

UNIVERSITY OF SOUTHERN QUEENSLAND

Faculty of Health, Engineering and Sciences

**Development of Smart Sewer Pipeline Fault Detection Method
using CCTV Inspection Data and Deep Learning**

A dissertation submitted by

Kyal Sharpe

in fulfilment of the requirements of

ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Honours) (Civil)

Submitted 13 October 2022

Abstract

Sewer networks are complex public infrastructure assets designed to deliver wastewater from property connection points to wastewater treatment facilities. Like all assets, sewer networks require ongoing inspection and rehabilitation to maintain an acceptable level of service which often requires generous resource allocation due to the complex infrastructure. For many municipalities and service authorities, large segments of existing sewer networks are approaching end of life, increasing the demand for maintenance and resources. Maintenance and rehabilitation of sewer networks are primarily founded on investigation programs that determine the existing asset condition, and consequently, the rehabilitation method. Current practice generally involves deploying a robotic closed circuit television video (CCTV) camera to inspect the network; this is an efficient process in isolation; however it commonly requires an extensive manual review process to determine faults within the network.

In this research, an automated fault detection model is developed to review and analyse CCTV inspection footage, and then locate and categorise faults within the data. The study utilises emerging deep transfer learning technology to locate and categorise abnormalities in the input data, based on a predetermined calibration dataset. The smart sewer detection model is adapted from the YOLOv2 object detection framework; twelve common convolutional neural networks were evaluated to determine the optimal feature extraction network, with Res-Net 101 determined to be the preferred network. Performance evaluation of the model included a mAP of 89.3%, and detection speed of 46.3fps which exceeds real-time capability. The success of this project will provide value in improved efficiencies, reliable and consistent fault detection and overall economic benefit to industry, and other users who may employ the findings of this research. Overall, the smart sewer detection model demonstrates capacity to significantly decrease the time required for data review and improve the overall accuracy of decision-making process by removing human error.

Keywords: *Sewer Pipeline Faults, Deep Learning, Object Detection*

UNIVERSITY OF SOUTHERN QUEENSLAND

Faculty of Health, Engineering and Sciences

ENG4111 & ENG4112 Research Project

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences or the staff of the University of Southern Queensland.

This dissertation report is an educational exercise and has no purpose of validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and any other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Certification of Dissertation

I certify that the ideas, design and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.



Kyal Sharpe



Acknowledgements

I would like to acknowledge my supervisor Dr. Andy Nguyen for his guidance and mentorship throughout this research project, as his assistance has been instrumental to the success of this research. Additionally, I extend my appreciation to UniSQ Research Assistant Canh Long Nguyen, for his assistance with MATLAB programming and coordination of UniSQ computational systems.

I would like to thank my employer, Albury City Council, for the extensive database of CCTV inspection data utilised within this research; access to the database was imperative to this project.

Lastly, I would like to thank my partner Alina, for her unconditional support throughout this dissertation and my broader studies.

Table of Contents

Limitations of Use.....	3
Certification of Dissertation.....	4
Acknowledgements.....	5
Table of Contents.....	6
List of Figures.....	10
List of Tables.....	12
Nomenclature.....	13
1 Introduction.....	14
1.1. Research Significance.....	14
1.2. Project Aim and Objectives.....	14
1.3. Background.....	16
1.3.1. Sewer Networks.....	16
1.3.2. Autonomous Fault Detection.....	17
1.4. Dissertation Structure.....	19
2. Literature Review.....	21
2.1. History of sewer networks.....	21
2.2. Inspection techniques for sewer mains.....	23
2.3. Sewer Pipeline Defects.....	24
2.4. Pipeline Rehabilitation Techniques.....	28
2.5. Artificial Intelligence.....	29
2.5.1. Machine Learning.....	30
2.5.2. Deep Learning & Neural Networks.....	30
2.5.2.1. Deep Transfer Learning.....	31
2.5.3. Computer Vision Techniques.....	32
2.5.3.1. Image Classification.....	33
2.5.3.2. Object Detection.....	33

2.5.3.3.	Semantic Segmentation and Instance Segmentation	35
2.5.4.	Neural Network Models.....	37
2.5.4.1.	Feature Extraction CNNs.....	37
2.5.4.2.	Object Detection Models	41
2.5.4.3.	Network Training Hyperparameters	44
2.5.4.4.	Model Overfitting & Underfitting	48
2.5.4.5.	Evaluation Metrics.....	49
2.5.5.	Application of Deep Learning	52
2.5.5.1.	Application of Object Detection to Sewer Pipeline Faults.....	53
2.6.	Data Processing & Sampling	54
2.6.1.	Data Augmentation	55
2.6.2.	Data Annotation Techniques.....	57
2.6.3.	Training, Validation and Test Datasets.....	59
2.7.	Feasibility and Justification.....	59
3.	Methodology	61
3.1.	Architecture of Smart Sewer Detection Model	62
3.1.1.	YOLOv2 Object Detector	62
3.1.2.	CNN Backbone	63
3.2.	Model Development and Evaluation.....	63
3.2.1.	Data Preparation.....	63
3.2.1.1.	Data Collection & Annotation.....	63
3.2.1.2.	Video Sampling	64
3.2.1.3.	Image Resizing	64
3.2.1.1.	Training, Validation and Test Datasets	67
3.2.1.2.	Training Data Augmentation	68
3.2.2.	Comparative Analysis of CNNs.....	69
3.2.3.	Optimisation of Training Hyperparameters	69

3.2.3.1.	Model Training Hyperparameters	69
3.2.4.	Evaluation of Detector Performance.....	70
3.2.1.	Multiclass Object Detection.....	71
3.3.	Project Resources	71
3.3.1.	MathWorks MATLAB.....	72
3.3.2.	CCTV Sewer Inspection Data.....	72
3.3.3.	Computer System for Analysis	72
3.4.	Project Consequences & Ethical Assessment	73
4.	Data Analysis & Discussion	74
4.1.	Initial Model Structure & Parameters	74
4.1.1.	Initial Training Hyperparameters.....	74
4.2.	Analysis of CNNs.....	75
4.2.1.	Comparison 1: Preliminary Analysis	75
4.2.2.	Comparison 2: Detailed Analysis	77
4.3.	Evaluation of Hyperparameters.....	80
4.3.1.	MiniBatchSize.....	80
4.3.2.	Maximum Epochs	82
4.4.	Alternative Defect Classes	84
4.4.1.	Multiclass Detection Performance	85
4.5.	Discussion	88
4.5.1.	Smart Sewer Detection Model	89
4.5.2.	CNN Evaluation Findings.....	89
4.5.3.	Network Hyperparameter Optimisation Findings.....	90
4.5.4.	Multiclass Object Detection.....	92
4.5.1.	Limitations of Research	93
5.	Conclusion	94
5.1.	Research Outcomes	94

5.2.	Benefit to Industry.....	95
5.3.	Future Research.....	95
5.3.1.	Field Deployment.....	96
5.3.2.	Graphical User Interface	96
5.3.3.	Suggested Repair Methodology	96
	Bibliography	98
	Appendix.....	110
5.4.	Appendix 1 – Project Specification.....	110
5.5.	Appendix 2 – Risk Assessment.....	111
5.6.	Appendix 3 – Smart Sewer Detection Model Code	118
5.7.	Appendix 4 – Data Augmentation Function	123

List of Figures

Figure 2.1: Collapsed sewer main invert (Albury City Council 2021).....	23
Figure 2.2: Sewer Pipeline Defects presented by Moradi (2020).....	27
Figure 2.3: Comparison of machine learning and deep learning processes (Oppermann, 2019)	31
Figure 2.4: Comparison of traditional machine learning and transfer learning (Pan & Yang, 2010).....	32
Figure 2.5: Comparison of object detection and image classification (Hulstaert, 2018).....	34
Figure 2.6: Basic CNN Structure by Zhang et al. (2019).....	35
Figure 2.7: Network architecture of YOLOv2, adapted by Liu et al. (2018).....	35
Figure 2.8: Comparison of object detection, semantic segmentation and instance segmentation, adapted from ByteBridge (2021).....	36
Figure 2.9: Architectures for common versions of Res-Net CNN (He et al., 2016).....	40
Figure 2.10: IOU calculation.....	49
Figure 2.11: Precision-recall curve for YOLOv2 detector.....	51
Figure 2.12: Review of computer vision technologies to automated sewer pipeline fault detection by Moradi (2019).....	53
Figure 2.13: mAP and AP of the model using different dataset sizes Cheng and Wang (2018)	53
Figure 2.14: Randomised Data Augmentation Examples.....	56
Figure 2.15: Video Labeller App Annotation Example (MathWorks, n.d.).....	58
Figure 3.1: Model development & evaluation methodology.....	61
Figure 3.2: Flow Diagram of Smart Sewer Detection Model.....	62
Figure 3.3: Label summary graph for CCTV inspection data annotated via Video Labeller App (MathWorks, n.d.).....	64
Figure 3.4: Sewer Inspection Image with cracking (Albury City Council Sewer Inspection Program, 2021).....	66
Figure 3.5: Sewer Inspection Image with cracking (resized) (Albury City Council Sewer Inspection Program, 2021).....	67
Figure 3.6: Example of training data after transformation function.....	68
Figure 4.1: Darknet-19 CNN preliminary evaluation:.....	76
Figure 4.2: Res-Net 50 CNN preliminary evaluation.....	76

Figure 4.3: Res-Net 101 CNN preliminary evaluation	77
Figure 4.4: CNN Comparison on primary dataset	77
Figure 4.5: Precision-recall curve for median Res-Net 101 result.....	78
Figure 4.6: Plot of Training & Validation Loss for Res-Net 101	79
Figure 4.7: CNN Comparison on primary dataset	79
Figure 4.8: Analysis of ‘MiniBatchSize’ training hyperparameter	81
Figure 4.9: Analysis of ‘MaxEpochs’ training hyperparameter	83
Figure 4.10: Precision-recall curve for Res-Net 101 trained for 30 epochs	84
Figure 4.11: Precision-recall curve for ‘cracking’ defects.....	86
Figure 4.12: Precision-recall curve for ‘deposit’ defects.....	87
Figure 4.13: Precision-recall curve for ‘root intrusion’ defects.....	87
Figure 4.14: Training & validation loss for multiclass defect training.....	88
Figure 4.15: Architecture of ‘Smart Sewer Detection Model’	89
Figure 4.16: Example of multiclass detection.....	92

List of Tables

Table 2.1: Sewer Pipeline Defects (Albury City Council Sewer Inspection Program, 2021) .	27
Table 2.2: Description of generic layers within a CNN (Zhang et al., 2019)	35
Table 2.3: Summary of common CNNs (MathWorks, n.d.)	38
Table 2.4: Comparison of YOLO and Fast/Faster R-CNN on VOC 2007 dataset (Redmon et al., 2017)	42
Table 2.5: Comparison of YOLOv2 other detectors on VOC 2007+2012 dataset (Redmon & Farhadi (2017)).....	43
Table 2.6: Training options for object detection model training, adapted from MathWorks (n.d.)	46
Table 2.7: Comparative results for different object detection frameworks, adapted from Moradi (2020).....	54
Table 2.8: Video Labeller App Automated Labelling Algorithms (MathWorks, n.d.)	59
Table 3.1: Training, Validation and Test Datasets	67
Table 3.2: Summary of CNNs for analysis	69
Table 3.3: Training options for object detection model training, adapted from MathWorks (n.d.)	70
Table 3.4: Summary of Computing Resources	72
Table 4.1: Key training hyperparameters for analysis	74
Table 4.2: Summary of initial CNN backbone analysis	75
Table 4.3: Hyperparameters for model training, reduced from Section 2.5.4.3 (MathWorks, n.d.)	80
Table 4.4: Key training hyperparameters for analysis	82
Table 4.5: ‘MaxEpochs’ Analysis Summary Tabulated	83
Table 4.6: Multiclass Dataset Information.....	85
Table 4.7: Summary of model information for multiclass analysis.....	85
Table 4.8: Multiclass Object Detection Results.....	86
Table 4.9: AP results for various ‘MiniBatchSize’	91
Table 5.1: Example relationship between defect and rehabilitation method.	97

Nomenclature

AI Artificial Intelligence

ANN Artificial Neural Network

AP Average Precision

CCTV Closed-Circuit Television

CNN Convolutional Neural Network

CNNs Convolutional Neural Networks

FPS Frames per Second

mAP Mean Average Precision

SSD Single Shot Multibox Detector

YOLO You Only Look Once

1 Introduction

1.1. Research Significance

This research is expected to provide a variety of benefits, including but not limited to:

- Increased efficiency in review of sewer inspection data and consequently, development of asset rehabilitation programs.
- Increased accuracy and reliability of fault detection through systematic analysis and decision making.
- Provide direct economic benefit and improved delivery timelines to industry, due to the reduced human input required in review and analysis of data.
- Real-time fault detection capabilities suitable for deployment in the field.

1.2. Project Aim and Objectives

The aim of this project is to reduce inefficiencies in asset data review through the development of an autonomous object detection model via deep learning. The project seeks to develop a methodology for the implementation of deep learning object detection to data extracted from closed-circuit television (CCTV) sewer inspection data.

The project will contribute to previous research conducted on computer vision by applying deep learning techniques to automate fault detection in sewer pipelines. The objective of providing improved efficiency to industry through computer vision application has not been well explored to date, particularly in relation to sewer infrastructure.

Utilising CCTV data sourced from an NSW local government sewer inspection program, the autonomous model shall aim to detect and classify various faults in sewer pipelines via multi-class object detection, providing tangible value to industry. This will provide the framework to reduce the extensive user input currently required to assess and categorise sewer pipeline condition from CCTV inspection data.

Considering the above, the objectives of this project include:

- To develop an extensive database utilising existing sewer pipeline inspection CCTV footage.
- Develop an autonomous object detection model for sewer pipeline faults, utilising computer vision technology and deep transfer learning.
- Evaluate and refine the developed model to achieve a high-confidence level and detection speed (i.e., average precision >0.8, real-time detection).

Despite the proposed scope of this project primarily relating to inspection data gained from Albury City Council's sewer rehabilitation program, value exists for other authorities and asset owners through the project objectives. The proposed research aims to reduce inefficiencies for a diverse range of users by providing a reliable, systematic analysis tool that will be applicable to a wide range of sewer pipelines.

1.3. Background

1.3.1. Sewer Networks

A sewer network is a complex system of pipes and associated infrastructure designed to transfer wastewater flow from the customer to a treatment plant. Sewer networks are generally managed and maintained by the municipal council or in some instances, a privatised authority. The size of these systems is influenced by factors of the service area, including area and number of connections; wastewater networks commonly include thousands of kilometres of pipeline. The large magnitude and complexity of many sewer networks results in complex maintenance programs that are both time consuming and expensive.

Similar to other public infrastructure assets, wastewater networks require inspection, rehabilitation, and renewal programs to ensure the required level of service to the paying customer is maintained. As many sewer mains possess a nominal diameter less than 600mm, manual entry is not possible and thus, conventional inspection methods include various forms of camera/CCTV survey. This camera survey is commonly completed with a remotely controlled robotic camera, or in smaller instances, manually with a series of rods or continuous reel. Most commonly, analysis of collected data involves manually viewing and annotating the recorded videos; a time consuming and mundane task that relies on the ability of the user to provide accurate and reliable results.

Inspection programs aim to identify the condition of the existing asset, inclusive of any recognised faults or issues. Common sewer faults may include:

- Spalling or flaking of the pipe;
- Cracking;
- Voids or collapses;
- Infiltration;
- Deposits; and
- Root ingress/blockages.

The lifespan of sewer mains can often be renewed or extended without the need to construct a new sewer main through conventional excavation techniques, due to a variety of rehabilitation methodology available. Rehabilitation of existing sewer mains can be completed utilising a variety of techniques, typically tailored to the identified fault type. Common methods of

rehabilitation include jetting and root cutting, patching, insitu relining and pipe-bursting; all methods are considered trenchless techniques, meaning rectification of a defect is completed without the requirement of highly invasive replacement works. Notwithstanding, in some instances where severe defects exist, namely sagging or inclined pipe segments, conventional trenched excavation must still be undertaken (Loss et al., 2018). The primary benefits presented by trenchless rehabilitation methods include reduced renewal cost, reduced works duration and significantly less disruption to the community and surrounding locality with most techniques simply requiring access to existing sewer manholes (Loss et al., 2018). Furthermore, trenchless pipe rehabilitation is often the only viable option in areas of high service congestion, such as inner-city urban environments (Lueke and Ariaratnam, 2001).

The sewer pipeline CCTV data utilised in this dissertation is provided by Albury City Council; a regional NSW council located near the border of New South Wales and Victoria. Albury's municipality spans in excess of 300km², with a population of approximately 55,000 people (Population Australia, 2022). A service population and area of this magnitude requires a complex wastewater system to deliver the required level of service to the community, with AlburyCity's sewer network including approximately 509 kilometres of gravity sewer mains (Albury City Council, 2019). Sewer inspections are conducted through an annual CCTV and cleaning program; this data is then used to develop a rehabilitation works program that is delivered via external specialist contractors. Analysis of the data obtained through inspection is completed manually by a member of staff; this process can take several weeks to complete each year. This process has been identified as a significant inefficiency due to the resource commitment required because of the large magnitude of data obtained. Due to the onerous nature of this data review, it is highly likely that numerous inaccuracies exist due to fatigue experienced by the person(s) completing the data review. Consequences of inaccurate asset data include unforeseen costs due to inaccuracies in the resultant works program, or complete omission due to overestimated remaining asset life. A smart analysis technique built upon machine learning will facilitate efficient, consistent and accurate review of raw asset data will provide fundamental benefit to the overarching asset management system.

1.3.2. Autonomous Fault Detection

Autonomous image and video detection is an emerging technology becoming prominent within the infrastructure industry and modern society as a whole. The abilities of autonomous detection technologies subsidise the requirement for human resourcing in tasks such as data review and repetitive recognition activities. Autonomous recognition of images and videos is

achieved through techniques such as image classification and object detection, which rely on deep learning networks. Deep learning is a function of machine learning; a fundamental subfield of artificial intelligence (Ongsulee, 2017). Whilst automated classification has existed for decades in varying formats, application has diversified exponentially in recent years with the progression of deep learning, in lieu of prior handcrafting techniques (Bharadi et al., 2017).

Neural Networks were first proposed in 1944, being defined as “a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state in response to external inputs” (Bharadi et al., 2017). These neural networks are the foundation to machine learning techniques, including deep learning. The most common form of neural network currently utilised for autonomous image detection are Convolutional Neural Networks (CNNs); CNNs are primarily used to solve difficult image-driven pattern recognition tasks (O'Shea and Nash, 2022), which aligns closely with the intent of this research.

CNNs are typically considered to be closely related with traditional Artificial Neural Networks (ANNs), however notable differences exist. Whilst both ANNs and CNNs self-optimize through learning, CNNs are primarily used in the field of pattern recognition within images; hence, image specific features can be integrated into the network architecture (O'Shea and Nash, 2022). Image recognition deep learning benefits from the nature of CNNs, as the program learns complex concepts, by building out from simpler ones (Goodfellow, Bengio and Courville, 2016).

Neural Networks and Deep Learning form the basis of fault detection through techniques such as image classification and object detection. Image classification utilises deep learning to categorise images into defined classes based on the content of the image; supervised or unsupervised classification describe the method in which this occurs. Supervised classification relies on a pre-classified training dataset to calibrate the network, whereas unsupervised classification functions without user provided sample classes (Manoj krishna et al., 2018). Deep learning consists of several modules, with the primary being feature extraction and feature classification; these fundamental components enable classification of complex visual data (Manoj krishna et al., 2018).

1.4. Dissertation Structure

This dissertation is comprised of six chapters. Each individual chapter represents a key component that has been completed to achieve the project aim and objectives detailed in Section 1.2. The structure of this dissertation is summarised below:

Chapter 1: Introduction

Chapter one introduces the subject of this dissertation and the research proposed, including the research significance as well as the related aims and objectives. The introduction provides a background into sewer network management and rehabilitation, which is then followed by an overview of autonomous fault detection through artificial intelligence.

Chapter 2: Literature Review

Chapter two reviews the history and associated literature of sewer infrastructure, asset management, artificial intelligence, and computer vision techniques. The chapter begins by discussing the history of sewer networks and the asset management principles behind network maintenance, including inspection programs, defects, and rehabilitation techniques. Following this, Chapter two completes a detailed review of artificial intelligence; machine learning and deep learning subsets. The intricacies of computer vision techniques such as convolutional neural networks and object detection are investigated, including the prior application to industry. The chapter concludes by discussing data processing techniques and the influence they have on computer vision techniques, before providing feasibility and justification of the research proposal.

Chapter 3: Methodology

Chapter three details the methodology implemented to achieve the project aim and objectives, employing information gained through the literature review completed in Chapter two. The methodology begins with an overview of the proposed smart sewer pipeline object detection model, then presents the proposed method for optimisation and evaluation,

Chapter 4: Data Analysis & Discussion

Chapter four focuses on implementation of the methodology outlined in Chapter three, including analysis and review of the results obtained. The data analysis process is separated into multiple stages, namely the preliminary analysis and detailed analysis of CNNs, which is then followed by network training optimisation. Chapter four then reviews application of the model to other defect classes and concludes with a formalised discussion of research findings and outcomes.

Chapter 5: Conclusion

Chapter five concludes this dissertation by outlining the connection between research outcomes and the project aims and objectives. Benefit of this research to industry is discussed, as well as suggested future research.

2. Literature Review

This chapter reviews the history and associated literature of sewer infrastructure, asset management within Albury City Council and the industry wholistically. Artificial intelligence is investigated in detail with particular regard to image classification through deep learning, as well as associated data management. The chapter concludes by reviewing the feasibility and justification of this dissertation.

2.1. History of sewer networks

Sewer networks are a critical component in modern day society and are a fundamental infrastructure required in the effective management of wastewater. The term ‘sewer network’ refers to the infrastructure that conveys wastewater from property connections to wastewater treatment plants; specifically, a sewer network is a complex system of pipes, manholes and pump stations which convey wastewater effluent (Weiner and Matthews, 2003). Components of sewer networks typically operate via gravity flow, however where this is not possible due to topographic conditions, pump stations may be implemented to create a sewer rising main. A sewer rising main refers to a sewer main that operates under pressure from a pump or pumpstation, meaning it can overcome increases in terrain conditions (Weiner and Matthews, 2003).

Sewer networks share characteristics of other civil infrastructure assets, being that they possess a finite lifespan, require recurring maintenance both scheduled and unscheduled and eventually, rehabilitation or renewal. As sewer networks provide an essential service to the public that is subsidised through rates and or taxes, it is imperative that the sewer network provides a satisfactory level of service with respect to performance and reliability. Above ground assets such as road pavements or structures are typically renewed in their entirety when approaching end of life; however, the renewal process is often more complicated for sub-

surface assets. The inherent nature of a piped system, being that in many cases it is buried amongst other services, means renewal by means of conventional replacement can result in significant cost and/or impact to customers. Considering this, it is noted that “proactive and preventive repair strategies for urban drainage systems are often more cost effective than the traditional approach of reactive sewer maintenance” (Fenner 2000). Trenchless rehabilitation techniques such as pipe relining, pipe bursting or patching are explained in detail in Section 2.4, however it is noted that these are typically preferred methods of renewing underground sewer pipelines in contrast to the traditional approach where a new pipeline is constructed through open trenching. Whilst trenchless rehabilitation methods offer a variety of benefits, it is imperative that it is supported by a well-managed asset management program to ensure correct prioritisation and scheduling (Tagherout, Bennis and Bengassem, 2011).

A successful rehabilitation program ensures that renewal of assets is effectively prioritised. Prioritisation of individual assets ensures that rehabilitation occurs at the correct point in the assets service life; this should occur at the end of the effective service life, but prior to failure that may inhibit efficient renewal (Tagherout, Bennis and Bengassem, 2011). In respect to sewer networks, it is most efficient to rehabilitate pipelines prior to major structural defects such as voids occurring. Considering this, an effective asset management program is dependent upon several factors, including knowledge of the systems condition (Tagherout, Bennis and Bengassem, 2011). A proactive inspection program ensures that information about the target asset is accurate, which is imperative for effective asset management. For this to occur, asset inspection must be scheduled on a recurring interval; This allows the life expectancy of an asset to be tracked and will identify irregular defects that may create issues prior to the expected end of life.

Implementation of a strategic long term inspection program will provide a stable basis for the development of an efficient rehabilitation program, minimising annual and overall costs throughout the asset’s lifespan (Park 2009). Historically, many managing authorities have lacked a cohesive plan for the management of water and wastewater systems (Fenner 2000); however, inspection programs have improved in-line with the development of new technologies such as robotic CCTV devices, which have offered improved efficiencies to the development of these programs (Fenner 2000).

Historically, evaluation of captured CCTV data is a manual process that relies on an asset engineer or officer to view and annotate the data, to which a condition rating is applied and/or

faults are identified (Hassan et al. 2019). From this evaluated data, a works program can be derived to rehabilitate sewer mains that are in poor condition or possess faults.

2.2. Inspection techniques for sewer mains

Sewer mains are commonly of a diameter between 150mm and 450mm, however the diameter of large trunk mains can exceed 1200mm. The diameter of sewer mains means that access by an inspector is almost always impossible, and where possible, highly restricted due to the nature of the system. Considering this, alternative techniques must be employed to undertake inspection of pipeline assets. Robotic CCTV and Sewer Scanner Evaluation Technology (SSET) are common forms of sewer inspection utilised in modern society (Yang et al. 2011). These systems are used to record video footage or produce still images of sewer pipelines; this data is then used to assess the structural integrity and inform the overall condition of the asset. For instances outside of large-scale survey, users may utilise manual cameras that require push-rod style operation; this method is generally employed to investigate isolated faults that would not require the automated analysis proposed by this research. Notwithstanding, it is important that the developed program is versatile and reliable, and thus, should possess the capability to analyse all common image or video types that may be encountered.

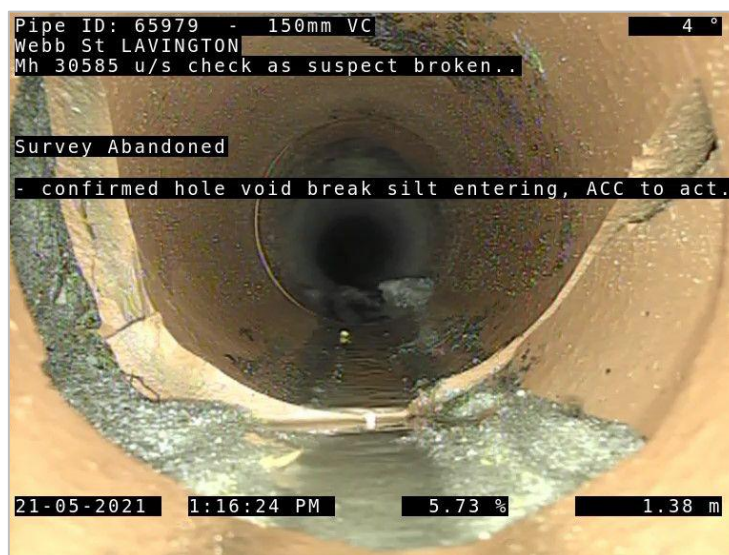


Figure 2.1: Collapsed sewer main invert (Albury City Council 2021)

Sewer inspection is generally completed by entering the sewer main at a nominated manhole, then surveying the length of the chosen segment, terminating the survey at the next manhole (downstream or upstream). This method of inspection generally accounts for all required sewer mains, however in some instances branch lines may not have manhole access and cannot be

accessed by a conventional CCTV robot. In the instance where inspection of a branch line is required, a 'lateral launch camera' can be employed to survey these lines (Pipeworks Inc 2017). The footage obtained from the CCTV survey is recorded in programs such as WinCan, which can allow the operator to associate relevant asset information, record pipeline grade and add comments (WinCan, 2022).

2.3. Sewer Pipeline Defects

Sewer mains are subject to various possible defects that affect the performance or integrity of the network over the lifetime of the asset (Moradi 2020). Depending on the type of defect, the sewer main may experience an abrupt decrease in the level of service, or the lifespan may gradually decrease as a defect worsens. Daher (2015) developed a *Defect-based Condition Assessment Model and Protocol of Sewer Pipelines*, which included comprehensive defect categories and types. The defects were arranged in a hierarchal structure, with high level categories describing the area in the network in which the defect occurs, i.e. pipe segment, pipe joints and manholes; defects were then further categorised as operational or structural. Similar categorisation is found across various research, with Moradi (2020) also employing the operation/structural categorisation model.



Structural defects are defects relating to the condition and integrity of the asset itself; typically, in inspection program will categorise these defects and provide a rating on the severity. Structural pipeline defects may include the following, amongst others:

- Cracking – visible crack in the pipe wall.
- Fracture – complete crack where pieces of the pipe wall appear to become segmented, without being dislodged.
- Spalling – Breaking of the surface into small pieces, visually similar to spalling of other concrete elements and often caused by expansion of internal reinforcement or other exterior forces.
- Hole or collapse – Segment of the pipe wall has become dislodged, size dependent.
- Void – Pipe Invert is severely damage/non-existent.
- Deformation – Internal pipe is no longer a circular cross section, often caused by external forces.
- Roughness defect – Pipe invert and or wall has worn, increasing the friction coefficient, and reducing the integrity of the pipe wall.

Operational defects are typically defined as defects of environmental cause due to insufficient maintenance, and may include the following:

- Root ingress – Tree root intruding through pipe joints, cracks, or holes.
- Infiltration – Ground water ingress into the sewer pipeline.
- Exfiltration – Wastewater leakage out of the pipeline.
- Deposits – Settled or attached foreign materials which restrict the pipes flow.

