A STUDY OF REAL TIME SEARCH IN FLOOD SCENES FROM UAV VIDEOS USING DEEP LEARNING TECHNIQUES

by

Gagandeep Singh Khanuja

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

December 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Baijian Yang, Chair

Department of Computer and Information Technology

Dr. John Springer

Department of Computer and Information Technology

Dr. Byung-Cheol Min

Department of Computer and Information Technology

Approved by:

Dr. Eric T. Matson

Head of the Graduate Program

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Yang, who guided me through the entire process of research and thesis. I would like to thank him for motivating and constantly encouraging me throughout my journey as a graduate student at Purdue University.

I would like to thank my committee members Professor Min and Professor Springer for extending their support and giving me valuable feedback and suggestions throughout my research.

TABLE OF CONTENTS

LIST OF TABLES 7
LIST OF FIGURES 8
LIST OF ABBREVIATIONS 10
GLOSSARY 11
ABSTRACT 12
CHAPTER 1. INTRODUCTION 13
1.1 Floods 13
1.2 Deep Learning 14
1.3 Problem Statement 15
1.4 Scope 16
1.5 Significance 17
1.6 Research Question 18
1.7 Assumptions 18
1.8 Limitations 18
1.9 Delimitations 19
1.10 Summary 20
CHAPTER 2. REVIEW OF LITERATURE 21
2.1 Background 21
2.2 Humanitarian search operations 23
2.2.1 Fire and hazardous materials 24
2.2.2 Confined workspace 25
2.3 Search operations with the assistance of ground robots 25
2.4 Search operations using UAVs (Drones) 26
2.5 Machine learning for search operations 27
2.6 Deep Learning in the context of machine learning 29
2.6.1 Neural Network 30
2.6.1.1 Structure of a neural network 31
2.6.1.2 Cost Function 32

2.6.2 Convolutional Neural Networks 33
2.7 Mask R-CNN 34
2.7.1 Faster R-CNN 35
2.7.2 RoI Align Layer 36
2.8 Single Shot Multi-Box Detector 300 37
2.9 You Only Look Once (YOLOv3) 39
2.9.1 Network Design of YOLOv3 39
2.10 Summary 43
CHAPTER 3. METHODOLOGY 44
3.1 Research Framework 44
3.2 General Methodology 44
3.3 Modifications to YOLOv3 46
3.4 Object detection using pre-trained models 47
3.4.1 ResNet-101 for Mask R-CNN 48
3.4.2 Darknet-53 for YOLOv3 49
3.5 Photogrammetry - Mapping detected objects from algorithms to GPS location 50
3.5.1 Ground Sample Distance 50
3.5.2 Haversine Formula 52
3.5.3 Elevation from the ground 53
3.6 Assumptions made in the research study 53
3.7 Research Type 54
3.8 Evaluation Criteria 54
3.8.1 Inference time 55
3.8.2 Performance Metrics 55
3.9 Experimental setup of computational resources on google colaboratory . 55
3.10 Summary 56
CHAPTER 4. EXPERIMENT RESULTS 57
4.1 Data collection and preparation 57
4.2 Annotation of objects using COCO (Common Objects in Context) 57
4.3 Implementation details and results 58

