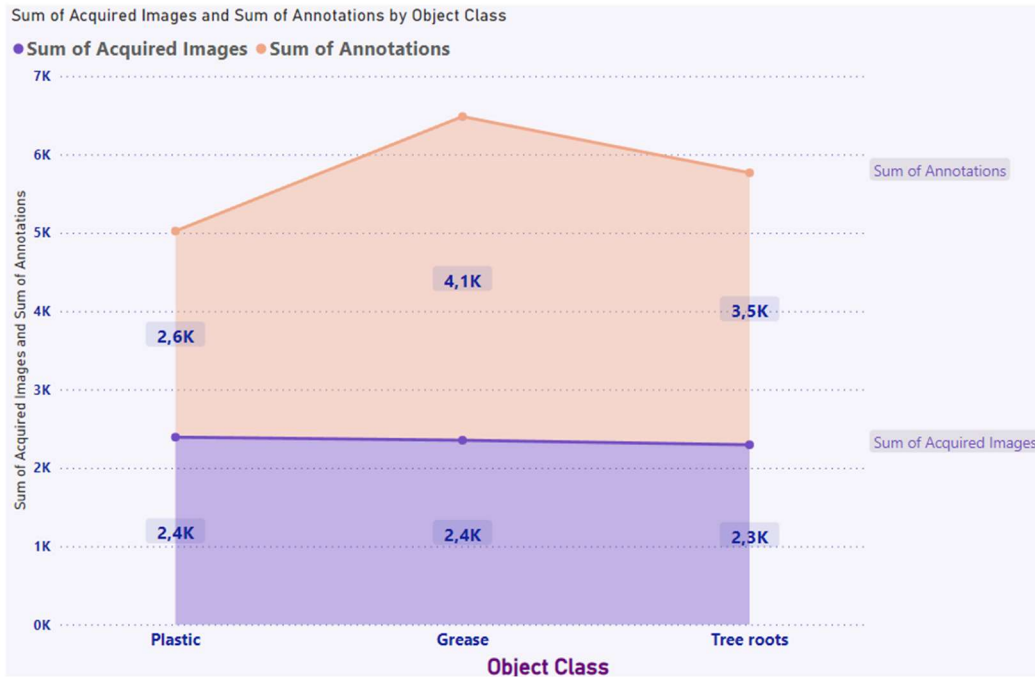


Figure 22 Annotated illustrations depicting the balance of sewer blockage types

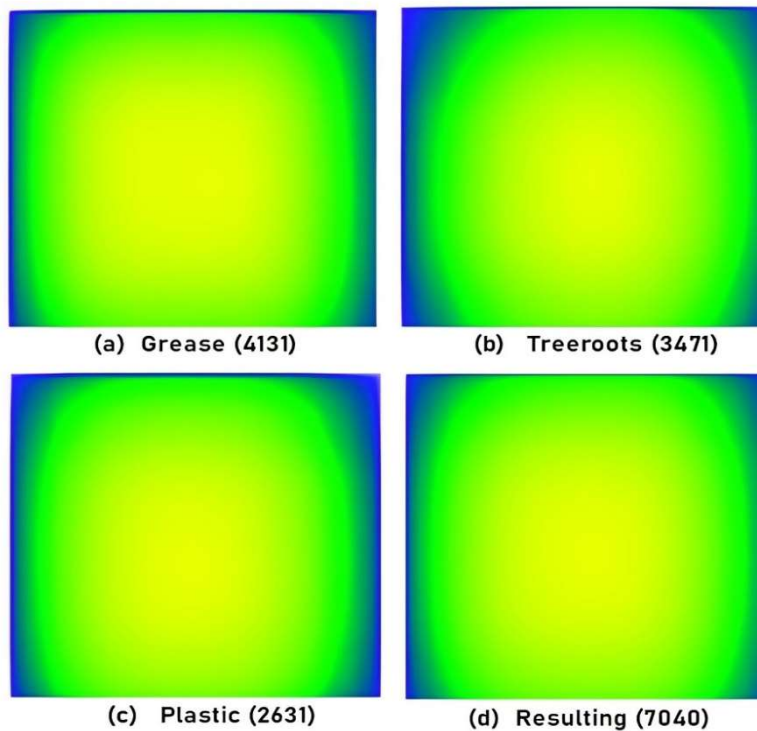


Figure 23 Heatmap visualization of annotation details for recorded images

The spatial distribution of annotations, represented by bounding boxes, for the considered blockage types across all captured frames is displayed as a heatmap in Figure 23. Heatmaps provide a graphical representation of informative data, employing a color-coding system to convey values. In this context, the values correspond to the annotation details. Heatmaps offer a quick and visually comprehensive summary, facilitating the understanding of the intricate nature of the dataset. The use of colours in a heatmap enables a more intuitive comprehension

of the correlations between annotated values, compared to traditional numerical tables. The heatmap presented here exhibits yellow colour for highly positioned regions of annotations, while light green colour denotes lower positioning. All depicted heatmaps demonstrate that the majority of annotations are concentrated towards the center of the object classes within the frames.

The imagery data is divided into three balanced groups: training data (70%) consisting of 4928 frames, validation data (20%) comprising 1408 frames, and testing data (10%) with 704 frames, as depicted in Figure 24.

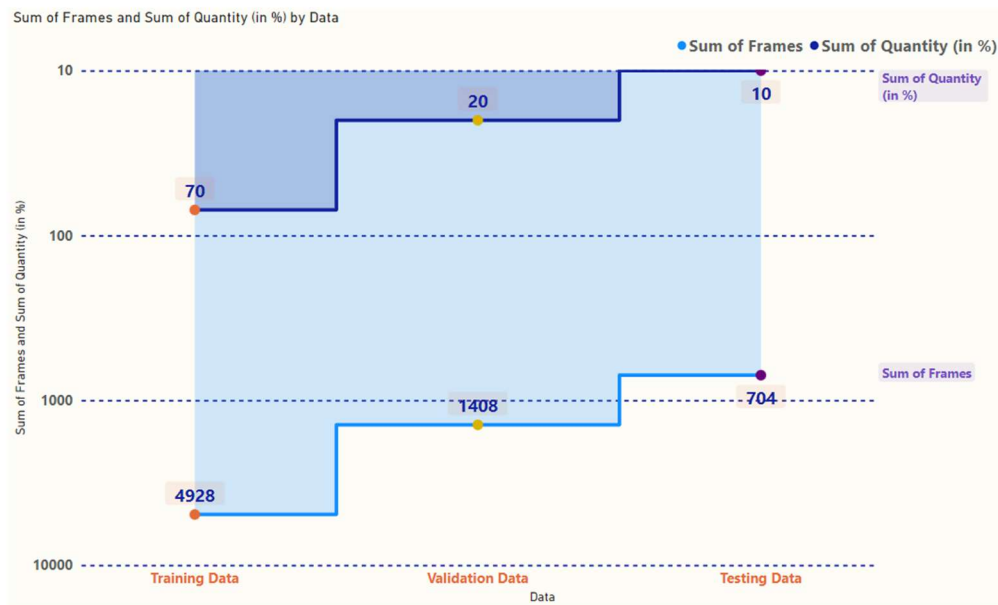


Figure 24 Data balancing for each class

Table 12 [4] provides annotation details specifically for the classes within the training data.

Table 12 Annotations for training data

Object Class (Sewer Blockage Type)	Annotations
Grease	2920
Tree roots	2455
Plastic	1821
Total	7196 (Average = 1.5 per frame)

These detailed annotations play a crucial role in training and validating detection models, enabling accurate identification and localization of sewer blockages.

4.5 Preprocessing and Augmentation Techniques

In this section, two representative preprocessing techniques have been employed on captured frames. Firstly, auto-orientation of pixel data was implemented by discarding the EXIF (i.e., image metadata) rotation and validating the pixel sort. Additionally, resizing the frames to 416 × 416 pixels was performed by stretching the frame without losing the source frame information.

To resize the frame from 352×240 to 416×416 without losing any information, we need to stretch the frame while maintaining its aspect ratio. Let us calculate the scaling factors for width and height:

$$\begin{aligned} \text{Scaling factor for width } (sf_w) &= \text{target_width} / \text{original_width} \\ &= 416 / 352 \approx 1.1818 \end{aligned} \quad (4.3)$$

$$\begin{aligned} \text{Scaling factor for height } (sf_h) &= \text{target_height} / \text{original_height} \\ &= 416 / 240 \approx 1.7333 \end{aligned} \quad (4.4)$$

Now, new dimensions of the resized frame can be calculated as follows:

$$\text{resized_width} = \text{original_width} * sf_w = 352 * 1.1818 \approx 416 \quad (4.5)$$

$$\text{resized_height} = \text{original_height} * sf_h = 240 * 1.7333 \approx 416 \quad (4.6)$$

These image preprocessing methods contribute to reducing model training time and accelerating inference for detection models.

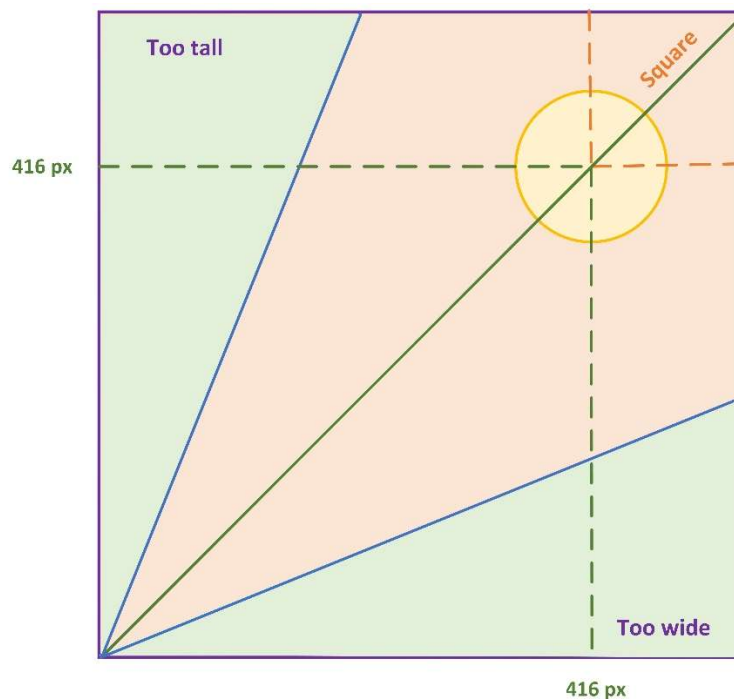


Figure 25 Distribution Graph of Aspect Ratios

Figure 25 illustrates the distribution graph of aspect ratios in the S-BIRD dataset, confirming that all frames are square-sized with dimensions of 416×416 pixels.

Furthermore, important image-level augmentation techniques have been employed to generate new training instances from existing data. Figure 26(a) demonstrates the visual result of applying a 25 percent gray scaling to the input training frame. This technique increases training variation while retaining colour information during inference. Figure 26(b) illustrates the application of salt and pepper noise, also known as impulse noise, to 5 percent of the pixels in the input frames. This noise helps the detection model adapt to camera artifacts during training

by adding bright and dark pixels to different regions of the frames, preventing adverse effects and overfitting.

To enhance the detection model's robustness against changes in light and camera settings, random exposure adaptations have been introduced. These adaptations randomly adjust the exposure of the input frame between -25 and +25 percent, as shown in Figure 26(c) [4]. The complete implementation is as given -

- Gray Scaling: To apply a 25 percent gray scaling to the input training frame, the formula is as follows:

$$New_pixel_value = (0.75 * R) + (0.75 * G) + (0.75 * B) \quad (4.7)$$

where R, G, and B represent the red, green, and blue colour channels of each pixel, respectively.

- Salt and Pepper Noise:
 - Determined the number of pixels in the image ($416 \times 416 = 173,056$ pixels),
 - Selected 5 percent of the total pixels ($0.05 * 173,056 = 8,653$ pixels) randomly,
 - Assigned a random intensity of either the maximum (255) or minimum (0) value (bright or dark pixel) to each selected pixel.
- Random Gamma Exposure Adaptations: To randomly adjust the exposure of the input frame between -25 and +25 percent, the formula is given below:

$$\begin{aligned} New_pixel_value \\ &= Old_pixel_value * (1 \\ &+ Random_number_between(-0.25, 0.25)) \end{aligned} \quad (4.8)$$

where $Random_number_between(-0.25, 0.25)$ generates a random number between -0.25 and 0.25.

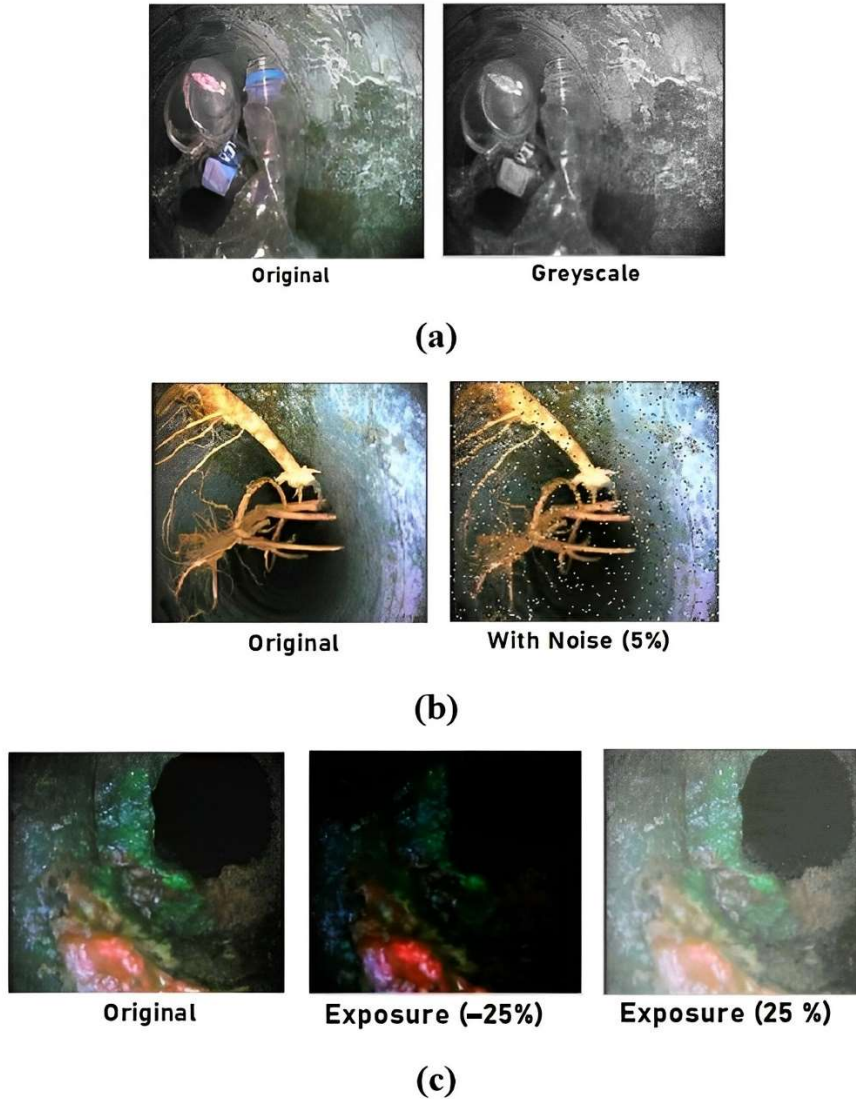


Figure 26 Illustrative outcomes of common augmentation methods: (a) grayscale transformation, (b) salt and pepper noise, (c) arbitrary exposure variation

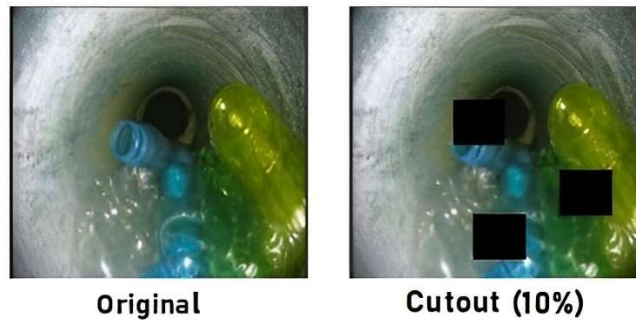
In addition, two important advanced augmentation techniques, namely cutout and mosaic, have been utilized. Figures 27(a) and 27(b) [4] depict the visual outcomes of these techniques, respectively. Cutout involves inserting three occlusions in 10 percent of the input frames, helping the detection model handle object occlusion. The mosaic technique combines multiple images from the training set to create a collage, improving the detection model in effectively detecting small objects. In this case, four different sewer block frames were added to a single frame i.e., Random Image Cropping and Patching (RICAP).

Overall, these augmentation techniques significantly contribute to improving the efficiency of the object detection model by increasing the number and diversity of training instances and annotations. They also help reduce training time and costs. Consequently, discrete output versions have been generated for the source frames.

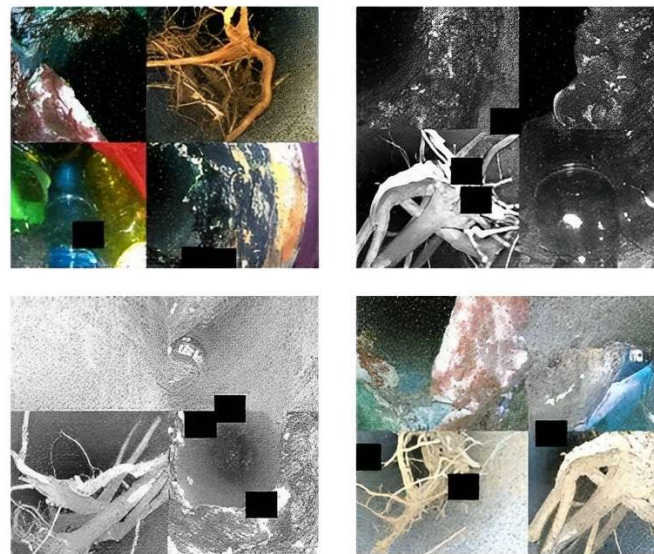
Table 13 [4] presents the quantitative details of the training frames in the S-BIRD dataset after applying preprocessing and augmentation techniques.

Table 13 Computational details of training samples in S-BIRD after preprocessing and augmentation.

Metric	Values
Total frames	14,765
Annotations	69,061 (Average = 4.7 per frame)
Average frame size	0.173 Megapixels
Mean frame ratio	416 × 416 (square)
Aspect ratio Class	1:1
Angle of diagonal	0.785 radian = 45°
Length of diagonal	588 pixels
Pixel density	12 pixels/mm or 290 pixels/inch



(a)



Mosaic

(b)

Figure 27 Visual outcomes of enhanced augmentation methods: (a) cutout and (b) mosaic

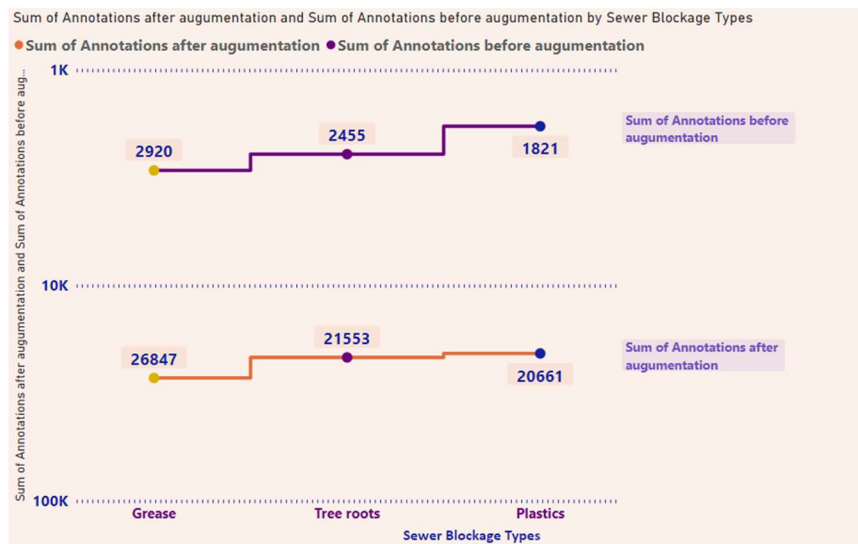


Figure 28 Annotation specifications for each class in the training dataset following image-level augmentation

Graph in Figure 28 displays the increased annotations for each sewer block type in the S-BIRD training data after applying augmentation techniques. The number of annotations for grease, tree roots, and plastics are now 26,847, 21,553, and 20,661, respectively, resulting in a total of 69,061 augmented annotations (bounding boxes). This represents a significant increase of 61,865 annotations, or 859.714%. The preprocessing and augmentation techniques were implemented using OpenCV, a popular computer vision and machine learning library, along with Python programming on the Linux platform, achieving the desired results.

4.6 Annotated Heatmap and Object Count Histogram

Two important metrics, the annotated heatmap and the object count histogram, have been analysed to evaluate the effectiveness of the training data. Figure 29 illustrates the location of all annotations for grease, plastic, and tree roots in the training data of S-BIRD through heatmaps. These heatmaps provide an overview of the most common positions and distribution of annotations for each class. From the colour information in the heatmaps, it is evident that the majority of annotations (yellow colour) are located at the far left and right of both top and bottom sides of the images for all object classes.

A histogram is a useful chart that represents numeric data in individual columns called bins. Figure 30 [4] presents the object count histogram, which details the number of frames on the y-axis and the corresponding object counts for all classes on the x-axis. The number of objects or annotations for grease and tree roots ranges up to nine instances, as shown in Figure 30(a) and 30(b). Grease objects appear once in 1730 frames and four to five times in 1400 to 1600 frames, as depicted in Figure 30(a). Similarly, there are 1926 frames with a single tree root object, and approximately 1500 frames contain three to four tree root objects, as shown in Figure 30(b). The number of plastic objects varies up to seven instances, with four plastic objects present in 2494 frames, and around 2200 frames containing one plastic object, as illustrated in Figure 30(c). Figure 30(d) represents the object count histogram for all classes, demonstrating that 11,339 frames contain four to five objects. It also indicates a significantly lower occurrence of frames with only one object compared to the total number of annotations (69,061). The findings from both the annotated heatmap and the object count histogram confirm the high accuracy and quality of each class of imagery data in S-BIRD.

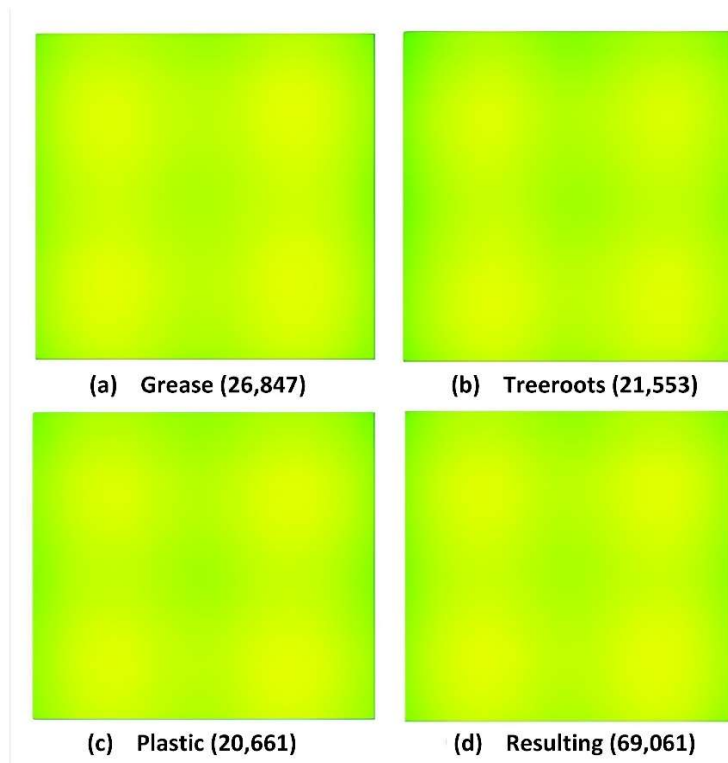


Figure 29 Heatmap of annotations providing location details of all classes

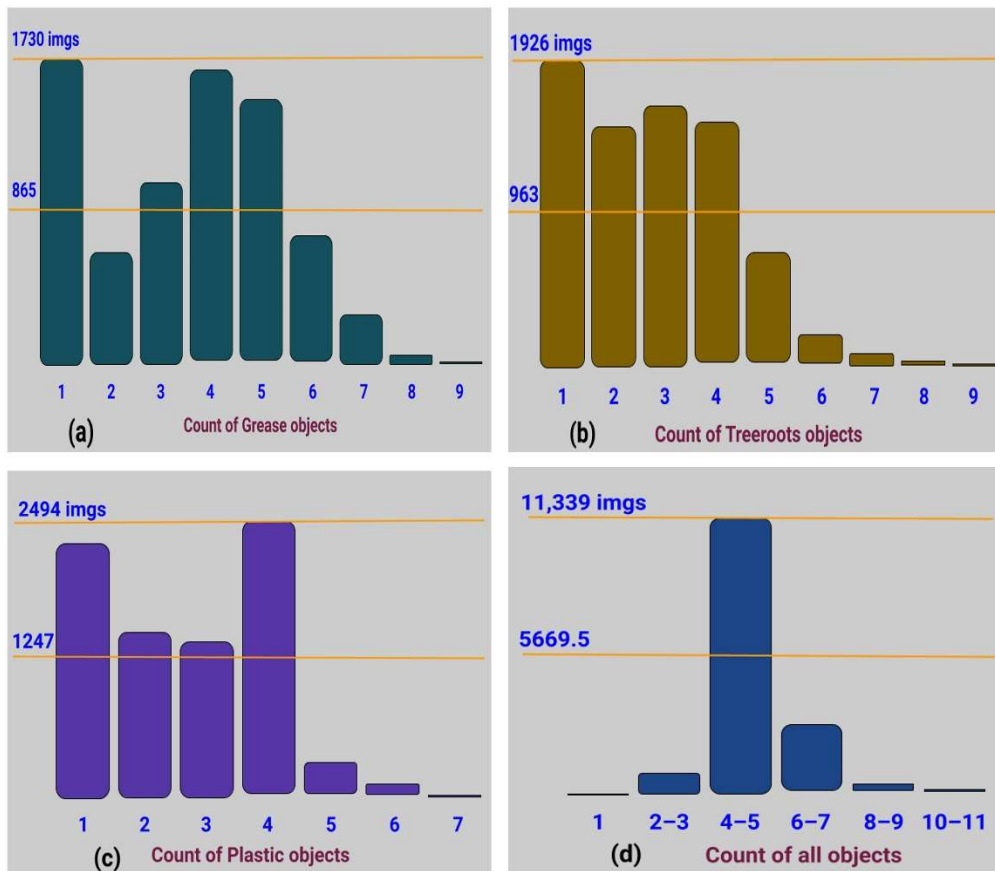


Figure 30 Histogram depicting the number of objects for: (a) grease, (b) tree roots, (c) plastic, and (d) all categories

4.7 Development of Sewer Blockage Detection Models using Transfer Learning and Fine Tuning

The AI models implemented using the PyTorch framework for mobile deployment, effectively detected sewer blockages including grease, plastic, and tree roots. The training process involved annotations in two different formats such as Pascal VOC and PyTorch TXT and utilized a Tesla V100-DGXS-32GB GPU workstation with a Docker Container for efficient training.

4.7.1 Optimization and Training of YOLOX using newly developed S-BIRD dataset

The crucial information about how transfer learning and fine-tuning was applied for training of YOLOX model on newly developed S-BIRD dataset is given as follows.

- The very first, a pretrained single stage YOLOX-small model architecture with DarkNet53 backbone that was used as the starting point for transfer learning. This backbone is a variant of the Darknet architecture with shortcut connections and downsampling layers.
- The input size of 14,765 frames from the training set of the S-BIRD dataset was then matched to 640×640 pixels, which was previously an average size of 0.173 megapixels and a square ratio of 416×416 pixels. The annotations in the dataset had consisted of 69,061 instances, resulting in an average of 4.7 annotations per frame.
- Next, the architecture of the YOLOX-s model was modified by setting it to identify 3 classes, including tree roots, plastics, and grease, to align with the object classes in the S-BIRD dataset. This modification was achieved by adjusting the 'num_classes' attribute to 3, indicating that the model would be trained to accurately detect and classify these specific classes.
- The depth parameter was set to 0.33 which controls the network depth and refers to the number of layers. Whereas the width parameter was 0.50 which determines the network width i.e., the number of channels or filters in each layer.
- Now, the modified YOLOX-Small model was inserted with the 'yolox_s.pth' weights which includes the learned representations and configurations for further training on new custom dataset.
- In the fine-tuning process, the YOLOX model aimed to optimize the loss function and improve its performance on the sewer blockage detection task. This optimization process was performed over 300 training epochs, denoted as max_epoch, which was set in this case.
- During fine-tuning, additional training parameters were considered as shown in Table 14 [4]. The training process involved minimizing the difference between the predicted bounding boxes and the ground truth annotations. This was achieved by optimizing the model's parameters using techniques such as stochastic gradient descent (SGD) with a suggested specific learning rate, weight decay, and momentum, which controlled the magnitude of parameter updates, regularization, and optimization dynamics, respectively. The model was trained with a learning rate warm-up for the first five epochs to stabilize the training process.
- Additionally, this process incorporated data augmentation techniques such as random rotations up to 10 degrees, translations up to 0.1, and scaling between 0.1 and 2.
- The weights had been updated as per the training progress and evaluation matrix was computed for the validation of the detection model.

Table 14 Key Training Parameters

Parameters	Significances
learning model	YOLOX-s
Annotation data type	Pascal VOC XML
max_epoch	300
batch_size	16
fp16	True
num_classes	3
Params	8.94 M
Gflops	26.64
depth	0.33
width	0.5
input_size	(640, 640)
random_size	(14, 26)
nmsthre	0.65
degrees	10.0
translate	0.1
scale	(0.1, 2)
mscale	(0.8, 1.6)
shear	2.0
warmup_epochs	5
weight_decay	0.0005
momentum	0.9

The timing and precision results of the developed detection Model-1 (using YOLOX-s) for S-BIRD are presented in Table 15 and Table 16, respectively [4].

Table 15 Timing analysis of the trained model

Timing Parameters	Outputs (Milliseconds)
Average forward time	3.19 ms
Average NMS time	0.88 ms
Average inference time	4.07 ms

Table 16 Precision evaluation of the trained model

Object Class (Sewer Block Types)	Average Precision	map_5095	map_50
grease	0.9004	0.7885	0.9005
tree roots	0.8930		
plastic	0.9081		

According to Table 15 and Figure 31, the sewer blockage detection Model-1 has achieved an average precision of 90.04% for grease blocks, 90.81% for plastic blocks, and 89.30% for tree root blocks. The mean average precision (mAP) computed at an Intersection over Union (IoU)

threshold of 0.5 is 90.05%. Additionally, the mAP calculated over different IoU thresholds, ranging from 0.5 to 0.95 with a step of 0.05, is 78.85%. The selection of the best-fit model was performed using cross-validation or rotation estimation technique [80]. Figure 32 illustrates visually accurate detections of sewer blocks, including tree roots, plastic, and grease.

The developed model successfully handled scenarios with multiple sewer blockages in the same frame, making it suitable for real-time detection. These results confirm the consistency and effectiveness of the newly introduced S-BIRD dataset.

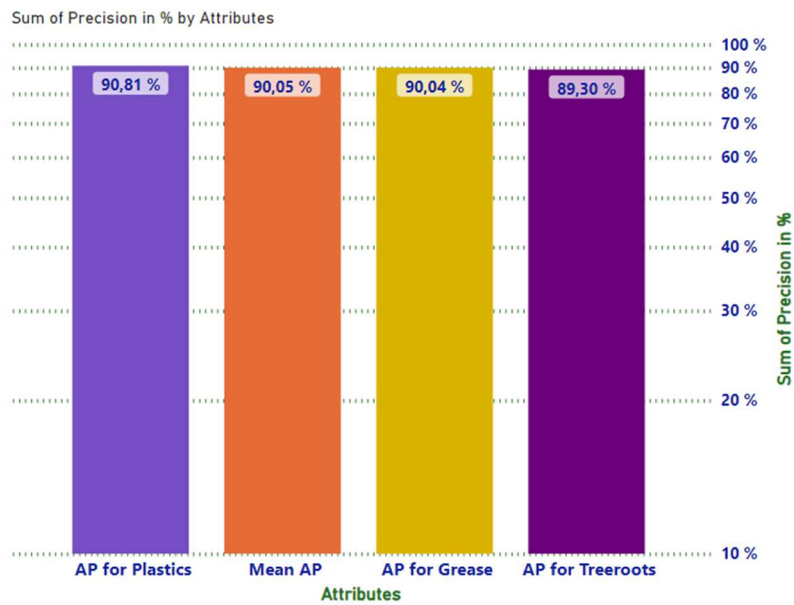


Figure 31 Detection Results of YOLOX-s for Sewer Block Classes in S-BIRD

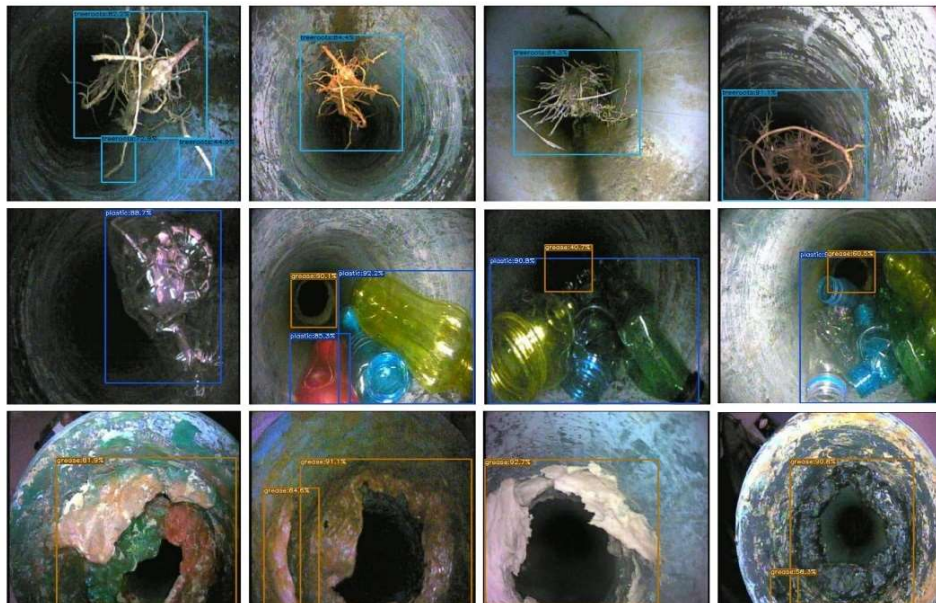


Figure 32 Visual Illustrations of Precise Detection of Tree Roots, Plastic, and Grease Sewer Block Types

4.7.2 Optimization and Training of YOLOv5 using newly developed S-BIRD dataset

The crucial information about how transfer learning and fine-tuning was applied for training of YOLOv5 on newly developed S-BIRD dataset is given as follows.