Images of Common Defects (Albury City Council, 2021):

Defect Name	Image
<p style="text-align: center;">Cracking</p>	
<p style="text-align: center;">Fracture</p>	

Collapse



Void



Root Ingress



Deposits	<div style="border: 1px solid black; padding: 5px;"> <p>Pipe ID: 70748 150mm VC otherside 3 ° Mh 30779 u/s Mh 30857 Lyne St LAVINGTON</p> <p>Survey Abandoned .. otherside .. heavy clean vacuum of cast iron material required.</p> <p>11-03-2021 3:43:31 PM -0.66 % 7.01 m</p> </div>
-----------------	---

Table 2.1: Sewer Pipeline Defects (Albury City Council Sewer Inspection Program, 2021)

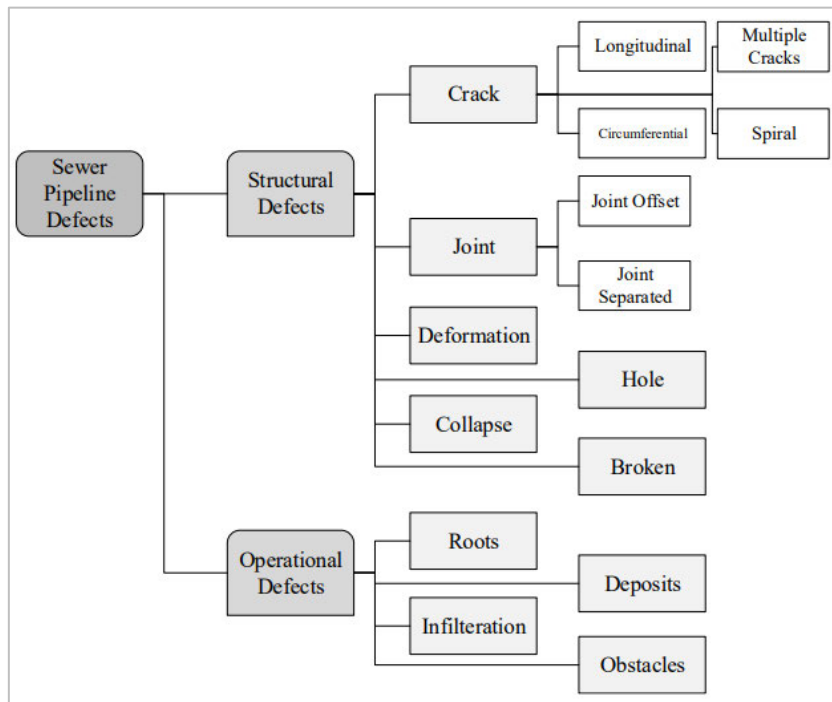


Figure 2.2: Sewer Pipeline Defects presented by Moradi (2020)

In addition to the structural and operation defects presented by Moradi (2020), defects occurring during installation or rehabilitation are also possible. Whilst these defects would typically be identified during a final inspection upon conclusion of construction or rehabilitation works, it is still possible for these defects to exist in a commissioned pipeline. Construction defects may include the following (Daher 2015):

- Obstructions – construction materials left within pipes or manholes may become stuck, creating an obstruction.

- Sagged joints – pipe segments can be knocked off grade during trench backfilling, creating a sag in the pipeline that will cause effluent to pool; this commonly occurs due to inadequate haunching.
- Improper jointing – joints can be improperly seated during construction; if this is not detected during pressure testing, it can lead to early occurrence of other defects such as root ingress and infiltration/exfiltration.

Defects present in sewer pipelines vary significantly in nature and magnitude as outlined above, meaning various repair methods are required to rehabilitation different defects. Additionally, defects also present varying similarities and differences with respect to visual attributes; the impact of this on autonomous recognition through AI will be further discussed in Section 2.5.

2.4. Pipeline Rehabilitation Techniques

Successful rehabilitation of sewer pipelines requires a variety of rehabilitation techniques orientated to the various pipeline defects reviewed in Section 2.3. Rehabilitation methods each require different equipment and incur different resources, hence it is important to understand the relationship between defect classification and appropriate rehabilitation technique. Common pipe rehabilitation techniques are presented below:

- **Pipeline Cleaning** is a maintenance technique and the most common form of pipeline rehabilitation. This method involves the use of a high-pressure jetting apparatus that is pushed through the pipeline segment, removing debris, deposits and obstructions. Generally, jetting units have specialised heads that facilitate the removal and cutting of roots. It is noted that high pressure jetting is utilised prior to the implementation of other rehabilitation methods such as pipe relining (Trenchlesspedia, 2022).
- **In situ Pipe Relining** is a trenchless rehabilitation technique that involves insertion of a flexible liner material into the host pipe, which is then cured through methods such as steam or ultraviolet light (UV). Upon completion of the curing process, property junctions connected to the rehabilitated pipeline can be cut out utilising a camera-guided cutting unit based on CCTV survey completed prior to relining (Westaway, 2001).

In addition to cured-in-place pipe (CIPP) liners, spiral-wound pipeline is also a common pipe relining technique, particularly for larger diameter pipelines. Spiral-wound pipe relining requires a winding machine to install a strip liner to the host pipe; an adhesive resin is used to secure the strip liner together to the host pipe (Ishmuratov et al., 2013).

Pipe relining is a suitable repair method for most defects found in sewer pipelines, however it is noted that the repair method may be unsuitable for large voids or collapses. This is due to the relining method requiring support of the host pipe during the curing/setting process.

- **Isolated Patching** is similar in principle to CIPP pipe relining techniques; however, it is intended to address isolated pipe failures. Patching is commonly undertaken with pneumatic inflation to expand the patch to the host pipe, to which the resin coating then adheres to the host pipe (BMC Corp, 2022).
- **Pipe Bursting** is the most extensive trenchless repair method used to rehabilitate existing pipeline segments between manholes/pits; the technique may also be known as “pipe cracking” and “pipe splitting”, as technique causes the host pipe to split and fracture during installation of the new pipe (Rameil, 2007). Conventional pipe bursting utilises a combination of hydraulic and pneumatic force to pull the new pipe (typically HDPE) through the host pipe, whilst simultaneously breaking the host pipe apart to expand its internal diameter. Pipe bursting can also be completed through a ‘static’ method, where a hydraulically driven bursting rod is driven down the host pipe to break it apart, then pull back a new pipe whilst it retracts and fragments the host pipe (Rameil, 2007). Property junction connections require excavation and fusion welding to reconnect to the new pipe, meaning there is notably more impact on the locality and users when compared with pipe relining techniques. Notwithstanding, pipe bursting is considered a superior technique for large structural defects such as faults and collapses.
- **Excavation Repairs** may still be required in instances where trenchless repair methods are not feasible. If the existing host pipe has a severe sag/deflection, trenchless repair methods will adopt this defect when they are installed. It is common to excavate and repair an isolated section of a pipeline, in combination with a trenchless technique. Additionally, for small segments or open sites, it may be more cost effective to trench and lay a new sewer pipeline than it would be to undertake pipe relining or pipe bursting.

2.5. Artificial Intelligence

Artificial Intelligence (AI) is defined as an intelligent machine(s) capable of demonstrating intelligence similar to that of the human brain, to undertake rational decision making and enable problem solving (IBM Cloud Education, 2022). Numerous industries are utilising AI to

improve efficiency and effectiveness in existing tasks, as well as in the development of emerging technologies such as driverless cars and robotics. AI encompasses the sub-fields of machine learning and deep learning, which are relevant to autonomous fault detection (IBM Cloud Education, 2022).

2.5.1. Machine Learning

Machine learning is a sub field of AI originating in the mid-1900s defined by El Naqa and Murphy (2015) as an “evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment”. Machine learning relies on structured data to learn and develop algorithms that can then be applied to tasks requiring problem solving and critical analysis. Generally, a training dataset will be classified by a human operator which is then utilised to train the machine learning program; this process is commonly known as supervised machine learning (Guyon and Elisseeff, 2008).

The initial categorisation process is part of the dimensionality reduction process known as feature extraction (Chatterjee, 2021). Feature extraction involves the selection and combination of variables within raw data, which are then referred to as features; this process reduces the complexity of the data whilst still maintaining the value and accuracy of the data set. The purpose of the feature extraction process can be summarised as the removal of redundant data, thus requiring less resources to complete the analysis process (Chatterjee, 2021).

Machine learning has a diverse range of applications in modern society, including language detection, speech recognition, self-driving cars, share trading and image recognition, amongst others (Padamkar, n.d.). Advanced applications of machine learning utilise a subset commonly referred to as ‘Deep Learning’, which is based on artificial neural networks (ANNs). This form of machine learning uses a ‘deep’ neural network consisting of numerous layers, capable of learning through its own data processing (Gayhardt et al., 2022).

2.5.2. Deep Learning & Neural Networks

IBM (2020) define deep learning as “neural networks that attempt to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognise, classify, and describe objects within the data. The neural networks are made up of multiple layers of interconnected nodes; as the network learns, new layers build upon the previous to optimise and refine the network’s ability”. The nature of deep learning allows the technology to be utilised across a broad range of applications, commonly image recognition, text recognition in documents, health issue recognition, and have even been used

in self-driving vehicles (Balas et al. 2019). Deep learning networks interpret and classify user input data (i.e. image, videos and/or text) through an intelligent Artificial Neural Network (ANN). ANNs allow unlabelled data to be grouped or sorted according to recognised similarities and patterns among the data samples; however, in terms of classification, the network is trained through supervised learning on a labelled dataset to produce categorical outputs (Oppermann, 2019). A significant advancement of deep learning in contrast to conventional machine learning, is the omitted requirement for feature extraction in pre-processing; this is represented diagrammatically in Figure 2.3:

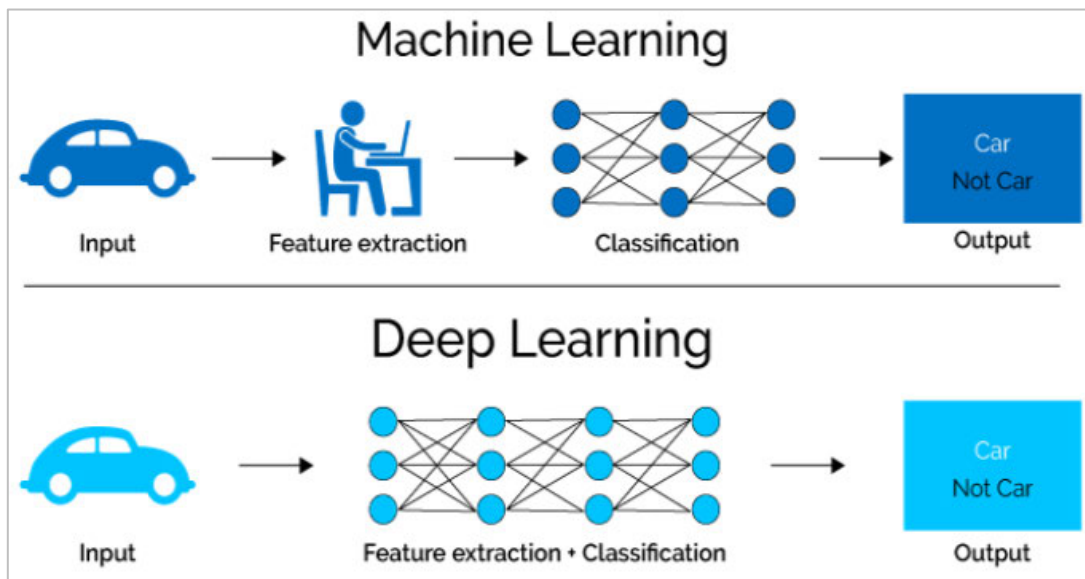


Figure 2.3: Comparison of machine learning and deep learning processes (Oppermann, 2019)

To provide further context on the intelligence of deep learning, Oppermann (2019) explains that a machine learning model must first be taught the unique features of a car by means of manual extraction and input; in the case of deep learning, the manual feature extraction step is completely unnecessary as “the deep learning model would recognize these unique characteristics of a car and make correct predictions”. In the example explained by Oppermann (2019), the user would simply need to define the categories as ‘Car’ and ‘Not Car’. This research investigation aims to apply the deep learning technologies described to identify and classify sewer faults from data collected through CCTV survey.

2.5.2.1. Deep Transfer Learning

Deep transfer learning describes the process of applying a pretrained deep learning network to a new task, with the intent of increased efficiency in contrast to standard learning techniques by leveraging the networks existing knowledge and capability. Pan and Yang (2010) provided a definition that “transfer learning aims to extract the knowledge from one or more source tasks

and applies the knowledge to a target task”. This is differentiated from conventional multitask learning as the primary aim of transfer learning is the target task, in contrast to the intent to learn the source and target tasks simultaneously (Pan & Yang, 2010); this is represented in Figure 2.4 below, adapted from the research by Pan and Yang (2010):

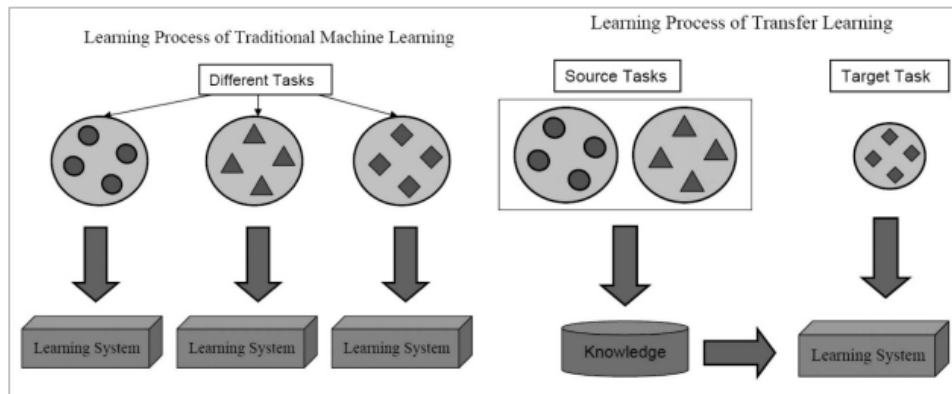


Figure 2.4: Comparison of traditional machine learning and transfer learning (Pan & Yang, 2010)

The transfer learning technique provides benefit to the feature extraction component of an object detection model. The deep CNNs traditionally utilised for feature extraction leverage transfer learning in the network training process. Common CNNs are reviewed in Section 2.6.4.1 and tested in the methodology to determine the preferred architecture for feature extraction of sewer faults. The deep CNN architectures available at this time of this research are generally high performing and complex, however selection is application specific, with the speed, accuracy and requirements of each CNN varying between applications (Talukdar et al., 2018).

2.5.3. Computer Vision Techniques

Computer vision is a form of artificial intelligence and a subset of machine learning that enables a computer system to interpret information from digital images, videos, and other visual data (IBM, n.d.). Computer vision techniques are integrated into many modern day technologies, including autonomous vehicles, facial recognition, medical imagery analysis, and other industrial applications such as manufacturing and agriculture (Marr, 2019). Within the field of computer vision, there are several different technologies, separated by differing features and applications. Common types of machine learning include image classification, object detection, image segmentation, edge detection and others (arm, n.d.); techniques relevant to this research are further discussed below.

2.5.3.1. Image Classification

Image classification is a form of computer vision that is defined as the task of identifying what an image represents (TensorFlow, 2022). Commonly image classifiers are CNN architectures, that can be developed and trained by various conventional processes such as supervised learning, unsupervised learning, and transfer learning. The image classification domain is one of the most widely researched in the computer vision field, with various competitions recurring annually considered to be large drivers for the rapid progression of the field (Pathak et al., 2018).

Image classification functions by interpreting various attributes of the data, and classifying the image based on recognised patterns and features. Whilst this ability of image classification has many applications, image classification does not have the functionality to localise the feature or area of interest within the image. Whilst the inability to localise objects within images may restrict image classification, it has consequentially become the foundation of developments in object detection technology (Pathak et al., 2018). Many deep learning CNNs that have emerged in the image classification field are adapted as backbone architecture for object detection models, to undertake the feature extraction process.

2.5.3.2. Object Detection

Object detection builds on the functionality of image classification technology, with the addition of feature localisation. An object detection model classifies the instance within its respective class and estimates the location of the feature within the image through an annotation output known as a bounding box or region of interest (ROI) (Pathak et al., 2018). Object detection technology is becoming increasingly prominent in society due to the extensive applications it has. Object detection is the primary technology utilise in some computer vision applications outlined earlier in this report, including autonomous vehicles and license plate recognition. Figure 2.5 provides a visual comparison between object detection and image classification techniques:

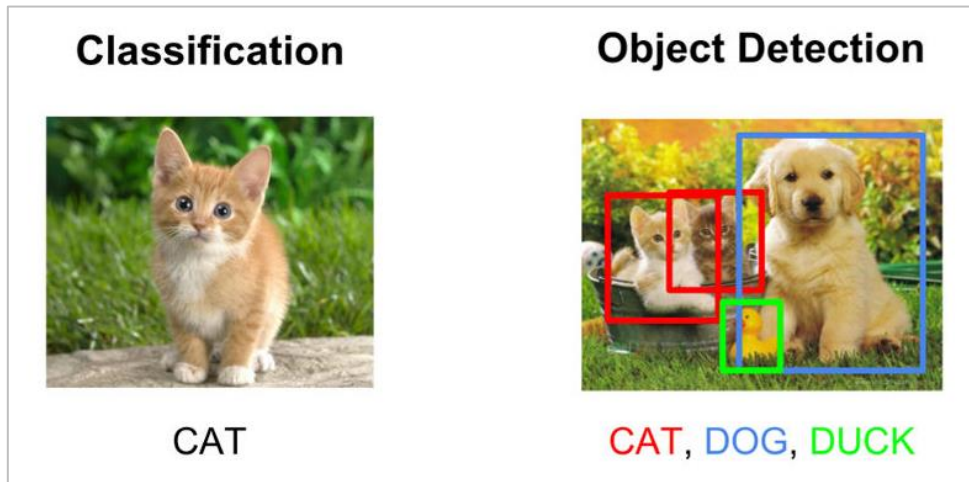


Figure 2.5: Comparison of object detection and image classification (Hulstaert, 2018)

Deep CNNs are extensively used founding structures for object detection models, containing many complex layers capable of completing the feature extraction process. These CNNs are often adapted from an image classification setting, by substituting the classification layers for an object detection framework such as ‘You Only Look Once’ (YOLO), ‘Single Shot Multibox Detector’ (SSD) or ‘Fast-CNN’. Conventional layers within an object detection model are outlined below in Table 2.2, and displayed diagrammatically in Figure 2.6:

Layer	Description
Input Layer	Contains the image data to be and is the input of the CNN
Convolutional Layer	The convolutional layer is the primary building block of a CNN, containing filters which are developed through the training process. This layer extracts features from the data. There are various types of convolutional layers, including high and low level.
Batch Normalisation Layer	Batch normalisation is not utilised in all CNNs, however is designed to normalising layer outputs. Specifically, batch normalisation scales the output of the layer, explicitly normalising the activations of each input variable per mini-batch (Singla, 2020).
Activation layer/function (ReLU)	An activation function is the last component of a convolutional layer, which produces a feature map (Pokhrel, 2019). Activation layers are not technically layers as no weights or parameters are learned, hence are frequently omitted from network diagrams (Rosebrock, 2021).
Pooling Layer	Pooling layers are designed to reduce the size of the network by sampling the data of previous layers, as to preserve useful information and remove redundant data.
Fully Connected Layer	Fully connected layers make up the final network layers; they combine information from former layers to facilitate classification.
Output Layer	Results of all previous layers is passed through the classifier in the form of a category or probability, resulting in an output result.

Table 2.2: Description of generic layers within a CNN (Zhang et al., 2019)

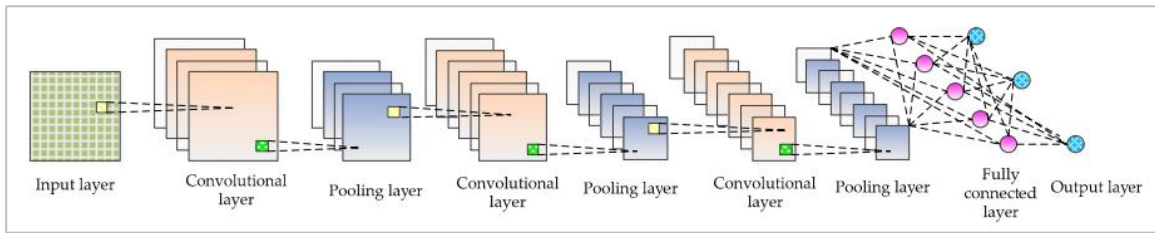


Figure 2.6: Basic CNN Structure by Zhang et al. (2019)

Figure 2.6 represents a conventional CNN structure that may be appropriate for a task such as image classification, however it is not representative of the architecture of an object detection model. For the CNN to be adapted to an object detection model, the fully connected layer and output layer are removed, and a convolution network is employed to detect the ROI and associated confidence level; this component of the model is typically labelled the ‘detection head’. An example of this is the standard YOLOv2 architecture presented below in Figure 2.7; YOLOv2 was released in December 2016 by Redmon and Farhadi (2017) and utilised the Darknet-19 CNN as a classification model. The YOLOv2 object detection model is reviewed in detail in Section 2.6.4.2 of this literature review and forms the basis of the detection model developed by this research.

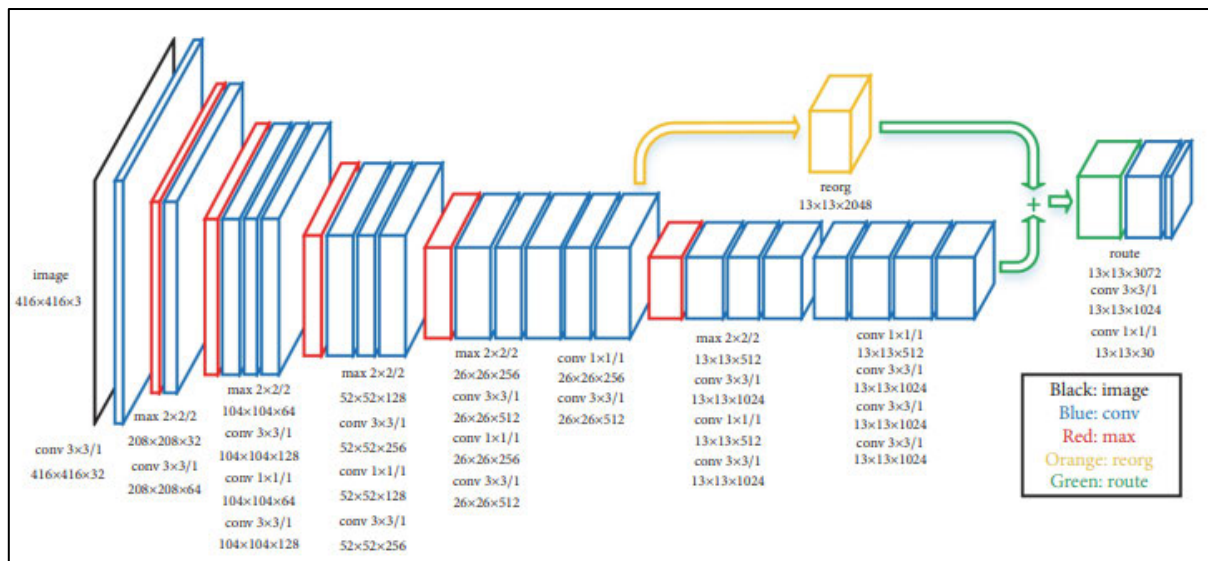


Figure 2.7: Network architecture of YOLOv2, adapted by Liu et al. (2018)

2.5.3.3. Semantic Segmentation and Instance Segmentation

Semantic segmentation is a form of deep learning algorithm that classifies every pixel within an image by a label or category (Lamba, 2019). The process allows an image to be partitioned into semantically meaningful parts, with categories represented by varying colour blocks (Long et al., 2019). That precise functionality of semantic segmentation means that the output is more precise than conventional object detection in many applications. In the application of

autonomous driving vehicles, object detection is capable of detecting and locating required information in the field of view; semantic segmentation allows precise interpretation of the geometrical space around objects (Xu et al., 2018). Instance segmentation is an extension of semantic segmentation, where each occurrence of a feature within an image is differentiated; a comparison between object detection, semantic segmentation and instance segmentation is shown in Figure 2.8:

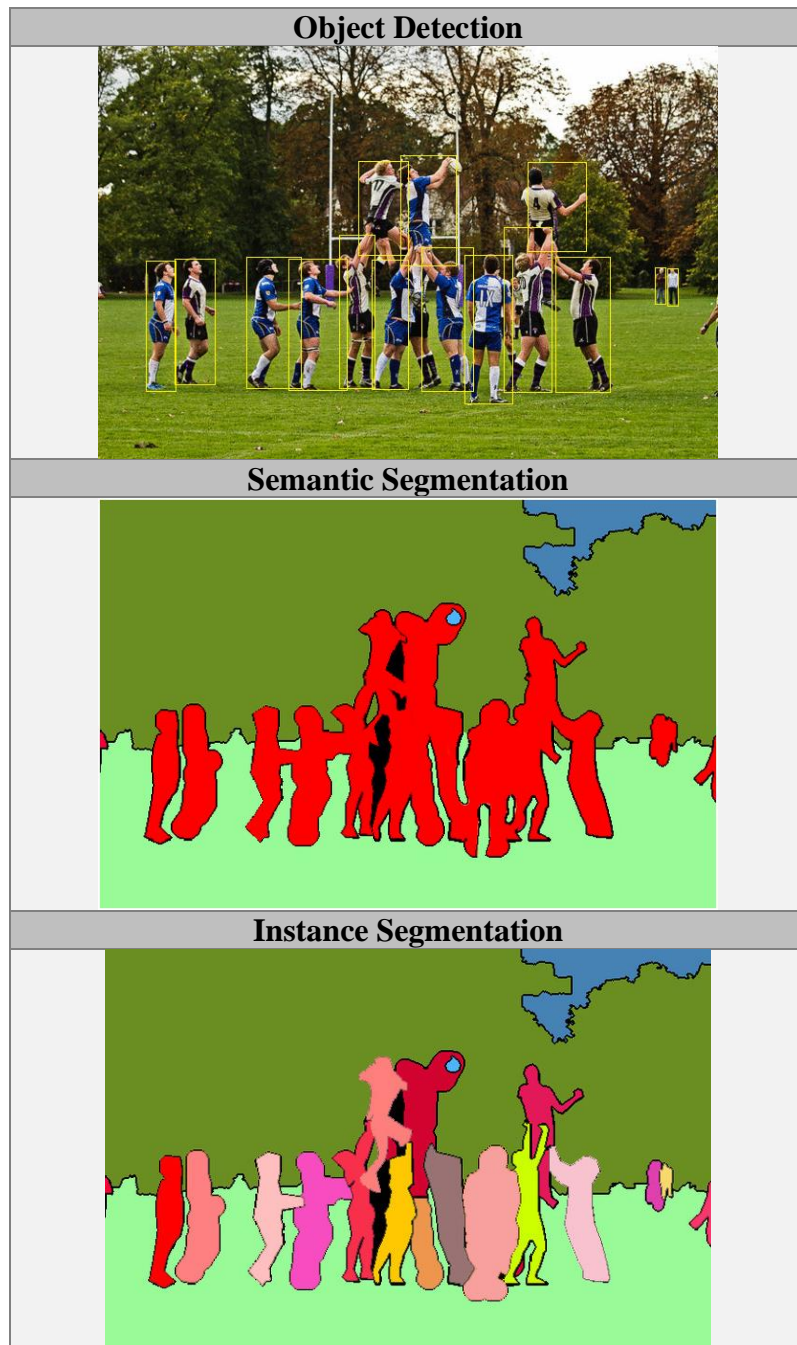


Figure 2.8: Comparison of object detection, semantic segmentation and instance segmentation, adapted from ByteBridge (2021)

The pixel level nature of semantic segmentation results in instances where the technology makes an incorrect categorisation due to the image data. Blurred images, semi-transparent objects, vignetting, and camouflaging are factors that may inhibit the performance of semantic segmentation detection (Thoma, 2016). Wang & Cheng (2019) investigated the application of semantic segmentation to sewer pipe defects and noted that application of the technique was very limited to this application. The ‘DilaSeg’ model developed by Wang & Cheng (2019) demonstrated effective detection with high levels of accuracy, however detection speed was on average less than four frames per second.

2.5.4. Neural Network Models

This section of the literature review builds upon the deep learning technologies discussed, to provide information and context to the methodology implemented to achieve the research aims and objectives. Various existing CNNs and object detection models will be reviewed to determine suitability for application to sewer pipeline fault detection, the preferred of which will be further analysed in the methodology section of this dissertation.

2.5.4.1. Feature Extraction CNNs

CNNs are an integral component of conventional computer vision models as discussed throughout Section 2.6.3. Forming the backbone of many detection models, performance of popular CNNs is readily available through extensive research papers and other resources available. The intent of this literature review section is to summarise the performance of prominent CNNs in similar applications such as concrete crack detection; this will inform analysis of various CNN application to sewer pipeline fault detection in the methodology.

A survey paper was published in the ‘Journal of Big Data’ by Alzubaidi et al. (2021) providing results of a comprehensive review of current deep learning technologies, including many CNN architectures. Additionally, Hamishebahar et al. (2022) published a journal article at the beginning of 2022, reviewing current deep-learning based crack detection methods which is highly relevant to this research.

Table 2.3 below provides information relating to the numerous high-performing CNNs currently implemented in various applications of deep learning; the networks are then discussed in further detail.

