

MASTER'S THESIS

Machine Learning-Based Detection of Urban Blight

Submitted in Partial Fulfilment of the Requirements of the Academic Degree
Master of Science in Engineering, MSc

Author: Vanessa Makafui Gbekor

Student Number: 2010362004

Supervisor (Host University): Mag. Dr. Michael Leitner, M.A., Full Professor, Department of Geography and Anthropology, Louisiana State University, Baton Rouge, USA

Supervisors (Home University): FH-Prof. Mag. Dr. Gernot Paulus, Department of Geoinformation and Environmental Technologies, Carinthia University of Applied Sciences

FH-Prof. Dr. Ing. Karl-Heinrich Anders, Department of Geoinformation and Environmental Technologies, Carinthia University of Applied Sciences

Villach, September 2021

Statutory Declaration

I hereby declare that:

- except for specific references which have been properly acknowledged, this work is the result of my own research, and it has not been submitted in part or whole for any other degree elsewhere.
- the electronically submitted master's thesis is identical to the hard copy.
- the enclosed master's thesis is the same version as that evaluated by the supervisors.
- one copy of this master's thesis is deposited and made available in the CUAS library (§8 Austrian Copyright Law [UrhG]).

Signature (Candidate)

(Place, Date)

Abstract

The correlation between crime and urban blight, a term used to denote disorder in a society, has been subjected to intense debate. The Broken Windows Theory introduced by Kelling and Wilson (1982), suggests that urban blight contributes to increased crime in a neighborhood. For such a study, researchers have resorted to methods including Systematic Social Observations, Google Street View and the use of geo-spatial technology such as spatial videos to be able to collect fine-scale and detailed data. The latter method is a relatively novel method which offers the advantage of collecting spatio-temporal data in a single spatial video. Nonetheless, to carry out further analysis, a process involving manual identification, collection, and classification of physical urban blight instances from the geospatial videos, is required prior to the spatial analysis of blight locations. This demands a lot of time and effort on the part of the researcher who is interested in assessing the amount of physical urban blight in the researcher's area of interest. This project seeks to capitalize on the automation capabilities offered by the field of Machine Learning (ML), to classify and automatically detect urban blight indicators in spatial videos. The workflow used in this study can be broken down into three major steps: the automatic detection of litter objects, the automatic mapping of the locations of the predicted litter objects and the classification of litter locations. From the results obtained in the automatic detection of litter objects, it was observed that the selected CNN model, YOLOv3, was capable of accurately detecting litter objects in spatial videos within a significantly shorter time than in the traditional approach. The model was seen to have a better performance when trained with a large dataset.

Keywords: Urban Blight, Machine Learning, Video Object Detection, Spatial Video Technology, Baton Rouge

Acknowledgements

Steve Jobs once said “Great things in business are never done by one person. They are done by a team of people”. This quote is a perfect description of my research journey and I deem it expedient to acknowledge the people who have supported me in diverse ways throughout this journey.

To begin with, I would like to express my gratitude to the Marshall Plan (MP) Foundation for funding this project. I am grateful for the opportunity to conduct this research in collaboration with Prof. Dr. Leitner from Louisiana State University, USA. It proved to be a very valuable experience.

I would also like to express my sincere gratitude to my supervisors Prof. Dr. Michael Leitner, Prof. Dr. Gernot Paulus and Prof. Dr. Karl-Heinrich Anders for supporting me throughout this research journey. From the conception of the project through to the implementation and documentation, they remained very supportive and dedicated to the success of this project. Their systematic guidance and valuable inputs throughout the course of this research contributed greatly to the successful completion of this research. I also appreciate the time they made for our weekly meetings, which were characterised by constructive criticisms, eye-opening discussions and recommendations, compliments as well as a healthy dose of laughter.

Moreover, I am grateful to Andrew Curtis and Jayakrishnan Ajayakumar for providing me with the Camera Player and Frame Selector software and assisting me with clarifications concerning different subject matters. I would also like to thank Judith Stratmann for providing her expert views on this research topic.

During this research, I had enormous support from my colleagues and friends. I would like to acknowledge Armstrong Aboah for his support and motivation at the beginning of this study, and Clement Nartey for his guidance and support during my Python programming efforts. My interactions with Lukas Schaffhauser, who was also working on an object detection project, were encouraging and I am grateful to him for those interactions and the exchange of ideas. Special thanks to Kobby Obeng for his massive encouragement and support in challenging times.

Furthermore, I would like to acknowledge my ever-supportive family. Special thanks go to David and Linda James, for their massive support. They have been a family to me while in Austria and supported me with genuine care, love and encouragement all through my journey. I would also like to express my immense gratitude to my dear parents, Edwin and Sally Gbekor, and my siblings. Despite the distance, they made it a point to constantly support and encourage me throughout my journey. Their prayers, support and encouragement played a vital role in spurring me on. I am also very grateful to “mein Schatz”, Andrew Filson, for standing by me, holding my hands, and lovingly supporting me through the ups and downs of this journey.

Finally, I would like to thank God for being with me all through, giving me strength and wisdom to complete this research and for strategically connecting me with all the above-mentioned people. To God be the glory!

Table of Contents

Statutory Declaration.....	ii
Abstract	iii
Acknowledgements	iv
List of Figures	viii
List of Tables.....	x
List of Abbreviations.....	xi
1 Introduction	1
1.1 Motivation	1
1.2 Research Questions	1
1.3 Approach	2
1.4 Significance of the Study	3
1.5 Organization of the Thesis	4
2 Theoretical Background and Literature Review	6
2.1 Crime Concepts	6
2.1.1 The Broken Windows Theory	6
2.1.2 Concept of Urban Blight	7
2.1.3 Fear of Crime	8
2.1.4 Discussion: Perspectives on Broken Windows Theory.....	9
2.2 Geospatial Technologies	10
2.2.1 Spatial Video Technology	11
2.3 Machine Learning Concepts.....	12
2.3.1 Supervised Learning.....	12
2.3.2 Unsupervised Learning	12
2.4 Frame Extraction Software Review	13
2.5 Review of Tools Used for Labeling Data.....	14
2.6 Object Detection Using Deep Learning	16
2.6.1 Convolutional Neural Networks.....	20
2.6.2 Deep Learning-based Object Detection: Algorithms	22
2.6.3 Deep Learning Frameworks for CNN Implementation.....	26
2.6.4 CNN Training Using Transfer Learning	26
2.7 Best Practice Examples	27
2.7.1 Non-ML-based Detection of Physical Urban Blight.....	27

2.7.2	ML-based Detection of Physical Urban Blight	29
2.7.3	Other Related Work: Object Detection	29
3	Methodology	32
3.1	General Project Workflow	32
3.2	Spatial Video Data Collection.....	33
3.3	Data Preparation.....	37
3.3.1	Data Pre-processing.....	37
3.4	Choice of Object Detection Model.....	38
3.4.1	Detailed Architecture of the Selected Model	39
3.5	Training and Validation of the Object Detector.....	39
3.6	Testing of the Object Detector	39
3.7	Evaluation of the Object Detector’s Performance.....	40
3.8	Visualization of the Detected Physical Urban Blight.....	40
4	Implementation.....	41
4.1	Study Area.....	41
4.1.1	Site Selection.....	44
4.2	Physical Urban Blight Indicator Selection	47
4.3	Litter Definition and Scope	50
4.4	Data Preparation.....	51
4.4.1	Spatial Video Frame Extraction and Selection	53
4.4.2	Dataset Splitting	55
4.4.3	Spatial Video Frames Labelling.....	56
4.4.4	Conversion of Annotations to YOLO Format.....	59
4.5	YOLOv3 Model Training and Validation	60
4.6	Testing of the Litter Detector	61
4.7	Evaluation of the Litter Detector’s Performance	61
4.8	Visualization of the Predicted Litter Objects	62
4.8.1	Spatial Distribution of Predicted Litter Objects	64
4.8.2	Kernel Density Estimation of Litter Intensity	65
5	Results and Analysis	66
5.1	Litter Detector Performance Evaluation	66
5.1.1	Training and Validation	66
5.1.2	Testing.....	68
5.1.3	Evaluation Metrics	74

5.2	Spatial Analysis of Detected Litter Objects	77
5.2.1	Visualization of Litter Locations.....	77
6	Discussion	81
6.1	Interpretation of Results	81
6.2	Benefits.....	83
6.2.1	Comparative Study : Traditional vs ML-based Approach	83
6.2.2	Benefits to Stakeholders	83
6.3	Challenges/Limitations.....	84
7	Conclusion.....	85
8	Future Work	87
	References	88
	Appendix A. Python Code to Convert Labelbox Annotation File to YOLO Format	95
	Appendix B. Python Code to Calculate Evaluation Metrics	96
	Appendix C. Training and Validation Losses.....	100

List of Figures

Figure 2.1: a) Object classification determines the category of the objects in the image; b) Object detection identifies the category of the objects as well as their location using bounding boxes; c) Semantic segmentation assigns labels of a category to the object without distinguishing between the object instances; d) Instance segmentation. Adopted from Xiao et al. (2020, p. 23730, Fig. 1).	17
Figure 2.2: The traditional object detector basic architecture. Adopted from Xiao et al. (2020, p. 23731, Fig. 3).	17
Figure 2.3: A general outlook of how object detection has performed over the years: there has been a major increase in performance (in mean average precision) since deep learning emerged in 2012; a) Excelling entries' detection results in the VOC 2007-2012 competitions; b) Results of the top object detection competition in ILSVRC2013-2017 (in both panels, only the training data which was provided is used). Adopted from Liu et al. (2020, p. 262, Fig. 3).	18
Figure 2.4: Deep learning as a sub-field of machine learning which is also a sub-field of artificial intelligence. Adopted from Patterson and Gibson (2017, p. 4, Figure 1-1).	19
Figure 2.5: The basic structure of an Artificial Neural Network. Adopted from O'Shea and Nash (2015, p. 2, Fig. 1).	19
Figure 2.6: The basic architecture of a convolutional neural network demonstrating a sequence of alternating convolutional and pooling layers. The receptive regions are shown using highlighted small boxes. The connections illustrate how features are learned implicitly. Adopted from Aloysius and Geetha (2017, p. 0589, Fig. 1).	20
Figure 2.7: The process of convolution on an RGB image. Adopted from Aloysius and Geetha (2017, p. 0589, Fig. 2).	21
Figure 2.8: The process of max pooling. Adopted from Aloysius and Geetha (2017, p. 0589, Fig. 3).	21
Figure 2.9: The operation principle of an R-CNN framework. Adopted from Tang et al. (2017, p. 725, figure 3).	23
Figure 2.10: The operation of the YOLO model. Adopted from Redmon et al. (2016, p. 780, Figure 2).	24
Figure 2.11: The trade-off between accuracy and speed for selected models including YOLOv3 and RetinaNet. Adopted from Redmon and Farhadi (2018, p. 4, Figure 3).	25
Figure 3.1: The Project Workflow	32
Figure 3.2: Spatial video equipment instalment illustration a) two cameras on the inside window on the right, b) two cameras placed on the inside window on the left, c) camera fixed on the inside windshield, d) charging equipment. Adopted from (Stratmann 2019; Ristea <i>et al.</i> 2021).	36
Figure 4.1: The four cities located in the East Baton Rouge Parish and their geographic boundaries. Adopted from Stratmann (2019, p.31, Figure 9)	41
Figure 4.2: Kernel density map showing crime densities throughout the city of Baton Rouge. Adopted from Stratmann (2019, p. 34, Figure 11).	42
Figure 4.3: Neighbourhood selection based on the crime density in the city of Baton Rouge. Adopted from Stratmann (2019, p. 36, Figure 12).	44

Figure 4.4: The spatial distribution of locations where property blight and environmental/infrastructural blight are present within the five selected neighbourhoods in Baton Rouge. Adopted from Stratmann (2019, p.52, Figure 23).	45
Figure 4.5: a. Property blight density; b. Environmental/Infrastructural blight density. Adopted from Stratmann (2019, p. 62, Figure 33).	46
Figure 4.6: The selected sub-areas in Fairfields and MidCity to be used in this study.	47
Figure 4.7: The frequency of individual property blight indicators. Adopted from Stratmann (2019, p. 50, Figure 19).	48
Figure 4.8: The frequency of different environmental/infrastructural blight indicators. Adopted from Stratmann (2019, p. 50, Figure 20).	49
Figure 4.10: An uploaded spatial video (middle) and its GPS track (upper right) shown in the Camera Player software	52
Figure 4.11: Parameter settings for extracting frames from videos using 'Scene Filter' in VLC media player	54
Figure 4.12: Interface of the Frame Selector software	56
Figure 4.13: Sample annotations of litter objects whose shapes naturally allow a section of the object's surrounding area to be included.	57
Figure 4.14: Sample annotation of a litter object where context (the object's surrounding) is deliberately included.	57
Figure 4.15: Workflow for linking GPS coordinates to their corresponding video frames	62
Figure 5.3: Training and validation losses on the first, smaller dataset.	67
Figure 5.4: Training and validation losses on the second, larger dataset	68
Figure 5.5: Predictions made by models 1 and 2 (true negative)	69
Figure 5.6: Predictions made by models 1 and 2 (false negative)	70
Figure 5.7: Predictions made by model 1 (false positive)	71
Figure 5.8: Predictions made by model 2 (false positive)	71
Figure 5.9: Predictions made by model 1 (true positive)	72
Figure 5.10: Predictions made by model 2 (true positive)	72
.....Figure 5.11: Predictions made by model 1 (true positive)	73
.....	73
Figure 5.12: Predictions made by model 1 (true positive)	73
Figure 5.13: Predictions made by model 2 (true positive)	74
Figure 5.14: Automatically generated map of litter locations	77
Figure 5.16: Spatial distribution of predicted litter objects	79
Figure 5.17: Kernel density map for litter intensity	80
Figure 6.1: Attempted solutions for encountered model implementation errors	84

List of Tables

Table 1.1: Sub-section of requirement catalogue for physical urban blight indicators adopted from (Weisburd et al. 2010; Hinkle and Weisburd 2008; Skogan 1990; Ross and Mirowsky 2001) as cited in Stratmann et al. (2020, pp. 5-6, Table 1).	3
Table 2.1: Comparison of selected two-stage methods including Faster R-CNN, and one-stage methods including YOLOv3, SSD, and RetinaNet. Adopted from Redmon and Farhadi (2018, p. 3, Table 3).	25
Table 3.1: The criteria catalogue for the physical urban blight indicators. Adopted from Stratmann (2019, pp. 16-17, Table 2).	34
Table 3.2: Details of the distance, duration and dates of the spatial video data collection. Adopted from Stratmann (2019, p. 44, Table 4).	37
Table 4.1 Differences between litter and illegal dumping	51
Table 4.2: The neighbourhood in Baton Rouge, US, video length and size, and the purpose of the selected videos.....	52
Table 4.3: Details of the datasets used in this study; the number of video frames and litter objects used at various stages in the study, the videos the frames were extracted from, and the neighbourhoods covered.	58
Table 4.4: Microsoft Excel file showing a section of a GPS log file extracted from Camera Player and the media time conversion from hh:mm:ss to seconds.	63
Table 5.1: Summary of training and validation losses for dataset 1 and 2 and the final epoch number for training and validation.....	68
Table 5.2: Summary of evaluation metric results for models 1 and 2	75
Table 5.3: IoU-based Pseudo-confusion matrix for model 1	75
Table 5.4: IoU-based Pseudo-confusion matrix for model 2	76
Table 5.5: Comparison of the results obtained using manual observation of the predicted litter objects to those obtained using the IoU at a threshold of 0.....	76
Table 5.6: The resulting attribute table from the automatic mapping process including the number of objects detected in each video frame.	78

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
BVLC	Berkeley Vision and Learning Center
BWT	Broken Windows Theory
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
CVAT	Computer Vision Annotation Tool
DL	Deep Learning
DPM	Deformable Part-Based Model
FC	Fully Connected
FN	False Negative
FP	False Positive
FPS	Frame Per Second
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
GSV	Google Street View
HOG	Histogram of Oriented Gradients
IoU	Intersection over Union
KDE	Kernel Density Estimation
LISA	Local Moran's I Spatial Autocorrelation
mAP	Mean Average Precision
ML	Machine Learning
MTurk	Amazon Mechanical Turk
NLP	Natural Language Processing

R-CNN	Region-based Convolutional Neural Network
ROI	Region of Interest
SIFT	Scale Invariant Feature Transform
SSD	Single Shot MultiBox Detector
SSO	Systematic Social Observation
SVAS	Spatial Video Acquisition System
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
VoTT	Visual Object Tagging Tool
YOLO	You Only Look Once

1 Introduction

1.1 Motivation

Urban blight denotes disorder in a neighborhood or the deterioration of elements such as buildings. It may be socially- or physically - related. The former describes anti-social behaviors displayed by residents. These behaviors may include public verbal harassment, urination in public places and open solicitation for prostitution (Sampson and Raudenbush 1999; Skogan 1990; Kelling and Wilson 1982). The latter is normally characterized by broken windows or doors, abandoned vehicles, litter, unkempt areas, abandoned property, illegal parking, and other forms of disorder. Over the years, researchers have sought to evaluate the correlation between urban blight and crime in a neighborhood. This includes the Broken Windows Theory introduced by Kelling and Wilson (1982), which suggests that urban blight contributes to increased crime in a neighborhood. Such a study will require the use of fine-scale and detailed data. Different data collection methods have been adopted including the use of geospatial video technology, which is also known as spatial video (Stratmann 2019; Stratmann *et al.* 2020). However, the methods used required a lot of time and effort on the part of the researcher. In addition, the manual classification of urban blight carried out was subjective. It is necessary to identify urban blight in spatial videos in a relatively easier, time-efficient and objective manner.

Machine Learning (ML) is a sub-set of Artificial Intelligence (AI), which enables processes to be automated and large data to be more effectively handled in order to derive relevant insight. ML can be defined as training a computer to think and learn like a human being in a defined situation without being told what to do explicitly. This requires training the computer using data and allowing it to improve its learning over time based on the training data (Samuel 1959). ML has a myriad of applications such as object detection, product recommendation, and virtual personal assistants (Packt 2018). This project seeks to capitalize on the automation capabilities of ML in order to apply it to automatically detect and classify physical urban blight indicators from spatial videos.

1.2 Research Questions

Researchers have utilized different methods in collecting data to study the correlation between crime and urban blight. Sampson and Raudenbush (1999) made use of the Systematic Social Observation (SSO) approach to derive physical and social disorder indicators from analogue videos. This derived information was useful in estimating the amount of urban blight in a neighbourhood. Another existing approach involves the use of Google Street View (GSV). The use of GSV offers the advantage of low cost and safety. However, the temporal components of images pose certain limitations due to the issues regarding flexibility and stability (Curtis *et al.* 2013a; Marco *et al.* 2017).

A relatively novel approach, which involves the application of geospatial technology methods, such as spatial videos and geo-narratives, has facilitated the collection and analysis of spatio-temporal data of physical urban blight locations using Local Moran's I Spatial Autocorrelation (LISA) and the Kernel Density Estimation (KDE) (Stratmann 2019; Stratmann *et al.* 2020). These methods were successful to accomplish the required tasks and to answer all research questions regarding the effectiveness of spatial videos and geo-narratives in identifying physical urban blight, and the relationship between physical urban blight and crime in Baton

Rouge, USA. However, the methods required a manual identification, collection, and classification of physical urban blight instances from the geospatial videos prior to the spatial analysis of blight locations. This demanded a lot of time and effort on the part of the researcher who is interested in assessing the amount of physical urban blight in the researcher's area of interest. Furthermore, the manual classification carried out was more subjective than objective, since the classification was determined by the researcher who was carrying out the classification tasks.

ML has already been successfully applied in many areas including object classification, identity confirmation, obstacle detection, and avoidance in autonomous driving (Zha *et al.* 2015). Mohana and Aradhya (2016) suggested that ML is useful in detection and classification of objects in video surveillance. Jogin *et al.* (2018) studied the use of different classifiers including the k-Nearest Neighbor, linear classifier, softmax classifier, Support Vector Machine and Convolutional Neural Network (CNN). They proposed the CNN as the optimum classifier for video classification based on their results. CNN has different architectures including "You Only Look Once" (YOLO) and AlexNet. Research has shown that YOLO is useful in increasing computation speed while correctly identifying objects in videos (Jana *et al.* 2018). This project seeks to harness the strengths of ML in automating the identification, collection, and classification of urban blight indicators in geospatial videos, possibly resulting in a massive reduction of time and efforts required by for such a task.

Therefore, this research aims to answer the following central question and sub questions:

- How suitable is ML in detecting urban blight indicators?
 - a. How suitable are Convolutional Neural Networks (CNN) in detecting urban blight indicators?
 - b. What considerations are necessary in developing a requirement catalogue for detecting urban blight within the context of ML?




1.3 Approach

The beginning of this project will be devoted to an extensive literature review of the state of the art of different ML algorithms that have already been adopted in detecting and classifying objects in images and videos. Studies have shown that Convolutional Neural Networks, which represent a class of ML algorithms, are applicable in face recognition and object detection (Howard *et al.* 2017; Girshick *et al.* 2013). An optimum ML algorithm will then be selected and implemented based on the literature review. For consistent analysis, a requirement catalogue used extensively in literature will be adapted. The suitability of this catalogue within the context of ML will be evaluated. An appropriate requirement catalogue will subsequently be defined. Labelbox, a platform which is used to scale the creation and management of quality training data, also provides image and video classification tools (Labelbox 2020). This platform may be relevant in labeling the different physical urban blight indicators. An example of the requirement catalogue to be adopted from existing literature is illustrated in Table 1.1.

The appropriate quantity of data required for the analysis will be determined and preprocessed to obtain as accurate and high-quality results as possible. The ML models will then be trained with a sufficiently large training data set (i.e., spatial videos). Some ML algorithms such as neural networks usually require their parameters to be adjusted during the training process until

desired results are obtained. Therefore, in the event of a neural network being selected, a process of parameter fine tuning will be included for optimum results. A portion of the data will be reserved for the testing process. This involves the ML model detecting and classifying physical urban blight indicators in new and previously unseen data samples (i.e., spatial videos). The accuracy assessment measures, such as error and accuracy rates, will be considered to guide the selection of an optimum ML model. Subsequently, the results from the selected model will be compared to the results of the manual detection and classification of physical urban blight indicators from the 2019 Marshall Plan (MP) research project. A subsequent comparison of different spatial analysis results between the ML algorithms and the manually collected urban blight locations will complete this proposed research.

Table 1.1: Sub-section of requirement catalogue for physical urban blight indicators adopted from (Weisburd et al. 2010; Hinkle and Weisburd 2008; Skogan 1990; Ross and Mirowsky 2001) as cited in Stratmann et al. (2020, pp. 5-6, Table 1).

Physical urban blight indicator	Description	Image
1. Property		
Abandoned property	Properties that show signs of decay or abandonment (e.g. doors are boarded up); Nobody lives in the house.	
Broken window/door	Broken windows/doors are left unrepaired.	
Boarded window/door	Windows are boarded up with plywood or other material or covered with plastic, foil, etc.	

1.4 Significance of the Study

- Identification of the necessary considerations to be made in developing a requirement catalogue for detecting urban blight within the context of ML. This adapted requirement catalogue may be applied by other researchers who may be interested in applying ML to object detection in spatial videos.

- Implementation of an optimum ML model which will be suitable for the automatic detection and classification process in identifying physical urban blight indicators, leading to reduced time and efforts on the part of researchers and analysts.
- To make a recommendation of the most appropriate ML method to be applied in evaluating and assessing urban blight depending on factors, such as time constraint. This will be useful to researchers who may be interested in assessing the relationship between crime and urban blight as well as the link between urban blight and health conditions.
- The proposed project and best performing ML algorithm can also be applied in detecting damages in infrastructure as part of disaster and risk management efforts.
- The proposed project and best performing ML algorithm may also be applicable to identify graffiti locations by researchers who are interested in graffiti research.

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows:

Chapter 2 presents an overview of relevant concepts which undergird this research. These concepts include crime concepts such as the Broken Windows Theory and other concepts about urban blight. Geospatial technologies are described with emphasis being placed on the spatial video technology used in this study. In addition, the concept of machine learning is explained. Overviews of some available frame extraction tools and labeling tools are also provided in the chapter. Furthermore, the concept of object detection using deep learning is discussed. The chapter concludes with a review of some best practice examples for detecting physical urban blight using both non-ML and ML methods. Other related work in the field of object detection which are relevant for this study are described as part of the best practice examples.

Chapter 3 describes the methodology to be adopted in this research. The data used as well as the steps to be employed in preparing the data for analysis are described. The architecture of the selected object detection model is explained. In addition, brief overviews of the training, validation and testing processes to be used in implementing the ML model are provided. Furthermore, some evaluation metrics used in evaluating deep learning models are presented. The last section of the chapter describes how the identified urban blight locations are visualized.

Chapter 4 describes the study area and the physical urban blight indicator that were selected for this study. To avoid ambiguity, the selected physical urban blight indicator is defined within the scope of this project. The steps followed in implementing the ML model for automatic urban blight detection are explained in detail. Moreover, the evaluation metrics used in evaluating the model's performance are described. The chapter concludes with a description of how the predicted physical urban blight objects are visualized.

Chapter 5 presents the results obtained after following the processes described in Chapter 4.

In Chapter 6, the results are interpreted and a discussion of the benefits and limitations of the proposed ML model for urban blight detection from spatial videos is carried out.

Finally, Chapter 7 summarizes the thesis and Chapter 8 provides recommendations for future work that will augment the contributions made by this research. Further recommendations are

made for future work in different fields such as health, in which the proposed ML model may be adopted.

2 Theoretical Background and Literature Review

This chapter begins with a section about criminological and urban blight concepts which tackle the link between urban blight and crime. Subsequently, the spatial video technology which is used in this study is described. Some applications of the technology are also discussed. The following sections explore machine learning, frame extraction software, tools required for labelling data, object detection and deep learning concepts. The chapter concludes with a review of some best practice examples for detecting physical urban blight using non-ML methods. Approaches involving the use of ML are also mentioned. Other related work in the field of object detection which are relevant for this study are described as part of the best practice examples.

2.1 Crime Concepts

Researchers in diverse fields including criminology, health, psychology, sociology and geography are commonly interested in theories that highlight principal neighbourhood and community features. Such theories may be relevant in identifying crucial relationships between community characteristics. As such, various concepts which describe disorder within neighbourhoods and their relationship to objective and subjective crime indicators exist. Data regarding objective crime indicators may be obtained from official crime reports. On the other hand, subjective crime indicators may show how an individual perceives safety or the fear of crime in a specific neighbourhood (van Bakergem *et al.* 2017; Stratmann 2019; Ross and Mirowsky 2001).

2.1.1 The Broken Windows Theory

In criminology, the Broken Windows Theory (BWT), plays a vital role. It was initially presented by Kelling and Wilson (1982) in *The Atlantic Monthly* magazine. The authors presented their point of view, stating that neighborhood disorder could be indicated by factors such as graffiti, broken windows or highly unkempt lawns filled with overgrown weeds. From their perspective and research, broken windows in a community often showed a lack of enforcement and community supervision. This could potentially encourage more disorder or crimes. Generally, one broken window in a community would facilitate other broken windows appearing on the scene of the community involved in a short time frame. This occurs irrespective of how well the community functions on a whole. Small violations build up into disorder over time. Disorder begets disorder. Hence, the lack of enforcement will encourage a lot of residents to have a lackadaisical approach to situations and to disregard pre-existing norms and regulations. Others may engage in violent activities. These forms of disorder may lead to increased fear within certain people in the community. This fear may force some residents to move to a different community where law and order may still be existent to a higher degree. This further leads to increased disorder on all level and may be characterized by increased violence, inordinate drinking activities as well as excessive and rampant littering. The resulting community will normally be characterized as a place where regular residents are afraid of crime and therefore choose not to go to public places as much as possible (Stratmann 2019; Kelling and Wilson 1982).

The Broken Windows Theory influenced policing strategies in urban areas. This is also referred to as broken windows policing. Patrols were undertaken by police officers in assigned neighborhoods where disorder was present. This approach to maintain law and order positively influenced prominent U.S. cities (Sampson and Raudenbush 1999).

The BWT is essential for the urban blight concept which is described in the next section.

2.1.2 Concept of Urban Blight

Over the years, the term “urban blight” has been defined by many authors in different ways. This may be attributed to their focus on its characteristics and effects, rather than establishing a standard definition for the term (Lousada *et al.* 2021; Hoffman 2012). In their Atlantic Monthly article, “Broken Windows”, Kelling and Wilson (1982) used the term “urban decay”. This term is synonymous to “urban blight”.

The word “blight” has its origins in the 16th century and was initially used by farmers to describe plant disease which causes the affected parts of the plant to wither and die (Oxford Dictionaries 2019). The word evolved and became associated with urban neighbourhoods which were falling into disrepair or disorderliness. Darling (1943) introduced the term “urban blight” and described it in broad terms as a condition of properties which is caused when such properties are neglected and inhabited by people who live below their society’s average standard of living. Ferreira *et al.* (2018) mentioned that such neighbourhoods are typically characterized by poor housing where there are limitations with space, air quality and light. Also, these areas are often inhabited by people who desperately need a place to stay and whose income would not allow them to afford alternative houses which have appropriate living conditions (Darling 1943). Studies have shown that urban blight is typical in areas where there are high poverty rates, minority households, slums and crime (Stratmann 2019; Stratmann *et al.* 2020).

Generally, researchers differentiate between social and physical urban blight. Social urban blight is mostly used as a broad term to denote behaviours which are considered anti-social and may be perpetrated by unpredictable or unstable people in society. These may include open solicitation for prostitution, urination or sleeping in public places, open verbal harassment, school truancy and nuisance neighbours (Sampson and Raudenbush 1999; Skogan 1990; Kelling and Wilson 1982). On the other hand, physical urban blight, which is the focus of this research, is used to denote deterioration or disorderliness of urban areas. A more recent definition of urban blight by Sun *et al.* (2019) depicts this form of urban blight; “abandoned or poorly maintained real properties, often characterized by overgrowth, litter, abandoned vehicles, junk, and dumping”. Other examples of physical urban blight include abandoned or burned cars, graffiti, broken windows, and broken streetlights, etc. Although social urban blight occurs more habitually than physical urban blight, the damage of the former is not as glaring as that of the latter (Stratmann 2019; Kelling and Wilson 1982; Skogan 2012).

Drawing a line between social and physical urban blight is not always a simple task in spite of the differences that exist between the two forms of urban blight. Rather than mostly considering physical factors, the concept of urban blight has evolved to envelope subjective factors as well. In order to enhance the understanding of disorder in urban areas, a combination of social and physical urban blight, as well as socio-economic factors is required (Stratmann 2019; Hinkle and Weisburd 2008; Skogan 1990).

2.1.3 Fear of Crime

In a study carried out by Kelling and Wilson (1982), it was observed that certain factors aggravated the fear of crime in people while others caused people in some neighborhoods to feel safer. One typical factor mentioned was the presence of police officers foot patrolling selected areas. It was recognized that although crime rates had rather increased, the mere presence of foot patrol officers made some citizens living in foot patrolled areas feel safer. Hence, they were less careful about putting in place security measures including locking their doors. Within the area under study, informal rules which were generally understood and accepted by the residents there were used to maintain some level of order. For example, bottles had to be put in paper bags, begging was not permitted, talking to or bothering people at the bus stops was not allowed, etc. This brought to light the idea that people are not only afraid of sudden violent attacks from strangers. Rather, they also fear being victims of disorderly behaviors exhibited by mentally disturbed people, loiterers, addicts, rowdy teens, panhandlers, to mention but a few.

Consequently, in areas where there is less social control, it is common to find people altering their behaviors or routines to avoid being in situations where their safety is compromised. For example, people who do not feel safe may reduce their use of streets or other public places especially at odd hours, stay indoors or walk hurriedly through such areas. Some may leave to a different neighborhood where they feel safer. These altered behaviors may potentially encourage crime since these areas are made more conducive for criminals to perpetrate their activities (Kelling and Wilson 1982).

In a research conducted by Sampson and Raudenbush (2001), showed that cohesion among residents led to the reduction of physical and social disorder. Furthermore, a combination of high cohesion among residents and increased social observation by police officers led to low violence. While police presence in some neighborhoods leads to residents feeling safer, some studies have also shown that in other neighborhoods, residents tend to become very fearful due to the police presence. This however does not include people who commute through such areas. This tendency is more applicable to people who monitor street behavior from their homes (Weisburd *et al.* 2010). Hinkle and Weisburd (2008) mentioned that the increased fear associated with increased police activity may be attributed to two possible reasons. They stated that when people notice increased police presence in their neighborhoods, they may be reminded of the already existing issues in their areas. Additionally, residents are likely to infer that there is an increase in crime in their areas when there is a sudden increase in police presence on their block. This leads them to begin to view their blocks as more dangerous than in previous times.

Generally, disorder elicits negative impressions in people's minds. Thus, urban blight has the potential to affect the value of properties negatively (Sun *et al.* 2019). Within the real estate market, blighted areas are more likely to have less sales since people will be hesitant to purchase houses in such areas (Sampson and Raudenbush 2001). Additionally, the perceived disorder within a neighborhood can affect the health of residents negatively, leading to chronic health issues and increased physical impairments.

2.1.4 Discussion: Perspectives on Broken Windows Theory

Kelling and Wilson (1982) proposed a correlation between disorder and crime in their broken windows theory. They threw more light on how disorderly conditions such as broken windows, could lead to much more serious crime if not tended to. This is similar to how one unfixed broken window could result in other windows being broken and subsequently, severe damage being done to the entire building (Gault and Silver 2008). There are differing perspectives about the BWT among researchers. Therefore, this hypothesis has been subjected to various tests and discussions. The perspectives of some researchers appear to be consistent with the BWT. However, there remain other researchers who stand as critics of the theory.

An influential article which was published in the *American Sociological Review* by Sampson and Raudenbush (1999) stated that there was no direct correlation between disorder and crime. They reported their findings as being contrary to the views pushed forward by the Broken Windows Theory (Gault and Silver 2008). Harcourt (2009) put forward a similar argument about the lack of evidence of a direct link between serious crime and disorder (Hinkle and Weisburd 2008).

In response to the afore-mentioned critique, some researchers have argued that the BWT does not propose a direct relationship between disorder and crime, but rather an indirect relationship involving the fear of crime and its associated effects on informal social controls (Hinkle and Weisburd 2008; Bratton and Kelling 2006). Gault and Silver (2008) maintained a neutral stance but sought to present their alternate interpretation of the findings of Sampson and Raudenbush (1999), which they believed is more consistent with the latter's empirical findings. They argue that Kelling and Wilson (1982) suggested a chain of events occurring between disorder and crime, and not a direct link. Furthermore, they mention that Sampson and Raudenbush's findings are not contradictory to the BWT but rather supports the BWT since it also shows that disorder weakens social control, which also results in increased crime (Gault and Silver 2008).