- The first step of the process involved selecting a small YOLOv5 version 6.0-187-gf3085ac, which is based on PyTorch 1.10.0a0 with CUDA support, specifically to fulfil our need to apply for real-time applications on mobile devices. This version utilized a lightweight backbone architecture called CSPDarknet53, which integrates Cross Stage Partial (CSP) connections.
- The backbone layers of this model were held constant, meaning they remained unchanged throughout the training process. This decision was made to preserve the valuable representations learned during the initial pre-training stage. This experimentation was done on first 10 modules in the backbone layers with trial basis as freezing and unfreezing.
- The input size of the training set from the S-BIRD dataset consisted of 14,765 frames, which remained unchanged at a square resolution of 416×416 pixels. The frames were accompanied by ground truth metadata, specifically annotations for 69,061 objects, resulting in an average of 4.7 annotations per frame.
- The depth parameter was set to 0.33 which controls the network depth and refers to the number of layers. Whereas the width parameter was 0.50 which determines the network width i.e., the number of channels or filters in each layer.
- Subsequently, the model architecture was modified to accommodate the detection of three specific classes present in the S-BIRD dataset: tree roots, plastics, and grease. This modification involved adjusting the 'num_classes' attribute to 3, signifying the model's training objective of accurately detecting and classifying these particular classes.
- To optimize the loss function and improve its performance on the sewer blockage detection task, the training process was performed over 6000 epochs, denoted as max_epoch, which was set in this case. But the Early Stopping mechanism was used with a patience of 100 epochs, meaning that if no improvement were seen in the validation results for 100 consecutive epochs, the training would stop early.
- The model architecture, which consists of 270 layers and a total of 7,027,720 parameters, is used for the training process. This training is performed using the stochastic gradient descent (SGD) optimizer, which is configured with specific hyperparameters including learning rate, weight decay, momentum as given in Table 17.
- The modified YOLOv5 small model was initialized with the 'yolov5s.pt' weights, which contained learned details and configurations. Additionally, the 'data.yaml' file was provided as a data source, containing the necessary information about the training and validation frames in the S-BIRD dataset. These resources were utilized to facilitate further training of the model.
- The training process included the following hyperparameters: an initial learning rate (lr0) of 0.01 (ranges from 0.001 to 0.1) that gradually decreases to a final learning rate (lrf) of 0.1, a weight decay value of 0.0005 to prevent overfitting, and a momentum value of 0.937 (ranges from 0 to 1) for faster convergence. These hyperparameters were tuned and customized to optimize the model's performance for detection of intended sewer blockages in the frames.
- Of course, the power of trial and error process was utilised to obtain efficient trained model for task.

- During the training process, the model's weights were continually updated based on the progress made.
- The training process stopped at 933 epochs because no improvement was observed in the last 100 epochs as shown in Figure 33. The best results were observed at epoch 832, and the corresponding model was saved as "best.pt". The optimizer was stripped from both the "last.pt" and "best.pt" model files, resulting in a file size of 14.3MB each. The "best.pt" model was then used for further evaluation and validation.

```

Epoch  gpu_mem    box    obj    cls  labels  img_size
832/5999  2.02G  0.01562  0.02882  0.002291    66    416: 100%|██████
Class    Images    Labels    P    R    mAP@.5  mAP@
all      1410    2003    0.944  0.939  0.963    0.792

```

(a)

```

Epoch  gpu_mem    box    obj    cls  labels  img_size
932/5999  2.02G  0.01569  0.02859  0.002407    74    416: 100%|██████
Class    Images    Labels    P    R    mAP@.5  mAP@
all      1410    2003    0.958  0.927  0.96    0.792

```

Stopping training early as no improvement observed in last 100 epochs. Best results observed at epoch 832, best model saved as best.pt.

(b)

Figure 33 Results from training process – (a) at epoch 832 (b) at epoch 932

Table 17 Key Training Parameters

Parameters	Significances
learning model	YOLOv5-s
Annotation data type	PyTorch TXT
max_epoch	6000
patience	100
batch_size	16
fp16	True
num_classes	3
Params	7.2 M
Gflops	15.9
depth	0.33
width	0.5
input_size	(416, 416)
workers	8
anchor_t	4.0
scale	0.5
hsv_h, hsv_s, hsv_v	0.015, 0.7, 0.4
warmup_epochs	3
weight_decay	0.0005
momentum	0.937
translate	0.1

Fig. 35 displays the confusion matrix for categories such as grease, plastic, and tree roots within S-BIRD. The instances in dataset and their corresponding labels are given in the scatter diagram in Fig. 36. The correlation connections within the images of S-BIRD are visualized in Fig. 37. This indicates the accurate linkage between instances and labels across different scenes. The graph in Fig. 38 illustrates the relationship between precision (P) and confidence (C) whereas the correlation between recall (R) and confidence (C) is given in Fig. 39. The graph in Fig. 40 displays the mean average precision (mAP), which compares the truth bounding box and detection box. At a 94% threshold with a confidence level of 0.566, the F1 score is presented in Fig. 41, emphasizing the importance of balancing precision and recall in the sewer blockage images dataset. The graph in Fig. 42 displays the training and validation losses of the detection model during the classification process over 932 epochs on the S-BIRD dataset.

Both precision (P) and recall (R) exhibit high values of 94.40% and 93.90% respectively across all classes at epoch 832 in the model training. This developed sewer blockage detection Model-2 (using YOLOv5) achieved highest average precision of 95.90% for grease blocks, 98.40% for plastic blocks and 94.50% for tree root blocks as shown in Figure 34. The overall Mean Average Precision (mAP) for all classes as shown in Table 19, is remarkably high, accurately modelling detections at 96.30% with a threshold of 0.5. Additionally, the mAP calculated over different IoU thresholds, ranging from 0.5 to 0.95 with a step of 0.05, is 79.20%. The timing results have been shown in the Table 18 for processing each image having details (1, 3, 416, 416). In the provided illustration (Figure 44), the outcomes of the proficiently trained model on the Google source images [81] are depicted.

In Figure 43, when the S-BIRD dataset was used for training the detection model without the exposure technique, accurate detection (mAP) at 96.70% with a threshold of 0.5 was achieved. The utilization of the exposure technique for training led to a slight improvement of approximately 0.41% in the Mean Average Precision compared to not using the technique.

To calculate the improvement percentage, we can compare the mAP values between the two cases:

$$\begin{aligned}
 & \textit{Improvement percentage} \\
 &= ((\textit{mAP with exposure technique} \\
 &\quad - \textit{mAP without exposure technique}) \\
 &\quad / \textit{mAP without exposure technique}) * 100 \\
 \\
 & \textit{Improvement percentage} = ((96.30 - 96.70) / 96.70) * 100 \\
 &= (-0.40 / 96.70) * 100 \approx -0.41\%
 \end{aligned}$$

Table 18 Timing analysis of the trained model

Timing Parameters	Outputs (Milliseconds)
Average forward time	0.2 ms
Average NMS time	1.1 ms
Average inference time	11 ms

Table 19 Precision evaluation of the trained model

Object Class (Sewer Block Types)	Average Precision	map_5095	map_50
grease	0.959	0.792	0.9630
tree roots	0.945		
plastic	0.984		

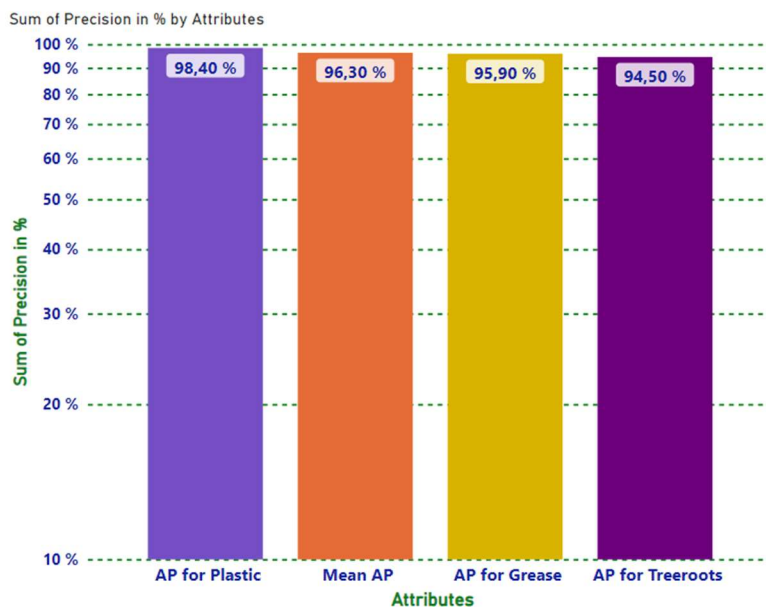


Figure 34 Detection Results of YOLOv5-s for Sewer Block Classes in S-BIRD

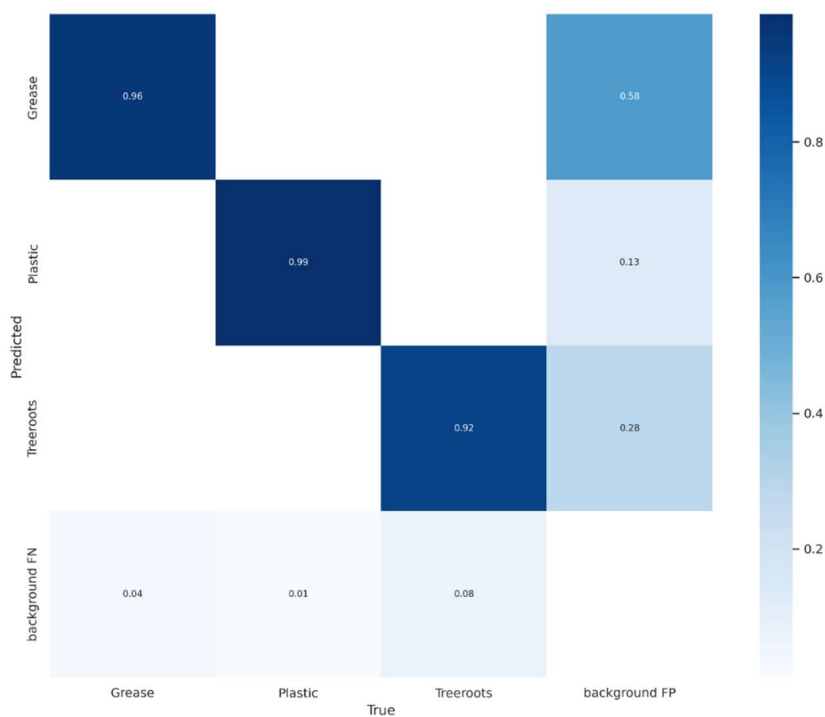


Figure 35 Confusion matrix for classes within dataset

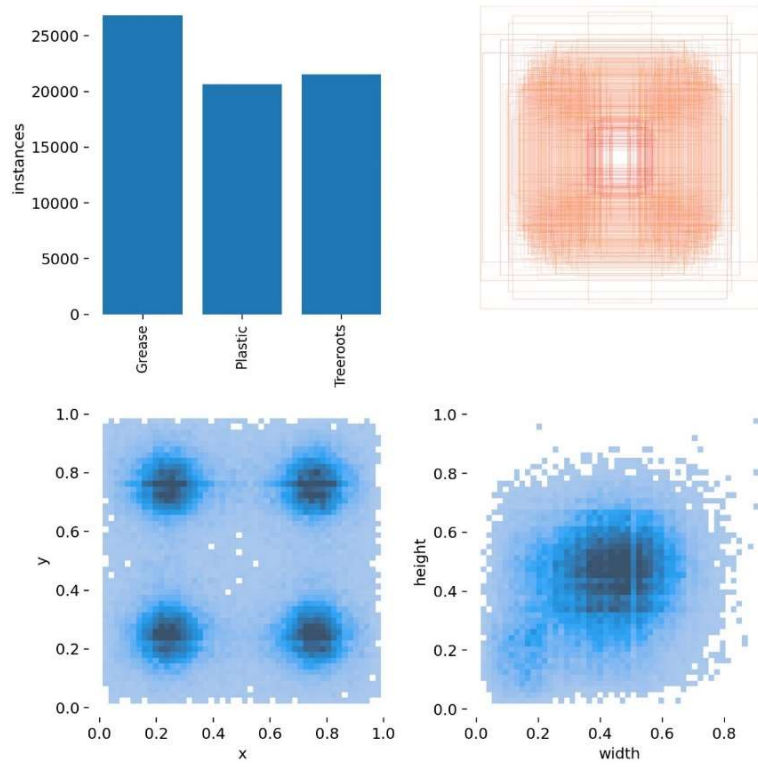


Figure 36 The scatter graph for instances and associated labels

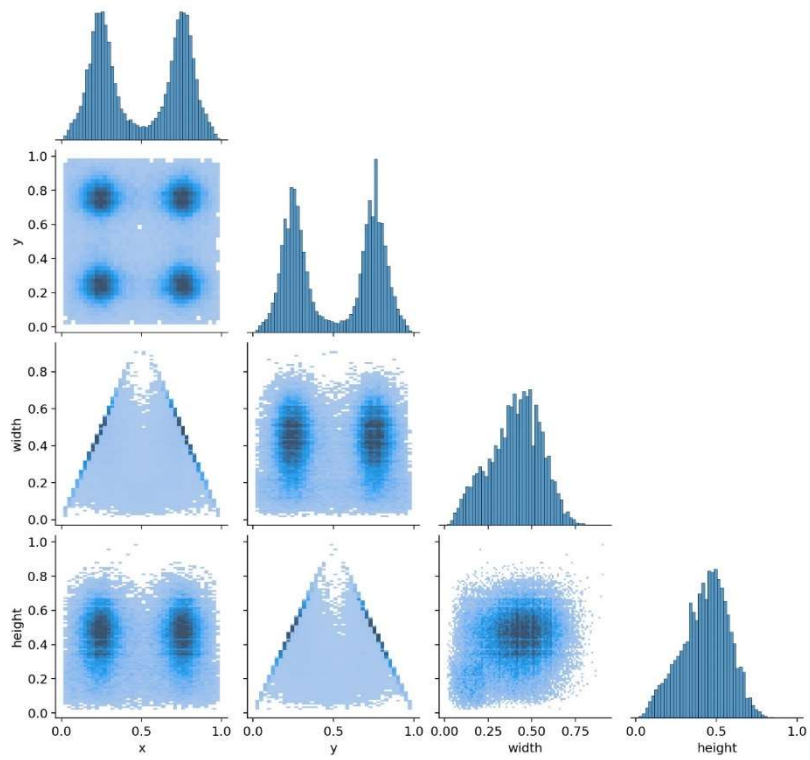


Figure 37 Correlations within the dataset of sewer blockage frames

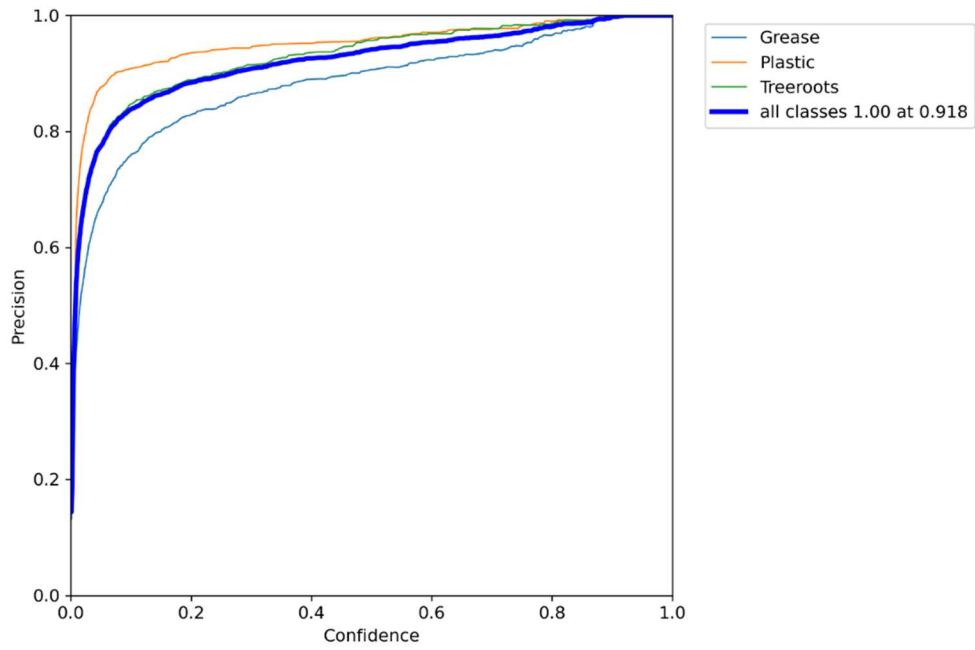


Figure 38 Precision (P) vs Confidence (C) graph

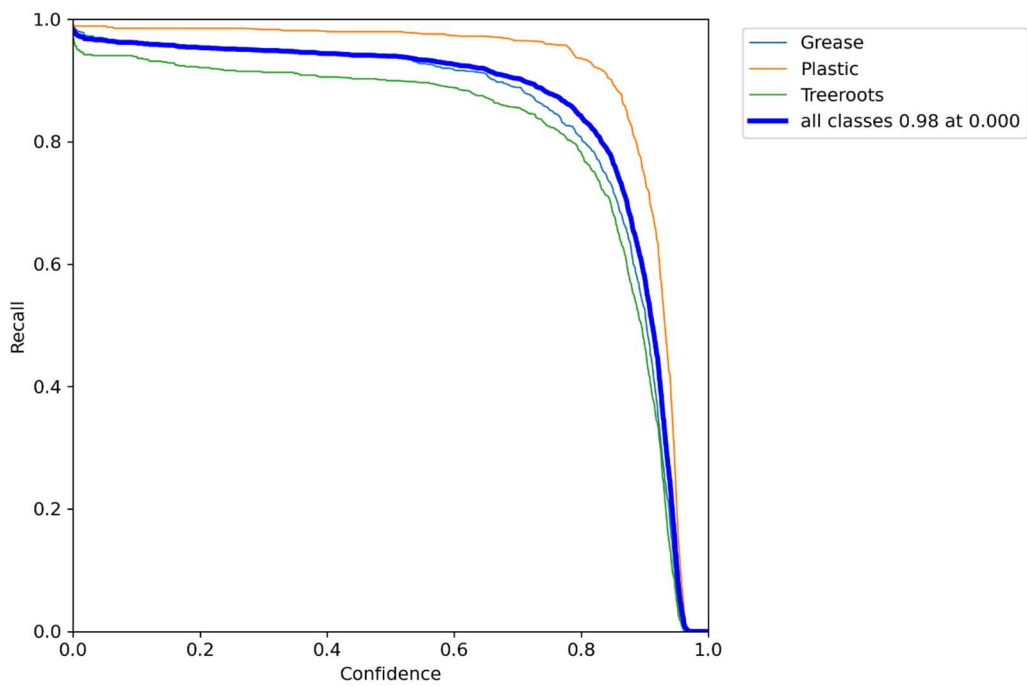


Figure 39 Recall (R) vs Confidence (C) graph

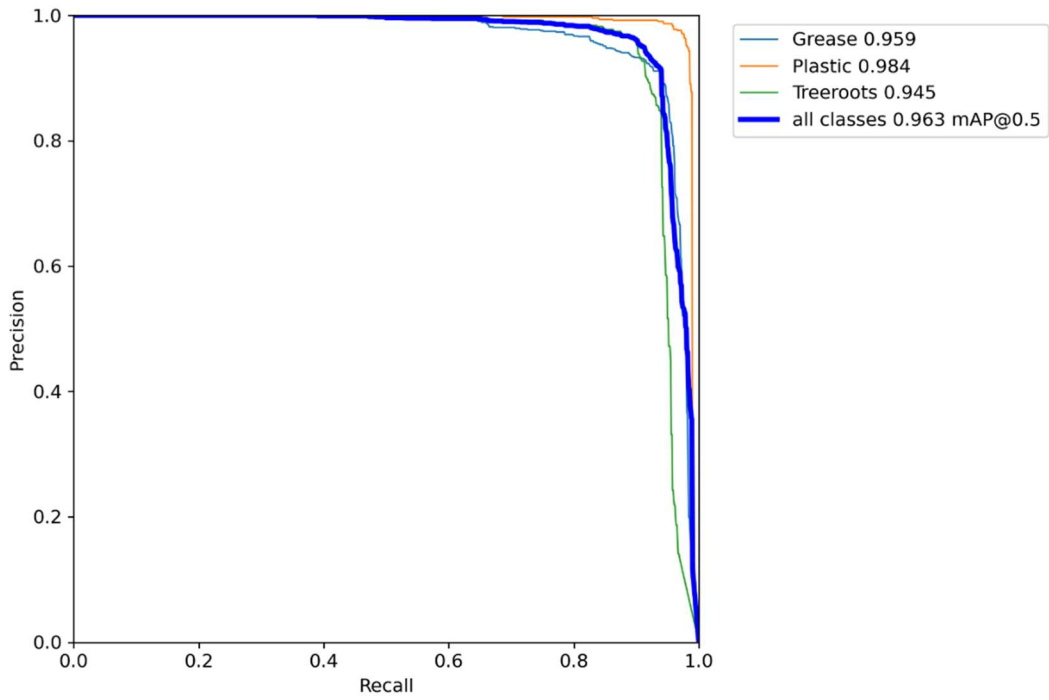


Figure 40 Precision (P) vs Recall (R) graph

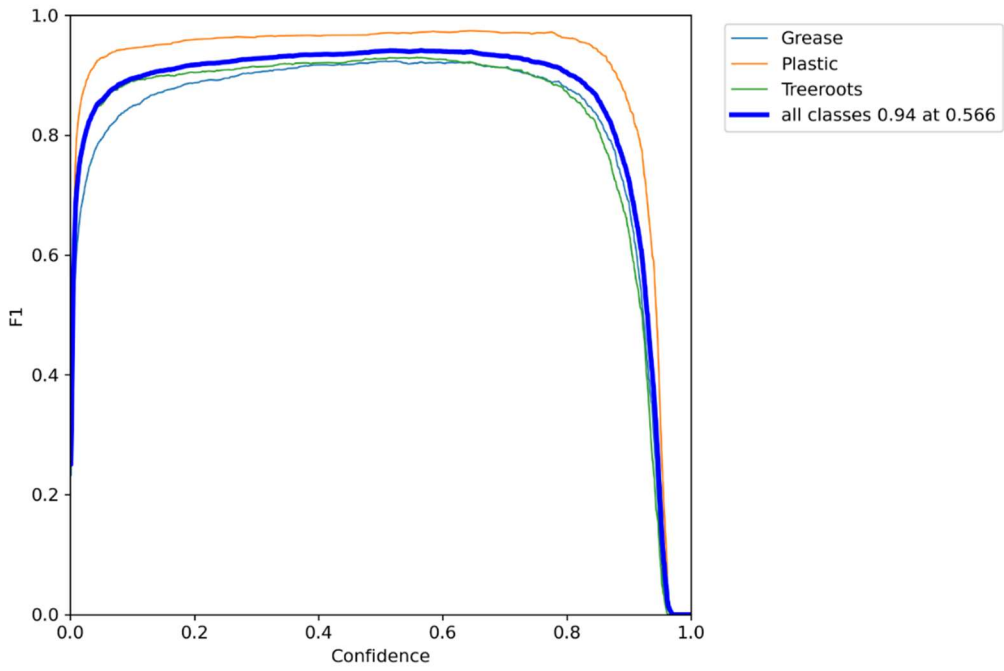


Figure 41 F1 score vs Confidence (C) graph

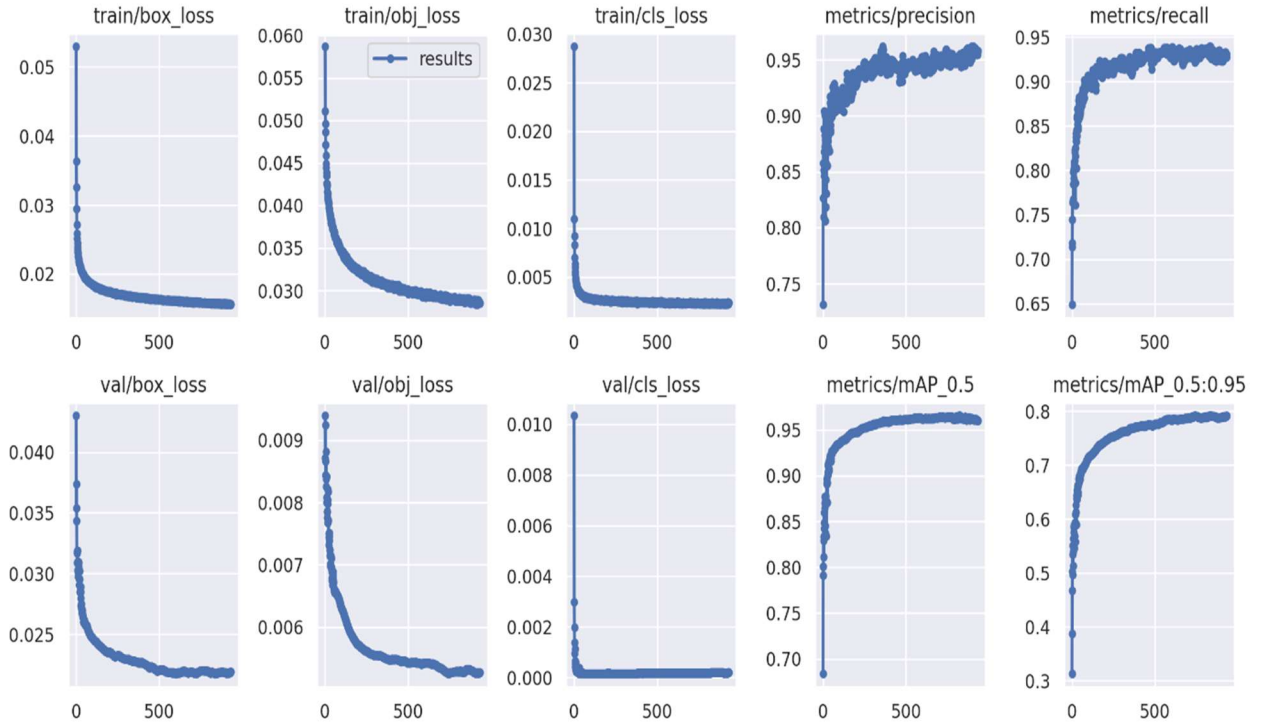


Figure 42 Training and validation losses of the detection model

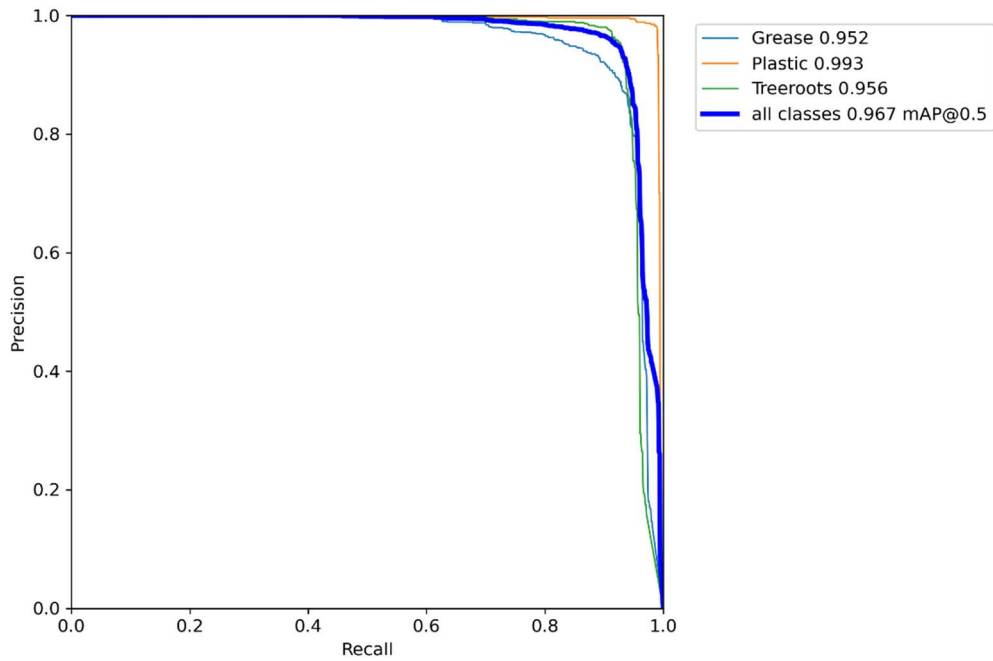


Figure 43 Precision (P) vs Recall (R) graph for model trained without using exposure in dataset



Figure 44 Detection Results on some Google Source images

By employing inductive transfer learning, fine-tuning techniques, and considering the specific details of the S-BIRD dataset, the developed Model-2 achieved highest precision consistently in detecting sewer blockages. The model's formulation, along with the training parameters and dataset characteristics, ensured the model's effective adaptation and suitability for real-world scenarios.

4.8 Embedded Vision Approach with S-BIRD

Embedded vision technology has emerged as an innovative and all-encompassing platform that enables the seamless integration of real-world visual applications across various domains, including home life equipment, healthcare, daily services, and security through detection and tracking. Within the realm of sewer robotics, the incorporation of embedded vision brings about significant advancements and benefits.

In particular, the integration of an object detection model, trained using the S-BIRD dataset, serves as a noteworthy enhancement to both existing and newly developed embedded vision-based sewer robotic systems. This model enables the system to accurately identify and detect sewer blockages, thereby assisting in the mitigation of recurring problems encountered in underground sewer networks. Figure 45 [4] emphasizes the vital role of the AI detector trained with the S-BIRD dataset in the embedded vision-based system.

For the embedded platform, the Jetson Nano was chosen due to its exceptional capabilities. With a 4 GB GPU card boasting 128 CUDA cores, the Jetson Nano is well-suited for executing deep neural network-based object detection models and processing consecutive frames in real-time. However, for even faster AI inference in real-world applications, an advanced version called Jetson Orin Nano is now available. It boasts an impressive 1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores and 40 TOPS, making it ideal for handling complex visual tasks.

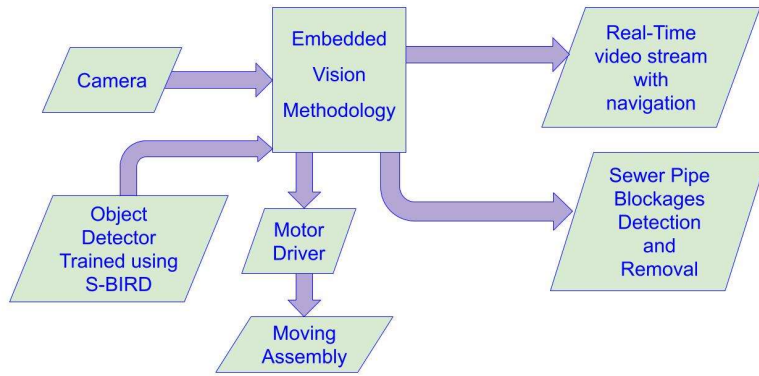


Figure 45 Embedded vision based system emphasizing AI detection with S-BIRD

To capture the surrounding frames for navigation and processing, a range of cameras are employed, including webcams, Arducam, or Raspberry Pi Camera (Raspicam). These cameras serve as the input source for the embedded vision system, enabling it to analyse the visual data in real-time. The output frames, depicting detected sewer blockages, are then displayed on a remote screen or location, facilitating prompt decision-making and remote monitoring. The embedded vision platform highlighted in Figure 46 [4] exemplifies the potential and effectiveness of this technology in sewer robotics.

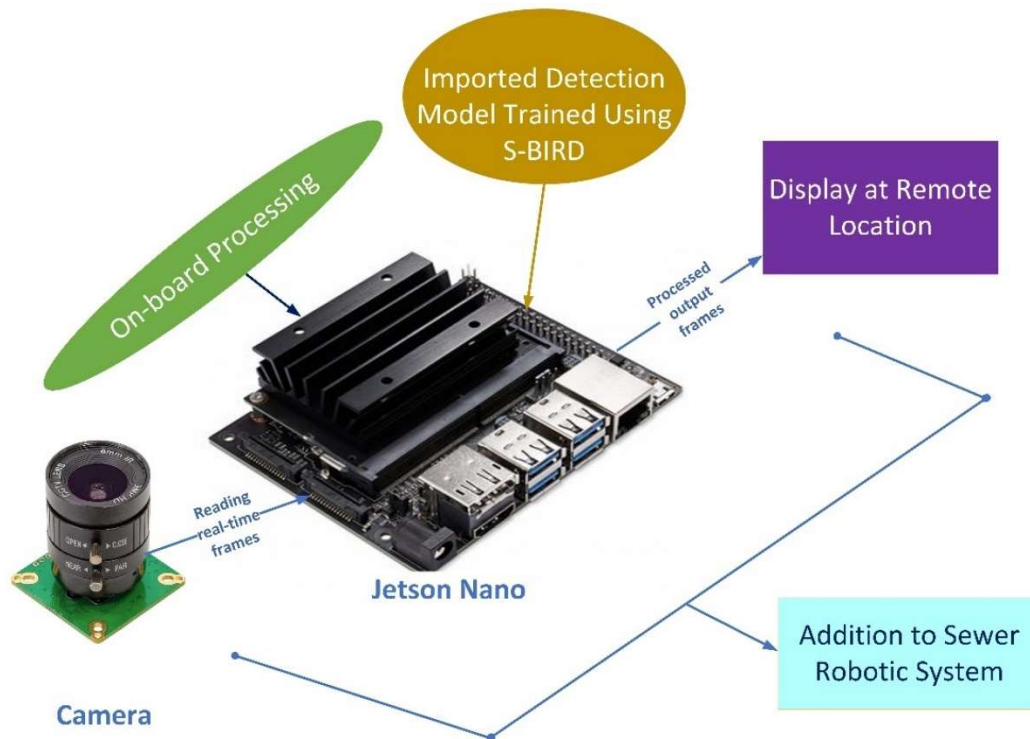


Figure 46 Incorporation of Embedded Vision platform into the sewer automated system

Overall, the integration of this embedded vision-based automation system, empowered by AI detectors trained using S-BIRD, provides a promising and economical solution to the persistent problem of underground sewer barriers. By meeting the needs of responsible authorities in any country, this advanced system contributes to the efficient management and maintenance of sewer networks, ensuring smooth and uninterrupted wastewater flow with accuracy.

4.9 Discussion

This thesis presents a development of representative dataset for sewerage blockages and developed deep neural detection models using transfer learning and fine tuning techniques for AI application. Through extensive experimentation and analysis, the research has demonstrated the effectiveness of above mentioned approaches in enhancing the accuracy, efficiency, and robustness of AI systems.

4.9.1 Discussion on Enhanced AI in Research Work

In research methodology, dataset development is a major contribution and begins when the research problem is defined, and the research design is established. Basically, data are of two types, namely primary and secondary where primary data is newly collected and original, while secondary data is previously collected and statistically processed [82]. In AI also, data search and development are the major research theme. As in my case, the image data is primary i.e., it has been originally developed using mechanical device like sewer camera and simulated sewer network. The literature review, critical survey, direct communications with municipalities and interviews of authorities, searching in open research community, experiments with object classes, these all clarify the need of presented research work via newly developed representative critical multi class dataset and developed deep neural network model for real world application in the urban sewer system.