Feature Extraction Network	Number of Layers
AlexNet	8 Layers
VGG-16	16 Layers
VGG-19	19 Layers
GoogLeNet	22 Layers
SqueezeNet	18 Layers
InceptionV3	48 Layers
Res-Net 18	18 Layers
Res-Net 50	50 Layers
Res-Net 101	101 Layers
Darknet-19	19 Layers
Darknet-53	53 Layers
Inception-ResNet-V2	164 Layers
Xception	71 Layers

Table 2.3: Summary of common CNNs (MathWorks, n.d.)

AlexNet

The AlexNet CNN is a well-known deep CNN that was first proposed in 2012 (Hamishebahar et al., 2022). The AlexNet CNN is eight layers deep, including five convolutional layers and three fully connected layers; ReLU is utilised as an activation function to enhance the rate of convergence (Hamishebahar et al., 2022). AlexNet has been applied in many instances within the civil engineering industry; in the review by Hamishebahar et al. (2022), AlexNet accounted for approximately 30% of CNNs used for image classification. In the research presented by Moradi (2020), AlexNet was compared with VGGNet, ResNet and GoogLeNet; whilst the within 3-5% for each metric, AlexNet was the least favourable feature extraction network (SSD framework) when measured for classification loss, localisation loss and total loss.

VGG-16 & VGG-19

Visual geometry group (VGG) was developed two years after AlexNet, demonstrating increased depth over the prior CNN (Alzubaidi et al., 2021). The VGG CNN has two common architectures, VGG-16 and VGG-19, which are distinguished by their respective network depth. VGG extended the use of small filter sizes from ZefNet, further demonstrating the efficiency of this architecture format. Whilst VGG is generally well respected, it is noted to

have a high computational cost due to the approximate 140 million parameters within the network (Alzubaidi et al., 2021). Moradi (2020) utilised VGG-16 as a backbone for the SSD object detection framework in an early experiment, achieving an mAP of 79.6% on four defect classes.

GoogLeNet (Inception-V1)

GoogLeNet, also known as Inception-V1, emerged in 2014 and won the ILSVRC competition achieving an error rate of 6.67% (Szegedy et al., 2015). GoogLeNet introduced ‘Inception modules’ which form the fundamental structure of the network; these inception modules are stacked on each other, with max-pooling layers introduced to reduce grid resolution as required (Szegedy et al., 2015). This approach saw GoogLeNet achieve high-level accuracy, whilst maintain lower computational cost (Alzubaidi et al., 2021). GoogLeNet has been implemented extensively in object detection applications and inspired the original YOLO framework that was later released (Redmon et al., 2016). In Moradi’s (2020) research into the application of object detection to sewer fault detection, GoogLeNet was found to be the highest performing CNN when compared to AlexNet, VGG-16 and ResNet.

SqueezeNet

Released in 2016, the SqueezeNet CNN sought to achieve equivalent levels of accuracy to prominent CNNs whilst reducing the number of network parameters (Iandola et al., 2016). Through the significant reduction in network parameters, SqueezeNet offers more efficient distributed training and improved deployment prospects. When compared with AlexNet, SqueezeNet boasts a reduction in network parameters by a factor of 50, whilst maintaining or exceeding the top-1 and top-5 accuracy of AlexNet (Iandola et al., 2016). Ullah et al. (2021) undertook a comparative study of asphalt crack detection using SqueezeNet, AlexNet and ResNet18; SqueezeNet performed least favourably, with very poor precision and recall metrics achieved for most crack types. SqueezeNet performance was more favourable in object detection (YOLOv2 structure) of concrete cracking, outperforming AlexNet, VGG and some ResNet variants (Ullah et al., 2021).

Inception-V3

Inception-V3 builds upon GoogLeNet (Inception-V1) and Inception-V2 with the aim to maintain the generalisation ability of a deeper network, while reducing the inherent computational cost typically proportionate to network depth (Alzubaidi et al., 2021). Inception-V3 is 48 layers deep and utilises asymmetric small-size filters in lieu of large size filters, and

employs a 1 x 1 convolution prior to the large-size filters within the network (Szegedy et al., 2016). Inception-V3 demonstrated improvements against the top published CNN results for the ILSVRC classification benchmark in its release paper, reducing the top-5 and top-1 error by 25% and 14% respectively (Szegedy et al., 2016).

ResNet

Residual Network (ResNet) was the winner of the ILSVRC in 2015, with numerous variations of the CNN available with network depths between 18 and 1202 layers (Alzubaidi et al., 2021). Common versions of the network are the 18-, 34-, 50-, 101- and 152-layer versions, detailed below in Figure 2.9; the ResNet-50 version comprises of 49 convolutional layers and one fully connected layer (He et al., 2016).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.9: Architectures for common versions of Res-Net CNN (He et al., 2016)

The specific variant presented at ILSVRC 2015 was ResNet-152; despite having a depth 8x VGG, the computational complexity remains lower. ResNet-18 was the preferred CNN for concrete crack detection using YOLOv2 framework in research by Teng et al. (2021); competing CNNs included AlexNet, GoogLeNet and VGG-16.

Darknet-19 & Darknet-53

Darknet-19 is a CNN that was introduced as the backbone of the YOLOv2 (YOLO9000) object detection model. The Darknet-19 CNN architecture is noted as similar to VGG models and comprises of 19 convolutional layers and five maxpooling layers (Redmon & Farhadi, 2017). Darknet-19 supported the YOLOv2 model to outperform competing object detection in both mean average precision (mAP) and detection speed on the VOC 2007 dataset (Redmon & Farhadi, 2017).

The Darknet-53 CNN was introduced in the release of YOLOv3, as an update to the Darknet-19 predecessor, described as a hybrid approach with emerging residual network technology (Redmon & Farhadi, 2018). In the release paper, Darknet-53 claimed to outperform ResNet-101 and ResNet-152 in accuracy and FPS metrics (Redmon & Farhadi, 2018); whilst noted as a high performing CNN, performance against other CNNs varies subject to application. In the comparison study completed by (Alzahrani et al., 2021), Darknet-53 was ranked ten of nineteen CNNs.

Inception-Resnet-v2

Inception-Resnet-v2 was introduced by Szegedy et al. (2016), building upon prior released of the Inception network such as InceptionV3. Inception-ResNet-v2 is similar to other members of the Inception CNN family, however, incorporates residual connections in lieu of the filter concatenation stage in the architecture (Szegedy et al., 2016). In research on the application of the YOLOv2 detector to crack detection in concrete, Inception-Resnet-v2 was tested as a feature extraction network; the CNN achieved top three results of those tested by Teng et al. (2021).

Xception

The Xception CNN utilises extreme inception architecture, achieving extra learning efficiency and better performance when compared to other Inception networks, without minimising the number of parameters. The Xception network is 71 layers deep, reliant on depthwise separable convolution layers; these layers were defined by Chollet (2017) as ‘similar properties to inception modules, yet are as easy to use as regular convolution layers’. In the research paper presented by Xu et al. (2021), Xception outperformed InceptionV3, ResNet50 and VGG19 in concrete crack detection, utilising a dataset of approximately 98,000 images.

2.5.4.2. Object Detection Models

Object detection models are an extension of conventional CNNs, having the ability to detect, categorise and localise features within a target image, as detailed in Section 2.6.3.2. This research intends to evaluate the performance of current object detection models, in the application of sewer fault detection. Review of various object detection structures will inform the proposed analysis in the methodology section of this dissertation.

Research by Moradi (2020) on the application of deep learning to sewer fault detection, included comparison of YOLOv1 (Redmon et al. 2016), SSD (Liu et al. 2015), R-CNN (Girshick et al. 2014), Fast R-CNN (Girshick 2015), and Faster R-CNN (Ren et al. 2017). In

this research SSD was the most favourable detection framework, however real time detection frame rate was not achieved (Moradi, 2020). Improved versions of the YOLO framework are reviewed below, and later form the basis of the methodology of this project.

R-CNN, Fast R-CNN & Faster R-CNN

The R-CNN family of object detection models commenced with the release of R-CNN in 2015 by Girshick et al. (2014). The functionality of R-CNN is based on region proposals, which are taken from the input image, and then interpreted and classified through a CNN (Parthasarathy, 2017). Fast R-CNN sought to improved shortcomings of the previous version; changes resulted in the convolution operation only being required once per image, in lieu of the prior 2000 region proposals (Gandhi, 2018). Faster R-CNN saw this introduction of a region proposal network (RPN), which sought to resolve the bottleneck created by the prior region proposal computation (Ren et al., 2017). Whilst improvements in detection speed were made in the evolution to Faster R-CNN, the network is still unable to achieve real-time detection ability (*Redmon et al., 2017*).

YOLO (v1)

The ‘You Only Look Once’ (YOLO) object detection framework was presented by Redmon et al. (2016) and has since become a prominent object detection model. YOLO considers object detection as a single regression problem, differing from other models such as R-CNN; this means that YOLO goes directly from the input data to a ROI bounding box and associated confidence value (Redmon et al. 2016). A reduced version of the model known as Fast YOLO was also released, which contains 9 layers instead of 24. The YOLO network is several times faster than comparable detection models at its time of release, whilst still maintain comparable accuracy; a comparison of object detectors at the release of YOLO is adapted from Redmon et al. (2016) below:

Detection Model	mAP	FPS	Real-Time
YOLO	63.4	45	Yes
Fast-YOLO	52.7	<u>155</u>	Yes
YOLO (VGG-16)	66.4	21	No
Faster R-CNN (VGG-16)	<u>73.2</u>	7	No
Faster R-CNN (ZF)	62.1	18	No
Fast R-CNN	70	0.5	No

Table 2.4: Comparison of YOLO and Fast/Faster R-CNN on VOC 2007 dataset (Redmon et al., 2017)

SSD

Single Shot MultiBox Detector (SSD) was introduced in late 2016 by Liu et al. (2016). Similarly to YOLO, SSD negates the need for a region proposal component required by R-CNN and similar; for SSD, all computations are captured in a single network (Liu et al., 2016). In comparison to other detectors at its time of release, SSD300 was the only real-time detection framework to achieve AP greater than 70% (Liu et al., 2016). Research into the application of deep learning to sewer fault detection by Moradi (2020) found SSD to be the preferred detection framework over YOLO, Fast R-CNN, and Faster R-CNN.

YOLOv2 (YOLO9000)

YOLOv2 (also known as YOLO9000) sought to resolve the primary shortcomings of YOLO, namely localisation errors and low recall (Redmon & Farhadi, 2017). The release of YOLOv2 introduced a new CNN as the feature extraction known as Darknet-19; this CNN is discussed in Section 2.6.4.1 of this dissertation. The comparison included in the release paper by Redmon & Farhadi (2017) is presented below in Table 2.5:

Detection Model	mAP	FPS	Real-Time
YOLOv2 (544 x 544)	78.6	40	Yes
YOLOv2 (416 x 416)	76.8	67	Yes
YOLOv2 (288 x 288)	69.0	91	Yes
SSD300	74.3	46	Yes
YOLO	63.4	45	Yes
SSD500	76.8	19	No
YOLO (VGG-16)	66.4	21	No
Faster R-CNN (VGG-16)	73.2	7	No
Faster R-CNN (ZF)	62.1	18	No
Fast R-CNN	70	0.5	No

Table 2.5: Comparison of YOLOv2 other detectors on VOC 2007+2012 dataset (Redmon & Farhadi (2017))

Other YOLO Detectors

The YOLO detector has demonstrated significant progression in recent years, with several updated versions being released. YOLOv3 introduced the Darknet-53 backbone, which included residual network characteristics (refer Section 2.6.4.1); YOLOv3 showed an increase in AP of 9.8%, however a reduction in FPS of 50% on the COCO dataset (Reyes et al., 2019).

The YOLOv4 release in July 2022 saw improvements in AP and FPS of 10% and 12% respectively (Bochkovskiy et al., 2020). More recently, version 7 of the YOLO object detector framework, YOLOv7 (Wang et al., 2022) was released at the beginning of 2022. This detection framework “surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS” (Wang et al., 2022).

2.5.4.3. Network Training Hyperparameters

Training hyperparameters are described as network variables that influence how the network is trained; they are set prior to training and relate to how the network is optimised (Radhakrishnan, 2017). Optimising a neural network by tuning hyperparameters through iterative analysis is an important step in achieving an efficient and accurate model (Stewart, 2020); for example, a high learning rate may result in the model never achieving maximum accuracy. Table 2.6 provides description of common training hyperparameters that can be adjusted during the model training process; the definitions are adapted from the MathWorks (n.d.) online help centre:

Training Hyperparameter	Description
Optimiser	The optimiser setting selects the solver algorithm to be used, including ‘sgdm’, ‘rmsprop’ and ‘adam’.
Mini-Batch Options	
MaxEpochs	An epoch is a full pass of the training algorithm over the entirety of the training dataset; ‘MaxEpochs’ defines the maximum number of epochs the network training can complete before finishing.
MiniBatchSize	The mini-batch is a subset of the training dataset used to evaluate the loss function gradient and update the layer weights. The ‘MiniBatchSize’ describes the size of the mini-batch.
Shuffle	‘Shuffle’ allows the training data to be shuffled ‘once’ before training, ‘every-epoch’, or ‘never’. Shuffling the data every epoch is a common technique implemented to reduce overfitting.

Validation	
ValidationFrequency	Sets the number of iterations between evaluation of validation metrics.
ValidationPatience	Sets the number of times that the validation loss can be larger or equal to that of the previously smallest loss before network training stops. The value can be set to 'inf' to never stop the training due to the described occurrence.
OutputNetwork	Specifies whether the final network is based on the last iteration, or the iteration with the lowest validation loss.
Solver Options	
InitialLearnRate	Sets the initial learning rate for the network. Learning rate control the magnitude of weight adjustments to the network during training.
LearnRateSchedule	Defines if the learning rate remains constant or drops during training. 'Piecewise' will set the learning rate to drop at a set interval, by a set factor during training.
LearnRateDropPeriod	Describes the interval at which the learning rate will be reduced by the 'LearnRateDropFactor'.
LearnRateDropFactor	Defines the factor by which the learning rate is dropped at each drop period; the value is applied to the learning rate as a multiplier.
L2Regularisation	A regularisation term for the weights to the loss function, to assist in reducing overfitting; also known as 'weight decay'.
Momentum	Applicable to the 'sgdm' solver only, 'Momentum' is a contribution of the parameter update step of the previous iteration to the current iterations. The intent of the function is to build momentum as the gradient trends correctly.
GradientDecayFactor	Applicable to the 'adam' solver only, it defines the decay rate of gradient moving average.

SquaredGradientDecayFactor	Decay rate of the squared gradient moving average for the ‘adam’ and ‘rmsprop’ solvers.
ResetInputNormalisation	Sets the option to reset input layer normalisation.
BatchNormalisationStatistics	Sets the mode to evaluate the statistics in batch normalisation layers, specified as a running estimate given by update steps (‘moving’), or population statistics by passing through the training data once more ‘population’.
Hardware Options	
ExecutionEnvironment	Selects the hardware resource for training the network: ‘cpu’, ‘gpu’, ‘multi-gpu’, ‘parallel’ or ‘auto’ (selects gpu if available, otherwise cpu).

Table 2.6: Training options for object detection model training, adapted from MathWorks (n.d.)

It is noted that further training hyperparameters exist in addition to those described in Table 2.6 above, however excluded training hyperparameters are considered unimportant, as they not deemed to influence the investigations of this dissertation and have remained constant through all trials. Key hyperparameters outlined in the table above are detailed further below; ‘MiniBatchSize’ and ‘MaxEpochs’ will be investigated through iterative analysis in Section 4 of this dissertation:

Optimiser

The training optimiser, or gradient descent optimisation algorithm, is the algorithm that adjusts and optimises hyperparameters such as weights within the CNN to fit the training dataset (Ruder, 2016). There are many optimisers available, with SGD, SGDM, Adam and RMSProp being some of the most common.

SGD adjusts weights by subtracting the product of the gradient and the learning rate from its rates. Although simplistic in contrast to other solvers, SGD is supported by theoretical foundation, as well as extensive practical application (Park, 2021). SGDM is adds a momentum function to the SGD optimiser, where the weights are modified by the moving average of gradients; this intends to support convergence when the gradient trends in the desired direction (Park, 2021).

RMSProp is similar to SGDM; it adaptively adjusts each parameter based on the gradients (Park, 2021). It seeks to resolve the issue of vanishing or exploding gradients by using a moving average of squared gradients to normalise the gradient (Sanghvirajit, 2021).

The Adaptive Moment Estimation (Adam) optimiser is an increasingly popular algorithm that adaptively computes learning rates for each parameter. Adam is described as a combination of SGDM and RMSProp, as it maintains an exponentially decaying average of past gradients, whilst also storing an exponentially decaying average of past square gradients (Ruder, 2016).

MiniBatchSize

MiniBatchSize is the size of a subset of the training data population used to evaluate the gradient of a loss function and update the weights. Whilst noting inconsistency in initial research findings, Kandel & Castelli (2020) found that a lower MiniBatchSize would typically result in increased accuracy of the CNN, providing the initial learning rate was low enough. From this, the conclusion was drawn that the highest accuracy will generally be achieved from a lower MiniBatchSize and low learning rate, however good accuracy can still be achieved with a higher MiniBatchSize when coupled with a higher learning rate (Kandel & Castelli, 2020). Whilst the aforementioned research indicates smaller MiniBatchSize generally leads to increased accuracy, there are many contradicting examples such as research by Peng et al. (2018), and research by Bochkovskiy et al. (2020); considering this, it can be inferred that effect of MiniBatchSize varies between application and should be optimised for the selected task.

InitialLearnRate

InitialLearnRate is a hyperparameter which restricts how much change the model will be made to the model in response to the error rate each time the model weights are updated (Brownlee, 2020). Learning rate is considered to be one of the most critical hyperparameters in CNN training due to its influence on model performance; a learning rate that is too large may result in poor performance due to model effectively skipping the optimal solution, whereas a low learning rate may restrict the model from learning the target data effectively (Brownlee, 2020).

MaxEpochs

An epoch is a term used in machine learning to describe the number of passes the model will make through the entirety of the training dataset. If the training process does not extend for enough epochs, the model may not converge; however, too many epochs may result in the model overfitting, as it begins to learn information such as noise and background information

(Afaq & Rao, 2020). Notwithstanding, it is not possible to derive a universally ideal number of epochs, as it varies between models and datasets (Afaq & Rao, 2020).

2.5.4.4. Model Overfitting & Underfitting

Model ‘fitting or ‘fit’ describes how well a network is trained to a particular target application, and its ability to generalise new data. The terminology of ‘underfitting’ and ‘overfitting’ are commonly utilised to describe a networks training performance, both of which are undesirable attributes (Jabbar & Khan, 2014). Underfitting and overfitting phenomenon are detailed below and referred to in application in Chapter four of this dissertation.

Underfitting

Underfitting of a model refers to an inability to learn the relationship between input and output variables, resulting in high error rate for seen and unseen data (IBM Cloud Education, 2021). Underfitting is typically easy to identify, as the model generally performs poorly in all applications. Underfitting can be reduced by increased dataset size and improved data quality, increased training time, decreased regularisation and increased network complexity (IBM Cloud Education, 2021).

Overfitting

Overfitting describes when a model begins memorising regularity or noise contained within the training dataset, decreasing its performance on unseen data (Jabbar & Khan, 2022). Overfitting is a common issue amongst CNN models and is typically attributed to a limited training dataset size, significant similarities within the training data, or overtraining of the model (IBM Cloud Education, 2021). Overfitting can be detected in several ways, with the most common being inspection of training and validation loss information; if validation loss begins to increase whilst training loss continues to decrease, it can be inferred that the model is likely overfitting (Carremans, 2018). To reduce overfitting, the following techniques are recommended by Ruizendaal (2017) and Rice et al. (2020):

- Increase the dataset
- Implement data augmentation
- Early stopping of training
- Add regularisation
- Select a suitable architecture, avoiding overly complex (deep) networks

2.5.4.5. Evaluation Metrics

Evaluation metrics refer to the quantitative means by which a deep learning network, such as an image classification or object detection model, can be scored based on its performance and ability. ‘Average Precision (AP)’ and ‘mean Average Precision (mAP)’ are frequently utilised to assess and rank the performance of object detection models, such as those outlined in Section 2.6.4.2 (Koech, 2022). In addition to AP and mAP, ‘F1-Score’ is another common evaluation metric utilised in machine learning; similarly to AP and mAP metrics, the calculation of a model’s F1-Score is completed using ‘precision’ and ‘recall’ data (LT, 2022). These common evaluation metrics and associated variables are discussed below.

True Positive

A true positive (TP) is defined as the correct detection of a ground-truth bounding box (Padilla et al., 2020); this occurs when the model successfully detects a fault that exists.

False Positive

A false positive (FP) is defined as an incorrect detection of an object that does not exist, or a detection that is misplaced from the ground truth data beyond the acceptable IOU (Padilla et al., 2020).

False Negative

A false negative (FN) is an undetected ground-truth within the data that is missed by the detection model (LT, 2020). In the application to sewer fault detection, false negative occurrences are highly undesirable, as it may mean that a critical fault has been missed by the model.

Intersection Over Union

An intersection over union (IOU) is a metric utilised to determine whether a detection is true or false, based on how closely the placed bounding box correlates to the ROI of the ground-truth (Padilla et al., 2020). Typically, an acceptable IOU value will be assigned to define the requirement for a TP to occur.

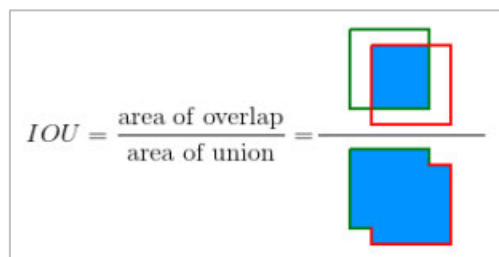

$$IOU = \frac{\text{area of overlap}}{\text{area of union}}$$

Figure 2.10: IOU calculation

Precision

In machine learning and its subfields, precision is defined as “the ability of a model to identify only relevant objects” (Padilla et al., 2020). Precision is the percentage of positive predictions that are correct, it does not consider positive ground truths that may have been missed within the dataset; this means that precision measures the extent of error caused by FPs (LT, 2022). The formula for calculation of precision is (Padilla et al., 2020):

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

Recall

Recall is typically considered the counterpart to the precision metric and is defined by LT (2022) as the model’s ability to find all relevant cases. In contrast to precision, the recall metric measures the extent of error caused by false negatives (LT, 2022) The formula for calculation of recall is (Padilla et al., 2020):

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

F1-Score

The F1-Score is a common metric that combines precision and recall; it is defined as weight average or ‘harmonic mean’ between the two variables (LT, 2020). F1-Score is very prominent in the assessment and ranking of CNNs; the formula for calculation F1-Score is provided below (LT, 2020):

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Average Precision

Average Precision (AP) is an evaluation metric most commonly utilised in the performance assessment of object detection models. AP is described as the area under a precision-recall curve, where precision is plotted against recall values between 0 and 1 (Hui, 2019). As defined by (Padilla et al., 2020), a detector is considered good if it maintains high precision whilst the recall increases; this will result in a high AUC and thus, high AP. An example of a precision recall curve is provided below in Figure 2.11:

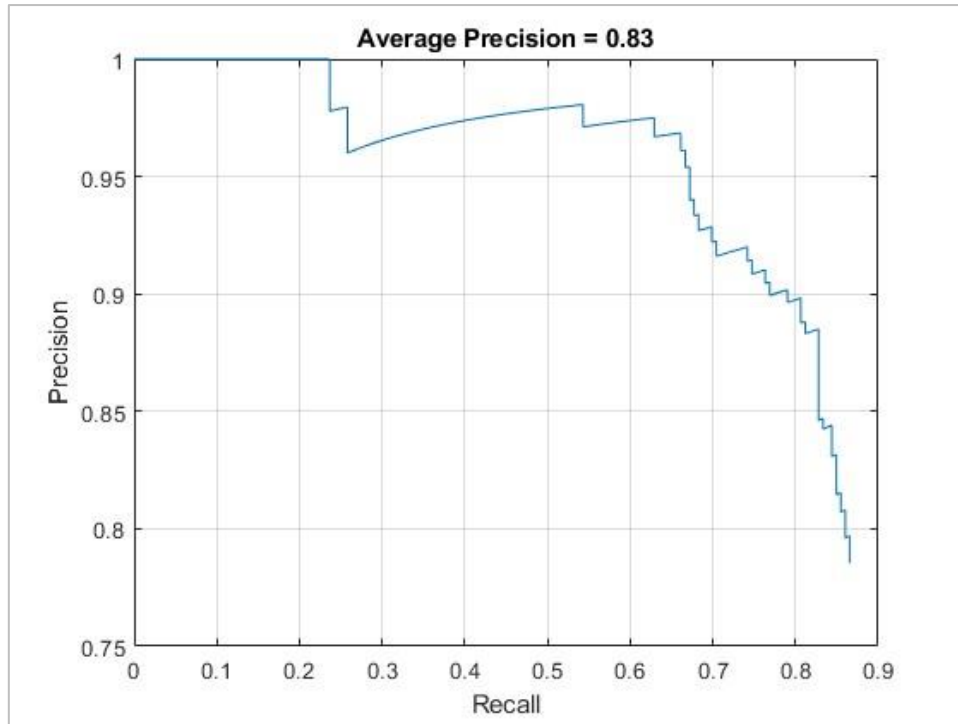


Figure 2.11: Precision-recall curve for YOLOv2 detector

The formula for AP is defined by Gad (2021):

$$AP = \sum_{k=0}^{k=n-1} [Recall(k) - Recall(k + 1)] \times Precision(k)$$

Where,

$$Recalls(n) = 0$$

$$Precision(n) = 1$$

$$n = \text{Number of thresholds}$$

Mean Average Precision

Mean Average Precision (mAP) is calculated as the average of AP across all classes the model must detect. (Hui, 2019). The formula for calculation of mAP is (Padilla et al., 2020):

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

Where,

$$AP_i = \text{AP of the 'i'th class}$$

$$N = \text{number of classes}$$

2.5.5. Application of Deep Learning

Deep learning-based automation is considered by many to be an emerging technology; application to various civil engineering elements has quickly become a prominent area of research. High levels of success have been found in applying deep learning networks to recognition of faults in road pavements, with accuracy and precision values found to exceed of 90% in networks of basic construction (Pauly et al. 2017). The investigation completed by Pauly et al. (2017) involved the collection of 500 RGB images, which were then sampled into smaller tiles, creating two image subsets totalling 240,000 images for analysis. The two subsets each had testing patches totalling 60,000; these were pre-classified as cracks and non-cracks, which were then used to teach the deep learning algorithm. As a network of this size is considered to be relatively ‘shallow’, the results were found to be of a satisfactory standard. Pauly et al. (2017) concludes that “an increase in the depth of the deep networks leads to better performances in terms of accuracy and recall”. Similarly, Mandal et al. (2018) completed an investigation on the application of deep learning networks to road fault detection, however this research considered several fault types. The investigation by Mandal et al. (2018) considered a total of eight fault types, utilising an image dataset of 9,053 raw images. The results obtained demonstrated values between 65% and 80% for precision, accuracy, and recall; whilst the accuracy is notably less than the previous mentioned study by Pauly et al. (2017), it is considered to be of increased complexity and may not have had the required dataset size to produce more accurate findings. The authors concluded that investigation was widely valuable and would likely benefit significantly from an increased dataset. The research completed by Mandal et al. (2018) is recognised as highly valuable to the investigation proposed by this paper, as the variety of faults is similar in nature to the requirements of deep learning application to sewer fault detection. Sewer fault detection has a variety of potential faults including spalling, cracking, collapses, voids, root ingress and water infiltration amongst others (Moradi 2020). Due to the appearance of sewer fault types being largely different in appearance to one another, a higher overall success rate is anticipated in contrast to classification of road faults, as differing road fault classifications are often similar in appearance. A review of computer vision techniques applied to automated sewer fault detection was completed by Moradi (2019); it was found that deep learning technologies accounted for only nine percent of recent research into automated sewer fault detection, as shown below in Figure 2.12:

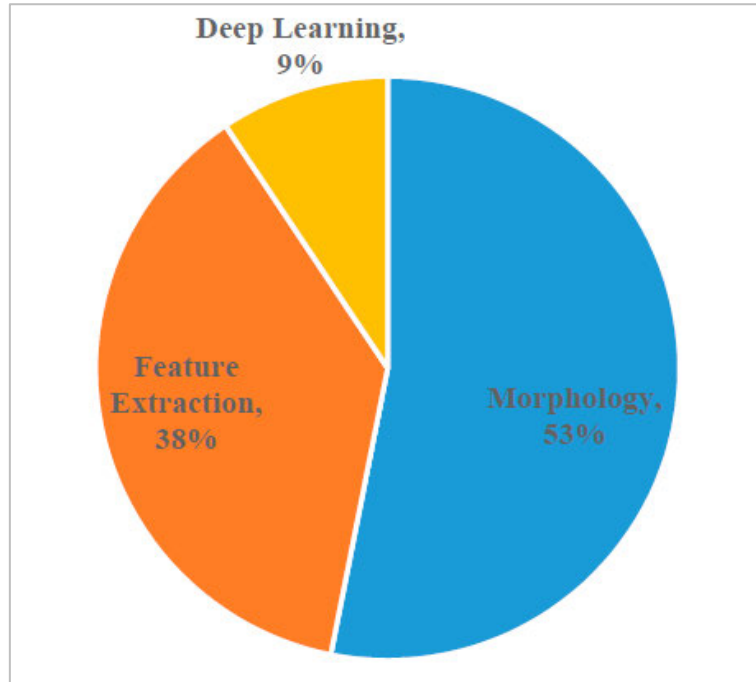


Figure 2.12: Review of computer vision technologies to automated sewer pipeline fault detection by Moradi (2019)

2.5.5.1. Application of Object Detection to Sewer Pipeline Faults

Cheng and Wang (2018) investigated the application of deep learning techniques to fault detection in sewer pipeline CCTV images. The research reviewed the use of deep learning neural networks in similar applications, considering both image classification and object detection techniques; Cheng and Wang (2018) found value in the application of deep learning techniques across various fields of civil infrastructure. Based on the literature review completed, Cheng and Wang (2018) applied the ‘faster R-CNN’ neural network, noting that it “demonstrated high precision and recall value and achieved the highest mean average precision” in other applications. The research trialled various data sizes, network types and parameters, finding all factors to influence the defect detection models performance.