The results of a study carried out by Hinkle and Weisburd (2008) showed that there is a link between disorder and fear of crime, thus substantiating the BWT hypothesis. They however mention that policing strategies which are based on the broken windows should focus on not only reduction of disorder, but also on warding off citizens' fear of crime (Skogan 1990; Weisburd *et al.* 2010; Stratmann 2019).

The broken windows theory heavily influenced policing strategies over many years. The broken windows policing strategy involves strict law enforcement by police where there is zero tolerance. An analysis of various disorder policing strategies which was carried out by implementing several experiments, showed that the strict law enforcement strategy is not always helpful. Alternatively, there exist policing strategies which are more problem-oriented and allow active participation by citizens in problem-solving interventions. These strategies have proven to be more effective in reducing urban blight and crime rates as compared to the strategies which focus solely on untended behaviour (Braga *et al.* 2015; Skogan 2008; Stratmann 2019).

(Branas *et al.* 2018) investigated the result of standardized, reproducible interventions aimed at restoring blighted vacant land on crime, violence and people's perception of fear and safety. To achieve this, they employed a mixed-methods approach including quantitative and ethnographic

analyses. The results obtained after analyzing 445 randomly sampled participants over a period of 38 months showed that for those who resided near restored vacant lots, there were significant reductions in the perceptions of crime, vandalism and safety concerns when moving outside of their homes. In addition, their use of public spaces for socializing and relaxation, increased significantly. Pertaining to crime, significant reductions were also obtained. Therefore, they concluded that blighted urban areas have an impact on people's perception of safety. Furthermore, they proposed that treatment of blighted vacant lots can be effective for intervening in crime, violence involving guns as well as fear in urban areas. Their conclusion is consistent with the BWT hypothesis.

Haney (2007) assessed the impact of living in a poor area on the self-esteem of residents in such areas. In his article, he mentioned that rather than merely eliciting fear in residents of blighted areas, residents were more likely to perceive a sense of fatalism; a feeling that misfortunes and incivilities will occur irrespective of their actions. He alluded this to severe psychological effects. Furthermore, he stated that perceived disorder is more related to who resides in a particular area than physical urban blight and observed behaviours. Thus, contradicting the theory.

The discussion held in this section shows the divergent opinions held by various researchers concerning the broken windows theory as well as diverse policing strategies. Nonetheless, a relationship can be observed between crime and urban blight, even though the extent and correlation are not clearly established yet.

2.2 Geospatial Technologies

Geospatial technologies consist of methods that are used to store, acquire, analyze, model and visualize spatial data. They include Geographic Information Systems (GIS), photogrammetry, remote sensing, Global Navigation Satellite Systems (GNSS), traditional survey methods and laser scanning, etc. These advanced methods are applicable in a wide range of location-based services. However, they pose certain limitations. For example, satellites are unable to capture vertical surfaces in an efficient manner due to their perspective. On the other hand, Unmanned Aerial Vehicles (UAVs) can efficiently capture vertical surfaces since they have the correct perspective for achieving that. UAVs also provide high resolution images. Nonetheless, there are legal issues surrounding the use of UAVs and these are largely dependent on the local legislation. In some areas, UAVs are prohibited from flying in urban areas without permission. It is possible to use handheld Global Positioning System (GPS) devices and other conventional survey techniques to correctly position contextual data. However, this approach is time-intensive and is therefore not suitable for studies involving large areas (Strelnikova *et al.* 2018; Stratmann 2019). A more efficient and cost-effective alternative has been used for environmental observations. Nonetheless, pictures provided by Google Street View have a limitation of temporal instability. This approach is therefore unsuitable for spatio-temporal studies (Curtis *et al.* 2013b; Stratmann 2019).

Presently, a lot more focus is on finding methods which overcome these limitations, i.e., methods which are time-efficient, cost-effective, capable of collecting contextual features and applicable in finding spatial correlations in a GIS (Curtis *et al.* 2015; Stratmann 2019).

2.2.1 Spatial Video Technology

Videography is a concept which has been used in diverse ways and within several disciplines. Basically, it refers to the capture of moving objects using video cameras. Aside the visual component, which is collected using video cameras, they are capable of also capturing audio. Videography has been applied in social science research mainly to observe different phenomena or events, provide feedback, and to learn from a distance. With the convergence of GPS, coupled with reduced costs of videography components and sensor enhancements, the resulting technology - spatial video - promises to be a great tool for collecting field data and may be used in different disciplines especially geography (Mills *et al.* 2010).

Spatial Video, otherwise known as Spatial Video Acquisition System (SVAS), is a relatively new technology that can capture contextual field data for in-depth research. Using the spatial video technology approach, spatio-temporal data can be captured and analyzed to determine geographic occurrences as well as any environmental changes. Furthermore, it is described as a ground-level remote sensing method which improves environmental monitoring due to its combination of GPS and video. The SVAS is made up of digital video cameras which are linked to a GPS unit. This enables individual frames of the video to be linked to GPS coordinates. As a result, each recorded frame has its corresponding geographic location and timestamp information. Varying surveying vehicles can serve as the platform onto which the SVAS is mounted. Several angles can also be recorded by using more than one camera (Curtis *et al.* 2013a; Curtis *et al.* 2015; Stratmann 2019; Ristea *et al.* 2021).

Recently, the spatial video technology has been adopted in several studies. For example, it has been applied in post-disaster assessment to track the recovery of selected areas following a hurricane or firestorm (Mills *et al.* 2010). It has also been applied in health-related studies to support field epidemiology. By analyzing the spatial videos, the effect of seasonal changes on cholera within endemic areas can be determined. To add to this, by extracting contextual characteristics such as standing water, accumulated trash etc., from the spatial videos, health risks such as tuberculosis can be identified. Therefore, spatial videos enhance spatial epidemiological analysis and make it possible to identify spatial patterns (Curtis *et al.* 2016; Curtis *et al.* 2015). Moreover, an underwater version of the spatial video which involves the attachment of the GPS and cameras to a boat, has been applied in coral health status analysis (Riegl *et al.* 2001; Stratmann 2019; Ristea *et al.* 2021). SVAS has been adopted to collect contextual data in crime-related research (Curtis *et al.* 2012; Stratmann 2019; Ristea *et al.* 2021). The technology has been used in assessing and evaluating the occurrence of urban blight within selected neighborhoods in Baton Rouge, USA. By leveraging the spatial video and some other technologies in a mixed methods approach, fine-scale information of the occurrences of physical urban blight as well as people's perception of safety could be analyzed (Stratmann 2019; Ristea *et al.* 2021).

Although the spatial video is time-efficient, cost-effective and provides valuable insights into spatial patterns of specific occurrences and can be applied in diverse fields, the mapping process is laborious. The spatial video has to be watched and the identified and desired occurrences must be digitized into a GIS layer (Stratmann 2019; Ristea *et al.* 2021; Ajayakumar *et al.* 2021; Stratmann *et al.* 2020). This makes the spatial video less scalable and sustainable (Ajayakumar

et al. 2021). Therefore, there is the need to explore the use of ML methods to automate the transformation of spatial video frames into maps.

2.3 Machine Learning Concepts

In 1956, a group of computer scientists introduced the term “Artificial Intelligence (AI)” at the Dartmouth Conferences. The term connotes intelligence which is demonstrated by machines. Since then, AI has evolved in a great measure due to the widespread availability of data and increased computational power. Intelligence is often associated with human minds. Therefore, when a machine imitates cognitive functions that are normally associated with human minds, the term “artificial intelligence” comes into play. These cognitive functions typically include “learning” and “problem solving”.

Machine Learning (ML), a subset of AI, was first defined by Arthur Samuel as programming computers in such a way that they learn how to accomplish a task from experience. Thus, removing the need to elaborately program the computers to perform desired tasks (Samuel, 1959). ML is mostly applied in complex cases where explicitly programming algorithms is infeasible. For example, in computer vision, email filtering and network intruder detection.

Computer programs can be taught to perform specific tasks using different machine learning techniques. These include supervised learning, unsupervised learning, reinforcement, and semi-supervised learning. However, the most commonly adopted techniques are supervised learning and unsupervised learning (Ongsulee 2017).

2.3.1 Supervised Learning

In supervised learning, algorithms are provided with labeled examples during training. These labeled examples contain some pre-defined inputs and their associated correct outputs. This implies that the algorithm is given examples of what the “right answer” is for each input. Given additional unlabeled data, the algorithm predicts the values of the labels. Learning is achieved by the algorithm by comparing its predictions to the correct outputs to identify errors. It then adjusts the model where necessary. Supervised learning is commonly applied in areas where historical data can be used to predict future events that are likely to occur. For example, it can be applied to detect fraudulent credit card transactions. Popular supervised learning techniques include classification, regression, gradient boosting and prediction (Ongsulee 2017).

2.3.2 Unsupervised Learning

The unsupervised learning technique, as its name implies, involves no supervision or guided learning. This means that no labeled examples are provided to the algorithm. Therefore, the algorithm has no information on what the “right answer” is. It is required to explore the data and identify a hidden structure or pattern within the provided data. Based on the identified structure, the algorithm can create segments of the data. Typical applications include identifying consumer segments for more targeted marketing efforts and segmentation of text topics. The nearest-neighbour mapping, self-organizing maps, singular value decomposition and k-means clustering are among the most popular unsupervised learning techniques (Ongsulee 2017).

2.4 Frame Extraction Software Review

In recent times, video content has gained a lot of attention. This is due to advancements in technology and the increasing use of social media. This brings to fore the challenge of identifying what the videos contain – video annotation which may also be referred to as video labeling. Video annotation may be carried out on individual frames which are extracted from the video or may be carried out directly on the video (Gaur *et al.* 2018). The former usually requires the use of a frame extraction software or tool to extract the individual frames for subsequent video labeling processes.

Due to the explosion of video content, many tools have been created to work with videos. As a result, there are multiple frame extraction software which can be used to generate individual video frames or images from the video content. Free Video to JPG Converter¹ is one such extraction software. It supports multiple video formats including mp4 and can perform frame extraction on multiple videos in one instance. The extraction of individual video frames from the video is done automatically and saved to a user-defined output folder as a JPG image file. It can extract batches of frames based on a specified time (e.g., every 1 second) or by a specified interval (e.g., every 50 frames). A specific total number of frames can be extracted. For example, only 10 frames out of all the video frames. This gives the user great flexibility and simplifies the extraction of many video frames. However, Free Video to JPG Converter requires that .NET framework is pre-installed before it can be installed. Another tool known as Batch Video to Image Extractor² works in a similar way to the Free Video JPG converter in terms of automatic extraction of frames in batches based on specified intervals, time or number of frames. However, Batch Video to Image Extractor offers support for other image formats such as TIFF, GIF etc. It can capture all the video frames as one image, with a specified number of columns and lines. Batch Video to Image Extractor tool allows for users to specify the output frame size and set the frame per second (fps). A drawback of this tool is that it has a relatively longer processing time. Another frame extraction software worth mentioning is the VLC Media Player.³ It supports multiple video formats and allows for video frames to be extracted in batches based on a defined parameter known as the recording ratio. VLC Media Player has the option of extracting frames manually as well as automatically. The manual option is suitable for capturing a few frames. Although the other two extraction tools can work on multiple videos at a go, VLC works as a complete media player with many other functions and performs the extraction of frames very well (Singh 2018). There is yet another tool called FFmpeg.⁴ It is a command line tool and is useful for changing the file formats of multimedia content. It can therefore be used to extract video frames. Compared to VLC Media Player, FFmpeg is less intuitive due to the use of the command line (FFmpeg 2021). VLC Media Player is therefore easier to use due to its simple and intuitive user interface. In addition, there is yet another tool called Frame Selector, which was developed by Andrew Curtis et al.⁵ Frame Selector is a stand-alone software which can be used to manually extract individual frames from a video while it

¹ <https://www.dvdvideosoft.com/products/dvd/Free-Video-to-JPG-Converter.htm>

² <https://batch-video-to-image-extractor.en.softonic.com/>

³ <https://www.videolan.org/vlc/>

⁴ <https://ffmpeg.org/download.html>

⁵ <https://www.kent.edu/geography/profile/andrew-curtis>

is playing. Hence, extracting many frames must be done manually. Nonetheless, it offers the advantage of being able to save frame extraction sessions, making it easier to identify the extracted frames from the video after a period of time. Moreover, Frame Selector’s naming convention for the extracted frames using the video file name and the timestamp may prove useful for properly identifying individual frames. To add to this, by using Frame Selector to manually extract the spatial video frames, the issue of varying image quality will be addressed (Ajayakumar *et al.* 2021).

For this study, VLC Media Player will be useful for automatically extracting many video frames. This may be useful for generating many video frames for training the object detection model. Furthermore, the media player may be useful for performing functions such as viewing or trimming the video as is appropriate for any platforms on which they will be uploaded onto. Frame Selector may be a more suitable option for differentiating between the various extracted video frames; know which video were extracted and at which time. This may be useful for the automatic mapping process which will be discussed in subsequent sections.

2.5 Review of Tools Used for Labeling Data

For computer vision tasks, generating ground truths is a vital part of training and testing of computer vision (CV) algorithms. This has led to increased development of tools and frameworks to support researchers in the creation and collection of datasets. After creating ground truth labels, the datasets that are derived are used in various CV tasks including object detection and tracking.

Research groups often develop stand-alone tools for annotation tasks. The tools are usually tailor-made for their specific needs. These tools include GTVT, ODViS, GTTool, ViPER-GT. These are however limited in terms of generating ground truth datasets on a large scale. Furthermore, they can only be used by a limited number of people. Moreover, sharing of labeled data is not supported. There are also web-based tools such as LabelMe⁶ which allow for collaborative efforts; large groups of people can combine their efforts in creating and collecting reliable ground truths. Nonetheless, LabelMe’s platform does not combine multiple annotations made by multiple annotators. As a matter of fact, the data from LabelMe is considered to be quite inaccurate. LabelMe supports the annotation of still images. It also has a version that supports video annotation. However, the video-based version is neither as successful nor as flexible as that of the image version. Alternative platforms such as Amazon’s Mechanical Turk (MTurk),⁷ leverage crowdsourcing to collect large, annotated datasets. On MTurk, workers are motivated by being paid for their work. This is because workers’ motivation affects their work and must be considered. Apart from workers’ motivation, quality control must be considered. Methods such as task redundancy (where several people label the same data), ground truth seeding (combining the ground truth with the test data) and user reputation have been used successfully to support the building of large-scale datasets. However, they may be expensive

⁶ <http://labelme.csail.mit.edu/Release3.0/>

⁷ <https://www.mturk.com/>

and may still be of low data quality since workers may not be as motivated as researchers even if they are paid (Kavasidis *et al.* 2014).

Other modern data annotation/labeling platforms include Labelbox,⁸ Amazon SageMaker GroundTruth,⁹ Clarifai,¹⁰ SuperAnnotate,¹¹ Computer Vision Annotation Tool (CVAT)¹² and Visual Object Tagging Tool (VoTT)¹³. Labelbox's platform is easy to use. It has a simple intuitive interface, easy project set up and a possibility of reusing ontologies. Ontologies contain all necessary information regarding a particular kind of annotation. It is useful for ensuring high-quality annotations and reducing the number of errors as well as inconsistencies for a particular annotation type. The platform also makes labeling tasks very efficient due to the availability of hotkeys – shortcuts used to perform certain tasks (e.g., copying a ground truth box to another object of similar size). With regards to the type of data that is supported for carrying out labeling tasks, Labelbox supports a diverse range of data including video, images, text, tiled imagery, etc. It also supports collaborative labeling and shows how a team is performing generally as well as on an individual basis. To ensure high quality labels, the labels that are created can be reviewed. It has other quality assurance measures such as setting benchmarks and a consensus approach. In terms of pricing, Labelbox offers a free version to small teams and individuals, especially developers. However, the free version is limited in certain ways including how many annotations can be made. To experience the full capabilities of the platform, the pro and enterprise version have been made available. A free non-commercial version is available for educational and research purposes. Furthermore, Labelbox supports manual, semi-automatic and automatic annotation methods. Different object recognition tasks such as object detection, semantic segmentation and image classification are supported. Another important consideration for the annotation tools is the output format used for the created labels. Labelbox outputs the created labels in JSON or CSV formats (Labelbox 2021).

G2, a platform on which people can find and review software, listed Amazon SageMaker GroundTruth, Clarifai and SuperAnnotate as the top 3 best alternatives to Labelbox (G2 2021). Indeed, they have many similarities to Labelbox. However, unlike Labelbox, Amazon SageMaker GroundTruth does not offer a free version for educational and non-commercial purposes. The latter only offers a free tier for the first two months of starting to use it (Amazon Web Services, Inc. 2021). Clarifai and SuperAnnotate are also very similar to Labelbox. Nevertheless, the additional possibility to set benchmarks and utilize a consensus approach to ensure high data quality, makes Labelbox's quality assurance measures more comprehensive. Although CVAT is a well-designed, web-based platform for annotating videos and images, it only works well in Google Chrome. This is because it was not optimized or tested for other browsers. VoTT on the other hand was developed by Microsoft and is a free and open-source alternative. Within the community of ML engineering and data scientists, VoTT has a good

⁸ <https://labelbox.com/>

⁹ <https://aws.amazon.com/sagemaker/groundtruth/>

¹⁰ <https://www.clarifai.com/>

¹¹ <https://superannotate.com/>

¹² <https://cvat.org/>

¹³ <https://github.com/microsoft/VoTT>

reputation. In addition, it supports a variety of annotation export formats such as CSV, generic JSONs etc. However, for it to be loaded in a browser, VoTT requires a relatively longer time since it is not light weight.

Multiple annotation tools have been compared based on factors such as ease of use, efficiency, the type of data that is supported, data quality control, pricing and available annotation file output formats. As with other ML tasks, high quality data is highly important for this study. Labelbox appears to outperform all the described tools in terms of data quality control measures due to the possibility to reusable ontologies, benchmarks, reviews and consensus. Moreover, the free non-commercial version for educational and research purposes will be suitable for this project. Therefore, Labelbox will be used for creating the labeled datasets required for the project.

2.6 Object Detection Using Deep Learning

Object detection is a principal area of computer vision which involves locating and classifying objects within a digital image or video (Zhou *et al.* 2017) with the aid of rectangular bounding boxes (Xiao *et al.* 2020). Its main goal is to develop and make use of computational models and techniques which answer the question, “What objects are in the given image or video and where are they located?” – a question that is usually required by computer vision applications (Zou *et al.* 2019).

Object detection is somewhat related to object classification, instance segmentation and semantic segmentation (Xiao *et al.* 2020). They all fall under object recognition in computer vision. However, there are some variations in how the object recognition task is accomplished and the information derived at the end. Object classification determines the category or class of an object within a given image. Object detection on the other hand, does not only determine the category of the object. It also aims to determine the location of the object in the image using a bounding box (Wu *et al.* 2020). Object detection can therefore be described as a combination of object classification and object localization (Zhao *et al.* 2019). In semantic segmentation, each pixel in the given image is assigned a class label. In comparison to object detection, semantic segmentation is unable to distinguish between multiple objects in the same category. For example, it can only predict that an object is a fruit, but it cannot distinguish between a banana and an apple. Instance segmentation accomplishes this by bringing object detection and semantic segmentation together. It also localizes objects using pixel-level localization while bounding boxes are used in object detection. These differences are illustrated in Figure 2.1. Object detection plays a vital role in computer vision and has been applied in scientific research and in practical industry production. Some of these applications include text detection, face recognition and video detection (Xiao *et al.* 2020).

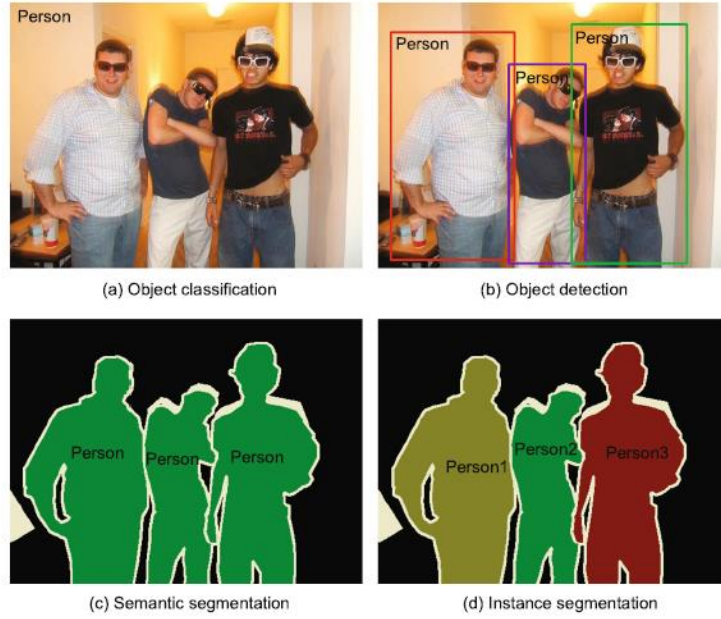


Figure 2.1: **a)** Object classification determines the category of the objects in the image; **b)** Object detection identifies the category of the objects as well as their location using bounding boxes; **c)** Semantic segmentation assigns labels of a category to the object without distinguishing between the object instances; **d)** Instance segmentation. Adopted from Xiao et al. (2020, p. 23730, Fig. 1).

For several decades, traditional object detection algorithms were used extensively. These algorithms include deformable part-based model (DPM), selective search, Oxford-MKL and NLPR-HOGLBP. Their basic architecture consists of the region selector, the feature extractor and classifier, as illustrated in Figure 2.2. At the region selection stage, the entire input image is scanned using multi-scale sliding windows. Hence, making it possible for objects of varying sizes and at different positions to be captured. The sliding window crops image blocks and forms a uniform-sized image out of those blocks. Since objects are recognizable due to certain distinct features which they possess, at the feature extraction stage, these features are extracted from the image using feature descriptors such as histograms of oriented gradients (HOG), scale-invariant feature transform (SIFT) and Haar-like features. Classifiers such as Adaboost, support vector machine (SVM) and deformable part-based model (DPM) (Zhao *et al.* 2019) are used to determine the object's category.



Figure 2.2: The traditional object detector basic architecture. Adopted from Xiao et al. (2020, p. 23731, Fig. 3).

Traditional methods used in object detection have several pitfalls including high computational cost and window redundancy due to the high number of sliding windows used. In addition, due to the multiplicity of backgrounds, appearances and lighting conditions, manually designing a

comprehensive feature descriptor which can describe objects perfectly is an arduous task. As a result, little gain was obtained from 2010 to 2012 by building ensemble systems and making use of minor variants of methods which proved to be successful (Xiao *et al.* 2020; Zhao *et al.* 2019).

In recent times, deep learning techniques (Liu *et al.* 2020; Hinton and Salakhutdinov 2006; LeCun *et al.* 2015) have proven to be valuable for automatically learning various feature representations from given data. This implies that there is no need for handcrafted features. They have led to significant improvements in object detection, as demonstrated in Figure 2.3 (Liu *et al.* 2020; Tang *et al.* 2017; Wu *et al.* 2020).

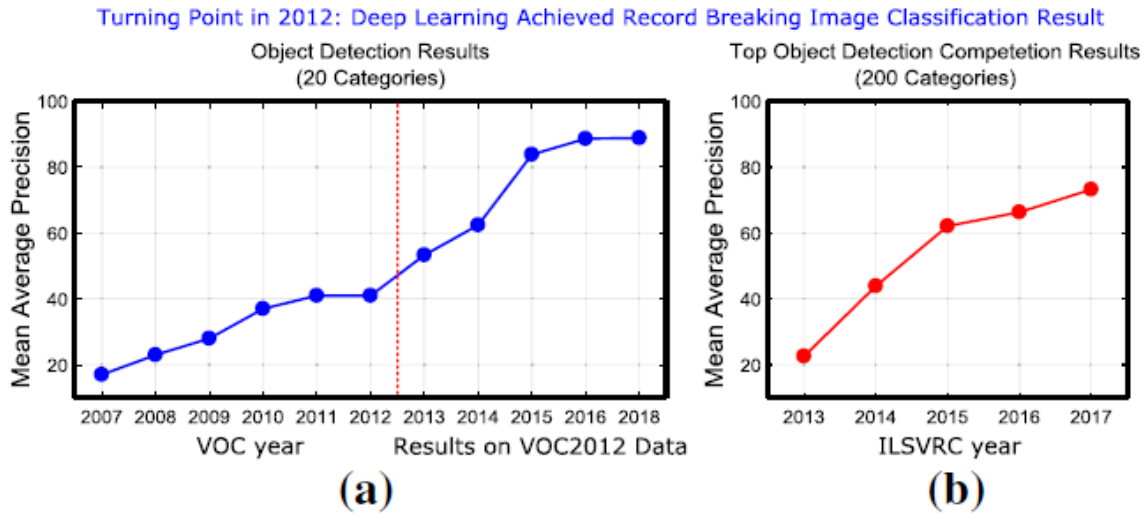


Figure 2.3: A general outlook of how object detection has performed over the years: there has been a major increase in performance (in mean average precision) since deep learning emerged in 2012; **a)** Excelling entries’ detection results in the VOC 2007-2012 competitions; **b)** Results of the top object detection competition in ILSVRC2013-2017 (in both panels, only the training data which was provided is used). Adopted from Liu *et al.* (2020, p. 262, Fig. 3).

Deep learning (DL) is a sub-field of machine learning, as illustrated in Figure 2.4. It makes it possible for computational models which possess several processing layers to imitate how the human brain sees and understands diverse forms of data. Thus, these models learn and represent data using many abstraction levels and they do this in a way that is similar to how humans perceive data. Deep learning can also be described as a large family of methods comprising of neural networks, diverse feature learning algorithms (supervised and unsupervised), and hierarchical probabilistic models (Voulodimos *et al.* 2018).

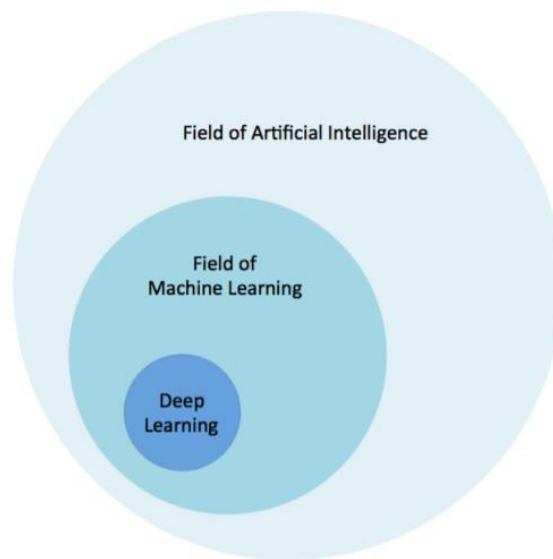


Figure 2.4: Deep learning as a sub-field of machine learning which is also a sub-field of artificial intelligence. Adopted from Patterson and Gibson (2017, p. 4, Figure 1-1).

Artificial neural networks (ANNs) are made up of many interconnected computational nodes which are also referred to as neurons. These neurons work together to learn from input data to improve the final output. ANNs typically have a basic structure which comprise an input layer, hidden layers and an output layer, as shown in Figure 2.5.

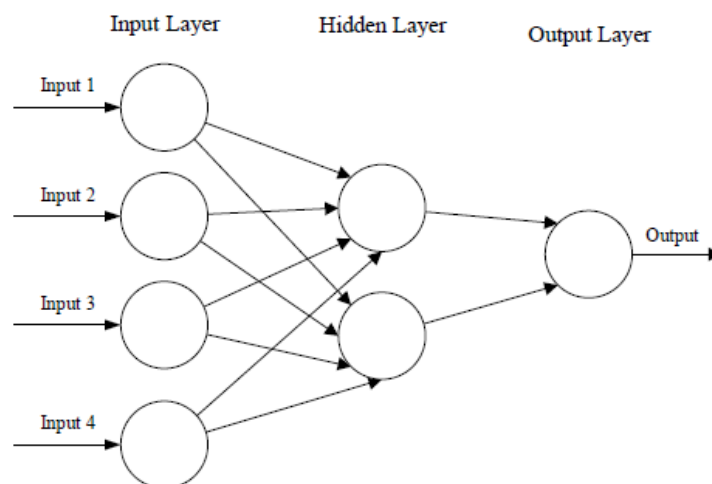


Figure 2.5: The basic structure of an Artificial Neural Network. Adopted from O'Shea and Nash (2015, p. 2, Fig. 1).

The input layer takes in the input which is often multidimensional in nature. It then assigns the input to the hidden layers. The hidden layers proceed to assign weights based on decisions made in the previous layer. These weights are randomly changed, and the final output is observed to understand how a change affects it – whether it deteriorates or improves. This process is what is commonly known as learning. The concept of deep learning in neural networks comes in when many hidden layers are stacked together (O'Shea and Nash 2015). In the field of deep

learning, the convolutional neural network (CNN) is the most representative model (Zhao *et al.* 2019).

2.6.1 Convolutional Neural Networks

Convolutional neural networks are comparable to traditional artificial neural networks in that they are made up of interconnected neurons. The arrangement of these neurons is acyclic (Aloysius and Geetha 2017). In addition, similar to traditional ANNs, CNNs improve their final output through learning (O'Shea and Nash 2015). The major difference between CNNs and traditional neural networks is that a neuron in the hidden layer of the CNN is connected to some of the neurons in the previous layer, rather than to all the neurons in the previous layer. As a result of this sparse connectivity, CNNs are capable of learning features in a comprehensive manner. Its deep architecture enables features to be extracted in a hierarchy, with different layers performing different functions. For example, the trained filters for the first layers can be visualized as colour blobs or a set of edges (e.g., lines). The trained filters for the second layers extract lower-level features in a combined manner, for instance, a combination of several lines to express shapes (Brownlee 2019; Aloysius and Geetha 2017). The subsequent layer may learn object parts while the final layers' filters determine what the objects are (Aloysius and Geetha 2017).

Deep CNNs have a basic architecture which comprise of a series of convolutional, pooling, non-linear activation, and fully connected (FC) layers (Wu *et al.* 2020). The basic architecture of a convolutional neural network is demonstrated in Figure 2.6.

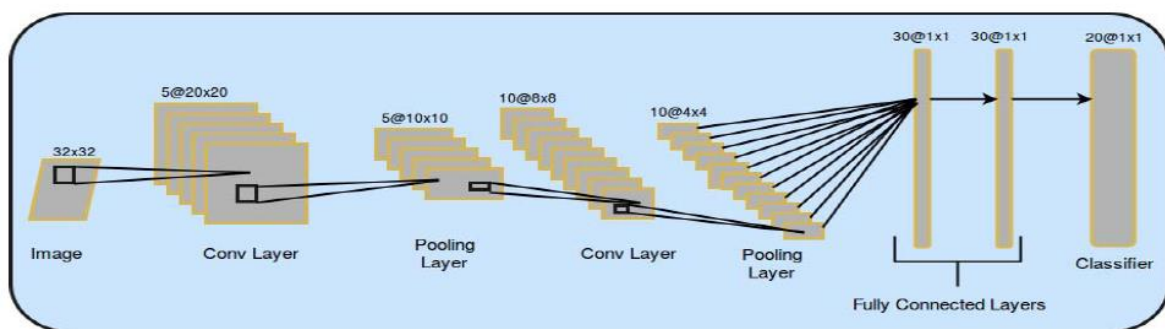


Figure 2.6: The basic architecture of a convolutional neural network demonstrating a sequence of alternating convolutional and pooling layers. The receptive regions are shown using highlighted small boxes. The connections illustrate how features are learned implicitly. Adopted from Aloysius and Geetha (2017, p. 0589, Fig. 1).

2.6.1.1 Convolutional Layer

The convolutional layer is the most basic unit of the convolutional neural network where the majority of the computation is done (Aloysius and Geetha 2017). This layer convolves over a received input image by $n \times n$ kernels in order to create a feature map. The convolution process usually involves the computation of the dot product of associated receptive region elements and filter (Wu *et al.* 2020; Brownlee 2019), as demonstrated in Figure 2.7. This implies that for each element, the input and filter are multiplied, and the result is summed up. As a result, a single value is obtained. Due to the single value derived at the end of this operation, it is often referred to as “scalar product”. The feature map which is generated can be seen as an image

with multiple channels. Each channel in the image symbolizes different information within the image. Pixels in the feature map, also known as neuron, are individually connected to a subset of neurons from the preceding map (Wu *et al.* 2020). This sparse connectivity between the neurons of adjacent layers represents a hyperparameter known as receptive field. Several hyperparameters are responsible for regulating the output volume. These include the depth (to regulate how many filters a layer has), stride (responsible for the movement of filters), and zero-padding (regulates the output's spatial size) (Aloysius and Geetha 2017).

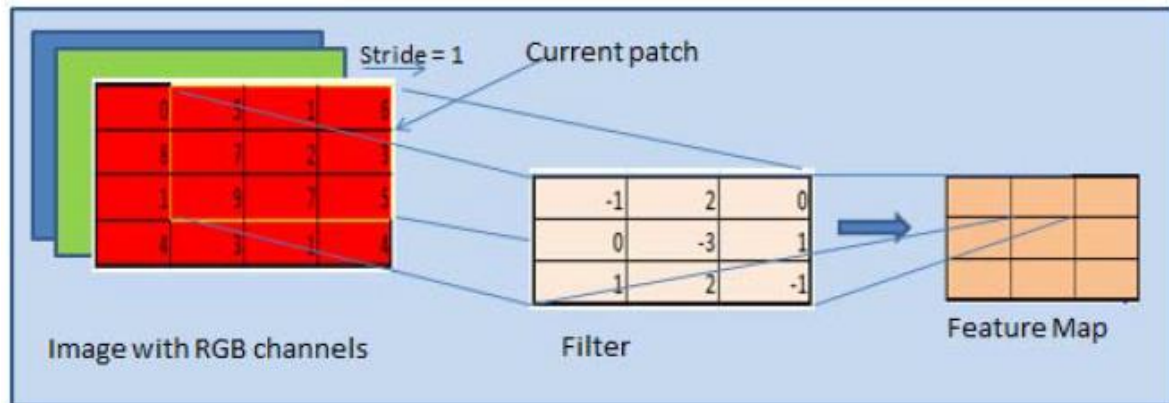


Figure 2.7: The process of convolution on an RGB image. Adopted from Aloysius and Geetha (2017, p. 0589, Fig. 2).

2.6.1.2 Pooling Layer

The basic architecture of a CNN is made up of alternating convolutional and pooling layers as well as additional functions. This results in a reduction of the spatial dimension of the activation maps while preserving all information. The number of parameters in the neural network is also kept to a minimum. This offers the advantages of a general reduction of computational complexity and regulation of the problem of overfitting. Average pooling and max pooling are examples of common pooling operations. The process of max pooling is shown in Figure 2.8.