4.3.1 Pre-trained Mask-RCNN model on 80 classes 58

4.4 Pre-trained SSD300 model on 80 classes 63

4.5 You Only Look Once (YOLOv3) 68

4.5.0.1 Pre-trained YOLOv3 on videos of resolution 1080p . . 73

4.5.0.2 Pre-trained YOLOv3 on videos of resolution 720p . . 75

4.5.0.3 Pre-trained YOLOv3 on videos of resolution 360p . . 77

4.6 Modified trained YOLOv3 model on 2 classes 79

4.6.0.1 Modified trained YOLOv3 on videos of resolution 1080p 83

4.7 Results and Analysis 85

4.8 Summary 87

CHAPTER 5. CONCLUSION AND FUTURE PLAN 88

5.1 Conclusion 88

5.2 Future Plan 89

REFERENCES 90

LIST OF TABLES

3.1 Hyperparameter values for YOLOv3 47
3.2 Comparisons of underlying backbone used in research from Joseph Redmon (2017a) 50
3.3 Flight log of DJI Drone 54
4.1 Pre-trained Mask-RCNN confusion matrix - Images 62
4.2 Pre-trained Mask-RCNN confusion matrix - Videos 62
4.3 Pre-trained SSD300 Confusion Matrix - Images 67
4.4 Pre-trained SSD300 Confusion Matrix - Videos 67
4.5 Pre-trained YOLOv3 Confusion Matrix - Images 73
4.6 Pre-trained YOLOv3 Confusion Matrix - Videos of resolution 1080p 74
4.7 Pre-trained YOLOv3 Confusion Matrix - Videos of resolution 720p 76
4.8 Pre-trained YOLOv3 Confusion Matrix - Videos of resolution 360p 78
4.9 Trained YOLOv3 Confusion Matrix - Images 83
4.10 Trained YOLOv3 Confusion Matrix - Videos of resolution 1080p 84
4.11 Comparison of results 86

LIST OF FIGURES

1.1 Traditional Machine Learning Vs Deep Learning from Wikipedia, the free encyclopedia (2013). 15
2.1 Performance graphs of machine learning methods from Armin Wasicek (2018) 29
2.2 Structure of Perceptron from Jose Ariza (2018) 30
2.3 A Fully Connected Network from Andrey Kurenkov (2015) 31
2.4 2D representation of a cost function from Conor McDonald (2017) 32
2.5 Illustration of CNN from Adit Deshpande (2016) 33
2.6 Architecture of Mask R-CNN from Chanuk Lim (2017) 34
2.7 Comparison of test time speed of R-CNN based algorithms from Fei-Fei Li (2017) 35
2.8 Architecture of Faster R-CNN for object detection from Shaoqing Ren (2016) 36
2.9 Accurate mapping of original image to feature maps by using RoI align layer from Lilian Weng (2017) 37
2.10 Architecture of Single Shot Multi-Box Detector from Wei Liu (2016) 38
2.11 Architecture of YOLO from Joseph Redmon (2016) 40
2.12 Inference time comparison of YOLOv3 with other models from Joseph Redmon (2017b) 42
3.1 Workflow architecture for object detection using pre-trained and modified trained YOLOv3 model 45
3.2 A visual illustration of object detection using modified trained YOLOv3 . . 45
3.3 A visual illustration of object detection using modified trained YOLOv3 . . 46
3.4 Building Block for ResNet-101 from Kaiming He (2015) 49
3.5 Concept of Ground Sample Distance from Drone Blocks (2013) 51
4.1 GPS estimation of objects using Mask R-CNN model 59
4.2 Validation of distance obtained in Figure 4.2 using Haversine Formula . . . 60
4.3 Inference time for pre-trained Mask R-CNN model 61
4.4 Precision Recall curve for pretrained Mask-RCNN 63
4.5 GPS estimation of objects using pre-trained SSD300 model 64

4.6 Validation of distance obtained in Figure 4.5 using Haversine Formula . . . 65
4.7 Inference time for pre-trained SSD300 model 66
4.8 Precision Recall curve for pretrained SSD300 68
4.9 GPS estimation of objects using pre-trained YOLOv3 model 70
4.10 Validation of distance obtained in Figure 4.8 using Haversine Formula . . . 71
4.11 Inference time for pre-trained YOLOv3 model 72
4.12 Inference time for pre-trained YOLOv3 model of resolution 1080p 73
4.13 Precision Recall curve for pretrained YOLOv3-1080p 74
4.14 Inference time for pre-trained YOLOv3 model of resolution 720p 75
4.15 Precision Recall curve for pretrained YOLOv3-720p 76
4.16 Inference time for pre-trained YOLOv3 model of resolution 360p 77
4.17 Precision Recall curve for pretrained YOLOv3-360p 78
4.18 GPS estimation of objects using trained YOLOv3 model 80
4.19 Validation of distance obtained in Figure 4.12 using Haversine Formula . . . 81
4.20 Inference time for trained YOLOv3 model 82
4.21 Inference time for trained YOLOv3 model of resolution 1080p 83
4.22 Precision Recall curve for modified YOLOv3-1080p 84
4.23 Accuracy of Modified YOLOv3-1080p 85

LIST OF ABBREVIATIONS

abbr	abbreviation
UAV	Unmanned Aerial Vehicle
SAR	Search and Rescue
MASK R-CNN	Mask Region Convolutional Neural Network
SSD	Single Shot Multibox Detector
YOLO	You Only Look Once
ANN	Artificial Neural Network
GPU	Graphical Processing Unit
CPU	Central Processing Unit
CNN	Convolutional Neural Network
GIS	Geographic Information System
SVM	Support Vector Machine
RoI	Region of Interest
FCN	Fully Connected Layer
VGG	Visual Geometry Group

GLOSSARY

GPU – Graphical Processing Unit. A single/multi chip processor with multiple, rendering cores that can process millions of instructions per second.