Network	Dataset	mAP	AP			
			Root	Crack	Infiltration	Deposit
Original ZF	A (1000 images)	0.680	0.649	0.546	0.727	0.799
	B (2000 images)	0.763	0.735	0.794	0.759	0.765
	C (3000 images)	0.798	0.723	0.789	0.879	0.802

Figure 2.13: mAP and AP of the model using different dataset sizes Cheng and Wang (2018)

The results above indicate increased performance of the network when the image dataset is increased. Whilst the average precision (AP) value for detection of root ingress defects was 72.3%, it is recognised that all other categories demonstrated an upwards trend with respect to the dataset size. Considering this, the research completed by Cheng and Wang (2018) is highly relevant to this research and demonstrates the feasibility and value the application to deep learning to industry presents.

Moradi (2020) investigated the application of object detection to sewer pipeline faults with the intent to develop a suitable framework capable to real-world effectiveness. Research by Moradi (2020) involved the comparison of four CNNs, and five object detection frameworks; the object detection models included the R-CNN (Fast and Faster), SSD and YOLOv1. Experiments completed indicated that the SSD detector framework coupled with the GoogLeNet CNN backbone was the most preferred model, achieving an AP of 76.3% for crack defects, and an overall mAP of 81.3% for all classes. A comparison of the detection frameworks within the research by Moradi (2020) is provided below in Table 2.7:

Framework	Average Precision (AP)				mAP
	Crack	Deposit	Infiltration	Joint Displacement	
R-CNN	68.1	71.8	43.5	69.8	63.8
Fast R-CNN	77.0	78.4	59.6	82.6	74.4
Faster R-CNN	84.3	82.0	67.8	88.6	80.7
YOLO	77.4	77.0	43.3	85.3	70.8
SSD	76.3	88.2	74.9	85.7	81.3

Table 2.7: Comparative results for different object detection frameworks, adapted from Moradi (2020).

Whilst research by Moradi (2020) was generally successful and provided reinforcement of the on the validity of this application, real-time object detection capability was not achieved by the proposed framework. Further, the object detection frameworks included in the comparison are not representative of more efficient frameworks available, such as improved versions of YOLO.

2.6. Data Processing & Sampling

AI is demonstrated as a highly intelligent technology, capable of analysis a diverse range of data; however, this does not negate the value data processing techniques may provide to an applied scenario. To train a network and make predications on new input data, all data must be consistent with the network input size (MathWorks, 2022). Additionally, certain image

adjustment techniques such as contrast and brightness may improve the definition of the feature within the image, improving the accuracy of the network. Image pre-processing can be completed in isolation to the network, or it can be an integrated process. MathWorks (2022) notes the following with respect to the MATLAB software:

- “Commonly, pre-processing occurs as a separate step that you complete before preparing the data to be fed to the network. The advantage of this approach is that the pre-processing overhead is only required once, then the pre-processed images are readily available as a starting place for all future trials of training a network” (MathWorks, 2022).
- “Preprocessing can be applied during training; the transformed images are not stored in memory. This approach is convenient to avoid writing a second copy of training data to disk if your preprocessing operations are not computationally expensive and do not noticeably impact the speed of training the network” (MathWorks, 2022).

Data processing requirements are subject to vary between application and associated target categories. In the context of this research, data processing techniques such as brightness and contrast will be investigated if the standard parameters impact success of the neural network. This is discussed further in the methodology of this dissertation.

2.6.1. Data Augmentation

Data augmentation is a common technique utilised in machine learning to improve network accuracy and robustness by increasing diversity of the training dataset through random transformation on the original data (MathWorks, n.d.). Data augmentation is beneficial to detector performance as it introduces data that differs from the raw dataset, increasing detector robustness and its performance in detecting unseen data; further, data augmentation techniques assist in reducing overfitting through the increased data population and variability (Shorten & Khoshgoftaar, 2019). Broadly, data augmentation is defined as techniques to increase the amount of data by generating additional examples from existing training data (Dickson, 2021); it includes methods such as random horizontal flipping, scaling of images and image colour adjustments (i.e. contrast, hue etc.), amongst others. When data augmentation is implemented to improve model performance, it is imperative that the transformation processed is performed on the image datastore and the label datastore; this ensures that ground truths within the data remain accurate and do not negatively impact training of the model. Common data augmentation techniques relative to this research are defined below and shown in Figure 2.14:

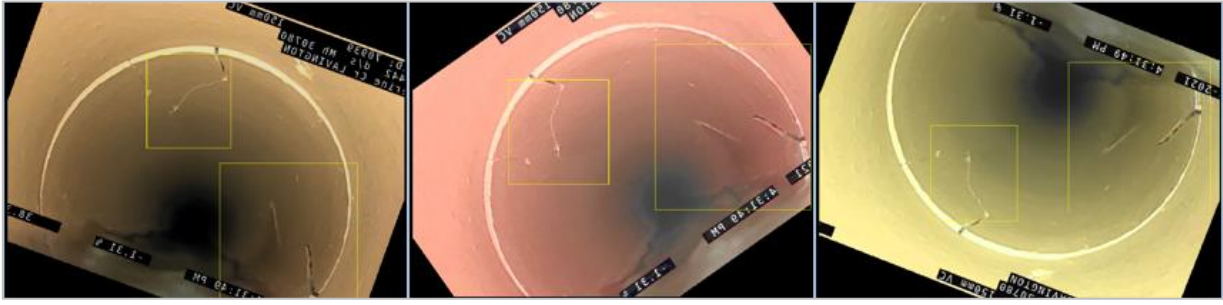


Figure 2.14: Randomised Data Augmentation Examples

Image Flipping

Image flipping is a common data augmentation technique that involves mirroring (flipping) the image over the horizontal or vertical axis (Sonogashira et al., 2020). Application of conventional image flipping techniques increases the dataset by a factor of three.

Rotation

Transformation by rotation is another common form of data augmentation where the positioning of data within the image is modified. This technique involves rotating the image from its native orientation by a factor between 1 and 359 degrees; commonly, 90-, 180- and 270-degree rotation is applied (Shorten & Khoshgoftaar, 2019).

Translation

Translation is a third geometric transformation commonly utilised to augment image datasets. Translation refers to shifting the image in a direction (up, down, left, right or a combination), cropping the extents and filling the space with a constant colour such as black or white (Shorten & Khoshgoftaar, 2019).

Colour Transformations

Augmentation by colour transformations relates to various common image adjustments such as contrast, brightness, hue, saturation, blur and noise, amongst others (Han et al., 2022). Colour transformations not only assist in creating more data, but can also be applied generally as a pre-processing technique to improve disability of features within the raw data (Shorten & Khoshgoftaar, 2019).

Jittering

Jittering is a form of colour transformation, where variable specific techniques mentioned above are applied on a random basis (Shorten & Khoshgoftaar, 2019). An example of this

might include contrast adjustment, increased brightness, and increased blue hue applied to the raw image; the next iteration would typically then differ from this.

2.6.2. Data Annotation Techniques

Data annotation describes the labelling process undertaken to facilitate supervised machine learning (Pokhrel, 2020). Conventional data annotation involves the user cycling through individual images or video frames to label features within the raw data, by utilising a bounding box or similar appropriate ground-truth dependent on the proposed training model (Pokhrel, 2020). This process is tedious and time consuming (Adhikari & Huttunen, 2021), however quality data annotation is essential in achieving an accurate and reliable model (Pokhrel, 2020).

To reduce the extensive time resourcing required by conventional data annotation, various automation tools have been created to improve efficiency of the annotation process. Adhikari and Huttunen (2021) developed a semi-automated process that leverages object detection technology to predict bounding boxes in raw data, which are confirmed or corrected manually, further improving the detection model. This research saw a reduction of up to 75% of manual annotation required for the dataset tested (Adhikari & Huttunen, 2021). Similarly, earlier research by Schreiner et al. (2006) involved generation of a suggested annotation, which would be verified by the operator; a one-minute video could be annotated in approximately thirty-five seconds using this method (Schreiner et al., 2006).

MATLAB Video Labeller App

The Video Labeller App allows the user to annotate raw video data by marking a region of interest (ROI) using various labelling tools; an example of the annotation process is provided below in Figure 2.15. The software tool facilitates an output in the ‘MAT-file’ format; this is known as a ‘ground truth’ and can be used as a direct input into various algorithms such as image classifiers, segmentation networks and in the instance of this research, object detection networks (MathWorks, n.d.).

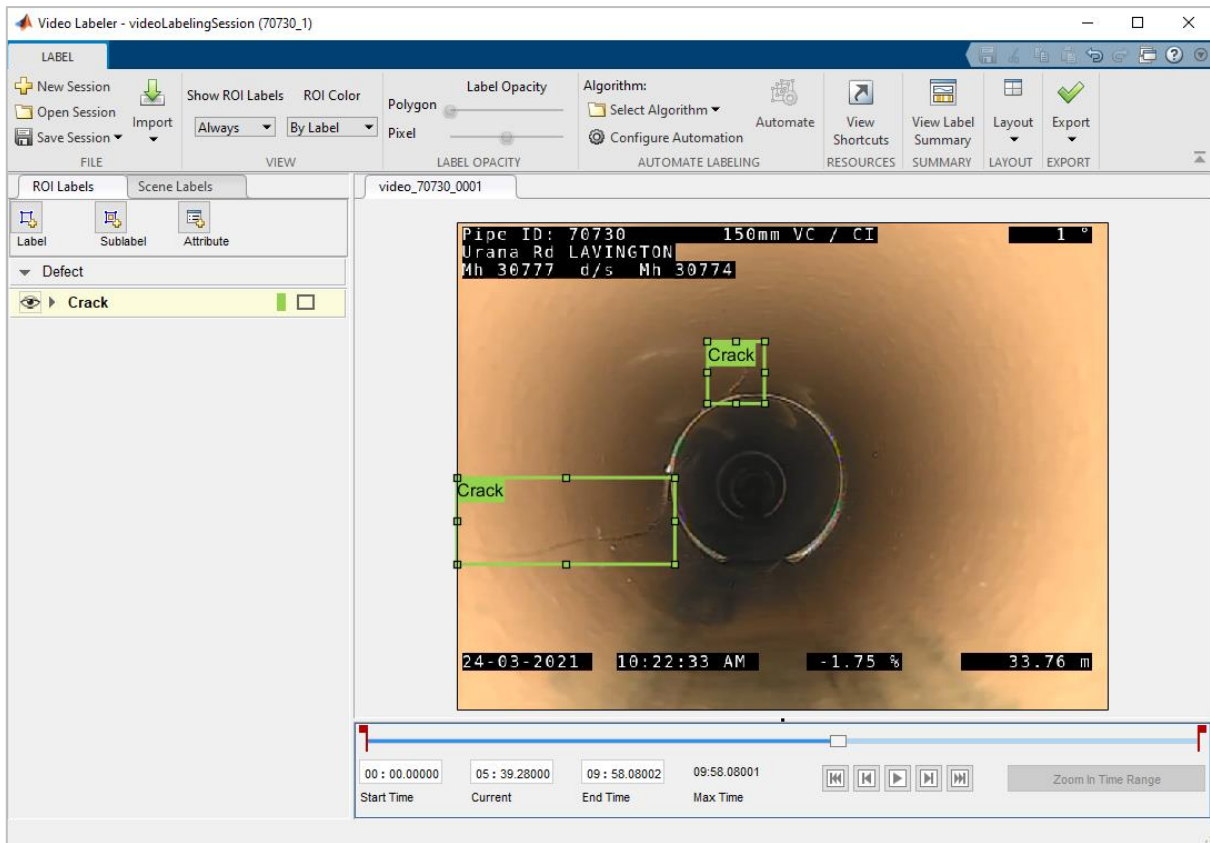


Figure 2.15: Video Labeller App Annotation Example (MathWorks, n.d.).

Labelling tools within the Video Labeller App feature a manual method and an automation algorithm; the manual method requires the user to annotate the ROI(s) within individual video frames, where the automation algorithm tracks an initial ROI through proceeding video frames. There are multiple annotation algorithms included in the Video Labeller App, including custom options; the available algorithms are summarised in Table 3.1:

Labelling Automation Algorithm	Description
ACF People Detector	A pretrained object detection algorithm designed to detect people using aggregate channel features (ACF).
Point Tracker	Tracks a ROI(s) using the Kanade-Lucas-Tomasi (KLT) feature-tracking algorithm; it works particularly well for tracking objects that do not change shape and for those that exhibit visual texture.
Temporal Interpolator	Estimates ROIs in intermediate frames using interpolation of ROIs in key frames.

Custom Algorithm	Custom algorithm allows the user to adjust automation parameters to improve performance of automated data annotation.
------------------	---

Table 2.8: Video Labeller App Automated Labelling Algorithms (MathWorks, n.d.).

2.6.3. Training, Validation and Test Datasets

The training and evaluation process of deep learning object detection models commonly requires the total dataset to be separated into three distinct samples, namely the training, validation, and test datasets (Shah, 2017). The ratio by which the population dataset is separated can influence performance of the detector model; the three samples and respective influence on model performance are detailed in the following paragraphs.

Training Dataset

The training dataset describes the sample of data that is used to train the model; the model sees the complete data and attempts to learn from this data (Shah, 2017). In most cases, the size and diversity of the training dataset is directly related to performance of the model; if limited data and/or data with a high degree of similarities is used for training, the model may experience overfitting. Model overfitting describes the occurrence where the model begins to memorise the training dataset, rather than learning the key attributes of the data (Boesch, n.d.). A model that demonstrates overfitting will perform poorly when new data is tested.

Validation Dataset

The validation dataset is an additional data sample utilised in the model training process. The purpose of the validation dataset is to evaluate the model fit without bias during the training process, allowing hyperparameters within the model to be adjusted (Shah, 2017).

Test Dataset

The test dataset is the sample of data that is used to evaluate the final model fit, produced by the training process utilising the training and validation datasets (Shah, 2017). The test dataset is not introduced to the model during the training process, hence the results produced can be considered unbiased.

2.7. Feasibility and Justification

The literature review completed demonstrates the growing importance of a reliable sewer network in modern society. Sewer pipelines have a finite design life like all infrastructure assets, meaning they require monitoring throughout the expected lifespan and eventually renewal. To effectively manage complex infrastructure such as a sewer network, a proactive

and efficient asset management must be implemented to track the performance of network components. Accuracy and precision in asset management are imperative in maintaining a high level of service for the subject infrastructure. Efficiency directly influences the resulting economic performance of the system; that is, an inefficient system will impose direct cost to the user in both administration and to the physical asset.

Sewer pipelines should be inspected via robotic CCTV inspection or an equivalent means at a recurring interval to track the health of the asset over its lifespan. Sewer pipelines are subject to a large variety of possible defects throughout their lifecycle, including structural defects, operational defects, and construction defects. Different types of defects typically require varying repair methods, subject to the severity of the defect. By understanding the complexity and magnitude of the resourcing required to manage complex sewer infrastructure, the gap for increased efficiency and streamlined processes can be appreciated. Reducing the resources required to review and assess asset inspection data will provide direct economic benefit to the organisation. Furthermore, a developed deep learning network has the potential to match or outperform a human user who may experience fatigue or reduced concentration through the extensive review process; thus, additional economic benefit is provided through increased accuracy in the classified data.

The extensive capability of deep learning neural networks and computer vision technology offers significant potential as a substitute to conventional sewer asset inspection practices. Whilst some research into the application of deep learning recognition of sewer faults exists, studies were found to be isolated and, in most instances, lacked the completeness required for application in a real-world setting. Real-time detection ability is essential for this research to provide industry benefit; where this has been achieved by others, accuracy of the model typically suffers, and vice-versa. High levels of success were demonstrated in several studies that applied deep learning capability to road fault detection; this related research provides confidence in the potential for success of the proposed project.

The primary objective of the proposed research is to apply the advanced capabilities of deep learning neural networks to object detection in sewer pipelines, reducing the extensive labour required for traditional analysis. This research aims to develop a smart sewer fault detection model that is accurate and robust, whilst capable of demonstrating real-time detection speed.

3. Methodology

This chapter details the methodology implemented to achieve the project aim and objectives, employing information gained through the literature review completed in Chapter two. The proposed methodology aims to utilise a deep learning neural network to identify, locate and classify faults in sewer pipelines via CCTV inspection data.

The project methodology is comprised of two primary components; the model architecture, and the model development and evaluation. The first component defines the proposed architecture for the smart sewer detection model based on the literature review completed in Chapter two; the second component details the methodology implemented to develop and evaluate the smart sewer detection model. Figure 3.1 presents an overview of the model development and evaluation methodology:

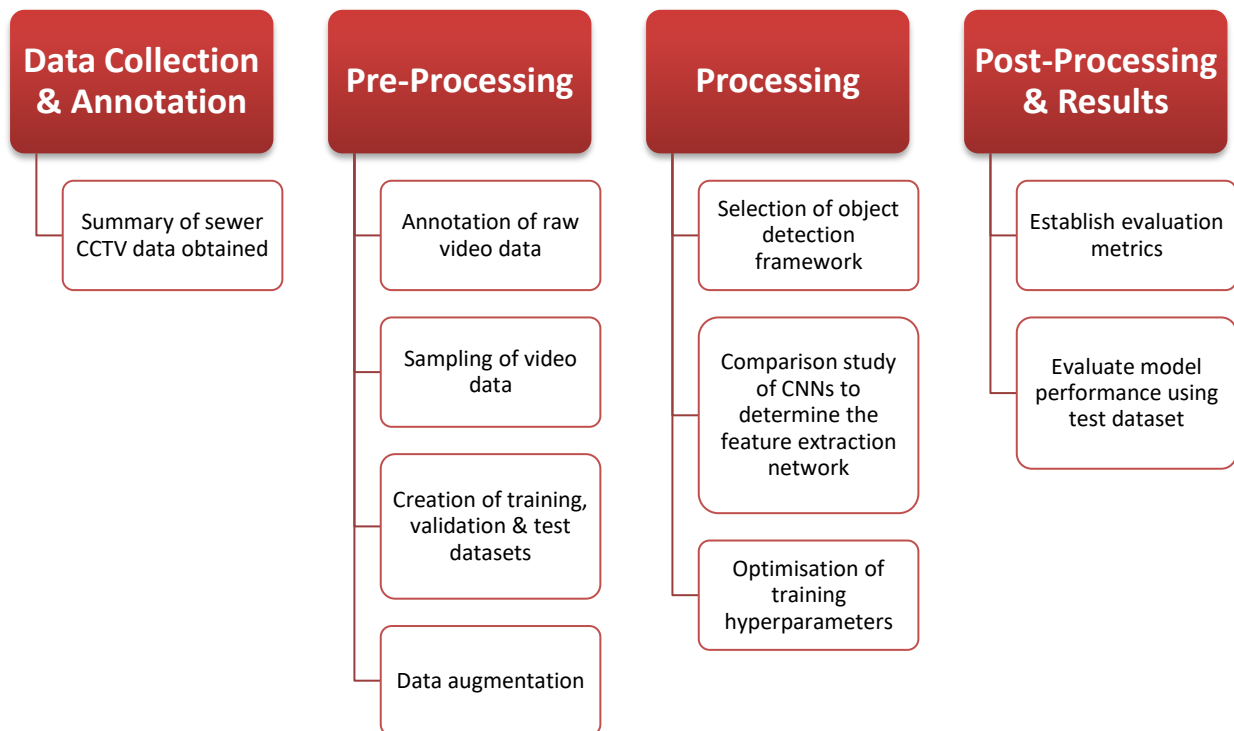


Figure 3.1: Model development & evaluation methodology

3.1. Architecture of Smart Sewer Detection Model

This section explains the components of the smart sewer pipeline fault detection model developed in this research. The common structure of an object detection model is defined as a feature extraction network, and an object detector; both components contain many layers that contribute to the overall network architecture. Figure 3.2 presents the proposed components of the smart sewer detection model, with the feature extraction network to be determined by Section 3.2 of the methodology. The final architecture of the smart sewer detection model is presented in Chapter four.

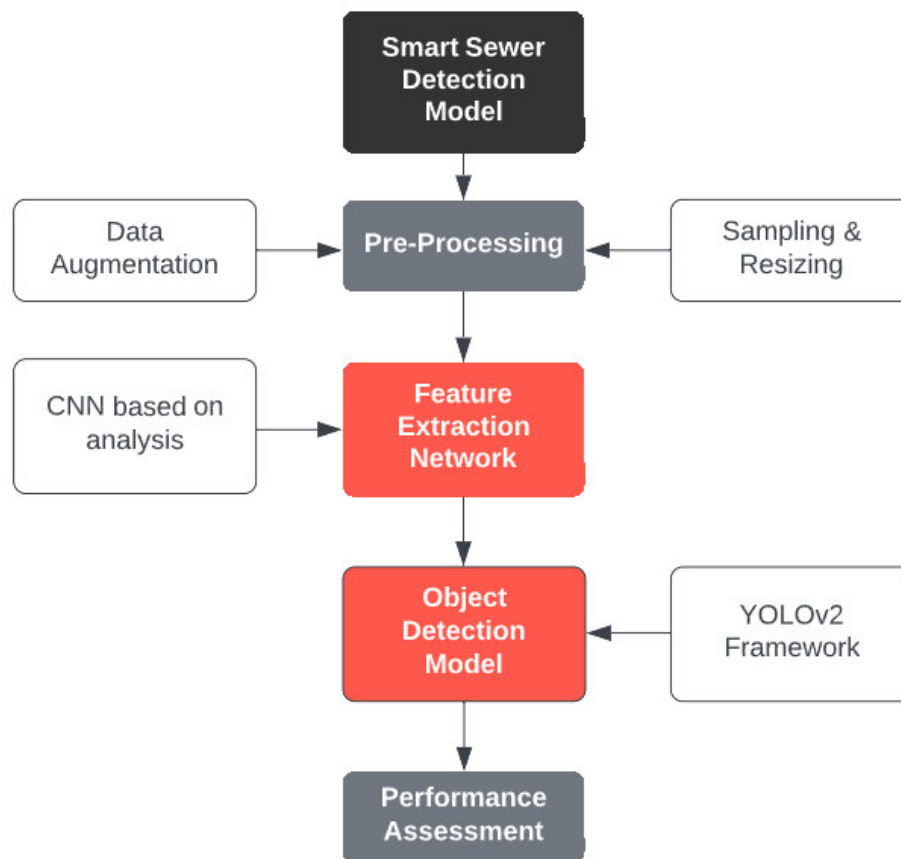


Figure 3.2: Flow Diagram of Smart Sewer Detection Model

3.1.1. YOLOv2 Object Detector

The review of literature included various common object detection frameworks, including YOLO, R-CNN (Fast & Faster) and SSD. As a primary aim of this research is to develop an object detection model with real-time detection capability, this research focuses on implementation of improved YOLO object detectors such as YOLOv2.

Implementation of the YOLOv2 object detector framework to widespread applications has reinforced its ability to perform at a high detection speed, well in excess of real-time detection,

whilst still maintaining high accuracy. This methodology utilises the refined ability of the YOLOv2 model to achieve real-time detection ability; a characteristic essential for effective practical implementation, however generally unachieved in comparable research (Li et al., 2022).

3.1.2. CNN Backbone

The feature extraction network interoperates input data provided by the model, extracts relevant features, and classifies the data, to then facilitate localisation by the object detector framework. As described within the literature review, there are many CNN architectures available for use in transfer learning; selecting a high performing CNN directly influences overall model performance, however, requires extensive analysis to maximise results. The implemented methodology evaluates various CNN architectures to determine the highest performing network in this application, then seeks to optimise hyperparameters within the preferred architecture. The analysis techniques implemented to determine the preferred CNN is detailed in Section 3.3.2.

3.2. Model Development and Evaluation

This section of the methodology describes the process implemented to develop, train, and evaluate the object detection model, including the analysis completed on components of the model. Specifically, this section will discuss the structure of the model framework, evaluation of feature extraction networks, and associated training hyperparameters.

3.2.1. Data Preparation

Data pre-processing describes the process undertaken to interpret and manipulate raw CCTV inspection data for use in training, validation, and testing of the developed fault detection program. This section of the methodology presents the various techniques employed to produce the quality dataset used within this research.

3.2.1.1. Data Collection & Annotation

The inspection data utilised in this research was obtained from Albury City Council's 2020-21 sewer CCTV inspection program; this is further discussed in Section 3.4.2.

Following compilation of raw data obtained through sewer asset inspection programs, the data must be manually annotated to facilitate training, validation, and testing of the network. Annotation of raw data can be completed through a variety of techniques discussed in Section

2.7.2, with the implemented method being the ‘Video Labeller App’, within the MATLAB Deep Learning Toolbox (MathWorks n.d.)

The point tracker algorithm was utilised to complete annotation of the dataset for this research project with moderate success. A combination of manual and automated labelling using the point tracking algorithm was employed to complete efficient annotation of the dataset, whilst maintaining accuracy required for quality training data. Figure 3.3 represents a label summary graph for an annotated sewer CCTV inspection video, where the y-axis shows the quantity of ROIs present in a frame, and the x-axis represents the video duration (seconds):

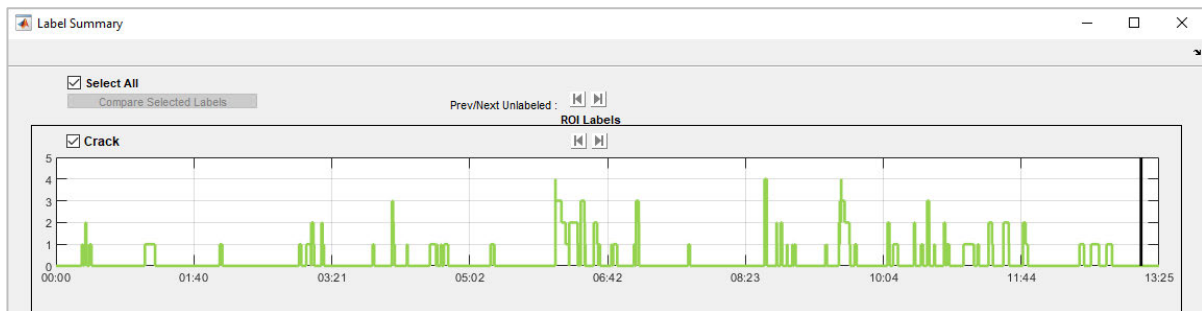


Figure 3.3: Label summary graph for CCTV inspection data annotated via Video Labeller App (MathWorks, n.d.)

3.2.1.2. Video Sampling

CNNs detailed within this research require input data to be in the form of common image formats, meaning video data must be processed prior to being input to the CNN. Automated image sampling has been incorporated into the fault detection model; ground truth data is interpreted by the model, then converted into an image datastore and box label data store which contain the image frames and ROIs respectively. This sampling occurs at a rate of five frames per second, which was found to produce noticeable differences between successive frames, while maximising the size of the dataset. A higher sampling rate may increase the likelihood of model overfitting, due to the similarity between successive frames; overfitting is discussed further in Chapter four of this research project.

3.2.1.3. Image Resizing

Object detection networks are generally developed to a target data input resolution; consequentially, optimal performance is typically achieved when resolution of the input image aligns with the network architecture (Solawetz, 2020). YOLO object detection networks incorporate functionality to resize images automatically, however it is understood that this can directly affect processing time and accuracy; further, an object detection model that is trained

on a specific image size may demonstrate varying reliability when test data an alternative resolution is input (Wu et al., 2018).

The YOLO object detection networks require an image input size that is a multiple of 32 (i.e. 224x224, 448x448, 608x608 etc) (Solawetz, 2020). For the YOLOv2 network, the minimum input size required is 224x224 pixels (MathWorks, 2022). The raw data obtained from Albury City Council's asset database primarily included .mp4 video files with a resolution of 768x576 pixels; consequentially, pre-processing will be undertaken to resize the data to conform with the network requirements. There are two common pre-processing methods utilised to data to the required resolution:

1. Compress image to required dimensions; this option is simplistic, however reduces image quality and can create undesirable distortion if the aspect ratio varies significantly.
2. Convert image into multiple tiles to align with the required dimensions; this technique preserves image quality and increases the number of images. This method is common for image classification models; however, can inhibit the performance of object detection models as it reduces the complexity of image background variance which may be desirable for a robust detection network.