It is observed that the representative dataset plays a crucial role in providing a comprehensive and diverse set of examples for training the deep neural detection models. But it is essential that data capture a wide range of real-world scenarios to enable the AI models to learn and generalize effectively. The S-BIRD dataset developed in this research not only facilitated the training process but also contributed to the models' overall performance by reducing biases and improving their ability to handle complex and varied inputs. So, the processing and analysis of the data plays a crucial role for validation purpose. As S-BIRD dataset comes under classification type of processing operation i.e., typically simple classification which further indicated according to attributes of each instance. This is because the classification process involves arranging data into groups or classes based on common characteristics. Also, each class ("grease," "plastics," and "treeroots") consist of instances i.e., frames possessing specific attributes for all classes and creates homogeneous groups within the dataset. For the images, descriptive attributes i.e., qualitative characteristics are color, texture, style, contents, etc. whereas numerical attributes i.e., quantitative measurements are dimensions, pixel values, aspect ratio, entropy (a measure of randomness in pixel values), and the number of objects or features detected within an image. The statistical parameters computed by histogram and generated heatmap, inform about the data pattern and location details i.e., it confirms strength of each class.

Furthermore, the utilization of transfer learning techniques proved to be instrumental in development of AI models with learned features for intended detection tasks. This training process falls under inductive transfer learning because the label information for both the source (S-BIRD dataset) and the target (target-domain instances) i.e., recognition of known classes was available. By initializing the deep neural networks with some prior knowledge, the models demonstrated improved convergence, faster training times, and better performance on the target detection tasks. Transfer learning effectively transferred the learned representations, enabling the learning networks to adapt to different domains and tasks with minimal additional training.

The results obtained from the experiments validate the significance (consistency and feasibility) of the developed representative dataset and the efficacy of inductive transfer learning in enhancing AI i.e., in detection model development. It additionally mitigates bias within the research approach. The improved accuracy, efficiency, and robustness achieved by the deep neural detection models underscore the practical benefits of these approaches in real-world applications. Specifically, statistical details of development are given and discussed in the case study section.

Certainly, conducting research is not without its obstacles and moments of difficulty. This research faced challenges such as the availability and quality of the representative dataset, as well as the transfer learning strategy for deep neural network so that new application oriented AI model can be developed. Furthermore, the task of procuring a GPU computation system presented an additional challenge. This thesis emphasizes the importance of careful dataset curation, ensuring data integrity, and addressing potential biases. Furthermore, choosing suitable learning network and optimizing transfer learning parameters require careful consideration and experimentation to achieve optimal results.

This research also highlights the potential of flexible integration of advanced embedded vision platform powered by AI detectors trained with representative datasets and supported by single-board computers with exceptional GPU capabilities. It offers a promising and affordable solution for real-time processing, effective decision making and improved performance, leading to advancements in various domains through accurate and efficient visual analysis enabled by AI techniques.

Overall, this thesis contributes to the advancement of AI by highlighting the value of a developed representative dataset, the effectiveness of transfer learning and fine tuning techniques for training and development of deep neural detection models, and integration of embedded vision approach. The insights gained from this research provide a solid foundation for further exploration and development in the field, fostering advancements in AI technology and its applications across various domains.

4.9.2 Discussion on Case Study in Wastewater Management

The research work conducted in this case study focused on the development of the S-BIRD (Sewer-Blockages Imagery Recognition Dataset), aiming to utilize AI techniques, specifically computer vision and deep learning i.e., advanced machine learning, for real-time detection and identification of sewer blockages. This work emphasizes the necessity of overcoming wastewater sewer barriers and highlights the limitations of existing algorithms and automated systems for sewer inspection. It underscores the significance of standardized datasets in addressing challenges in wastewater management, considering the difficulties associated with obtaining such datasets due to the unhygienic and malodorous nature of sewers, as well as copyright or confidentiality concerns. The study showcases the potential of computer vision techniques and machine learning algorithms as valuable tools for enhancing strategies in this domain.

The S-BIRD dataset introduced in this study includes diverse multi-class imagery samples of prevalent sewer blockages caused by grease, plastic, and tree roots. It serves as a benchmark for evaluating real-time detection results and facilitates the development of effective recognition models. The tools used for dataset development, including a constructed sewer

pipeline and an inspection camera for sewerage systems, enable the capture of real-time frames of sewer blockages in a simulated sewer environment, ensuring authenticity and relevance in training detection models. The dataset includes comprehensive annotations for each captured frame, providing vital information for subsequent computations and analysis. The thesis presents arithmetical details of the dataset, such as the number of frames, annotations, and aspect ratios, and utilizes visualizations, such as class balance and heatmaps, to represent the dataset's characteristics. Preprocessing and augmentation techniques, such as auto-orientation, resizing, gray scaling, and noise addition, were applied to enhance the dataset's quality and improve the robustness of detection models.

The study successfully developed deep neural object detection models for sewer blockage detection on the S-BIRD dataset using transfer learning and fine-tuning techniques in AI, specifically the customized YOLOX and YOLOv5 models for mobile deployment with high accuracy. The training process involved the use of the PyTorch framework, annotations in Pascal VOC and PyTorch TXT formats, and a Tesla V100-DGXS-32GB GPU workstation with a Docker Container for efficient training.

For the YOLOX-s, transfer learning was applied with a DarkNet53 backbone. The model was trained on 14,765 frames with annotations for three classes in S-BIRD, and various training parameters were optimized. The model architecture was modified to accommodate these classes, and the training process involved fine-tuning the model over 300 epochs. Data augmentation techniques, including random rotations, translations, and scaling, were also applied during training. The developed detection Model-1 achieved an average precision of 90.04% for grease blocks, 90.81% for plastic blocks, and 89.30% for tree root blocks, with a mean average precision (mAP) of 90.05% at an IoU threshold of 0.5, demonstrating its consistency and feasibility of the presented S-BIRD dataset for detection task.

Similarly, for the training of YOLOv5 using the S-BIRD dataset, transfer learning and fine-tuning were applied. A YOLOv5 small version with a CSPDarknet53 backbone was selected, and the model architecture was modified to detect the same three classes. The training process involved training the model over 6000 epochs, but best results were observed at 832 epochs, with early stopping after 100 epochs of no improvement. Various hyperparameters were customized to optimize the model's performance. The developed detection Model-2 achieved the highest average precision of 95.90% for grease blocks, 98.40% for plastic blocks and 94.50% for tree root blocks, with the highest mean average precision (mAP) of 96.30% for all classes at a threshold of 0.5.

Both detection models have also been tested on pure negative samples, images without blockages, to assess their ability to correctly identify instances with no blockage. This evaluation contributes to a comprehensive assessment of the models' performance in diverse scenarios.

The timing analysis showed that the developed Model-1 had lower inference times compared to the Model-2. The Model-1 had an average inference time of 4.07 ms compared to 11 ms for the Model-2. This indicates that the Model-1 model is more computationally efficient in detection.

Overall, both the developed models (Model-1 and 2) demonstrated high accuracy and precision in detecting sewer blockages, with mean average precision (mAP) values above 90%. The

models successfully handled scenarios with multiple blockages in the same frame, making them suitable for real-time detection. The results confirmed the effectiveness of the S-BIRD dataset and the applicability of transfer learning and fine-tuning techniques for detection task.

Also, the integration of an embedded vision-based automation system, featuring AI detectors trained with the S-BIRD dataset and empowered by advanced GPU-based single-board computers like Jetson Nano or Jetson Orin Nano, offers a compelling solution to the long-standing challenges of underground sewer barriers. This innovative approach holds great potential for improving wastewater management strategies and ensuring efficient maintenance of sewer networks.

4.9.3 Comparative discussion on AI-Driven Approach and MOEAs

The AI-driven strategy proposed in this study holds notable advantages over Multi-Objective Evolutionary Algorithms (MOEAs) [83], commonly employed in wastewater system management. While MOEAs like NSGA-II, SPEA2, MOPSO, and MODE excel at optimizing multiple objectives, they often necessitate intricate mathematical models and substantial computational resources [84, 85]. Conversely, the AI approach harnesses cutting-edge computer vision and deep learning techniques to rapidly and precisely identify sewer blockages. Demonstrating an impressive mean Average Precision (mAP) of 96.30% at a confidence threshold of 0.5, the model's exceptional precision in sewer blockage detection enhances wastewater management system reliability and efficiency. Furthermore, the AI method capitalizes on labelled training data and lightweight deep learning models, enhancing efficiency and real-time capabilities. This aligns with the pressing need for swift sewer blockage resolution to avert disruptions and overflows. The model's accuracy, speed, and dedicated focus on sewer blockage detection position it as a promising solution for immediate and effective urban wastewater management.

In contrast, MOEAs such as the sensitivity-based adaptive procedure (SAP) [86], optimal control algorithms [87], and novel methodologies [88] have proven effective across aspects like sewer rehabilitation and optimal scheduling. Nevertheless, their computational demands and reliance on intricate algorithms might impede real-time suitability. The AI-driven approach's real-time data processing ability, coupled with its superior detection accuracy, gives it a distinct advantage in addressing dynamic and critical scenarios such as sewer blockages.

While both AI-driven methods and MOEAs contribute to wastewater management progress, the AI approach's swift identification and response to sewer blockages render it particularly appropriate for immediate, practical applications in modern urban sanitation systems.

5 Conclusions and Further work

5.1 Conclusions

The research work presented in this thesis is concluded as follows:

- The creation of the representative S-BIRD dataset addressed the lack of appropriate data for training AI models in sewer blockage detection, capturing real-time frames of grease, plastic, and tree root blockages. This dataset provides valuable training data to improve the accuracy and robustness of detection models.
- The methodology employed various tools and techniques, including a simulated sewer network, a watertight sewer camera, and advanced image preprocessing and augmentation methods. These techniques ensured the authenticity and diversity of the dataset, allowing for effective training of detection models.
- The results obtained from the case study for developed sewer blockage detection models (Model-1 and 2) demonstrated high precision and feasibility, affirming the effectiveness of the S-BIRD dataset and their performance in real-world scenarios.
- The implementation of transfer learning and fine tuning techniques proved to be highly beneficial for improved convergence, faster training times, and enhanced performance in sewer blockage detection. This approach effectively transferred the learned representations, enabling the models to adapt to different domains and tasks with minimal additional training.
- The achievement of a mean average precision of 96.30% at 0.5 IoU demonstrates the effectiveness of methodical approach.
- The AI models trained on the S-BIRD dataset provide a valuable benchmark for assessing localization performance in real-time scenarios, serving as a crucial resource for researchers and developers in the field.
- The research filled the gap of a standardized matrix for implemented algorithms, offering reliable evaluation frameworks in the field of sewer blockage detection.
- The intelligent vision-based systems significantly enhance the performance of sewer maintenance processes in comparison to blind systems, which lack the same level of competence.
- The integration of embedded vision technology with AI detectors trained using the S-BIRD dataset provides an efficient and reliable solution for sewer blockage detection, contributing to enhanced wastewater management practices globally.

Overall, this research significantly contributes to the field of AI by providing a representative benchmark dataset, deep neural network-based evaluation frameworks using transfer learning and fine-tuning, and integration of embedded vision approach for sewer blockage detection, thereby enhancing wastewater management practices. The established foundation and findings from this thesis facilitate future advancements in AI technology and its applications. The methodologies and insights presented in this research expand the knowledge in the field and open avenues for further exploration and development in diverse domains.

5.2 Recommendations for Further Work

Based on the achievements and insights gained from this thesis, the following recommendations are suggested for further research:

- For further work, it would be beneficial to explore and incorporate additional AI techniques, such as semantic segmentation, instance segmentation and panoptic segmentation, to enhance the detection and identification tasks.
- It is recommended to explore additional data augmentation techniques and experiment with different backbone architectures to further improve the models' performance.
- As technology advances and new data becomes available, expanding the developed representative dataset would be beneficial. Increasing the dataset's size, incorporating additional needful classes, and challenging scenarios, can further enhance the performance and generalization capabilities of deep neural detection models..
- It is worth exploring other neural network architectures and object detection models that may exhibit varying strengths and weaknesses, leading to improved performance for specific applications.
- Continuous learning or incremental training approaches can be explored to ensure that models remain effective over extended periods.
- Evaluate and update the developed AI models and dataset as new techniques, technologies, and challenges emerge in the field of AI. Continuously strive for improvement in accuracy, efficiency, and robustness to keep the models up-to-date and effective.
- Foster collaboration with industry partners, wastewater management authorities, and researchers to exchange knowledge, share experiences, and explore opportunities for implementing the developed techniques and solutions on a larger scale. Collaborative efforts can accelerate the adoption of AI-based technologies in the different fields.
- The insights and methodologies gained from this research can be implemented in other domains that require computer vision and deep learning techniques such as: environmental monitoring, infrastructure maintenance and public safety, and beyond.

Further research and exploration in the above recommended areas would deepen our understanding and pave the way for continued advancements in the development of robust and efficient AI models, thus propelling the field of AI towards greater innovation and practical applications.

References

- [1] McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4), 12-12.
- [2] Mirats Tur, J. M., & Garthwaite, W. (2010). Robotic devices for water main in-pipe inspection: A survey. *Journal of Field Robotics*, 27(4), 491-508.
- [3] Caffoor, I. (2019) Robotics and Autonomous Systems in the Water Industry. TWENTY65 Report, 18/2/19.).
- [4] Patil, R.R.; Mustafa, M.Y.; Calay, R.K.; Ansari, S.M. S-BIRD: A Novel Critical Multi-Class Imagery Dataset for Sewer Monitoring and Maintenance Systems. *Sensors* 2023, 23, 2966. <https://doi.org/10.3390/s23062966>
- [5] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).
- [6] Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).
- [7] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.
- [8] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
- [9] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (pp. 21-37). Springer International Publishing.
- [10] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988).
- [11] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6569-6578).
- [12] Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 734-750).
- [13] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).
- [14] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- [15] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.
- [16] Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Iii, H. D., & Crawford, K. (2021). Datasheets for datasets. *Communications of the ACM*, 64(12), 86-92.
- [17] Bhardwaj, A.; Karger, D.; Subramanyam, H.; Deshpande, A.; Madden, S.; Wu, E.; Elmore, A.; Parameswaran, A.; Zhang, R. Collaborative data analytics with DataHub. *Proc. VLDB Endow.* 2015, 8, 1916.

- [18] Umbrich, J., Neumaier, S., & Polleres, A. (2015, August). Quality assessment and evolution of open data portals. In 2015 3rd international conference on future internet of things and cloud (pp. 404-411). IEEE.
- [19] Koesten, L., Simperl, E., Blount, T., Kacprzak, E., & Tennison, J. (2020). Everything you always wanted to know about a dataset: Studies in data summarisation. *International Journal of Human-Computer Studies*, 135, 102367.
- [20] Roh, Y.; Heo, G.; Whang, S.E. A survey on data collection for machine learning: A big data-ai integration perspective. *IEEE Trans. Knowl. Data Eng.* 2019, 33, 1328–1347.
- [21] Kaggle. Available online: <https://www.kaggle.com/> (accessed on 22 January 2023).
- [22] 22 [34]Mendeley Data. Available online: <https://data.mendeley.com/> (accessed on 18 January 2023).
- [23] Google Dataset Search.
Available online: <https://datasetsearch.research.google.com/> (accessed on 12 January 2023).
- [24] IEEE DataPort. Available online: <https://iee-dataport.org/dataset> (accessed on 9 January 2023).
- [25] Chapman, A.; Simperl, E.; Koesten, L.; Konstantinidis, G.; Ibáñez, L.D.; Kacprzak, E.; Groth, P. Dataset search: A survey. *VLDB J.* 2020, 29, 251–272.
- [26] European Commission. Digital Agenda: Commission’s Open Data Strategy, Questions and Answers. Memo/11/891. 12 December 2011. Available online: https://ec.europa.eu/commission/presscorner/detail/en/MEMO_11_891 (accessed on 4 January 2023).
- [27] Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., & Starr, J. (2018). Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction*, 91, 273-283.
- [28] Cheng, J. C., & Wang, M. (2018). Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, 95, 155-171.
- [29] Gutierrez-Mondragon, M. A., Garcia-Gasulla, D., Alvarez-Napagao, S., Brossa-Ordoñez, J., & Gimenez-Esteban, R. (2020). Obstruction level detection of sewer videos using convolutional neural networks. arXiv preprint arXiv:2002.01284.
- [30] Halfawy, M. R., & Hengmeechai, J. (2014). Automated defect detection in sewer closed circuit television images using histograms of oriented gradients and support vector machine. *Automation in Construction*, 38, 1-13.
- [31] Yin, X., Chen, Y., Bouferguene, A., Zaman, H., Al-Hussein, M., & Kurach, L. (2020). A deep learning-based framework for an automated defect detection system for sewer pipes. *Automation in construction*, 109, 102967.
- [32] Moradi, S., Zayed, T., & Golkhoo, F. (2018). Automated sewer pipeline inspection using computer vision techniques. In *Pipelines 2018: Condition Assessment, Construction, and Rehabilitation*, pp. 582-587. Reston, VA: American Society of Civil Engineers.
- [33] Kumar, S. S., Wang, M., Abraham, D. M., Jahanshahi, M. R., Iseley, T., & Cheng, J. C. (2020). Deep learning-based automated detection of sewer defects in CCTV videos. *Journal of Computing in Civil Engineering*, 34(1), 04019047.
- [34] Haurum, J. B., & Moeslund, T. B. (2020). A Survey on image-based automation of CCTV and SSET sewer inspections. *Automation in Construction*, 111, 103061.

- [35] Moradi, S., Zayed, T., & Golkhoo, F. (2019). Review on computer aided sewer pipeline defect detection and condition assessment. *Infrastructures*, 4(1), 10.
- [36] Liu, Z., & Kleiner, Y. (2013). State of the art review of inspection technologies for condition assessment of water pipes. *Measurement*, 46(1), 1-15.
- [37] Mirats Tur, J. M., & Garthwaite, W. (2010). Robotic devices for water main in-pipe inspection: A survey. *Journal of Field Robotics*, 27(4), 491-508.
- [38] Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., & Dario, P. (2020). Visual-based defect detection and classification approaches for industrial applications—a survey. *Sensors*, 20(5), 1459.
- [39] Patil, R.R.; Ansari, S.M.; Calay, R.K.; Mustafa, M.Y. Review of the State-of-the-art Sewer Monitoring and Maintenance Systems Pune Municipal Corporation-A Case Study. *TEM J.* 2021, 10, 1500–1508.
- [40] Kirkham, R., Kearney, P. D., Rogers, K. J., and Mashford, J. (2000). PIRAT—a system for quantitative sewer pipe assessment. *The International Journal of Robotics Research*, 19(11), 1033-1053.
- [41] Kuntze, H. B., Schmidt, D., Haffner, H., & Loh, M. (1995, September). KARO-A flexible robot for smart sensor-based sewer inspection. In *Proc. Int. Conf. No Dig'95*, Dresden, Germany, 19, pp. 367-374.
- [42] Kirchner, F., & Hertzberg, J. (1997). A prototype study of an autonomous robot platform for sewerage system maintenance. *Autonomous robots*, 4(4), 319-331.
- [43] Rome, E., Hertzberg, J., Kirchner, F., Licht, U., & Christaller, T. (1999). Towards autonomous sewer robots: the MAKRO project. *Urban Water*, 1(1), 57-70.
- [44] Nassiraei, A. A., Kawamura, Y., Ahrary, A., Mikuriya, Y., & Ishii, K. (2007, April). Concept and design of a fully autonomous sewer pipe inspection mobile robot" kantaro". In *Proceedings 2007 IEEE international conference on robotics and automation*, pp. 136-143.
- [45] Alejo, D., Mier, G., Marques, C., Caballero, F., Merino, L., & Alvito, P. (2020). SIAR: A ground robot solution for semi-autonomous inspection of visitable sewers. In *Advances in Robotics Research: From Lab to Market*, pp. 275-296. Springer, Cham.
- [46] Abidin, A. S. Z., Zaini, M. H., Pauzi, M. F. A. M., Sadini, M. M., Chie, S. C., Mohammadan, S., ... & Ming, C. Y. (2015). Development of cleaning device for in-pipe robot application. *Procedia Computer Science*, 76, 506-511.
- [47] Vaani, I., Sushil, S. J., Kunjamma, U. V., Ramachandran, A., Bai, V. T., & Thyla, B. (2017, May). BhrtArtana (A pipe cleaning and inspection robot). In *2017 Third IEEE International Conference on Sensing, Signal Processing and Security (ICSSS)*, pp. 422-425.
- [48] Gobinath, M., and Malathi, S. (2018, December). Sewage Sludge Removal Method Through Arm-Axis by Machine Robot. In *International Conference on Intelligent Systems Design and Applications* (pp. 345-353). Springer, Cham.
- [49] Nesaian, K. P., & Karthikeyan, M. B. (2012). Design and development of vision based blockage clearance robot for sewer pipes. *IAES International Journal of Robotics and Automation*, 1(1), 64.
- [50] Abro, G. E. M., Jabeen, B., Ajodhia, K. K., Rauf, A., & Noman, A. (2019). Designing Smart Sewerbot for the Identification of Sewer Defects and Blockages. *International Journal of Advanced Computer Science and Applications*, 10(2), 615-619.

- [51] Information Manual—Standard Operating Procedure (SOP) for Cleaning of Sewers and Septic Tanks by Central Public Health & Environmental Engineering Organization (CPHEEO), Ministry of Housing and Urban Affairs, Government of India. Available online:
<http://cpheeo.gov.in/upload/5c0a062b23e94SOPforcleaningofSewersSepticTanks.pdf>
 (accessed on 28 January 2023)
- [52] Available online:
https://cpheeo.gov.in/upload/uploadfiles/files/operation_chapter2.pdf
- [53] Ansari, S. M., Patil, R. R., Calay, R. K., & Mustafa, M. Y. (2023). Introduction To Machine Learning Techniques. In IoT, Machine Learning and Blockchain Technologies for Renewable Energy and Modern Hybrid Power Systems (pp. 93-120). River Publishers.
- [54] Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018.
- [55] Deng, J., Xuan, X., Wang, W., Li, Z., Yao, H., & Wang, Z. (2020, November). A review of research on object detection based on deep learning. In *Journal of Physics: Conference Series* (Vol. 1684, No. 1, p. 012028). IOP Publishing.
- [56] Vaidya, O.S.; Patil, R.; Phade, G.M.; Gandhe, S.T. Embedded Vision Based Cost Effective Tele-operating Smart Robot. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* 2019, 8, 1544–1550. 22.
- [57] Patil, R.R.; Vaidya, O.S.; Phade, G.M.; Gandhe, S.T. Qualified Scrutiny for Real-Time Object Tracking Framework. *Int. J. Emerg. Technol.* 2020, 11, 313–319.
- [58] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YoloX: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430.
- [59] Available online: <https://github.com/ultralytics/yolov5>
- [60] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. *Advances in neural information processing systems*, 27.
- [61] Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717-1724).
- [62] Long, M., Cao, Y., Wang, J., & Jordan, M. (2015, June). Learning transferable features with deep adaptation networks. In *International conference on machine learning* (pp. 97-105). PMLR.
- [63] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- [64] Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., ... & Raskar, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 172-181).
- [65] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378, 686-707.
- [66] Feng, C., Zhang, H., Wang, S., Li, Y., Wang, H., & Yan, F. (2019). Structural damage detection using deep convolutional neural network and transfer learning. *KSCE Journal of Civil Engineering*, 23, 4493-4502.

- [67] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671), 195-202.
- [68] Moen, E., Bannon, D., Kudo, T., Graf, W., Covert, M., & Van Valen, D. (2019). Deep learning for cellular image analysis. *Nature methods*, 16(12), 1233-1246.
- [69] Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., & Walsh, A. (2018). Machine learning for molecular and materials science. *Nature*, 559(7715), 547-555.
- [70] Hino, M., Benami, E., & Brooks, N. (2018). Machine learning for environmental monitoring. *Nature Sustainability*, 1(10), 583-588.
- [71] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).
- [72] Vapnik, V. N. (1995). *The nature of statistical learning Theory*.
- [73] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- [74] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786), 504-507.
- [75] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90.
- [76] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13* (pp. 740-755). Springer International Publishing.
- [77] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [78] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial networks. *arXiv 2014. arXiv preprint arXiv:1406.2661*.
- [79] Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., ... & Amodei, D. (2018). The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*.
- [80] Berrar, D. Cross-Validation. In *Encyclopedia of Bioinformatics and Computational Biology*; Ranganathan, S., Gribskov, M., Nakai, K., Schönbach, C., Eds.; Academic Press: Oxford, UK, 2019; pp. 542–545.
- [81] Online URL for Google Source Images -
- <https://www.drainmasterohio.com/red-flags-of-tree-root-intrusion-in-your-drain-pipes/>
 - <https://arboriculture.files.wordpress.com/2016/02/treerootpipe.jpg>
 - https://fastcdn.impakter.com/wp-content/uploads/2020/05/plastic-bottles-issue.jpg?strip=all&lossy=0&quality=92&sharp=1&w=2560&ssl=1&_gl=1*112iyjw*_ga*ODg3OTYwMzM4LjE2ODgwNDE1NDc.*_ga_FVBES2BYX0*MTY4ODA0MTU0Ni4xLjEuMTY4ODA0MTYzNy4wLjAuMA..*_ga_FQWCH74CDR*MTY4ODA0MTU0Ni4xLjEuMTY4ODA0MTYzNy42MC4wLjA.&_ga=2.224344539.1550272950.1688041547-887960338.1688041547
 - <https://www.istockphoto.com/photo/plastic-bottles-isolated-on-white-gm1202347223-345153972>

- https://spunout.ie/wp-content/uploads/elementor/thumbs/Plastic_bottles_in_the_sea-q0ubkb8pkwa5boeuhpaj6o0v1e8l43mla862l6488o.jpg
 - https://img.hunkercdn.com/750x/cppd/getty/article/117/115/86534645_XS.jpg?type=webp
 - <https://bbwsd.com/wordpress/wp-content/uploads/2018/03/FOG-850x425.jpg>
 - <https://images.squarespace-cdn.com/content/v1/55e97d2de4b0a47f46957437/1499308890029-VM48EFRJMCISOFFHFETV/iStock-482437666.jpg?format=1000w>
 - <https://dependableplastic.com/product/blue-recycling-bags/>
 - <https://sagewater.com/wp-content/uploads/2019/06/Unclogging-The-Mystery-of-Clogged-Pipes-1024x768.jpg>
- [82] Kothari, C. R. (2004). *Research methodology*. new Age.
- [83] Wang, Z., Pei, Y., & Li, J. (2023). A Survey on Search Strategy of Evolutionary Multi-Objective Optimization Algorithms. *Applied Sciences*, 13(7), 4643.
- [84] Jiang, L., Geng, Z., Gu, D., Guo, S., Huang, R., Cheng, H., & Zhu, K. (2023). RS-SVM machine learning approach driven by case data for selecting urban drainage network restoration scheme. *Data Intelligence*, 5(2), 413-437.
- [85] Yazdi, J. (2018). Rehabilitation of urban drainage systems using a resilience-based approach. *Water resources management*, 32, 721-734.
- [86] Cai, X., Shirkhani, H., & Mohammadian, A. (2022). Sensitivity-based adaptive procedure (SAP) for optimal rehabilitation of sewer systems. *Urban Water Journal*, 19(9), 889-899.
- [87] Rathnayake, U. (2015). Migrating storms and optimal control of urban sewer networks. *Hydrology*, 2(4), 230-241.
- [88] Draude, S., Keedwell, E., Kapelan, Z., & Hiscock, R. (2022). Multi-objective optimisation of sewer maintenance scheduling. *Journal of Hydroinformatics*, 24(3), 574-589.

Appendix 1

List of Published Journal Papers

These research articles are relevant to the thesis.

1. **Patil, R. R.**, Calay, R. K., Mustafa, M. Y., & Ansari, S. M. AI-Driven High-Precision Model for Blockage Detection in Urban Wastewater Systems. *Electronics* 2023, 12(17), 3606. (SCIE and Scopus Indexed)
 - DOI: <https://doi.org/10.3390/electronics12173606>
2. **Patil, R. R.**, Mustafa, M. Y., Calay, R. K., & Ansari, S. M. (2023). S-BIRD: A Novel Critical Multi-Class Imagery Dataset for Sewer Monitoring and Maintenance Systems. *Sensors* 2023, 23(6), 2966. (SCIE and Scopus Indexed)
 - DOI: <https://doi.org/10.3390/s23062966>
3. **Patil, R.R.**, Ansari, S.M., Calay, R.K., & Mustafa, M.Y. Review of the State-of-the-art Sewer Monitoring and Maintenance Systems Pune Municipal Corporation-A Case Study. *TEM J.* 2021, 10(4), pp. 1500–1508. (ESCI and Scopus Indexed)
 - DOI: <https://doi.org/10.18421/tem104-02>

Published Book Chapter

4. Ansari, S. M., **Patil, R. R.**, Calay, R. K., & Mustafa, M. Y. Introduction To Machine Learning Techniques. In *IoT, Machine Learning and Blockchain Technologies for Renewable Energy and Modern Hybrid Power Systems*. River Publishers 2023, pp. 93-120. eBook ISBN - 9781003360780. (Book Citation Index and Scopus)
 - DOI: <https://doi.org/10.1201/9781003360780-5>

Other Publications

5. Ansari, S., Khairnar, S. M., **Patil, R. R.**, and Nikalje, N. M. Design Study of Smart Robotic Framework for Sewer Conservation. *International Journal of Engineering Trends and Technology* 2022, 70(8), pp. 247–255. (Scopus Indexed)
 - DOI: 10.14445/22315381/ijett-v70i8p226

6. Ansari, S., Khairnar, S. M., **Patil, R. R.**, & Kokate, R. S. An assessment-water quality monitoring practices and sewer robotic systems. *Information Technology In Industry* 2021, 9(1), 140-148. (ESCI Indexed)
 - DOI: <https://doi.org/10.17762/itii.v9i1.113>

Appendix 2

Creating an implementation of the applied methodology involves following these steps using self-developed programming codes

(a) Implementation of Preprocessing and Augmentation from Scratch

This demonstrates the step-by-step application of different preprocessing and augmentation techniques to an input instance. It includes functions for each technique and displays the original and augmented frames. Additionally, the resulting frames are saved to the 'dataset' directory.

```
import cv2
import numpy as np
from skimage.util import random_noise

# Auto-orientation and resizing
def preprocess_frame(frame):
    # Discard EXIF rotation and validate pixel sort
    # Assuming the frame is already loaded using OpenCV
    # perform EXIF rotation correction if needed

    # Resize the frame to 416x416 pixels
    target_width = 416
    target_height = 416

    original_height, original_width = frame.shape[:2]

    sf_w = target_width / original_width
    sf_h = target_height / original_height

    resized_width = int(original_width * sf_w)
    resized_height = int(original_height * sf_h)

    resized_frame = cv2.resize(frame, (resized_width, resized_height))
```



```

return resized_frame

# Gray scaling
def apply_gray_scale(frame):
    # Convert the frame to grayscale
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Convert the grayscale frame back to BGR (retaining color information)
    gray_frame_bgr = cv2.cvtColor(gray_frame, cv2.COLOR_GRAY2BGR)

    return gray_frame_bgr

# Salt and pepper noise
def apply_salt_and_pepper_noise(frame, noise_percentage):
    # Add salt and pepper noise to the frame
    noisy_frame = random_noise(frame, mode='s&p', amount=noise_percentage)
    # Convert the noisy frame to uint8 format
    noisy_frame = (255 * noisy_frame).astype(np.uint8)

    return noisy_frame

# Random exposure adaptation
def apply_random_exposure_adjustment(frame, min_percent=-25, max_percent=25):
    # Generate a random exposure adjustment factor
    adjustment_factor = np.random.uniform(min_percent / 100, max_percent / 100)
    # Apply the exposure adjustment to the frame
    adjusted_frame = np.clip(frame * (1 + adjustment_factor), 0, 255).astype(np.uint8)

    return adjusted_frame

# Cutout augmentation
def apply_cutout(frame, occlusion_percentage=0.1):
    # Generate three occlusions in random positions

```

```

occlusion_size = int(frame.shape[0] * 0.1) # 10% of frame size
for _ in range(3):
    x = np.random.randint(0, frame.shape[1] - occlusion_size)
    y = np.random.randint(0, frame.shape[0] - occlusion_size)
    frame[y:y+occlusion_size, x:x+occlusion_size] = 0 # Black out the occlusion region

return frame

# Mosaic augmentation
def apply_mosaic(frames):
    # Randomly select four frames
    selected_frames = np.random.choice(frames, size=4, replace=False)
    mosaic_frame = np.zeros_like(selected_frames[0]) # Initialize the mosaic frame

    # Determine the mosaic layout
    layout = [(0, 0), (0, 1), (1, 0), (1, 1)]
    mosaic_height = mosaic_frame.shape[0] // 2
    mosaic_width = mosaic_frame.shape[1] // 2

    # Patch the selected frames into the mosaic frame
    for i, (row, col) in enumerate(layout):
        frame = selected_frames[i]
        y_start = row * mosaic_height
        y_end = y_start + mosaic_height
        x_start = col * mosaic_width
        x_end = x_start + mosaic_width
        mosaic_frame[y_start:y_end, x_start:x_end] = frame

    return mosaic_frame

# Example usage
input_frame = cv2.imread('input_frame.jpg')

```

```

# Preprocess frame
preprocessed_frame = preprocess_frame(input_frame)

# Apply gray scaling
gray_scaled_frame = apply_gray_scale(preprocessed_frame)

# Apply salt and pepper noise
noise_percentage = 0.05 # 5%
noisy_frame = apply_salt_and_pepper_noise(preprocessed_frame, noise_percentage)

# Apply random exposure adjustment
adjusted_frame = apply_random_exposure_adjustment(preprocessed_frame)

# Apply cutout augmentation
cutout_frame = apply_cutout(preprocessed_frame)

# Mosaic augmentation
frames = [preprocessed_frame1, preprocessed_frame2, preprocessed_frame3,
preprocessed_frame4] # Replace with actual frame list
mosaic_frame = apply_mosaic(frames)

# Add resulting frames to the list
# Save resulting frames to the dataset
for i, frame in enumerate(resulting_frames):
    filename = f'dataset/resulting_frame_{i}.jpg'
    cv2.imwrite(filename, frame)

# Display the frames
cv2.imshow('Input Frame', input_frame)
cv2.imshow('Preprocessed Frame', preprocessed_frame)
cv2.imshow('Gray Scaled Frame', gray_scaled_frame)
cv2.imshow('Noisy Frame', noisy_frame)
cv2.imshow('Adjusted Frame', adjusted_frame)

```

```
cv2.imshow('Cutout Frame', cutout_frame)
cv2.imshow('Mosaic Frame', mosaic_frame)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(b) Object count histogram and heatmap implementation

This performs object counting, generates histograms, and creates object heatmaps for each image's annotations. It gives you a starting point for analysing object distributions and generating visualizations based on your image and annotation data.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# Load the training set images and annotations
image_paths = ['image1.jpg', 'image2.jpg', ...]
annotation_paths = ['annotation1.txt', 'annotation2.txt', ...]

images = []
annotations = []

for image_path, annotation_path in zip(image_paths, annotation_paths):
    image = cv2.imread(image_path)
    images.append(image)

    # Assuming annotations are stored in text files as bounding box coordinates
    annotation_data = np.loadtxt(annotation_path)
    annotations.append(annotation_data)

# Perform object counting and generate the object count histogram
object_counts = [annotation.shape[0] for annotation in annotations] # Number of objects in
each image

# Generate the object count histogram
plt.figure(figsize=(8, 6))
sns.histplot(object_counts, bins='auto', kde=True)
```

```

plt.title('Object Count Histogram')
plt.xlabel('Number of Objects')
plt.ylabel('Number of Images')
plt.show()

# Generate the object heatmap for each class
class_names = ['grase', 'plastics', 'treeroots']
class_colors = [(0, 255, 0), (255, 0, 0), (0, 0, 255)] # Green, Blue, Red

heatmaps = []

for annotation, image in zip(annotations, images):
    heatmap = np.zeros_like(image, dtype=np.uint8)

    for bbox in annotation:
        x, y, w, h = bbox.astype(int)
        class_index = int(bbox[-1])
        class_color = class_colors[class_index]

        cv2.rectangle(heatmap, (x, y), (x + w, y + h), class_color, thickness=-1)

    heatmaps.append(heatmap)

# Display the heatmaps
for i, heatmap in enumerate(heatmaps):
    plt.figure(figsize=(8, 6))

    plt.imshow(cv2.cvtColor(heatmap, cv2.COLOR_BGR2RGB))
    plt.title(f'Object Heatmap - Image {i+1}')
    plt.axis('off')
    plt.show()