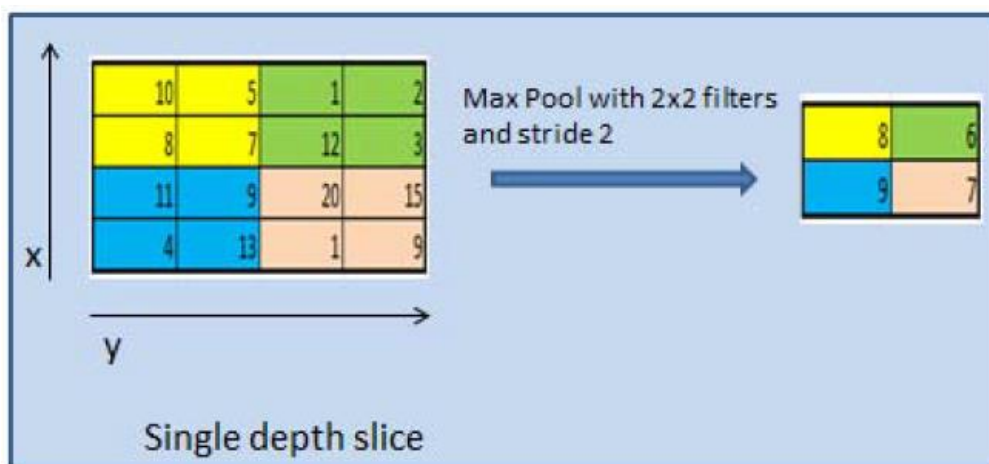


Figure 2.8: The process of max pooling. Adopted from Aloysius and Geetha (2017, p. 0589, Fig. 3).

2.6.1.3 Fully Connected Layer

The fully connected (FC) layer is so called because the neurons included in this layer are connected fully to the neurons in the preceding layer. In this layer, high level reasoning is carried out.

2.6.1.4 Loss Layer

This is the last FC layer. It is responsible for computing the error or loss resulting from inconsistencies between the desired output and the actual output. Softmax loss is a widely used loss function which predicts one class out of N mutually exclusive classes.

2.6.1.5 Activation Functions

Activation functions take an input (often a single number) and perform some mathematical operation on the received input. This implies that the operation of the activation function influences the output. There are many activation functions including sigmoid, Tanh, ReLU and LeakyReLU (Aloysius and Geetha 2017).

2.6.2 Deep Learning-based Object Detection: Algorithms

Presently, the deep learning algorithms that are used for object detection tasks, can be categorized into the following two options:

- Two-stage detectors: For example, Region-based CNN (R-CNN) and its revised versions (Wu *et al.* 2020)
- One-stage detectors: For example, You Only Look Once (YOLO) and its revised versions (Wu *et al.* 2020)

The two-stage detectors operate by first generating a random set of region proposals and extracting features from the generated region proposals. The goal of this first stage is to generate region proposals which have a high recall. This ensures that all the objects present in the image are contained within at least one of the generated region proposals. This initial step is then followed by a classification of each of the region proposals by a classifier (a deep learning-based model). The model may be classified using a predefined class label or classified as a background. The model also improves the initial localization recommendations made by the proposal generator. The processes undergone by the two-stage detectors imply that a single image may be viewed multiple times and hence increase the time required to complete the task at hand (Wu *et al.* 2020). Figure 2.9 illustrates how the R-CNN, which is a two-stage detector operates.

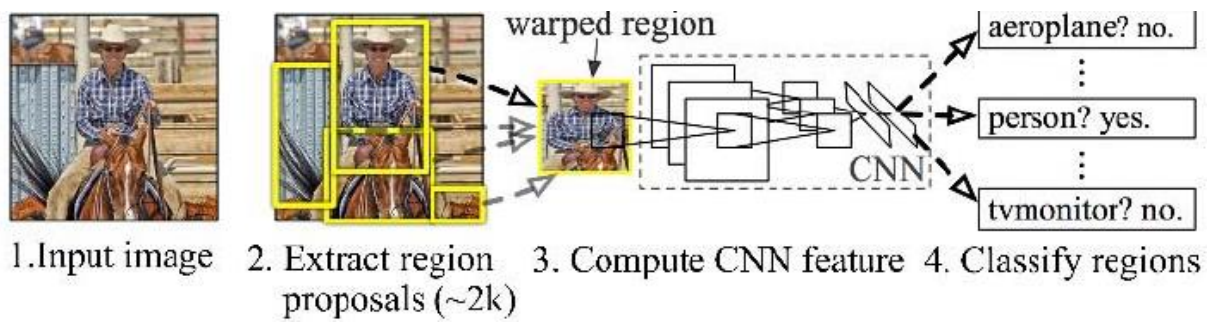


Figure 2.9: The operation principle of an R-CNN framework. Adopted from Tang et al. (2017, p. 725, figure 3).

On the other hand, one-stage detectors can predict the categories of objects at every location of the feature maps in a single step. These detectors do not generate region proposals. Instead, they treat all the positions on a given image as prospective objects. Therefore, all the regions of interest (ROI) are classified individually. The ROI may be a background or a target object. From its operations, it implies that the image is only viewed once, and the categorical prediction of objects is done in that single step. This makes them a more time-efficient option. In addition, they are known to achieve real time detection.

While the two-stage detectors do not achieve real-time detection like the one-stage detectors, they are widely known to have better detection performance. On public benchmarks, the two-stage detectors record state-of-the-art results (Wu *et al.* 2020). Nonetheless, for this study, the one-stage detectors are preferred due to their time-efficiency and real time detection application which is suitable for videos.

There are several one-stage detectors including YOLO, its variants, Single Shot MultiBox Detector (SSD) and RetinaNet. The first one-stage detector, YOLO, was proposed in 2015 by Redmon et al., (2016). YOLO outperforms the two-stage detectors in terms of speed. It has a VOC07 mean Average Precision (mAP) of 52.7% and runs at 155 fps. Its enhanced version has a VOC07 mAP of 63.4%, VOC12mAP of 57.9% and runs at 45 fps. As its name implies, YOLO applies one neural network to the whole image. The image is then divided into regions. Bounding boxes and probabilities are predicted for each region. These are carried out simultaneously by the network, as demonstrated in Figure 2.10. The improved versions of YOLO, as proposed by R. Joseph (Redmon and Farhadi 2017; Redmon and Farhadi 2018) are known as YOLOv2 and YOLOv3. These versions improve detection accuracy while maintaining an extremely high detection speed. Although YOLO outperformed the two-stage detectors, its localization accuracy is lower compared to the two-stage detectors. However, YOLOv2 and v3, as well as the SSD, give more attention to this problem (Zou *et al.* 2019). Due to Joseph Redmon's (Redmon and Farhadi 2018) dissatisfaction with the military application of his open-source algorithm as well as privacy concerns, he chose to discontinue computer vision (CV) research. Alexey Bochkovskiy and a team from Ultralytics LLC proposed YOLOv4 and YOLOv5 respectively. Alexey's work has received an official approval from the original author of YOLO (Yang *et al.* 2020). YOLOv4 is said to improve YOLOv3's performance. It increases its Average Precision (AP) by 10% and FPS (frame per second) by 12% (Bochkovskiy *et al.* 2020). Although YOLOv5 has not received an official approval from the original author of YOLO, its usefulness remains unaffected. When compared with other

YOLO versions especially YOLOv4, YOLOv5 achieves 140 FPS while YOLOv4 achieves only 50 FPS on Tesla P100 rapid detection. Moreover, with a size of 27 MB, YOLOv5 proves to be smaller than YOLOv4 which uses Darknet architecture and has a size of 244 MB. In terms of accuracy, there is no difference between YOLOv4 and YOLOv5. Both have the same accuracy. Furthermore, YOLOv5 has taken on YOLOv4's advantages, namely adding SPP-Net, making changes to the SOTA method, and introducing modern data enhancement approaches such as mosaic training (Yang *et al.* 2020).

SSD made its debut as the second one-stage detector in 2015. It was proposed by W. Liu et al. SSD is fast and has an advantage in terms of detection accuracy. Using some benchmark metrics, SSD is said to run at 59 fps and has a VOC07 mAP of 76.8%, VOC12 mAP of 74.9%, COCO mAP (at .5) of 45.5% and mAP[.5, .95] = 26.8% (Zou *et al.* 2019). However, from Table 2.1, YOLOv3 performs better than the variants of the SSD on most of the metrics including the AP_S metric. On the AP_{50} metric, YOLOv3 is comparable to the alternative models shown (Redmon and Farhadi, 2018).

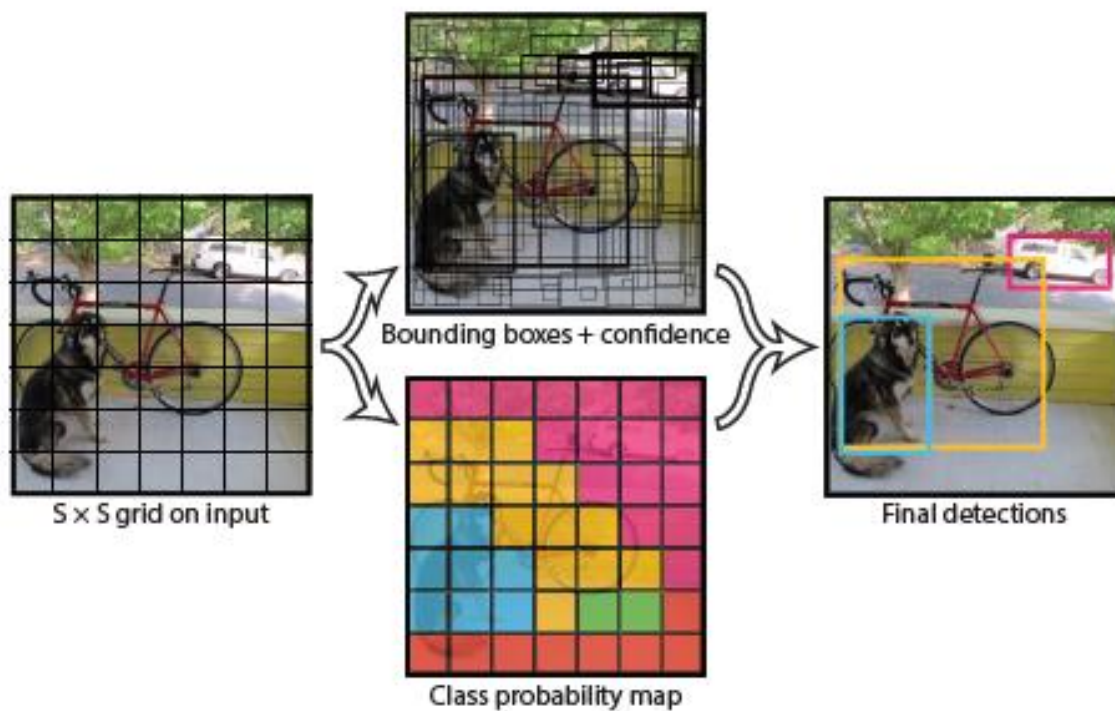


Figure 2.10: The operation of the YOLO model. Adopted from Redmon et al. (2016, p. 780, Figure 2).

Table 2.1: Comparison of selected two-stage methods including Faster R-CNN, and one-stage methods including YOLOv3, SSD, and RetinaNet. Adopted from Redmon and Farhadi (2018, p. 3, Table 3).

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

RetinaNet is another popular one-stage detector which was introduced by T.-Y. Lin et al. in 2017. With the introduction and adoption of a new loss function known as “focal loss”, the detectors were made to focus more on the examples that were misclassified during training. This makes it possible for the one-stage detector to achieve accuracies that are comparable to that of the two-stage detectors, while still running at very high detection speed (Zou *et al.* 2019).

As demonstrated in Table 2.1, RetinaNet has a higher AP compared to the other models. However, when the inference speed is taken into consideration, YOLOv3 appears to offer a good trade-off between accuracy and detection speed compared to RetinaNet, as illustrated in Figure 2.11. On a Titan X, YOLOv3 runs in 51 ms and achieves 57.9 mAP₅₀ while RetinaNet runs at a speed of 198 ms and achieves 57.5 mAP₅₀. Therefore, the two models achieve similar performance but YOLOv3 is 3.8x faster (Redmon and Farhadi, 2018).

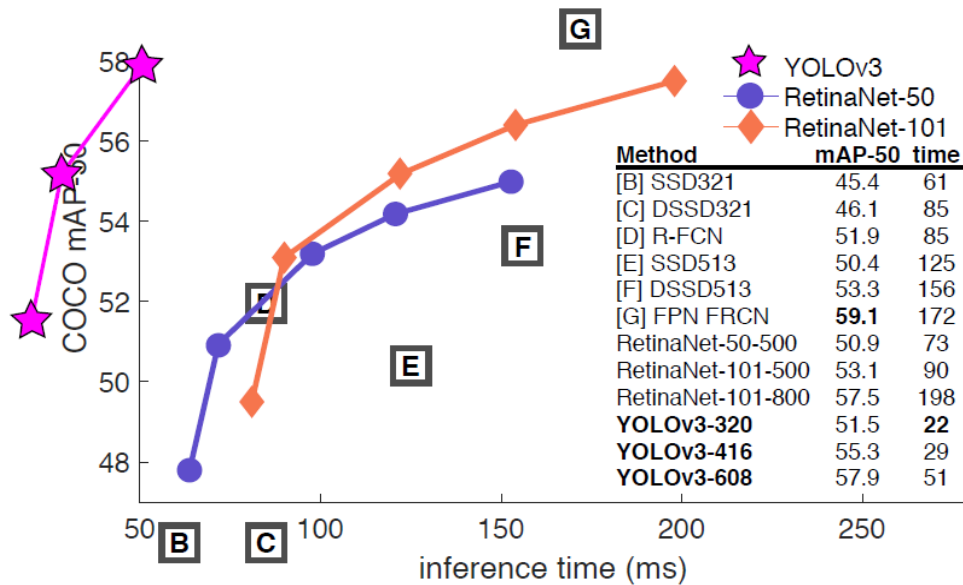


Figure 2.11: The trade-off between accuracy and speed for selected models including YOLOv3 and RetinaNet. Adopted from Redmon and Farhadi (2018, p. 4, Figure 3).

2.6.3 Deep Learning Frameworks for CNN Implementation

A myriad of deep learning frameworks exists and can be used for implementing deep learning models. Out of these, Google's Tensorflow¹⁴ is considered to be the fastest growing framework. Tensorflow (Abadi *et al.* 2016) is an open-source machine learning platform where high computational tasks can be performed using data flow graphs. The edges in the graphs represent tensors. With this, load can be distributed between several nodes (GPUs or CPUs) using a single API. This framework has been available to the public since November 2015. The second fastest growing DL framework is Keras.¹⁵ It is also open-source and is written in Python. Keras can run with Theano or Tensorflow as backend. Theano (The Theano Development Team *et al.* 2016) is an alternative open-source framework written in Python.¹⁶ It can be used for numerical computations and makes the writing of DL models simpler. Caffe (Jia *et al.* 2014),¹⁷ another DL framework, was developed by Berkeley Vision and Learning Center (BVLC). Caffe has already existing Python-based deep learning examples. Furthermore, there is a framework known as Torch. It places more importance on GPUs and has an underlying C implementation. Torch, developed by Torch, and Matlab's matconvnet are popular deep learning frameworks (Aloysius and Geetha 2017). Moreover, PyTorch¹⁸ is a python library that was developed based on Torch. It was designed such that a balance between speed and ease of use can be achieved. Since it leverages python, a programming language that many data scientists are familiar with, it is simple to use. In addition, it makes use of GPU acceleration. Thus, it performs fast. It does so while not compromising on performance, making it comparable to the fastest DL frameworks (Paszke *et al.* 2019).

2.6.4 CNN Training Using Transfer Learning

The process of training a deep convolutional neural network is laborious because of non-convex loss functions. When large training datasets are to be used for the deep learning task, then high computing speed (possible with GPUs) will be required. As a result, CNNs are seldom trained from the ground up. With the right network initialization, convergence can be sped up. If a significant weight initialization is desired or a small training dataset is available, then transfer learning can be employed. In transfer learning, only the last few layers or the classifier of the neural network undergo retraining. This aids in the reduction of the overall training time. A good model can be obtained if parameters such as the number of iterations, learning rate, batch size and the amount of training and testing samples are carefully chosen (Aloysius and Geetha 2017).

¹⁴ <https://www.tensorflow.org/>

¹⁵ <https://keras.io/>

¹⁶ <https://pypi.org/project/Theano/>

¹⁷ <https://caffe.berkeleyvision.org/>

¹⁸ <https://pytorch.org/>

2.7 Best Practice Examples

2.7.1 Non-ML-based Detection of Physical Urban Blight

Many governments, decision makers, urban planners, researchers, health workers and other institutions are interested in collecting data about urban blight for varied purposes. These may include utilising urban blight data as the basis for important decisions about budget allocations, planning of urban areas, testing of hypothesis or theories such as the Broken Windows Theory, determining the impact of urban blight on the health of people within a community as well as to guide policing efforts. Owing to its relevance in diverse fields, several approaches have been adopted to collect urban blight related data.

In the attempt to fight urban blight, many cities previously resorted to occasional ground-level inspections and arbitrary detection of blight by city officials within their borders. The ground-level inspections involve a city official performing rounds within assigned locations and taking note of locations where blight indicators are observed. Some blighted properties may also be arbitrarily spotted by the city officials in other locations and noted accordingly (Pough and Wan 2007).

Methods which have been adopted previously by researchers to assess the level of disorder in a public place include the Systematic Social Observation (SSO) method and the self-report approach. With regards to the SSO method, Sampson and Raudenbush (1999) employed analogue videos to collect data about physical (e.g., broken windows, boarded windows, graffiti, etc.) and social disorder indicators (e.g., public harassment, open solicitation prostitution, etc.). These indicators could be systematically observed and extracted from the video recordings. However, the collected data cannot be readily integrated into a GIS software for further spatial analysis. This is due to the absence of a link between the images and coordinates. The spatial video technology provides an improvement in this area since it makes use of Global Positioning System (GPS) sensors. This allows both the spatial and temporal aspects to be captured easily. The data can therefore be integrated in a GIS software for further visualization and analysis (Mills *et al.* 2010; Curtis *et al.* 2013a).

Varying self-report approaches - residents' providing information pertaining to their neighbourhood - have also been used on a wide scale. One self-report approach involves the collection of individual perceptions about fear of crime and safety in public places from a sample of residents who are deemed as reliable (Marco *et al.* 2017; Stratmann 2019). Due to the subjectivity of such data, an index known as the "Ross-Mirowsky neighbourhood disorder scale" is used in the data collection process. This index comprises of indicators which cover physical and social disorder. The selected residents evaluate pre-defined statements regarding disorder in their neighbourhood using this index. For increased objectivity, the collected data is compared to more objective measures such as census tracts which may include information about poverty, population density and crime rates (Ross and Mirowsky 2001; Stratmann 2019). This approach poses a risk for the research staff since they are required to be present in areas where crime is prevalent and to engage in conversations with total strangers. Additionally, it has not proven to be time-efficient (Stratmann 2019; Stratmann *et al.* 2020).

Another self-report approach adopted by some cities over the years includes the collection of data from residents via 311 calls. Since residents are more likely to encounter blight indicators

during their day-to day-activities, cities have resorted to leveraging on calls and complaints from residents in a neighbourhood. With the advent of technology and the increased use of smartphones, more cities are encouraging residents to actively participate in the provision of blight data through crowdsourcing applications. Residents can therefore send photographs and communicate their impressions of housing blight via these applications within the speed of light. Nonetheless, the quality of such data is a major issue for consideration. A typical use case is seen in Detroit's blight tracking initiative. This initiative comprises of citizens sending photographs and personal impressions of housing blight through the 'Blexting' app, an activity which has come to be known as "blexting" in Detroit, USA. Subsequently, the crowdsourced data is cleaned and put into a standard format by data analysts. The standardized data is then combined with other data such as data from fire departments and code violations. This is done to provide a different perspective which may prove useful to understanding the occurrence of blight in the city of Detroit (Pough and Wan 2007).

2.7.1.1 State of-the-art Methods to Detect Urban Blight

A recent study involving the use of Google Street View (GSV) to remotely conduct systematic observation has proved to be cheaper, safer, and scalable. This serves as an alternative approach to the use of analogue videos for systematic observations as conducted by Sampson and Raudenbush (1999). GSV is easily accessible and provides high resolution 360° images of a myriad of locations worldwide. Nonetheless, there are limitations with regards to the stability and flexibility of the temporal components of the images (Curtis *et al.* 2013b; Marco *et al.* 2017).

A relatively novel approach involving Spatial Video Acquisition System (SVAS) was employed to collect urban blight data within selected neighbourhoods in Baton Rouge, USA. The SVAS, also known as spatial videos, enabled spatio-temporal data to be collected at a single instance. GPS sensors were linked to the spatial video to ensure that the geographical information of a recorded location was appropriately assigned to each recording frame. Each recording frame also had a timestamp. The spatial videos were shown to be a cost-effective and reliable approach to collecting exhaustive and standardized urban blight data. The combination of both the spatial and temporal aspects in the spatial videos, made the integration of the collected data into a GIS program relatively simpler (Stratmann 2019; Stratmann *et al.* 2020). The spatial video offers a myriad of benefits compared to other methods. In addition to the above-mentioned advantages, the utilization of spatial videos reduces the time spent by research staff gathering data in the field. When contrasted with Google Street View, the spatial video technology offers researchers the opportunity to control the data collection and utilize the acquired data in a more flexible manner (Curtis *et al.* 2015).

To carry out further analysis on the collected data, the spatial videos were watched and paused each time an urban blight indicator was observed. The indicators were determined based on a pre-defined requirements catalogue. Subsequently, the location at which the urban blight indicator was identified was then digitized using a GIS program. This required an intensive effort and a lot of time on the part of the researcher. This manual detection and classification process also introduced subjectivity into the data (Stratmann 2019; Stratmann *et al.* 2020).

Urban blight is a phenomenon which has affected over 10,000 homes in various cities. Considering the high presence of urban blight over widespread geographical locations, it is

advisable to reduce the time, resources as well as the effort required in detecting urban blight (Pough and Wan 2007). Therefore, to reap the benefits offered by spatial videos while cutting down on the extraction of urban blight locations from the spatial videos, it is necessary to automate the detection and classification of urban blight from spatial videos. Machine Learning (ML) offers automation possibilities which can be leveraged on to easily detect and classify urban blight indicators from spatial videos. This will enable researchers who are interested in conducting urban blight related studies to easily access details of urban blight within their area of choice and to carry out further analysis with ease and in a time-efficient manner.

2.7.2 ML-based Detection of Physical Urban Blight

Some researchers have sought to apply ML to automatically detect urban blight, thereby reducing the human interventions required. Some studies have shown that ML can be applied in the automatic detection of urban blight as well as in predicting the risk of urban blight.

A number of cities have resorted to collecting urban blight data via 311 calls. Athens *et al.* (2020) assessed how data from the 311 call system could be used to display urban blight at fine-grained geographies within various periods of time. They applied natural language processing (NLP), an ML technique, to develop an algorithm which can detect and classify 311 data as urban blight-related calls, as well as to investigate the distribution of these calls across New York City. In addition, they allowed a group of people called ‘raters’ to manually categorize the 311 calls. Each call was assigned one of the pre-defined categories including social conditions, air quality, deserted properties, sanitary conditions, sidewalk maintenance and building safety. To test for the accuracy of their algorithm, they compared the categorization results of the NLP algorithm with that of the manual categorization. It was observed that both results were consistent with each other.

Reyes *et al.* (2016) proposed a predictive approach to enable city officials to conduct city inspections in a prioritised manner. Rather than carrying out inspections at locations which did not have urgent need of such inspections, the city officials desired to maximize the inspections carried out. Using a combination of geographical and historical data of the City of Cincinnati in the USA, an ML model was developed and trained to provide recommendations of properties at risk of urban blight by means of a ranked list of all such properties. This led to more targeted and prioritised inspections which in turn reduced the time and cost required for the inspections.

2.7.3 Other Related Work: Object Detection

Urban blight related studies are not very common. Therefore, it is worth selecting and taking a look at individual physical urban blight indicators to have an overview of research directions for the selected indicators. A typical example of physical urban blight indicator which has been tackled in various studies is litter.

The methods used to monitor garbage and waste demand intensive human effort. To support smart city development and initiatives, Carolis *et al.* (2020) developed YOLO TrashNet – a YOLOv3 model that was trained to detect waste in video streams. By applying YOLO TrashNet to the video streams, the model could identify any waste around garbage dumpsters or bins. This helped to determine localities that required cleaning or not. Good prediction speed and accuracy results were achieved, indicating that the model could successfully perform the

desired detection task. Ping Ping *et al.* (2020) also developed a deep learning model, Faster R-CNN, to analyse street imagery to automatically detect, classify and analyse various categories of litter including tree branches, leaves, bottles etc. The Faster R-CNN model was discovered to be effective and showed a strong potential in smart city applications. Nevertheless, both studies were not applied to very small objects. The question regarding how ML models can be used to detect small objects effectively remains. There is therefore an interest to explore the use of a deep learning model to detect small objects in videos.

Due to the possibility of capturing fine-scale and contextual data using spatial video technology, spatial videos are useful for identifying health risks. Nonetheless, their sustainability and scalability are limited as a result of the intensive effort needed to convert the spatial video frames into maps. Ajayakumar *et al.* (2021) applied YOLOv3 to spatial videos collected from Haiti. This was done to identify potential health risks automatically. The results showed that ML can be used in conjunction with spatial videos to identify environmental risks that have a correlation with common health problem. The test sites in focus were informal settlements. To develop a more sustainable method of creating maps from spatial videos and updating them, it is necessary to first identify the risk features automatically. Subsequently, the locations of the identified risk features can be mapped. However, Ajayakumar *et al.* (2021) focused on automatically extracting the risk features from spatial videos in their study. Therefore, there is still a need to explore how maps showing the locations of automatically identified features can be created from spatially encoded video frames.

In conclusion, the Broken Windows Theory and the urban blight concept have been discussed in this chapter. Varying perspectives of the BWT were also reviewed. Although there were divergent views, it was realized that there was a correlation between crime and urban blight, although the full extent of this relationship had not yet been determined. The spatial video technology was also introduced and was seen to be useful for collecting fine-scale data within their spatial context. A limitation of the spatial videos was identified as being the laborious effort and intensive time required to convert the spatially encoded video frames into maps. To reap the full benefits of the spatial video, automating the mapping process appeared to be a promising direction. Since ML offers automation capabilities, the concepts of ML and object detection using deep learning were introduced. The state-of-the-art tools required in the object detection process including frame extraction software and tools used for labelling data were also reviewed. Furthermore, deep learning algorithms were assessed to identify a suitable one for this study. From these concepts and reviews, it was concluded that ML may be applicable to spatial videos to detect physical urban blight in the spatial videos. VLC Media Player was identified as a good alternative for extracting many frames from the spatial videos due to its ease of use and the availability of other media player functions relevant for pre-processing the video (e.g., trimming the video due to file size limits). For the purposes of mapping the identified physical urban blight, it may be necessary to identify the video frames used in terms of their timestamp. Therefore, the Frame Selector software will be suitable for this since the frames are automatically named by the video name and timestamp of the extracted video frame. Labelbox was also identified as the best alternative for labelling the extracted frames to make up the training, validation and test datasets. From the assessments made, Labelbox showed to have a more comprehensive quality control mechanism compared to other competitors such as

Amazon Sagemaker GroundTruth, SuperAnnotate and Clarifai. YOLOv3 was seen to be a suitable option out of the other deep learning algorithms such as Faster R-CNN due to its good trade-off between accuracy and speed. Moreover, some best practice examples were reviewed. From these reviews, applying ML to detect urban blight as well as small objects, and convert spatially encoded video frames into maps were identified as potential research directions. Therefore, this study may prove to be valuable in answering questions in the identified research directions.

3 Methodology

This chapter presents the conceptual workflow to be used in this study. The spatial video data collection, data preparation and architecture of the selected model are described. The concepts of training, validating, testing and evaluating deep learning models are also explained in general terms. Furthermore, evaluation metrics that are frequently used to evaluate object detection models are introduced. The chapter concludes with a brief description of how the detected physical urban blight indicators will be visualized in a GIS environment.

3.1 General Project Workflow

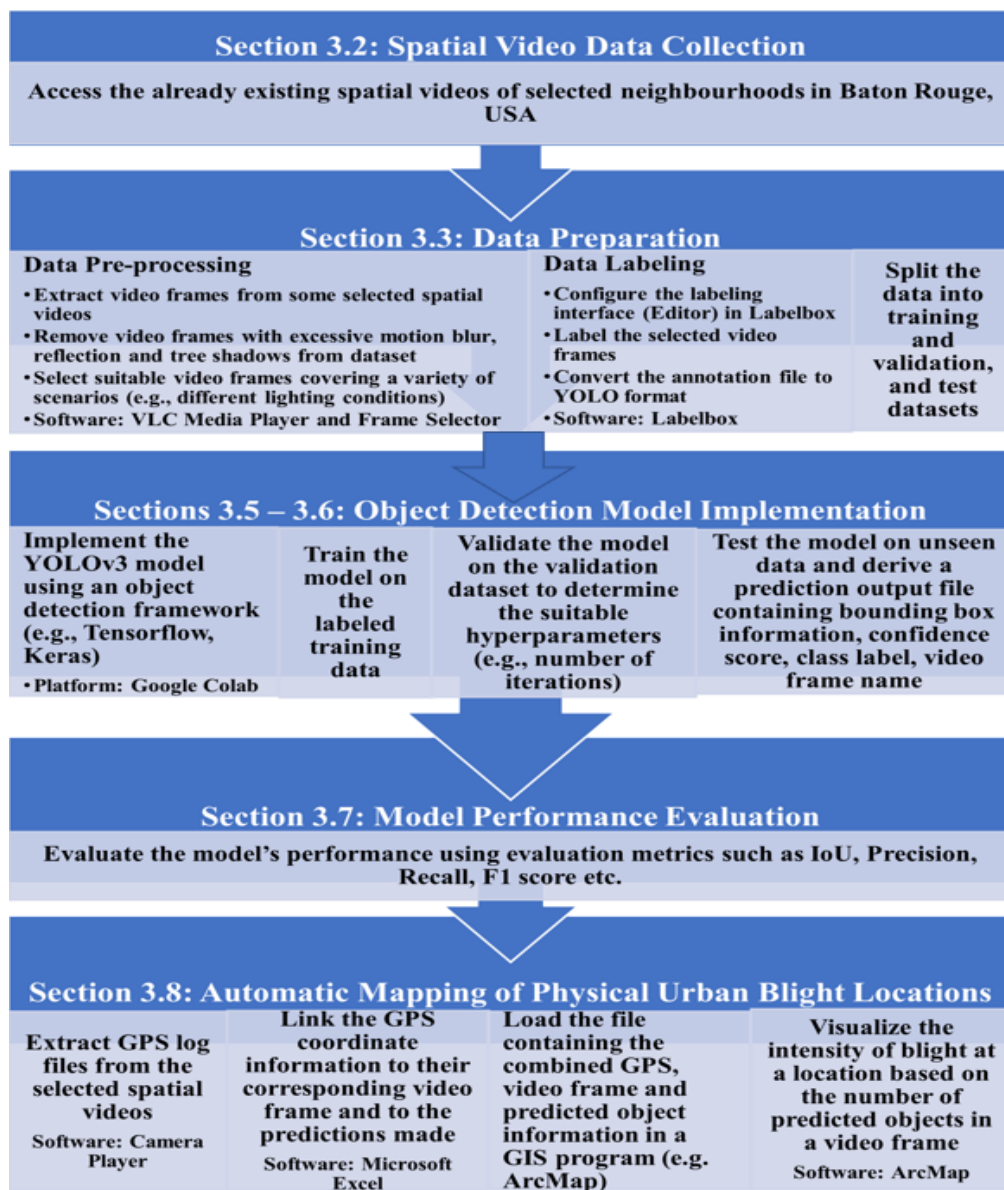


Figure 3.1: The Project Workflow

Figure 3.1 represents the conceptual workflow for this research project. The project will begin with gaining access to the spatial videos collected by Stratmann (2019). Once the data is obtained, some of the spatial videos will be selected. Video frames will be extracted from these video frames, and some will be selected depending on how suitable they are. Subsequently, the selected video frames will be labeled and split into training, validation and test data. The model will then be trained, validated and tested on these datasets. While the training and validation datasets will be made available to the model during its learning phase, the test dataset will be kept away from the model until the testing phase. The model will then be evaluated using evaluation metrics that are widely accepted for object detection tasks. Finally, a map of showing the physical urban blight indicator locations will be generated. Further details of the afore-mentioned steps are provided in the next sub-sections.

3.2 Spatial Video Data Collection

In this study, secondary data will be used. The data were collected during the research conducted in (Stratmann 2019; Ristea *et al.* 2021; Stratmann *et al.* 2020). The data collection process will be described in this section since it is an important part of this research.

The spatial video technology makes it possible to collect video data which include the geographical location on a frame-by-frame basis. Therefore, the identified indicators that are relevant for physical urban blight studies in Baton Rouge are collected using the spatial video technology. These indicators are summarized in the criteria catalogue in Table 3.1. In the criteria catalogue, a distinction is made between environmental or infrastructural blight, and property blight.

Table 3.1: The criteria catalogue for the physical urban blight indicators. Adopted from Stratmann (2019, pp. 16-17, Table 2).