It is proposed that option one will be adopted for its simplicity and effectiveness; the conversion from the native aspect ratio of 4:3 to a square image does not overly distort the data in this application. To automate the data resizing process, a transformation function was included in the MATLAB program, which transforms the image datastore to the nominated resolution. This can be found within the MATLAB program in Appendix 3 of this research paper. An example of a resized image is provided below:

Original Image (768x576 pixels):



Figure 3.4: Sewer Inspection Image with cracking (Albury City Council Sewer Inspection Program, 2021)

Resized Image (448x448 pixels):



Figure 3.5: Sewer Inspection Image with cracking (resized) (Albury City Council Sewer Inspection Program, 2021)

3.2.1.1. Training, Validation and Test Datasets

The primary dataset utilised for model development is focused on pipeline cracking defects. The population of this dataset is 3834 images, which are separated into training, validation, and test datasets as summarised in Table 3.1 below:

Dataset	Percentage	Images
Training	70%	2684
Validation	10%	383
Test	20%	767

Table 3.1: Training, Validation and Test Datasets

The dataset is limited to cracking defects due to the constrained timeline in which this research has been undertaken. Notwithstanding, analysis of model performance on other defect classes is completed in Section 4.4 of this dissertation, however the evaluation process of this was limited due to time constraints. Cracking defects were selected as the primary defect class due to the complex nature and variability associated with the visual appearance

of cracks. Further, the literature review undertaken indicated that model performance is consistently lower on cracking defects than other classes; this is reflected in research by Moradi (2020), Cheng and Wang (2018), and Yi et al. (2020); as such, it is likely that the model will exhibit equivalent or improved performance on other defects when trained on a multiclass dataset.

3.2.1.2. Training Data Augmentation

Augmentation of the training data was completed autonomously by including a function within the model to augment training data prior to the network training. The function applies various augmentation techniques including flipping, scaling and colour jitter as defined within the literature review; this effectively increases the training dataset by a factor of 4. The augmentation function utilised by the model is included in Appendix 4; an example of augmented training data is provided below in Figure 3.6:

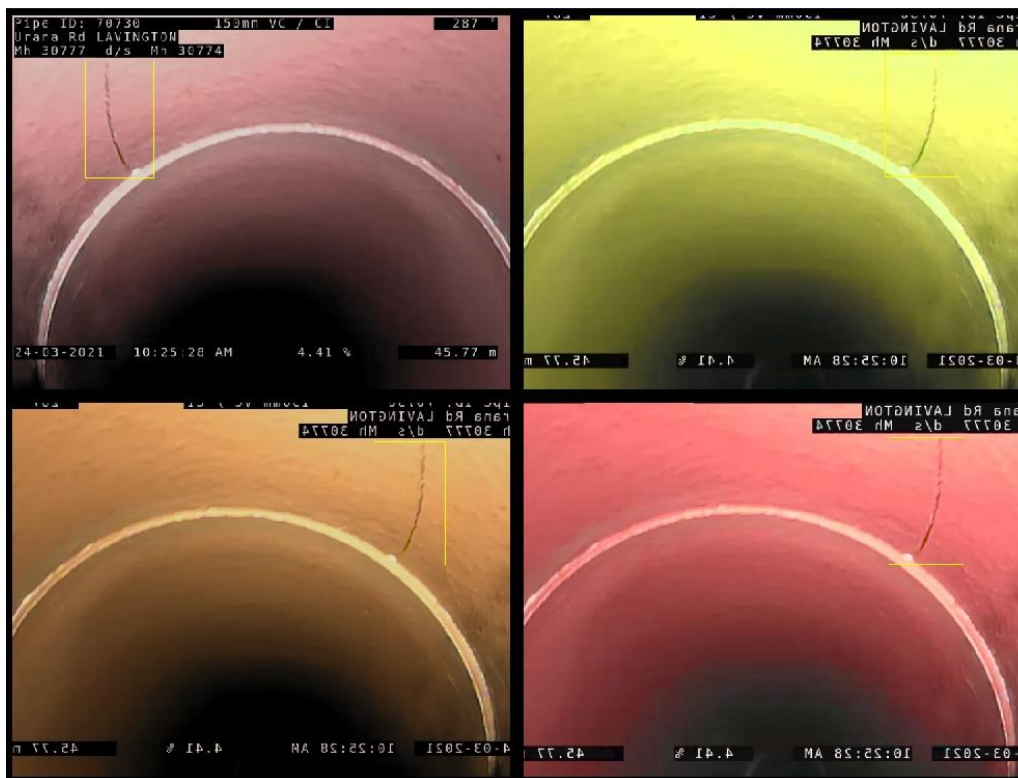


Figure 3.6: Example of training data after transformation function

3.2.2. Comparative Analysis of CNNs

The literature review completed identified several pre-trained CNNs that may be suitable as feature extraction networks for this application; these nominated CNNs are:

Feature Extraction Network	
Small-size CNNs (<i>less than 30 layers</i>)	Medium-size CNNs (<i>greater than 30 layers</i>)
AlexNet	Xception
GoogLeNet	Res-Net 50
VGG-16	Inception-ResNet-V2
VGG-19	InceptionV3*
SqueezeNet	Res-Net 101
Res-Net 18	
Darknet-19	

Table 3.2: Summary of CNNs for analysis

Preliminary training options are to be utilised to evaluate the performance of the CNNs, using the YOLOv2 object detector framework. The training hyperparameters will then be refined with the preferred feature extraction network utilised in the final object detection model. The preliminary training hyperparameters are outlined in Section 4.2.2.

Analysis of the twelve CNNs listed in Table 3.2 above will be completed on a reduced dataset due to resourcing constraints (Comparison 1); several of the highest performing CNNs will then be selected for extensive analysis on the primary dataset to determine the preferred CNN backbone (Comparison 2).

3.2.3. Optimisation of Training Hyperparameters

The object detection training process utilises test and validation datasets to train the detection model for the target task. The training process can be controlled through a range of training options known as hyperparameters. Training hyperparameters help the model estimate and develop the final parameters of the trained model. Section 3.3.3.1 provides a detailed explanation of key hyperparameters utilised for training.

3.2.3.1. Model Training Hyperparameters

Detailed explanation of the hyperparameters used in conventional training process is provided in Section 2.6.4.3 of this dissertation. A summary of the initial hyperparameters utilised in this research is shown below in Table 3.3; where hyperparameter values are noted as ‘varies’,

further analysis has been completed to determine the preferred value for this application. Results of the hyperparameter tuning and final values are discussed within Section 4.4 of this dissertation.

Training Hyperparameter	Hyperparameter Value
Optimiser	'adam'
MaxEpochs	Varies
MiniBatchSize	Varies ('8' typically)
Shuffle	'every-epoch'
ValidationFrequency	'50'
ValidationPatience	'Inf'
OutputNetwork	'last-iteration'
InitialLearnRate	0.0001
LearnRateSchedule	'none'
LearnRateDropPeriod	N/A
LearnRateDropFactor	N/A
L2Regularisation	0.005
Momentum	N/A
GradientDecayFactor	'0.9'
SquaredGradientDecayFactor	'0.999'
ResetInputNormalisation	'false'
BatchNormalisationStatistics	'moving'
ExecutionEnvironment	'gpu'

Table 3.3: Training options for object detection model training, adapted from MathWorks (n.d.)

Key hyperparameters identified from the literature review include 'MaxEpochs' and 'MiniBatchSize'; iterative analysis will be undertaken to determine the preferred value for these hyperparameters in the application of sewer pipeline fault detection.

3.2.4. Evaluation of Detector Performance

The smart sewer pipeline detection model must be evaluated at various stages throughout the methodology, including during adjustment of hyperparameters, and testing of the finalised model. Evaluation of the detection model is completed using a predetermined testing dataset that has not been seen by the network during training and validation. A test dataset is a dataset with known outcomes, that is input to the system as raw data; the success rate is then evaluated against the known outcomes to determine the systems performance. The performance is measured based on the following metrics as detailed in the literature review:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths}$$

$$AP = \sum_{k=0}^{k=n-1} [Recall(k) - Recall(k + 1)] \times Precision(k)$$

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

The detection speed will also be recorded by capturing the time taken for the model to process the test dataset; this will be used to derive a performance metric of frames per second. Whilst the threshold for ‘real-time’ detection is not explicitly defined, the target FPS for this research is greater than 30fps, as most sewer inspection CCTV videos are between 25 and 30fps.

3.2.1. Multiclass Object Detection

This dissertation primarily focuses on ‘cracking’ defects due to the constrained timeline in which it must be completed. Notwithstanding, the smart sewer detection model will be applied to a revised multiclass dataset to determine its performance in such a task. As hypothesised in Section 3.2.1.1, supporting literature indicates the model should exhibit increased AP results on alternative defect classes, due to the complexity of cracking defects. Performance of the smart sewer detection model will be evaluated utilising metrics detailed above.

3.3. Project Resources

Various resources will be required to achieve the aims and objectives outlined in the proposed research. The primary elements required for this research are outlined below; additional commonly available computer software such as Microsoft Excel was also implemented in this research. Further, this project requires extensive data classification and analysis to develop a high performing deep learning network; consequentially, the project will require extensive allocation of personal time; timeline information is provided in the Project Specification appended to this report (refer Appendix 1).

3.3.1. MathWorks MATLAB

The literature review completed indicates that MathWorks MATLAB provides appropriate functionality and capability for this project. MathWorks Inc (2021) offers the Deep Learning Toolbox™ which “provides a framework for designing and implementing deep neural networks with algorithms, pretrained models, and apps”.

3.3.2. CCTV Sewer Inspection Data

The inspection data utilised in this research was obtained from Albury City Council’s 2020-21 sewer CCTV inspection program. This program included over 10 lineal kilometres of sewer pipelines, including varying material types, internal diameters, and pipe condition; this allowed an extensive dataset to be developed for use in this research. Acknowledgement to Albury City Council is noted for this contribution.

3.3.3. Computer System for Analysis

Computer Component	Specification
Custom Workstation	
System Model	Custom
CPU	Intel i7-4790k 4 Cores 8 Threads 8MB Cache 4.40GHz (Max Boost)
GPU	Nvidia RTX3080 8704 CUDA Cores 10GB GDDR6X VRAM
RAM	16GB DDR3
Operating System	Windows 10
Nvidia DGX	
System Model	Nvidia DGX Station A100
CPU	AMD 7742 64 Cores 3.4GHz (Max Boost)
GPU	4x Nvidia A100
RAM	512GB DDR4
Operating System	Ubuntu Linux OS

Table 3.4: Summary of Computing Resources

3.4. Project Consequences & Ethical Assessment

The Code of Ethics developed by Engineers Australia (2019) states that “As engineering practitioners, we use our knowledge and skills for the benefit of the community to create engineering solutions for a sustainable future. In doing so, we strive to serve the community ahead of other personal or sectional interests”. The Code of Ethics defines the values and principles by which professional engineers undertake the practice of engineering; the four key elements outlined by Engineers Australia are (Engineers Australia, 2019):

- a) Demonstrate integrity
- b) Practice competently
- c) Exercise leadership
- d) Promote sustainability

In undertaking all elements of this dissertation, the Code of Ethics by Engineers Australia (2019) has been strictly adhered to.

3.5. Risk Assessment

Refer to Appendix 2 for the risk assessment undertaken for this project. The risk assessment template has been adapted from the University of Southern Queensland template. As the working environment of this project is essentially limited to computer use, the potential risks are limited to personal safety, ergonomics, and project delivery timelines.

4. Data Analysis & Discussion

This chapter details the results and findings derived from the development of the smart sewer pipeline fault detection model. Firstly, the preliminary architecture and hyperparameters are discussed, as well as the initial dataset. Then, analysis of various CNNs is completed to determine a preferred feature extraction backbone for the model. Following this, analysis of training hyperparameters is undertaken to optimise the smart sewer pipeline fault detection model. This section is concluded by a discussion of findings and associated outcomes.

4.1. Initial Model Structure & Parameters

This section proposes the preliminary model structure and training hyperparameters used for analysis. This preliminary model will be optimised through multiple phases of analysis as outlined in the methodology of this dissertation.

Preliminary architecture of the smart sewer pipeline fault detection model utilises the YOLOv2 detection framework as defined in Section 3.1; this is used to facilitate comparison of various CNN backbones as shown in Section 4.2 below.

4.1.1. Initial Training Hyperparameters

General training parameters utilised throughout this research are defined in Section 3.2.3 of this paper. Additionally, the literature review and methodology determined several key hyperparameters that will be optimised through iterative analysis after a preferred model structure has been determined; initial values for the key hyperparameters are defined below:

Training Hyperparameter	Hyperparameter Value
MaxEpochs	30
MiniBatchSize	8
Detection MiniBatchSize	64

Table 4.1: Key training hyperparameters for analysis

Training hyperparameters not listed above are defined Section 3.2.3 of this paper and remain constant throughout the investigations undertaken.

4.2. Analysis of CNNs

As discussed previously in Chapter three, a total of twelve CNNs were selected for evaluation to determine the preferred feature extraction backbone for the smart sewer pipeline fault detection model. The analysis was completed in two primary stages, being an initial preliminary comparison of the twelve CNNs, following by extensive further analysis of the preferred CNNs from the first stage. In all cases, the CNNs were trained using deep transfer learning, with consistent training hyperparameters as outlined in Section 4.1.1.

4.2.1. Comparison 1: Preliminary Analysis

In the first comparison, the twelve CNNs were integrated with the YOLOv2 object detection framework and evaluated using a reduced dataset of 527 images to allow all CNNs to be assessed within the available timeline; results of the initial investigation are summarised below in Table 4.2:

Feature Extraction Network	Average Precision	Training Time (hh:mm:ss)	Detection Speed (frames per second)
Small-size CNNs (less than 30 layers)			
AlexNet	0.61	00:13:20	54.50fps
GoogLeNet	0.67	00:15:55	48.07fps
VGG-16	0.72	00:17:43	31.05fps
VGG-19	0.72	00:19:35	30.06fps
SqueezeNet	0.73	00:14:44	49.25fps
Res-Net 18	0.75	00:15:44	53.91fps
Darknet-19	0.82	00:17:38	37.06fps
Medium-size CNNs (greater than 30 layers)			
Xception	0.71	00:27:19	36.70fps
Inception-ResNet-V2	0.76	01:44:53	38.26fps
InceptionV3*	0.78	00:34:01	43.54fps
Res-Net 50	0.82	00:22:46	43.91fps
Res-Net 101	0.83	00:42:06	42.01fps

Table 4.2: Summary of initial CNN backbone analysis

As shown in the table above, Darknet-19, Res-Net 50 and Res-Net 101 were determined to be the preferred CNN backbones in the preliminary analysis. The three CNNs each achieved an average precision greater than 80%, whilst maintaining real-time detection speeds between 37 and 44 frames per second. Precision-recall curves from the preliminary analysis are provided below in Figures 4.1-4.3 for the preferred networks:

Darknet-19

The Darknet-19 CNN is the native feature extraction network for the YOLOv2 object detector; an average precision of 82% was achieved in the preliminary evaluation trial:

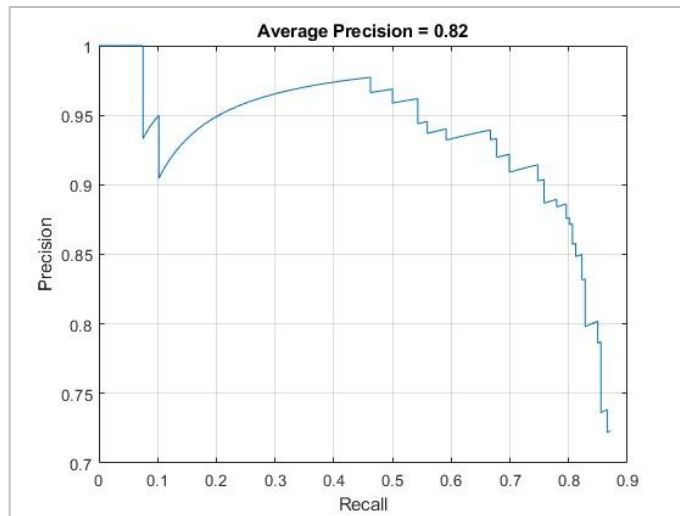


Figure 4.1: Darknet-19 CNN preliminary evaluation:

Res-Net 50

The Res-Net 50 CNN was integrated using layer 'activation_40_relu' as the feature layer; an average precision of 82% was achieved in the preliminary evaluation trial:

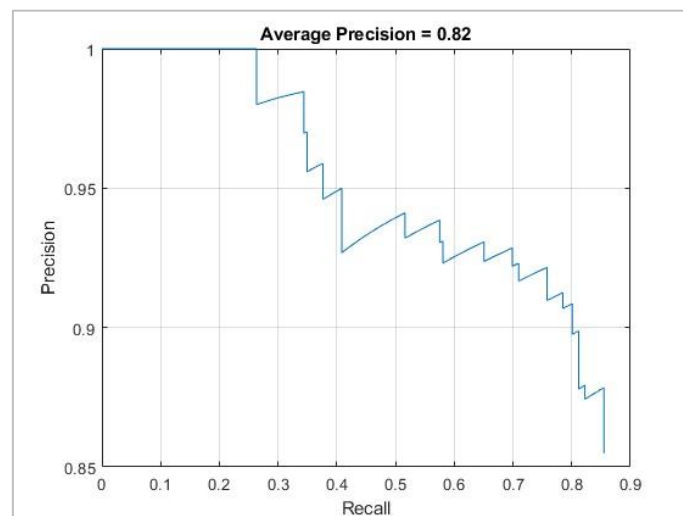


Figure 4.2: Res-Net 50 CNN preliminary evaluation

Res-Net 101

The Res-Net 101 CNN is the deepest network of the three preferred and was integrated with ‘res4b22_relu’ as the feature layer. Res-Net 101 achieved the highest average precision of 83% in the preliminary evaluation trial:

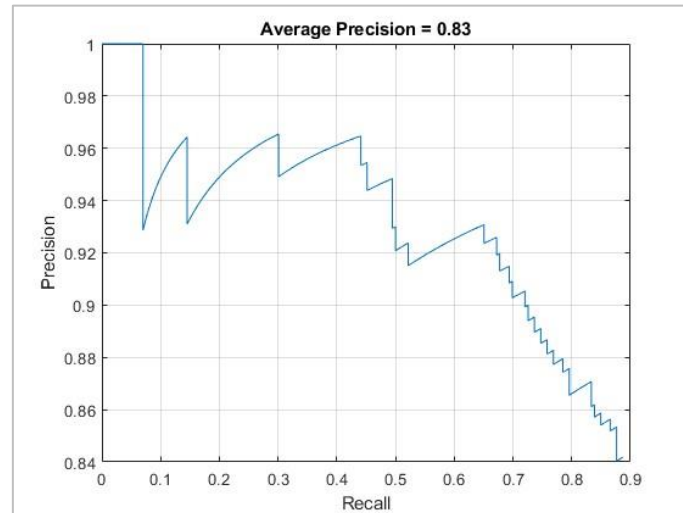


Figure 4.3: Res-Net 101 CNN preliminary evaluation

4.2.2. Comparison 2: Detailed Analysis

From the initial comparison trials completed above, three networks were selected for further evaluation on the primary dataset (3834 images). This comparison was repeated four times to identify any outliers or instability within the preliminary models; Figure 4.4 below presents the mean of the four iterations completed:

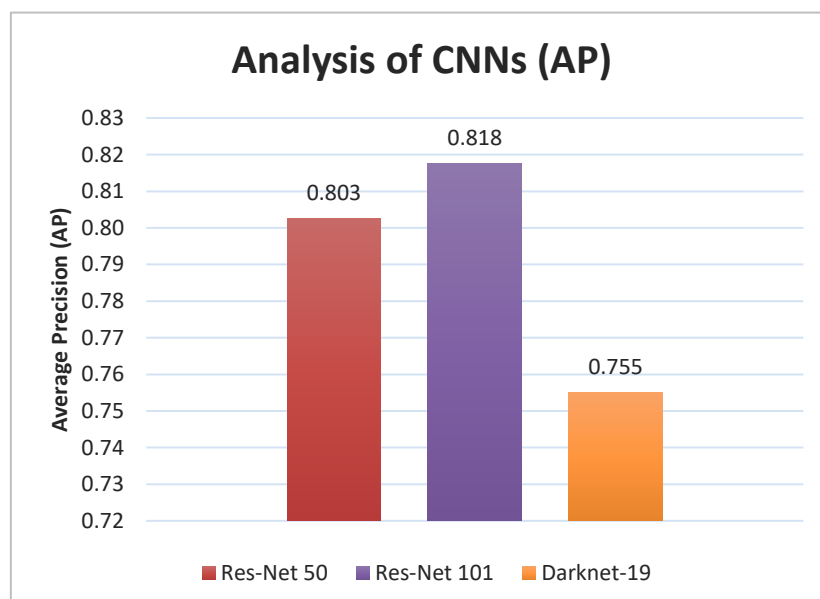


Figure 4.4: CNN Comparison on primary dataset

The Res-Net 101 CNN backbone attained the highest average precision score of the three networks, being 81.8%. Whilst the result of Res-Net 50 was within a narrow margin, Darknet-19 returned a notably lower average precision score of 75.5%. The Res-Net 101 precision-recall curve for the median average precision iteration is provided below in Figure 4.5:

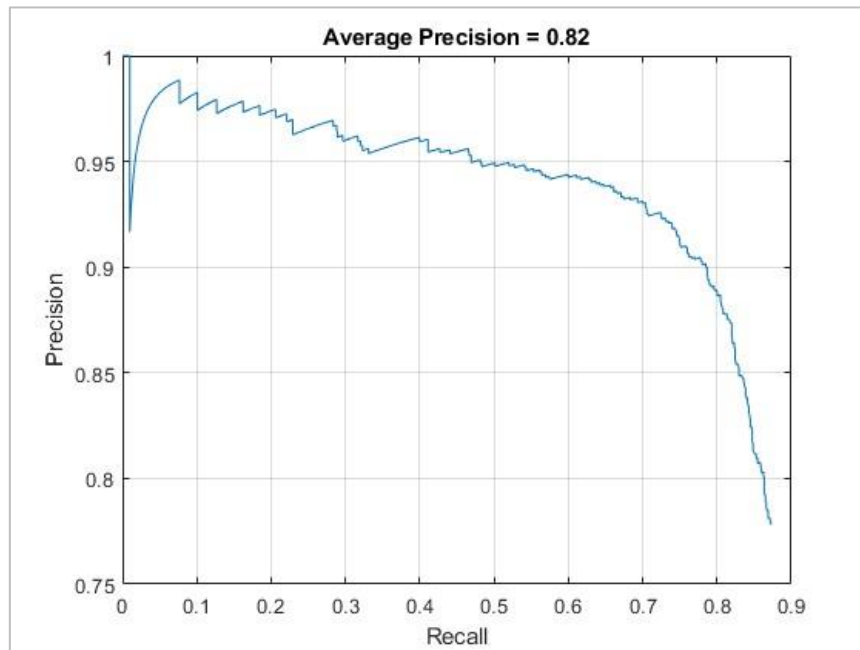


Figure 4.5: Precision-recall curve for median Res-Net 101 result

The training and validation loss information is plotted in Figure 4.6 to show the network training behaviour. This figure shows slight overfitting beginning to occur; whilst it is considered negligible in this instance, it indicates that the network is not improving from further iterations. It can be inferred that an increase in the training dataset would further improve model performance in this instance.

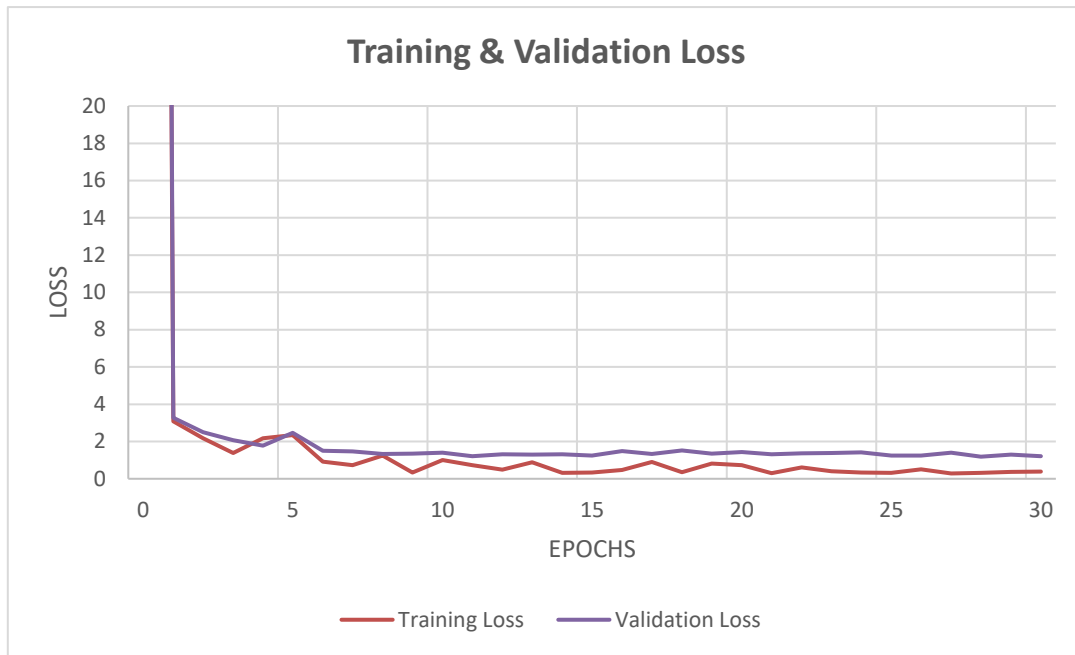


Figure 4.6: Plot of Training & Validation Loss for Res-Net 101

In addition to the average precision metric, model detection speed when utilising the three different CNNs was recorded for comparison. It was found that all three detectors maintained a speed over 30 frames per second; thus, all models exhibit real-time detection ability.

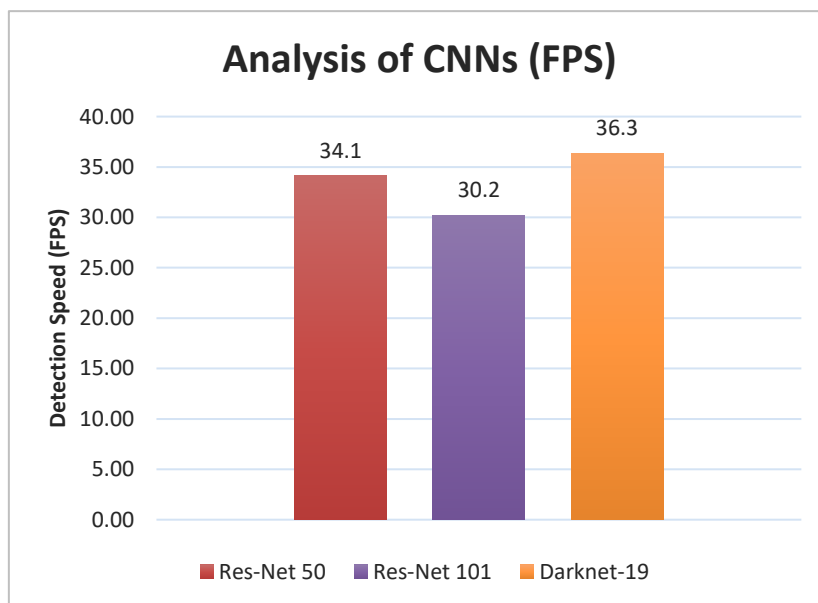


Figure 4.7: CNN Comparison on primary dataset

From the comparison of CNN architectures completed, it can be concluded that the Res-Net 101 CNN is the preferred feature extraction backbone for the smart sewer pipeline detection model. The average precision result of 81.8% represents strong performance in detecting and localising cracking faults within the CCTV data, whilst still maintaining a detection speed greater than 30 frames per second.

4.3. Evaluation of Hyperparameters

As the structure of the smart sewer pipeline detection model has been determined based on the preferred CNN backbone, the network training parameters can now be optimised to further improve detector performance and robustness. This will be completed by undertaking multiple iterative analysis' where key hyperparameters are individually evaluated to determine optimal settings. Chapter three defines the key hyperparameters selected for analysis; these are restated below in Table 4.3:

Training Hyperparameter	Description
MaxEpochs	An epoch is a full pass of the training algorithm over the entirety of the training dataset; 'MaxEpochs' defines the maximum number of epochs the network training can complete before finishing.
MiniBatchSize	The mini-batch is a subset of the training dataset used to evaluate the loss function gradient and update the layer weights. The 'MiniBatchSize' describes the size of the mini-batch.

Table 4.3: Hyperparameters for model training, reduced from Section 2.5.4.3 (MathWorks, n.d.)

Evaluation of training hyperparameters was completed using the Nvidia DGX A100 Station at the University of Southern Queensland. As detailed in Section 3.3.3, the DGX is a server grade system designed for artificial intelligence and deep learning; this system facilitates network training options that may not be possible on a conventional computer system, such as large values for the 'MiniBatchSize' hyperparameter.

Whilst the CNN comparison determined Res-Net 101 to be the preferred CNN backbone for the smart sewer pipeline detection model, the high-speed multi-GPU functionality of the DGX allows up to four training instances to run parallel; hence, training hyperparameters were also evaluated on the Res-Net 50 and Darknet-19 versions of the model for further comparison.

4.3.1. MiniBatchSize

The research undertaken in Chapter two indicated that the effect 'MiniBatchSize' on detector performance varies depending on the specific application, hence, it is imperative to complete optimise this hyperparameter for the selected task.