```

(c) Development of Model-1 using YOLOX, and its Training and Evaluation method in Code Pieces

- i. Configuration structure for S-BIRD dataset from scratch

```
import os
import sys
from addict import Dict
from my_utils.utils import merge_opt # Assume you have a custom utility for merging options

def update_yolox_model(cfg, inp_params):
    # Transfer learning and fine-tuning details
    # Modified model architecture, loss function, and training parameters
    cfg.num_classes = 3 # Number of classes: tree roots, plastics, grease
    cfg.max_epoch = 300
    cfg.learning_rate = 0.01
    cfg.weight_decay = 5e-4
    cfg.random_size = (14, 26)
    # ... (other custom training parameters as per need can be defined and here it is S-BIRD
    dataset)

def main():
    opt = Dict() # Use 'addict' library for configuration

    # Update experiment details and dataset paths
    opt.exp_id = "sewer_blockage_detection"
    opt.dataset_path = "/path/to/s_bird_dataset"

    # Update model details
    opt.backbone = "YOLOX-s"
    opt.input_size = (640, 640)
    opt.random_size = (14, 26)
    opt.test_size = (640, 640)
    opt.num_epochs = 300
```

```

# Update label names and reid_dim
opt.label_name = ['treeroots', 'plastics', 'grease']
opt.reid_dim = 0

# Update training parameters
opt.learning_rate = 0.01
opt.weight_decay = 5e-4
opt.random_size = (14, 26)
opt.degrees = 10.0
opt.translate = 0.1
opt.scale = (0.1, 2)
# ... (other training parameters)

opt, input_params = merge_opt(opt, sys.argv[1:])
opt.num_classes = len(opt.label_name)
opt.gpus_str = opt.gpus
opt.gpus = [int(i) for i in opt.gpus.split(',')]

# Replace the following line with your desired logic
opt.gpus = [i for i in range(len(opt.gpus))] if opt.gpus[0] >= 0 else [-2] # Different logic

opt.root_dir = os.path.dirname(__file__)
opt.save_dir = os.path.join(opt.root_dir, 'exp', opt.exp_id)
if opt.resume and opt.load_model == ":
    opt.load_model = os.path.join(opt.save_dir, 'model_last.pth')

print("\n{} final config: {}\n{}".format("-" * 20, "-" * 20, opt))

update_yolox_model(opt, input_params)

if __name__ == "__main__":
    main()

```


- ii. Development of model for training and validation operation from scratch on corresponding dataset

```
import os
import sys
import datetime
import torch
import torch.optim as optim
from addict import Dict
from my_utils.utils import merge_opt
from my_utils.data_loader import SbirDataset # Replace with your dataset loader
from my_utils.model import YOLOX # Replace with your YOLOX model definition
from my_utils.losses import YOLOXLoss # Replace with your loss function
from my_utils.metrics import calculate_metrics # Replace with your metrics calculation
function

def train_one_epoch(model, dataloader, criterion, optimizer, device):
    model.train()
    total_loss = 0.0

    for batch_idx, (images, targets) in enumerate(dataloader):
        images, targets = images.to(device), targets.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    return total_loss / len(dataloader)

def validate(model, dataloader, device):
```

```

model.eval()
metrics = calculate_metrics() # Implement your metrics calculation function
with torch.no_grad():
    for batch_idx, (images, targets) in enumerate(dataloader):
        images, targets = images.to(device), targets.to(device)

        outputs = model(images)
        metrics.update(targets, outputs)

return metrics.get_metrics()

def main():
    opt = Dict() # Use 'addict' library for configuration
    # ... (initialize opt as shown in the previous code snippet)

    # Initialize dataset and dataloaders
    train_dataset = Sbird_Dataset(opt.dataset_path, train=True) # Implement your dataset class
    train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=opt.batch_size,
shuffle=True)
    val_dataset = Sbird_Dataset(opt.dataset_path, train=False) # Implement your dataset class
    val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=opt.batch_size,
shuffle=False)

    # Initialize YOLOX model
    model = YOLOX(opt.num_classes) # Implement your YOLOX model class
    model.to(device)

    # Initialize loss function and optimizer
    criterion = YOLOXLoss() # Implement your YOLOX loss function
    optimizer = optim.SGD(model.parameters(), lr=opt.learning_rate,
momentum=opt.momentum, weight_decay=opt.weight_decay)

    # Training loop
    for epoch in range(opt.num_epochs):

```

```

start_time = datetime.datetime.now()

train_loss = train_one_epoch(model, train_loader, criterion, optimizer, device)
val_metrics = validate(model, val_loader, device)

end_time = datetime.datetime.now()
elapsed_time = end_time - start_time

print(f'Epoch [{epoch+1}/{opt.num_epochs}] - '
      f'Train Loss: {train_loss:.4f} - '
      f'Validation Metrics: {val_metrics} - '
      f'Elapsed Time: {elapsed_time}')

# Save model checkpoint
if (epoch + 1) % opt.save_epoch == 0:
    checkpoint_path = os.path.join(opt.save_dir, f'model_epoch_{epoch+1}.pth')
    torch.save(model.state_dict(), checkpoint_path)

if __name__ == "__main__":
    main()

```

iii. Implementation of model evaluation from scratch on corresponding dataset

```
import os
import json
import cv2
import numpy as np
import tqdm
import torch
from torchvision.transforms import functional as F
from models.yolox import YOLOX

# Define the paths and parameters
dataset_path = "/data/dataset/S_BIRD_dataset"
annotations_path = os.path.join(dataset_path, "annotations/instances_val.json")
images_dir = os.path.join(dataset_path, "images/val")
model_path = "/path/to/your/model_best.pth"
class_names = ['grease', 'plastics', 'tree roots']

# YOLOX Detector class for inference
class YOLOXDetector:
    def __init__(self, model_path, confidence_threshold=0.001):
        self.model = YOLOX().eval()
        self.model.load_state_dict(torch.load(model_path, map_location=torch.device('cpu')))
        self.confidence_threshold = confidence_threshold

    def preprocess(self, image):
        image_tensor = F.to_tensor(image).unsqueeze(0)
        return image_tensor

    def run_inference(self, images):
        processed_images = [self.preprocess(image) for image in images]
        inputs = torch.cat(processed_images, dim=0)
        with torch.no_grad():
```

```

        outputs = self.model(inputs)
    return outputs

def evaluate():
    detector = YOLOXDetector(model_path)
    num_samples = len(os.listdir(images_dir))

    print("Performing inference on images in {}".format(images_dir))

    results_list = []

    for image_index in tqdm.tqdm(range(num_samples)):
        image_filename = f"{image_index:06d}.jpg"
        image_path = os.path.join(images_dir, image_filename)
        assert os.path.isfile(image_path), "Image not found: {}".format(image_path)

        image = cv2.imread(image_path)
        batch_images = [image]

        batch_outputs = detector.run_inference(batch_images)

        for index in range(len(batch_outputs)):
            output_results = batch_outputs[index].cpu().numpy()

            for result in output_results:
                confidence = result[4]
                if confidence > detector.confidence_threshold:
                    class_index = int(result[5])
                    class_label = class_names[class_index]
                    bbox = result[:4]
                    x_min, y_min, width, height = bbox
                    x_max, y_max = x_min + width, y_min + height
                    results_list.append(

```

```
        {'bbox': [x_min, y_min, x_max, y_max],
         'category_id': class_index + 1, # Assuming class indices start from 1
         'image_id': image_index + 1, # Assuming image indices start from 1
         'score': confidence})

result_file_path = "s_bird_results.json"
with open(result_file_path, 'w') as f_dump:
    json.dump(results_list, f_dump, indent=4)

print("Results saved to:", result_file_path)

if __name__ == "__main__":
    evaluate()
```

- iv. Implementation of Real-time Detection task using multi-threading on embedded platform in given Code

```
import os
import cv2
import threading
import time
from queue import Queue
from custom_models import CustomDetector
from custom_utils import mkdir, get_img_path, vis_result # Define necessary utilities

class ImageProcessingThread(threading.Thread):
    def __init__(self, img_queue, results_queue, detector):
        super(ImageProcessingThread, self).__init__()
        self.img_queue = img_queue
        self.results_queue = results_queue
        self.detector = detector

    def run(self):
        while True:
            image_path = self.img_queue.get()
            if image_path is None:
                break

            img = cv2.imread(image_path)
            results = self.detector.detect_objects(img)
            self.results_queue.put((image_path, img, results))
            self.img_queue.task_done()

def process_images():
    img_dir = "path/to/your/image/directory"
    output = "output_images"
    mkdir(output, rm=True)
```

```

img_list = get_img_path(img_dir, extend=".jpg")
assert len(img_list) != 0, "No images found in {}".format(img_dir)

detector = CustomDetector(model_path="path/to/your/model.pth")

img_queue = Queue()
results_queue = Queue()
num_threads = 4

threads = []
for _ in range(num_threads):
    thread = ImageProcessingThread(img_queue, results_queue, detector)
    thread.start()
    threads.append(thread)

for image_path in img_list:
    img_queue.put(image_path)

img_queue.join()

for _ in range(num_threads):
    img_queue.put(None)
for thread in threads:
    thread.join()

while not results_queue.empty():
    image_path, img, results = results_queue.get()
    print("Processing image:", image_path)

    classes_of_interest = ["grease", "plastic", "treeroots"]
    filtered_results = [res for res in results if res["class_name"] in classes_of_interest and
res["confidence"] > detector.conf_threshold]
    img = vis_result(img, filtered_results)

```



```

save_p = os.path.join(output, os.path.basename(image_path))
cv2.imwrite(save_p, img)
print("Saved image to", save_p)

def detect_realtime():
    detector = CustomDetector(model_path="path/to/your/model.pth")
    classes_of_interest = ["grease", "plastic", "treeroots"]

    cap = cv2.VideoCapture(0) # Open the webcam
    time.sleep(2.0)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame = cv2.resize(frame, (400, 400))
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        cv2.putText(frame, timestamp, (10, frame.shape[0] - 10),
cv2.FONT_HERSHEY_SIMPLEX,
                    0.35, (0, 0, 255), 1)

        # Perform real-time detection using the custom model
        results = detector.detect_objects(frame)
        filtered_results = [res for res in results if res["class_name"] in classes_of_interest and
res["confidence"] > detector.conf_threshold]

        frame = vis_result(frame, filtered_results)
        cv2.imshow("Real-time Detection", frame)

        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break

```

```
cv2.destroyAllWindows()
```

```
cap.release()
```

```
if __name__ == "__main__":
```

```
    process_images_thread = threading.Thread(target=process_images)
```

```
    process_images_thread.start()
```

```
    detect_realtime()
```

```
    process_images_thread.join()
```

(d) Development of Model-2 using YOLOv5, and its Training and Evaluation method in Code Pieces

- i. Development of C3, SPPF, and Conv actual layer types from scratch, and necessary adjustments based on specific developed dataset -

```
import torch
import torch.nn as nn
import time
from torchvision.transforms import Resize, InterpolationMode

# Custom Conv-BN-Activation (CBA) block
class CBA(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, stride, padding,
activation=nn.ReLU()):
        super(CBA, self).__init__()
        self.cba = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding, bias=False),
            nn.BatchNorm2d(out_channels),
            activation
        )

    def forward(self, x):
        return self.cba(x)

# Custom Bottleneck block
class Bottleneck(nn.Module):
    def __init__(self, in_channels, out_channels, width_multiple=1):
        super(Bottleneck, self).__init__()
        c_ = int(width_multiple * in_channels)
        self.c1 = CBA(in_channels, c_, kernel_size=1, stride=1, padding=0)
        self.c2 = CBA(c_, out_channels, kernel_size=3, stride=1, padding=1)

    def forward(self, x):
        return self.c2(self.c1(x)) + x
```

```
# Customized Spatial Pyramid Pooling - Fast (SPPF) layer
```

```
class SPPF(nn.Module):
```

```
    def __init__(self, in_channels, out_channels):
```

```
        super(SPPF, self).__init__()
```

```
        c_ = int(in_channels // 2)
```

```
        self.c1 = CBA(in_channels, c_, kernel_size=1, stride=1, padding=0)
```

```
        self.pool = nn.MaxPool2d(kernel_size=5, stride=1, padding=2)
```

```
        self.c_out = CBA(c_ * 4, out_channels, kernel_size=1, stride=1, padding=0)
```

```
    def forward(self, x):
```

```
        x = self.c1(x)
```

```
        pool1 = self.pool(x)
```

```
        pool2 = self.pool(pool1)
```

```
        pool3 = self.pool(pool2)
```

```
        return self.c_out(torch.cat([x, pool1, pool2, pool3], dim=1))
```

```
# Custom CSPDarknet53 backbone
```

```
class CSPDarknet53(nn.Module):
```

```
    def __init__(self, in_channels, first_out, width_multiple=0.5, depth_multiple=0.33):
```

```
        super(CSPDarknet53, self).__init__()
```

```
        c_ = int(first_out * width_multiple)
```

```
        self.c1 = CBA(in_channels, c_, kernel_size=6, stride=2, padding=2)
```

```
        self.c2 = CBA(c_, c_ * 2, kernel_size=3, stride=2, padding=1)
```

```
        c3_channels = int(c_ * (2 ** depth_multiple))
```

```
        self.c3 = self._make_C3(c_, c3_channels, depth=2, width_multiple=width_multiple)
```

```
        self.c4 = CBA(c3_channels, c3_channels * 2, kernel_size=3, stride=2, padding=1)
```

```
        c5_channels = int(c3_channels * (2 ** depth_multiple))
```

```
        self.c5 = self._make_C3(c3_channels, c5_channels, depth=4, width_multiple=width_multiple)
```

```
        self.c6 = CBA(c5_channels, c5_channels * 2, kernel_size=3, stride=2, padding=1)
```

```
        c7_channels = int(c5_channels * (2 ** depth_multiple))
```

```
        self.c7 = self._make_C3(c5_channels, c7_channels, depth=6, width_multiple=width_multiple)
```

```

c8_channels = int(c7_channels * (2 ** depth_multiple))
self.c8 = CBA(c7_channels, c8_channels, kernel_size=3, stride=2, padding=1)
self.sppf = SPPF(c8_channels, c8_channels)

def _make_C3(self, in_channels, out_channels, width_multiple=1, depth=1):
    layers = []
    for _ in range(depth):
        layers.append(Bottleneck(in_channels, out_channels,
                                width_multiple=width_multiple))
        in_channels = out_channels
    return nn.Sequential(*layers)

def forward(self, x):
    x = self.c1(x)
    x = self.c2(x)
    x = self.c3(x)
    x = self.c4(x)
    x = self.c5(x)
    x = self.c6(x)
    x = self.c7(x)
    x = self.c8(x)
    x = self.sppf(x)
    return x

# YOLOv5s model
class YOLOv5s(nn.Module):
    def __init__(self, first_out, num_classes, anchors, width_multiple=0.5,
                 depth_multiple=0.33):
        super(YOLOv5s, self).__init__()
        self.backbone = CSPDarknet53(in_channels=3, first_out=first_out,
                                     width_multiple=width_multiple, depth_multiple=depth_multiple)
        self.num_classes = num_classes
        self.anchors = anchors
        self.head = self._make_head(first_out, num_classes, anchors)

```

```

def _make_head(self, first_out, num_classes, anchors):
    heads = []
    for in_channels in [first_out * 4, first_out * 8, first_out * 16]:
        heads.append(nn.Conv2d(in_channels, (5 + num_classes) * len(anchors[0]),
                               kernel_size=1))
    return nn.ModuleList(heads)

def forward(self, x):
    x = self.backbone(x)
    outputs = []
    for i, layer in enumerate(self.head):
        out = layer(x[i])
        bs, _, grid_y, grid_x = out.shape
        out = out.view(bs, len(self.anchors[0]), (5 + self.num_classes), grid_y, grid_x)
        res = out.permute(0, 1, 3, 4, 2).contiguous()
        outputs.append(res)
    return outputs

if __name__ == "__main__":
    batch_size = 32
    image_height = 416
    image_width = 416
    num_classes = 3
    anchors = [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]]
    first_out = 48

    x = torch.rand(batch_size, 3, image_height, image_width)
    model = YOLOv5s(first_out=first_out, num_classes=num_classes, anchors=anchors)
    start = time.time()
    out = model(x)
    end = time.time()
    print("Timing Details: {:.2f} seconds".format(end - start))

```

ii. Configuration structure for S-BIRD dataset from scratch –

```
import os
from pathlib import Path
import imgaug.augmenters as iaa
import torch.cuda

# Define root directory
parent_dir = Path(__file__).parent.parent
ROOT_DIR = os.path.join(parent_dir, "datasets", "coco")

# Configuration for the number of classes and class names
num_classes = 3
class_names = ['Grease', 'Plastic', 'Treeroots']

# Model configuration parameters
input_channels = 3
first_output = 48

class_loss_weight = 1.0
object_loss_weight = 1.0

learning_rate = 5e-4
weight_decay = 5e-4

device = "cuda" if torch.cuda.is_available() else "cpu"
image_size = 416

confidence_threshold = 0.01
nms_iou_threshold = 0.6
map_iou_threshold = 0.5

# Custom anchor settings
```

```

custom_anchors = [
    [(10, 13), (16, 30), (33, 23)], # P3/8
    [(30, 61), (62, 45), (59, 119)], # P4/16
    [(116, 90), (156, 198), (373, 326)] # P5/32
]
# Data augmentation using the imgaug library
train_transforms = iaa.Sequential([
    iaa.SomeOf((1, 4), [
        iaa.Multiply((0.8, 1.2)),
        iaa.Flipud(0.5),
        iaa.Fliplr(0.5),
        iaa.Affine(rotate=(-20, 20)),
        iaa.GaussianBlur(sigma=(0.0, 2.0)),
        iaa.CLAHE(),
        iaa.Posterize(1),
        iaa.ChannelShuffle(0.5),
    ])
])
# Custom class list
my_classes = ['Grease', 'Plastic', 'Treeroots']

num_classes = len(my_classes)
class_names = my_classes
# Custom instance
print(f"Number of classes: {num_classes}")
print(f"Class names: {class_names}")

```


iii. Loss computation during training

```
import time
import os
import numpy as np
import torch
import torch.nn as nn
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from custom_utils.training_utils import adjust_multiscale
from custom_utils.bbox_utils import (
    calculate_iou,
    calculate_intersection_over_union,
    custom_non_max_suppression as custom_nms,
)
from custom_utils.visualization import visualize_bounding_boxes
import custom_config
from custom_model import CustomYOLOv5m
from custom_dataset import CustomTrainingDataset
import torch.nn.functional as F

class CustomYOLOLoss:
    def __init__(self, model, rect_training, save_logs=False, filename=None, resume=False):
        self.rect_training = rect_training
        self.mse_loss = nn.MSELoss()
        self.bce_class_loss = nn.BCEWithLogitsLoss(pos_weight=torch.tensor(custom_config.CLASS_PW))
        self.bce_obj_loss = nn.BCEWithLogitsLoss(pos_weight=torch.tensor(custom_config.OBJECT_PW))
        self.sigmoid = nn.Sigmoid()

        self.lambda_class = 0.5 * (model.head.num_classes / 80 * 3 / model.head.num_layers)
        self.lambda_object = 1 * ((custom_config.IMAGE_SIZE / 640) ** 2 * 3 / model.head.num_layers)
```

```

self.lambda_box = 0.05 * (3 / model.head.num_layers)

self.balance = [4.0, 1.0, 0.4]

self.num_classes = model.head.num_classes
self.anchors_d = model.head.anchors.clone().detach()
self.anchors = model.head.anchors.clone().detach().to("cpu")

self.num_anchor_sets = self.anchors.reshape(9, 2).shape[0]
self.num_anchors_per_scale = self.num_anchor_sets // 3
self.S = model.head.stride
self.ignore_iou_threshold = 0.5
self.ph = None
self.pw = None
self.save_logs = save_logs
self.filename = filename

if self.save_logs:
    if not resume:
        log_folder = os.path.join("training_evaluation_metrics", filename)
        if not os.path.isdir(log_folder):
            os.makedirs(log_folder)
        with open(os.path.join(log_folder, "loss.csv"), "w") as f:
            writer = csv.writer(f)
            writer.writerow(["epoch", "batch_idx", "box_loss", "object_loss", "class_loss"])
            print("-----")
            print(f"Training Logs will be saved in
{os.path.join("training_evaluation_metrics", filename, "loss.csv")}")
            print("-----")
            f.close()

def __call__(self, predictions, targets, prediction_size, batch_idx=None, epoch=None):
    # Rest of the code remains the same
    pass

```

```

def build_targets(self, input_tensor, bounding_boxes, prediction_size):
    # Rest of the code remains the same
    pass

def compute_loss(self, predictions, targets, anchors, balance):
    # Rest of the code remains the same
    pass

if __name__ == "__main__":
    calculate_loss = True
    batch_size = 32
    image_size = 416
    strides = [8, 16, 32]

    anchors = custom_config.ANCHORS
    first_output = 48

    model = CustomYOLOv5m(first_output=first_output,
num_classes=len(custom_config.CLASSES), anchors=anchors,
channel_sizes=(first_output * 4, first_output * 8, first_output * 16),
inference=False).to(custom_config.DEVICE)

    model.load_state_dict(state_dict=torch.load("custom_yolov5m.pt"), strict=True)

    dataset = CustomTrainingDataset(num_classes=len(custom_config.CLASSES),
root_dir=custom_config.ROOT_DIR,
transform=custom_config.TRAIN_TRANSFORMS,
train=True, rect_train=True, default_size=image_size,
batch_size=batch_size, bbox_format="coco")

    yolo_loss = CustomYOLOLoss(model, rect_training=dataset.rect_train)

    data_loader = DataLoader(dataset=dataset, batch_size=batch_size, shuffle=False if
dataset.rect_train else True,

```

```
collate_fn=dataset.collate_fn)
```

```
if calculate_loss:
```

```
    for images, bounding_boxes in data_loader:
```

```
        images = images / 255
```

```
        if not dataset.rect_train:
```

```
            images = adjust_multiscale(images, target_shape=image_size, max_stride=32)
```

```
        predictions = model(images)
```

```
        start_time = time.time()
```

```
        loss = yolo_loss(predictions, bounding_boxes, prediction_size=images.shape[2:4])
```

```
        print(loss)
```

```
        """torch.manual_seed(1)
```

```
        images = torch.rand((batch_size, 3, image_size, image_size))
```

```
        #img_idx = torch.arange(batch_size).repeat(3, 1).T.reshape(12, 1)
```

```
        classes = torch.arange(batch_size).repeat(3, 1).T.reshape(12, 1)
```

```
        bounding_boxes = torch.randint(low=0, high=image_size, size=(batch_size * 3, 4)) /
```

```
100
```

```
        labels = torch.cat([bounding_boxes, classes], dim=-1).tolist()
```

```
        print(loss(model(images), labels))"""
```

```
else:
```

```
    for images, bounding_boxes in data_loader:
```

```
        images = images / 255
```

```
        if not dataset.rect_train:
```

```
            images = adjust_multiscale(images, target_shape=image_size, max_stride=32)
```

```
        images = torch.unsqueeze(images[0], dim=0)
```

```
        bounding_boxes = bounding_boxes[0]
```

```
        targets = yolo_loss.build_targets(images, bounding_boxes, images[0].shape[2:4])
```

```
        targets = [torch.unsqueeze(target, dim=0) for target in targets]
```

```
strides = [8, 16, 32]
boxes = cells_to_bboxes(targets, torch.tensor(anchors), strides, list_output=False)
boxes = custom_nms(boxes, iou_threshold=1, threshold=0.7, max_detections=300)

visualize_bounding_boxes(images[0].permute(1, 2, 0).to("cpu"), boxes[0])
```

- iv. Programming Development of model training operation from scratch on corresponding dataset

```
import argparse
import os
import yaml
import torch
import torch.optim as optim
import torch.nn.functional as F
from pathlib import Path
from model import YOLOV5m
from custom_loss import CustomYOLOLoss
from evaluation import YOLOEvaluator
from data_loading import get_data_loaders
from utils import save_checkpoint, load_checkpoint
import config

class ArgumentParser:
    def __init__(self):
        self.parser = argparse.ArgumentParser ()
        self.parser.add_argument("--data", type=str, default="coco", help="Path to dataset")
        self.parser.add_argument("--resume", action='store_true', help="Resuming learning on a saved checkpoint")
        self.parser.add_argument("--load_weights", action='store_true', help="Load pretrained weights")
        self.parser.add_argument("--epochs", type=int, default=100, help="Number of training epochs")
        self.parser.add_argument("--batch_size", type=int, default=16, help="Batch size")
        self.parser.add_argument("--lr", type=float, default=0.001, help="Learning rate")
        self.parser.add_argument("--save_dir", type=str, default="checkpoints", help="Directory to save checkpoints")
        # ... (other arguments)

    def parse(self):
        return self.parser.parse_args()
```

```

class Trainer:
    def __init__(self, model, loss_fn, optimizer, device):
        self.model = model
        self.loss_fn = loss_fn
        self.optimizer = optimizer
        self.device = device

    def train_one_epoch(self, data_loader, epoch):
        self.model.train()
        total_loss = 0.0

        for batch_idx, (data, target) in enumerate(data_loader):
            data, target = data.to(self.device), target.to(self.device)

            self.optimizer.zero_grad()
            outputs = self.model(data)
            loss = self.loss_fn(outputs, target)
            loss.backward()
            self.optimizer.step()

            total_loss += loss.item()

        avg_loss = total_loss / len(data_loader)
        print(f"Epoch {epoch}: Average Loss = {avg_loss:.4f}")

    def train(self, train_loader, epochs):
        for epoch in range(1, epochs + 1):
            self.train_one_epoch(train_loader, epoch)
            # ... (validation and checkpoint saving)

def main():
    args = ArgumentParser().parse()

```

```

if args.data == "coco":
    # Load S-BIRD dataset
    nc = 3
    labels = ['Grease', 'Plastic', 'Tree roots'] # Replace with actual class labels
    anchors = config.ANCHORS
    # ... (other dataset specific settings)
else:
    # Handle other datasets
    pass

model = YOLOV5s(nc=nc, anchors=anchors, ch=(64, 128, 256), inference=False)
model.to(config.DEVICE)

if args.load_weights:
    model.load_state_dict(torch.load("pretrained_weights.pth"))

optimizer = optim.Adam(model.parameters(), lr=args.lr)
loss_fn = CustomYOLOLoss() # Custom loss implementation
evaluator = YOLOEvaluator(model, labels, config.DEVICE)

if args.resume:
    load_checkpoint(model, optimizer, args.save_dir, args.load_weights)

train_loader, val_loader = get_data_loaders(args.data, args.batch_size)

trainer = Trainer(model, loss_fn, optimizer, config.DEVICE)
trainer.train(train_loader, args.epochs)

if __name__ == "__main__":
    main()

```


v. SGD programme for customization

```
import torch
import argparse
from models import YOLOv5s, YOLOXs # Import your model architectures
from loss import CustomYOLOLoss # Import your loss function
from data_loading import get_data_loaders # Import your data loading function
import config

class ArgumentParser:
    def __init__(self):
        self.parser = argparse.ArgumentParser()
        self.parser.add_argument("--model_type", type=str, default="yolov5", help="Type of
model to use (yolov5/yolox)")
        self.parser.add_argument("--data", type=str, default="s_bird", help="Path to dataset")
        self.parser.add_argument("--resume", action='store_true', help="Resume training on a
saved checkpoint")
        self.parser.add_argument("--load_weights", action='store_true', help="Load pretrained
weights")
        self.parser.add_argument("--max_epochs", type=int, default=6000, help="Number of
training epochs")
        self.parser.add_argument("--batch_size", type=int, default=16, help="Batch size")
        # ... (other arguments)

    def parse(self):
        return self.parser.parse_args()

class Trainer:
    def __init__(self, model, loss_fn, optimizer, device):
        self.model = model
        self.loss_fn = loss_fn
        self.optimizer = optimizer
        self.device = device
```

```

def train_one_epoch(self, data_loader, epoch):
    self.model.train()
    total_loss = 0.0

    for batch_idx, (data, target) in enumerate(data_loader):
        data, target = data.to(self.device), target.to(self.device)

        self.optimizer.zero_grad()
        outputs = self.model(data)
        loss = self.loss_fn(outputs, target)
        loss.backward()
        self.optimizer.step()

        total_loss += loss.item()

    avg_loss = total_loss / len(data_loader)
    print(f'Epoch {epoch}: Average Loss = {avg_loss:.4f}')

def train(self, train_loader, max_epochs):
    for epoch in range(1, max_epochs + 1):
        self.train_one_epoch(train_loader, epoch)
        # ... (validation and checkpoint saving)

def main():
    args = ArgumentParser().parse()

    if args.model_type == "yolov5":
        model = YOLOv5s(num_classes=3, depth=0.33, width=0.5) # Customize model
        architecture
    elif args.model_type == "yolox":
        model = YOLOXs(num_classes=3) # Customize model architecture
    else:
        raise ValueError("Invalid model_type")

```

```
if args.load_weights:
    model.load_weights("pretrained_weights.pth") # Load pretrained weights if needed

optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.937,
weight_decay=0.0005)
loss_fn = CustomYOLOLoss(num_classes=3) # Customize loss function
train_loader, val_loader = get_data_loaders(args.data, args.batch_size) # Customize data
loading

trainer = Trainer(model, loss_fn, optimizer, config.DEVICE)
trainer.train(train_loader, args.max_epochs)

if __name__ == "__main__":
    main()
```

- vi. Implementation of Real-time Detection task using multi-threading on embedded platform in given Code

```
import cv2
import numpy as np
import argparse
import os
import torch
from concurrent.futures import ThreadPoolExecutor
from model import YOLOV5small
from utils.utils import load_model_checkpoint
from utils.bboxes_utils import non_max_suppression
from PIL import Image
import configparser
from imutils.video import VideoStream
from imutils.video import FPS
import imutils
import datetime
import time

# Define classes
CLASSES = ['treeroots', 'plastics', 'grease']

def preprocess_image(image_path):
    img = np.array(Image.open(image_path))
    img = img.transpose((2, 0, 1))
    img = img[None, :]
    img = torch.from_numpy(img)
    img = img.float() / 255
    return img

def process_image(image_path, model):
    img = preprocess_image(image_path)
```

```

with torch.no_grad():
    out = model(img.to(device))

    bboxes = cells_to_bboxes(out, model.head.anchors, model.head.stride, is_pred=True,
to_list=False)
    bboxes = non_max_suppression(bboxes, iou_threshold=0.45, threshold=0.25, to_list=False)

return img[0].permute(1, 2, 0).to("cpu"), bboxes

def webcam_inference_thread():
    vs = VideoStream().start()
    time.sleep(2.0)
    fps = FPS().start()

    while True:
        frame = vs.read()
        frame = imutils.resize(frame, width=400)

        timestamp = datetime.datetime.now()
        ts = timestamp.strftime("%A %d %B %Y %I:%M:%S%p")
        cv2.putText(frame, ts, (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX,
            0.35, (0, 0, 255), 1)

        model.conf = 0.80
        model.iou = 0.45
        model.agnostic = False
        model.multi_label = False
        model.classes = None
        model.max_det = 1000
        model.amp = False

        results = model(frame, size=400)
        cv2.imshow("Frame", np.squeeze(results.render()))

```

```

key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
    break

fps.update()

fps.stop()
print("Processed time:", (fps.elapsed()))
print("Counted FPS:", (fps.fps()))

cv2.destroyAllWindows()
vs.stop()

def main():
    parser = argparse.ArgumentParser()

    parser.add_argument("--name_model", type=str, default="model_1", help="Specify the
directory within SAVED_CHECKPOINT")

    parser.add_argument("--checkpoint", type=str, default="checkpoint_epoch_8.pth.tar",
help="Specify the ckpt name within SAVED_CHECKPOINT/ name_model ")

    args = parser.parse_args()

    model = YOLOV5small(first_out=config.FIRST_OUT, nc=len(CLASSES),
anchors=config.ANCHORS,
ch=(config.FIRST_OUT * 4, config.FIRST_OUT * 8, config.FIRST_OUT *
16)).to(device)

    path2model = os.path.join("SAVED_CHECKPOINT", args.model_name, args.checkpoint)
    load_model_checkpoint(model=model, model_name=path2model, training=False)

```

```
with ThreadPoolExecutor(max_workers=2) as executor:  
    executor.submit(webcam_inference_thread)
```

```
while True:  
    if cv2.waitKey(1) & 0xFF == ord('q'):  
        break
```

```
if __name__ == "__main__":  
    main()
```

**Research Article – Review of the State-of-the-art Sewer
Monitoring and Maintenance Systems Pune Municipal
Corporation-A Case Study**

Review of the State-of-the-art Sewer Monitoring and Maintenance Systems Pune Municipal Corporation - A Case Study

Ravindra R. Patil¹, Saniya M. Ansari², Rajnish Kaur Calay¹, Mohamad Y. Mustafa¹

¹ Department of Building, Energy and Material Technology, Faculty of Engineering Science and Technology, UiT The Arctic University of Norway, Narvik, Norway

² Department of E & TC Engineering DYPSOE, Pune, India

Abstract – There is an increasing trend of using automated and robotic systems for the tasks that are hazardous or inconvenient and dirty for humans. Sewers maintenance and cleaning is such a task where robots are already being used for inspection of underground pipes for blockages and damage. This paper reviews the existing robotic systems and various platforms and algorithms along with their capabilities and limitations being discussed. A typical mid-size city in a developing country, Pune, India is selected in order to understand the concerns and identify the requirements for developing robotic systems for the same. It is found that major concern of sewers are blockages but there is not enough information on both real-time detection and removal of it with robotic systems. On-board processing with computer vision algorithms has not been efficiently utilized in terms of performance and determinations for real-world implementations of sewer robotic systems. The review highlights the available methodologies that can be utilized in developing sewer inspection and cleaning robotic systems.

Keywords – sewer monitoring, robotic artifices, review, computer vision, purview, AI techniques

1. Introduction

Sewers are important part of modern sewerage system that discreetly and safely carry waste and storm water away from the buildings to a treatment place. For the whole system to function securely, sewers have to be in good conditions. Regular maintenance and improvement of sewers are essential responsibilities of authorities that operate the system.

There are many practical causes that lead to early deterioration of the sewers. These include blockages, cracks, joint displacement, tree roots intrusions. Failure of sewer may result in large volume of leakage causing environment risk and public health issues. Sewer blockage is a big concern which causes overflowing of dirty water causing foul smell and health risks to people. Thus, a lot of money and manpower are spent by authorities to ensure proper functionality of sewer systems.