Physical urban blight type	Description	Source
1. Properties		
Abandoned property	Properties that show signs of decay or abandonment (e.g. doors are boarded up); Nobody lives in the house.	(Weisburd et al. 2010; Longley et al. 2015; Maghelal et al. 2013)
Broken window/door	Broken windows/doors are left unrepaired.	(Weisburd et al. 2010; Kelling and Wilson 1982; Pasadena community development commission 2010)
Boarded/covered window/door	Windows are boarded up with plywood or other material or covered with plastic, foil, etc.	
No glass in window/door	Windows/doors without glass and are left unrepaired.	
Building graffiti	Graffiti on buildings. <i>Note: Graffiti that is not situated on buildings is considered as infrastructural graffiti in environmental/infrastructural blight.</i>	(Weisburd et al. 2010)
Structural integrity	Severe cracks in the foundation of the building's structure; holes in plaster, wood, masonry, rooftop; unstable foundations.	(Pasadena community development commission 2010)
Building overgrowth	Unsafe amount of vegetation touching the building; constitutes fire, health, or safety hazard. <i>Note: overgrown vegetation that does not touch a building is considered as overgrown vegetation in environmental/infrastructural blight.</i>	(Pasadena community development commission 2010)
Other (specify)		
2. Environment/Infrastructure		
Damaged sidewalk	Structural failure in pavement what is left unrepaired.	(Gau and Pratt 2010; Ross and Mirowsky 2001)
Damaged roads	Structural failure in a road surface (e.g. potholes in asphalt pavement).	(Maghelal et al. 2013)

Overgrown vegetation	Any vegetation that appears to be overgrown in garden/vacant lots etc.; May block sidewalks. <i>Note: in contrast to building overgrowth, this indicator does not appear on buildings, but exclusively in empty spaces or on infrastructure.</i>	(Maghelal et al. 2013)
Litter	Single pieces of trash that are not within a garbage disposal bag; trash left on the ground in a public place.	(Weisburd et al. 2010)
Illegal dumping	Trash piles: unlawful deposit of any type of waste material (e.g. construction materials, tires, asbestos, etc.).	(Maghelal et al. 2013)
Unkempt areas	Empty areas (e.g. vacant lots) that appear unkempt (e.g. overgrown grass, trash, etc.).	(Maghelal et al. 2013)
Illegal parking	Car parked illegally on sidewalk; should be reported to city agency for removal.	(Weisburd et al. 2010)
Abandoned vehicle	Number plates missing, flat tires, missing wheels, broken windows, etc.; Any vehicle that has been parked illegally for >72 hours.	(Weisburd et al. 2010)
Infrastructural graffiti	Graffiti on walls or other surfaces. <i>Note: graffiti on buildings is considered as building graffiti in property blight.</i>	(Weisburd et al. 2010)
Other (specify)		

The physical urban blight data were recorded using five extreme sport cameras. The cameras were from the Contour +2 brand. They were mounted on the windshield and windows using suction window clamps. The setup was done as follows: one camera on the inside windshield (Figure 3.2c), two cameras on the right and on the left inside windows (Figure 3.2a, b). Multiple cameras were used to ensure that a backup was available if one of the cameras failed to record properly or if a GPS connection was lost.

Due to the high crime level in the selected neighbourhoods in Baton Rouge, the cameras were mounted inside the vehicle for an unobtrusive data collection. By connecting GPS sensors to the Contour +2 cameras, the geographical location of each recorded video frame was obtained. The camera's battery lasted for a duration of approximately two hours while in power mode. This can be extended by connecting the SVAS setup to the car's charger (Figure 3.2d).

Factors such as the weather condition and the vehicle speed were taken into consideration to ensure high data quality. Therefore, the data were only collected in good weather conditions where there was no precipitation. In addition, the drives were conducted during the daytime for increased visibility. An average speed of 26.29 km was used during the drives. High speed was avoided because of its potential impact on the quality of the video. As a result, all the streets of the selected neighbourhoods were covered, with the exception of highways. Over a period of 8

days, a total of 383.84 km was driven during the data collection. This was achieved in 14 hours and 36 minutes. Table 3.2 contains the details of the distance covered, how long the data collection process lasted (duration) and the dates on which the data collection was done.

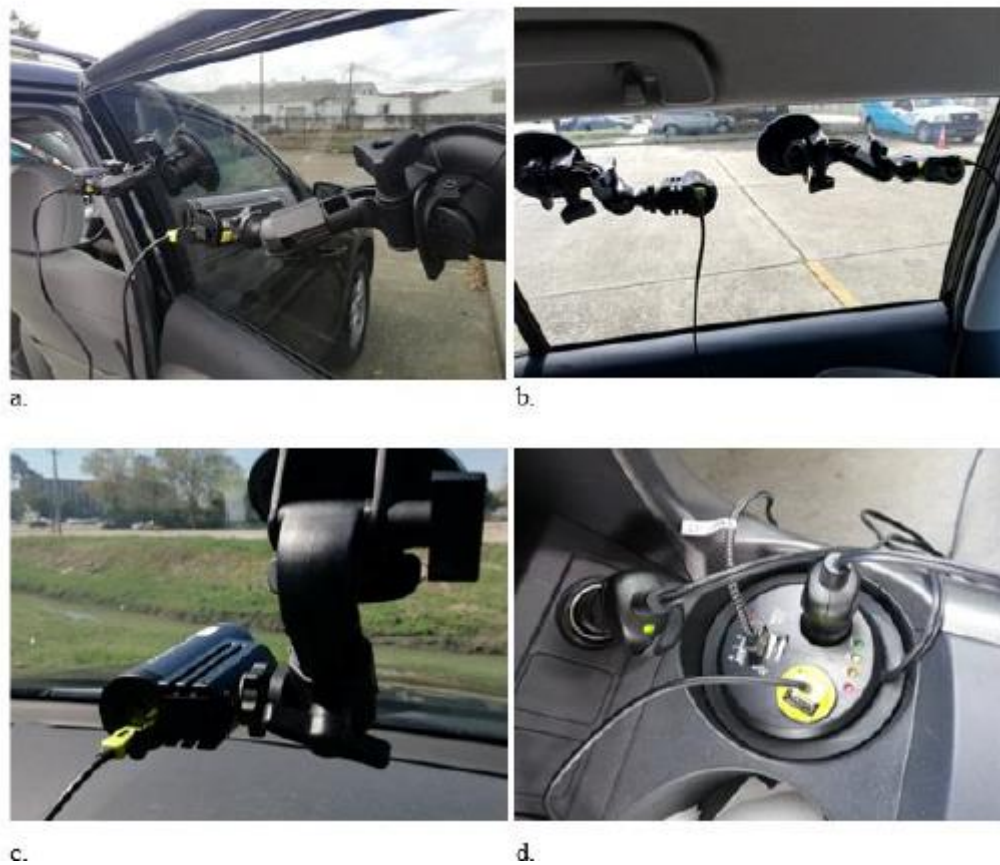


Figure 3.2: Spatial video equipment instalment illustration a) two cameras on the inside window on the right, b) two cameras placed on the inside window on the left, c) camera fixed on the inside windshield, d) charging equipment. Adopted from (Stratmann 2019; Ristea *et al.* 2021)

Table 3.2: Details of the distance, duration and dates of the spatial video data collection. Adopted from Stratmann (2019, p. 44, Table 4).

	Distance [km]	Duration [h:mm:ss]	Date
Fairfields	71.12 km	3:11:54	March 6 th 2019; March 8 th 2019; March 11 th 2019
Mid City	122.02 km	4:29:06	March 11 th 2019; March 13 th 2019; March 18 th 2019
Southside	56.4 km	2:03:00	March 25 th 2019
Pollard/Woodchase	65.9 km	2:23:00	March 27 th 2019
University Acres/Woodstone	68.4 km	2:29:00	March 22 nd 2019
Total	383.84 km	14:36:00	8 days

3.3 Data Preparation

In supervised learning, the algorithms are provided with labeled examples from which they can learn from. Therefore, for the selected deep learning model to detect physical urban blight indicators in spatial videos, the model must be provided with labeled examples, i.e., examples of physical urban blight indicators in spatial videos.

Convolutional Neural Networks are known to perform better when provided with large, labeled datasets to learn from, as compared to small amounts of data. This also applies to computer vision tasks including object detection (Pathak *et al.* 2018).

Apart from the amount of data used, the quality of the dataset also plays a big role in the accuracy of the CNN's output (Zhou *et al.* 2017). This implies that garbage in, garbage out. Hence, it is important to prepare the data before they are passed to the model. This is often known as pre-processing.

3.3.1 Data Pre-processing

The pre-processing steps to be performed for any data is determined by the kind of data under consideration. In this research, spatial videos will be used. Spatial videos are like other videos. They mainly vary from them due to their capability to capture the geographic information as

part of the videos captured. Therefore, the pre-processing steps carried out for other videos may be applicable to the spatial videos.

Video streams may often be characterized by factors such as motion blur and video defocus. These lead to varying quality across the video frames (Zhu *et al.* 2020). There may also be varying lighting conditions across the video frames. All these factors need to be taken into consideration and handled appropriately for improved results. Depending on the intensity of destruction arising from motion blur, video defocus etc., some video frames may be removed from or included in a dataset.

In this study, the intended approach to be used for training the deep learning model is by using extracted video frames. Thus, individual video frames have to be extracted using a suitable frame extraction tool. Since a large dataset is more suitable for CNNs, VLC Media Player will be used to automatically extract the video frames from selected videos. VLC may also be helpful to edit certain parameters of the video such as the video size to suit the requirements of diverse platforms. Frame Selector, an alternative frame extraction software, will be used for extracting the test images from selected videos. This is because during the mapping process, it may be vital to distinguish between the various test video frames being used. After the frame extraction process, suitable video frames which cover a variety of scenarios (e.g., different lighting conditions) will be selected. Subsequently, the frames will be labeled using bounding boxes in Labelbox. This will involve drawing bounding boxes around physical urban blight indicators and assigning the appropriate class to each object. By using the bounding boxes, the spatial extent of the object within the video frames can be determined. Thus, enabling the object to be classified and localized. The resulting annotation file will then be converted into the appropriate format for the selected model.

For this research, the dataset will be split into training, validation and test datasets. The training dataset will be used to show the model some examples of physical urban blight indicators. The validation dataset will be used to determine if it will be necessary to tune the model's parameters any further. This depends on how well the model identifies physical urban blight indicators on the validation dataset. On the other hand, the test dataset will be used to assess how well the model identifies physical urban blight in spatial video frames that it has never seen.

3.4 Choice of Object Detection Model

YOLOv3 will be used in this study. YOLOv3 is known to outperform two-stage detectors such as R-CNN in terms of speed. Although the two-stage detectors were previously known to produce better localization accuracy, subsequent versions of YOLO namely, YOLOv2 and YOLOv3, give more attention to this challenge. YOLOv3 also performs better than other one-stage detectors including variants of Single Shot MultiBox Detector (SSD) (Zou *et al.* 2019). It also offers a better trade-off between accuracy and detection speed compared to RetinaNet. In addition, it is 3.8x faster than RetinaNet (Redmon and Farhadi 2018).

Within the scope of this study, YOLOv3 is preferred due to its ability to detect multiple objects of different classes, the good trade-off between detection speed and accuracy that it offers, its ability to detect objects in real-time, and its successful application to detect objects in spatial videos (Ajayakumar *et al.* 2021).

3.4.1 Detailed Architecture of the Selected Model

Darknet-53 is the backbone network that YOLOv3 uses for its feature extraction. An input image is split into an $N \times N$ grid, where N is usually 7. For each grid cell, five bounding boxes are created, together with an “objectness” score per bounding box. A number of class probabilities are also output. If K represents the total number of classes, then the number of values produced by each grid cell is $25 + K$ ($5 \times 4 + 5 + K$). Only bounding boxes whose center lies in a specific grid cell are predicted (Ajayakumar *et al.* 2021). Further details of the YOLOv3 architecture can be found in the original paper by Redmon and Farhadi (2018) .

3.5 Training and Validation of the Object Detector

After the data preparation stage, the YOLOv3 model will be trained to identify litter. This will be carried out using the training dataset containing the labeled video frames. At this point, the goal is to fit the model on the training dataset.

There are some parameters such as number of iterations, batch size, learning rate etc., that need to be carefully selected to get a good model. These parameters are known as hyperparameters and are not determined by the model. They are rather user-specified inputs. The tuning of the model’s parameters is done at the validation stage. Using the validation dataset, the model can be assessed and appropriate parameter values for the model will be determined (Aloysius and Geetha 2017; Mohri *et al.* 2018). Different parameter settings will lead to different models. Hence, at this stage, the model whose parameter gives the best results will be selected as the model for this study.

The loss function, a function that measures how different a predicted label is from a true label (Mohri *et al.* 2018), will be used to determine the progress of the model and to avoid underfitting or overfitting the model. Underfitting occurs when a model’s bias is very high. In such a case, the model fails to adequately capture complex patterns that exist in the given data. The underfitted model’s failure to capture the complex patterns in the data, results in increased training and validation errors (Minhas 2021). The model can therefore be described as too simplified to learn the problem well. Underfitting often occurs when there is insufficient data leading to too little learning (Brownlee 2018). In the case of overfitting, the model is too complex, i.e., it learns intricate details of the training dataset including the noise in the data. As a result, although the model may perform well on the training dataset, it will not be able to generalise well to data it has never seen before. The goal is to achieve a good model that learns well from the training dataset and can generalise well to data it has never seen before (Brownlee 2018; Minhas 2021).

3.6 Testing of the Object Detector

At the testing phase, the test dataset will be used to evaluate the model’s performance. The test data will be kept separate from the training and validation datasets. This implies that it will not be made available to the model during the learning stage. After training and validating the model, the model will then be required to make predictions based on the features it has learnt from the labeled examples. These predictions will be made on video frames that it has never seen before.

The model's performance will be evaluated by comparing the predicted labels to the ground truth labels of the test dataset. It must be noted that the ground truth labels of the test dataset will not be provided to the model. Several evaluation metrics exist for this evaluation process. These will be discussed in the following sub-section.

3.7 Evaluation of the Object Detector's Performance

Evaluation metrics used to evaluate the performance of an object detection algorithm include Recall, Precision, Intersection over Union (IoU), Frame Rate, Precision-Recall curve (PR curve) etc. These metrics are very much related to one another. The IoU measures how much the predicted bounding box overlaps with the ground truth bounding box. The closer the match, the higher the IoU value. Using the IoU metric, it is possible to determine correct predictions and wrong predictions, i.e., false positives generated. Precision measures how many of positive detections made by the model are true. The recall focuses on the how many positive detections have been made out of all the ground truth data. The Average Precision (AP) is the derived statistical mean of the Precision (Xiao *et al.* 2020). The AP is often described as the area under the PR-curve. Hence using the AP, it is possible to summarize the PR-curve as one value. This value is the mean of all the derived precisions. The precision-recall curve shows the point at which the recall and precision are both high. This is usually the best point. However, some PR-curves may be too complex to easily identify the best point. In such cases, the F1 score can be used to evaluate the balance between recall and precision. Higher F1 values suggest that both the recall and precision are high while lower values suggest a higher level of imbalance between the recall and precision (Gad 2020).

3.8 Visualization of the Detected Physical Urban Blight

The final outcome of this study is to obtain a map showing the locations of the predicted physical urban blight indicators as a point pattern within a GIS environment. This should be done in such a way that further spatial analysis to determine an appropriate threshold for classifying an area as blighted can be carried out.

To map the identified physical urban blight indicators automatically, the indicators will be linked to the GPS coordinates of the video frames in which they occur. The result will be stored in a file in a format that can be used to display the GPS locations in a GIS environment such as ArcMap (e.g., CSV). After the locations are displayed, a threshold for physical urban blight will be determined using ArcMap.

4 Implementation

This chapter provides details about the study area, i.e., its geography and demography. Additional information about the selection of test areas within the entire study area is provided. This is followed by a description of the selection process of a physical urban blight indicator to be used within this study. The selected physical blight indicator is defined within the scope of this project. Moreover, the data preparation, model training and validation, testing, model performance evaluation processes are described into detail. The chapter ends with a detailed explanation about how the predicted litter objects are visualized on a map as point patterns.

4.1 Study Area

The area in focus for this research is Baton Rouge, highlighted in red in Figure 4.1. The City of Baton Rouge is the capital of Louisiana in the United States. Baton Rouge is listed among the four cities located in the East Baton Rouge Parish (EBRP). The other cities in the EBRP are Baker, Central and Zachary (Stratmann 2019; Ristea *et al.* 2021).

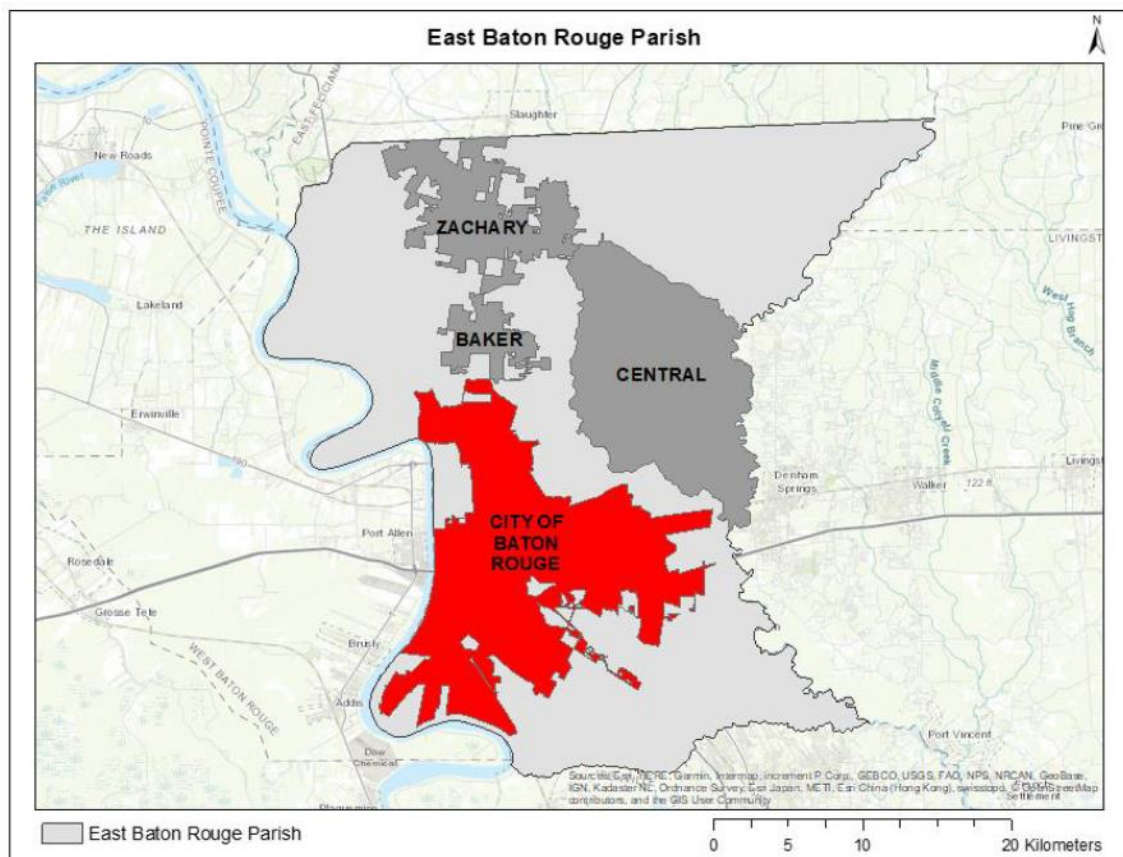


Figure 4.1: The four cities located in the East Baton Rouge Parish and their geographic boundaries. Adopted from Stratmann (2019, p.31, Figure 9)

There are 58 neighbourhoods in the city of Baton Rouge, with a total area of 123.84 km². Per the 2010 census data, there are officially 229,422 inhabitants living in Baton Rouge. As per the US Census data from July 2018, the estimated number of inhabitants was 221, 599.

Based on the Uniform Crime Reporting (UCR) Program conducted by the FBI, Baton Rouge is categorized as one of the cities in the US where violence is highest (Stratmann 2019). The UCR program is dedicated to reporting crime data of the US and ranks the reported crime data for various cities or states in the US. From the kernel density map in Figure 4.2, crime incidents in Baton Rouge appear to be the highest (red colour) in the northern and middle areas of the city, and lowest (blue colour) in the southern part. Different crime types including burglary, narcotics, assault, robbery, battery, theft, homicide, firearm, vice, criminal damage to property, juvenile and sexual assault are considered.

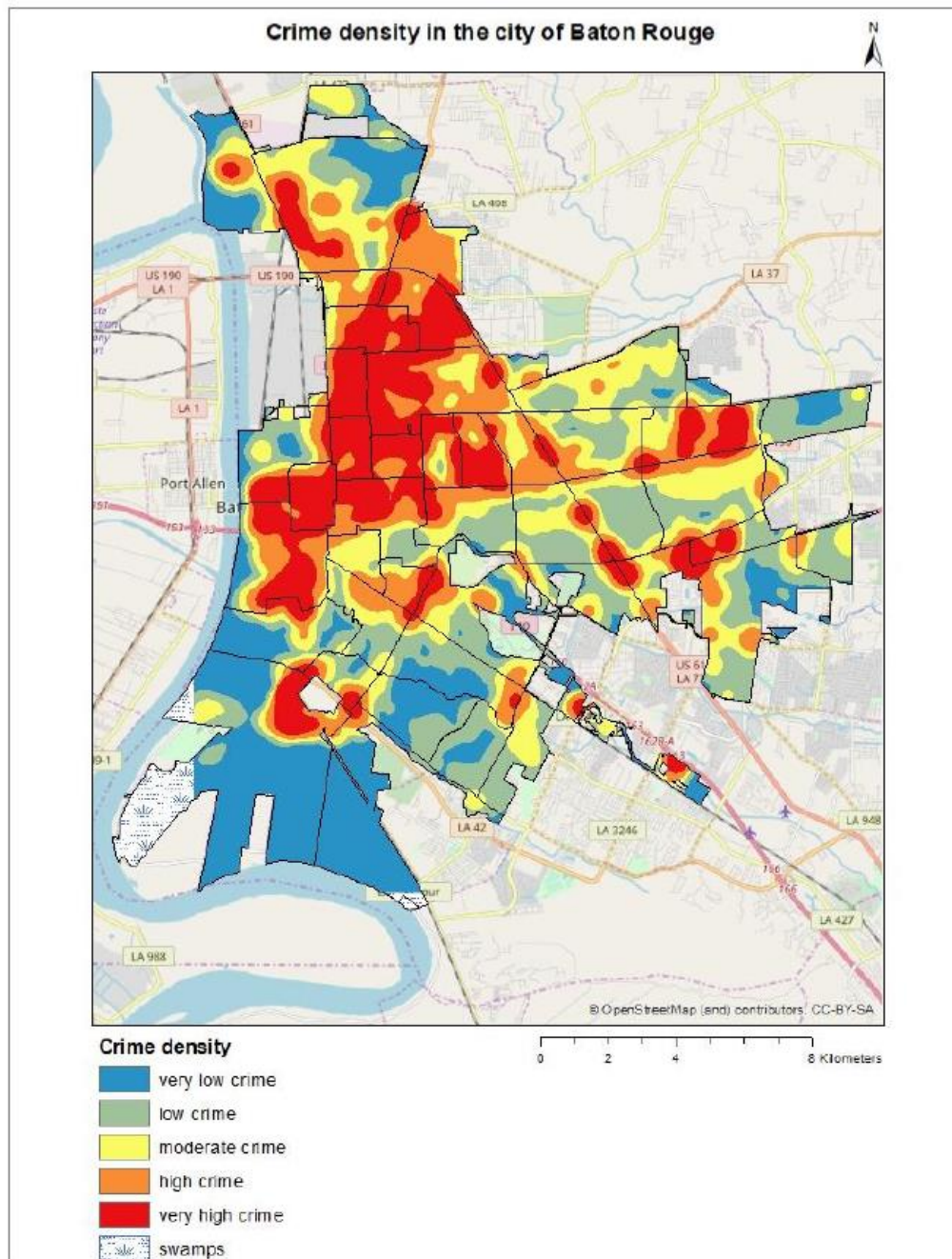


Figure 4.2: Kernel density map showing crime densities throughout the city of Baton Rouge. Adopted from Stratmann (2019, p. 34, Figure 11).

Although physical urban blight indicators as described in the criteria catalogue are relevant for studying physical urban blight, the presence of an indicator (e.g., broken windows) on one street does not imply that the whole area has physical urban blight. Also, crime locations are not distributed uniformly. Therefore, to determine the variation of physical urban blight and crime in one neighbourhood, census block groups were selected.

Neighbourhoods in Baton Rouge were selected based on the crime density, i.e., the total number of crime incidents divided by the area of the selected neighbourhoods in km². In total, five neighbourhoods that fall within Baton Rouge's city limits were selected. Using quantile classification, the crime density was classified into five classes ranging from very low crime to very high crime. Other factors that were considered for the neighbourhood selection include the following: absence of highways and lakes, connectivity between different parts of the selected areas, similar street length and ease of movement through the selected areas by automobile. In addition, to ensure that blighted areas were present in the selected sites within the study area, a density map showing urban data from 311 reports were analysed.

The selected neighbourhoods include Fairfields, MidCity, Southside, Pollard/Woodchase and University Acres/Woodstone. From Figure 4.3, it can be observed that Fairfields has a very high crime density. It is followed by MidCity with a high crime density. Southside is seen to have a moderate crime density. Pollard/Woodchase and University Acres/Woodstone have a low crime density and very low crime density, respectively. The selected neighbourhoods comprise a total of 22 census block groups (Stratmann 2019; Stratmann *et al.* 2020).

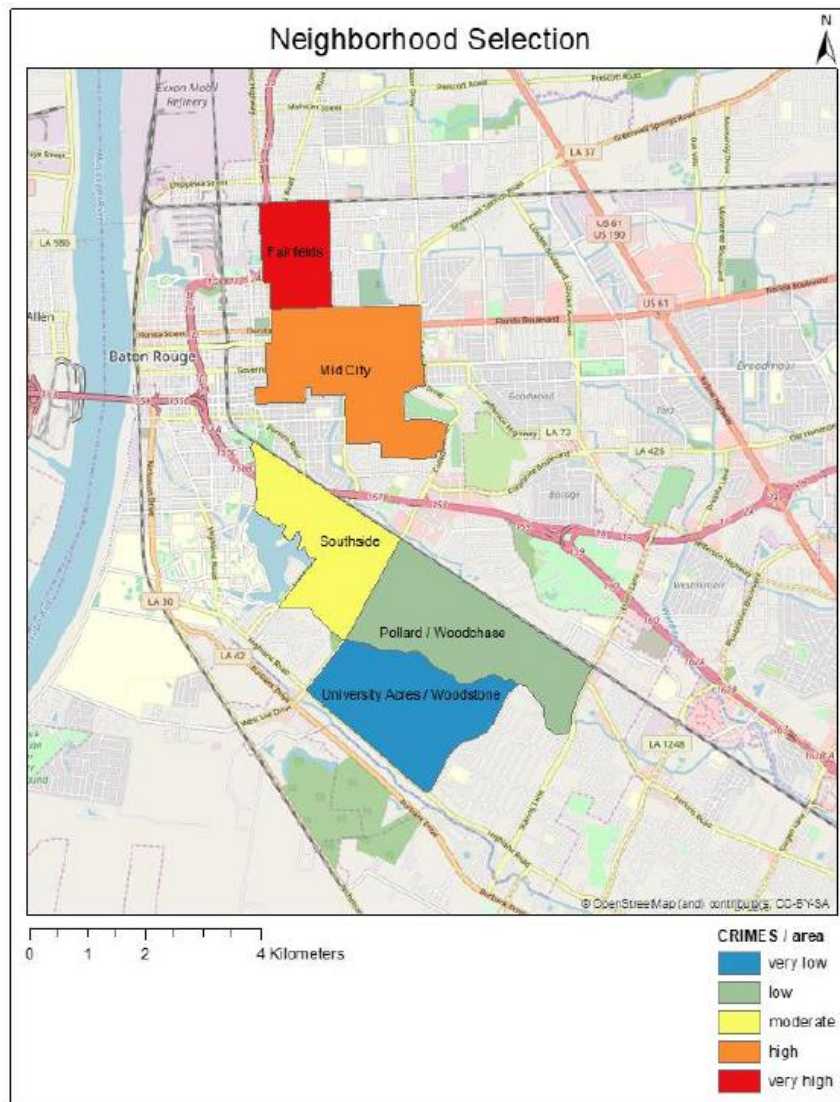


Figure 4.3: Neighbourhood selection based on the crime density in the city of Baton Rouge. Adopted from Stratmann (2019, p. 36, Figure 12).

4.1.1 Site Selection

The major criteria for selecting a few neighbourhoods and certain areas within the five neighbourhoods was the spatial distribution of physical urban blight locations. Neighbourhoods with a high concentration of physical urban blight locations were selected. This was done to obtain a dataset with a large sample size and as many varying conditions as possible to enhance the training of the object detection model.

The spatial distribution of physical urban blight locations within the five initially selected neighbourhoods, is illustrated in Figure 4.4. The red points represent locations where property blight was identified while the green points represent environmental/infrastructural blight. The majority of the points are concentrated in the northern part of Baton Rouge, where Fairfield and MidCity are located. Therefore, Fairfield and MidCity were selected as the main sites for the object detection process.

Furthermore, sub-areas within Fairfields and MidCity were chosen depending on whether there was a high density of both property and environmental/infrastructural blight. Figure 4.5 visualizes the density of property blight and that of environmental/infrastructural blight separately. The density map was obtained by dividing the number of blight occurrences associated with the specific blight type (property or environmental) by the area. Thus, comparable maps were obtained since the blight indicators were not normalized with varying variables (number of buildings, street length) (Stratmann 2019). It can be observed that the area which records the highest blight densities in Figure 4.5a form a part of the areas that record the highest densities in Figure 4.5b. Consequently, those census tracts, as shown in Figure 4.6 were selected as the sub-areas to be focused on throughout the project. The sub-areas selected in Fairfields are indicated in a light purple colour while light blue is used to represent MidCity.

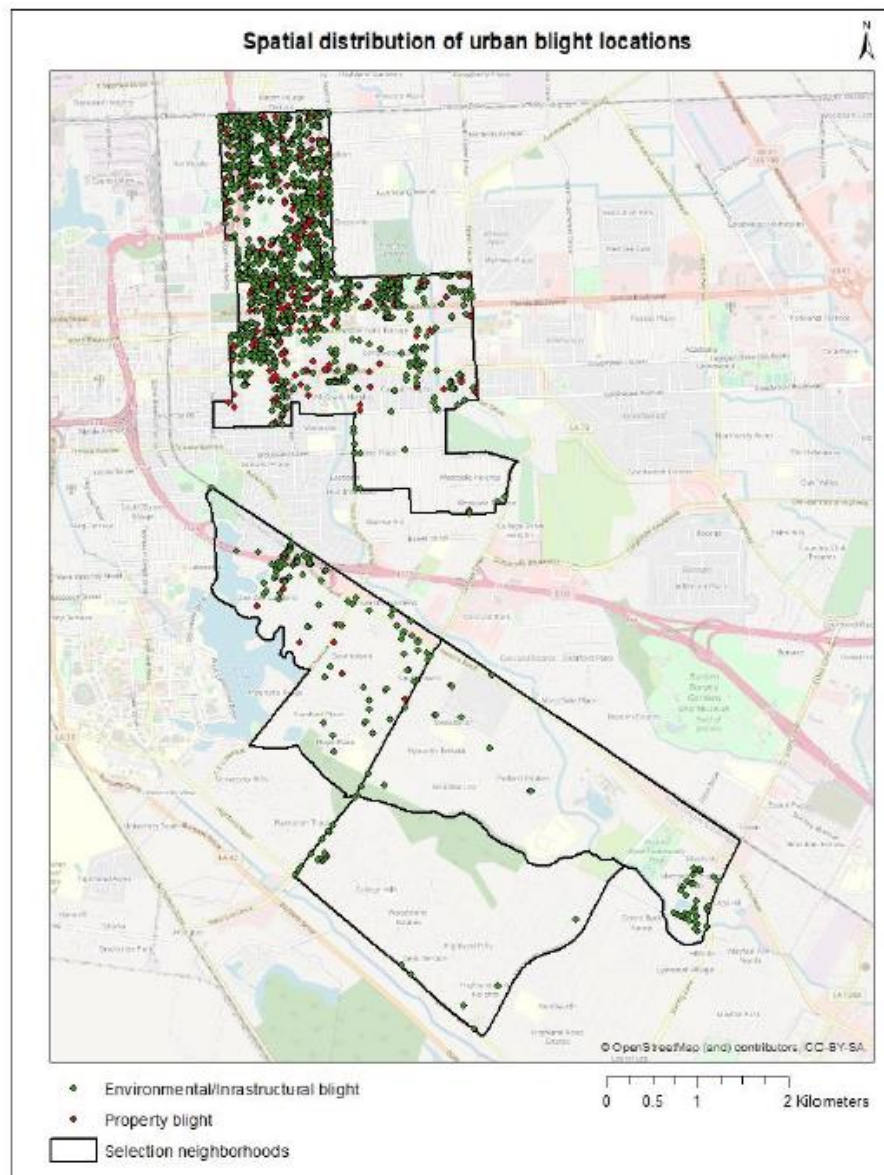
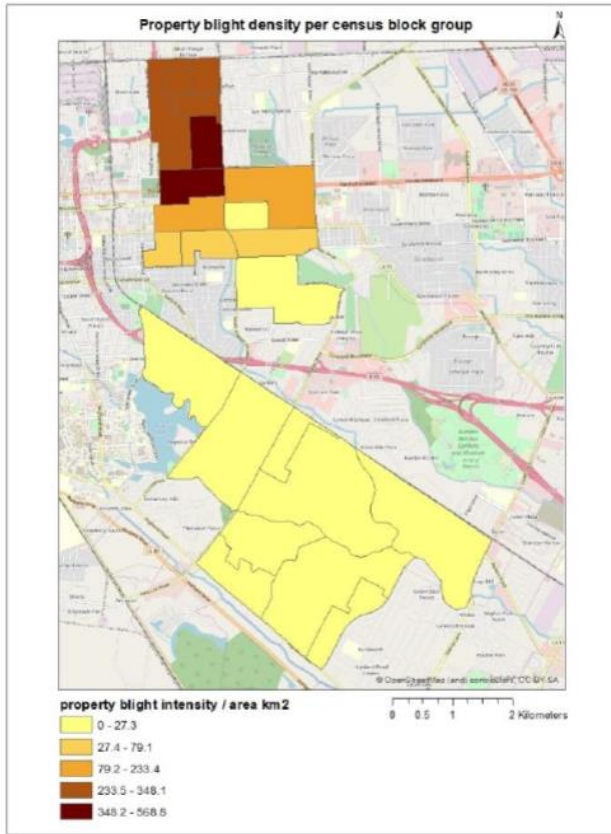
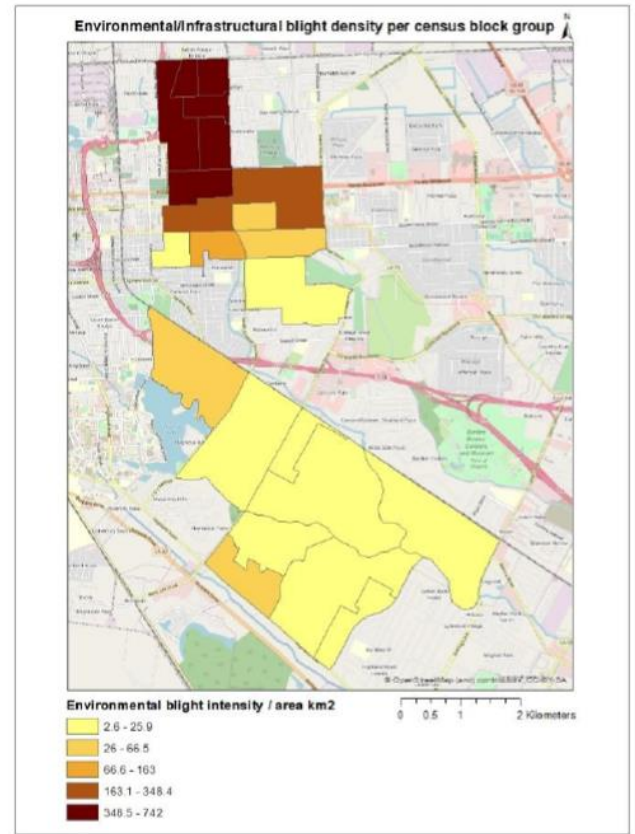


Figure 4.4: The spatial distribution of locations where property blight and environmental/infrastructural blight are present within the five selected neighbourhoods in Baton Rouge. Adopted from Stratmann (2019, p.52, Figure 23).



a.



b.