The MiniBatchSize variable was analysed for values between 4 and 64; similarly to the CNN backbone evaluation, the analysis was repeated four times to achieve a representative sample. Figure 4.8 below presents the results of the analysis on the 'MiniBatchSize' hyperparameter:

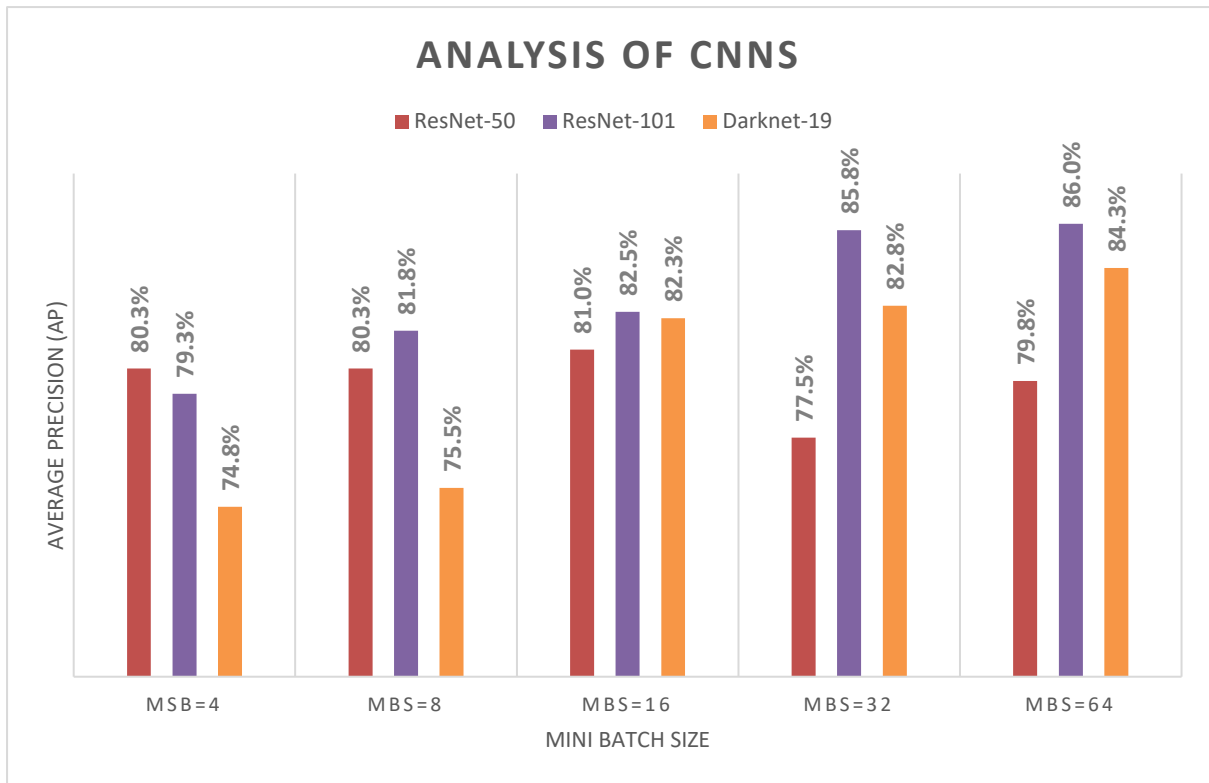


Figure 4.8: Analysis of 'MiniBatchSize' training hyperparameter

The results presented in Figure 4.8 generally indicate a favourable trend toward the higher 'MiniBatchSize' of 32 and 64, with only Res-Net 50 indicating a preferred value of 16. By inspection, it is noted that one Res-Net 50 trial resulted in an average precision of 68% for 'MiniBatchSize' 32, which is generally inconsistent with other iterations and may be an outlier. Notwithstanding, it is maintained in the comparison above as it cannot be excluded without doubt; however is noted that exclusion of the iteration would increase the average precision to 80.7% for Res-Net 50 (MiniBatchSize=32).

The highest performance was achieved with the preferred Res-Net 101 CNN, returning an average precision of 85.8% and 86.0% for 'MiniBatchSize' 32 and 64 respectively. The Res-Net 50 CNN performed best on a 'MiniBatchSize' of 64, returning an average precision of 84.3%.

The trend between increased 'MiniBatchSize' and average precision infers a positive linear correlation, meaning that an increase in the hyperparameter will likely result in improved detector performance within the range tested. From the literature review undertaken in Chapter two, this is likely due to the accuracy of gradient adjustments being increased, as the calculation is completed using more data due to the larger batch size (Peng et al., 2018). Analysis of larger 'MiniBatchSize' values (128, 256 etc) was not undertaken due to time resourcing constraints.

Based on the literature review, it is expected that detector performance would eventually decrease with increased batch size (Kandel & Castelli, 2020); further, the results obtained indicate limited benefit increasing from a MiniBatchSize of 32 to 64, meaning it is plausible that a decrease would be realised if the hyperparameter was increased to 128 or larger. Additionally, it is noted that high performance computing capability is required to utilise larger batch sizes, with the Nvidia RTX3080 GPU experiencing unreliability for a ‘MiniBatchSize’ larger of 32 or larger for this model when tested.

Considering the above, it is proposed that a ‘MiniBatchSize’ of 32 is adopted when completing further analysis on other training hyperparameters. The adopted value considers the computational cost benefits of the smaller ‘MiniBatchSize’ and recognises the increase from 32 to 64 yields insignificant benefit to detector performance.

4.3.2. Maximum Epochs

As discussed in Chapter two, ‘MaxEpochs’ describes the maximum number of epochs permitted in the training process of the model, where an epoch is complete pass through the entire dataset. ‘MaxEpochs’ effectively defines the amount of time the model spends learning the data; as detailed in Section 2.5.4.3, limited epochs will reduce model performance as it will not learn adequately, whereas a high ‘MaxEpochs’ setting will promote overfitting.

Similarly to the ‘MiniBatchSize’ comparison, the three preferred CNNs were evaluated to determine the optimal ‘MaxEpochs’ hyperparameter for this application. Comparison between 10, 20, 30, 40 and 50 epochs were completed, with key training hyperparameters for this analysis are defined below in Table 4.4:

Training Hyperparameter	Hyperparameter Value
Training Dataset	3834 Images
MaxEpochs	10, 20, 30, 40 & 50
MiniBatchSize	32
Detection MiniBatchSize	64
InitialLearnRate	0.0001

Table 4.4: Key training hyperparameters for analysis

The analysis was again repeated four times to achieve a representative sample; Figure 4.9 and Table 4.5 below present the results of the analysis on the ‘MaxEpochs’ hyperparameter:

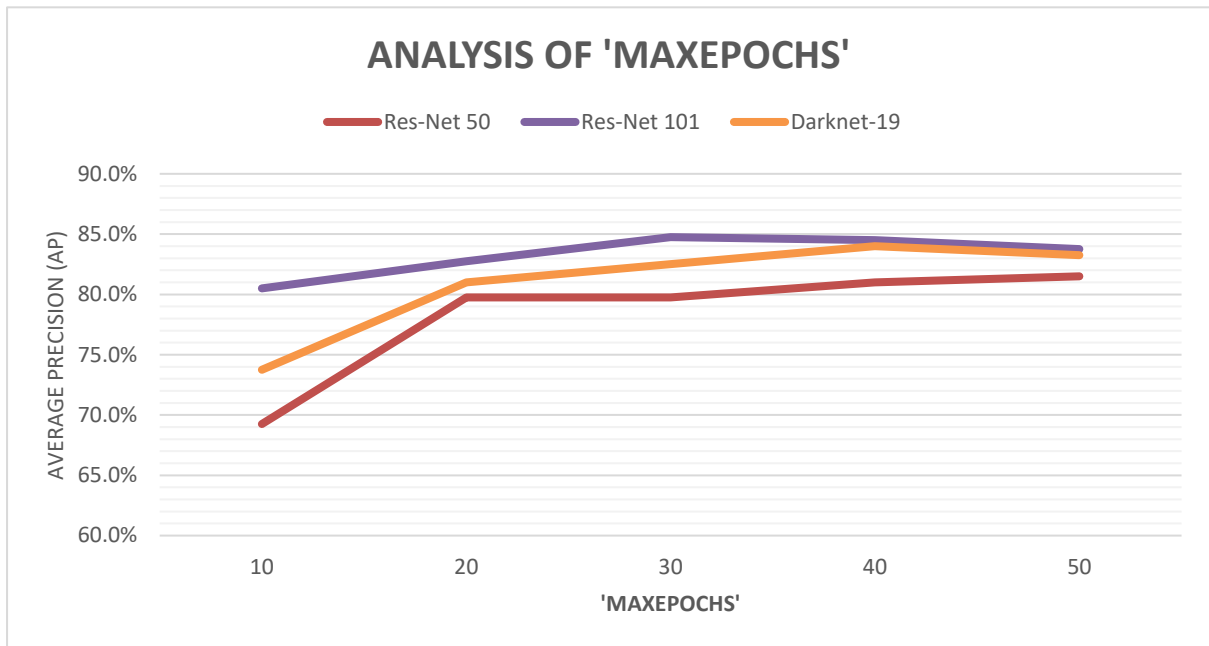


Figure 4.9: Analysis of 'MaxEpochs' training hyperparameter

Analysis of 'MaxEpochs'					
'MaxEpochs'	10	20	30	40	50
Res-Net 50	69.3%	79.8%	79.8%	81.0%	81.5%
Res-Net 101	80.5%	82.8%	84.8%	84.5%	83.8%
Darknet-19	73.8%	81.0%	82.5%	84.0%	83.3%

Table 4.5: 'MaxEpochs' Analysis Summary Tabulated

The results presented above demonstrate that the preferred value for 'MaxEpochs' varies between the three CNNs, however the variance in the average precision metric between the hyperparameter values of 30, 40 and 50 is insignificant. All model variations performed worse at 'MaxEpoch' values of 10 and 20, than values of 40 or 50.

The Res-Net 50 network returned the highest average precision value (81.5%) when trained for 50 epochs; however, the performance increase over 20 'MaxEpochs' was limited to 1.7%. It is noted that detector performance was still increasing at 50 'MaxEpochs', however it is considered unlikely that further increases in performance would be achieved without the possibility of overfitting.

The Res-Net 101 network again achieved the highest average precision value (84.8%) of all CNNs, with the optimal 'MaxEpochs' value being 30. An average precision of 84.5% and 83.8% were achieved for 'MaxEpoch' values of 40 and 50 respectively; whilst this is a minor negative trend, it is considered to be within typical training variance. It can be concluded that

‘MaxEpoch’ values greater than 30 do not significantly improve performance of the trained model in this application. A recall-precision curve for the iteration where an average precision of 85.6% was returned is presented below in Figure 4.10:

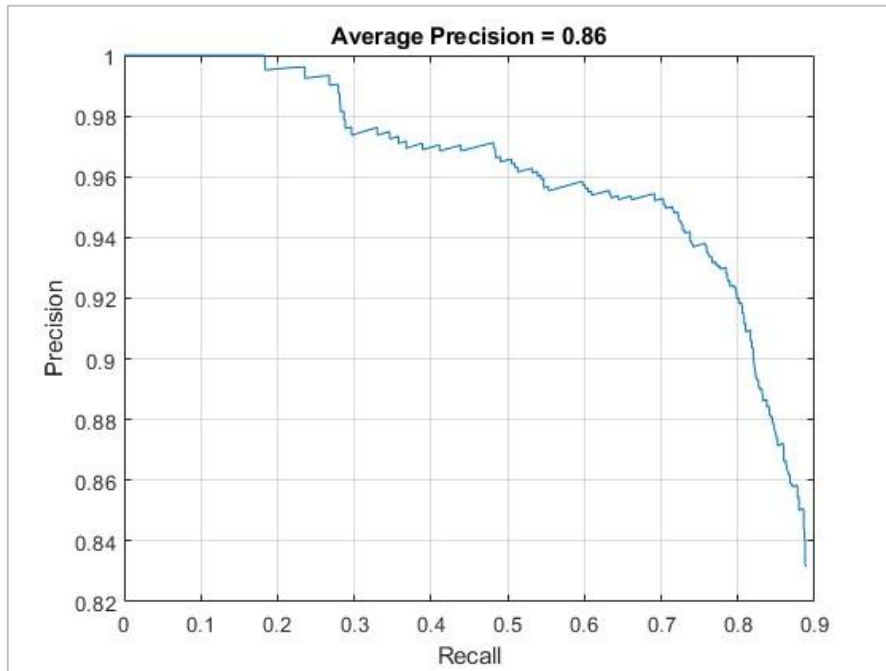


Figure 4.10: Precision-recall curve for Res-Net 101 trained for 30 epochs

The Darknet-19 variation of the detection model achieved an optimal result of 84.0% at a ‘MaxEpochs’ value of 40. Performance of the detector reduces slightly (0.7%) at a value of 50 epochs; however this is considered too small to infer a conclusion.

In summarising the analysis, it can be generally concluded that ‘MaxEpoch’ values of 30-50 are preferred for the smart sewer fault detection model. A value of 40 epochs will be adopted for the proposed model, and further analysis moving forward. In practice, 40 epochs demonstrated consistent performance across all CNNs tested and is unlikely to promote overfitting of the model. Further, utilising a value of 40 epochs increases resiliency to complex datasets that may require additional learning time.

4.4. Alternative Defect Classes

The primary dataset used for the analysis in Sections 4.2 and 4.3 contained only images of cracking defects within the sewer pipelines; the primary reason for this was due to the limited timeframe in which this research was undertaken. Notwithstanding, the realised time available allowed the smart sewer detection model to be trained and tested utilising an alternative dataset containing two additional defect classes.

4.4.1. Multiclass Detection Performance

To determine the performance of the smart sewer detection model on alternative defect classes, additional data was prepared and annotated containing defects such as root intrusions and deposits. The revised dataset containing 3626 images was used to analyse performance of the model on three defect classes simultaneously. Table 4.6 shows the occurrence of each defect class within the dataset (note, frames containing multiple defects of the same type are only counted as one occurrence):

Defect Class	Occurrence	Percentage of Total
Crack	1931	46.3%
Deposit	735	17.6%
Root Intrusion	1502	36.1%
Combined	4168	100%

Table 4.6: Multiclass Dataset Information

Analysis of the multiclass dataset was completed utilising the model developed in earlier sections of this Chapter; key information relating to the model and associated training hyperparameters is restated in Table 4.7 below for clarity. Data pre-processing techniques such as data augmentation and division of training, validation and test subsets remain consistent with the methodology detailed in Chapter 3.

Variable	Value
Smart Sewer Detection Model	
Object Detection Framework	YOLOv2
Feature Extraction Network	Res-Net 101
Feature Extraction Layer	'res4b22_relu'
Key Training Hyperparameters	
'Optimiser'	Adam
'InitialLearnRate'	0.0001
'MiniBatchSize'	16
'MaxEpochs'	40

Table 4.7: Summary of model information for multiclass analysis

The findings of average precision for the three defect classes are presented below in Table 4.8, as well as the mean average precision (mAP) for the detection model; the associated precision-recall curves for the median iteration are provided in Figures 4.11 – 4.13. The analysis was repeated four times to achieve a representative sample and determine stability of the model. From the results presented below, it is evident that the smart sewer detection model is adaptable to other defect classes, having achieved significantly higher performance for ‘root intrusion’ and ‘deposit’ defects than ‘cracking’ defects. Further, the average detection speed of the trained model is 46.7fps.

Smart Sewer Detection Model – Multiclass Analysis			
AP			mAP
Crack	Deposit	Root Intrusion	
77.5%	96.3%	94.8%	89.3%

Table 4.8: Multiclass Object Detection Results

Precision-Recall Curve: ‘Cracking’ Defects

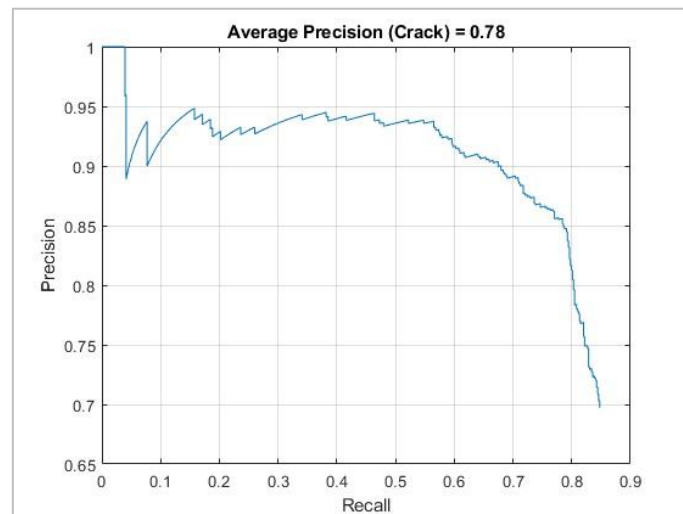


Figure 4.11: Precision-recall curve for ‘cracking’ defects

Precision-Recall Curve: 'Deposit' Defects

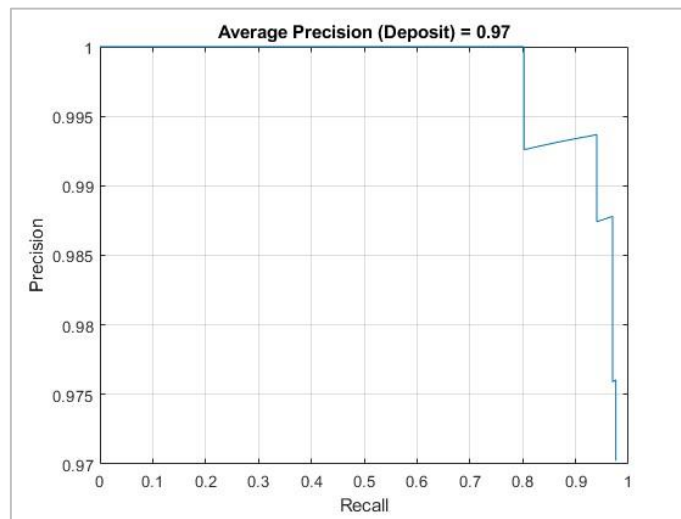


Figure 4.12: Precision-recall curve for 'deposit' defects

Precision-Recall Curve: 'Root Intrusion' Defects

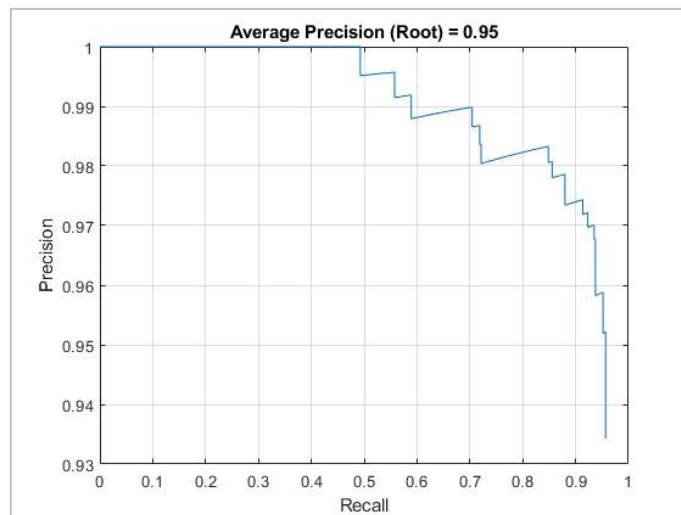


Figure 4.13: Precision-recall curve for 'root intrusion' defects

A graph showing the training and validation loss behaviour is provided below in Figure 4.14. Similar to Figure 4.6, the training behaviour indicates a negligible degree of overfitting; whilst this has minimal impact on detector performance, it is possible that the number of epochs could have been reduced to shorten training time with no impact to model performance. Notwithstanding, time constraints did not facilitate further investigation.

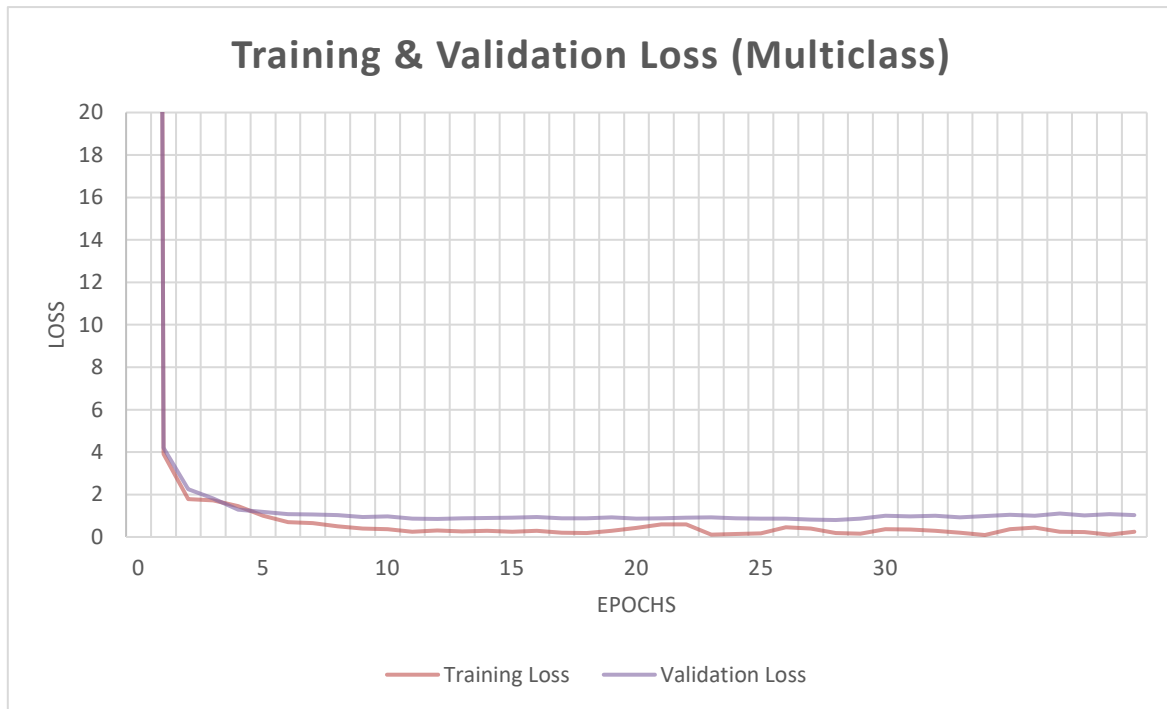


Figure 4.14: Training & validation loss for multiclass defect training

4.5. Discussion

The outcomes of this research satisfy the intended aim and objectives set out within Chapter one, having developed an accurate and precise framework capable of real-time detection. The smart sewer detection model is an innovative technology capable of rapidly and accurately detecting a range of faults within sewer pipelines. To create this model, deep transfer learning and object detection technology was utilised; the model was trained on a comprehensive dataset derived from raw CCTV data, and then tested on unseen data to evaluate its performance.

Through the methodology of comparative CNN analysis and hyperparameter optimisation, an adapted version of the YOLOv2 object detection model was created, utilising the Res-Net 50 CNN as the feature extraction network. The Res-Net 101 network was preferred out of the twelve CNNs in the comparative analysis, which included common networks such as AlexNet and Darknet-19. This smart sewer detection model achieved a mAP of 89.3% and detection speed of 46.7fps in the final multiclass analysis.

4.5.1. Smart Sewer Detection Model

The smart sewer detection model that was developed in this research utilises the YOLOv2 object detection framework, with adaption of the Res-Net 101 CNN as the feature extraction network in lieu of the native Darknet-19 CNN. A comparative study was completed on twelve common CNNs, of which the initial selection was informed through the literature review in Chapter two; the CNNs were evaluated based on the average precision metric, as well as detection speed. Figure 4.15 below presents the finalised architecture of the smart sewer detection model:

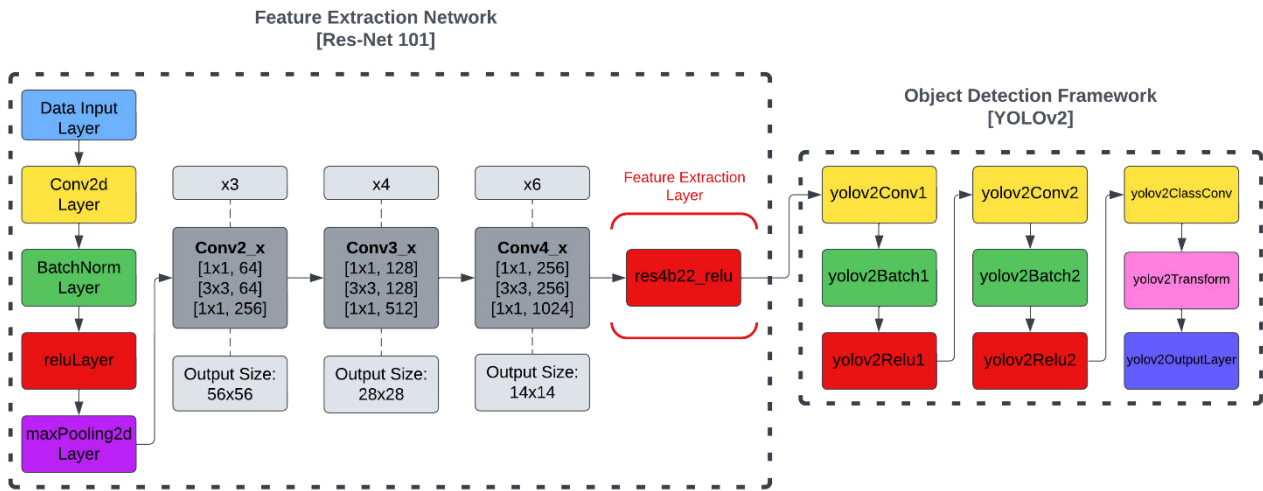


Figure 4.15: Architecture of 'Smart Sewer Detection Model'

4.5.2. CNN Evaluation Findings

Utilising the YOLOv2 framework, the twelve CNNs were evaluated by metrics of average precision and detection speed on a preliminary dataset containing 527 images. AlexNet achieved the highest detection speed of 54.5fps, however the average precision was the lowest of all CNNs at 61%. GoogLeNet was the only other CNN to return an average precision score below 70.0%. The three preferred CNNs from the preliminary comparison were Darknet-19, Res-Net 50 and Res-Net 101, all returning an average precision score greater than 80.0%; the remaining seven CNNs scored between 70.0% and 80.0%. All twelve networks that were evaluated achieved a real-time detection speed (>30fps); the slowest results were VGG-16 and VGG-19 which achieved 31.05fps and 30.06fps respectively. The slower detection speeds achieved by the VGG CNNs can be attributed to the large number of parameters and high computational cost, which was discussed further in Chapter two (Alzubaidi et al., 2021). The preferred networks Darknet-19, Res-Net 50 and Res-Net 101 returned detection speeds of 37.06fps, 43.91fps and 42.01fps respectively.

Following on from the preliminary analysis, the three preferred CNNs were further evaluated to determine the most suitable network for use in the smart sewer detection model. The three CNNs were evaluated on an increased dataset of 3834 images; this analysis was repeated four times to demonstrate stability of the model. This investigation was completed using the Nvidia A100 DGX Station at the University of Southern Queensland; use of this computing system allows multiple simultaneous trials to be completed. Results of the analysis indicated that Res-Net 101 is the preferred CNN, returning an average precision value of 81.8%; Res-Net 50 and Darknet-19 achieved an average precision of 80.3% and 75.5% respectively. From the results, it can be inferred that the increased depth of Res-Net 101 is beneficial in this application; however, the Res-Net 101 CNN was slower in the detection speed metric than other CNNs, likely due to the network size. Res-Net 101 achieved a detection speed of 30.2fps, where Res-Net 50 and Darknet-19 achieved 34.1fps and 36.3fps respectively. Notwithstanding, all three CNNs evaluated in the comparison demonstrated real-time detection capability, thus Res-Net 101 was determined to be the preferred feature extraction network due to its higher average precision. Whilst the dataset utilised for evaluation contained only cracking defects, it is hypothesised that Res-Net 101 would be favourable in multiclass object detection and other tasks of increased complexity due to its added network depth.

4.5.3. Network Hyperparameter Optimisation Findings

Following the evaluation of CNNs discussed above, several investigations were undertaken to determine the optimal network training hyperparameters. Chapter two reviewed training parameters and their associated influence on the network training process; from this, two primary hyperparameters were identified for optimisation through an iterative analysis, namely ‘MiniBatchSize’ and ‘MaxEpochs’. These hyperparameters were selected for evaluation as the literature review undertaken indicated that optimal values vary significantly between applications and cannot be inferred from general research. Analysis of the hyperparameters was completed on each of the three CNNs discussed in Section 4.5.2, as multi-gpu functional of the DGX station allowed this to be completed within the required research timeline.

The first hyperparameter analysed was the ‘MiniBatchSize’, which was evaluated for values 4, 8, 16, 32 and 64. ‘MiniBatchSize’ refers to the size of the data sample the network uses to calculate and adjust weightings; generally, a large ‘MiniBatchSize’ performs faster with reduced accuracy, and a smaller value performs slower with increased accuracy. The results from this investigation are restated in Table 4.9:

Average Precision (AP) for various ‘MiniBatchSize’					
CNN	‘MiniBatchSize’				
	4	8	16	32	64
Res-Net 50	80.3%	80.3%	81.0%	77.5%	79.8%
Res-Net 101	79.3%	81.8%	82.5%	85.8%	86.0%
Darknet-19	74.8%	75.5%	82.3%	82.8%	84.3%

Table 4.9: AP results for various ‘MiniBatchSize’

The results of this analysis show that a higher value for the ‘MiniBatchSize’ hyperparameter will generally promote increased model performance, except for the Res-Net 50 network where a value of 16 was favourable. Values larger than 64 were not included the evaluation due to the time constraints associated with this research; notwithstanding, it is noted that a high value (i.e. 128, 256) requires very high computational capability, and cannot be utilised on most common computers for this application. The workstation computer (RTX3080 GPU) utilised in this research experienced unreliability in the training process with a ‘MiniBatchSize’ of 32 or larger for the three CNNs, meaning there may be instances where a value of 16 is appropriate when used in practical applications. All variations of the model demonstrated strong performance with a ‘MiniBatchSize’ of 16, meaning it is a suitable value for this hyperparameter when limited computational capability is available. This is particularly valuable when assessing the benefit of this research from an industry perspective, as access to high performance computing equipment varies significantly. In reviewing the results of this analysis, it was determined that a ‘MiniBatchSize’ of 32 was preferred for the smart sewer detection model, as it provides balance between performance and computational cost.