Sewer maintenance and cleaning issues have drawn attention of operators and developers around the world. In developing countries like India blockages have been removed by manual cleaning, which is an undignified method and also harmer health hazard for the persons involved. Thus, mechanical and chemical cleaning methods have replaced manual cleaning. Sewer inspection is an important part of sewer maintenance to identify potential problems and resolve them part of routine maintenance program. Over the time automated and robotic systems were developed. Earlier tele-operated robot platforms were controlled by the human operator and connected by cable with an external energy supply (Stein and Niederehe, 1992). Since then, several improvements were made and robotic systems are now widely available for inspection and cleaning of sewer systems. The robotic systems are a preeminent alternatives for navigation and performing a task in the dull, harmful, and unmanned area.

DOI: 10.18421/TEM104-02

<https://doi.org/10.18421/TEM104-02>

Corresponding author: Ravindra R. Patil,
PhD Research Scholar, Department of Building, Energy and Material Technology, Faculty of Engineering Science and Technology, UiT The Arctic University of Norway, Narvik, Norway.


Email: ravindra.r.patil@uit.no

Received: 16 July 2021.

Revised: 17 August 2021.

Accepted: 27 August 2021.

Published: 26 November 2021.

 © 2021 Ravindra R. Patil et al; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDeriv 4.0 License.

The article is published with Open Access at www.temjournal.com

In this paper, the state-of-the-art review on various automated sewer maintenance and inspection systems is presented and future development needs for automated systems are discussed. A case study of Pune Municipal Corporation (PMC) is considered to highlight the specific requirements for a typical metropolitan city in India.

2. Sewerage Systems and Maintenance in India

Indian sewerage system is a huge problem. Traditionally, manual techniques and manual scavenging was used all over India, which used cleaner entering the sewer pipe and septic tanks for cleaning. However, in the last decades the Government of India (GOI) has taken various initiatives to stop hazardous cleaning and to avoid accidents and human casualties during improper practice of cleaning of sewers, septic tanks etc. Only recently the GOI announced measures to end the discriminatory and hazardous practice of manual scavenging by August 2021. Ministry of Housing and Urban Affairs issued Standard Operating Procedure (SOP) For Cleaning of Sewers and Septic Tanks in Nov 2018 [27]. The details for type of inspections and examinations of sewers are provided and recommendations are made for sewer cleaning strategies in the report. However, more funds are required for the organizations responsible for sewerage systems to buy the necessary equipment. Indirect inspection technologies for sewer systems applicable for Indian conditions are identified [27] as shown in Table 1. and Fig.1.

Table 1. Sewer System Inspection Technologies considered applicable to Indian conditions

No	Viability			
	Technology	Sewer Material	Sewer state	Sewer Dimension
a)	Sonar Technique	varying	Completely carrying	Varied Dimension
b)	Technique of Light and Mirror	varying	Vacant	ready for 300 mm
c)	CCTV	varying	Vacant	Varied Dimension

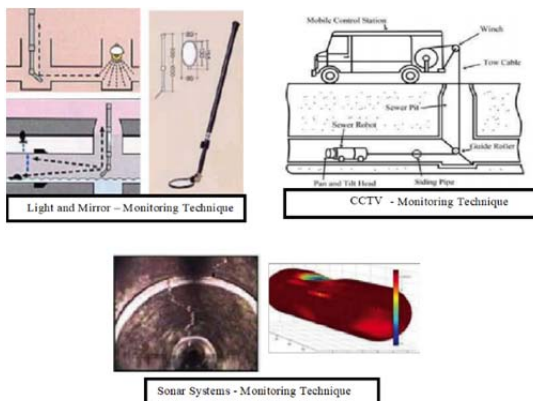


Figure 1. Sewer Inspection Technologies

Table 2. shows the cleaning techniques used in Indian conditions as recommended in [27].

Case Study - Pune Municipal Corporation

The first sewer system was constructed in Pune city in 1928. This system was designed for 31.8 MLD to cater to the ultimate design capacity for population of 0.26 million in the year 1951. Population of the city grew to 7.4 million in 2020. To date, there are 11 sewerage treatment plants (STP) that process 396 MLD in the city. A survey was conducted to assess the current provisions for treating sewerage in the city and issues relating to operating of the system. Table 3. shows the basic data and available tools in the municipality for maintaining sewers pipes.

Table 2. Sewer Cleaning Techniques

Sewer cleaning Techniques				
Labour-intensive Practices	Cloth Ball and Manila Rope	Automated Practices	Gully Emptier	
			Hydraulically Driven Tactics	
			Bucket Machine	
	A collected wood board - Scraper		Rodding Machine with Flexible Sewer Rods	
			Sectional Rods for Sewer	Speedy cleaners (Jetting Machines)
				Dredger (Clam-shell)

Table 3. PMC Surveyed Data

Terms	Details
Sewer Line	2167 kilometre
Sewer Pipe Diameter	Ranges from 100 mm to 1800 mm
Total Chambers (manhole)	2187
Sewer Pipe Material	<ul style="list-style-type: none"> RCC High-density polyethylene (HDPE) bid-iron PVC
Distance Between Chambers	10 to 15 meters
Sewer Net pressure	1 to 4
Sewer Cleaning Techniques	<ul style="list-style-type: none"> Suction Cum Jetting Machine with a Recycler Suction Cum Jetting Machine Jetting Machine
Total Generated Sewage	744 MLD
Intermediate pump stations (IPS)	6
Sewage Treatment Plants (STPs)	9
Main Sewer Lines	<ul style="list-style-type: none"> Below road River side Canal side
Cleaning Tools	Charges/Shift (8 hours shift)
<ul style="list-style-type: none"> Suction Cum Jetting Machine 	6400 INR
<ul style="list-style-type: none"> Suction Cum Jetting Machine with a Recycler 	37000 INR
<ul style="list-style-type: none"> Jetting Machine 	5360 INR

Figure 2. shows some of the real incidents of cleaning operation in the city.

It is evident that the mechanical cleaning is mainly used. During interviews with the officials, it was revealed that their goal of maintenance of the sewers is to reduce the number of sewer blockages per unit length. Therefore, inspection and scheduled cleaning is very important part of sewer maintenance. The PMC tries to follow recommended government guidelines for regular inspection and cleaning of the sewers but reliable techniques and tools are not available.



Figure 2. Visible outturns at PMC survey site

There are several GOI schemes to upgrade the technology for cleaning sewers across India, the PMC officials informed that due to budget restraints they do not have adequate tools.

Existing machines use suction method and jetting to carry dirt out of sewers and pure pipes with lofty-pressure jets of water. At places mainly in densely populated areas, the machines are often too big to enter some narrow streets sometimes cleaning is manually performed. In such scenarios small and portable robotic system would be ideal.

The robotic systems also have cameras for locating the blockages and help the cleaning arm navigate toward it. In the next section advances in the robotic system are discussed.

3. Features of Various Robotic & Automated Systems

Robotic systems are classified as no-autonomy, semi-autonomy, and full-autonomy and are capable for detecting and measuring damage and cleaning. The CCTV (Closed Circuit Television), SSET (Sewer Scanner and Evaluation Technology), Laser Scanning are different techniques which are used for sewer pipe inspection. Also, the computer vision is extending its power with AI revolution on embedded platform.

Many sewer robotic systems such as PIRAT, KARO, KURT, MAKRO, KANTARO, and SIAR are reported by many researchers as explained in the following sections.

Kirkham et al. [1] developed PIRAT (Pipe Inspection Real-Time Assessment Technique) sewer inspection semi-autonomous tethered system that could evaluate the physical data using some interpretation technique. AI techniques were developed system to find out and categorize damages using the three-dimensional model data. A human operator had to find out real damages, as well as the damaged regions in the images marked manually. The system is a decade old with employed algorithms, and the performance parameters are poor.

Kuntz et al. [2] presented tethered, semi-autonomous KARO (KANalROboter) sewer inspection equipment which was capable for auto-correction of tilting pose and slippage in wheel. Pipe bends, larger cracks in pipe, and obstacles within the pipe were identified by a 3D optical sensor and a microwave sensor. This means that the robotic system was mostly dependent on sensors and read data.

The PIRAT and KARO both had main control routines on a computer in the movable control unit and did not comprise on-board hardware.

Kirchner and Hertzberg progressed six-wheeled, untethered KURT (KANal-Untersuchungs-Roboter-Testplattform) for autonomous navigation in a dry sewers test net in [3]. KURT1 was competent to classify a pipe junction type and this patented method was complimented as probabilistic mapping of objects, similar to sewer landmarks. The new KURT2 included sensors for odometry and inclinometers, ultrasound distance or infrared transducers for obstacle detection, and optional bumpers. In this, sensors may not work in a real sewer pipe due to dirt covering. Also, ultrasound sensors are too large in size. The overall reliability of this robotic system is sensor dependent and only inspects the sewers and has no ability to solve issues. Rome et al. [4] came up with an untethered, self-steering MAKRO (Mehrsegmentiger Autonomer KanalROboter) robot for fully autonomous

navigation in roughly cleaned sewer pipes. It carried all resources on-board. In this, the ultrasound range sensor was exploited to detect obstacles that block the pipe. All tasks such as collision avoidance, movement control, obstacle detection, and landmark detection were done by the sensors. The computer vision algorithm or methodology was not clearly present and focused only on applications of sensors.

Nassiraei et al. developed KANTARO, a fully autonomous, un-tethered, passive-active intelligent robot having intelligent modular architecture involved in mechanism and sensor [5]. They also proposed a small and smart 2D laser scanner for directional landmarks detection and utilized the fish eye camera to assess pipe condition and defect detection. They proposed a horizontal and vertical similarity approach for automated faults detection in sewer pipes using images. In this work, the accuracy of faults detection software was not high enough as needed. Alejo et al. introduced SIAR (Sewer Inspection Autonomous Robot), a system that can detect critical structural defects in sewer pipelines by employing 3D structure reconstruction in real-time and also take water or gas samples of the environment for further analysis [6]. This robotic system comprises RGB-D sensors with a powerful wireless communication system.

Abidin developed an in-pipe robot for cleaning soft and moderate clog [7]. The ultrasonic sensor was used to detect diameter difference that means if the detected diameter is small then it will be considered that blockages are present inside the pipe. In this, the cleaning operation was performed when the detected distance is less than 30mm. This system was not capable to remove stubborn clog. The development was lab scale based on very basic experiment and there was not waterproofing feature for real-time application.

Vaani et al. [8] developed an automated sewer robot named as BhrtyArtana where ‘Bhrtya’ stands for robot and ‘Artana’; stands for waste. This robot was capable of inspecting cracks, corrosion, and obstacles as well as clearing any blockage within it. A camera was installed to get real-time video feed for analysis and a proximity sensor was connected to detect obstacle in front of it so that the turbine will start cutting and clearing the obstacle. The implemented prototype did not have intelligence of automated defect detection feature. It is sensor dependent for obstacle detection.

Gobinath and Malathi implemented a Machine Robot having a Robot-Arm [9]. That Robot-Arm was utilized by a few Axis with Stepper Motor to progress with distinct angles from left to right and then from top to bottom. An LCD was used to display the sewage cleaning process. In this, toxic

gases were detected by a board of SewerSnort gas sensor with a MicaZ mote. The developed system does not comprise camera-based automated defect detection and depends on the sensory network. It is costly and needs modification for the real-world prototype.

Prasad and Karthikeyan executed a robot for cleaning and removing the blockage in large sewer pipes [10]. The blockages were detected by ultrasonic sensors and cleaned by a drilling mechanism. A MATLAB tool was used for monitoring video and captured images from a wireless camera. The developed mechanism was not advanced and did not utilize computer vision excepting video feed from the camera.

Abro et al. conferred an autonomous sewerbot that detected the defects in sewerage pipelines as well as blockages using digital image processing [11]. They also investigated the attributes of a specific sewerage line utilizing IoT. The gradient and segmentation techniques were applied for sewer pipe blockage detection with a wireless camera. Overall, they tried to solve all inspection issues but the developed algorithm and performance were inferior for real-world implementation. Table 4. shows the confines and respective remarks for the illustrated robotic artifices.

Table 4. Implemented robotic artifices with their confines and remarks

Robotic Artifices	Confines and Remarks	Ref. No.
PIRAT	No main control routines onboard and reliability depends on human operator	[1]
KARO	Reliability depends on human operator and fully sensory data	[2]
KURT	Fully sensory system and sometimes do not work due to environmental aspects	[3]
MAKRO	Lack of efficient Computer vision methodology and focused only on sensors applications. No ability to move inside of bending pipe.	[4]
KANTARO	Low accuracy of faults detection software, absence of methodical approach to amend practically	[5]
SIAR	Advanced system but having no ability to clear and reform pipe condition in real-time	[6]
In-pipe Robot	Very basic prototype and cannot be accessed for real-time applications.	[7]
BhrtyArtana	No efficient methodology for defect detection and removal for real-time applications	[8]
Machine Robot	Very costly and needs modification in comprised techniques for the real-world prototype	[9]
MATLAB Based Robot	Poor computer vision technique	[10]
Sewerbot	The algorithm and performance were inferior for real-world implementation	[11]

The key differences between types of sewer robotic systems have been depicted in depth in the following Table 5.

Table 5. Differences between types of sewer robotic systems

No-autonomy	Semi-autonomy	Full-autonomy
entirely teleoperated	teleoperated with some amount of self-intelligence	full intelligence for self-navigation
tethered	may be tethered or un-tethered	un-tethered
assessment reliability depends on human operator	assessment reliability depends on both human operator and system intelligence	assessment reliability depends on system intelligence
less sensory system and simply driven by human operator	involve moderate sensors with moving assembly	comprises several sensors and critical moving assembly
fine in small diameter pipes	preferable in small diameter pipes	not trustworthy in small diameter pipes
control unit at remote location	may fetch all obligatory resources onboard or control unit may be at remote location.	fetches all obligatory resources onboard

The robots working in the pipes are categorized depending on their moving techniques as shown in Figure 3.

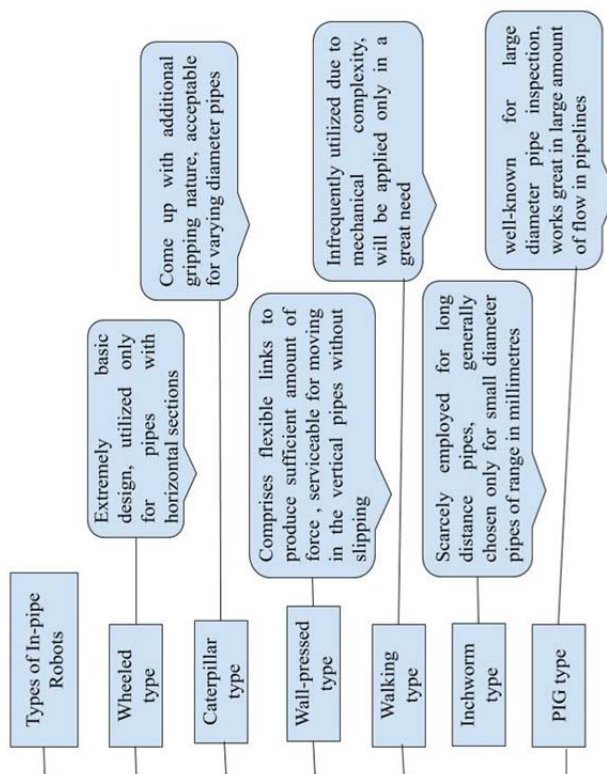


Figure 3. Categorizations of robots based on moving techniques

4. Initiated Computer Vision Algorithms and Perusal

Here, some identified methodologies are discussed for their influence and limitations. Kumar and Abraham made a contribution of the framework that applies Deep Convolution Neural Networks (CNNs) to classify various issues such as root intrusions, cracks, and deposits in sewer CCTV frames [12]. They trained and assessed CNNs using 12,000 frames gathered from over 200 pipelines for accuracy, precision, and recall. It is observed that generated consequences are from images and not from real-time navigation and various defects have been classified and not detected with locations.

Cheng and Wang initiated an automated approach for identification of sewer pipe faults centred on faster R-CNN [13]. In this, 3000 images of sewer pipes captured from CCTV inspection videos were applied for training the detection model. Then the model was analysed for detection accuracy and calculation cost by consuming missing rate, mean average precision (MAP), training time, and detection speed. This approach only functions for standing frames and not for the real-time video feed. It also consists of a few incorrect classifications for cracks in the experiments. Gutiérrez-Mondragón et al. originated a technique to train a Convolutional Neural Network for detecting the obstruction level in pipes [14]. By gathering video database from CCTV, they generated useful frames to train the model. They integrated the Layerwise Relevance Propagation explainability technique for understanding the neural network behaviour for this task. It has been predicted that the proposed system can provide greater accuracy, speed, and consistency for sewer examination in real-time. This work only focused on the quantity of obstruction in the sewers and not on type and locations.

Halfawy and Hengmeechai mentioned a methodical algorithm of HOG (histograms of oriented gradients) and SVM (support vector machine) to find tree root intrusions' defects in images collected from conventional CCTV inspection videos [15]. This was two steps processed as: (1) image segmentation to extract regions of interest (ROI) showing defect areas and (2) classification of the ROI using SVM classifier trained by the HOG features. Here, the algorithm was applied only on static images and not on a video sequence and larger data sets. Yin et al. proposed a framework for real-time automated defect detection in sewer pipe by using the CNN based YOLOv3 object detector [16]. The model had been trained with a data set of 4056 images that includes six types of defects such as broken, hole, deposits, crack, fracture, and root and one type of construction

feature tap. The proposed model had not been tested in real-time in the sewer pipe and it needs some improvisation in performance parameters.

Moradi et al. presented an automated sewer pipeline inspection and condition assessment method using computer vision techniques [17]. In this, a region of interest (ROI) of sewer defects was identified first and then classification was done on frames. The hidden Markov models (HMM) had been used to extract frames from sewer CCTV videos and CNN was proposed to detect the defects and classify them. This work was also based on dataset testing with average results.

Kumar et al. evaluated a deep learning-based framework such as single-shot detector (SSD), you only look once (YOLO), and faster region-based convolutional neural network (Faster R-CNN) for speed and accuracy in classifying and localizing root intrusions and deposits in sewer CCTV images [18]. For training and testing of the models, 3800 annotated images of defects were used. Here, the Faster R-CNN model had the highest accuracy for defects detection and the slowest speed for processing each image. The YOLOv3 model presented a slightly lower accuracy than the Faster R-CNN and was nearly twice as fast as the Faster R-CNN to treat every frame. The SSD model appeared to have the lowest accuracy but the highest speed to process each image. On average in this research, the incorporated dataset of training and testing was very little to attain expected consequences. Also, there is a need to enhance the speed and accuracy of the prototype.

5. Review of Earlier Surveys

Haurum and Moeslund surveyed the last 25 years of research for sewer inspection. They presented a detailed outline inside the field of image-based automation of Closed-Circuit Television (CCTV) and Sewer Scanner and Evaluation Technology (SSET) for sewer inspection [19]. A review was also performed of the pipeline algorithmic, and datasets and protocols. Authors investigated all aspects of automated inspection pipeline such as image acquisition, preprocessing, detection and segmentation, feature description, classification, and temporal filtering. From the survey, it is suggested that free and publicly available datasets should be created, should have open-source code for each publication and standardized evaluation metrics. Moradi et al. reviewed the current state of sewer pipeline inspection technology associated with computer vision and machine learning techniques [20]. The assessment compared advantages and

disadvantages of one and all methods. The image preprocessing, Image representation and Learning have been deeply examined for defect detection in sewer pipe. In this, it is highlighted that CCTV cameras must be standard, must have influential hardware with lofty specifications as well as standard dataset and robust algorithms.

Liu and Kleiner explored the techniques for pipe inspection and for assessing the condition of water distribution and transmission pipes [21]. In their paper they also discuss various technologies such as smart pipe, augmented reality, and intelligent robots and scrutinized for their performance and real-world relevance. They also shed light on the significance of the CCTV and laser scanning techniques. Tur and Garthwaite reviewed existing robotic tools and noticed unclick problems for development of a successful robotic sewer pipe inspection device [22]. Types and mechanisms of robotic systems, acquired sensing technology, and the CCTV and SSET techniques for visual perceptions have been highlighted. They discussed principal affairs of communication, data management, and energy sources. The robots should be implemented for performing specific tasks so that these robotic systems will be cheap and will consume less energy to move.

Czimmermann et al. focused on automated visual-based defect detection methods appropriate to materials such as metals, ceramics and textiles [23]. They pointed to two types of defects such as visible and palpable. They also described acutely artificial visual processing techniques, supervised and non-supervised classifiers and deep learning algorithms for detection and classification of defects. It is noticed that the inadequate test samples, mostly incompatible database, and not developed concrete algorithms are the issues for a perfect inspection system.

Following are the common sewer affairs that are considered in earlier research papers as shown in Figure 4.

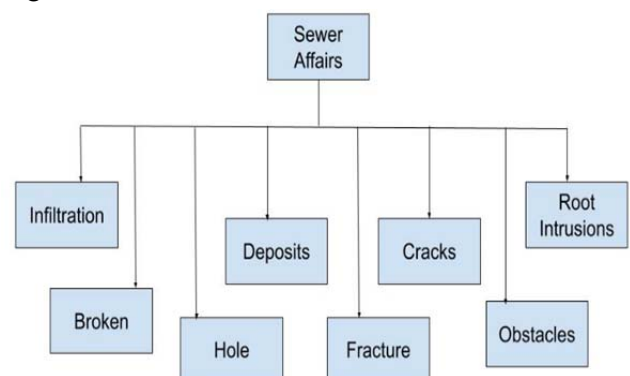


Figure 4. Appeared sewer affairs in the research work

Out of all these sewers issues, sewer blockages are the major issues. Significant causes of sewers blockages due to accumulated debris are identified as follows –

- sand
- silt (i.e., sludge)
- plastic
- grease
- roots and leaves
- rocks
- toiletries waste (such as clip on toilet freshener holders)
- foreign objects such as baby diapers and wipes, tampons, oil, sanitary napkins, cat litter, cotton balls, hair, children’s toys etc.

There is no reliable general algorithm and robotic system formulated for both identification and removal of different sewer blockages in real-world scenario.

6. Sewer Monitoring Techniques

A. Modern computer vision techniques

The most of the area of computer vision is untouchable by the AI techniques. This area comprises intelligent algorithms to return evocative information from frames and videos. These conventional computer vision algorithms are enough to produce admissible output for lower imagery data but outcomes of these algorithms get saturated for larger datasets. At this point, machine learning and deep learning techniques confer sublime outturns. The machine learning techniques are handcrafted algorithms whereas deep learning techniques use deep neural networks for solving classification and regression problems. The deep leaning models need a large number of images for enhancement in accuracy [28]. The features selection and training platforms are also an important aspect in object detection and classification tasks [29], [30]. The precision rate and efficiency of these AI techniques depend on the quality of imagery data.

In below Figures 5. and 6., the general mechanism in machine learning and in deep learning strategies have been depicted for identification of sewer affairs.

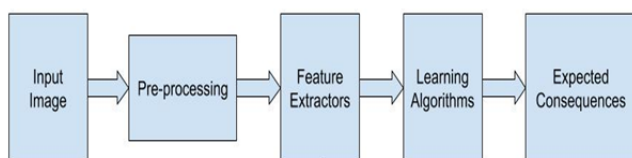


Figure 5. Mechanism in Machine learning strategy

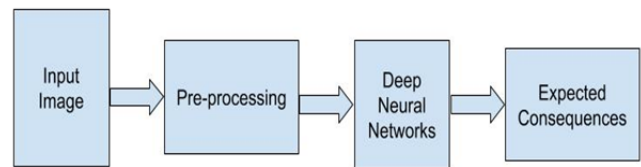


Figure 6. Mechanism in Deep learning strategy

In Table 6., the acquired methodologies in object detection tasks have been listed due to their significance and involvement in the earlier research work.

Table 6. Crucial methodologies

Conventional Algorithms for pre-processing and detection task in Computer Vision	<ul style="list-style-type: none"> ▪ Colour spaces ▪ Image stitching, mosaicking, and unwrapping ▪ Thresholding ▪ Noise removing ▪ Morphological operations ▪ Image enhancement and filtering ▪ Geometric transformations etc.
Learning and Classification Techniques in Machine Learning	<ul style="list-style-type: none"> ▪ SVM ▪ k-means ▪ k-NN ▪ Decision Trees ▪ Logistic Regression ▪ Random Forests ▪ Naïve Bayes
Deep Learning based object detection modules	<ul style="list-style-type: none"> ▪ SSD VVG ▪ YOLOv3 ▪ Faster-RCNN ▪ Tiny YOLOv2
Classifiers in Deep Learning	<ul style="list-style-type: none"> ▪ GoogleNet ▪ AlexNet ▪ CaffeNet ▪ ResNet – 18v1, ResNet – 50v1 ▪ ZFNet 512 ▪ MobileNet v2 ▪ SqueezeNet ▪ ShuffleNet ▪ DenseNet 121 ▪ CNN Mnist
Segmentation Deep Neural Network Modules	<ul style="list-style-type: none"> ▪ Mask R-CNN ▪ FCN ▪ ResNet 101_DUC_HDC ▪ ENet

The robotic systems need to be energy efficient and cost effective for realistic applications and it depends on selection process of finest hardware and software combinations [24]. In this, the embedded platform is the foremost optative with computer vision methodologies for real-world visual implementations [25], [26]. So, embedded vision is a spacious area of research for pragmatic evolution in diverse fields.

7. Conclusion

In this review, the existing sewer robotic systems are analysed for features, resorted frameworks and mechanisms. Overall, it is concluded that the sewers blockages are the predominant issues of buried infrastructure. The earlier and modern techniques for unblocking sewers are discussed with particular reference to the PMC cleaning and maintaining techniques of the sewer infrastructure.

The review of published research results revealed that computer vision algorithms with on-board processing are not efficiently utilized. To the authors' knowledge, no robust algorithm and robotic system available for both real-time detection and removal of sewer pipe blockages exists to date. This presents a research opportunity to develop such algorithm that may be integrated with existing or newer robotic systems for inspecting and cleaning of sewer systems.

Acknowledgements

Thanks to the SPRING Eu-India Project and UiT The Arctic University of Norway, Narvik, Norway for PhD studies of Ravindra R. Patil (No. 821423 and GOI No. BT/IN/EU-WR/60/SP/2018).

The publication charges for this article have been funded by a grant from the publication fund of UiT The Arctic University of Norway.

References

- [1]. Kirkham, R., Kearney, P. D., Rogers, K. J., & Mashford, J. (2000). PIRAT—a system for quantitative sewer pipe assessment. *The International Journal of Robotics Research*, 19(11), 1033-1053.
- [2]. Kuntze, H. B., Schmidt, D., Haffner, H., & Loh, M. (1995, September). KARO-A flexible robot for smart sensor-based sewer inspection. In *Proc. Int. Conf. No Dig'95, Dresden, Germany*, 19 (pp. 367-374).
- [3]. Kirchner, F., & Hertzberg, J. (1997). A prototype study of an autonomous robot platform for sewerage system maintenance. *Autonomous robots*, 4(4), 319-331.
- [4]. Rome, E., Hertzberg, J., Kirchner, F., Licht, U., & Christaller, T. (1999). Towards autonomous sewer robots: the MAKRO project. *Urban Water*, 1(1), 57-70.
- [5]. Nassiraei, A. A., Kawamura, Y., Ahrary, A., Mikuriya, Y., & Ishii, K. (2007, April). Concept and design of a fully autonomous sewer pipe inspection mobile robot" kantaro". In *Proceedings 2007 IEEE international conference on robotics and automation* (pp. 136-143). IEEE.
- [6]. Alejo, D., Mier, G., Marques, C., Caballero, F., Merino, L., & Alvito, P. (2020). SIAR: A ground robot solution for semi-autonomous inspection of visitable sewers. In *Advances in Robotics Research: From Lab to Market* (pp. 275-296). Springer, Cham.
- [7]. Abidin, A. S. Z., Zaini, M. H., Pauzi, M. F. A. M., Sadini, M. M., Chie, S. C., Mohammadan, S., ... & Ming, C. Y. (2015). Development of cleaning device for in-pipe robot application. *Procedia Computer Science*, 76, 506-511.
- [8]. Vaani, I., Sushil, S. J., Kunjamma, U. V., Ramachandran, A., Bai, V. T., & Thyla, B. (2017, May). BhrtArtana (A pipe cleaning and inspection robot). In *2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS)* (pp. 422-425). IEEE.
- [9]. Gobinath, M., & Malathi, S. (2018, December). Sewage Sludge Removal Method Through Arm-Axis by Machine Robot. In *International Conference on Intelligent Systems Design and Applications* (pp. 345-353). Springer, Cham.
- [10]. Nesaian, K. P., & Karthikeyan, M. B. (2012). Design and development of vision based blockage clearance robot for sewer pipes. *IAES International Journal of Robotics and Automation*, 1(1), 64.
- [11]. Abro, G. E. M., Jabeen, B., Ajodhia, K. K., Rauf, A., & Noman, A. (2019). Designing Smart Sewerbot for the Identification of Sewer Defects and Blockages. *International Journal of Advanced Computer Science and Applications*, 10(2), 615-619.
- [12]. Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., & Starr, J. (2018). Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction*, 91, 273-283.
- [13]. Cheng, J. C., & Wang, M. (2018). Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, 95, 155-171.
- [14]. Gutierrez-Mondragon, M. A., Garcia-Gasulla, D., Alvarez-Napagao, S., Brossa-Ordoñez, J., & Gimenez-Esteban, R. (2020). Obstruction level detection of sewer videos using convolutional neural networks. *arXiv preprint arXiv:2002.01284*.
- [15]. Halfawy, M. R., & Hengmeechai, J. (2014). Automated defect detection in sewer closed circuit television images using histograms of oriented gradients and support vector machine. *Automation in Construction*, 38, 1-13.
- [16]. Yin, X., Chen, Y., Bouferguene, A., Zaman, H., Al-Hussein, M., & Kurach, L. (2020). A deep learning-based framework for an automated defect detection system for sewer pipes. *Automation in construction*, 109, 102967.
- [17]. Moradi, S., Zayed, T., & Golkhoo, F. (2018). Automated sewer pipeline inspection using computer vision techniques. In *Pipelines 2018: Condition Assessment, Construction, and Rehabilitation* (pp. 582-587). Reston, VA: American Society of Civil Engineers.
- [18]. Kumar, S. S., Wang, M., Abraham, D. M., Jahanshahi, M. R., Iseley, T., & Cheng, J. C. (2020). Deep learning-based automated detection of sewer defects in CCTV videos. *Journal of Computing in Civil Engineering*, 34(1), 04019047.

- [19]. Haurum, J. B., & Moeslund, T. B. (2020). A Survey on image-based automation of CCTV and SSET sewer inspections. *Automation in Construction*, *111*, 103061.
- [20]. Moradi, S., Zayed, T., & Golkhoo, F. (2019). Review on computer aided sewer pipeline defect detection and condition assessment. *Infrastructures*, *4*(1), 10.
- [21]. Liu, Z., & Kleiner, Y. (2013). State of the art review of inspection technologies for condition assessment of water pipes. *Measurement*, *46*(1), 1-15.
- [22]. Tur, J. M. M., & Garthwaite, W. (2010). Robotic devices for water main in-pipe inspection: A survey. *Journal of Field Robotics*, *4*(27), 491-508.
- [23]. Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., & Dario, P. (2020). Visual-based defect detection and classification approaches for industrial applications—a survey. *Sensors*, *20*(5), 1459.
- [24]. Carabin, G., Wehrle, E., & Vidoni, R. (2017). A review on energy-saving optimization methods for robotic and automatic systems. *Robotics*, *6*(4), 39.
- [25]. Vaidya, O. S., Patil, R., Phade, G. M., & Gandhe, S. T. (2019). Embedded Vision Based Cost Effective Tele-operating Smart Robot. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *8*(7), 1544-1550.
- [26]. Patil, R. R., Vaidya, O. S., Phade, G. M., & Gandhe, S. T. (2020). Qualified Scrutiny for Real-Time Object Tracking Framework. *International Journal on Emerging Technologies*, *11*(3), 313-319.
- [27]. Information Manual.(2018). Standard Operating Procedure (SOP) for Cleaning of Sewers and Septic Tanks by Central Public Health & Environmental Engineering Organisation (CPHEEO), Ministry of Housing and Urban Affairs, Government of India. Retrieved from: <http://cpheeo.gov.in/upload/5c0a062b23e94SOPforcleaningofSewersSepticTanks.pdf> [accessed: 20 June 2021].
- [28]. Salem, M. S. H., Zaman, F. H. K., & Tahir, N. M. (2021). Effectiveness of Human Detection from Aerial Images Taken from Different Heights. *TEM Journal*, *10*(2), 522–530. <https://doi.org/10.18421/TEM102-06>
- [29]. Vandana, C. P., & Chikkamannur, A. A. (2021). Feature Selection: An Empirical Study. *International Journal of Engineering Trends and Technology*, *69*(2), 165-170.
- [30]. Wwwwtkmrndb, W., Isuru, J., & Premaratne, S. (2021). Modeling abandoned object detection and recognition in real-time surveillance. *International Journal of Engineering Trends and Technology*, *69*(2), 188–193. <https://doi.org/10.14445/22315381/IJETT-V69I2P226>

**Research Article – S-BIRD: A Novel Critical Multi-Class Imagery
Dataset for Sewer Monitoring and Maintenance Systems**

Article

S-BIRD: A Novel Critical Multi-Class Imagery Dataset for Sewer Monitoring and Maintenance Systems

Ravindra R. Patil ^{1,*} , Mohamad Y. Mustafa ¹ , Rajnish Kaur Calay ¹ and Saniya M. Ansari ²¹ Faculty of Engineering Science and Technology, UiT The Arctic University of Norway, 8514 Narvik, Norway² Department of E & TC Engineering, Ajeenkya D Y Patil School of Engineering, Pune 411047, India

* Correspondence: ravindra.r.patil@uit.no

Abstract: Computer vision in consideration of automated and robotic systems has come up as a steady and robust platform in sewer maintenance and cleaning tasks. The AI revolution has enhanced the ability of computer vision and is being used to detect problems with underground sewer pipes, such as blockages and damages. A large amount of appropriate, validated, and labeled imagery data is always a key requirement for learning AI-based detection models to generate the desired outcomes. In this paper, a new imagery dataset S-BIRD (Sewer-Blockages Imagery Recognition Dataset) is presented to draw attention to the predominant sewers' blockages issue caused by grease, plastic and tree roots. The need for the S-BIRD dataset and various parameters such as its strength, performance, consistency and feasibility have been considered and analyzed for real-time detection tasks. The YOLOX object detection model has been trained to prove the consistency and viability of the S-BIRD dataset. It also specified how the presented dataset will be used in an embedded vision-based robotic system to detect and remove sewer blockages in real-time. The outcomes of an individual survey conducted at a typical mid-size city in a developing country, Pune, India, give ground for the necessity of the presented work.

Keywords: sewer monitoring; S-BIRD dataset; object detection; computer vision; YOLOX training; AI techniques



Citation: Patil, R.R.; Mustafa, M.Y.; Calay, R.K.; Ansari, S.M. S-BIRD: A Novel Critical Multi-Class Imagery Dataset for Sewer Monitoring and Maintenance Systems. *Sensors* **2023**, *23*, 2966. <https://doi.org/10.3390/s23062966>

Academic Editors: Zhaoyang Wang, Minh P. Vo and Hieu Nguyen

Received: 30 January 2023

Revised: 25 February 2023

Accepted: 7 March 2023

Published: 9 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

An underground sewerage system is an essential feature of town planning as it transports the wastewater away from its source for safe disposal in the environment with minimum impact on the surroundings. However, underground pipe systems have maintenance problems. Sewer blockages and various damages such as cracks, fractures, joint displacement, etc. all can cause overflow, leaching of sewage into soil and interference with drinking water supply lines. Poor maintenance also leads sewer pipes to deteriorate early.