Figure 4.5: a. Property blight density; b. Environmental/Infrastructural blight density. Adopted from Stratmann (2019, p. 62, Figure 33).

It is worth noting that later in this study, an additional site – a sub-area in Southside - was randomly selected and added for the purposes of testing the object detector on an entirely new video stream covering a completely new neighbourhood.

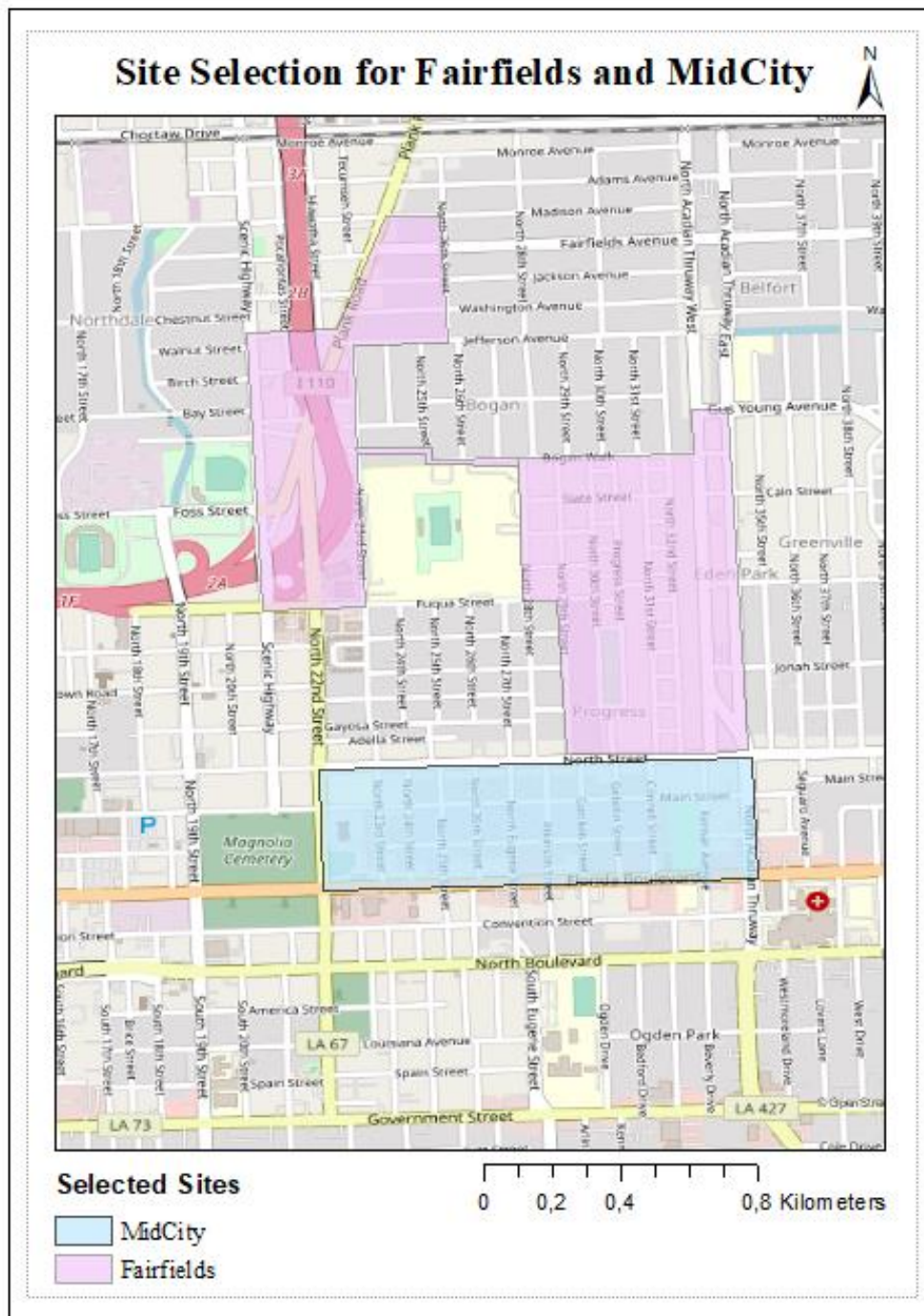


Figure 4.6: The selected sub-areas in Fairfields and MidCity to be used in this study

4.2 Physical Urban Blight Indicator Selection

The defined criteria catalogue illustrated in Table 3.1 contains 16 different types of physical urban blight indicators. Owing to the limited timeframe for this study, a single type of physical urban blight indicator is selected as the main indicator throughout the study. The selection was done based on the following factors:

- The prevalence of the physical urban blight indicator as compared to the other blight indicators;

- The ease of identification of the physical urban blight indicator by a human being.

In terms of the prevalence of physical urban blight indicators, Stratmann (2019) identified a total of 1,717 physical urban blight locations within the five neighbourhoods. 1,182 points (68.4% of the study area) were identified as being environmental/infrastructural blight while property blight accounted for 538 points (31.6%) These values did not take into consideration the occurrence of multiple blight indicators at a single location. For example, one location can be characterised by a boarded door, litter and broken windows. When the aforementioned factor was taken into consideration, 880 property blight indicators were identified in the study area while 1,498 points were attributed to environmental/infrastructural blight. The frequency of the different property blight indicators and the environmental/infrastructural blight indicators are illustrated in Figure 4.7 and Figure 4.8, respectively. Litter, which had 798 points, was identified to be the most frequently occurring blight indicator.

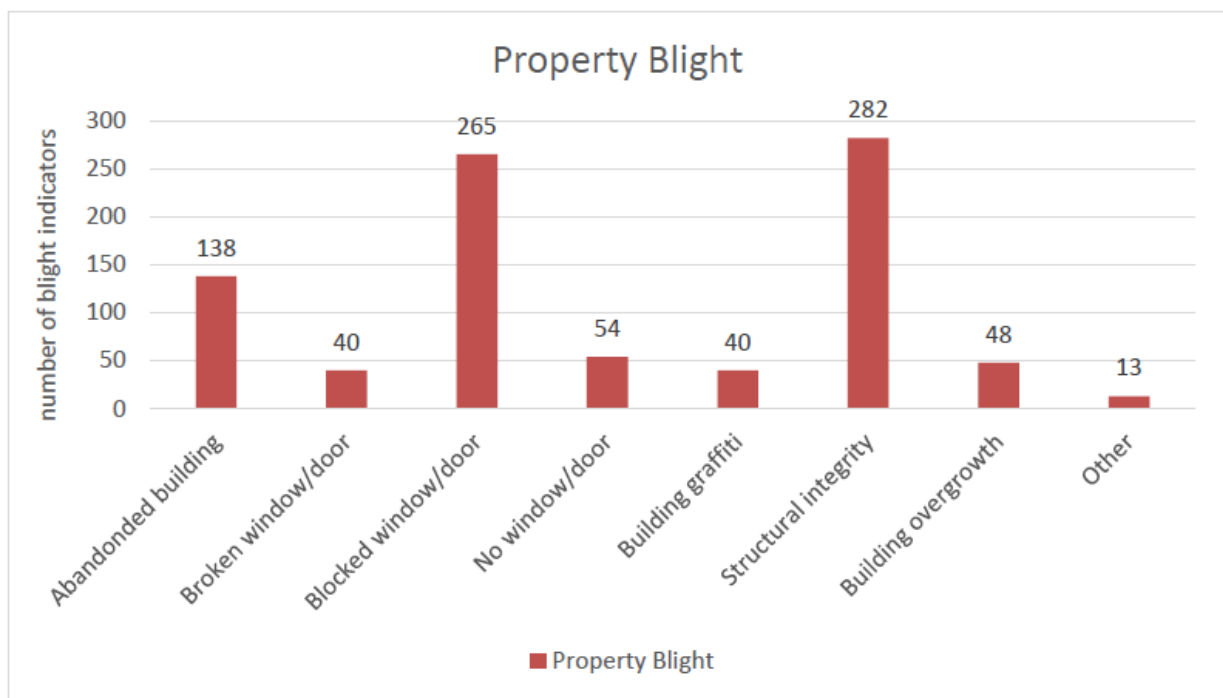


Figure 4.7: The frequency of individual property blight indicators. Adopted from Stratmann (2019, p. 50, Figure 19).

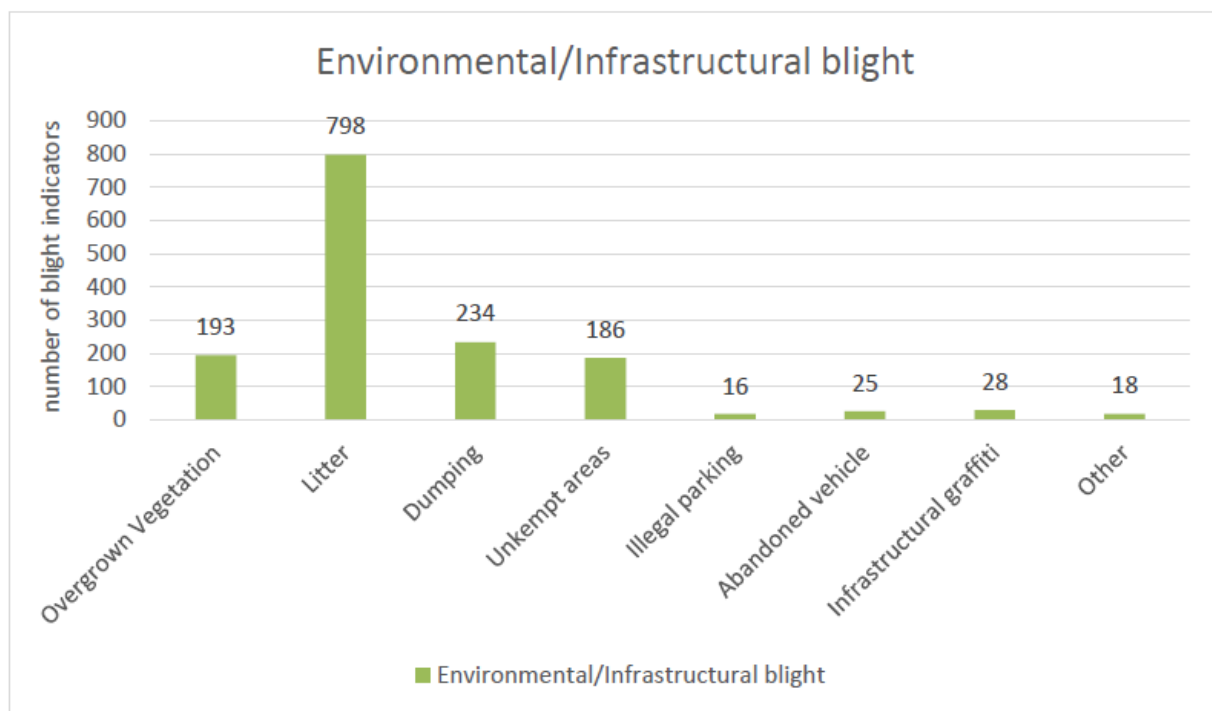


Figure 4.8: The frequency of different environmental/infrastructural blight indicators. Adopted from Stratmann (2019, p. 50, Figure 20).

The other factor which was considered to determine the physical urban blight indicator to be selected was how easy it is for humans to detect the blight indicator from the spatial videos. In the attempt to further explore the mapping of physical urban blight indicators using the traditional method, the digitizing process described in (Stratmann 2019) was replicated. Therefore, videos for the corresponding test sites (Fairfields, Southside, University Acres/Woodstone) were randomly selected. The videos were watched and each time a blight indicator was spotted, a point was placed at the corresponding blight location on a map layer in ArcMap 10.6.1.

During the process, it was observed that the indicators which occurred at the foreground of the video frames were the most easily identifiable. These were mostly environmental/infrastructural blight indicators, of which litter stood out. Litter was closely followed by unkempt areas, overgrown vegetation and illegal dumping. The latter three could almost be observed at the same time and were therefore assigned to a similar rank. It was also observed that property blight became more obvious in the absence of any environmental/infrastructural, except in the case of bright-coloured buildings. While viewing the video, it was also noted that some blight indicators such as broken windows were the most difficult to spot. They were however more noticeable if they were of a higher intensity, i.e., if they occurred in higher numbers.

Based on the experience with this manual detection and mapping of physical urban blight, a ranked list was developed as follows: 1) Litter; 2) Unkempt area/overgrown vegetation/illegal dumping; 3) Structural integrity/boarded windows; 4) Building graffiti/building overgrowth; 5) Broken windows/doors, no windows/door.

To add to this, the opinion of Judith Stratmann, an expert in detecting physical urban blight from spatial videos, was sought via email correspondence. She mentioned abandoned property, building graffiti and illegal dumping, litter and blocked windows as easy to detect. On the other hand, she stated broken windows as being one of the hardest indicators to detect.

Litter was therefore selected as the main physical urban blight indicator for this research. This is because it was identified as the most prevalent or frequently occurring physical urban blight indicator and was also easily identifiable in the spatial videos.

4.3 Litter Definition and Scope

It is commonplace for the terms ‘litter’ and ‘illegal dumping’ to be confused or have different meaning to different people. Hence, it is important for these two physical urban blight indicators to be distinguished from each other in a way that brings clarity to the work being carried out in this study.

Several definitions exist for litter and illegal dumping. For instance, litter may be used to describe pieces of trash that are discarded onto the ground, rather than being placed in a garbage bag (Weisburd *et al.* 2010). A waste commission known as iLiveHere was established as an environmental outreach program for Scott County in Minnesota, United States. They describe litter as little amounts of trash pieces that are thrown carelessly onto the ground. They further add that although the act of littering may involve dropping small pieces of trash at a time, litter tends to pile up very quickly over time. In differentiating between litter and illegal dumping, the latter was described as a large amount of garbage that is dumped rather than being disposed of properly (iLiveHere 2021). Maghelal *et al.* (2013) describes illegal dumping as trash piles, i.e., waste material (e.g., construction materials, tyres, asbestos) that is unlawfully deposited.

From the definitions provided, some key factors for differentiating between litter and illegal dumping are identified. These include the volume/size of the trash item, the type of waste, how the trash pieces are packaged (in a disposable garbage bag or not) and the appearance/dispersion of the trash items. Trash items are classified as litter, if they have a small volume or size and are disposed of onto the ground in a careless manner. Such objects are usually not placed in a garbage disposable bag. They are also relatively lighter and may include food packages, drink cans and bottles. Illegal dumping, on the other hand, denotes heavier and bulkier objects such as tyres, mattresses, construction materials and discarded furniture. It may also include disposable garbage bags containing trash pieces that are dumped at undesignated places. With regards to appearance, litter often appears scattered and randomly dispersed while items which are illegally dumped often mostly appear in heaps. Nonetheless, continuous littering may generate piles of litter if the act is not controlled. Table 4.1 summarizes the differences between litter and illegal dumping based on the identified factors: volume/size, type of waste, packaging of the waste and appearance/dispersion of the trash items.

Table 4.1 Differences between litter and illegal dumping

Differentiating Factor	Litter	Illegal Dumping
Volume/ Size	Lower/Smaller	Higher/Larger
Type of Waste	Relatively lighter objects, such as food packages, drink cans, bottles, etc.	Heavier and bulkier objects, such as tyres, mattresses, discarded furniture and construction materials, disposable garbage bags containing waste
Appearance/Dispersion	Mostly scattered or randomly dispersed Exception: Piles of litter may be generated due to continuous littering over time	Heaped/piled
Packaging	Trash pieces are dropped onto the ground as they are, rather than into a garbage bag/ bin	Trash may be deposited into a disposable garbage bag

Within the scope of this project, litter is defined as artificial objects which are discarded or disposed of carelessly onto the ground in private areas (e.g., in a person's garden) or in public areas (e.g., on the street, on vacant lots), rather than being placed within designated garbage bins. These artificial objects may include drinking cans, food wrappers, bottles etc., and are usually lighter and smaller than illegally dumped objects such as tyres. Due to the random nature in which littering is done, litter may often appear randomly dispersed or scattered. However, it may also appear in piles in cases where it has not been controlled.

It should be noted that naturally occurring objects such as tree branches and leaves are not considered as litter within the scope of this project.

4.4 Data Preparation

As previously discussed, some sub-areas in Fairfield and Mid-City were selected to be the focus areas in this study. This was because the selected areas had a high density of property blight and environmental/infrastructural blight.

To identify the corresponding spatial videos for the selected areas, they were viewed in the Camera Player software. In Camera Player, the uploaded spatial videos can be viewed alongside their GPS tracks, as illustrated in Figure 4.9. By viewing the GPS tracks of the collected spatial videos, the appropriate ones were identified. In total 3 videos were selected out of all the spatial videos. These represented the selected areas in Fairfield and MidCity. A random video of an area in Southside was added to the dataset. This was due to the need to test the model on an entirely new spatial video from a completely different neighbourhood. In addition, an additional video from Fairfield was also included in the dataset because of the high presence of large

piles of litter and large individual litter objects. This addition was necessary in order to obtain a well-balanced dataset that covers a wide span of scenarios in the spatial videos. Therefore, the total number of videos used in this study is 5. The videos have a frame rate of 25 frame per second and a resolution of 1920×1080 . This high resolution ensured that the spatial videos were of good quality. Apart from the video from Southside, video frames from the other selected videos were used during training, validation and testing. Extracted frames from the selected Southside video were only used during the testing stage. The selected videos, their lengths and the purpose for which they were used in this study are described in Table 4.2.

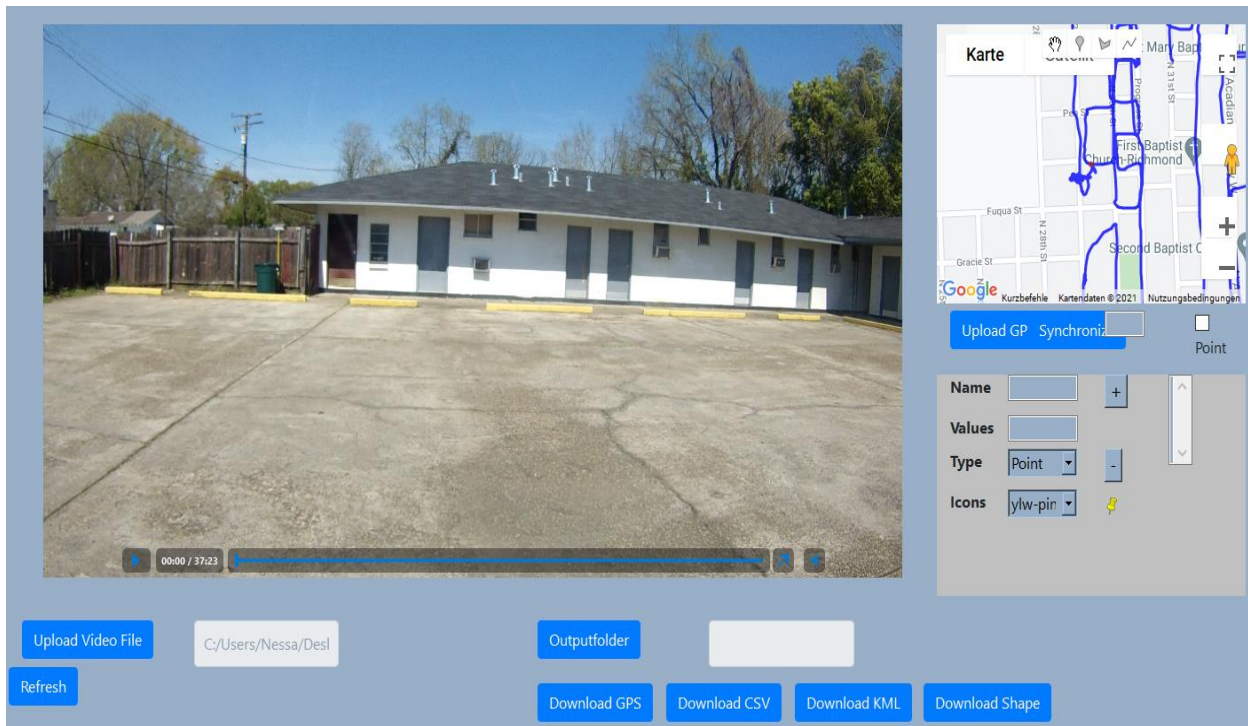


Figure 4.9: An uploaded spatial video (middle) and its GPS track (upper right) shown in the Camera Player software

Table 4.2: The neighbourhood in Baton Rouge, US, video length and size, and the purpose of the selected videos

Video	Neighbourhood	Video length	Size	Purpose in the study
Fairfields_2.1_left	Fairfields	00:37:23	3.66 GB	Training and Validation only
Fairfields_2.2_left		00:7:37	777 MB	Training, Validation and Testing
Fairfields_3&4_left		00:31:44	3.08 GB	Training and Validation only
MidCity_1_left	MidCity	00:24:41	2.40 GB	Training, Validation and Testing
Southside_1_left	Southside	00:30:40	3,04 GB	Testing only

It must be noted that the entire data collected covered angles from the left, right and front since multiple cameras were used at different angles. However, for a better view of the objects in a video frame, the left and right recordings were the most suitable. Those from the left were selected for use in this study, while those from the right were watched occasionally for confirmation (e.g., in case of uncertainty of the class of an object).

To further ensure high quality of the selected dataset and transform the spatial videos into the appropriate format for the model, some pre-processing steps were performed. These include extracting and selecting suitable video frames, splitting the dataset to be used in various stages of the model implementation and conversion of the annotation file to YOLO format. These steps are elaborated on in the ensuing sub-sections.

4.4.1 Spatial Video Frame Extraction and Selection

For video object detection, videos can be labelled in two ways. The first method involves the extraction of individual frames from the video. Objects in the individual video frames are then labelled. This approach leads to feature extraction redundancies between frames which are adjacent to each other. This is because there are spatial and temporal relationships between video frames. In addition, there may be computational inefficiencies, if features are detected in each image/video frame (Zhu *et al.* 2020). The second approach involves directly annotating the videos. This is done in such a way that the spatial and temporal correlations between video frames are preserved. Objects which appear in consecutive video frames are only labelled once. Their labels may however be adjusted due to changes in the angle or position.

Considering the advantages of the second approach, it was desired to annotate the spatial videos as complete videos rather than as individual images/video frames. Nonetheless, some challenges were encountered during the upload of the spatial videos onto the selected annotation platform, Labelbox. Due to a file size limit of 256 MB, the videos had to be trimmed or have their resolutions lowered. Sample spatial videos were therefore trimmed while the resolutions of the remaining sample spatial videos were lowered. After several trials from different people including the Labelbox support team, all but one video could finally be uploaded successfully. Since the trimmed and lower resolution videos could both be uploaded, the trimmed was preferred. This is because a higher resolution is much suitable for obtaining more accurate results. It is not yet clear why one of the sample spatial videos could not be uploaded. Due to time limitations the first approach, outlined in the previous paragraph, was adopted. Labelbox was also maintained as the data annotation platform for the project due to the interest to ensure high data quality.

In Labelbox, the individual images/video frames must already exist prior to the upload. There is no option to upload the video and choose between the first or second approach. Hence, it was necessary to include a frame extraction process.

VLC media player and Frame Selector were used to extract the video frames from the selected spatial videos. VLC media player was used to automatically extract a large batch of frames from some of the selected videos. The extraction process was achieved using ‘Scene Filter’, one of the many filters available for videos within VLC. Figure 4.10 shows the parameter settings used in extracting the frames from the spatial video. The image format was set to jpg while the image width, height and filename prefix were kept at the default settings. Leaving the

image width and height at -1 ensured that the output images were not resized (Julie 2014). The resulting output images were stored in a user-specified location. The recording ratio was set to 1 in order to capture all the video frames. The recording ratio is a parameter that informs VLC how often an image should be captured. For example, a recording ratio of 50 implies that every 50th frame should be captured. Apart from the specific settings, it was necessary to ensure that 'Scene Video Filter' was checked, under the list of video filters. Subsequently, the spatial videos were uploaded to VLC one after the other. As long as 'Scene Video Filter' was checked, the scene filter settings were applied automatically to all uploaded videos while they were being played. The output images were then saved to the specified folder. After the extraction of the selected spatial videos, 'Scene Video Filter' was unchecked to prevent further frame extraction on other videos.

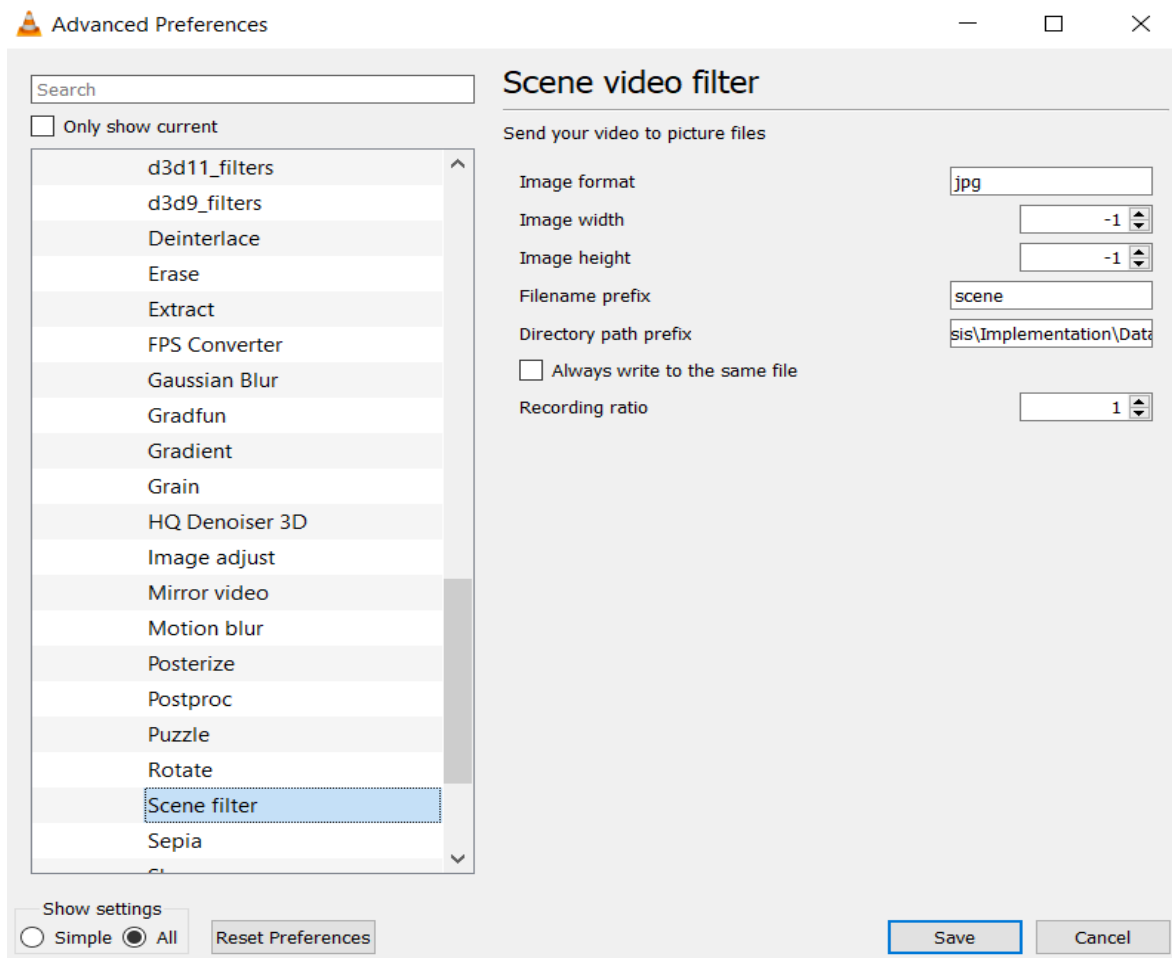


Figure 4.10: Parameter settings for extracting frames from videos using 'Scene Filter' in VLC media player

After the frames were extracted, some of them were selected to form the training and validation datasets. The frames were selected based on the following criteria: lighting conditions, the type of property on which the litter object occurs, the type of surface on which it occurs, the type/shape, size and colour of the litter object as well as the concentration nature of the litter object. The selected spatial videos were collected under different lighting conditions. In some cases, it was bright and sunny while at other times, it was dark and cloudy. A spatial video could have a mixture of both cases. Furthermore, litter could be found on either private or public

property (e.g., an individual's garden) or public property (e.g., vacant lots, on the street, etc.). On these varying types of property, they could also be found on grass, bare soil, on the road, etc. Additionally, the litter objects were of different types (and hence shapes) and varied in size; the objects ranged from small to large. They also had different colours with white being the most predominant. Moreover, there could be single instances of litter or piled up litter. Therefore, the dataset was selected to represent these variations. The selections were made in such a way that the resulting dataset would reflect as many real-life scenarios as possible. This was done to improve the accuracy of the model on different conditions.

It should be noted that some video frames were affected by reflections, motion blur and tree shadows. The video frames that were badly affected by these were not included in the dataset. However, those which were not badly affected were included because in real-life scenarios, there will be some motion blur, reflections and tree shadows. Including these variations was relevant for the model's training process and achieving good results. Furthermore, successive video frames containing the same objects were only selected if the objects were shown at different angles. Successive video frames in which objects remained unchanged were not included in the dataset in order to avoid redundancy.

Although VLC media player was suitable for automatically extracting video frames to make up the training and validation dataset, Frame Selector was used to extract the video frames for the test dataset. Frame Selector, illustrated in Figure 4.11, was selected for this task mainly because of its naming convention of the extracted video frames. The naming convention includes the spatial video name and the timestamp of the video frame, and this makes it easier to distinguish between the video frames. This is necessary for the mapping of the identified litter locations using the spatial video frame GPS coordinates. Therefore, Frame Selector was used to extract frames from the selected videos including the Southside spatial video. The resulting frames were used as the test dataset. The spatial videos were uploaded to Frame Selector one by one and played. Each time a litter object was spotted, and the aforementioned frame selection criteria were met, the video frame under consideration was added by clicking on 'Add Frame'. In Frame Selector, it is possible to save the frame extraction session for future work. Therefore, the output images were downloaded, and the session was saved. Saving the session created a JSON file containing information about the uploaded spatial videos and their respective frames. The output images/video frames were saved using the video name and the timestamp at which the frame occurs (in seconds).

4.4.2 Dataset Splitting

This research was ran two times, the first time with a small dataset of 138 video frames and the second time with a larger data set of 338 video frames. The 138 video frames of the first smaller data set were also included in the larger data set of 338 frames. 110 frames of the first smaller data set were split by a 90:10 ratio into 99 frames for training and 11 for validation. This 90:10 ratio for splitting the training and validation data is suitable for very large datasets (Kumar 2021). Since the sample size of the litter objects was high, this was maintained as the default value of 90:10 as set by Muehlemann (2021) within the selected YOLOv3 model. The remaining 28 frames were selected for testing. The larger data set of 338 frames was also split by a 90:10 ratio into 279 frames for training and 31 for validation. The same 28 frames from the first smaller data set that were used for testing were included as the test dataset for the

second larger data set. This was done to obtain comparable results and better understand the model’s performance after being trained and validated on different data sets. It should be noted that although the two datasets vary in size, another difference exists. The second larger dataset has more examples of large litter objects than the first smaller dataset. Details about the number of video frames and litter objects used during training, validation and testing are included in Table 4.3.

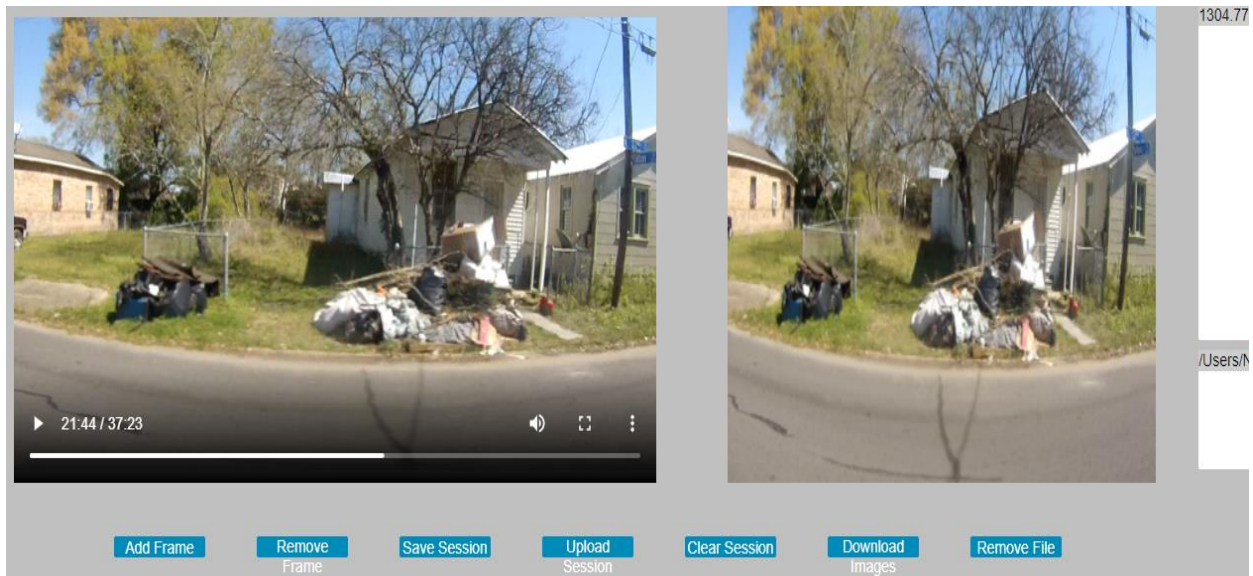


Figure 4.11: Interface of the Frame Selector software

4.4.3 Spatial Video Frames Labelling

To create labels in Labelbox, a project setup process was completed. To begin with, data were uploaded onto the Labelbox platform. Then the ‘Editor’, the area in which the labeling was to be done, was configured. This involved defining an ontology. In the ontology, the object label was specified as ‘litter’ and the bounding box was selected as the annotation/ labeling method to be used in the project. The final step for the project setup was the enablement of quality assurance tools. Either benchmarks (to measure the quality of labels using a gold standard) or a consensus (to measure the agreement level of labelers on a dataset) could be selected. In addition, a review step could be put on or off to determine whether labels should be reviewed manually or not. The settings for these quality assurance tools were left at the default: benchmarks (on), consensus (off) and review step (off).