Following analysis of the ‘MiniBatchSize’ hyperparameter, the ‘MaxEpochs’ was investigated; this was completed on the DGX station by implementing the same methodology discussed above. The analysis on the number of training epochs indicated a general trend to the higher values of 30-50, with the optimal value for each CNN within this range. From the results of the investigation, it can be inferred that negligible performance increase, if any, would be derived from a ‘MaxEpochs’ value beyond 50. Higher values are also likely to promote overfitting of the model, meaning the models ability to generalise would be reduced, impacting its performance in a practical application. Based on the findings of the investigation, it was determined that a value of 40 was preferred for the ‘MaxEpochs’ hyperparameter. This value intends to balance model performance and overfitting, whilst still allowing contingency for datasets with complexity greater than that utilised in this research.

4.5.4. Multiclass Object Detection

Multiclass object detection was not the primary focus of this research due to the additional time associated with data annotation and model optimisation for multiple defect classes being beyond the forecast time available. Notwithstanding, the ability of the model to perform in a multiclass application is critical to its value in practical application and will be essential in achieving industry utilisation of this technology. Following development of the smart sewer detection model discussed above, it was determined that there was availability to investigate the model's performance on a multiclass dataset, containing three different types of pipeline defects.

Application of the smart sewer detection model to the multiclass dataset yielded strong results, with the model achieving a mAP of 89.3% and detection speed of 46.7fps. This aligns with the hypothesis discussed in Section 3.2.1.1, where model performance on other defect classes was anticipated to be better than cracking defects due to their complexity and variability. The results achieved reflect high recall, precision, and detection speed, appropriate for use in practical application. An example of an image output from the smart sewer detection model is presented below in Figure 4.16:

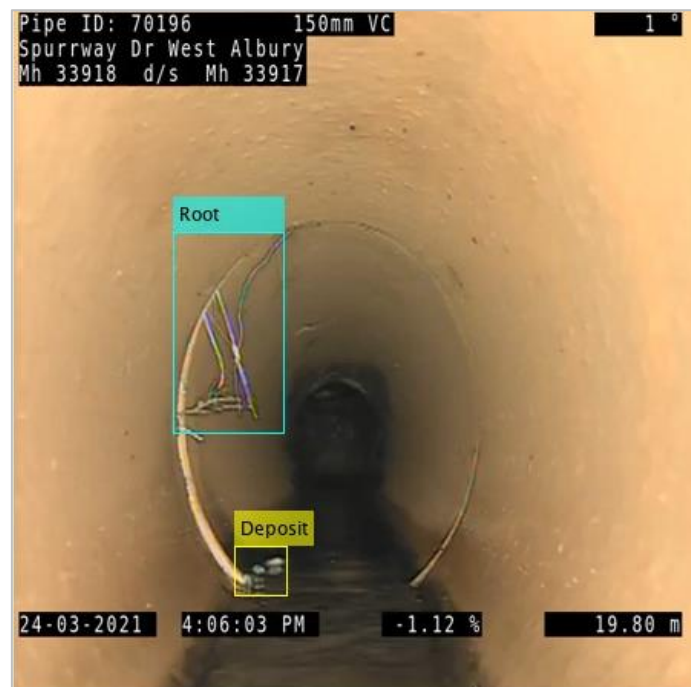


Figure 4.16: Example of multiclass detection

Overall, the results achieved on the multiclass dataset indicate that the model is robust and able to adapt to new datasets and defects well. This provides strong indication that the model would exhibit strong performance if implemented on data obtained from a completely different

municipality, where minor differences in background data and defect behaviour may exist. To further investigate the model's ability to generalise, raw data from videos not included in the training, validation or test datasets was input to the model and visually inspected; the model's ability to detect defects within this data was generally consistent with its performance on the test dataset.

4.5.1. Limitations of Research

Due to the timeframe associated with completion of this dissertation, several limitations within this research exist. The application of object detection technologies to sewer pipeline faults or any related industry task requires extensive time allocation for data annotation and model development; considering this, the scope of research primarily focused on cracking defects within sewer pipelines to align with the required timeline. Whilst later application of the model to multiclass object detection was highly successful, it is possible that further optimisation of the model could occur. Specifically, a multiclass dataset should be utilised to undertake the evaluation of CNNs and training hyperparameters completed in this research. Additionally, it is noted that further increase in the size of the dataset would likely yield further improvements to model performance.

The dataset developed in this research is primarily derived from vitrified clay (VC) sewer pipelines, due to the asset inspection program from which this data was collected including predominantly VC pipelines. While this is not considered a significant limitation, it is important that a diverse range of material types are utilised when training the smart sewer detection model for industry application, as to alleviate any bias that may exist.

Whilst this research provides meaningful contribution to industry, practical implementation would require a user-friendly graphical interface to be developed based on this proposed model. Development of a graphical user interface (GUI) would allow the model to be utilised by any person(s) involved with management of sewer assets. Development of a GUI is discussed further in Section 5.3.1.

5. Conclusion

5.1. Research Outcomes

Consistent with the aims and objectives of this dissertation, the results demonstrate that a successful object detection model has been developed for sewer pipeline defects, utilising computer vision technology and deep transfer learning. The smart sewer detection model is capable of accurate and precise location of sewer pipeline faults at a speed exceeding the threshold of real-time capability.

Raw CCTV inspection data was obtained from industry to develop a comprehensive dataset utilised for model training, validation, and testing. Two primary datasets were created for this research, one which contained only cracking defects, and a second dataset containing three defects for multiclass analysis. In combination, the two datasets contain over 7400 annotated images that were extracted from the CCTV data after annotation. Annotation of the raw video data was completed using the MATLAB Video Labeller App; a semi-autonomous data annotation tool.

The smart sewer detection model is an object detection model that leverages deep transfer learning capabilities. The model is based on the YOLOv2 object detection framework, with the Res-Net 101 CNN adapted as the feature extraction network. The research included a comprehensive analysis of twelve CNNs to determine the most appropriate, maximising the performance of the detection model for this application. The smart sewer detection model achieved initial performance metrics of 81.8% for average precision, and 30.2fps for detection speed; this was further refined through training optimisation.

Performance of the smart sewer detection model was refined by optimising training hyperparameters through an iterative analysis. The key hyperparameters identified through the literature review were 'MiniBatchSize' and 'MaxEpochs'; the optimal values for each were

found to be heavily dependent on the specific task. The iterative analysis was completed using the DGX Station at the University of Southern Queensland; this facilitated concurrent trials due to its multi-gpu capability. The analysis found that values of 32 for 'MiniBatchSize', and 40 for 'MaxEpochs' were preferred by the model for this application. Optimisation of these training parameters provided an increase in average precision of 3.0% over the results obtained with preliminary hyperparameters.

Analysis of the smart sewer detection model was completed on a multiclass dataset as an extension to the investigation outlined above. The multiclass dataset contained 'cracking', 'deposit' and root 'intrusion' defects; the model achieved a mAP of 89.3% and a detection speed of 46.7fps.

Overall, the smart sewer detection model developed in this research aligns wholly with the project aim and objectives, providing valuable prospect of industry application that would deliver direct economic and qualitative benefit to users.

5.2. Benefit to Industry

Sewer pipelines are essential infrastructure that assist in maintaining public health, and like all civil infrastructure assets, require ongoing maintenance and renewal. As detailed throughout this dissertation, the inspection and data review process associated with renewal of sewer infrastructure is extremely tedious, requiring extensive time allocation to review and categorise data.

The smart sewer detection model developed in this research demonstrates the ability to autonomously detect, locate and classify faults with CCTV inspection data, with high accuracy and detection speed. The ability of this model has potential to supplement and/or replace the human resources currently required to evaluate sewer inspection data. Further, the model is not subject to fatigue or common human errors, meaning it provides a consistent, systematic analysis of data. Considering this, the smart sewer detection model has potential to provide direct economic benefit to industry by reducing the required human input, and further, by improving the decision-making process through a systematic approach.

5.3. Future Research

Several areas of future research have been identified in completing this dissertation, including improving existing limitations, improved dataset scale and quality, field deployment, development of a GUI and automated rehabilitation suggestions.

Improving the limitations of research outlined in Section 4.5.1 is of foremost importance, as it will provide further improvements to the object detection model before application to other future works discussed below. The primary limitation to be addressed is the basis of developing the model with focus on one defect class; it is recommended to test the three CNNs again utilising an improved dataset with all relevant pipeline defects required by industry. Completing this will ensure optimisation of the employed model is maximised.

5.3.1. Field Deployment

Industry discussion within this dissertation primarily relates to supplementing the data review process that follows after inspection of sewer mains is complete. While significant potential for application of this technology exists in that area, benefit may also exist in field deployment. Use of this technology in a field application would allow operational staff to autonomously assess pipeline condition in real-time whilst inspecting. It is suggested that a device such as a Nvidia Jetson Nano may be appropriate for this application, however it is suggested that the network is pretrained on a computer with higher computational capability.

5.3.2. Graphical User Interface

For technology to be adopted by modern industry, it is essential that it is intuitive and user friendly. Whilst this research focuses on the development of the object detection model framework, it is known that a supporting GUI is required for the model to be practical to industry. It is envisaged that the GUI would interpret an .CSV file or similar from the detection model and reassociate it with the raw CCTV data for ease of interpretation. Development of this GUI would see the smart sewer detection model form an end-to-end solution for industry, that entirely alleviates the requirement for human assessment of the inspection data.

5.3.3. Suggested Repair Methodology

As a means of further extension to the proposed GUI discussed above, automation of suggested rehabilitation techniques based on the model's detection results is proposed. This would further expand the benefit to industry provided by this research. The post-processing component of the program would derive a suggested rehabilitation method based on an assigned defect output. It is suggested that this would be completed in accordance with the literature review, by applying the appropriate rehabilitation methods for each nominated defect. The detected fault type could be assigned a suggested rehabilitation method through a look-up table or similar means; a generalised example of this relationship is provided below:

Defect	Rehabilitation Methodology
Cracking	Insitu Relining
Hole	Insitu Relining
Collapse	Pipe Bursting
Void	Pipe Bursting
Root Ingress	Root cutter and Heavy Clean
Deposit	Heavy Clean

Table 5.1: Example relationship between defect and rehabilitation method.

Bibliography

Adhikari, B., and Huttunen, H., 2021. Iterative bounding box annotation for object detection. 2020 25th International Conference on Pattern Recognition (ICPR).

Afaq, S., and Rao, S., 2020. Significance Of Epochs On Training A Neural Network. INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH , 9(6).

Anon, 2017, *The Various Methods Used in Pipe Inspection*, [online] PIPEWORKS Inc, viewed 18 October 2021, <<https://www.pipeworksinc.com/blog/drain-and-sewer-service/methods-used-pipe-inspection>>.

Anon, 2020, *Deep Learning*, [online] Ibm.com, viewed 19 October 2021, <<https://www.ibm.com/cloud/learn/deep-learning#toc-how-deep-l-vLJwLmX4>>.

Albury City Council, 2019. *Council Meeting Agenda 27 May 2019*. Albury: AlburyCity, pp.94-98.

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., and Farhan, L., 2021. Review of Deep Learning: Concepts, CNN Architectures, challenges, applications, Future Directions. *Journal of Big Data*, 8(1).

Arm Ltd., n.d. What is Computer Vision. [online] Arm, viewed 1 October 2022, <<https://www.arm.com/glossary/computer-vision#:~:text=Different%20types%20of%20computer%20vision,image%20classification%20and%20feature%20matching.>>>.

Balas, V., Roy, S., Sharma, D. and Samui, P., 2019, *Handbook of Deep Learning Applications, Smart Innovation, Systems and Technologies*.

Bharadi, D., Naimuddin Panchbhai, M., Irfan Mukadam, A. and Narayan Rode, N., 2017. Image Classification Using Deep Learning. *International Journal of Engineering Research & Technology (IJERT)*, [online] 6(11), viewed 18 May 2022, <<https://www.ijert.org/research/image-classification-using-deep-learning-IJERTV6IS110016.pdf>>.

BMC Corp, 2022. *Cured-in-Place Pipe Point Repair*. [online] BMC Corp, <<https://pipejetter.com/cipp-point>>

do Ceu Almeida, M., Covas, D. and Maria Garcia Beceiro, P., 2015, *Rehabilitation of sewers and manholes: technologies and operational practices*.

El Naqa, I. and Murphy, M., 2015. What Is Machine Learning?. *Machine Learning in Radiation Oncology*, [online] pp.3-11, viewed 22 May 2022.
<https://link.springer.com/chapter/10.1007/978-3-319-18305-3_1#citeas>.

Fenner, R., 2000, Approaches to sewer maintenance: a review, *Urban Water*, 2(4), pp.343-356.

Gad, A.F., 2021. Evaluating Object Detection Models Using Mean Average Precision (mAP). [online] Paperspace Blog, viewed 8 October 2022, <<https://blog.paperspace.com/mean-average-precision/>>.

Gandhi, R., 2018. R-CNN, fast R-CNN, Faster R-CNN, YOLO - object detection algorithms. [online] Medium, viewed 3 October 2022 <<https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>>.

Gayhardt, L., Gronlund, C., Quintanilla, L., Withee, K., PRMerger6, Coulter, D., Martens, J., JulieMSFT and Lu, P., 2022. *Deep learning vs. machine learning in Azure Machine Learning*. [online] Microsoft, viewed 22 May 2022, <<https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning>>.

Girshick, R., Donahue, J., Darrell, T., and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. MIT Press.

Guyon, I. and Elisseeff, A., 2008. An Introduction to Feature Extraction. *Feature Extraction*, [online] pp.1-25, viewed 22 May 2022 <https://link.springer.com/chapter/10.1007/978-3-540-35488-8_1>.

Hamishebahr, Y., Guan, H., So, S., and Jo, J., 2022. A comprehensive review of deep learning-based crack detection approaches. *Applied Sciences*, 12(3), p.1374.

Han, J., Fang, P., Li, W., Hong, J., Ali Armin, M., Reid, I., Petersson, L., and Li, H., 2022. You Only Cut Once: Boosting Data Augmentation with a Single Cut. *Proceedings of the 39th International Conference on Machine Learning*.

Hassan, S., Dang, L., Mehmood, I., Im, S., Choi, C., Kang, J., Park, Y. and Moon, H., 2019, Underground sewer pipe condition assessment based on convolutional neural networks, *Automation in Construction*, 106, p.102849.

He, K., Zhang, X., Ren, S., and Sun, J., 2016. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Hui, J., 2019. Map (mean average precision) for object detection. [online] Medium, , viewed 6 October 2022, <<https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>>.

Hui, J., 2019. Object detection: Speed and accuracy comparison (faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3). [online] Medium, viewed 1 October 2022, <<https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359>>.

Hulstaert, L., 2018. Object detection tutorial with SSD & Faster RCNN. [online] DataCamp, viewed 11 October 2022, <<https://www.datacamp.com/tutorial/object-detection-guide>>.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., and Keutzer, K., 2016. Squeezenet: Alexnet-Level Accuracy With 50x Fewer Parameters And <0.5mb Model Size. Iclr 2017.

IBM Cloud Education, 2021. What is overfitting? [online] IBM, viewed 11 October 2022, <<https://www.ibm.com/cloud/learn/overfitting#toc-how-to-det-Aqv1nwvv>>.

IBM Cloud Education, 2021. What is underfitting? [online] IBM, , viewed 12 October 2022, <<https://www.ibm.com/cloud/learn/underfitting>>.

Ishmuratov, R., Orlov, V. and Andrianov, A., 2013. The spiral wound pipeline rehabilitation technique for pipe networks: an application and experience in Moscow city. *International No-Dig*, [online] 31, pp.1-7, viewed 21 May 2022, <https://www.researchgate.net/publication/267452568_THE_SPIRAL_WOUND_PIPELINE_REHABILITATION_TECHNIQUE_FOR_PIPE_NETWORKS_AN_APPLICATION_AND_EXPERIENCE_IN_MOSCOW_CITY>

Jabbar, H.K., and Khan, R.Z., 2014. Methods to avoid over-fitting and under-fitting in Supervised Machine Learning (Comparative Study). Computer Science, Communication and Instrumentation Devices.

Kandel, I., and Castelli, M., 2020. The effect of batch size on the generalizability of the Convolutional Neural Networks on a histopathology dataset. ICT Express, 6(4), pp.312–315.

Koech, K.E. (2022) On object detection metrics with worked example, Medium. Towards Data Science, viewed 6 October 2022, <<https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e>>.

Lamba, H., 2019. Understanding semantic segmentation with UNET. [online] Medium, viewed 1 October 2022, <<https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>>.

Lampola, T. and Kuikka, S., 2018, Inspection methods for sewer pipes.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A.C., 2016. SSD: Single shot multibox detector. Computer Vision – ECCV 2016, pp.21–37.

Liu, Z., Chen, Z., Li, Z., and Hu, W., 2018. An efficient pedestrian detection method based on yolov2. Mathematical Problems in Engineering, 2018, pp.1–10.

Li, Y., Wang, H., Dang, L.M., Song, H.-K., and Moon, H., 2022. Vision-based defect inspection and condition assessment for Sewer Pipes: A comprehensive survey. Sensors, 22(7), p.2722.

Long, X., Du, M., and Hu, X., 2019. Exploring Computer Vision in Deep Learning: Object Detection and Semantic Segmentation. Cary, NC.

LT, Z., 2022. Essential things you need to know about F1-score. [online] Medium, viewed 6 October 2022, <<https://towardsdatascience.com/essential-things-you-need-to-know-about-f1-score-dbd973bf1a3#:~:text=F1%2Dscore%20is%20one%20of,competing%20metrics%20%E2%80%94%20precision%20and%20recall.>>.

Manoj krishna, M., Neelima, M., Harshali, M. and Venu Gopala Rao, M., 2018. Image classification using Deep learning. International Journal of Engineering & Technology, 7(2.7), p.614.

Marr, B., 2019. 7 amazing examples of computer and machine vision in practice. [online] Forbes, viewed 1 October 2022, <<https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=5ab74e421018>>.

MathWorks, 2022. *AlexNet convolutional neural network*. [online] MathWorks, viewed 23 May 2022, <<https://au.mathworks.com/help/deeplearning/ref/alexnet.html>>.

MathWorks, n.d. Create Automation Algorithm for Labeling. [online] MATLAB & Simulink - MathWorks Australia, viewed 8 October 2022, <<https://au.mathworks.com/help/vision/ug/create-automation-algorithm-for-labeling.html>>.

MathWorks, 2021, *Deep Learning Toolbox*, [online] MathWorks, viewed 19 October 2021, <<https://au.mathworks.com/products/deep-learning.html>>

MathWorks, 2022. *Get Started with Image Preprocessing and Augmentation for Deep Learning*. [online] MathWorks, viewed 22 May 2022, <<https://au.mathworks.com/help/images/get-started-with-image-preprocessing-and-augmentation-for-deep-learning.html>>.

MathWorks, n.d. Get Started with the Video Labeler. [online] Get Started with the Video Labeler - MATLAB & Simulink - MathWorks Australia, viewed 8 October 2022, <<https://au.mathworks.com/help/vision/ug/get-started-with-the-video-labeler.html>>.

MathWorks, n.d. Ground Truth Labeler. [online] MATLAB & Simulink - MathWorks Australia, viewed 8 October 2022, <<https://au.mathworks.com/help/driving/ref/groundtruthlabeler-app.html>>.

MathWorks, n.d.. peopleDetectorACF. [online] MATLAB & Simulink - MathWorks Australia, viewed 8 October 2022, <<https://au.mathworks.com/help/vision/ref/peopledetectoracf.html>>.

MathWorks, 2022. *Preprocess Images for Deep Learning*. [online] MathWorks, viewed 22 May 2022, <<https://au.mathworks.com/help/deeplearning/ug/preprocess-images-for-deep-learning.html#:~:text=Preprocess%20Images%20for%20Deep%20Learning%20To%20train%20a,or%20crop%20your%20data%20to%20the%20required%20size.>>>.

MathWorks, n.d. Object detection using Yolo V2 Deep Learning. [online] MathWorks Australia, viewed 29 September 2022, <<https://au.mathworks.com/help/deeplearning/ug/object-detection-using-yolo-v2.html>>.

MathWorks, n.d.. vision.PointTracker. [online] MATLAB & Simulink - MathWorks Australia, viewed 8 October 2022, <<https://au.mathworks.com/help/vision/ref/vision.pointtracker-system-object.html>>.

May, A., 2021. *Is deep learning supervised or unsupervised?*. [online] AIMed, viewed 22 May 2022, <<https://ai-med.io/ac-observations/is-deep-learning-supervised-or-unsupervised/#:~:text=Therefore%2C%20deep%20learning%20can%20be%20supervised%2C%20unsupervised%2C%20semi-supervised%2C,mostly%20on%20how%20the%20neural%20network%20is%20used.>>>.

Moradi, S., 2020, *Defect Detection and Classification in Sewer Pipeline Inspection Videos Using Deep Neural Networks*, Ph.D., Concordia University.

Moradi, S., Zayed, T., and Golkhoo, F., 2019. Review on computer aided sewer pipeline defect detection and condition assessment. *Infrastructures*, 4(1), p.10.

O'Shea, K. and Nash, R., 2022. An Introduction to Convolutional Neural Networks. ArXiv e-prints.

Ongsulee, P., 2017. Artificial intelligence, machine learning and deep learning. 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE).

Oppermann, A., 2019. *What is Deep Learning and How does it work?*. [online] Toward Data Science Incorporated, viewed 22 May 2022, <<https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac>>.

Padilla, R., Netto, S.L., and da Silva, E.A., 2020. A survey on performance metrics for object-detection algorithms. 2020 International Conference on Systems, Signals and Image Processing (IWSSIP).

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

Park, S., 2021. A 2021 guide to improving cnn-optimizers: Adam vs sgd. [online] Medium, viewed 3 October 2022, <<https://medium.com/geekculture/a-2021-guide-to-improving-cnn-optimizers-adam-vs-sgd-495848ac6008>>.

Park, T., 2009, *A COMPREHENSIVE ASSET MANAGEMENT SYSTEM FOR SEWER INFRASTRUCTURES*, Ph.D., The Pennsylvania State University.

- Parthasarathy, D., 2017. A brief history of cnns in image segmentation: From R-CNN to mask R-CNN. [online] Medium, viewed 3 October 2022, <<https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4>>.
- Pathak, A.R., Pandey, M., and Rautaray, S., 2018. Application of deep learning for object detection. *Procedia Computer Science*, 132, pp.1706–1717.
- Pauly, L., Hogg, D., Fuentes, R. and Peel, H., 2017, Deeper Networks for Pavement Crack Detection, pp.479-485.
- Pedamkar, P., n.d. *Applications of Machine Learning*. [online] EDUCBA, viewed 22 May 2022, <<https://www.educba.com/applications-of-machine-learning/>>.
- Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., and Sun, J., 2018. Megdet: A large Mini-Batch Object Detector. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.
- Pokhrel, S., 2019. Beginners guide to understanding Convolutional Neural Networks. [online] Medium, viewed 1 October 2022, <<https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d>>.
- Population Australia, 2022. *Albury Population 2022*. [online] Population Australia, viewed 24 May 2022, <<https://www.population.net.au/albury-population/>>.
- Radhakrishnan, P., 2017. What are hyperparameters ? and how to tune the hyperparameters in a deep neural network? [online] Medium, viewed 3 October 2022, <<https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a>>.
- Rameil, M. ed., 2007. *Handbook of Pipe Bursting Practice*. Vulkan-Verlag GmbH.
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., 2016. You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- Redmon, J., and Farhadi, A., 2018. YOLOv3: An Incremental Improvement.
- Redmon, J., and Farhadi, A., 2017. Yolo9000: Better, faster, stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ren, S., He, K., Girshick, R., and Sun, J., 2017. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), pp.1137–1149.

Reyes, E., Gómez, C., Norambuena, E., and Ruiz-del-Solar, J., 2019. Near real-time object recognition for pepper based on deep neural networks running on a backpack. *RoboCup 2018: Robot World Cup XXII*, pp.287–298.

Rice, L., Wong, E. and Kolter, Z., 2020, November. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning* (pp. 8093-8104). PMLR.

Rosebrock, A., 2021. Convolutional Neural Networks (cnns) and layer types. [online] PyImageSearch, viewed 1 October 2022, <<https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/>>.

Rouault, P., Hernandez, N., Torres, A., Werey, C., Rulleau, B. and Clemens, F., 2019. Sewer asset management – state of the art and research needs. *Urban Water Journal*, 16(9), pp.662-675.

Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Ruizendaal, R., 2017. Deep learning #3: More on CNNs & handling overfitting. [online] Medium, viewed 11 October 2022, <<https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d>>.

Sanghvirajit, 2021. A complete guide to adam and RMSPROP optimizer. [online] Medium, viewed 3 October 2022, <<https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be>>.

Schreiner, C., Torkkola, K., Gardner, M., and Zhang, K., 2006. Using machine learning techniques to reduce data annotation time. *PsycEXTRA Dataset*.

Shah, T., 2017. About train, validation and test sets in machine learning. [online] Towards Data Science, viewed 29 September 2022, <<https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>>.

Sharpe, K., 2021. Research Proposal - Automated identification and analysis of sewer main faults utilising deep learning, in Albury City Council, NSW, Australia.. Undergraduate. University of Southern Queensland.

Shorten, C., and Khoshgoftaar, T.M., 2019. A survey on image data augmentation for Deep Learning. *Journal of Big Data*, 6(1).

Singla, S., 2020. Why is batch normalization useful in deep neural network? [online] Medium, viewed 1 October 2022, <<https://towardsdatascience.com/batch-normalisation-in-deep-neural-network-ce65dd9e8dbf>>.

Solawetz, J., 2020. Yolov4 - ten tactics to build a better model. [online] Roboflow Blog, viewed 27 September 2022, <<https://blog.roboflow.com/yolov4-tactics/>>.

Sonogashira, M., Shonai, M., and Iiyama, M., 2020. High-resolution bathymetry by deep-learning-based image superresolution. *PLOS ONE*, 15(7).

Stewart, M., 2020. Simple guide to hyperparameter tuning in Neural Networks. [online] Medium, viewed 3 October 2022, <<https://towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594>>.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A., 2017. Inception-V4, inception-resnet and the impact of residual connections on learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., 2015. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tagherouit, W., Bennis, S. and Bengassem, J., 2011. A Fuzzy Expert System for Prioritizing Rehabilitation of Sewer Networks. *Computer-Aided Civil and Infrastructure Engineering*, 26(2), pp.146-152.

Talukdar, J., Gupta, S., Rajpura, P.S., and Hegde, R.S., 2018. Transfer learning for object detection using state-of-the-art deep neural networks. *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*.

Teng, S., Liu, Z., Chen, G., and Cheng, L., 2021. Concrete crack detection based on well-known feature extractor model and the yolo_v2 network. *Applied Sciences*, 11(2), p.813.

Thoma, M., 2016. *A Survey of Semantic Segmentation*.

Tscheikner-Gratl, F., Caradot, N., Cherqui, F., Leitão, J., Ahmadi, M., Langeveld, J., Le Gat, Y., Scholten, L., Roghani, B., Rodríguez, J., Lepot, M., Stegeman, B., Heinrichsen, A., Kropp, I., Kerres, K., Almeida, M., Bach, P., Moy de Vitry, M., Sá Marques, A., Simões, N.,

Ullah, A., Elahi, H., Sun, Z., Khatoon, A., and Ahmad, I., 2021. Comparative analysis of Alexnet, Resnet18 and squeezenet with diverse modification and arduous implementation. *Arabian Journal for Science and Engineering*, 47(2), pp.2397–2417.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y.M., 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.

Wang, M., and Cheng, J.C.P., 2019. Semantic segmentation of sewer pipe defects using deep dilated Convolutional Neural Network. *Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC)*.

Weiner, R. and Matthews, R., 2003. *Environmental Engineering*. [online], viewed 19 May 2022, <<https://www.sciencedirect.com/book/9780750672948/environmental-engineering#book-description>>.

Westaway, D., 2001. Cured-in-place pipe rehabilitates city sewers. *Reinforced Plastics*, [online] 45(12), pp.36-38, viewed 21 May 2022, <<https://www.sciencedirect.com/science/article/pii/S003436170180415X>>.

WinCan, 2022. *WinCan – Smarter Sewer Inspection*. [online] Wincan.com, viewed 21 May 2022, <<https://www.wincan.com/en/home/>>.

WIRAHADIKUSUMAH, R., ABRAHAM, D. and CASTELLO, J., 1999. Markov decision process for sewer rehabilitation, *Engineering, Construction and Architectural Management*, 6(4), pp.358-370.

Wolff, R., 2020, [online] What Is Training Data in Machine Learning?, viewed 19 October 2021, <<https://monkeylearn.com/blog/training-data/>>

Wu, M., Xie, W., Shi, X., Shao, P., and Shi, Z., 2018. Real-time drone detection using Deep Learning Approach. *Machine Learning and Intelligent Communications*, pp.22–32.

Xu, W., Li, B., Liu, S., and Qiu, W., 2018. Real-time object detection and semantic segmentation for autonomous driving. MIPPR 2017: Automatic Target Recognition and Navigation.