Therefore, it is important for any responsible authority to ensure that sewers are in good condition and run properly. The Ministry of Housing and Urban Affairs conferred Standard Operating Procedure (SOP) for cleaning sewers and septic tanks in November 2018 [1]. Regular inspections are necessary to identify any event of crack or blockage so that corrective measures are taken in time to avoid a crisis. In the past, manual inspection was often used followed by circuit television (CCTV) which has been one of the most used methods in the US and European municipalities in recent decades. However, these methods are labor-intensive and error-prone.

Artificial Intelligence (AI) is used in computer vision technology that consists of intelligent algorithms to interpret meaningful digital information from images and videos, which, when combined with automated robotic systems, provide powerful vision and intelligence to detect various sewer problems and to plan corrective actions. However, training AI-based *Deep Neural Object Detection Models* and achieving sewer inspection

objectives based on them requires large amounts of appropriate and labeled data. A dataset is a collection of featured and significant information in any field that is used to learn AI models for purposes such as detection, classification, regression, clustering, segmentation, etc. Data is usually in the form of images, text, numbers, time series, graphs, etc. The performance of the best detection model trained using a poor dataset is always inferior to the performance of a poor detection model trained using a highly featured and quality dataset. At the center of every object detector, whether single-stage or two-stage, is a classifier that secures the identities of all desired object classes. Clearly, the accuracy rate and performance of any detection model are highly dependent on the quality of the input imagery dataset.

Therefore, relevant dataset collection is a very important prerequisite for any AI model to predict outcomes with the desired accuracy and also has emerged as a prominent research theme in respective research communities. This involves data acquisition or collection, appropriately labeling the data and finally enhancement of obtainable data or models [2]. Due to the open-access research policy of many funding agencies, a large amount of data pertaining to many fields is available on various platforms. In many instances data may be available from data-sharing platforms like DataHub [3], Kaggle datasets [4], Mendeley Data [5], etc. and data searching platforms like Google Dataset Search [6], IEEE DataPort [7], etc. After tackling several challenges in data search, a researcher can succeed in obtaining the required dataset [8]. However, the European Commission recognized the difficulties in obtaining and tracing open data in 2011 and started to regulate data publishing activities in Europe [9]. Six snags in obtaining and tracing open data were identified: deficient details about the existence and accessibility of data, ambiguity about data ownership by public authorities, ambiguity about reuse terms, critical nature and cost of data, complex licensing processes and restrictive fees, specific reuse agreements with commercial members and reuse restrictions for state-owned companies.

Specifically, data acquisition includes tasks such as searching, augmenting and generating as needed, and in our case, the dataset is not only created due to unavailability but also prepossessed, augmented and labeled individually for classification and detection tasks. Manual or automated techniques are used for dataset generation, while synthetic data is generated to fill the lacking portion of the dataset. A standardized or benchmark dataset is always a central aspect to obtain the best-fit learning models and the application of transfer learning techniques with the developed dataset plays an important role in the advancement of AI-based models [10]. In computer vision, a dataset of digital images containing object class information is grouped as needed into a training set, validation set, and test set to serve as input to a detection model for learning, evaluation, and testing purposes, respectively. A workflow with decision-making for the S-BIRD dataset presented in this paper is shown in Figure 1, which displays the process from generation requirements to the training results.

In this paper, a new critical multi-class imagery dataset S-BIRD (Sewer-Blockages Imagery Recognition Dataset) is presented to identify sewer blockages caused by grease, plastics and tree roots. The lack of a standardized matrix for algorithms applied in the real-world development of sewer monitoring and maintenance systems is a critical issue, and the submitted dataset addresses this. So, the S-BIRD sets the standard for detection outcomes in real-time scenarios. Validation results of the S-BIRD dataset are given and development on an embedded vision platform to overcome actual sewer blockages problem is considered. In the conferred work, all computer vision and model training operations are implemented using Python programming, OpenCV, PyTorch framework, and some other machine learning libraries on the DGX workstation system including the Linux platform. Both the presented dataset and the corresponding results highlight the importance and necessity of such research work for the treatment of wastewater sewer blockages.

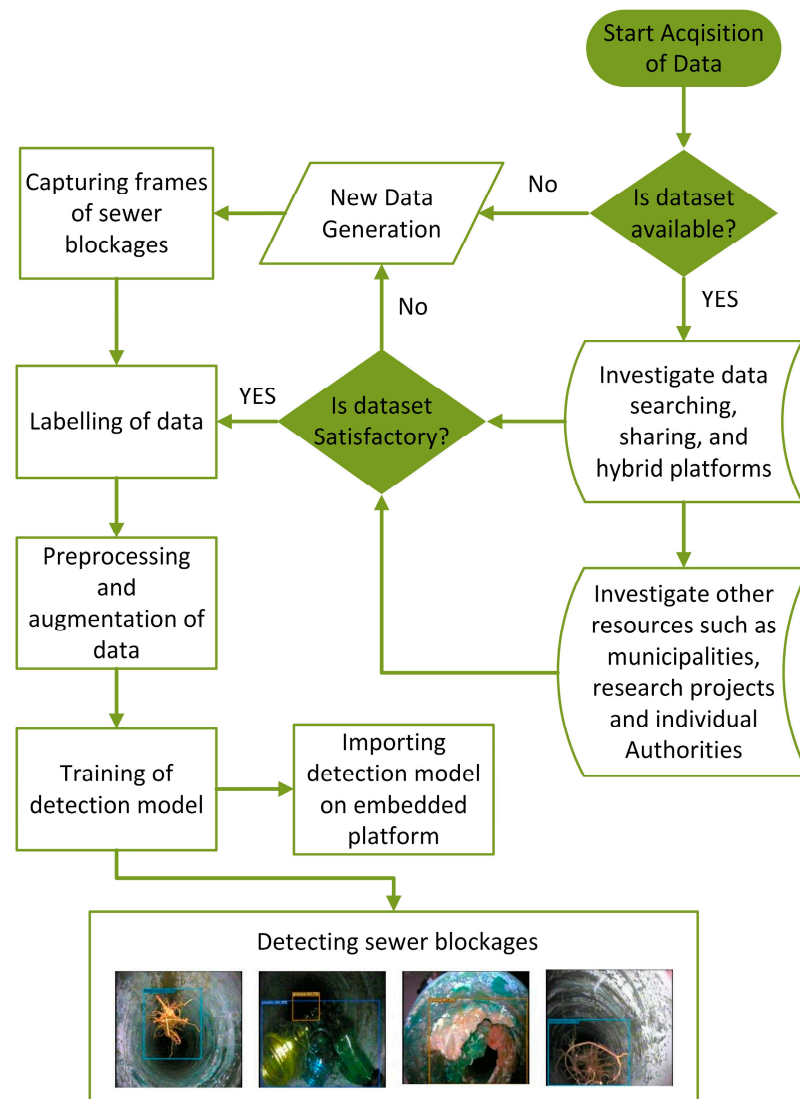


Figure 1. Workflow diagram of the presented S-BIRD dataset.

2. Needs of the S-BIRD Dataset

In earlier work, a survey on sewer robotic systems and computer vision practices in sewer inspection works was carried out and that gave information about practical issues concerning sewerage systems under the Pune Municipal Corporation (PMC), India [11]. It was concluded that sewer blockage is the main issue of sewers in Pune and to date, there is no robust algorithm and robotic system available for both real-time detection and removal of sewer pipe blockages.

Unlike many Western countries, India has single sewer lines for both sewage and stormwater. Thus, this combined drainage system is a big problem, particularly for cleaning and removing blockages.

In order to develop the function of detecting and identifying sewer blockages in real time, authenticated datasets are a prerequisite. Thus, all available means were used to search for datasets. Several municipalities and various authorities were also contacted for relevant data information, but no concrete work and datasets that may be used for real-time detection of sewer blockages were available. Furthermore, it was not possible to acquire a specific dataset for Indian conditions focusing on the issue of sewer blockages. The harmful, unhygienic and foul smell of a sewer environment is always a major concern when capturing frames of sewer problems for dataset generation. It is appropriate to imply

that independent binding, copyright or confidentiality issues relating to earlier works are also responsible for the unavailability of the datasets.

Sewer blockages are mainly caused by grease, plastic and tree roots. Other elements inside the sewer mix up with the black water and become difficult to identify. So, other elements are usually treated as a blackish sewer blockage, which is identified as black grease in the dataset. We also considered imagery data of grease, plastic and tree roots as mentioned above in the dataset S-BIRD, which is used for training of object detection model to locate and recognize the sewer blockages in real-time.

Obviously, blind systems cannot be as efficient as vision-based sewer robotic systems. Figure 2 shows the concept of constructing the S-BIRD dataset that takes grease, plastic and tree roots into account.



Figure 2. S-BIRD dataset for main sewer blockages.

3. Tools in S-BIRD Dataset Creation

In this section, the tools involved in creating the S-BIRD dataset are provided for detailed viewing.

3.1. Sewer Pipeline

In an unhygienic, muddy and smelly sewer pipe environment due to sewage, toiletry, sanitation, and stormwater from combined drainage systems, capturing real-time frames of sewer issues was a very difficult task for an individual. For simulating a sewer network, PVC pipelines of 200 mm diameter, which are widely used in residential sewers, were used to construct a typical sewer pipeline. The constructed sewer pipeline is shown in Figure 3.



Figure 3. Constructed sewer pipeline.

In this case, there is no big difference between a real sewer environment and a laboratory setup or simulated sewer network. Exactly the same blockage types with inherent nature have been created inside the sewer network consisting of all featured information. The only difference was that the simulated sewer network did not have the stench and noxious atmosphere. The detection model trained using the developed S-BIRD dataset in the respective sewer network is capable to work in practical situations.

3.2. Sewer Inspection Camera

Real-time frames of sewer barriers that include grease, plastics, and tree roots are captured by the watertight sewer camera shown in Figure 4, and its characteristics are given in Table 1.



Figure 4. Watertight sewer camera.

Table 1. Specifications of a utilized sewer camera.

Facets	Details
camera dimension	23 mm × 120 mm
camera light	12 modifiable white LEDs
watertight grade	IP68
vision angle	140 degree

This camera sensor is capable of capturing real-time frames at different angles not only for the intended aspect ratio but also for varying brightness due to attached modifiable white LEDs.

4. A Novel S-BIRD and Corresponding Results

This section discusses compiled imagery data (Section 4.1), its arithmetic details (Section 4.2), preprocessing and augmentation techniques applied to captured frames (Section 4.3), and annotated heatmap and object count histograms (Section 4.4).

4.1. Imagery Data Collection

All images of sewer blockages are captured under different lighting conditions and from different angles to gather the necessary perceptions and features. Figure 5 reveals some blockage frames of tree roots in the newly created dataset.

Dissimilar colored plastic is captured in the picture and key information for the detection and recognition task is achieved as shown in Figure 6.

There could be other elements within the black sewage mass such as plastic bags or other debris, but they look completely blackish as they are often mixed with black water and grease.

Figure 7 exhibits grease blockage frames capturing diverse and significant colored information. There are a number of sources for grease-type sewer blockages which mainly include wastage from domestic and high- or low-density production plants that produce huge chemical and processed waste.

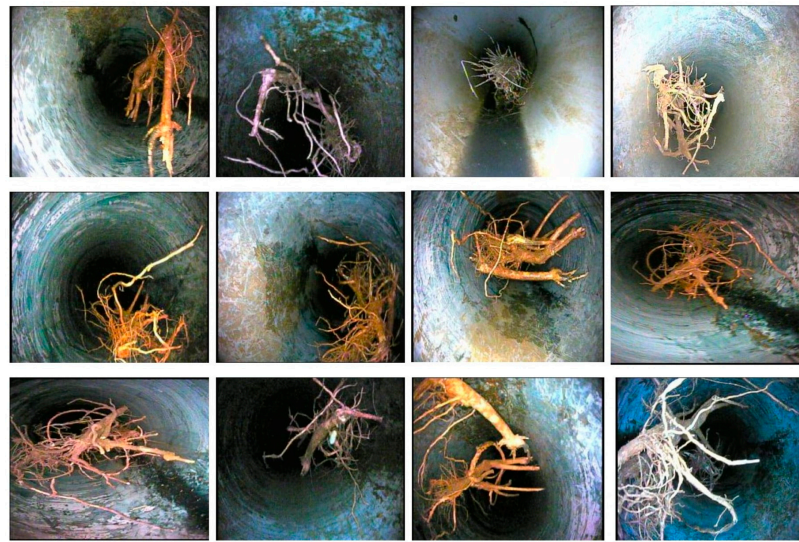


Figure 5. Tree root blockage frames in the S-BIRD dataset.



Figure 6. Plastic blockage frames in the S-BIRD dataset.

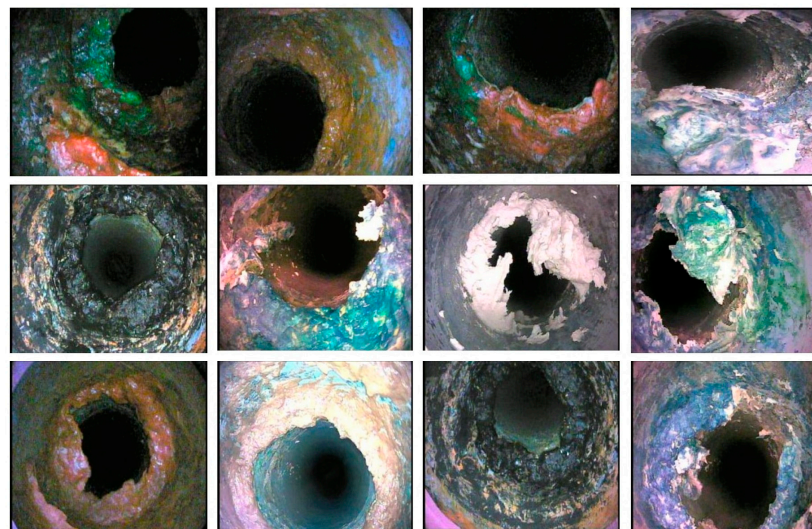


Figure 7. Grease blockage frames in the S-BIRD dataset.

4.2. Arithmetic Details of Captured Frames

The arithmetic details of the captured frames are listed in Table 2 for further implementation. Certainly, annotating the objects in each captured frame was time-consuming but the task was still performed individually with high skill and accuracy without labeling errors. The annotations contain information about the location, i.e., center x, center y, width, height and class of objects present in each frame of the S-BIRD dataset.

Table 2. Arithmetical details of captured frames.

Captured frames	Object Class (Sewer Blockage Type)	Captured Frames
	Tree roots	2295
	Plastic	2392
	Grease	2353
Total frames	7040	
Annotations	10,233 (Average = 1.5 per frame)	
Average frame size	0.08 Megapixels	
Mean frame ratio	352 × 240 (wide)	
Angle of diagonal	0.598 radian = 34.3°	
Length of diagonal	426 pixels	
Aspect ratio Class	1.467:1	
Pixel density	9 pixels/mm or 230 pixels/inch	

Figure 8 stipulates the total number of annotations for class balance, i.e., annotations for each sewer block type and these are 4131 for grease, 3471 for tree roots and 2631 for plastic.

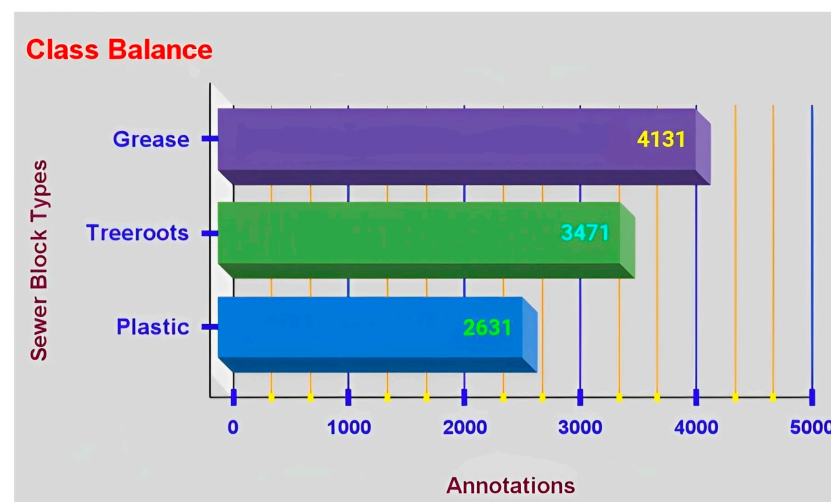


Figure 8. Annotation figures for class (sewer blockage type) balance.

The location of annotations, i.e., bounding boxes for considered blockage types in all captured frames is shown by heatmap in Figure 9. A heatmap represents informative data in a graphical or two-dimensional form where a color-coding system is used to represent values, and in the above heatmap, values are annotation details. It confers a quick visible summary to perceive the intricate nature of the dataset. Here, the correlation between annotated values is made easier to understand using colors in a heatmap compared to numerical tables. The yellow color denotes a highly positioned region of annotations whereas the light green color indicates lower positioning. All depicted heatmaps show that the locations of annotations are mostly in the center of the frames of object classes.

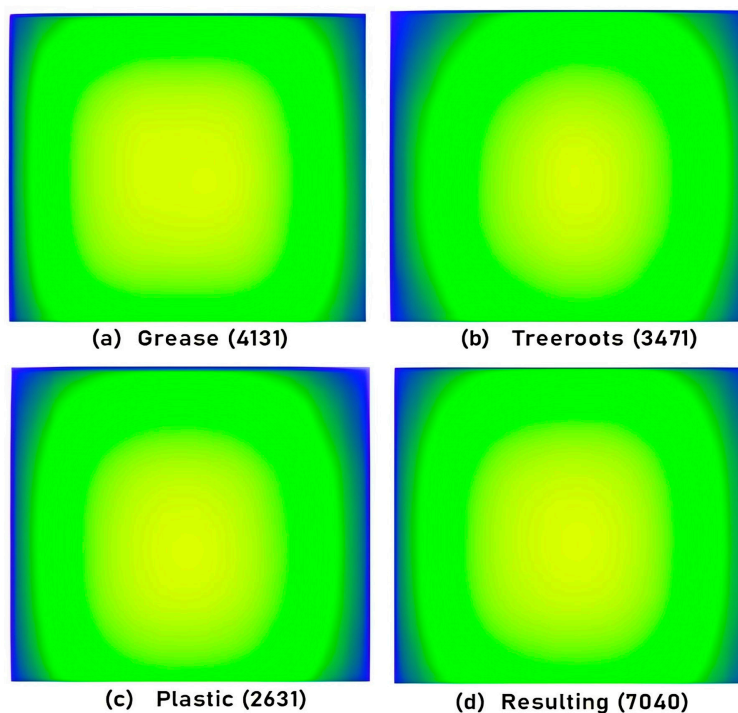


Figure 9. Annotation heatmap details for captured frames.

The imagery data is balanced into three groups such as training data with 4928 frames (70%), validation data with 1408 frames (20%) and testing data with 704 frames (10%) as shown in Figure 10.

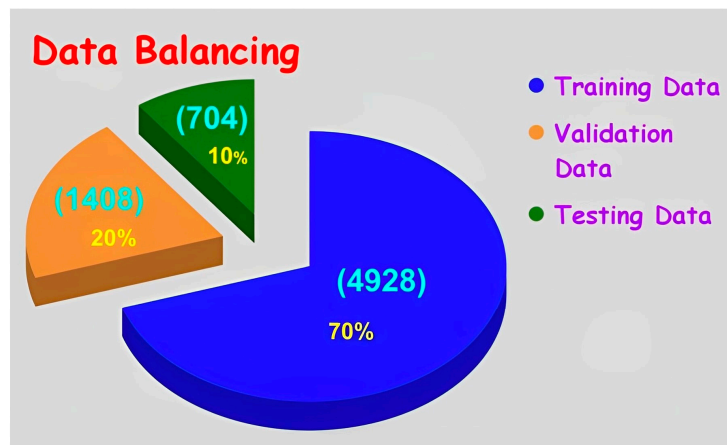


Figure 10. Imagery data balancing of particular sewer blockage type.

Table 3 provides annotation details for the classes in the training data.

Table 3. Annotations for training data.

Object Class (Sewer Blockage Type)	Annotations
Grease	2920
Tree roots	2455
Plastic	1821
Total	7196 (Average = 1.5 per frame)

4.3. Preprocessing and Augmentation Techniques

Here, two preprocessing techniques have been implemented on captured frames such as auto-orientation of pixel data, i.e., discarding the EXIF rotation and validating the pixel sort as well as resizing to 416×416 (px) by stretching the frame without losing source frame information. An image preprocessing benefits to reduce model training time and speed up inference of detection models.

Here, two preprocessing techniques have been implemented on captured frames such as auto-orientation of pixel data, i.e., discarding the EXIF rotation and validating the pixel sort as well as resizing to 416×416 (px) by stretching the frame without losing source frame information. Image preprocessing benefits from reduced model training time and sped-up inference of detection models.

Figure 11 shows the aspect ratio distribution graph for the S-BIRD dataset and makes clear that all frames are 416×416 (px), i.e., square in size.

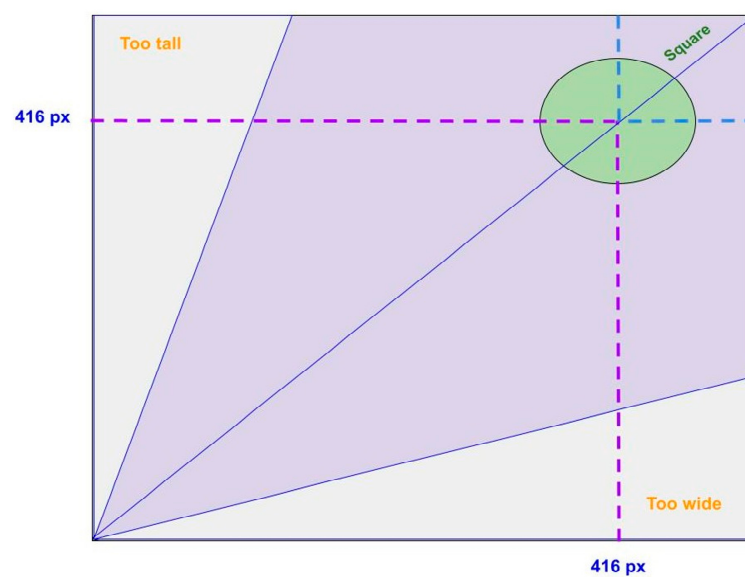


Figure 11. Aspect ratio distribution graph.

Further, image-level augmentation techniques have also been implemented to generate new training instances from existing training data.

Figure 12a shows the output frame of the gray scaling applied 25 percent to the input training frame which helps to increase the training variation but does not remove the color information when making inferences. Salt and pepper noise, also known as impulse noise, is applied to 5 percent of the pixels of the input frames as shown in Figure 12b which helps the detection model to turn out to be more flexible for camera artifacts through training. This noise involves adding some bright pixels to dark regions and some dark pixels to bright regions of the frames. It also helps to prevent adverse effects and avoid overfitting.

To strengthen the detection model against light and camera setting changes, random exposure adaptations were instigated between -25 and $+25$ percent for the input frame as shown in Figure 12c.

Two advanced augmentation techniques, namely cutout and mosaic, were exploited as shown in Figures 13a and 13b, respectively. Adding cutouts to training frames is extremely useful for the detection model to be strong against the object occlusion state. For this, three cutouts were inserted in 10 percent of each of the total sizes of the input frames. Next, the mosaic technique helps the detection model to work well on small objects by joining several images from the training set in collage [12]. In this, four different sewer block frames were added in a single frame.

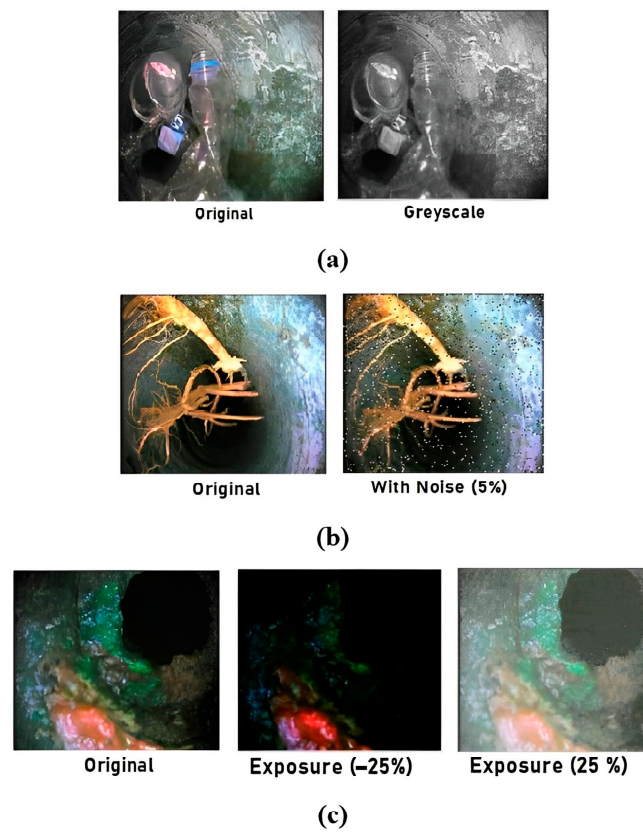


Figure 12. Visual upshots of standard augmentation techniques: (a) greyscaling, (b) salt and pepper noise, (c) random exposure.

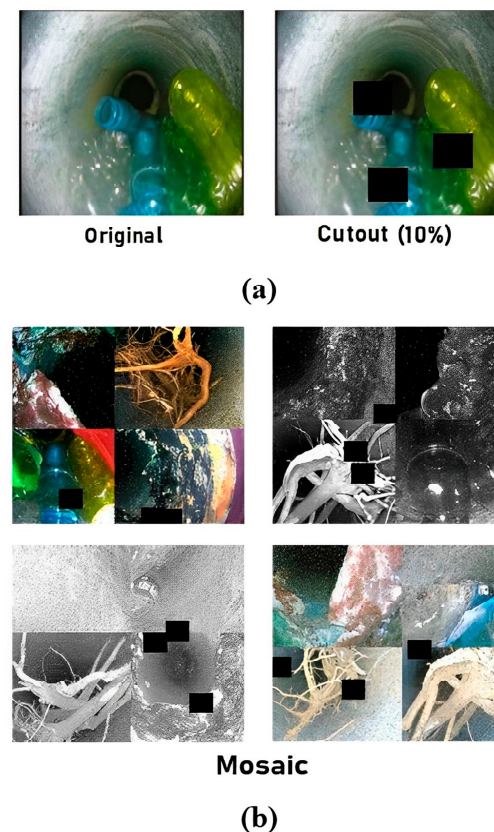


Figure 13. Visual upshots of advanced augmentation techniques: (a) cutout and (b) mosaic.

Augmentation techniques facilitate enhancing the efficiency of the object detection model by increasing the number and variegation of learning instances and related annotations. These techniques also reduce training time and costs for search detection models. So, discrete output versions have been generated for source frames.

In Table 4, the numerical details of training frames in S-BIRD are demonstrated after applying preprocessing and augmentation techniques.

Table 4. Arithmetical details of training frames in S-BIRD after preprocessing and augmentation.

Terms	Details
Total frames	14,765
Annotations	69,061 (Average = 4.7 per frame)
Average frame size	0.173 Megapixels
Mean frame ratio	416 × 416 (square)
Aspect ratio Class	1:1
Angle of diagonal	0.785 radian = 45°
Length of diagonal	588 pixels
Pixel density	12 pixels/mm or 290 pixels/inch

The graph in Figure 14 shows the escalated annotations for each sewer block type in S-BIRD's training data, after using annotation techniques. Now there are 26,847 annotations for grease, 21,553 for tree roots and 20,661 for plastics making a total of 69,061 augmented annotations, i.e., bounding boxes. Total annotations have increased by 61,865, i.e., 859.714%. Both preprocessing and augmentation techniques have been implemented using OpenCV, a computer vision and machine learning library, along with Python programming on the Linux platform from scratch to achieve the desired results.

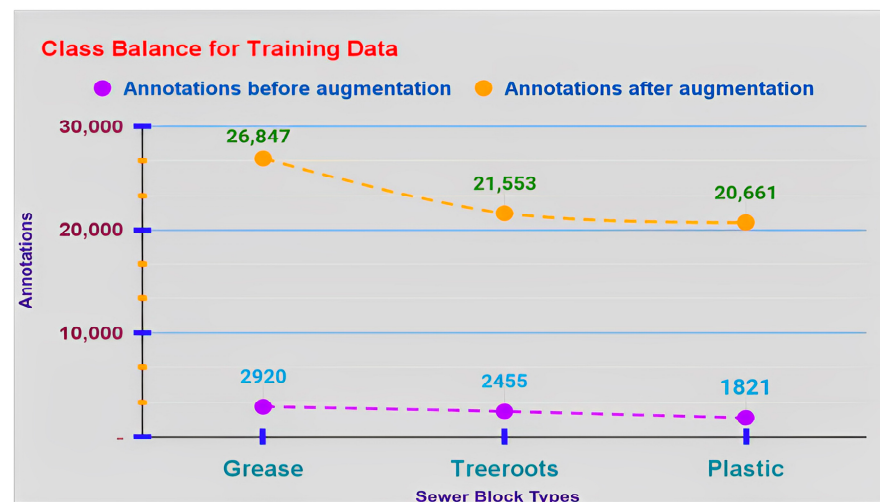


Figure 14. Annotation details for every single class in training data after image level augmentation.

4.4. Annotated Heatmap and Object Count Histogram

Two important parameters, namely the annotated heatmap and the object count histogram have been examined to assess the efficacy of the training data. The location of the entire annotations for grease, plastic and tree roots in S-BIRD's training data is illustrated by heatmaps in Figure 15. The specified heatmap informs us of the utmost generic position and weightage of all the annotations for revealed classes. From the color information of the heatmaps, it can be seen that most of the annotation locations are at the far left and right of both the top and bottom sides of the frames of object classes.

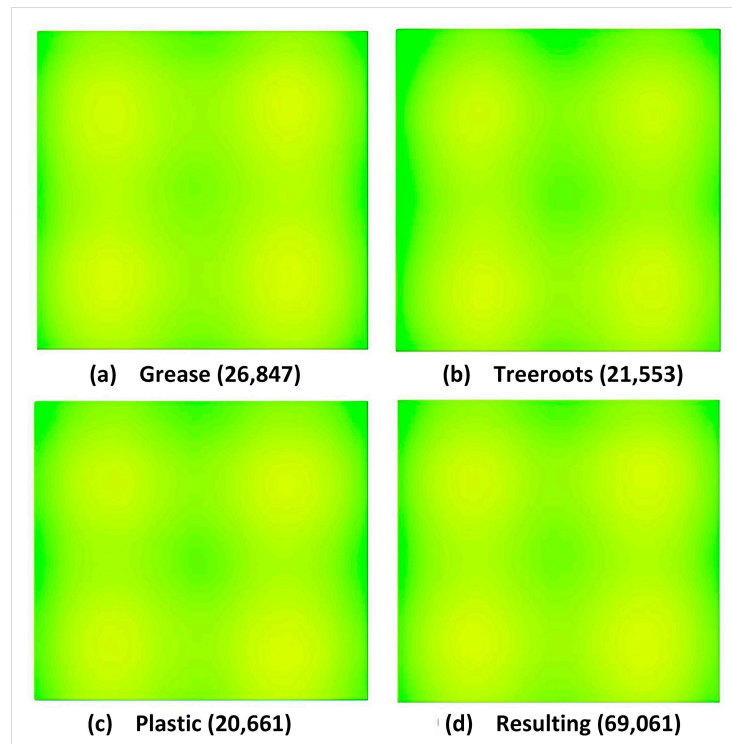


Figure 15. Annotation heatmap details for all classes.

A histogram is a chart that plots numeric data into bins represented by individual columns. Figure 16 details the number of frames on the y-axis and bins, i.e., the number of corresponding objects for all classes on x-axis, with the help of the object count histogram.

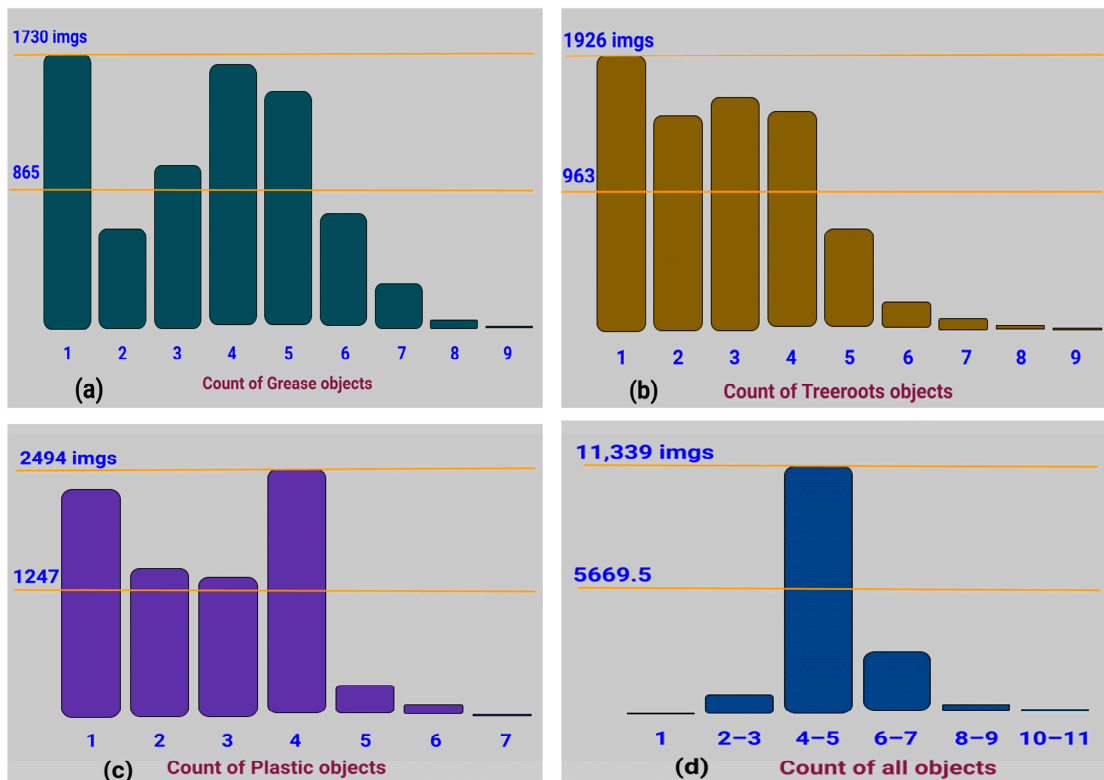


Figure 16. Object count histogram for: (a) grease, (b) tree roots, (c) plastic, and (d) all classes.

The number of objects, i.e., annotations for both grease and tree roots blocks are up to nine shreds as shown in Figure 16a,b. There is obviously one grease object for 1730 frames and four to five grease objects for 1400 to 1600 frames as given in Figure 16a. In total, 1926 frames contain a single tree root object and about 1500 frames contained three to four tree root objects as shown in Figure 16b. The number of plastic objects varies up to seven shreds as shown in Figure 16c in which four plastic objects are in 2494 frames and perceptibly one plastic object in about 2200 frames.