After setting up the project, labeling could proceed. For every image/spatial video frame in the uploaded dataset, a bounding box was drawn to fit tightly around the identified litter objects. Due to the angles and shapes of some litter objects, a small section of the object’s surrounding was captured, as illustrated in Figure 4.12. On the contrary, some objects were either rectangular in shape or almost rectangular, just like the bounding boxes used. In such a case, a little space was left between the litter object and the bounding box’s boundaries, as shown in Figure 4.13. This was deemed necessary to provide the model with some context of the object’s immediate surrounding. Context was considered vital to achieve good results. For example, without context, rectangular, white litter objects may easily be mistaken for a small section of any other white object (e.g., a white wall, white floor etc.). Labelbox uses a queueing system for easy

workflows. Once labeling was completed on an image/ video frame, it was moved out of the queue.



Figure 4.12: Sample annotations of litter objects whose shapes naturally allow a section of the object's surrounding area to be included.



Figure 4.13: Sample annotation of a litter object where context (the object's surrounding) is deliberately included.

Overall, 469 objects were labelled as litter in the first dataset which consisted of 138 spatial video frames. The count takes into consideration all the labeled objects in the training, validation and test datasets. The test dataset on its own, consisted of 28 video frames with a total of 135 litter objects. This test dataset was the same for the first and second dataset. Therefore, the first dataset constituted of 334 sample litter objects for training and validation only. The overall sample size was increased from 469 to 1,141 litter objects in the second dataset; from the 200 additional video frames, 672 litter objects were identified. Out of the 1,141 litter objects in the second dataset, 1,006 litter objects were reserved for the training and validation stages. Detailed information about the number of video frames and the number of litter objects used in the different stages of the model implementation for both datasets are described in Table 4.3. The specific videos as well as the neighbourhood names are also shown.

Table 4.3: Details of the datasets used in this study; the number of video frames and litter objects used at various stages in the study, the videos the frames were extracted from, and the neighbourhoods covered.

Dataset name	Machine Learning Stage	Number of video frames selected	Number of litter objects in the selected video frames	Selected videos	Neighbourhood information
Dataset 1 (Smaller dataset)	Training	99	334	Fairfields_2.1_left	Fairfields and MidCity
	Validation	11		Fairfields_2.1_left Fairfields_2.1_left MidCity_1_left	
	Testing	28 (Test set 1 = Test set 2)	135	Fairfields_2.2_left MidCity_1_left Southside_1_left	Fairfields, MidCity and Southside
	Total number	138			
			469		
Dataset 2 (Larger dataset)	Training	279	1,006	Fairfields_2.1_left	Fairfields and MidCity
	Validation	31		Fairfields_2.1_left Fairfields_2.1_left MidCity_1_left	
	Testing	28 (Test set 1 = Test set 2)	135	Fairfields_2.2_left MidCity_1_left Southside_1_left	Fairfields, MidCity and Southside
	Total number	338			
			1,141		

According to Muehleemann (2019), to obtain decent results, at least 100 images should be annotated. At least 300 images should be labeled if good results are desired and 1000 images for great results. The sample images used in his GitHub project mostly contain one object. Since videos can be described as moving images (Oxford Learner's Dictionaries 2021), the extracted

video frames used in this study can be likened to images. In this study, some video frames contain more than one litter object. Based on this premise, the labels created for the two datasets in this study, were deemed sufficient to achieve accurate results.

It should be noted that labels were created for all objects that could clearly be identified as litter. In cases of uncertainty, the object was not labeled. This was to avoid passing wrong examples to the model. Furthermore, all objects in each video frame were labeled. In an initial labeling attempt, only objects within 2 metres of the road (in the foreground of the video frame) were labelled. This initial attempt led to the model identifying only litter objects in the foreground of the video frame. The model may have associated litter objects as objects which occur at the foreground of the video frame. Moreover, it was often necessary to zoom in to label some objects due to the small size or uncertain object boundaries. This did not influence the labels created since the video frames were restored to their default sizes.

After labeling all video frames in the labeling queue, the corresponding annotation files for the two datasets were generated and exported in the JSON format. The JSON format was preferred to the CSV format because its structure makes it easy to read and understand the information contained in the annotation file.

4.4.4 Conversion of Annotations to YOLO Format

As discussed previously, YOLO is a deep neural network that can detect objects in images and videos in a single step. Due to the good trade-off that it offers between high-speed detections and accurate predictions, it is widely adopted in object detection tasks. In the next step of this research, YOLOv3 -the selected object detection algorithm- will be trained to detect litter objects in spatial videos. However, prior to this, it is necessary to ensure that the annotation files obtained from Labelbox are in a suitable format for the YOLOv3 model. This is because YOLO has its own format for annotation files. The format requires that the image name precedes the bounding box information, and the class label as follows:

<image><xmin><ymin><xmax><ymax><label>.

The bounding box information consists of the coordinates of the top (ymin) left (xmin) corner of the box and the bottom (ymax) right (xmax) corner. The origin of the bounding box is taken to be the top left corner since the bounding box is usually drawn starting from the top left corner.

The annotation file exported from Labelbox had a different format and contained a lot more information than is required in the YOLO format. The format used for the bounding box information is as follows:

```
{... "bbox": {"top": top_value, "left": left_value, "height": "height_value", "width": width_value}}.
```

Therefore, it was necessary to convert the Labelbox JSON annotation file into the YOLO format. The conversion was done using a python script. The python script worked by getting all the image names and appending the corresponding bounding box information and label to the image name. The bounding box information was converted into the YOLO format for the bounding boxes using the following logic: The 'left' and 'top' values represented the 'xmin' and 'ymin' values, respectively. As previously discussed, the latter represent the top left corner of the bounding box. To derive the 'xmax' and 'ymax' values, the 'width' and 'height' for each

object was added to their corresponding ‘xmax’ and ‘ymax’ values. Furthermore, the newly derived information, in addition to the image names and corresponding labels were saved to a CSV file to complete the conversion to the YOLO format. The entire code is included in Appendix A.

4.5 YOLOv3 Model Training and Validation

Training CNNs from scratch with random weight initializations is rarely done. This is because deep CNNs have a non-convex loss function. Hence, training them is a rather tedious process. To obtain a meaningful weight initialization within a relatively shorter amount of time and hence reduce the training time, the technique of transfer learning was employed (Aloysius and Geetha 2017). A pre-trained YOLOv3 model developed by (Muehleemann 2021) was adopted in this study. It was selected because it was pre-trained on the ImageNet 1,000 dataset. This dataset has a wide range of categories and objects including bottles, which form a part of litter per the scope of this project. The model was developed using Tensorflow 2.3 and Keras 2.4 The entire repository is accessible via <https://github.com/AntonMu/TrainYourOwnYOLO>. The repository contains detailed information for implementing the model successfully.

The repository was cloned to a local machine and customized to suit this study’s use case of detecting litter. To customize the cloned repository, its pre-existing images and annotation files were replaced with the ones pertaining to this project. These changes were made in the directory within which it was stored on the computer. The newly customized repository on the local machine was pushed to GitHub, a platform that supports code storage, management and version control. In the ‘TrainYourOwnYOLO’ project, a Colab notebook which implements the model already exists. Therefore, it was adapted for this project. The new, customized repository containing the sample video frames for this project was then cloned in Google Colaboratory.

The YOLOv3 model was trained in Google Colab. Colab is a free, cloud-based Jupyter notebook, which comes with computing resources such as CPUs, GPUs and TPUs. The GPUs available in Colab usually include T4s, Nvidia K80s, P4s and P100s (Google 2021). Owing to the fact that deep learning tasks often deal with large datasets and therefore require higher computational power, Colab was deemed suitable for the implementation of the YOLOv3 pre-trained model. Compared to training the model on the local machine, the training process on Colab is significantly sped up when the GPU is enabled.

The training was performed for the two variations of the dataset:

- **Relatively small dataset:** This dataset contains more small-sized litter objects compared to the larger dataset. However, the difference between the number of larger litter objects and small litter objects is not very significant. The dataset can therefore be described as more balanced than the larger dataset.
- **Larger dataset:** This dataset contains more larger-sized litter objects than in the small dataset. On its own, the number of larger litter objects was significantly higher compared to the smaller litter objects.

The training was monitored in Tensorboard. The number of epochs, i.e., the number of complete iterations to be made over the training data, was set at 51. The model has 252 layers. To begin

with, the last few layers were trained on the respective datasets with a batch size of 32. The first 249 layers were frozen. To fine-tune the model on the respective litter datasets, all layers were subsequently unfrozen. The training was done per the dataset and with a batch size of 4. Too many epochs may result in overfitting. Overfitting occurs when a model is overly specialized on a training dataset and can therefore not generalize on new datasets. To avoid overfitting, a call-back function called ‘EarlyStopping’ was used in the pre-trained model. Early stopping helps to monitor a value and then stop the model from going through further epochs when no further improvements are observed. In this YOLOv3 model, the validation loss was the monitored value. However, no sign of further improvements on the first go may not always imply that there can be no more improvements. Therefore, a parameter within the early stopping call-back function known as ‘patience’ value was set at 10. Therefore, the training was only stopped if no further improvements were obtained after 10 more epochs.

It should be noted that the validation process was carried out concurrently with the training process. In addition, model checkpoints were introduced to ensure that the model was saved after each epoch. This was to avoid losing any work done and to save the best model in the end.

4.6 Testing of the Litter Detector

The customized YOLOv3 model was tested on video frames in the test dataset. The test data was not made available to the model during the training and validation stages. It should also be noted that no annotation file was provided to the model. The model made predictions and these predictions along with their bounding box information, confidence scores, etc. per video frame were saved in a detection results csv file.

4.7 Evaluation of the Litter Detector’s Performance

After testing the model, its performance was evaluated by comparing the predicted labels to the ground-truth labels. The IoU was calculated from this comparison of the ground-truth labels to the predicted labels. IoU can be described as how similar a predicted bounding box is to its corresponding ground truth bounding box. Hence, it measures how accurately objects have been detected. This IoU calculation was done by taking the ratio of the intersection between the predicted bounding boxes and the ground truth bounding boxes, and the union of both. With a range from 0 to 1, IoU values that are closer to 1 denote higher accuracies.

Other evaluation metrics were also calculated using a python script and these include Precision, Recall, and F-score. The Precision and Recall were calculated from the derived IoU values. Typically, to derive the Precision and the Recall, a threshold of value has to be defined. The threshold value varies. For example, the IoU threshold value in PASCAL VOC is 0.5. On the other hand, this value is converted to a composite value with the range [0.5, 1.0] in the MS COCO competition. All IoU values that are greater than the threshold value, are correct predictions while those less than the threshold, are wrong. From this, the statistical True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) are derived from the testing dataset. In the case of this study, the true negatives were ignored since no labels can be created to represent them. Also, a range of thresholds [0.0, 0.8] was used. The threshold was varied using values from the specified range in steps of 0.1 to observe the behaviour of the litter predictions. The precision was determined by calculating the ratio of true positives to the

combined true positives and false positives. The recall was calculated as the ratio of true positive to the combination of the true positive and the false negative. The F1 score was calculated from the derived precision and recall values using the formula: $(2 * ((precision * recall) / (precision + recall)))$. A python script was used for these calculations. The entire code is provided in Appendix B.

It was also necessary to manually inspect the results obtained after testing to verify the performance of both models. This was done by manually viewing and comparing the ground truth labels made in a video frame to the predicted labels in that same video frame. Each time a predicted box was drawn around a particular litter object and a ground truth box was identified for that same object, it was counted as a true positive. Otherwise, it was counted as a false positive, that is, a box around a litter object was predicted, but no ground truth box could be identified. When no predictions were identified for a ground truth label, it was counted as part of the false negatives. The true negatives were not relevant in this study because they neither identified ground truth labels nor predicted labels.

4.8 Visualization of the Predicted Litter Objects

To automatically map the predicted litter objects, the GPS coordinates of each video frame in the test dataset were linked to the predicted objects. The workflow is illustrated in Figure 4.14.

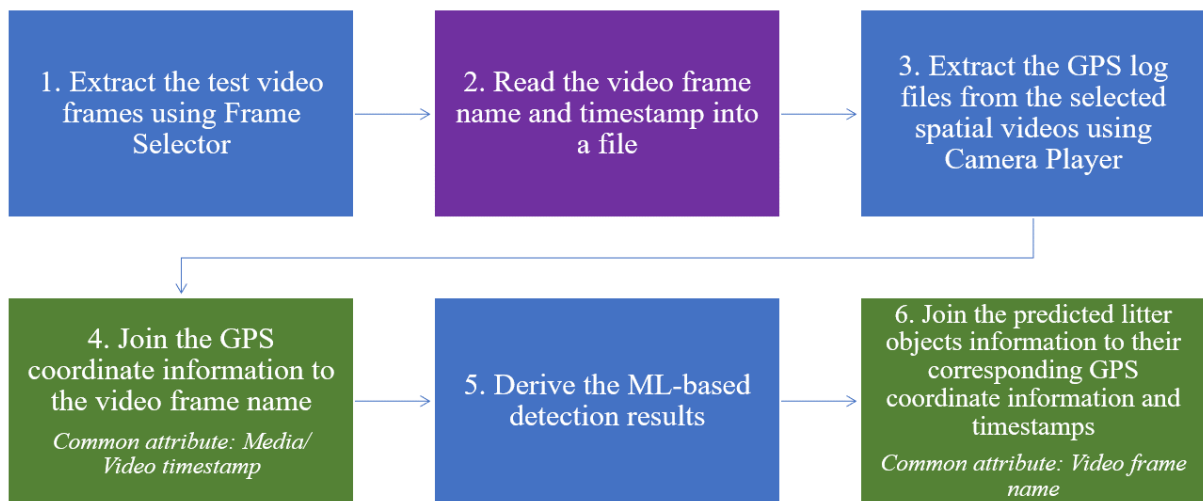


Figure 4.14: Workflow for linking GPS coordinates to their corresponding video frames

To begin with, the names of the video frames that exist in the test dataset and their corresponding timestamps were stored in a single CSV file. The naming convention used by Frame Selector to store these video frames made it possible to determine the corresponding timestamps for each video frame. The naming convention was as follows: ‘video path + ‘_’ + timestamp’. The timestamp (in seconds) represents the time at which the video frame occurs in the video. This is also known as the media time. For uniformity’s and simplicity’s sake, the naming convention was changed to a standard of ‘video name + ‘_’ + timestamp’ for all the video frames. The video path could also be very long and therefore made the video frame name very long.

Subsequently, the GPS log files for each of the spatial videos used in creating the test dataset were extracted using the Camera Player. The GPS log file contained the following information including the media time, GPS timestamp, GPS coordinates (latitude and longitude), elevation, speed, etc. However, the only fields that were of interest for this study were the media time, GPS timestamp, and GPS coordinates. Therefore, these fields were preserved. With the media time being the common attribute in both the csv file and the GPS log file, both files were joined together in Microsoft Excel. The media time in the csv file was searched for in the GPS log file and the associated GPS information was attached to the video frame name. Prior to the join, the media time in the GPS log file was converted from hours, minutes, and seconds (hh:mm:ss) to seconds. This was to make identification of a video frame's media time within the GPS log file easier. For example, a media time of 337 seconds in the csv file may not be easily identified as 00:05:37 without calculations. Therefore, the media time in the GPS log file was converted to seconds using Microsoft Excel's Time Functions (Hour, Minute, and Second). The formula used for the conversion is as follows: ((hour (cell) * 3,600) + (minute (cell) * 60) + second (cell)), where cell represents the active cell in Excel containing the media time in hh:mm:ss. The resulting file is shown in Table 4.4.

Table 4.4: Microsoft Excel file showing a section of a GPS log file extracted from Camera Player and the media time conversion from hh:mm:ss to seconds.

media time	media time(s)	time	latitude	longitude
00:00:09	9	2019-03-11T18:36:25Z	30.451811333333332	-91.165783
00:00:10	10	2019-03-11T18:36:25.500000Z	30.451813	-91.1657775
00:00:10	10	2019-03-11T18:36:26Z	30.451808	-91.16578
00:00:11	11	2019-03-11T18:36:26.500000Z	30.451805666666665	-91.165777833333334
00:00:13	13	2019-03-11T18:36:28.500000Z	30.451805	-91.165786166666666
00:00:13	13	2019-03-11T18:36:29Z	30.4518015	-91.165783333333334
00:00:16	16	2019-03-11T18:36:31.500000Z	30.451774666666665	-91.1657945
00:00:16	16	2019-03-11T18:36:32Z	30.451756	-91.165797666666666
00:00:17	17	2019-03-11T18:36:32.500000Z	30.451743	-91.165798
00:00:17	17	2019-03-11T18:36:33Z	30.451735166666666	-91.165803333333333
00:00:18	18	2019-03-11T18:36:33.500000Z	30.451738166666665	-91.1657995
00:00:18	18	2019-03-11T18:36:34Z	30.451737666666666	-91.165798166666666
00:00:19	19	2019-03-11T18:36:34.500000Z	30.451740166666667	-91.165788833333334

It is worth mentioning that a GPS file showing a frame rate of 25 fps was expected. However, the GPS files obtained using Camera Player appeared to extract the GPS coordinates at different frame rates. For instance, the frame rate was 2 fps in some videos and 5 fps in others. To crosscheck, if all GPS information had really been captured from the spatial videos, two approaches were used and compared. These include:

1. Extracting all information in a sample spatial video
2. Extracting the GPS coordinates and timestamp information from the sample spatial video.

Exiftool was used in both approaches. For the first approach, all the information contained in the sample spatial video was extracted using the following command: "exiftool -ee -G3 <file>". The records were extracted as sub-documents. These sub-documents had the frame numbers,

making it possible for the tags to be related to the time in the spatial video. This also showed that the GPS coordinates were stored on a frame-by-frame basis. However, the number of sub-documents and the number of frames in the sample spatial video did not match. The latter far exceeded the former. The former was however similar to the GPS log file that was extracted in the second approach. The command used to extract the GPS coordinates and timestamps is as follows: “exiftool -ee -p “\$sampletime, \$gpsdatetime, \$gpslatitude, \$gpslongitude” ‘<file>’ > output_file.csv”. The derived GPS log file was also compared to that which was obtained using Camera Player. The two GPS log files were similar. Their GPS recordings were made using the same frame rates and their resulting output files were the same when their location data were mapped. The Exiftool GPS log file had to be converted from degrees, minutes, seconds to decimal degrees to be comparable to Camera Player’s GPS log file.

It should also be noted that Camera Player’s GPS log file had multiple variations of GPS locations for each media time. For example, if the GPS locations were extracted at 2 fps at a media time of 00:00:03, it implied that two different GPS coordinate values and GPS timestamps were recorded successively for 00:00:03. These variations were insignificant since the GPS coordinates could be located on or almost on the same parcel. Therefore, after searching for a media time within the GPS log file, only the first GPS coordinates were used. Furthermore, since the GPS signal was occasionally lost during data collection, it is possible to have to use the last known location of a preceding video frame. This was however not necessary for the selected test datasets since no video frame had a missing GPS value.

After joining the video frame names to their corresponding GPS coordinates, it was necessary to combine this with the predicted objects in each video frame. The locations of the correctly predicted objects were to be identified based on their video frame GPS location, rather than their specific locations. The detection results file (in csv format) that was obtained during the testing phase was joined to the file containing the GPS information and the media time on the basis of the video frame name.

4.8.1 Spatial Distribution of Predicted Litter Objects

An additional field ‘number of objects per location’ was created to show how many litter objects were at a location. This was relevant for classifying a location as having a high number of litter objects. Therefore, the true positives and false positives within a video frame were manually counted and assigned to this newly created field. To avoid duplicate counts of the same object, duplicate objects in successive frames were counted only once. The resulting csv file was exported to ArcMap. Then the XY data were displayed as a map layer. The WGS 1984 coordinate system was used. A base map of Baton Rouge was also included to provide some context.

In ArcMap, a classification scheme was used to differentiate between varying levels of litter occurrences per location. This was done based on the attribute ‘number of litter objects detected.’ Natural breaks were used, and the number of classes was set at 3 to represent low intensity, medium intensity and high intensity. The corresponding ranges were 0-2, 3-6 and 7-11 respectively. These were represented using a graduated colour ramp from green to red, where green represents low intensity and red represent high intensity. Low intensity implies that there was little to no litter at a location. Medium intensity implies a moderate amount of litter at a location and high intensity represents a high number of litter objects at a location.

4.8.2 Kernel Density Estimation of Litter Intensity

To determine the litter intensity on the test dataset used in testing the models, the kernel density estimation map was created using CrimeStat IV. The parameters used are as follows: normal function, a minimum sample size of 10 with an adaptive bandwidth, cell grid size of 50 x 50 and the use of quantiles to put the density values into 5 ranges, where red represents very high blight density and blue represents very low blight density.

5 Results and Analysis

This chapter describes the results obtained from the ML-based litter detection process, the automatic mapping of detected litter objects and the classification of litter objects. The chapter concludes with a comparison of the traditional approach of detecting and mapping physical urban blight with the ML-based approach of detecting and mapping physical urban blight locations.

5.1 Litter Detector Performance Evaluation

The results achieved at various stages of the litter detection process namely: training and validation, testing and evaluation using standard metrics are described in the following sub-sections.

5.1.1 Training and Validation

The YOLOv3 litter detector was trained and validated using 51 epochs. The training and validation losses were monitored in TensorBoard. During training and validation, the goal is to reduce the epoch loss value, which is the difference between the ground truth labels and the predicted labels. Lower epoch loss values imply that the predictions made are closer to the ground truth labels.

5.1.1.1 Training on the relatively small dataset (Dataset 1)

At the onset, i.e., after the first iteration with the smaller dataset (99 training samples: 11 validation samples) using a batch size of 32 (i.e., the model was trained on 32 samples at a time, until all the samples had passed through it), the loss values for training and validation were 8956.1982 and 7416.4028 respectively. After unfreezing all layers and fine-tuning the model on the dataset using a batch size of 4, the training and validation loss values reduced significantly to 20.6424 and 22.9669, respectively. This was recorded at the 87th epoch, after which early stopping was introduced due to no further improvements on the validation model. This is illustrated in Appendix C. Figure 5.1 demonstrates how the training loss (in blue) and the validation loss (in orange) reduce over 51 epochs.

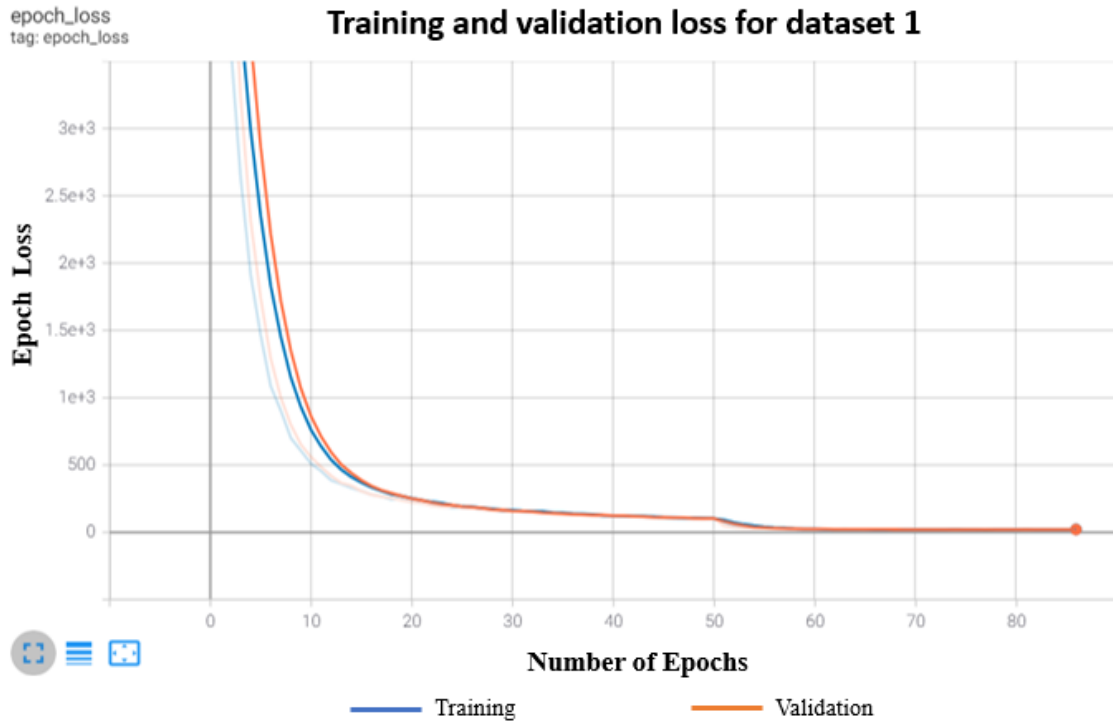


Figure 5.1: Training and validation losses on the first, smaller dataset.

5.1.1.2 Training on the large dataset (Dataset 2)

The training was done on 279 video frames and validated on 31 samples, with a batch size of 32. After the first epoch, the training and validation losses were 5657.1567 and 3264.6936, respectively. By the 75th epoch, the training and validation losses had reduced to 19.1792 and 18.5987, respectively. At the 75th epoch, early stopping was introduced since no improvement were seen on the validation loss value. This is demonstrated in Appendix C. Figure 5.2 illustrates how the training and validation losses reduce over 51 epochs. Table 5.1 summarizes the training and validation losses for dataset 1 and 2 and shows the final epoch number for training and validation.

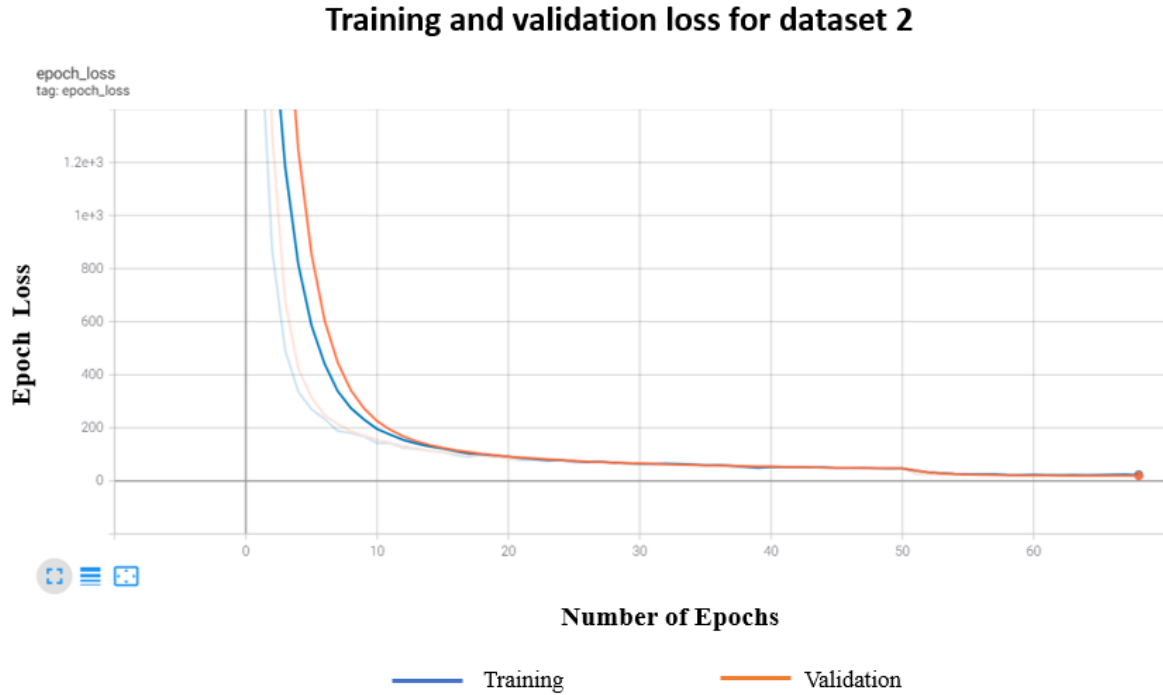


Figure 5.2: Training and validation losses on the second, larger dataset

Table 5.1: Summary of training and validation losses for dataset 1 and 2 and the final epoch number for training and validation

		Dataset 1	Dataset 2
Initial values	Training loss	8956.1982	5657.1567
	Validation loss	7416.4028	3264.6936
Final values	Training loss	20.6424	19.1792
	Validation loss	22.9669	18.5987
	Final epoch number	87	75

When comparing both data sets, the larger dataset 2 appears to have lower loss values than dataset 1 at all stages. This implies that the training and validation on dataset 2 yielded predictions that were closer to the ground truth labels than dataset 1. Furthermore, better results were obtained from the training and validation on dataset 2 over a fewer number of epochs compared to the results of dataset 1. Considering that dataset 2 is the larger dataset, the model can be seen to have learnt better on a larger sample size.

5.1.2 Testing

The respective models trained on dataset 1 (Model 1) and dataset 2 (Model 2) were tested on the 28 test video frames. A few of the results are compared in this section.

5.1.2.1 Case Study I: True negative -No litter object in video frame

Both models were tested on a video frame where there was clearly no litter object. Both models did not wrongly predict a non-litter object as litter. Thus, the same result was obtained in both cases. They correctly identified that there was no litter object in the video frame. This is considered a true negative. This is demonstrated in Figure 5.3.



Figure 5.3: Predictions made by models 1 and 2 (true negative)

5.1.2.2 Case Study II: False negative -No prediction made even though there was litter

In another scenario, both models wrongly predicted that there was no litter, even though there was litter. This implies that the litter object in the video frames was not detected as being litter, even though it is litter. This is considered as a false negative. This may be because of the position/angle of the litter object coupled with its small size. Both models therefore obtained the same result as shown in Figure 5.4. A red arrow is used to point out the litter object in the video frame. Only one object is pointed out because it was clearly identified as being litter. The other small objects were not selected due to uncertainty about the kind of objects they are.



Figure 5.4: Predictions made by models 1 and 2 (false negative)

5.1.2.3 Case Study III: False positives – Non-litter objects predicted to be litter objects

In some video frames, objects which were clearly not litter were wrongly predicted as litter objects. These are considered as false positives. The models generated green bounding boxes around detected objects. To add to this, they predicted the class of the identified objects as litter. The class is shown using the green labels ‘litter’. Moreover, it also included a confidence score to show how likely it is for the objects to be litter. Confidence scores typically range from 0 to 1. The higher the confidence score, the more confident the model is that the object is litter. Comparing Figure 5.5 and Figure 5.6, it can be observed that although both models correctly predict some litter objects, both also wrongly predict other objects (e.g., pavement blocks) as being litter. However, model 1 seems to predict more false positives than model 2 in Figure 5.5 and Figure 5.6. The false positives are shown using red arrows in Figure 5.5 and Figure 5.6. In addition, the false positives are identified as having low confidence score values in both cases.



Figure 5.5: Predictions made by model 1 (false positive)



Figure 5.6: Predictions made by model 2 (false positive)

5.1.2.4 Case Study IV: True positives – Correctly predicting litter objects as litter

Both models were capable of correctly identifying litter objects (true positives) in the video frames as shown in Figure 5.7 and Figure 5.8. They did so with varying confidence scores. In some cases, model 1's predictions had higher confidence scores, whereas in other cases model 2's predictions had higher confidence scores. However, model 1 made more correct predictions on smaller objects. This may be attributed to the dataset used for training model 1, which included more smaller litter objects than for the training of model 2. This is illustrated in Figure 5.7 and Figure 5.8. In this case study, the litter objects are not piled up on top of each other.

Moreover, both models did not make predictions for some litter objects which can be identified by looking at the image. These often occur at the background of the video frame. This may be

attributed to the fact that the video frames used for training mostly had litter occurring within the foreground of the video frame. An insignificant number of video frames had litter occurring at the background of the video frame. Therefore, the model may have identified litter as being an object that occurs in the forefront of the video frame.



Figure 5.7: Predictions made by model 1 (true positive)



Figure 5.8: Predictions made by model 2 (true positive)

However, in some cases, the model is able make correct predictions for all litter objects in the video frame. Figure 5.9 illustrates such a case where the model correctly predicts all the litter objects in a video frame.



Figure 5.9: Predictions made by model 1 (true positive)

5.1.2.5 Case Study V: True positives – Variation of Case Study IV

This is yet another variation of the model making correct litter predictions. In this case, large piles of litter are considered. From Figure 5.10 and Figure 5.11, it can be observed that when it comes to large litter piles, model 2 performs better. This may be attributed to the presence of more large litter objects in dataset 2 than in dataset 1.



Figure 5.10: Predictions made by model 1 (true positive)



Figure 5.11: Predictions made by model 2 (true positive)

Overall, both models can correctly identify objects in spatial video frames as being litter. Due to the dataset variations, model 1 and model 2 make more correct predictions for small litter objects and large litter objects, respectively. Both models wrongly identify a few objects as being litter when they are not litter. These objects include blocks and tyres. However, this is more predominant in model 1.

5.1.3 Evaluation Metrics

Table 5.2 summarizes the evaluation results for different metrics. These were obtained on the various threshold values used: [0.0, 0.8] in steps of 0.1. Generally, it can be observed that model 1 performs better in the recall, while model 2 performs better in the precision. Model 2 performs better in precision possibly due to the larger amount of training samples provided to it. Therefore, it predicted fewer false positives, hence leading to a higher precision. When the F1 score is considered, it is observed that both models perform better in 4 different F1 scores (highlighted in red for each model) and obtained the same results at a threshold of 0.4 (highlighted in yellow). It can also be seen that the higher the threshold value, the more litter objects are ignored due to lower matches with the predicted bounding box. From inspections made on the individual video frames, it was realized that many litter objects may be counted as false negatives when they are actually litter objects simply due to low IoU values. Considering the nature of the litter dataset, where there are a lot of smaller objects to be detected, it was deemed as a better option to have no threshold, i.e., a threshold of 0. To accurately determine locations that are blighted by litter, it is necessary to consider all true litter occurrences. Moreover, both models record the highest F1 scores when the threshold is 0. High F1 scores indicate the best point for the model; the point at which both precision and recall record high values.