Xu, Z., Qian, S., Ran, X., and Zhou, J., 2021. Application of deep convolution neural network in crack identification. *Applied Artificial Intelligence*, 36(1).

Yang, M., Su, T., Pan, N. and Yang, Y., 2011, Systematic image quality assessment for sewer inspection, *Expert Systems with Applications*, 38(3), pp.1766-1776.

Yin, X., Chen, Y., Bouferguene, A., Zaman, H., Al-Hussein, M., and Kurach, L., 2020. A deep learning-based framework for an automated defect detection system for Sewer Pipes. *Automation in Construction*, 109, p.102967.

Appendix

5.4. Appendix 1 – Project Specification

ENG4111/4112 Research Project

Project Specification

For: Kyal Sharpe

Title: Development of a smart sewer pipeline fault detection method using CCTV inspection data and deep learning

Major: Civil Engineering

Supervisors: Andy Nguyen

Enrollment: ENG4111 – EXT S1, 2022

ENG4112 – EXT S2, 2022

Project Aim: To develop an autonomous system that processes CCTV inspection data of sewer pipelines, identifies faults, and proposes a suitable repair method based on the fault identified.

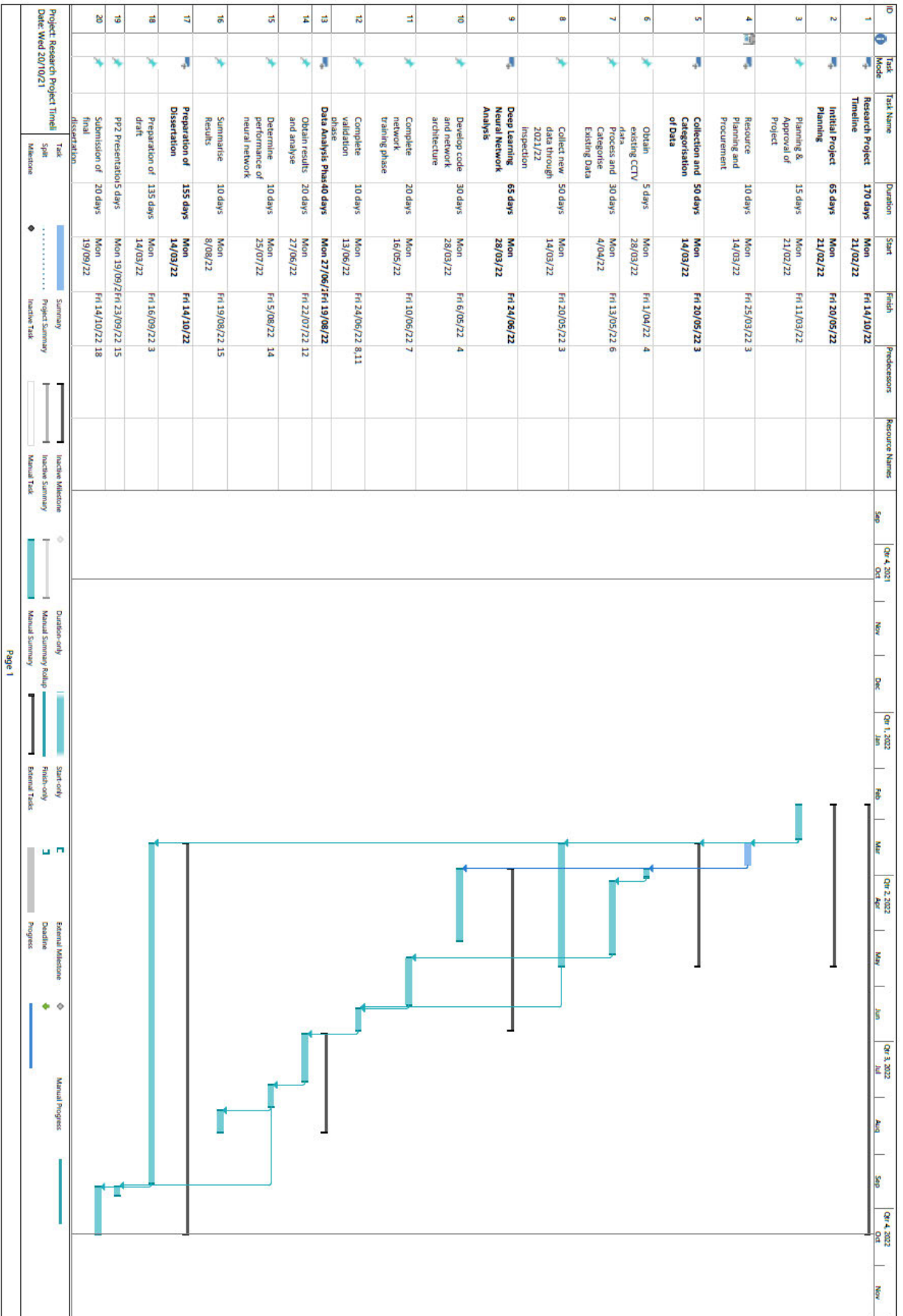
Programme: Version 1, 16th March 2022

1. Conduct research and expand upon my current literature review to determine/develop a strategy for the deep learning program.
2. Investigate and review data management to determine the most appropriate input format, including video sampling techniques.
3. Research and analyse current methods of sewer fault identification, including any industry standard fault categorization systems.
4. Begin developing deep learning prototype program based on previous points
5. Develop/classify a training dataset from CCTV inspection data, then attempt to train deep learning program on the training dataset.
6. Undertake test/validation phase with a new dataset to determine the effectiveness/efficiency of the program.
7. Compare the effectiveness/efficiency of the program with current industry techniques (i.e. manual review and recognition)
8. Prepare, develop, finalise and submit thesis.

If time and resource permit:

9. Automatically determine a suitable repair method based on the fault classification obtained.
10. Determine approximate cost estimation based on the fault type and repair method.

Gaunt Chart (March 2022)



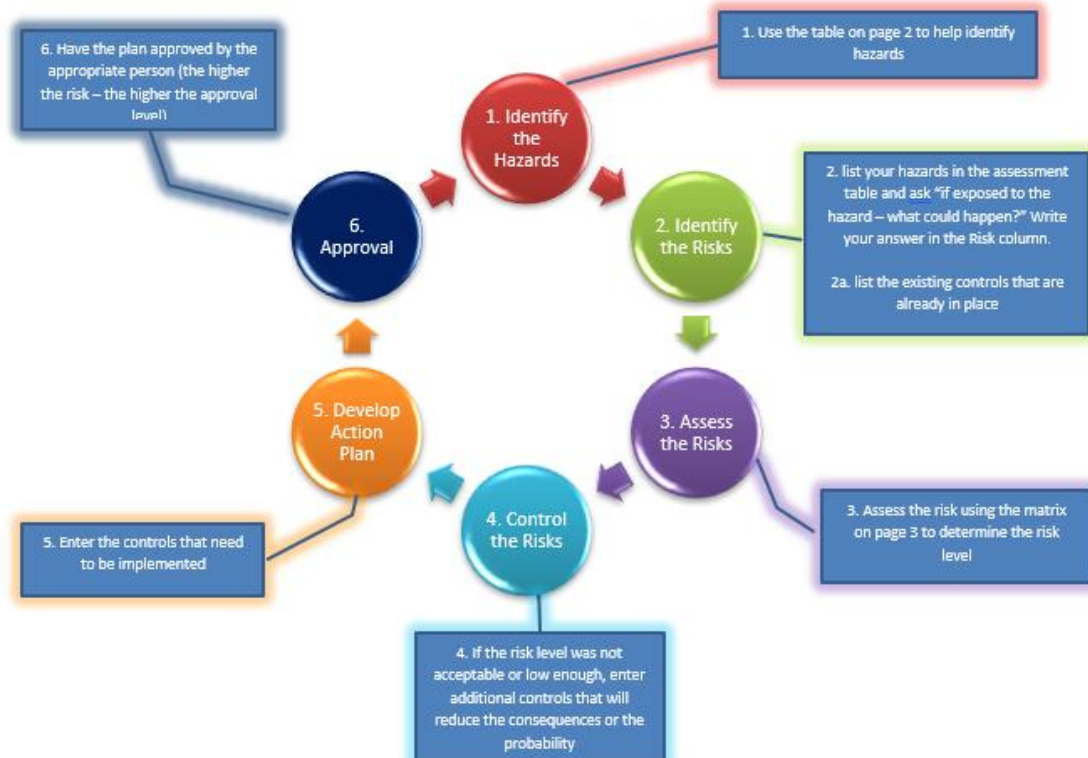
Appendix 2 – Risk Assessment



University of Southern Queensland Generic Risk Management Plan

Workplace (Division/Faculty/Section): School of Engineering		
Assessment No: ENG4111/2	Assessment Date: 13/10/2022	Review Date: (5 years maximum) 01/03/2022
Context: What is being assessed? Describe the item, job, process, work arrangement, event etc: This risk management plan relates to identification and management of risks associated with completion of the ENG4111/2 research project for student Kyal Sharpe (0061109786)		
Assessment Team – who is conducting the assessment?		
Assessor(s): Belal Yousif, Andy Nguyen		
Others consulted: (eg elected health and safety representative, other personnel exposed to risks) Nil		

The Risk Management Process

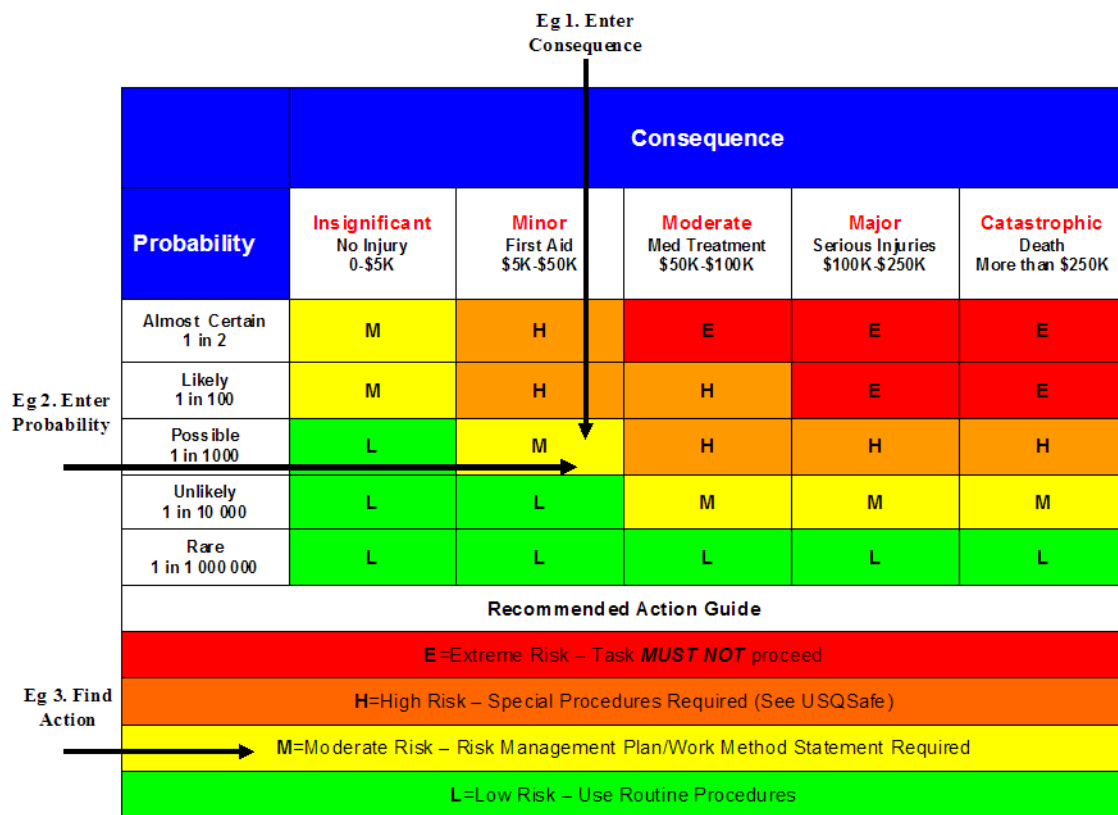


Step 1 - Identify the hazards (use this table to help identify hazards then list all hazards in the risk table)

General Work Environment		
<input type="checkbox"/> Sun exposure	<input type="checkbox"/> Water (creek, river, beach, dam)	<input type="checkbox"/> Sound / Noise
<input type="checkbox"/> Animals / Insects	<input type="checkbox"/> Storms / Weather/Wind/Lightning	<input type="checkbox"/> Temperature (heat, cold)
<input type="checkbox"/> Air Quality	<input type="checkbox"/> Lighting	<input type="checkbox"/> Uneven Walking Surface
<input type="checkbox"/> Trip Hazards	<input type="checkbox"/> Confined Spaces	<input type="checkbox"/> Restricted access/egress
<input type="checkbox"/> Pressure (Diving/Altitude)	<input type="checkbox"/> Smoke	<input type="checkbox"/>
Other/Details:		
Machinery, Plant and Equipment		
<input type="checkbox"/> Machinery (fixed plant)	<input type="checkbox"/> Machinery (portable)	<input type="checkbox"/> Hand tools
<input type="checkbox"/> Laser (Class 2 or above)	<input type="checkbox"/> Elevated work platforms	<input type="checkbox"/> Traffic Control
<input type="checkbox"/> Non-powered equipment	<input type="checkbox"/> Pressure Vessel	<input type="checkbox"/> Electrical
<input type="checkbox"/> Vibration	<input type="checkbox"/> Moving Parts	<input type="checkbox"/> Acoustic/Noise
<input type="checkbox"/> Vehicles	<input type="checkbox"/> Trailers	<input type="checkbox"/> Hand tools
Other/Details:		
Manual Tasks / Ergonomics		
<input type="checkbox"/> Manual tasks (repetitive, heavy)	<input type="checkbox"/> Working at heights	<input type="checkbox"/> Restricted space
<input type="checkbox"/> Vibration	<input type="checkbox"/> Lifting Carrying	<input type="checkbox"/> Pushing/pulling
<input type="checkbox"/> Reaching/Overstretching	<input checked="" type="checkbox"/> Repetitive Movement	<input type="checkbox"/> Bending
<input checked="" type="checkbox"/> Eye strain	<input type="checkbox"/> Machinery (portable)	<input type="checkbox"/> Hand tools
Other/Details:		
Biological (e.g. hygiene, disease, infection)		
<input type="checkbox"/> Human tissue/fluids	<input type="checkbox"/> Virus / Disease	<input type="checkbox"/> Food handling
<input type="checkbox"/> Microbiological	<input type="checkbox"/> Animal tissue/fluids	<input type="checkbox"/> Allergenic
Other/Details:		
Chemicals Note: Refer to the label and Safety Data Sheet (SDS) for the classification and management of all chemicals.		
<input type="checkbox"/> Non-hazardous chemical(s)	<input type="checkbox"/> 'Hazardous' chemical (Refer to a completed hazardous chemical risk assessment)	
<input type="checkbox"/> Engineered nanoparticles	<input type="checkbox"/> Explosives	<input type="checkbox"/> Gas Cylinders
Name of chemical(s) / Details:		
Critical Incident – resulting in:		
<input type="checkbox"/> Lockdown	<input type="checkbox"/> Evacuation	<input type="checkbox"/> Disruption
<input type="checkbox"/> Public Image/Adverse Media Issue	<input type="checkbox"/> Violence	<input type="checkbox"/> Environmental Issue
Other/Details:		
Radiation		
<input type="checkbox"/> Ionising radiation	<input type="checkbox"/> Ultraviolet (UV) radiation	<input type="checkbox"/> Radio frequency/microwave
<input type="checkbox"/> infrared (IR) radiation	<input type="checkbox"/> Laser (class 2 or above)	<input type="checkbox"/>
Other/Details:		
Energy Systems – incident / issues involving:		
<input type="checkbox"/> Electricity (incl. Mains and Solar)	<input type="checkbox"/> LPG Gas	<input type="checkbox"/> Gas / Pressurised containers
Other/Details:		
Facilities / Built Environment		
<input type="checkbox"/> Buildings and fixtures	<input type="checkbox"/> Driveway / Paths	<input type="checkbox"/> Workshops / Work rooms
<input type="checkbox"/> Playground equipment	<input type="checkbox"/> Furniture	<input type="checkbox"/> Swimming pool
Other/Details:		
People issues		
<input type="checkbox"/> Students	<input type="checkbox"/> Staff	<input type="checkbox"/> Visitors / Others
<input type="checkbox"/> Physical	<input checked="" type="checkbox"/> Psychological / Stress	<input type="checkbox"/> Contractors
<input type="checkbox"/> Fatigue	<input checked="" type="checkbox"/> Workload	<input type="checkbox"/> Organisational Change

<input type="checkbox"/> Workplace Violence/Bullying	<input type="checkbox"/> Inexperienced/new personnel	<input type="checkbox"/>
Other/Details:		
Step 1 (cont) Other Hazards / Details (enter other hazards not identified on the table)		
Project Proposal Rejection		
Resource Allocation Not Available		
Failure to achieve scheduled progress and submission deadline		

Risk Matrix



Risk register and Analysis

Step 1 (cont)	Step 2	Step 2a	Step 3			Step 4				
Hazards: From step 1 or more if identified	The Risk: What can happen if exposed to the hazard with existing controls in place?	Existing Controls: What are the existing controls that are already in place?	Risk Assessment: (use the Risk Matrix on p3) Consequence x Probability = Risk Level			Additional controls: Enter additional controls if required to reduce the risk level	Risk assessment with additional controls: (use the Risk Matrix on p3 – has the consequence or probability changed?)			Controls Implemented? Yes/No
			Consequence	Probability	Risk Level		Consequence	Probability	Risk Level	
Example										
Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	Regular breaks, chilled water available, loose clothing, fatigue management policy.	catastrophic	possible	high	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
Eye Strain	Short term or long term damage to vision, physical pain	Informal breaks from computer	Moderate	Possible	High	Formalise a schedule of 15 minute breaks and a longer break depending on study duration. Ensure hydraton and fresh air.	Moderate	Unlikely	Moderate	Yes
Repetitive Movement	Possible cramping, muscle strain and/or extended damage	Nil	Minor	Possible	Moderate	Formalise a schedule of 15 minute breaks, ensure stretching is completed.	Minor	Unlikely	Low	Yes
Stress	Mental stress and /or exhaustion due to deadlines and complexity of task	Study schedule	Moderate	Possible	High	Develop a formal study schedule and associated timeline, monitor throughout the duration and adjust when required in discussion with supervisor if necessary.	Moderate	Unlikely	Moderate	Yes

Step 1 (cont)	Step 2	Step 2a	Step 3			Step 4				
Hazards: From step 1 or more if identified	The Risk: What can happen if exposed to the hazard with existing controls in place?	Existing Controls: What are the existing controls that are already in place?	Risk Assessment: (use the Risk Matrix on p3) Consequence x Probability = Risk Level			Additional controls: Enter additional controls if required to reduce the risk level	Risk assessment with additional controls: (use the Risk Matrix on p3 – has the consequence or probability changed?)			Controls Implemented? Yes/No
Consequence	Probability	Risk Level	Consequence	Probability	Risk Level					
Example										
Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	Regular breaks, chilled water available, loose clothing, fatigue management policy.	catastrophic	possible	high	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
Workload	Fatigue and stress due to high workload completing research project amongst other subjects and full-time employment	Study schedule	Moderate	Possible	High	Develop a formal study schedule and associated timeline, monitor throughout the duration and adjust when required in discussion with supervisor if necessary.	Moderate	Unlikely	Moderate	Yes
Scheduled Progress	Failure to achieve required progress and deadlines due to time management, poor results or unforeseen circumstances	Supervision whilst completing project	Moderate	Unlikely	Moderate	Fortnightly meetings with supervisor	Moderate	Unlikely	Moderate	Yes

Step 5 – Action Plan (for controls not already in place)

Control Option	Resources	Person(s) responsible	Proposed implementation date
Formal study schedule w/ 15 breaks	Personal time management	Kyal Sharpe	01/03/2022
Develop formal study plan with timeline for the year	Personal time management	Kyal Sharpe	01/03/2022
Fortnightly supervisor meeting	Personal time, supervisor time	Kyal Sharpe, Dr Andy Nguyen	01/03/2022

Step 6 – Approval**Drafter's Comments:**

Nil

Drafter Details:

Name: Kyal Sharpe

Signature: Kyal Sharpe

Date: 01/03/2022

Assessment Approval: (Extreme or High = VC, Moderate = Cat 4 delegate or above, Low = Manager/Supervisor)

I am satisfied that the risks are as low as reasonably practicable and that the resources required will be provided.

Name: Dr Andy Nguyen

Signature: 

Date: 01/03/2022

Position Title: Senior Lecturer (Structural Engineering)

5.6. Appendix 3 – Smart Sewer Detection Model Code

Smart Sewer Detection Model (YOLOv2 Adapted)

Load Data Sources

```
%% Multiple Data Sources - Loads several Ground Truths and combine for
trainingData input

% Source 1
data1=load('VIDE01.mat');
gTruth1=data1.('gTruth');
trainingData1=objectDetectorTrainingData(gTruth1);

% Source 2
data2=load('VIDE02.mat');
gTruth2=data2.('gTruth');
trainingData2=objectDetectorTrainingData(gTruth2);

% Source 3
data3=load('VIDE03.mat');
gTruth3=data3.('gTruth');
trainingData3=objectDetectorTrainingData(gTruth3);

% Source 4
data4=load('VIDE04.mat');
gTruth4=data4.('gTruth');
trainingData4=objectDetectorTrainingData(gTruth4);

% Source 5
data5=load('VIDE05.mat');
gTruth5=data5.('gTruth');
trainingData5=objectDetectorTrainingData(gTruth5);

% Source 6
data6=load('VIDE06.mat');
gTruth6=data6.('gTruth');
trainingData6=objectDetectorTrainingData(gTruth6);

% Source 7
data7=load('VIDE07.mat');
gTruth7=data7.('gTruth');
trainingData7=objectDetectorTrainingData(gTruth7);

% Source 8
data8=load('VIDE08.mat');
gTruth8=data8.('gTruth');
trainingData8=objectDetectorTrainingData(gTruth8);

% Source 9
data9=load('VIDE09.mat');
gTruth9=data9.('gTruth');
trainingData9=objectDetectorTrainingData(gTruth9);

% Combine
u = transpose(table2array(trainingData1));
```

```

v = transpose(table2array(trainingData2));
w = transpose(table2array(trainingData3));
x = transpose(table2array(trainingData4));
y = transpose(table2array(trainingData5));
z = transpose(table2array(trainingData6));
a = transpose(table2array(trainingData7));
b = transpose(table2array(trainingData8));
c = transpose(table2array(trainingData9));

Combine_transpose = [a b c u v w x y z];
trainingDataCell = transpose(Combine_transpose);

```

Creating Training, Test & Validation Datasets

```

% Split the dataset into training, validation & test sets (70%, 10%, 20%)
rng(0);
shuffledIndices = randperm(height(trainingData));
idx = floor(0.7 * length(shuffledIndices) );

trainingIdx = 1:idx;
trainingDataTbl = trainingData(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = trainingData(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = trainingData(shuffledIndices(testIdx),:);

```

Create Datastores For Processing

```

% Create datastores for loading the image and label data during training &
% evaluation

imdsTrain = imageDatastore(trainingDataTbl{:, "imageFilename"});
bldsTrain = boxLabelDatastore(trainingDataTbl(:, 2:end));

imdsValidation = imageDatastore(validationDataTbl{:, "imageFilename"});
bldsValidation = boxLabelDatastore(validationDataTbl(:, 2:end));

imdsTest = imageDatastore(testDataTbl{:, "imageFilename"});
bldsTest = boxLabelDatastore(testDataTbl(:, 2:end));

% Combine image and box label datastores

trainingData = combine(imdsTrain, bldsTrain);
validationData = combine(imdsValidation, bldsValidation);
testData = combine(imdsTest, bldsTest);

```

Create Object Detection Network

```

%% Create YoloV2 Object Detector Network
inputSize = [448 448 3]; %Specify input size used for training
numClasses = 3;

% Resize data and nominate number of anchors
trainingDataForEstimation =
transform(trainingData, @(data) preprocessData(data, inputSize));
numAnchors = 9;
[anchorBoxes, meanIoU] =
estimateAnchorBoxes(trainingDataForEstimation, numAnchors);

```

```
% Create the Detector
```

```
featureExtractionNetwork = resnet101;
```

```
featureLayer = 'res4b22_relu';
```

```
lgraph =
```

```
yolov2Layers(inputSize,numClasses,anchorBoxes,featureExtractionNetwork,featureLayer);
```

Apply Data Augmentation Function

```
%% Data Augmentation
```

```
augmentedTrainingData = transform(trainingData,@augmentData);
```

```
augmentedData = cell(4,1);
```

```
for k = 1:4
```

```
    data = read(augmentedTrainingData);
```

```
    augmentedData{k} = insertShape(data{1}, "rectangle", data{2});
```

```
    reset(augmentedTrainingData);
```

```
end
```

Pre-Process Training Data

```
%% Preprocess Training Data
```

```
preprocessedTrainingData =
```

```
transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
```

```
preprocessedValidationData =
```

```
transform(validationData,@(data)preprocessData(data,inputSize));
```

Specify Training Options

```
%% Specify Training Options
```

```
options = trainingOptions("adam",...
```

```
    GradientDecayFactor=0.9,...
```

```
    SquaredGradientDecayFactor=0.999,...
```

```
    InitialLearnRate=0.0001,...
```

```
    LearnRateSchedule="none",...
```

```
    MiniBatchSize=16,...
```

```
    L2Regularization=0.0005,...
```

```
    MaxEpochs=40,... %Default value 70
```

```
    BatchNormalizationStatistics="moving",...
```

```
    DispatchInBackground=false,...
```

```
    ResetInputNormalization=false,...
```

```
    Shuffle="every-epoch",... %usually 'every-epoch'
```

```
    VerboseFrequency=20,...
```

```
    CheckpointPath=tempdir,...
```

```
    CheckpointFrequency=5,...
```

```
    ValidationData=preprocessedValidationData, ...
```

```
    ExecutionEnvironment="gpu");
```

Train YOLOv2 Object Detector or Load Trained Detector

```
%% Train YOLO v2 Object Detector
```

```
doTraining = true;
```

```
if doTraining
```

```
    % Train the YOLO v2 detector.
```

```
    [detector,info] =
```

```
trainYOLOv2ObjectDetector(preprocessedTrainingData,lgraph,options);
```

```
    TrainedDetector = detector;
```



```

    save 'TrainedDetector'; % Saves trained YOLO v2 detector to filepath
else
    % Load pretrained detector from the file path
    pretrained = load('TrainedDetector.mat');
    detector = pretrained.TrainedDetector;

end

```

Evaluate Detector Using Test Dataset

```

%% Evaluate Detector Using Test Set
tic; % Start Timer

preprocessedTestData =
transform(testData,@(data)preprocessData(data,inputSize));
detectionResults = detect(detector, preprocessedTestData, "MiniBatchSize" ,64);

newt=toc; % Stop Timer
fps=height(testDataTbl)./newt; % Calculate detection FPS (testdata/detection
time)

[ap,recall,precision] = evaluateDetectionPrecision(detectionResults,
preprocessedTestData);

classID=1;
figure
plot(recall{classID},precision{classID})
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision (Crack) = %.2f',ap(classID)))

classID=2;
figure
plot(recall{classID},precision{classID})
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision (Deposit) = %.2f',ap(classID)))

classID=3;
figure
plot(recall{classID},precision{classID})
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision (Root) = %.2f',ap(classID)))

```

Detect Specified Image

```

%% Test Image
I = imread("IMAGE.png");
I = imresize(I,inputSize(1:2));
[bboxes,scores,labels] = detect(detector,I);

I = insertObjectAnnotation(I,'rectangle',bboxes,labels,Color=["cyan","yellow"]);
figure
imshow(I)

```

Helper Functions

```
function data = preprocessData(data,inputSize)
% Resize the images and scale the pixels to between 0 and 1. Also scale the
% corresponding bounding boxes.

for ii = 1:size(data,1)
    I = data{ii,1};
    imgSize = size(I);

    bboxes = data{ii,2};

    I = im2single(imresize(I,inputSize(1:2)));
    scale = inputSize(1:2)./imgSize(1:2);
    bboxes = bboxresize(bboxes,scale);

    data(ii,1:2) = {I,bboxes};
end
end
```

5.7. Appendix 4 – Data Augmentation Function

Data Augmentation Function

```
function data = augmentData(A)
% Apply random horizontal flipping, and random X/Y scaling. Boxes that get
% scaled outside the bounds are clipped if the overlap is above 0.25. Also,
% jitter image color.

data = cell(size(A));
for ii = 1:size(A,1)
    I = A{ii,1};
    bboxes = A{ii,2};
    labels = A{ii,3};
    sz = size(I);

    if numel(sz) == 3 && sz(3) == 3
        I = jitterColorHSV(I,...
            contrast=0.0,...
            Hue=0.1,...
            Saturation=0.2,...
            Brightness=0.2);
    end

    % Randomly flip image.
    tform = randomAffine2d(XReflection=true,Scale=[1 1.1]);
    rout = affineOutputView(sz,tform,BoundsStyle="centerOutput");
    I = imwarp(I,tform,OutputView=rout);

    % Apply same transform to boxes.
    [bboxes,indices] = bboxwarp(bboxes,tform,rout,OverlapThreshold=0.25);
    labels = labels(indices);

    % Return original data only when all boxes are removed by warping.
    if isempty(indices)
        data(ii,:) = A(ii,:);
    else
        data(ii,:) = {I,bboxes,labels};
    end
end
end
```