Figure 16d represents the object count histogram of all classes where 11,339 frames contain four to five objects. It also shows details for a much lower aggregate overall for a single object in frames as compared to the ratio for 69,061 annotations. The findings obtained for both parameters such as the annotated heatmap and the object count histogram prove the high veracity and standard for each imagery data class in S-BIRD.

5. Training of Object Detection Model

5.1. Insight on Conformation of Object Detector Models

Ordinarily, object detectors have two important segments, the backbone with pretraining to extract the features of input frames and the head which utilizes feature maps to predict classes and bounding boxes. Some layers are placed between the backbone and the head of recent object detectors to collect feature maps from distinct phases known as the neck. Object detectors with a backbone and densely predicted head are known as single-stage detectors, such as YOLO and SSD, while two-stage detectors have a backbone and head with dense and sparse predictions such as R-FCN, Faster R-CNN as shown in Figure 17. However, since single-stage detectors are faster than two stage detectors, they are used for multifarious real-time embedded applications. These object detectors embedded in robotic artifices are utilized to detect various faults in the sewerage system [13,14].

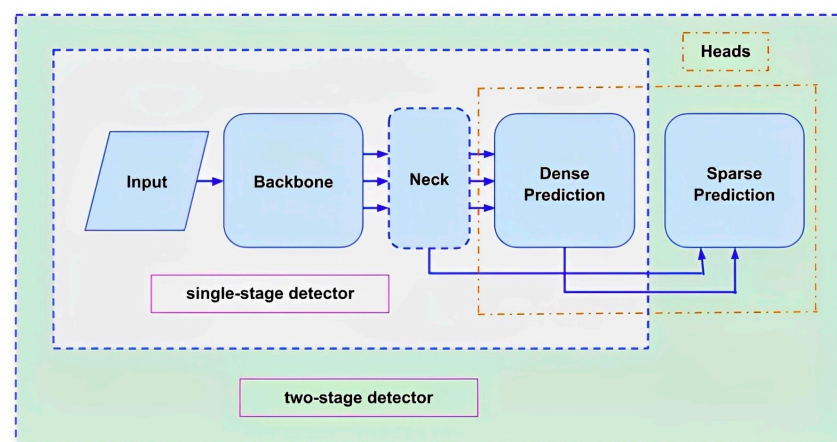


Figure 17. Conformation of object detector models.

Table 5 lists some instances of the conformation parts in the object detector models.

Table 5. Instances of conformation parts in the object detector models.

Conformation Parts		Details
Input		frames, multi-scaled frames, frame patches
Backbones		CSPDarknet-53 [15], Darknet53 [16], ResNet-50, ResNet-152, ResNet-10, GoogLeNet, Inception-ResNet-V2, EfficientNet-B0/B7, DetNet-59, ThunderNet, CBNet, VGG16, ViT, etc.
Neck		Bi-FPN, FPN, SFAM, PAN, etc.
Heads	Dense	YOLO [17], SqueezeDet, DetectNet, SSD, RetinaNet, MatrixNet, CenterNet, etc.
	Sparse	Mask R-CNN, R-FCN, Faster R-CNN [18], Cascade R-CNN, etc.

The popular one-stage YOLO detection model is constantly being improved for better performance. An advanced version of the YOLO detection model is the recently introduced YOLOX which comprises three different basic embarkations, such as (a) anchor-free design which uses a center-based approach with each pixel detection mechanism for the selection of just one positive instance which then estimate four distances such as left, top, right, and bottom from positives to the border, i.e., prediction consists of a single 4D vector to encode the location of the bounding box at every foreground pixel, (b) decoupled head for classification and regression, and (c) advanced label allocation tactics namely SimOTA which lessen the training time and evade other clarifier hyperparameters in the SinkhornKnopp algorithm, making it faster and more efficient than its equivalents [19]. The performance of YOLOX has been improved with addition of mosaic and mixup augmentation. YOLOv3 and Spatial Pyramid Pooling (SPP) layers with Darknet53 are employed as baseline by YOLOX. This detection model of different sizes has attained consistent improvements against all compatible counterparts when tested on modified CSPNet backbone in addition to the Darknet53 backbone.

5.2. Training of YOLOX Using S-BIRD

So, the small YOLOX detection model in PyTorch framework allowing mobile deployment has been trained to detect the main types of sewer blockages such as grease, plastic and tree roots using the newly developed S-BIRD. Annotations for sewer block types in S-BIRD were implemented in Pascal VOC format as per the requirement to advance the training process. The Tesla V100-DGXS-32GB GPU workstation was used as a training platform via Docker Container with a defined image.

Table 6 makes available particulars on crucial traits in the YOLOX-s training process.

Table 6. Crucial traits in training.

Traits	Values
learning model	YOLOX-s
Annotation data type	VOC
max_epoch	300
batch_size	16
fp16	True
num_classes	3
Params	8.94 M
Gflops	26.64
depth	0.33
width	0.5
input_size	(640, 640)
random_size	(14, 26)
nmsthre	0.65
degrees	10.0
translate	0.1
scale	(0.1, 2)
mscale	(0.8, 1.6)
shear	2.0
warmup_epochs	5
weight_decay	0.0005
momentum	0.9

The results obtained for the timing and precision of the YOLOX-s trained model for S-BIRD are given in Tables 7 and 8, respectively.

Table 7. Time results of the trained model.

Timing Parameters	Outturns (Milliseconds)
Average forward time	3.19 ms
Average NMS time	0.88 ms
Average inference time	4.07 ms

Table 8. Precision results of the trained model.

Class (Sewer Block Type)	Average Precision	Map_5095	Map_50
grease	0.9004		
tree roots	0.8930	0.7885	0.9005
plastic	0.9081		

From Table 7 and Figure 18, YOLOX-s has achieved 90.04% AP for grease blocks, 90.81% AP for plastic blocks, 89.30% AP for tree root blocks, and 90.05% mean-AP computed at IoU (Intersection over Union) threshold 0.5. Another m-AP calculated over different IoU thresholds, from 0.5 to 0.95 with a step of 0.05 is 78.85%. The best-fit model is selected using cross-validation or rotation estimation technique [20]. The visual upshots of precisely detected sewer blocks such as tree roots, plastic and grease, are delineated in Figure 19. Of course, multiple sewer blockages in the same frame have also been considered for real-time detection purposes. Overall, the obtained results of the YOLOX-trained model prove the consistency and viability of the new S-BIRD dataset presented.

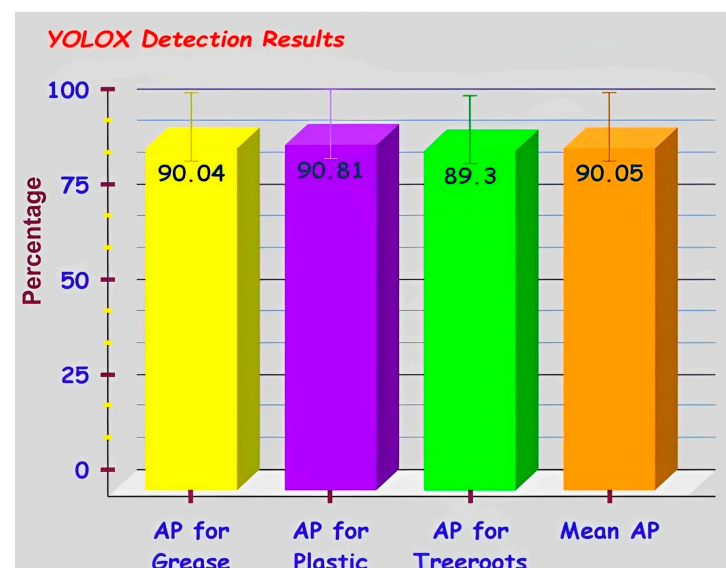


Figure 18. YOLOX detection results for all classes in S-BIRD.

5.3. Embedded Vision with S-BIRD

The embedded vision is a pioneering and comprehensive platform for real-world visual implementations in the areas of home life equipment, health, daily services, security through detection and tracking, etc. [21,22]. So, the object detection model trained using S-BIRD will be a significant addition to existing or newly developed embedded vision-based sewer robotic systems.

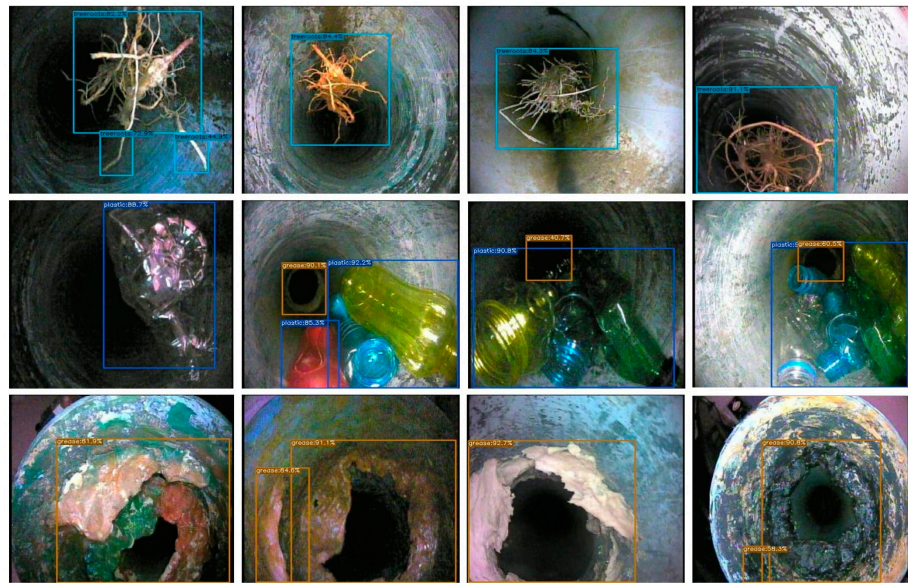


Figure 19. Visual upshots of detected tree roots, plastic and grease types of sewer blocks.

PIRAT [23], KARO [24], KURT [25], MAKRO [26], KANTARO [27], SIAR [28], etc. are some of the popular developments in the field of sewer robotics that serve the purpose of sewer inspection. Figure 20 shows the block diagram of an automated system that has a power-driven cutting tool to remove sewer blocks located by a detector trained using S-BIRD.

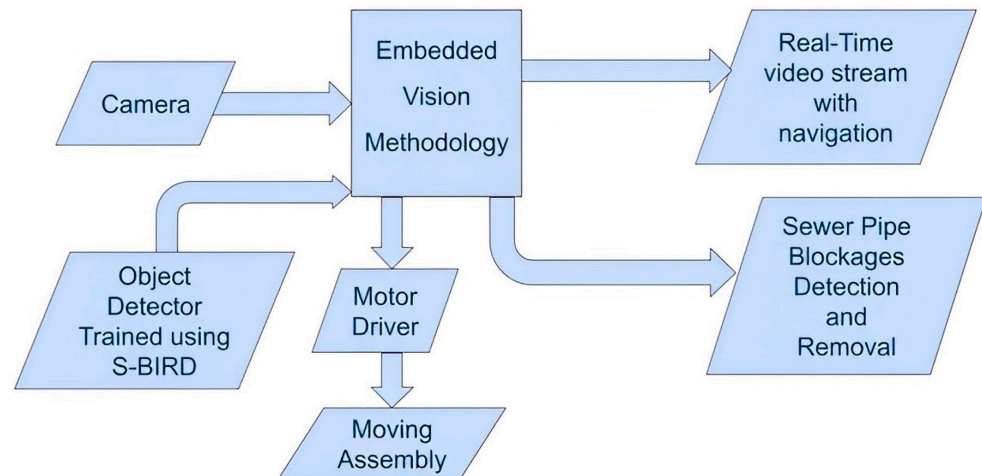


Figure 20. Block diagram of an automated system.

Here, Jetson nano has been selected as the embedded platform having a 4 GB GPU card of 128-Cuda cores and is suitable for running deep neural-network-based object detector models and for processing contiguous frames in real-time. Cameras such as a webcam, arducam, or raspicam are used to capture the surrounding frames for the purpose of navigation and processing, and then the output frames of detected sewer blockages are displayed on the screen to a remote location as shown in Figure 21.

In order to solve the recurring problem of underground sewer barriers in the practical world, a smart and comprehensive vision-based automation system with an AI detector trained using S-BIRD is certainly capable of meeting the needs of responsible authorities of any country.

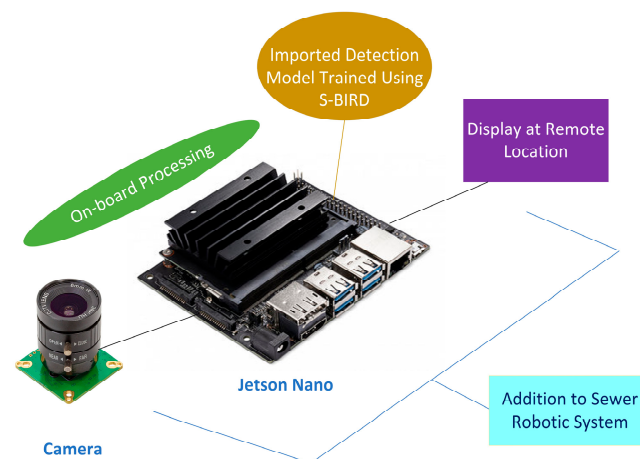


Figure 21. Embedded vision platform for sewer robotic system.

6. Conclusions

In conclusion, a new critical multi-class imagery dataset S-BIRD which includes frames of main sewer blocks such as grease, plastic and tree roots is introduced to fulfill the need for implementing computer vision to automated robotic systems for identifying blockages in the sewerage pipes.

Arithmetic details for both compiled, as well as preprocessed and augmented data are discussed. The obtained results for preprocessing and augmentation demonstrate the increased number and variegation of learning instances and related annotations for the efficient performance of the object detection model. The procured details of heatmaps and object count histograms prove the high strength, veracity and standard for each imagery data class in S-BIRD.

The trained small YOLOX model achieved 90.04% AP for grease blocks, 90.81% AP for plastic blocks, 89.30% AP for tree root blocks, 90.05% Mean-AP at 0.5 IoU threshold, and 78.85% Mean-AP at 0.5 to 0.95 IoU thresholds for 300 epochs using S-BIRD. The relevant outcomes prove the consistency and viability of the new S-BIRD dataset presented. The object detectors trained using the presented S-BIRD will be a valuable addition to the existing or newly developed embedded vision-based sewer monitoring and maintenance systems for detecting sewer blockages in real-time scenarios.

Author Contributions: Conceptualization, R.R.P. and M.Y.M.; methodology, R.R.P.; software, R.R.P.; Dataset Creation, R.R.P.; validation, R.R.P. and M.Y.M.; formal analysis, R.R.P. and M.Y.M.; investigation, R.R.P., M.Y.M., R.K.C. and S.M.A.; writing—original draft preparation, R.R.P.; writing—review and editing, M.Y.M., R.K.C. and R.R.P.; visualization, M.Y.M. and R.R.P.; project administration, R.K.C., M.Y.M. and S.M.A.; funding acquisition, R.K.C. and M.Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: The publication charges for this article have been funded by a grant from the publication fund of UiT the Arctic University of Norway.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The research data will be made available on the request.

Acknowledgments: Authors acknowledge the support from SPRING Eu-India Project and UiT the Arctic University of Norway, Narvik, Norway for PhD studies of Ravindra R. Patil (No. 821423 and GOI No. BT/IN/EU-WR/60/SP/2018). Thanks are due to the Department of Computer Engineering and IT, COEP Technological University (COEP Tech) for providing the high computing GPU server facility procured under TEQIP-III (A world bank project) for our research work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Information Manual—Standard Operating Procedure (SOP) for Cleaning of Sewers and Septic Tanks by Central Public Health & Environmental Engineering Organization (CPHEEO), Ministry of Housing and Urban Affairs, Government of India. Available online: <http://cpheeo.gov.in/upload/5c0a062b23e94SOPforcleaningofSewersSepticTanks.pdf> (accessed on 28 January 2023).
2. Roh, Y.; Heo, G.; Whang, S.E. A survey on data collection for machine learning: A big data-ai integration perspective. *IEEE Trans. Knowl. Data Eng.* **2019**, *33*, 1328–1347. [[CrossRef](#)]
3. Bhardwaj, A.; Karger, D.; Subramanyam, H.; Deshpande, A.; Madden, S.; Wu, E.; Elmore, A.; Parameswaran, A.; Zhang, R. Collaborative data analytics with DataHub. *Proc. VLDB Endow.* **2015**, *8*, 1916. [[CrossRef](#)] [[PubMed](#)]
4. Kaggle. Available online: <https://www.kaggle.com/> (accessed on 22 January 2023).
5. Mendeley Data. Available online: <https://data.mendeley.com/> (accessed on 18 January 2023).
6. Google Dataset Search. Available online: <https://datasetsearch.research.google.com/> (accessed on 12 January 2023).
7. IEEE DataPort. Available online: <https://ieee-dataport.org/dataset> (accessed on 9 January 2023).
8. Chapman, A.; Simperl, E.; Koesten, L.; Konstantinidis, G.; Ibáñez, L.D.; Kacprzak, E.; Groth, P. Dataset search: A survey. *VLDB J.* **2020**, *29*, 251–272. [[CrossRef](#)]
9. European Commission. Digital Agenda: Commission’s Open Data Strategy, Questions and Answers. Memo/11/891. 12 December 2011. Available online: https://ec.europa.eu/commission/presscorner/detail/en/MEMO_11_891 (accessed on 4 January 2023).
10. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [[CrossRef](#)]
11. Patil, R.R.; Ansari, S.M.; Calay, R.K.; Mustafa, M.Y. Review of the State-of-the-art Sewer Monitoring and Maintenance Systems Pune Municipal Corporation—A Case Study. *TEM J.* **2021**, *10*, 1500–1508. [[CrossRef](#)]
12. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
13. Cheng, J.C.; Wang, M. Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Autom. Constr.* **2018**, *95*, 155–171. [[CrossRef](#)]
14. Kumar, S.S.; Wang, M.; Abraham, D.M.; Jahanshahi, M.R.; Iseley, T.; Cheng, J.C. Deep learning–Based automated detection of sewer defects in CCTV videos. *J. Comput. Civ. Eng.* **2020**, *34*, 04019047. [[CrossRef](#)]
15. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. 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 (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 1571–1580.
16. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
17. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
18. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems—Volume 1 (NIPS’15), Montreal, QC, Canada, 7–12 December 2015; MIT Press: Cambridge, MA, USA, 2015; pp. 91–99.
19. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
20. Berrar, D. Cross-Validation. In *Encyclopedia of Bioinformatics and Computational Biology*; Ranganathan, S., Gribskov, M., Nakai, K., Schönbach, C., Eds.; Academic Press: Oxford, UK, 2019; pp. 542–545. [[CrossRef](#)]
21. Vaidya, O.S.; Patil, R.; Phade, G.M.; Gandhe, S.T. Embedded Vision Based Cost Effective Tele-operating Smart Robot. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2019**, *8*, 1544–1550.
22. Patil, R.R.; Vaidya, O.S.; Phade, G.M.; Gandhe, S.T. Qualified Scrutiny for Real-Time Object Tracking Framework. *Int. J. Emerg. Technol.* **2020**, *11*, 313–319.
23. Kirkham, R.; Kearney, P.D.; Rogers, K.J.; Mashford, J. PIRAT—A system for quantitative sewer pipe assessment. *Int. J. Robot. Res.* **2000**, *19*, 1033–1053. [[CrossRef](#)]
24. Kuntze, H.B.; Schmidt, D.; Haffner, H.; Loh, M. KARO—A flexible robot for smart sensor-based sewer inspection. In Proceedings of the International Conference No Dig, Dresden, Germany, 22 September 1995; Volume 95, pp. 367–374.
25. Kirchner, F.; Hertzberg, J. A prototype study of an autonomous robot platform for sewerage system maintenance. *Auton. Robot.* **1997**, *4*, 319–331. [[CrossRef](#)]
26. Rome, E.; Hertzberg, J.; Kirchner, F.; Licht, U.; Christaller, T. Towards autonomous sewer robots: The MAKRO project. *Urban Water* **1999**, *1*, 57–70. [[CrossRef](#)]
27. Nassiraei, A.A.; Kawamura, Y.; Ahrary, A.; Mikuriya, Y.; Ishii, K. Concept and design of a fully autonomous sewer pipe inspection mobile robot “kantaro”. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 10–14 April 2007; pp. 136–143.
28. Alejo, D.; Mier, G.; Marques, C.; Caballero, F.; Merino, L.; Alvito, P. SIAR: A ground robot solution for semi-autonomous inspection of visitable sewers. In *Advances in Robotics Research: From Lab to Market*; Springer: Cham, Switzerland; Berlin/Heidelberg, Germany, 2020; pp. 275–296.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

**Research Article – AI-Driven High-Precision Model for Blockage
Detection in Urban Wastewater Systems**

Article

AI-Driven High-Precision Model for Blockage Detection in Urban Wastewater Systems

Ravindra R. Patil ^{1,*} , Rajnish Kaur Calay ¹, Mohamad Y. Mustafa ¹  and Saniya M. Ansari ²

¹ Faculty of Engineering Science and Technology, UiT the Arctic University of Norway, 8514 Narvik, Norway; rajnish.k.calay@uit.no (R.K.C.); mohamad.y.mustafa@uit.no (M.Y.M.)

² Department of E & TC Engineering, Ajeenkya D Y Patil School of Engineering, Pune 411047, India

* Correspondence: ravindra.r.patil@uit.no

Abstract: In artificial intelligence (AI), computer vision consists of intelligent models to interpret and recognize the visual world, similar to human vision. This technology relies on a synergy of extensive data and human expertise, meticulously structured to yield accurate results. Tackling the intricate task of locating and resolving blockages within sewer systems is a significant challenge due to their diverse nature and lack of robust technique. This research utilizes the previously introduced “S-BIRD” dataset, a collection of frames depicting sewer blockages, as the foundational training data for a deep neural network model. To enhance the model’s performance and attain optimal results, transfer learning and fine-tuning techniques are strategically implemented on the YOLOv5 architecture, using the corresponding dataset. The outcomes of the trained model exhibit a remarkable accuracy rate in sewer blockage detection, thereby boosting the reliability and efficacy of the associated robotic framework for proficient removal of various blockages. Particularly noteworthy is the achieved mean average precision (mAP) score of 96.30% at a confidence threshold of 0.5, maintaining a consistently high-performance level of 79.20% across Intersection over Union (IoU) thresholds ranging from 0.5 to 0.95. It is expected that this work contributes to advancing the applications of AI-driven solutions for modern urban sanitation systems.

Keywords: AI; object detection; S-BIRD dataset; computer vision; transfer learning; YOLOv5; wastewater management



Citation: Patil, R.R.; Calay, R.K.; Mustafa, M.Y.; Ansari, S.M.

AI-Driven High-Precision Model for Blockage Detection in Urban Wastewater Systems. *Electronics* **2023**, *12*, 3606. <https://doi.org/10.3390/electronics12173606>

Academic Editors: Dong Zhang and Dah-Jye Lee

Received: 10 August 2023

Revised: 22 August 2023

Accepted: 24 August 2023

Published: 26 August 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Computer vision is a field of artificial intelligence (AI) with its own conventional algorithms that extract required information from various visual forms such as photos and videos, and based on that information form, perform actions, or make recommendations in order to detect and identify distinct objects. Thus, the large datasets should increase the performance properties of computer vision.

Object detection techniques of computer vision detect the occurrence of objects in an image or video with bounding boxes and identify their classes. Initially, machine learning was mainly used for object detection tasks but when deep neural networks, i.e., deep learning methods emerged, they became popular due to automatic representative feature extraction from large datasets for training purposes [1]. Occlusion, clutter, and low resolution are some of the sub-problems that are handled very efficiently by deep learning-based detection frameworks [2,3]. It has two method types such as single-stage, which works for inference speed and real-time use, and two-stage, which works for model performance, i.e., detection accuracy. The single-stage detectors remove the process of region of interest (ROI) extraction and moves for classification and regression whereas two-stage detectors extract ROI and then apply classification and regression. The YOLO detection model (YOLOv2 [4], YOLOv3 [5], YOLOv4 [6], and YOLOv5 [7]), SSD [8], CenterNet [9], CornerNet [10], etc., are some single stage detectors. Region proposal models (R-CNN [11], Fast-RCNN [12], Faster

RCNN [13], Cascade R-CNN [14], and R-FCN [15]) are two-stage detectors. Classification and localization accuracy and inference speed are two important metrics for object detectors. In the advancement of detection models, transfer learning techniques with quality datasets meet the requirements with a minimum training time [16,17]. Transfer learning harnesses prior knowledge to enhance performance on novel tasks. By fine-tuning, pre-trained deep neural models are adapted to new contexts with certain layers preserved and others refined. This leads to many advantages such as achieving quick convergence, good performance, and adaptability in real-world scenarios. As the applications of AI evolve, such as video surveillance, military applications, security aspects, health monitoring, and critical detection tasks, the AI techniques are being enhanced to suit these needs.

Addressing the application-based needs to produce sensible and accurate results, detection models need to be adapted and modified, which usually have heavy computational demands. However, there are methods such as the embedded vision approach with AI that has an ability to enable real-time, efficient, and intelligent visual processing directly on edge devices, which reduces dependency on cloud computing and enhances privacy and responsiveness in many applications [18,19].

Detecting various sewer blockages is a major challenge due to their complex and heterogeneous nature. Moreover, their locations in the sewer network may vary, including main lines, lateral connections, and junctions. Blockages can exhibit varying levels of severity, from partial restrictions that gradually reduce flow to complete blockages that cause sewer overflows. The dynamic and unpredictable nature of urban wastewater systems, influenced by factors such as climate, wastewater composition, and hydraulic conditions adds another layer of complexity. In this research work, transfer learning and fine-tuning techniques are utilized to achieve a high precision rate in the detection of blockages within urban wastewater systems. This approach is intended for real-time implementation on mobile devices and other environments with limited resources, with the goal of effectively removing such blockages. Our primary emphasis is on the training of the single-stage YOLOv5 model using the S-BIRD dataset [20,21], which contains representative and critical multi-class images depicting prevalent sewer blockage scenarios.

The study implements all computer vision and model training procedures using Python programming, OpenCV, PyTorch framework, and other machine learning libraries. These operations are carried out on a DGX GPU workstation system running on the Linux platform, ensuring a robust and efficient experimental environment. The results are analyzed and discussed to demonstrate the effectiveness of the methodology used.

2. Structural Insights of YOLOv5 Model

YOLOv5 is an anchor-based single-stage detection model, which is built on the PyTorch framework. It focuses on simplicity, model scaling, and transfer learning, making it versatile for a wide range of object detection tasks. The model's backbone is CSP Darknet-53, which incorporates Cross Stage Partial (CSP) connections to enhance information flow and feature representation.

To create feature pyramids for effective object scaling and generalization, YOLOv5 employs the Path Aggregation Network (PAN) as its neck. The head design utilizes anchor boxes to generate output vectors that contain class probabilities, objectness scores, and bounding box coordinates (center_x, center_y, height, and width). The model parameters are updated during training using the following loss function:

$$\text{Loss} = \lambda_1 * L_{\text{cls}} + \lambda_2 * L_{\text{obj}} + \lambda_3 * L_{\text{loc}} \quad (1)$$

where L_{cls} represents the Binary Cross Entropy loss for predicted classes, L_{obj} represents the Binary Cross Entropy loss for objectness scores, and L_{loc} represents the Complete Intersection over Union loss for bounding box locations. Here, λ_1 , λ_2 , and λ_3 are hyperparameters controlling the contribution of each component to the overall loss. The employed auto anchor automatically determines and generates anchor boxes based on the distribution of bounding boxes in the custom dataset using K-means clustering and a genetic learning

algorithm. In this, SiLU (Sigmoid Linear Unit) activation function in hidden layers acquire intricate details and Sigmoid activation function in the output layer functions for binary classification.

As shown in Figure 1, the backbone employs Convolutional and C3 layers to extract image features, which are then combined at various levels using Conv, Upsample, Concat, and C3 layers in the head. The object detection process is facilitated by a Detect layer that uses anchor boxes and the indicated class count. Particularly, each C3 (CSP-3) block consists of two parallel convolutional layers, the first layer channels input features through a bottleneck layer, compressing the information and the second layer directly outputs feature. These streams are then concatenated and processed through pooling and convolutional layers. The C3 blocks also use skip connections and attention mechanisms to enhance information flow and reduce noisy features.

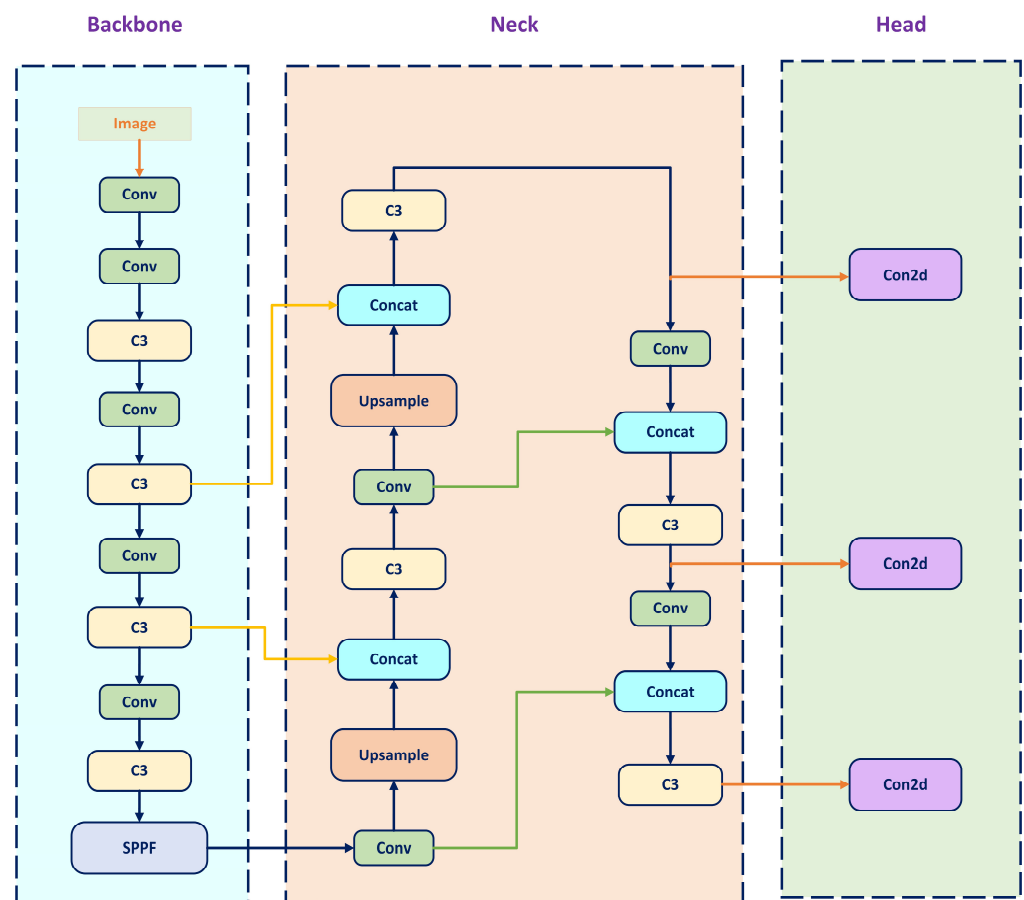


Figure 1. Architectural perception of YOLOv5 model.

3. Details of Training Instances in Critical Multi-Class S-BIRD

The dataset comprises a total of 14,765 training frames of classes (grease, plastics, and tree roots), which are meticulously annotated with 69,061 objects as shown in Figure 2, resulting in an average of 4.7 annotations per frame. Specifically, the dataset comprises 26,847 annotations for grease, 21,553 annotations for tree roots, and 20,661 annotations for plastics. To ensure uniformity and standardization, the frames were preprocessed and augmented, resulting in an average frame size of 0.173 Megapixels. The frames were resized to a square aspect ratio of 416×416 pixels, thereby maintaining a 1:1 aspect ratio class. The angle of the diagonal was calculated to be 0.785 radians (equivalent to 45 degrees), with the diagonal length measuring 588 pixels.

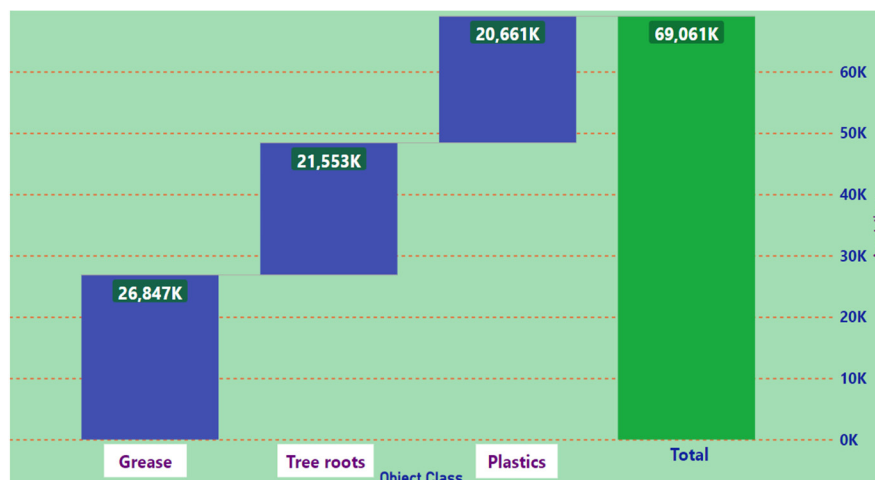


Figure 2. Labelling details of training instances from dataset.

Regarding pixel density, the dataset exhibits a density of 12 pixels per millimeter or 290 pixels per inch. These specific computational details are vital for understanding the characteristics and intricacies of the S-BIRD dataset, which plays a crucial role in effectively training the deep neural network. Figure 3 illustrates the distribution of object classes in each training frame based on the center x for the S-BIRD dataset. Figure 3 shows the relative distribution of center x coordinates across different classes during training. Each segment is color-coded and displays data values and percentiles, providing a clear understanding of object positions along the x-axis. This section provides valuable insights into the dataset’s dimensions, resolutions, and geometric properties, which contribute to the successful implementation of transfer learning and fine-tuning techniques for the deep neural detection model.

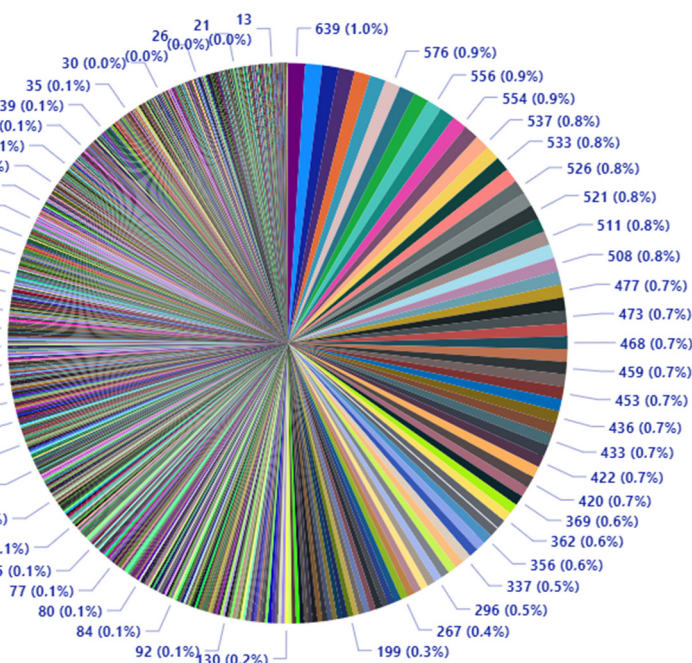


Figure 3. Object classes in each training frame by center x.