A pseudo IoU-based confusion matrix was developed to better observe true positives, false positives, and false negatives. From Table 5.3 and Table 5.4, a similar trend was observed, namely that the higher the threshold, the more true positives were recorded as false negatives. The number of false positives was not affected by the threshold values, since these were non-

litter objects which were wrongly predicted to be litter. They therefore had no ground truth labels and their numbers remained constant throughout. Nonetheless, model 2 was observed to have a lower number of wrong predictions compared to model 1.

Table 5.2: Summary of evaluation metric results for models 1 and 2

Initial Model - Trained on 110 Video Frames and Tested on 28 Video Frames									
IoU Threshold	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Total ground truth labels	135	135	135	135	135	135	135	135	135
True positives (Litter)	92	88	80	70	54	37	18	7	2
False positives (Other objects)	12	12	12	12	12	12	12	12	12
False negatives (Undetected litter)	43	47	55	65	81	98	117	128	133
Precision	0,88	0,88	0,87	0,85	0,82	0,76	0,60	0,37	0,14
Recall	0,68	0,65	0,59	0,52	0,40	0,27	0,13	0,05	0,01
F1 score	0,77	0,75	0,70	0,65	0,54	0,40	0,22	0,09	0,03

Second Model - Trained on 310 Video Frames and Tested on 28 Video Frames									
IoU Threshold	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Total ground truth labels	135	135	135	135	135	135	135	135	135
True positives (Litter)	85	82	77	72	52	36	23	6	1
False positives (Other objects)	4	4	4	4	4	4	4	4	4
False negatives (Undetected litter)	50	53	58	63	83	99	112	129	134
Precision	0,96	0,95	0,95	0,95	0,93	0,90	0,85	0,60	0,20
Recall	0,63	0,61	0,57	0,53	0,39	0,27	0,17	0,04	0,01
F1 score	0,76	0,74	0,71	0,68	0,54	0,41	0,28	0,08	0,01

Table 5.3: IoU-based Pseudo-confusion matrix for model 1

IoU-Based Pseudo-Confusion Matrix for First Model - Trained on 110 Video Frames and Tested on 28 Video Frames									
Ground-truth	t = 0.0	Predicted		135	Ground-truth	t = 0.1	Predicted		135
		Litter	Not Litter				Litter	Not Litter	
		Litter	92	43			Litter	88	47
Ground-truth	t = 0.2	Predicted		135	Ground-truth	t = 0.3	Predicted		135
		Litter	Not Litter				Litter	Not Litter	
		Litter	80	55			Litter	70	65
Ground-truth	t = 0.4	Predicted		135	Ground-truth	t = 0.5	Predicted		135
		Litter	Not Litter				Litter	Not Litter	
		Litter	54	81			Litter	37	98
Ground-truth	t = 0.6	Predicted		135	Ground-truth	t = 0.7	Predicted		135
		Litter	Not Litter				Litter	Not Litter	
		Litter	18	117			Litter	7	128
Ground-truth	t = 0.8	Predicted		135	Ground-truth	t = 0.8	Predicted		135
		Litter	Not Litter				Litter	Not Litter	
		Litter	2	133			Litter	12	N/A

Table 5.4: IoU-based Pseudo-confusion matrix for model 2

IoU-based Pseudo-Confusion Matrix for Second Model - Trained on 310 Video Frames and Tested on 28 Video Frames														
Ground-truth	t = 0.0	Predicted		135	Ground-truth	t = 0.1	Predicted		135	Ground-truth	t = 0.2	Predicted		135
	Litter	85	50			Litter	82	53			Litter	77	58	
	Not Litter	4	N/A			Not Litter	4	N/A			Not Litter	4	N/A	
Ground-truth	t = 0.3	Predicted		135	Ground-truth	t = 0.4	Predicted		135	Ground-truth	t = 0.5	Predicted		135
	Litter	72	63			Litter	52	83			Litter	36	99	
	Not Litter	4	N/A			Not Litter	4	N/A			Not Litter	4	N/A	
Ground-truth	t = 0.6	Predicted		135	Ground-truth	t = 0.7	Predicted		135	Ground-truth	t = 0.8	Predicted		135
	Litter	23	112			Litter	6	129			Litter	1	134	
	Not Litter	4	N/A			Not Litter	4	N/A			Not Litter	4	N/A	

Furthermore, the manual assessment of the performance of both models compared to the IoU-based assessment at a threshold of 0 is demonstrated in Table 5.5. The results of both methods are almost similar when the true positives and false negatives are considered. The false positives report the largest differences between the two models. Model 1 has a better recall, while model 2 has a higher precision and F1 score compared to model 1. Since model 2's F1 score is higher than model 1's, it implies that it provides a better trade-off between precision and recall. Furthermore, model 2's precision far outweighs that of model 1 in the manual method and the IoU-based calculation at a threshold of zero. In addition, model 2 has fewer false positives in the manual method and the IoU-based assessment than model 1. In this study, it is desired to correctly identify litter locations and to reduce the event of wrongly classifying a location as being blighted by litter. Therefore, model 2 proves to be a better model compared to model 1.

Table 5.5: Comparison of the results obtained using manual observation of the predicted litter objects to those obtained using the IoU at a threshold of 0.

IoU Threshold	Model 1: Initial Model		Model 2	
	Manual	0.0	Manual	0.0
Total ground truth labels	135	135	135	135
True positives (Litter)	91	92	83	85
False positives (Other objects)	20	12	7	4
False negatives (Undetected litter)	43	43	52	50
Precision	0,82	0,88	0,92	0,96
Recall	0,68	0,68	0,61	0,63
F1 score	0,74	0,77	0,74	0,76

5.2 Spatial Analysis of Detected Litter Objects

The results of the automatic mapping of the litter locations as well as the results of spatial analysis of the mapped litter locations are described in the following sub-sections. The results from model 2 are used throughout this section.

5.2.1 Visualization of Litter Locations

The result of the automatic mapping process is shown in Figure 5.12. It can be observed that locations where litter was predicted were successfully displayed as point patterns on a map from the automatic mapping process. In addition, Table 5.6 shows the resulting attribute table including the number of detected litter objects per video frame. It is important to note that four objects were identified as duplicate objects. Since duplicate litter objects were only counted once, the total number of objects predicted as litter (true positives and false positives) by model 2 reduced by four. In Table 5.5, 83 true positives and 7 false positives were predicted by the model. Therefore, a total of 90 objects were predicted as litter by model 2. However, at this stage of the project, the duplicates were eliminated. As a result, only 86 litter objects were visualized. The attribute table in Table 5.6 shows this reduction.

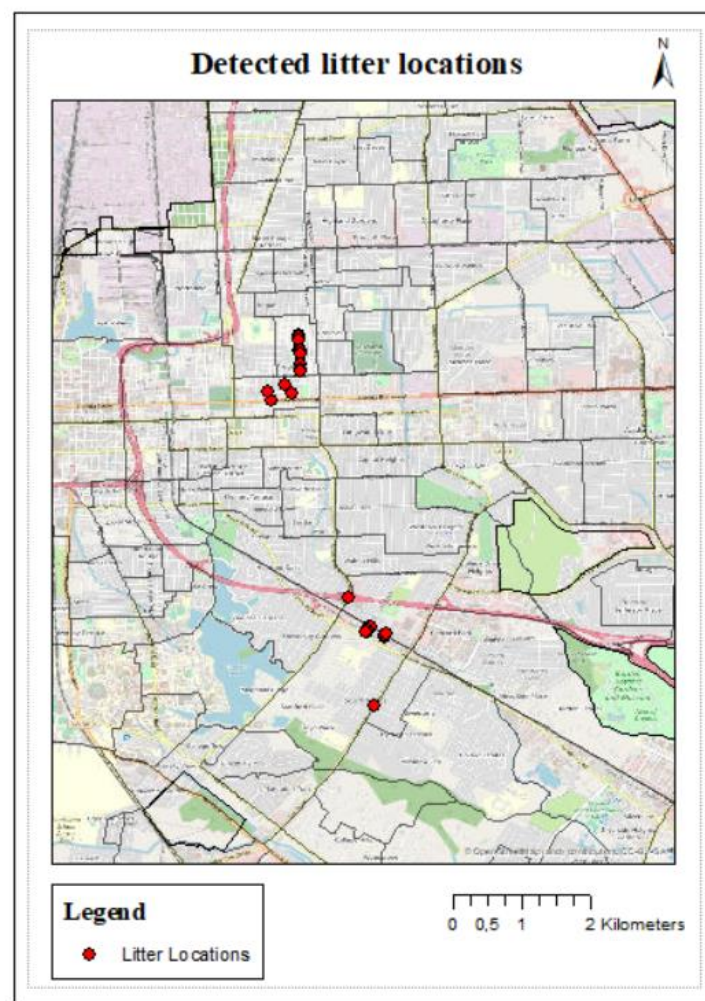


Figure 5.12: Automatically generated map of litter locations

Table 5.6: The resulting attribute table from the automatic mapping process including the number of objects detected in each video frame.

image name	latitude	longitude	Shape *	media time	number of litter objects detected
Fairfields_2_2_left_91.270834	30,458654	-91,157364	Point	91,270834	2
Fairfields_2_2_left_93.843773	30,458455	-91,157344	Point	93,843773	2
Fairfields_2_2_left_97.493917	30,458127	-91,157325	Point	97,493917	2
Fairfields_2_2_left_98.673876	30,458078	-91,157317	Point	98,673876	3
Fairfields_2_2_left_159.15401	30,456722	-91,157289	Point	159,15401	6
Fairfields_2_2_left_160.447467	30,456591	-91,157273	Point	160,447467	9
Fairfields_2_2_left_167.222202	30,456133	-91,157214	Point	167,222202	3
Fairfields_2_2_left_184.01176	30,454784	-91,15711	Point	184,01176	2
Fairfields_2_2_left_190.646618	30,454178	-91,157102	Point	190,646618	2
Fairfields_2_2_left_191.922024	30,45409	-91,1571	Point	191,922024	4
MidCity_1_left_0.389636	30,45194	-91,165797	Point	0,389636	0
MidCity_1_left_397.935932	30,452197	-91,158147	Point	397,935932	0
MidCity_1_left_1057.635215	30,450244	-91,160985	Point	1057,635215	4
MidCity_1_left_1124.770491	30,452237	-91,159163	Point	1124,770491	7
MidCity_1_left_1173.392209	30,450948	-91,158262	Point	1173,392209	3
MidCity_1_left_1435.731165	30,451183	-91,161464	Point	1435,731165	3
Southside_1_left_469.157425	30,410048	-91,147414	Point	469,157425	4
Southside_1_left_833.254359	30,419094	-91,146099	Point	833,254359	4
Southside_1_left_835.252831	30,419179	-91,146052	Point	835,252831	11
Southside_1_left_861.184831	30,419372	-91,145928	Point	861,184831	1
Southside_1_left_862.627329	30,419345	-91,145951	Point	862,627329	0
Southside_1_left_1007.328643	30,419763	-91,148413	Point	1007,328643	1
Southside_1_left_1014.362663	30,420263	-91,14811	Point	1014,362663	5
Southside_1_left_1050.321604	30,420411	-91,148012	Point	1050,321604	4
Southside_1_left_1056.874234	30,420004	-91,148265	Point	1056,874234	2
Southside_1_left_1061.774523	30,419607	-91,148528	Point	1061,774523	1
Southside_1_left_1236.145091	30,424185	-91,150868	Point	1236,145091	1

5.2.1.1 Spatial Distribution of Litter Object Detections

Figure 5.13 illustrates the spatial distribution of litter objects that were predicted by model 2. Green represents low intensity while red represents high intensity. The test dataset used does not represent the entire spatial video collection. Therefore, this mostly shows how identified litter locations can be classified in ArcMap based on the number of litter objects at the location. It is worth mentioning that a litter location may have more than one litter object. To classify an area as blighted or not, all other physical urban blight indicators must be taken into consideration. As a result, this visualization pertains to litter only. Although, the dataset is only a small section and imbalanced for each neighbourhood, it can be observed that most of the high intensity locations occur in the north where Fairfields and MidCity are located. The low intensity locations can be seen in the South where Southside is located. It is also worth mentioning that some points may not fall exactly in the middle of the road due to base map or GPS error.

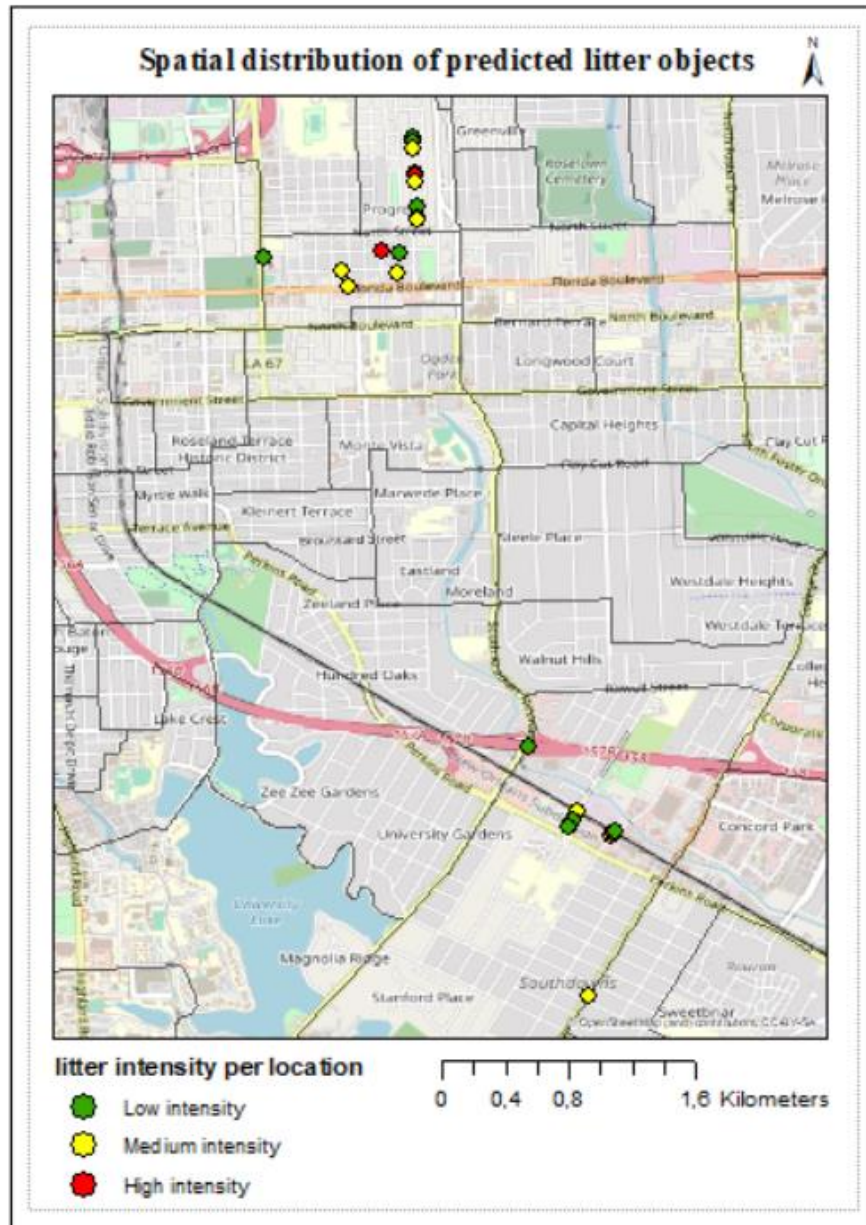


Figure 5.13: Spatial distribution of predicted litter objects

5.2.1.2 Kernel Density Estimation of Detected Litter Objects

Figure 5.14 shows the kernel density map for the litter intensity across the selected areas. Since the samples being considered are close to each other, there is a polarized effect leading to concentric circles. Nonetheless, it can be observed that the intensity of litter in the northern area (where Fairfield and MidCity are) is higher than in the Southern area (where Southside is).

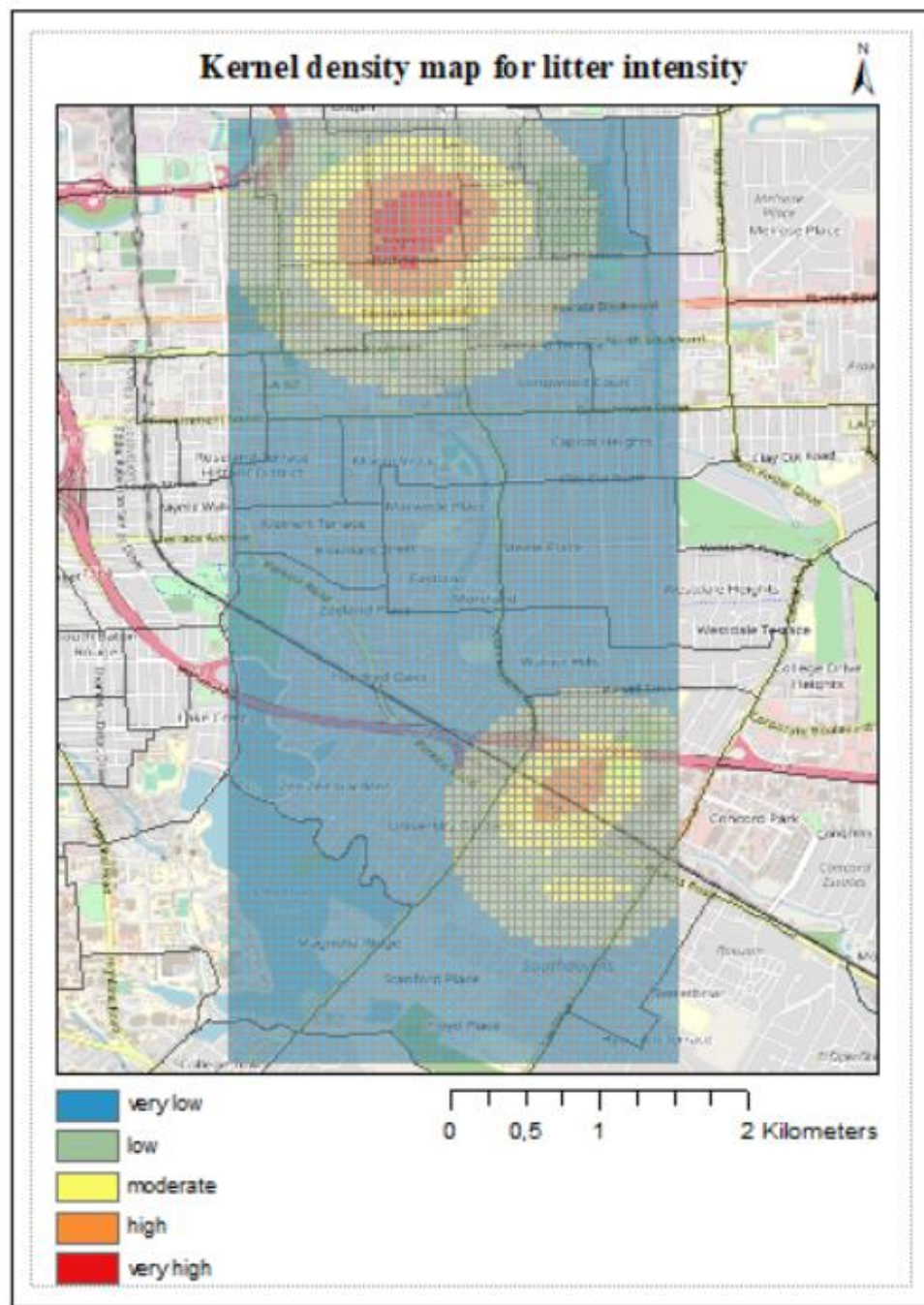


Figure 5.14: Kernel density map for litter intensity

6 Discussion

In this chapter, the results obtained are interpreted, and the benefits and limitations of this study are discussed. The challenges encountered during the implementation are also described.

6.1 Interpretation of Results

The results are interpreted in light of the research questions posed at the beginning of the study:

- How suitable is ML in detecting urban blight indicators?
 - a. How suitable are Convolutional Neural Networks (CNN) in detecting urban blight indicators?
 - b. What considerations are necessary in developing a requirement catalogue for detecting urban blight within the context of ML?

In an attempt to replicate the traditional, manual approach of detecting and mapping litter, the manual approach was confirmed as requiring a lot of effort and time. In terms of the time needed, much more time was required by the author of this thesis, due to a lack of firsthand experience with the study areas. It was therefore observed that experience with the study area in question was a major factor in the manual approach. Furthermore, there was a constant need to refer to the criteria catalogue to determine the intensity of an identified physical urban blight indicator. In spite of constantly referring to it, there was still a lot of ambiguity concerning the intensity of some indicators. Therefore, personal discretion was used. In view of these drawbacks, an automatic detection and mapping approach was deemed to be very relevant.

The workflow used in this study can be broken down into three major steps: The automatic detection of litter objects, the automatic mapping of the locations of the predicted litter objects, and the classification of litter locations. From the results obtained in the automatic detection of litter objects, it was observed that the selected CNN model, YOLOv3, was capable of accurately detecting litter objects in spatial videos. The model was seen to have a better performance when trained with a large dataset. Therefore, CNNs are suitable for detecting litter, which is one of the physical urban blight indicators. This therefore implies that ML is suitable for this task. More specifically, deep learning, which is a subset of ML is more suitable for object detection tasks.

Furthermore, to develop a requirement catalogue within the context of ML, it is necessary to consider that a variety of scenarios could occur. For instance, some video frames may contain no litter object or just one litter object. Other video frames may have multiple litter objects. These multiple litter objects may either be piled or not piled up. In the case of multiple litter objects which are not piled up, the individual litter objects may also occur at different positions or at different scales. Therefore, it may be useful to factor these different scenarios in the development of an ML-based criteria catalogue. Although sub-classes of litter objects were not

considered in this study, from experience with the spatial videos, it was also gathered that multiple litter objects can be of different types. For examples, some may be paper, bottles etc.

The following paragraphs address questions pertaining to how well the models were trained and how more litter examples will affect the models, possible reasons why the models mostly detected objects at the forefront of the video frames as well as potential reasons why the models could not detect some objects.

Model training is generally affected by different factors including the quality of the data used. In this study, high quality data was used. The spatial videos had a resolution of 1920 x 1080. In addition, during the labeling process, high data quality was further ensured by only labeling litter objects that could be clearly identified as being litter. Ambiguous objects were not labeled. Moreover, the training dataset contained a variety of real-life scenarios that were observed in the spatial videos. Therefore, the models can be said to have been trained quite well.

Nonetheless, involving other experts in the labeling process may improve the training of the models used in this study. By having other experts labeling or reviewing the created labels, the data quality may be improved and the issue of subjectivity in the training data may reduce. In addition, collecting a dataset in which background objects are very clear may also greatly lead to better trained models since more examples of litter objects occurring in the background will be provided to the models.

From the results obtained by model 1 and 2, it was observed that model 2 generally performed better due to more examples in its dataset. This led to it making fewer false positive predictions than model 1. This shows that more examples can help the model to improve its learning of what litter is. Therefore, more examples may lead to even better results.

Furthermore, the models mostly detected litter objects at the forefront of the video frames due to the training dataset used. In most of the selected video frames, litter mostly occurred at the forefront of the video frame. In other cases, it occurred at the background of the video frame. However, the latter occurrence was insignificant compared to the former. This is because the former far outnumbered the latter scenario. Moreover, most litter objects that occurred at the background could not be seen clearly and were therefore not labeled. This also affected the number of background litter object samples that were provided to the models. Therefore, the models may have learnt litter to be an object that usually occurs at the forefront of a video frame.

In terms of the failure of model 1 and 2 to detect some litter objects, it was observed that this usually happened for very small litter objects. While some small objects were detected by the models, others were not. As a result, the small size of the litter objects may be a factor that leads to some objects not being detected. Nonetheless, it is not entirely clear from this study as to why some litter objects were detected while others were not detected.

6.2 Benefits

6.2.1 Comparative Study: Traditional vs ML-based Approach

Comparing the traditional approach to the ML-based approach, the ML-based approach offers the advantage of fast detection and mapping of detected litter locations.

In addition, the implemented model can readily be tested on other spatial videos collected during the data collection process. The results will be made available within a short time. On the other hand, the traditional approach requires that the same laborious and time-intensive process be followed.

Moreover, as afore mentioned, experience with the study area was identified as having a large impact on the ease of the traditional process, as well as on the time taken to complete the process. On the other hand, experience with the study area does not greatly impact the ML-based approach. While it may be an advantage to properly classify litter objects, especially if various litter types are to be detected, the model is not greatly affected.

6.2.2 Benefits to Stakeholders

The outcome of this study has several benefits for different stakeholders including local governments, decision makers, urban planners, real-estate workers and researchers in diverse fields including criminology, disaster management, health, and graffiti research.

Governments who are interested in fighting urban blight can now assess urban blight in their cities/districts/states more quickly. Rather than relying on occasional ground-level inspections and other methods which are not time-efficient and require more resources, this study provides a more time-efficient and cheaper approach for identifying urban blight.

Moreover, decision makers and urban planners who are interested in assessing urban blight at a location in order to make important decisions about budget allocations, planning of urban areas etc., can also benefit from this study. They can also assess urban blight at a location more quickly. This may help to speed up their decision-making processes and support them to make good decisions since relevant and relatively accurate information about urban blight at a selected location can be accessed more quickly. Real-estate agents can also benefit since such this study can help to quickly assess urban blight in an area and make decisions on how to make their sales.

This study can also enhance the work of researchers in diverse fields. Researchers who are interested in testing hypothesis or theories such as the Broken Windows Theory can do so more easily. To add to this, the possibility to quickly assess urban blight in a location can help health researchers who are interested in determining the effect of urban blight on people's health. Furthermore, by extending this study to other physical urban blight indicators such as structural integrity, broken windows and graffiti, the impact of a disaster on building structures can easily be determined. To add to this, the work of researchers who are interested in graffiti research is enhanced.

6.3 Challenges/Limitations

In the quest to implement the YOLOv3 pre-trained model, several errors were encountered. These include: ‘Module ‘tensorflow’ has no attribute ‘get_default_graph’’, ‘ImportError: cannot import name ‘multi_gpu_model’, and ‘ImportError: cannot import name ‘get_config’ from ‘tensorflow.python.eager.context’ (/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/context.py)’. The discrepancies between different versions of keras and tensorflow was identified as a possible reason. The errors were solved by changing from ‘keras’ to ‘tensorflow.keras’. This was coupled with installing the requirements file in the model’s repository as part of the model implementation steps in Colab. Figure 6.1 summarizes the attempted solutions. The solution that worked is highlighted in green while those that failed are highlighted in red.

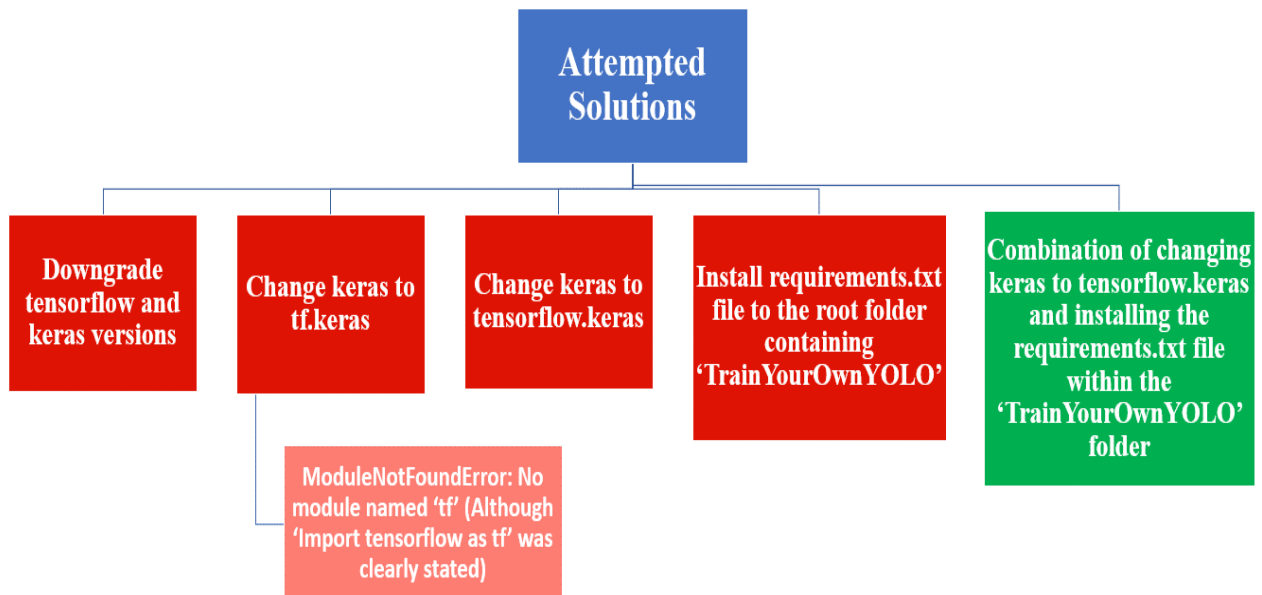


Figure 6.1: Attempted solutions for encountered model implementation errors

There are several limitations to this study. The model currently works on only extracted video frames, rather than on a whole video. However, it may be applied to videos. Further tests are required to determine how to successfully do this. Furthermore, the general category ‘litter’ is considered rather than specific types of litter (e.g., paper, food package). In addition, litter is used as the main physical urban blight indicator in the study. Hence, to derive a complete urban blight concept, the workflow must be tested on the other blight indicators, as well. To add to this, the litter locations are (video) ‘frame-based’ locations rather than the specific location of each litter instance. Moreover, the ground-truth labels created were not evaluated by other experts, as well. This implies that some subjectivity may be introduced into the model at the data labeling stage.

7 Conclusion

This study presented a workflow to automatically detect physical urban blight in spatial videos, map the detected physical urban blight indicators, and classify the locations where these indicators occur. Physical urban blight indicators were the focus of this study. More specifically, litter was selected as the main physical urban blight indicator in this study.

Secondary data was used in this study. The data covered five neighbourhoods in Baton Rouge namely Fairfields, MidCity, Southside, Pollard/Woodchase and University Acres. The data collection was done using the spatial video technology, which allows for fine-contextual data to be obtained. GPS sensors were linked to the cameras used and as a result, the GPS information of a location within a video frame could be obtained. For the purposes of this study, Fairfields, MidCity and Southside were selected as the main study areas. The first two neighbourhoods were selected because of the high density of physical urban blight in those areas while Southside was added for additional model testing purposes. Specific sub-areas within Fairfields and MidCity were selected based on whether there was a high presence of both property and environmental/infrastructural blight. The corresponding spatial videos of the selected sub-areas were identified and used in this study.

To automatically detect litter in spatial videos, it was necessary to make use of an object detection algorithm. The YOLOv3 model was selected due to its capability of detecting objects in videos including spatial videos. In addition, compared to other object detection models such as RetinaNet, it offered a better trade-off between accuracy and detection speed. To achieve a meaningful weight initialization faster, a pre-trained YOLOv3 model was adopted in this study.

For the YOLOv3 model to detect litter in the spatial videos, the model was first provided with examples of litter during the training and validation stage. These examples were created by extracting video frames from the selected spatial videos, selecting suitable video frames, labeling the litter objects in the video frames by drawing a bounding box around the identified litter objects, and then converting the resulting annotation file to the appropriate YOLO format. The training and validation dataset were split by a 90:10 ratio. After training and validating the model, the model was tested on video frames which it had not seen before. The predictions made by the model were then evaluated using evaluation metrics such as the Intersection over Union, Precision, Recall and F1-score. The evaluation was done at different thresholds [0.0, 0.8] in steps of 0.1. However, a threshold of 0.0 was identified to be optimum for this study due to the varying nature and small sizes of the litter objects.

It is worth mentioning that the research was ran two times; the first time with a small dataset of 138 video frames (469 litter objects) and the second time with a larger data set of 338 video frames (1,141 litter objects). The 138 video frames of the first smaller data set were also included in the larger data set of 338 frames. The same test video frames were used in the two runs of this research. Since the datasets vary, two different models - models 1 and 2 - were obtained.

At the training and validation stage, model 2 recorded lower training and validation losses compared to model 1. The results from the testing of both models, showed that both models could detect litter objects in the video frames with confidence scores as high as 0.95. In some

cases, the confidence scores were as low as 0.30. However, lower confidence scores were mostly observed for some small litter objects and false positives. To add to this, model 2 made fewer false positive predictions compared to model 1. Since it was desired to reduce the number of false positives in this study and avoid wrongly classifying a location as blighted, model 2 was selected as the better model.

The predictions made by model 2 were used in the automatic mapping and location classification process. The predicted litter objects were linked to their corresponding GPS coordinates. These predicted litter objects were then visualized as points on a map layer within a GIS program. Locations where litter objects were predicted were classified as having either a low, medium or high intensity of litter. This was done based on the number of litter objects at each location.

When compared to the traditional approach of detecting and mapping physical urban blight, the ML-based method was observed to be faster. Therefore, the ML-based approach was identified as a time-efficient approach. Furthermore, the implemented ML-based approach can readily be extended to other videos collected during the data collection process without demanding a lot of effort. The traditional approach on the other hand is time-intensive and laborious.

This study has some limitations. The model currently works on only extracted video frames, rather than on a whole video. Furthermore, the general category ‘litter’ is considered rather than specific types of litter (e.g., paper, food package). In addition, litter is used as the main physical urban blight indicator in the study. Hence, to derive a complete urban blight concept, the workflow must be tested on the other blight indicators, as well. To add to this, the litter locations are (video) ‘frame-based’ locations rather than the specific location of each litter instance. Moreover, the ground-truth labels created were not evaluated by other experts, as well. This implies that some subjectivity may be introduced into the model at the data labeling stage.

8 Future Work

This chapter describes possible future research directions that may be explored in future studies.

Further tests may be performed to assess how the model can be tested directly on spatial videos. In this study, extracted video frames were labeled and then used for training, validating and testing the model. This resulted in duplicates which were accounted for prior to the automatic process. Nonetheless, it would be interesting to explore how directly labeled videos impact the model's performance. It would also be valuable to test the workflow on spatial videos of other areas.

In addition, the identified workflow may be tested on the other physical urban blight indicators. Considerations for each physical urban blight indicator will be relevant for obtaining a model that works well for all the blight indicators. This is because the physical urban blight indicators vary. For example, one blight indicator may occur at different scales (e.g., close to the road, far away from the road) and in different positions. Moreover, different physical urban blight indicators also vary in terms of size. For example, litter objects are much smaller than objects that are considered under the physical urban blight indicator 'illegal dumping'. Due to this vast variation, it is assumed that the selected model may not perform in equal measure on all the physical urban blight indicators. In such a case, it may be useful to explore how an approach such as ensemble learning can be used to improve the results. In ensemble learning, several models are trained for a problem and then combined to improve the results obtained (Rocca 2019).

Since the YOLOv3 model could successfully detect objects in the spatial videos, this study may also be applicable in disaster efforts. For example, detecting structural damage of houses after an earthquake. Other areas which may be explored include detecting potholes on a street from a video footage collected from a drone flying above the street surface.

Furthermore, the data quality may be improved using a combination of the following approaches in Labelbox: labeling of the litter objects by a number of experts and making use of the quality assurance tools provided in Labelbox (review of labels, defining benchmarks, consensus).

To add to this, it will be helpful to explore how to extract frames while preserving the exif. Moreover, the identification of specific litter classes may prove to be a valuable step due to smart city implementations and the like.

It is worth mentioning that the software used in this study are all open source. Therefore, this approach may be a much cheaper option for local governments and academic researchers due to their financial budget constraints.

References

- Abadi, M., *et al.*, 2016. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*.
- Ajayakumar, J., *et al.*, 2021. Exploring convolutional neural networks and spatial video for on-the-ground mapping in informal settlements. *International Journal of Health Geographics*, 20 (1), 5. Available from: <https://ij-healthgeographics.biomedcentral.com/articles/10.1186/s12942-021-00259-z>.
- Aloysius, N., and Geetha, M., 2017. A review on deep convolutional neural networks. *In*: 2017: IEEE.
- Amazon Web Services, Inc., 2021. *Amazon SageMaker Ground Truth Pricing / AWS* [online]. Available from: <https://aws.amazon.com/sagemaker/groundtruth/pricing/> [Accessed 22 July 2021].
- Athens, J., *et al.*, 2020. Using 311 data to develop an algorithm to identify urban blight for public health improvement., 15 (7). Available from: 10.1371/journal.pone.0235227 [Accessed 19 January 2021].
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y.M., 2020. *YOLOv4: Optimal Speed and Accuracy of Object Detection*.
- Braga, A.A., Welsh, B.C., and Schnell, C., 2015. Can Policing Disorder Reduce Crime? A Systematic Review and Meta-analysis. *Journal of Research in Crime and Delinquency*, 52 (4), 567–588.
- Branas, C.C., *et al.*, 2018. Citywide cluster randomized trial to restore blighted vacant land and its effects on violence, crime, and fear. *Proceedings of the National Academy of Sciences of the United States of America*, 115 (12), 2946–2951.
- Bratton, W., and Kelling, G., 2006. *There are no cracks in the broken windows*.
- Brownlee, J., 2018. How to Avoid Overfitting in Deep Learning Neural Networks. *Machine Learning Mastery*, 2018.
- Brownlee, J., 2019. How Do Convolutional Layers Work in Deep Learning Neural Networks? *Machine Learning Mastery*, 2019.
- Carolis, B.D., Ladogana, F., and Macchiarulo, N., 2020. YOLO TrashNet: Garbage Detection in Video Streams. *In*: . 2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS) 2020, 1–7.
- Curtis, A., *et al.*, 2012. A Methodology for Assessing Dynamic Fine Scale Built Environments and Crime: A Case Study of the Lower 9th Ward After Hurricane Katrina. *In*: M. Leitner, ed. *Crime modeling and mapping using geospatial technologies*. Dordrecht, London: Springer, 203–225.
- Curtis, A., *et al.*, 2013a. A ubiquitous method for street scale spatial data collection and analysis in challenging urban environments: mapping health risks using spatial video in Haiti. *International Journal of Health Geographics*, 12 (1), 21. Available from: <https://ij-healthgeographics.biomedcentral.com/articles/10.1186/1476-072X-12-21>.

- Curtis, A., *et al.*, 2015. Spatial video geonarratives and health: case studies in post-disaster recovery, crime, mosquito control and tuberculosis in the homeless. *International Journal of Health Geographics*, 14 (1), 22. Available from: <https://ij-healthgeographics.biomedcentral.com/articles/10.1186/s12942-015-0014-8>.
- Curtis, A., *et al.*, 2016. Mapping to Support Fine Scale Epidemiological Cholera Investigations: A Case Study of Spatial Video in Haiti. *International Journal of Environmental Research and Public Health*, 13 (2), 187. Available from: <https://www.mdpi.com/1660-4601/13/2/187/htm>.
- Curtis, J.W., *et al.*, 2013b. Using Google Street View for systematic observation of the built environment: analysis of spatio-temporal instability of imagery dates. *International Journal of Health Geographics*, 12 (1), 53. Available from: <https://ij-healthgeographics.biomedcentral.com/articles/10.1186/1476-072X-12-53>.
- Darling, P.V., 1943. Some Notes on Blighted Areas. *Journal of the American Institute of Planners*, 9 (1), 9–18. Available from: <https://www.tandfonline.com/doi/pdf/10.1080/01944364308979041>.
- Ferreira, F.A.F., *et al.*, 2018. A prioritisation index for blight intervention strategies in residential real estate. *Journal of the Operational Research Society*, 69 (8), 1269–1285.
- FFmpeg, 2021. *About FFMpeg* [online]. Available from: <https://ffmpeg.org/about.html> [Accessed 2 June 2021].
- G2, 2021. *Labelbox Alternatives & Competitors / G2* [online]. Available from: <https://www.g2.com/products/labelbox/competitors/alternatives> [Accessed 20 May 2021].
- Gad, A.F., 2020. Mean Average Precision (mAP) Explained | Paperspace Blog. *Paperspace Blog*, 2020.
- Gault, M., and Silver, E., 2008. Spuriousness or mediation? Broken windows according to Sampson and Raudenbush (1999). *Journal of Criminal Justice*, 36 (3), 240–243. Available from: <https://www.sciencedirect.com/science/article/pii/S0047235208000457>.
- Gaur, E., Saxena, V., and Singh, S.K., 2018. Video annotation tools: A Review. In: . 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN) 2018, 911–914.
- Girshick, R., *et al.*, 2013. *Rich feature hierarchies for accurate object detection and semantic segmentation*.
- Google, 2021. *Colaboratory – Google* [online]. Available from: <https://research.google.com/colaboratory/faq.html> [Accessed 5 January 2021].
- Haney, T.J., 2007. “Broken windows” and Self-Esteem: Subjective understandings of neighborhood poverty and disorder. *Social Science Research*, 36 (3), 968–994.
- Harcourt, B.E., 2009. *Illusion of Order. The False Promise of Broken Windows Policing*: Harvard University Press.
- Hinkle, J.C., and Weisburd, D., 2008. The irony of broken windows policing: A micro-place study of the relationship between disorder, focused police crackdowns and fear of crime. *Journal of Criminal Justice*, 36 (6), 503–512. Available from:

https://www.sciencedirect.com/science/article/pii/S0047235208001128?casa_token=zg0tj8ygrlkaaaaa:z6gwi6mykb-jooet198h_i6mn8yg6n1nkbvbwof4jr5aethekw2dy28usirmzpu5crnbrrlbzqc.

Hinton, G.E., and Salakhutdinov, R.R., 2006. Reducing the Dimensionality of Data with Neural Networks. *Science*.

Hoffman, J., 2012. Raze the dead: Urban blight, private universities, and the path towards revitalization. *University of Pittsburgh Law Review*, 74 (1), 85–105.

Howard, A.G., *et al.*, 2017. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*.

iLiveHere, 2021. *Litter & Illegal Dumping* [online]. Available from: <https://www.ilivehereqc.org/Content/Litter-Illegal-Dumping.aspx> [Accessed 30 April 2021].

Jana, A.P., Biswas, A., and Mohana, 2018. YOLO based Detection and Classification of Objects in video records. *In*: . 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) 2018, 2448–2452.

Jia, Y., *et al.*, 2014. Caffe. *In*: K.A. Hua, *et al.*, eds. MM '14: 2014 ACM Multimedia Conference, 03 11 2014 07 11 2014 2014. New York, New York: Association for Computing Machinery, 675–678.

Jogin, M., *et al.*, 2018. Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning. *In*: . 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) 2018, 2319–2323.

Julie, 2014. How to Take Batch Screenshots or Screensaps in VLC Media Player. *TurboFuture*, 2014.

Kavasidis, I., *et al.*, 2014. An innovative web-based collaborative platform for video annotation. *Multimedia Tools and Applications*, 70 (1), 413–432. Available from: <https://link.springer.com/article/10.1007/s11042-013-1419-7>.

Kelling, G.L., and Wilson, J.Q., 1982. Broken Windows: The police and neighborhood safety. *Atlantic Monthly*, 249 (3), 29–38.

Kumar, A., 2021. *Machine Learning - Training, Validation & Test Data Set - Data Analytics* [online]. Available from: <https://vitalflux.com/machine-learning-training-validation-test-data-set/> [Accessed 11 August 2021].

Labelbox, 2020. *Labelbox: Pricing* [online]. Available from: <https://labelbox.com/pricing> [Accessed 10 January 2021].

Labelbox, 2021. *Labelbox: The leading training data platform for data labeling*.

LeCun, Y., Bengio, Y., and Hinton, G., 2015. Deep learning. *Nature*, 521 (7553), 436–444.

Liu, L., *et al.*, 2020. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128 (2), 261–318. Available from: <https://link.springer.com/article/10.1007/s11263-019-01247-4>.

Lousada, A.L., *et al.*, 2021. A sociotechnical approach to causes of urban blight using fuzzy cognitive mapping and system dynamics. *Cities*, 108.

- Maghelal, A., *et al.*, 2013. *From Blight to Light. Assessing Blight in the City of Dallas. Final Report*: Department of Public Administration, University of North Texas.
- Marco, M., *et al.*, 2017. Validation of a Google Street View-Based Neighborhood Disorder Observational Scale. *Journal of urban health : bulletin of the New York Academy of Medicine*, 94 (2), 190–198.
- Mills, J.W., *et al.*, 2010. Geospatial video for field data collection. *Applied Geography*, 30 (4), 533–547. Available from: https://www.researchgate.net/publication/248337913_Geospatial_video_for_field_data_collection.
- Minhas, M.S., 2021. Techniques for handling underfitting and overfitting in Machine Learning. *Towards Data Science*, 2021.
- Mohana, and Aradhya, H.V.R., 2016. Elegant and efficient algorithms for real time object detection, counting and classification for video surveillance applications from single fixed camera. In: . 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), October 4-6, 2016 2016. [Piscataway, New Jersey]: IEEE, 1–7.
- Mohri, M., Rostamizadeh, A., and Talwalkar, A., 2018. *Foundations of Machine Learning, second edition*: MIT Press.
- Muehlemann, A., 2019. *TrainYourOwnYOLO/1_Image_Annotation at master · AntonMu/TrainYourOwnYOLO* [online]. Available from: https://github.com/AntonMu/TrainYourOwnYOLO/tree/master/1_Image_Annotation [Accessed 5 August 2021].
- Muehlemann, A., 2021. *TrainYourOwnYOLO: Building a Custom Object Detector from Scratch*: Zenodo.
- Ongsulee, P., 2017. Artificial intelligence, machine learning and deep learning. In: 2017: IEEE.
- O'Shea, K., and Nash, R., 2015. *An Introduction to Convolutional Neural Networks*.
- Oxford Dictionaries, 2019. *Blight* [online]. Available from: <https://en.oxforddictionaries.com/definition/blight>.
- Oxford Learner's Dictionaries, 2021. *video_1 noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com* [online]. Available from: https://www.oxfordlearnersdictionaries.com/definition/english/video_1?q=video [Accessed 4 September 2021].
- Packt, P., 2018. *Categories of Machine Learning - Towards Data Science* [online]. Available from: <https://towardsdatascience.com/categories-of-machine-learning-8b427b3a7374> [Accessed 10 January 2021].
- Paszke, A., *et al.*, 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32. Available from: <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-highperformance-deep-learning-library>.

- Pathak, A.R., Pandey, M., and Rautaray, S., 2018. Application of Deep Learning for Object Detection. *Procedia Computer Science*, 132, 1706–1717. Available from: <https://www.sciencedirect.com/science/article/pii/S1877050918308767>.
- Ping Ping, *et al.*, 2020. *Smart street litter detection and classification based on Faster R-CNN and edge computing*.
- Pough, B., and Wan, Q., 2007. *Data analytics and the fight against housing blight: A guide for local leaders*.
- Redmon, J., and Farhadi, A., 2017. YOLO9000: Better, Faster, Stronger. *In*: 2017: IEEE.
- Redmon, J., and Farhadi, A., 2018. *YOLOv3: An Incremental Improvement*.
- Reyes, E.B., *et al.*, 2016. *Early detection of properties at risk of blight using spatiotemporal data*.
- Riegl, B., Korrubel, J.L., and Martin, C., 2001. Mapping and monitoring of coral communities and their spatial patterns using a surface-based video method from a vessel. *Bulletin of Marine Science -Miami-*, 69 (2). Available from: https://www.researchgate.net/publication/263337005_Mapping_and_monitoring_of_coral_communities_and_their_spatial_patterns_using_a_surface-based_video_method_from_a_vessel.
- Ristea, A., *et al.*, 2021. Applying Spatial Video Geonarratives and Physiological Measurements to Explore Perceived Safety in Baton Rouge, Louisiana. *International Journal of Environmental Research and Public Health*, 18 (3), 1284. Available from: <https://www.mdpi.com/1660-4601/18/3/1284/htm>.
- Rocca, J., 2019. Ensemble methods: bagging, boosting and stacking - Towards Data Science. *Towards Data Science*, 2019.
- Ross, C.E., and Mirowsky, J., 2001. Neighborhood Disadvantage, Disorder, and Health. *Journal of Health and Social Behavior*, 42 (3), 258.
- Sampson, R.J., and Raudenbush, S.W., 1999. Systematic Social Observation of Public Spaces: A New Look at Disorder in Urban Neighborhoods. *American Journal of Sociology*, 105 (3), 603–651.
- Sampson, R.J., and Raudenbush, S.W., 2001. *Disorder in urban neighbourhoods: Does it lead to crime*: American Psychological Association (APA).
- Samuel, A.L., 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3 (3), 210–229.
- Singh, L., 2018. *Extract Frames from Video with These Free Software* [online]. Available from: <https://www.ilovefreesoftware.com/23/featured/extract-frames-from-video-free-software.html> [Accessed 2 May 2021].
- Skogan, W.G., 1990. *Disorder and Decline: Crime and the Spiral of Decay in American Neighborhoods*: New York: Free Press.
- Skogan, W.G., 2008. BROKEN WINDOWS: WHYAND HOWWE SHOULD TAKE THEM SERIOUSLY. *Criminology & Public Policy*, 7 (2), 195–201.

- Stratmann, J., 2019. *Fine-scale Analysis and Modeling of Urban Blight and Crime Applying Geospatial Technology: A Case Study in Baton Rouge, Louisiana. (Unpublished Master Thesis)*. Master's thesis: Spatial Information Management, Carinthia University of Applied Sciences.
- Stratmann, J., *et al.*, 2020. Exploring Urban 'Blightsapes' Applying Spatial Video Technology and Geographic Information System. *In*: D. Edler, C. Jenal, and O. Kühne, eds. *Modern Approaches to the Visualization of Landscapes*. Wiesbaden: Springer Fachmedien Wiesbaden, 497–515.
- Strelnikova, D., Schneider, T., and Leitner, M., 2018. Utilizing Spatial Video to Analyse Roadside Advertisements in Villach, Austria. *GI_Forum*, 1, 34–46. Available from: https://www.researchgate.net/publication/326184430_Utilizing_Spatial_Video_to_Analyse_Roadside_Advertisements_in_Villach_Austria.
- Sun, W., *et al.*, 2019. Neighborhood Blight Indices, Impacts on Property Values and Blight Resolution Alternatives. *Journal of Real Estate Research*, 41 (4), 555–603.
- Tang, C., *et al.*, 2017. The Object Detection Based on Deep Learning. *In*: . 2017 4th International Conference on Information Science and Control Engineering (ICISCE), 7/21/2017 - 7/23/2017 2017: IEEE, 723–728.
- The Theano Development Team, *et al.*, 2016. *Theano: A Python framework for fast computation of mathematical expressions*.
- van Bakergem, M., *et al.*, 2017. Objective reports versus subjective perceptions of crime and their relationships to accelerometer-measured physical activity in Hispanic caretaker-child dyads. *Preventive Medicine*, 95 Suppl, S68-S74.
- Voulodimos, A., *et al.*, 2018. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018, 7068349. Available from: <https://www.hindawi.com/journals/cin/2018/7068349/>.
- Weisburd, D., *et al.*, 2010. Legitimacy, Fear and Collective Efficacy in Crime Hot Spots: Assessing the Impacts of Broken Windows Policing Strategies on Citizen Attitudes. Available from: <https://www.ojp.gov/pdffiles1/nij/grants/239971.pdf> [Accessed 8 February 2021].
- Wu, X., Sahoo, D., and Hoi, S.C., 2020. Recent advances in deep learning for object detection. *Neurocomputing*, 396, 39–64. Available from: <https://www.sciencedirect.com/science/article/pii/S0925231220301430>.
- Xiao, Y., *et al.*, 2020. A review of object detection based on deep learning. *Multimedia Tools and Applications*, 79 (33-34), 23729–23791. Available from: <https://link.springer.com/article/10.1007/s11042-020-08976-6>.
- Yang, G., *et al.*, 2020. Face Mask Recognition System with YOLOV5 Based on Image Recognition. *In*: . 2020 IEEE 6th International Conference on Computer and Communications (ICCC), 12/11/2020 - 12/14/2020 2020: IEEE, 1398–1404.
- Zha, S., *et al.*, 2015. *Exploiting Image-trained CNN Architectures for Unconstrained Video Classification*.

- Zhao, Z.-Q., *et al.*, 2019. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30 (11), 3212–3232.
- Zhou, X., *et al.*, 2017. Application of deep learning in object detection. *In*: 2017: IEEE.
- Zhu, H., *et al.*, 2020. A Review of Video Object Detection: Datasets, Metrics and Methods. *Applied Sciences*, 10 (21), 7834. Available from: <https://www.mdpi.com/2076-3417/10/21/7834/htm>.
- Zou, Z., *et al.*, 2019. *Object Detection in 20 Years: A Survey*.

Appendix A. Python Code to Convert Labelbox Annotation File to YOLO Format

The following figure shows the python code that was used to convert the Labelbox annotation file to the appropriate YOLO format.

```
1 # Author: Vanessa Gbekor
2 # Assisted by: Clement Nartey
3 # Date: 15.06.2021
4
5 import pandas as pd
6 import json
7 import csv
8
9 with open(r'<file_path.json>') as f:
10     data = json.load(f)
11
12 df = pd.DataFrame(data)
13 print(df)
14 label=df['Label']
15 all_image_names = df['External ID']
16
17 # create annotations table
18 df_annotations = pd.DataFrame(columns= ['image', 'xmin', 'ymin', 'xmax', 'ymax', 'label'])
19
20 # get all labels
21 for i in range(len(label)):
22     image_name = all_image_names[i]
23     labelled_objects = label[i]['objects']
24
25 # for each of the labels get the title and bbox
26     for j in labelled_objects:
27         title = j["title"]
28         bbox = j["bbox"]
29
30 # perform the calculation for xmax and ymax
31         xmin = bbox['left']
32         ymin = bbox['top']
33         xmax = xmin + bbox['width']
34         ymax = ymin + bbox['height']
35
36 # insert the values into a row in the table for annotations
37         df_annotations = df_annotations.append(
38             {
39                 'image': image_name,
40                 'xmin': xmin,
41                 'ymin': ymin,
42                 'xmax': xmax,
43                 'ymax': ymax,
44                 'label': title
45             }, ignore_index=True
46         )
47
48 #print(df_annotations)
49
50 # export to a csv file
51 df_annotations.to_csv(r'<file_path>.csv', sep=',', index=False)
```

Appendix B. Python Code to Calculate Evaluation Metrics

The following figures show the python code used to identify the corresponding ground truth and predicted bounding boxes, and calculate the following evaluation metrics: IoU, Precision, Recall and F1-score. The various codes are connected although each image starts numbering from 1.

```
1 # Author: Vanessa Gbekor
2 # Assisted by: Clement Nartey
3 # Date: 25.06.2021
4 # Code adapted from: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
5 # Code access date: 15.06.2021
6
7
8 # import the required packages
9 from collections import namedtuple
10 import numpy as np
11 import cv2
12 import pandas as pd
13
14 # *****
15 # Functions to use in script
16 # *****
17
18 # perform the calculation of the IoU
19
20
21 def calculate_IoU(box_A, box_B):
22
23     # get the bounding box information/(x,y) coordinates of the intersection rectangle
24     xA = max(box_A[0], box_B[0])
25     yA = max(box_A[1], box_B[1])
26     xB = min(box_A[2], box_B[2])
27     yB = min(box_A[3], box_B[3])
28
29     # compute the area of overlap
30     # the value 1 is added to prevent the numerator from being 0
31     area_of_overlap = max(0, xB - xA + 1) * max(0, yB - yA + 1)
32
33     # compute the area of both the prediction and ground-truth bounding boxes
34     box_A_area = (box_A[2] - box_A[0] + 1) * (box_A[3] - box_A[1] + 1)
35     box_B_area = (box_B[2] - box_B[0] + 1) * (box_B[3] - box_B[1] + 1)
36
37     # the intersection over union formula
38     intersection_over_union = area_of_overlap / \
39         float(box_A_area + box_B_area -
40             area_of_overlap) # subtract area of overlap to avoid double counting
41
42     return intersection_over_union
43
```

```

1 # match the predicted bboxes to their corresponding ground truth bboxes
2 def bb_intersection_over_union(box_gt_list, box_pred_list):
3     IoU_list = []
4
5     used_pred_ind = []
6
7     used_gt_ind = []
8
9     # compare each ground truth bounding box to all the prediction bounding boxes
10    for box_gt_ind in range(0, len(box_gt_list)):
11        box_gt = box_gt_list[box_gt_ind]
12
13        # define variables to keep track of max IoU for the comparisons
14        max_IoU = -1
15        max_pred_ind = -1
16
17        for box_pred_ind in range(0, len(box_pred_list)):
18            box_pred = box_pred_list[box_pred_ind]
19            IoU_box = calculate_IoU(box_gt, box_pred)
20
21            # if the IoU for the current prediction bbox is greater than the previous pred bbox, change the max value
22            if(IoU_box > max_IoU):
23                max_IoU = IoU_box
24                max_pred_ind = box_pred_ind
25
26            # keeping track of the indices for the prediction bounding boxes used
27
28            # if the ground truth has a corresponding pred box then keep track of it
29            if (max_IoU >= 0): # setting it at 0 makes it ignore some gt labels
30                used_gt_ind.append(box_gt_ind)
31                used_pred_ind.append(max_pred_ind)
32
33            IoU_list.append(
34                {"gt_box": box_gt, "pred_box": box_pred_list[max_pred_ind], "IoU": max_IoU})
35
36    # dealing with unequal number of bboxes in gt and pred bbox lists
37
38    # if there are more bboxes in the gt list than in the pred list, assign 0s to the missing bboxes in the pred list
39    if (len(box_gt_list) > len(box_pred_list)):
40        for gt_ind in range(0, len(box_gt_list)):
41            if gt_ind not in used_gt_ind:
42                gt_IoU_box = calculate_IoU(box_gt_list[gt_ind], [0, 0, 0, 0])
43                IoU_list.append({"gt_box": box_gt_list[gt_ind], "pred_box": [
44                    0, 0, 0, 0], "IoU": gt_IoU_box})
45
46    # if there are more bboxes in the pred list than in the gt list, match the unmatched bboxes to gt boxes filled with 0s
47    if (len(box_pred_list) > len(box_gt_list)):
48        for pred_ind in range(0, len(box_pred_list)):
49            if pred_ind not in used_pred_ind:
50                pred_IoU_box = calculate_IoU(
51                    [0, 0, 0, 0], box_pred_list[pred_ind])
52                IoU_list.append(
53                    {"gt_box": [0, 0, 0, 0], "pred_box": box_pred_list[pred_ind], "IoU": pred_IoU_box})
54
55    # return the intersection over union value
56    return IoU_list

```

```

1  # *****
2  # Main Code Section
3  # *****
4
5  # set a range of IoU thresholds to be tested in order to identify the best threshold
6  IoU_threshold_start = 0.0
7  IoU_threshold_stop = 0.8
8  step_value = 0.1
9
10 """ define a list to keep all the values for the true positives, false positives and
11     false negatives which were calculated based on the threshold"""
12 threshold_labels_list = []
13
14 # define a list to hold image names which are in the pred file and not in the ground truth
15 images_in_pred_not_gt = []
16
17 # read the csv files containing the ground truth and prediction bounding box information
18 ground_truth_file_df = pd.read_csv('<file_name>.csv')
19 # print(ground_truth_file_df)
20
21 prediction_file_df = pd.read_csv('<file_name>.csv')
22 # print(prediction_file_df)
23
24 # get the unique names of the images existing in the ground truth and prediction files
25 unique_ground_truth_image_list = ground_truth_file_df['image'].unique()
26 unique_pred_image_list = prediction_file_df['image'].unique()
27
28 print("{} images from the ground truth file are not in the prediction file".format(
29     len(unique_ground_truth_image_list) - len(unique_pred_image_list)))
30
31 """ define the 'Detection' object
32 # image_path is the path to the input image
33 # gt is the ground-truth bounding box information
34 # pred is the predicted bounding box information as detected by the model"""
35 Detection = namedtuple("Detection", ["image_path", "gt", "pred"])
36
37 for image_name in unique_ground_truth_image_list:
38     if image_name in unique_pred_image_list:
39         image_rows_pred = prediction_file_df[prediction_file_df['image'] == image_name]
40         # print(image_rows_pred)
41         image_rows_groundtruth = ground_truth_file_df[ground_truth_file_df['image'] == image_name]
42
43         b_box_list_per_image_pred = []
44         b_box_list_per_image_groundtruth = []
45
46         # identify and store the bbox info for all the detected objects in an image
47         for index, b_box_row in image_rows_pred.iterrows():
48             b_box = [
49                 b_box_row['xmin'],
50                 b_box_row['ymin'],
51                 b_box_row['xmax'],
52                 b_box_row['ymax']
53             ]
54             b_box_list_per_image_pred.append(b_box)
55         # print(b_box_list_per_image_pred)
56
57         # identify and store the bbox info for all labelled objects in an image
58         for index, b_box_row in image_rows_groundtruth.iterrows():
59             b_box = [
60                 b_box_row['xmin'],
61                 b_box_row['ymin'],
62                 b_box_row['xmax'],
63                 b_box_row['ymax']
64             ]
65             b_box_list_per_image_groundtruth.append(b_box)
66         # print(b_box_list_per_image_groundtruth)
67
68         # append all bbox info - pred and gt - to their corresponding images and store this in a list
69         examples.append(Detection(
70             image_name, b_box_list_per_image_groundtruth, b_box_list_per_image_pred))

```

```

1  else:
2      # images_in_pred_not_gt.append(image_name) # test to identify missing images
3
4      # define a dictionary to hold the image name and bbox information of [0, 0, 0, 0] since there was no detection
5      zero_pred_bbox = {'image': image_name,
6                        'image_path': 'NULL',
7                        'xmin': 0,
8                        'ymin': 0,
9                        'xmax': 0,
10                       'ymax': 0,
11                       'label': 'NULL',
12                       'confidence': 'NULL',
13                       'x_size': 'NULL',
14                       'y_size': 'NULL'}
15
16     # append row (dict) to the prediction dataframe
17     prediction_file_df = prediction_file_df.append(
18         zero_pred_bbox, ignore_index=True)
19
20     image_rows_pred = prediction_file_df[prediction_file_df['image'] == image_name]
21     # print(image_rows_pred)
22     image_rows_groundtruth = ground_truth_file_df[ground_truth_file_df['image'] == image_name]
23
24     b_box_list_per_image_pred = []
25     b_box_list_per_image_groundtruth = []
26
27     # identify and store the bbox info for all the detected objects in an image
28     for index, b_box_row in image_rows_pred.iterrows():
29         b_box = [
30             b_box_row['xmin'],
31             b_box_row['ymin'],
32             b_box_row['xmax'],
33             b_box_row['ymax']
34         ]
35         b_box_list_per_image_pred.append(b_box)
36     # print(b_box_list_per_image_pred)
37
38     # identify and store the bbox info for all labelled objects in an image
39     for index, b_box_row in image_rows_groundtruth.iterrows():
40         b_box = [
41             b_box_row['xmin'],
42             b_box_row['ymin'],
43             b_box_row['xmax'],
44             b_box_row['ymax']
45         ]
46         b_box_list_per_image_groundtruth.append(b_box)
47     # print(b_box_list_per_image_groundtruth)
48
49     # append all bbox info - pred and gt - to their corresponding images and store this in a list
50     examples.append(Detection(
51         image_name, b_box_list_per_image_groundtruth, b_box_list_per_image_pred))
52
53     """declaring an empty dictionary to hold the values of the image path and the intersection over union list
54     values which contains the gt and pred box information"""
55     IoU_calc_summary = []
56
57     # loop over the example detections
58     for detection in examples:
59
60         # compute the intersection over union
61         intersection_over_union_list = bb_intersection_over_union(
62             detection.gt, detection.pred)
63
64         IoU_calc_summary.append(
65             {'image': detection.image_path, 'gt_and_pred_info': intersection_over_union_list})
66         #IoU_calc_summary.append( detection.image_path, intersection_over_union_list)
67
68     # perform a trial and error on different threshold values
69     while (IoU_threshold_start <= IoU_threshold_stop):
70         threshold_labels_list.append({'threshold': format(IoU_threshold_start, ".1f"), 'label_values': [
71             get_labels(IoU_threshold_start, IoU_calc_summary)]})
72         #print('threshold:', IoU_threshold_start, ' values:', get_labels(IoU_threshold_start, IoU_calc_summary))
73         IoU_threshold_start = IoU_threshold_start + step_value
74
75     # call the calculate_precision_recall_F1score function and print what it returns
76     print(calculate_precision_recall_F1score(threshold_labels_list))
77

```

Appendix C. Training and Validation Losses

The following are training and validation results on dataset 1 and dataset 2. They show the initial training and validation loss values at the beginning of training and at the end of training, as well as the epoch at which early stopping is introduced. The epoch being considered, and the corresponding loss values are highlighted in red. The point at which early stopping is introduced is highlighted in green.

1. Training and validation results on dataset 1

Train on 99 samples, val on 11 samples, with batch size 32.

```
Epoch 1/51
2021-06-01 13:12:57.532237: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] lay
2021-06-01 13:12:57.554327: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] rer
2021-06-01 13:12:57.765007: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] rer
2021-06-01 13:12:58.174626: I tensorflow/stream_executor/platform/default/dso_loader.cc:48]
2021-06-01 13:13:02.715703: I tensorflow/stream_executor/platform/default/dso_loader.cc:48]
1/3 [=====>.....] - ETA: 0s - loss: 9855.21482021-06-01 13:13:08.195471:
2021-06-01 13:13:09.759420: I tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1513] C
2021-06-01 13:13:09.790991: I tensorflow/core/profiler/internal/gpu/device_tracer.cc:223] (
3/3 [=====] - ETA: 0s - loss: 8956.19822021-06-01 13:13:23.303198:
2021-06-01 13:13:23.319850: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] rer
2021-06-01 13:13:23.482090: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] rer
3/3 [=====] - 20s 7s/step - loss: 8956.1982 - val_loss: 7416.4028
Epoch 2/51
3/3 [=====] - 20s 7s/step - loss: 6404.1538 - val_loss: 5216.7432
Epoch 3/51
3/3 [=====] - 20s 7s/step - loss: 4601.8955 - val_loss: 3643.4924
Epoch 4/51
3/3 [=====] - 19s 6s/step - loss: 3248.5750 - val_loss: 2645.0276
```

```
Epoch 00083: ReduceLROnPlateau reducing learning rate to 9.99999939225292e-10.
24/24 [=====] - 24s 1s/step - loss: 19.3424 - val_loss: 20.5158
Epoch 84/102
24/24 [=====] - 24s 1s/step - loss: 21.1110 - val_loss: 21.3028
Epoch 85/102
24/24 [=====] - 25s 1s/step - loss: 20.5472 - val_loss: 18.9072
Epoch 86/102
24/24 [=====] - ETA: 0s - loss: 21.4068
Epoch 00086: ReduceLROnPlateau reducing learning rate to 9.999999717180686e-11.
24/24 [=====] - 24s 1s/step - loss: 21.4068 - val_loss: 20.7945
Epoch 87/102
24/24 [=====] - 24s 1s/step - loss: 20.6424 - val_loss: 22.9669
Epoch 00087: early stopping
```

2. Training and validation results on dataset 2

Train on 279 samples, val on 31 samples, with batch size 32.

Epoch 1/51

```
2021-06-11 15:16:31.561791: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] lay
2021-06-11 15:16:31.579544: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] ren
2021-06-11 15:16:31.750725: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] ren
2021-06-11 15:16:32.097448: I tensorflow/stream_executor/platform/default/dso_loader.cc:48]
2021-06-11 15:16:37.779327: I tensorflow/stream_executor/platform/default/dso_loader.cc:48]
1/8 [==>.....] - ETA: 0s - loss: 8276.31352021-06-11 15:16:40.758205:
2021-06-11 15:16:41.365783: I tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1513] Cl
2021-06-11 15:16:41.393152: I tensorflow/core/profiler/internal/gpu/device_tracer.cc:223] (
8/8 [=====] - ETA: 0s - loss: 5657.15672021-06-11 15:17:10.921142:
2021-06-11 15:17:10.936927: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] ren
2021-06-11 15:17:11.064278: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:581] ren
8/8 [=====] - 34s 4s/step - loss: 5657.1567 - val_loss: 3264.6936
Epoch 2/51
8/8 [=====] - 34s 4s/step - loss: 2223.1108 - val_loss: 1328.4005
Epoch 3/51
8/8 [=====] - 34s 4s/step - loss: 968.1423 - val_loss: 642.4367
Epoch 4/51
8/8 [=====] - 34s 4s/step - loss: 531.6675 - val_loss: 402.0786
```

Epoch 00071: ReduceLROnPlateau reducing learning rate to 9.999999974752428e-08.

69/69 [=====] - 41s 590ms/step - loss: 19.1370 - val_loss: 17.9515

Epoch 72/102

69/69 [=====] - 40s 583ms/step - loss: 19.0890 - val_loss: 18.6497

Epoch 73/102

69/69 [=====] - 40s 585ms/step - loss: 19.9294 - val_loss: 21.2099

Epoch 74/102

69/69 [=====] - ETA: 0s - loss: 19.0034

Epoch 00074: ReduceLROnPlateau reducing learning rate to 1.0000000116860975e-08.

69/69 [=====] - 40s 586ms/step - loss: 19.0034 - val_loss: 19.3762

Epoch 75/102

69/69 [=====] - 41s 587ms/step - loss: 19.1792 - val_loss: 18.5987

Epoch 00075: early stopping