4. Training Method and Evaluation

The training process for the YOLOv5-s model (Based on PyTorch 1.10.0a0 with CUDA support) on the S-BIRD dataset involved a series of steps aimed at achieving the highest precision in detecting sewer blockages. Through the application of transfer learning

and fine-tuning techniques, the model's formulation was optimized to suit the specific characteristics of the representative dataset, enabling its effective adaptation for real-world scenarios. To facilitate the training process, annotations for object classes were applied in PyTorch TXT format, as needed. The training process was performed over 6000 epochs, using the stochastic gradient descent (SGD) optimizer with specified hyperparameters. The training process utilized the configurations listed in Table 1. The DGX-1 (utilized 32 GB GPU Card) available at UiT, Narvik, running a Docker container with a defined image served as the training platform, leveraging GPU parallelization for faster computations. Overfitting was mitigated using Early Stopping with a patience of 100 epochs.

Table 1. Principal training configurations.

Attributes	Implications
learning model	YOLOv5-s
Annotation data type	PyTorch TXT
max_epoch	6000
patience	100
batch_size	16
fp16	True
num_classes	3
Params	7.2 M
Gflops	15.9
depth	0.33
width	0.5
input_size	(416, 416)
workers	8
anchor_t	4.0
scale	0.5
hsv_h, hsv_s, hsv_v	0.015, 0.7, 0.4
warmup_epochs	3
weight_decay	0.0005
momentum	0.937
translate	0.1

The training progression concluded at 933 epochs due to a lack of improvement in the last 100 epochs. The most promising results were obtained at epoch 832, leading to the selection of the corresponding model for practical applications. The evaluation metrics are essential for quantifying the model's performance, and they are computed using the following formulas:

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (2)$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) \quad (3)$$

$$\text{mAP} = \sum (\text{AP for each class}) / \text{Number of classes} \quad (4)$$

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (5)$$

Here, TP—true positive, FP—false positive, FN—false negative, and mAP—mean average precision.

During the training, at epoch 832, the model exhibited impressive precision (P) and recall (R) values of 94.40% and 93.90%, respectively, across all classes. Notably, Figure 4 illustrates that the developed detection model achieved outstanding average precision values of 95.90% for grease blocks, 98.40% for plastic blocks, and 94.50% for tree root blocks. These high precision values are indicative of the model's ability to accurately detect and classify instances belonging to these specific classes. The overall mean average precision (mAP) for all classes, as indicated in Table 2, is remarkably high at 96.30% with a confidence

threshold of 0.5. This highlights the model’s proficiency in making precise detections across all classes within the dataset. Moreover, the calculated mAP over various Intersection over Union (IoU) thresholds, ranging from 0.5 to 0.95 with an increment of 0.05, yielded a consistent performance of 79.20%. This demonstrates that the model maintains accurate localization of objects across a broad range of IoU thresholds. The timing results in Table 3 show that the model has efficient inference times, with an average forward time of 0.2 ms, average NMS time of 1.1 ms, and average inference time of 11 ms. These low inference times make the model suitable for real-time applications.

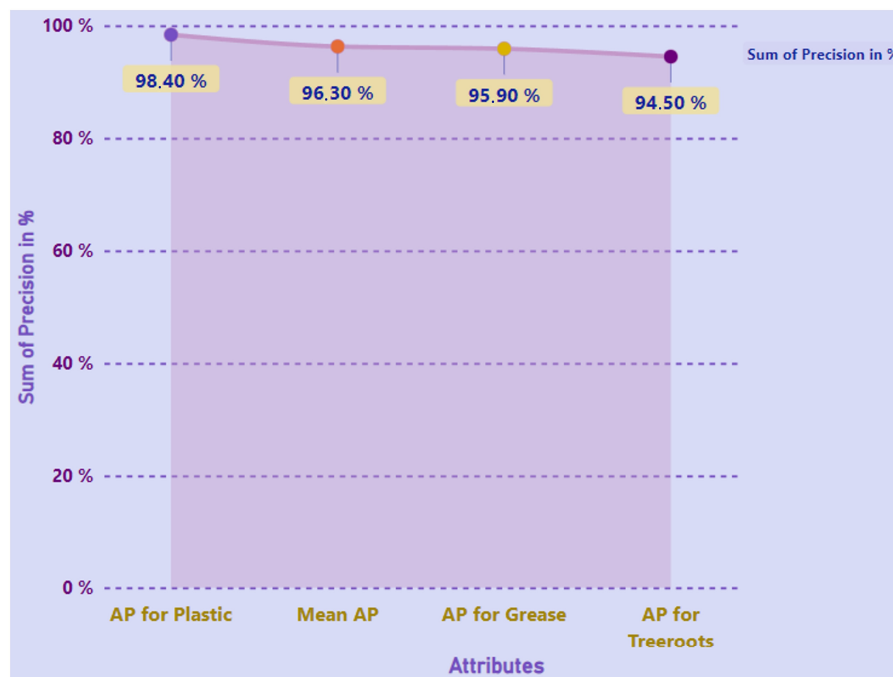


Figure 4. Obtained higher precision rate for each class.

Table 2. Temporal evaluation details.

Timing Attributes	Outturns (Milliseconds)
Average forward time	0.2 ms
Average NMS time	1.1 ms
Average inference time	11 ms

Table 3. Precision assessment details.

Object Class	Average Precision	map_5095	map_50
tree roots	0.945		
grease	0.959	0.792	0.9630
plastic	0.984		

The confusion matrix in Figure 5, provides an overview of the model’s performance in correctly classifying instances of grease, plastic, and tree roots. This visualization provides a clear breakdown of correct and incorrect classifications for each category.

Figure 6 shows correlation connections within the frames of the dataset, demonstrating the exact connection between instances and their labels among discrete views. It is also evident that a majority of instances in the dataset are situated towards the outer edges of both the top and bottom sides of the images in the dataset. This indicates the efficiency of the trained model to detect and classify multiple objects in various real-world scenarios.

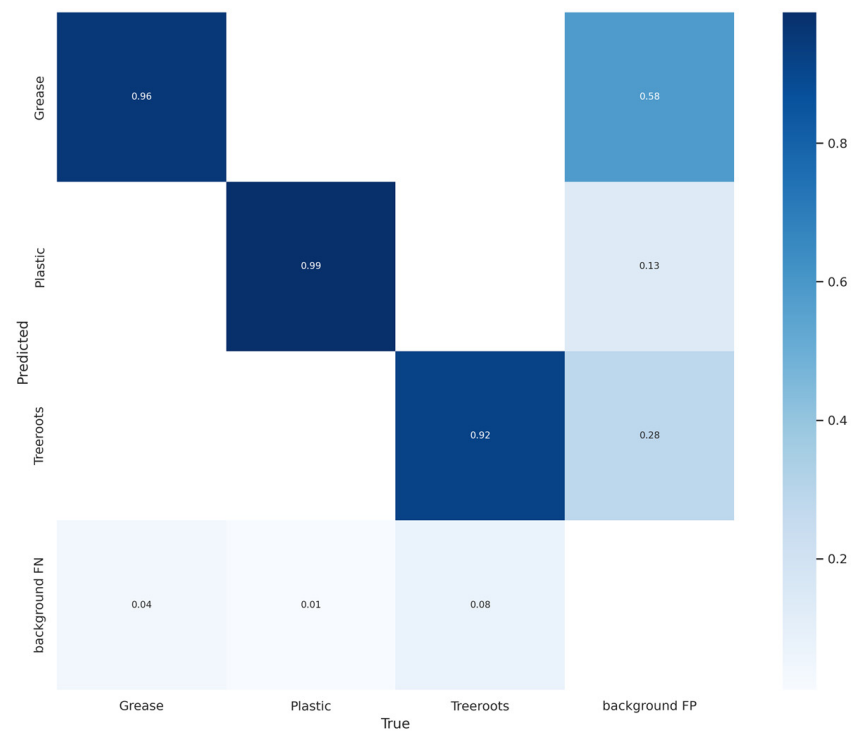


Figure 5. Confusion matrix details for all classes.

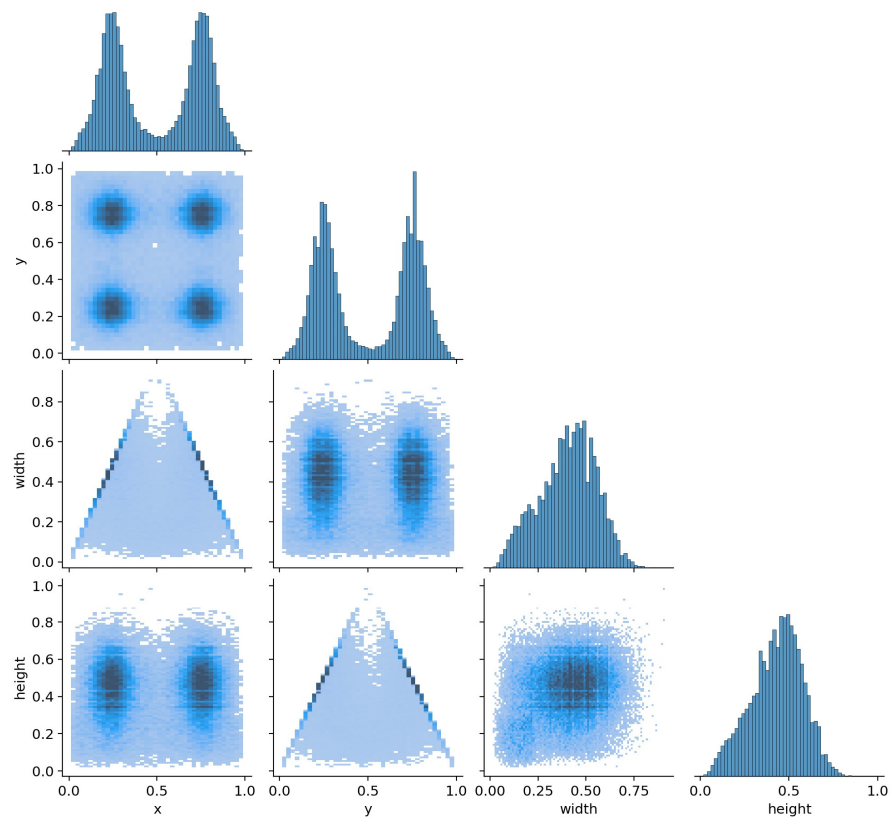


Figure 6. Correlogram for frames detailing.

The scatter diagram, Figure 7, displays the instances in the dataset and their corresponding labels. This visualization helps with understanding the distribution of instances across different classes and assists with identifying potential clustering patterns.

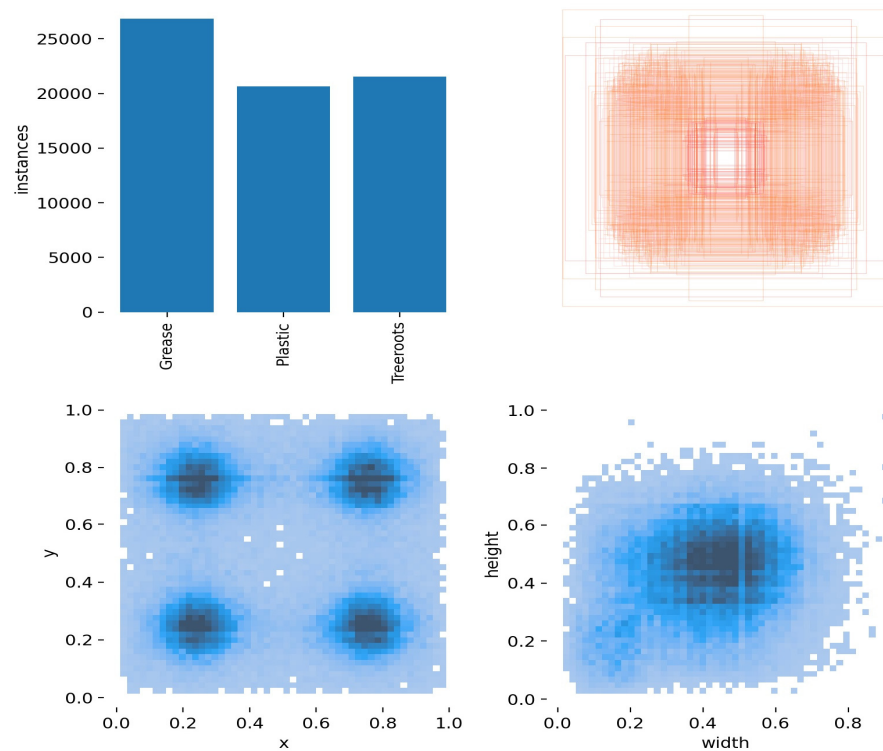


Figure 7. Scatter chart for instances and linked labels.

The graph in Figure 8 illustrates the relationship between precision (P) and confidence (C) that informs concerning changes in the model’s precision at different confidence levels, providing insights into the model’s ability to make accurate detections at various confidence thresholds.

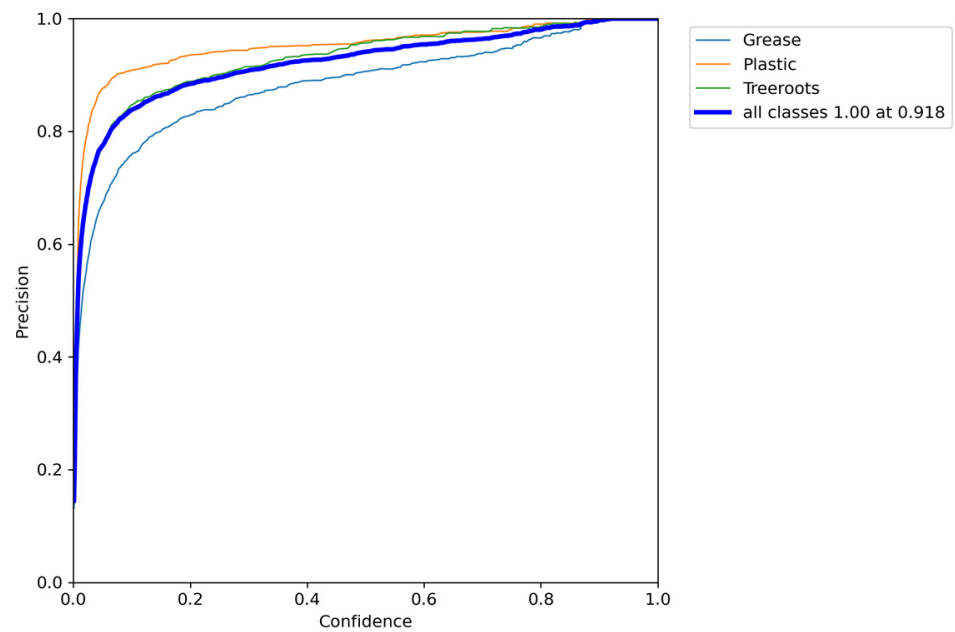


Figure 8. Precision (P) versus confidence (C) chart.

Figure 9 displays the correlation between recall (R) and confidence (C), which clarifies how well the model can recall positive instances at different confidence levels, giving sensitivity details to detection of true positives.

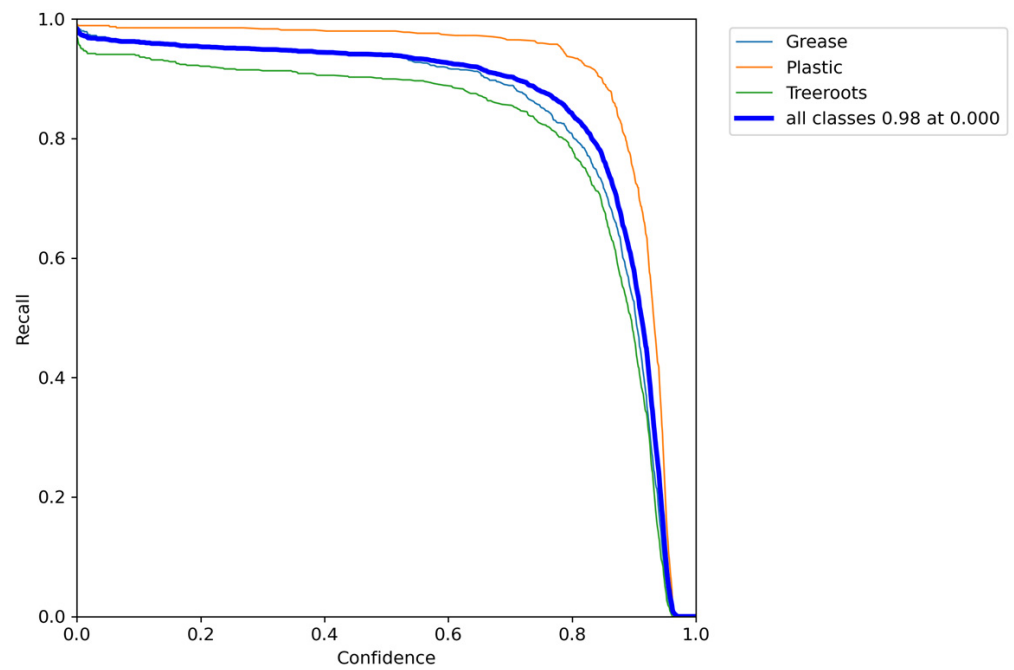


Figure 9. Recall (R) versus confidence (C) chart.

Figure 10 showcases the mean average precision (mAP) of the model, comparing the truth bounding box and the detection box. A higher mAP indicates better overall performance in detecting and localizing objects across all classes.

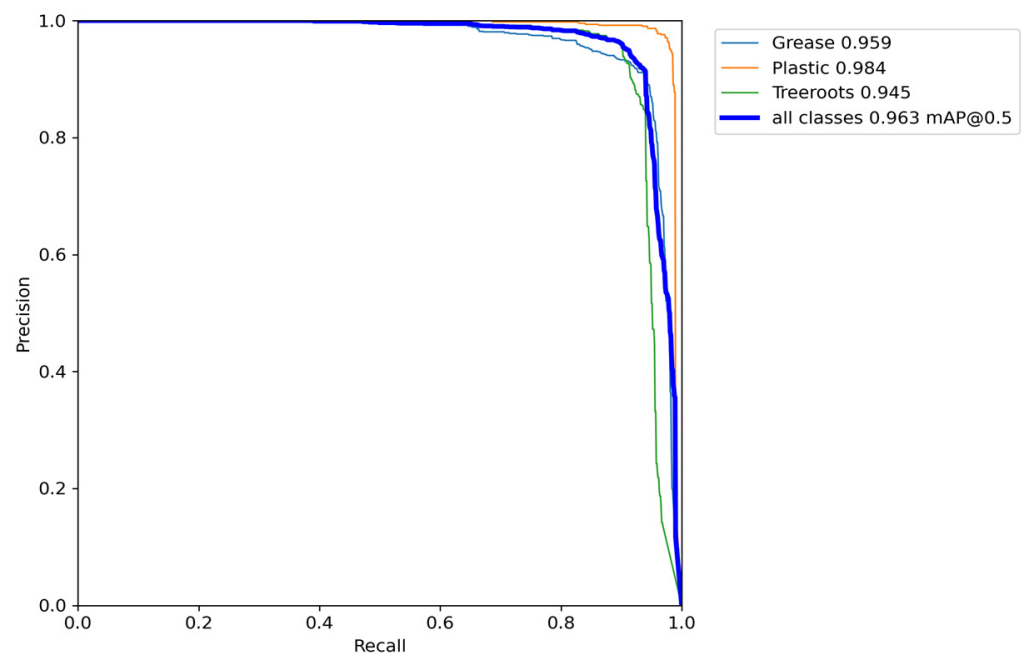


Figure 10. Precision (P) versus recall (R) chart.

Figure 11 exhibits the F1 score at a 94% threshold with a confidence level of 0.566. The F1 score considers both precision and recall, making it a valuable metric for assessing model performance.

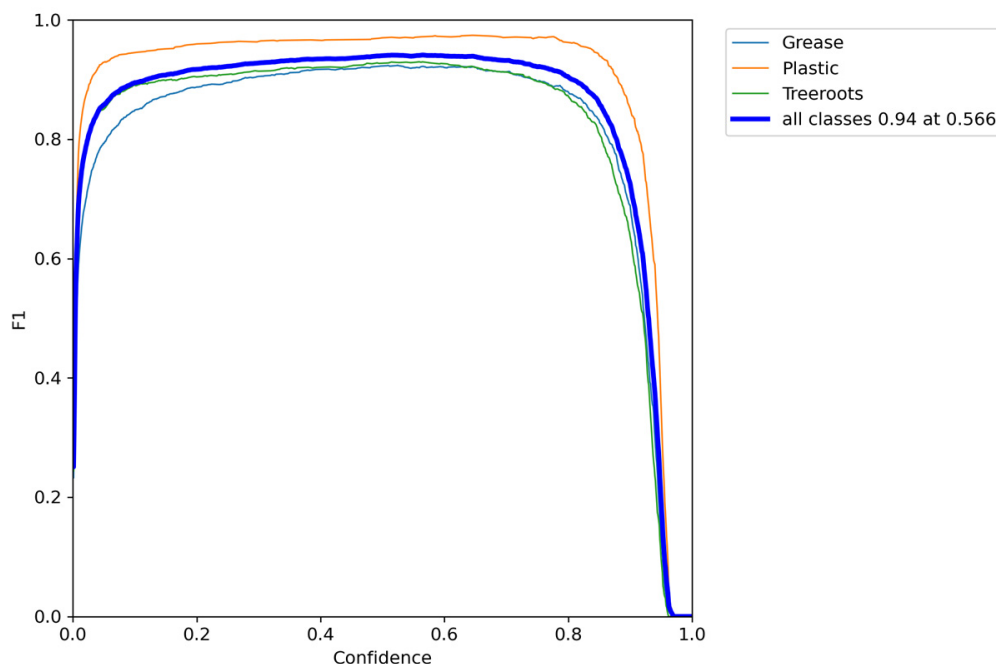


Figure 11. F1 score versus confidence (C) chart.

Figure 12 exhibits the training and validation losses of the detection model over 932 epochs on the S-BIRD dataset. This graph helps in understanding the model’s learning progress during training and validation phases. A decrease in loss indicates that the model is learning to make better predictions.

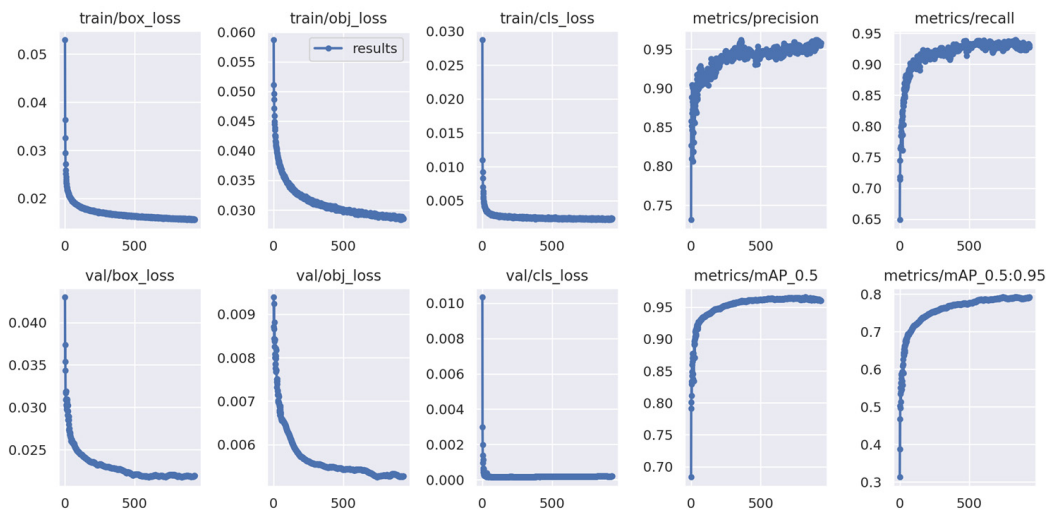


Figure 12. Detailing of losses in training and validation.

Figure 13 exhibits the detection outcomes obtained by deploying the trained model on Google Source frames [22–27] as input data. The outcomes include the location of objects and corresponding class labels (tree roots, grease, or plastic) predicted by the model. These results are of utmost importance as they enable a thorough evaluation of the model’s performance and adaptability when dealing with new and diverse data in real-world scenarios. Additionally, the model has been specifically optimized to handle multiple sewer blockages within the same frame, making it highly suitable for real-time detection in various practical situations.

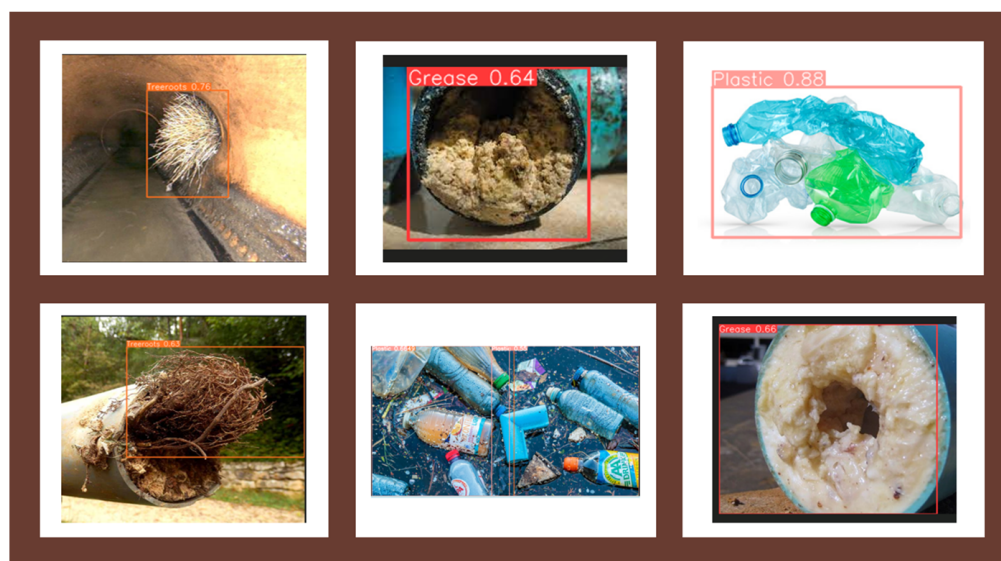


Figure 13. Identification and localization outcomes.

5. Comparing AI-Driven Approach to MOEAs

The AI-driven approach presented in this research offers several advantages over Multi-Objective Evolutionary Algorithms (MOEAs) [28] commonly used in wastewater system management. While MOEAs such as NSGA-II, SPEA2, MOPSO, and MODE are effective at optimizing multiple objectives, they often come with the burden of complex mathematical models and high computational requirements [29,30]. In contrast, the AI approach leverages advanced computer vision and deep learning techniques to detect sewer blockages promptly and accurately. The model achieves a remarkable mean average precision (mAP) of 96.30% at a confidence threshold of 0.5, highlighting its exceptional precision in sewer blockage detection, which in turn enhances the reliability and efficiency of wastewater management systems.

Furthermore, the AI approach relies on labelled training data and lightweight deep learning models, enhancing its efficiency and real-time capabilities. This aligns well with the urgent need to address sewer blockages swiftly and prevent disruptions and overflows. The model's accuracy, speed, and specialized focus on sewer blockage detection make it a highly promising solution for immediate and effective urban wastewater system management. In comparison, MOEAs such as the sensitivity-based adaptive procedure (SAP) [31], optimal control algorithms [32], and novel methodologies [33] have shown efficiency in various aspects of wastewater management, such as sewer rehabilitation and optimal scheduling. However, their computational demands and reliance on complex algorithms might hinder their real-time applicability. The AI-driven approach's ability to process data in real-time, coupled with its high accuracy in detection, gives it a distinct edge for addressing dynamic and critical scenarios like sewer blockages.

Overall, while both AI-driven approaches and MOEAs contribute to the advancement of wastewater management, the AI approach's ability to quickly detect and respond to sewer blockages makes it particularly well-suited for immediate, on-the-ground applications in modern urban sanitation systems.

6. Conclusions

This research highlights the potential of artificial intelligence, by employing the YOLOv5 single-stage detection model and transfer learning on the critical S-BIRD image dataset in sewer blockage detection. By harnessing the power of AI, we achieved a high precision rate suitable for real-time deployment on resource-constrained mobile devices.

Based on the current work, the following specific conclusions may be made.

- The developed model demonstrated noticeable precision and recall rates, achieving 94.50%, 95.90%, and 98.40% average precision for tree roots, grease, and plastics, respectively. The mean average precision (mAP) reached an outstanding 96.30% at a confidence threshold of 0.5 and maintained consistent performance at mAP of 79.20% across IoU thresholds ranging from 0.5 to 0.95, indicating the model's proficiency in handling different sewer blockage scenarios. The inference times were efficient, making the model suitable for real-time applications. The detection outcomes on Google Source frames further validated the model's adaptability to diverse data.
- The results emphasize the effectiveness of transfer learning and fine tuning, reducing training time, enhancing performance, and in adapting deep neural network models to new contexts.
- The presented model's ability to accurately detect sewer blockages holds promise for its application in modern wastewater management systems. The AI-driven sewer blockage detection system showcased in this research has significant implications for real-world applications, ranging from urban infrastructure management to environmental conservation.

As AI technologies continue to advance, the integration of computer vision and deep learning models will pave the way for more efficient and intelligent solutions in various new domains.

Author Contributions: Conceptualization, R.R.P., M.Y.M. and R.K.C.; methodology, R.R.P.; software, R.R.P.; dataset creation, R.R.P.; validation, R.R.P., M.Y.M. and R.K.C.; formal analysis, R.R.P., M.Y.M. and R.K.C.; investigation, R.R.P.; writing—original draft preparation, R.R.P.; writing—review and editing, R.K.C. and R.R.P.; visualization, R.K.C. and R.R.P.; project administration, R.K.C., M.Y.M. and S.M.A.; and funding acquisition, R.K.C. All authors have read and agreed to the published version of the manuscript.

Funding: The research visit of R.R.P. is funded by project PEERS (UTF 2020/10131). The publication charges for this article have been funded by the publication fund of UiT The Arctic University of Norway.

Data Availability Statement: The research data will be made available on the request.

Acknowledgments: Authors acknowledge the support from SPRING EU-India Project (No. 821423 and GOI No. BT/IN/EU-WR/60/SP/2018) and UiT The Arctic University of Norway, Narvik, Norway, for the Ph.D. studies of Ravindra R. Patil. We extend our thanks to ADY Patil School of Engineering, Pune, India.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [[CrossRef](#)] [[PubMed](#)]
2. Kaur, R.; Singh, S. A comprehensive review of object detection with deep learning. *Digit. Signal Process.* **2022**, *132*, 103812. [[CrossRef](#)]
3. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, *111*, 257–276. [[CrossRef](#)]
4. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
5. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
6. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
7. Ultralytics/yolov5. Available online: <https://github.com/ultralytics/yolov5> (accessed on 9 June 2023).
8. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37.
9. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. Centernet: Keypoint triplets for object detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6569–6578.
10. Law, H.; Deng, J. Cornernet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 734–750.

11. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
12. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
13. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *39*, 1137–1149. [[CrossRef](#)] [[PubMed](#)]
14. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6154–6162.
15. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object detection via region-based fully convolutional networks. In Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, 5–10 December 2016.
16. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014.
17. Long, M.; Cao, Y.; Wang, J.; Jordan, M. Learning transferable features with deep adaptation networks. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 97–105.
18. Vaidya, O.S.; Patil, R.; Phade, G.M.; Gandhe, S.T. Embedded Vision Based Cost Effective Tele-operating Smart Robot. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*, 1544–1550.
19. Patil, R.R.; Vaidya, O.S.; Phade, G.M.; Gandhe, S.T. Qualified Scrutiny for Real-Time Object Tracking Framework. *Int. J. Emerg. Technol.* **2020**, *11*, 313–319.
20. Patil, R.R.; Ansari, S.M.; Calay, R.K.; Mustafa, M.Y. Review of the State-of-the-art Sewer Monitoring and Maintenance Systems Pune Municipal Corporation-A Case Study. *TEM J.* **2021**, *10*, 1500–1508. [[CrossRef](#)]
21. Patil, R.R.; Mustafa, M.Y.; Calay, R.K.; Ansari, S.M. S-BIRD: A Novel Critical Multi-Class Imagery Dataset for Sewer Monitoring and Maintenance Systems. *Sensors* **2023**, *23*, 2966. [[CrossRef](#)] [[PubMed](#)]
22. Google Source Images. Available online: <https://www.drainmasterohio.com/red-flags-of-tree-root-intrusion-in-your-drain-pipes/> (accessed on 30 June 2023).
23. Google Source Images. Available online: <https://arboriculture.files.wordpress.com/2016/02/treerootpipe.jpg> (accessed on 30 June 2023).
24. Google Source Images. Available online: https://spunout.ie/wp-content/uploads/elementor/thumbs/Plastic_bottles_in_the_sea-q0ubkb8pkwa5boehpaj6o0v1e8l43mla862l6488o.jpg (accessed on 30 June 2023).
25. Google Source Images. Available online: <https://bbwsd.com/wordpress/wp-content/uploads/2018/03/FOG-850x425.jpg> (accessed on 30 June 2023).
26. Google Source Images. Available online: <https://images.squarespace-cdn.com/content/v1/55e97d2de4b0a47f46957437/1499308890029-VM48EFRJMCOSOFFHFETV/iStock-482437666.jpg?format=1000w> (accessed on 30 June 2023).
27. Google Source Images. Available online: <https://www.istockphoto.com/photo/plastic-bottles-isolated-on-white-gm1202347223-345153972> (accessed on 30 June 2023).
28. Wang, Z.; Pei, Y.; Li, J. A Survey on Search Strategy of Evolutionary Multi-Objective Optimization Algorithms. *Appl. Sci.* **2023**, *13*, 4643. [[CrossRef](#)]
29. Jiang, L.; Geng, Z.; Gu, D.; Guo, S.; Huang, R.; Cheng, H.; Zhu, K. RS-SVM machine learning approach driven by case data for selecting urban drainage network restoration scheme. *Data Intell.* **2023**, *5*, 413–437. [[CrossRef](#)]
30. Yazdi, J. Rehabilitation of urban drainage systems using a resilience-based approach. *Water Resour. Manag.* **2018**, *32*, 721–734. [[CrossRef](#)]
31. Cai, X.; Shirkhani, H.; Mohammadian, A. Sensitivity-based adaptive procedure (SAP) for optimal rehabilitation of sewer systems. *Urban Water J.* **2022**, *19*, 889–899. [[CrossRef](#)]
32. Rathnayake, U. Migrating storms and optimal control of urban sewer networks. *Hydrology* **2015**, *2*, 230–241. [[CrossRef](#)]
33. Draude, S.; Keedwell, E.; Kapelan, Z.; Hiscock, R. Multi-objective optimisation of sewer maintenance scheduling. *J. Hydroinform.* **2022**, *24*, 574–589. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