Machine Learning - It is the ability of a machine to learn from the data and is used for the purpose of decision making.

Deep Learning - It is a study that involves working on sequential data or finding patterns from large amount of data for use in decision making.

UAVs - An aerial vehicle that can autonomously fly and can be piloted remotely with minimum human intervention.

ABSTRACT

Following a natural disaster, one of the most important facet that influence a persons chances of survival/being found out is the time with which they are rescued. Traditional means of search operations involving dogs, ground robots, humanitarian intervention; are time intensive and can be a major bottleneck in search operations. The main aim of these operations is to rescue victims without critical delay in the shortest time possible which can be realized in real-time by using UAVs. With advancements in computational devices and the ability to learn from complex data, deep learning can be leveraged in real time environment for purpose of search and rescue operations. This research aims to solve the traditional means of search operation using the concept of deep learning for real time object detection and Photogrammetry for precise geo-location mapping of the objects(person,car) in real time. In order to do so, various pre-trained algorithms like Mask-RCNN, SSD300, YOLOv3 and trained algorithms like YOLOv3 have been deployed with their results compared with means of addressing the search operation in real time.

CHAPTER 1. INTRODUCTION

This chapter gives a brief introduction to the research study conducted. First, we present a background on floods, followed by a brief description of the problem statement, the scope of the study and the significance of it.

1.1 Floods

Hydrological natural disasters such as floods, flash floods, debris flow are a typical example of fast-developing disaster. Since it is a fast process, the disaster management team gets little time and information to assess the seriousness and scale of it. Cities that are based in the low-lying plains get seriously affected and can leave people trapped thereby reducing their chances of survival with the course of time. Therefore, the areas that get inundated by floods need to be mapped in such a way so that precise search operations can be maximized. In these situations, manned aerials or unmanned aerial vehicles (drones) can be deployed in segmented areas for precise search operations.

Post-disaster flood activity can be assisted by using unmanned aerial vehicles and informing the responsible disaster management authority of any unusual activity so that they can react in time and carry out search operations. In case of unaccepted damage to dams during a natural disaster, humans get trapped in the flooded areas due to rapid runoff of water from the dam, and thus evacuation by traditional search operations is not recommended as the affected areas are oversized, hence managing these disasters by aerial imagery is the most suitable thing to do.

In the case of runoff or unwanted inundation of the land by flood, drones can be used in many ways. These devices can be used to predict the floods, and capture images of the location from where the people need to be evacuated and what buildings are at a risk. The essence of this application is to use unmanned aerial vehicles that can assist the disaster management team with precise search operations using deep learning.

1.2 Deep Learning

Object recognition or object classification means identifying a feature or an object in an image. This approach has been a topic of research in the field of science and engineering. The algorithms based on object detection rely on learning, matching or finding patterns in an image based on various feature-based techniques. Machine learning is the process of learning from data and making informed decisions with minimum human intervention. Deep learning algorithms are used to translate complex hierarchical concepts into simpler ones.

Deep learning algorithms can perform tasks like planning, perception, control, and localization and can learn from complex data that has been acquired from the environment in real-time which makes it suitable for image processing and similar applications.

Traditional methods of machine learning as shown in Figure 1.1 involved feature extraction/engineering methods which were manually driven by statistics and computational mathematics (complex design) and did not perform well in terms of accuracy because feature extraction of unstructured data was very difficult. With the introduction of better computational devices in the market, new methods of extracting features and classifying them in a single shot were made possible with the aid of deep learning. It was found out, with an increase in the amount of data fed in the machines the performance of traditional machine learning algorithms reduced but the performance of deep learning algorithms was maximized.

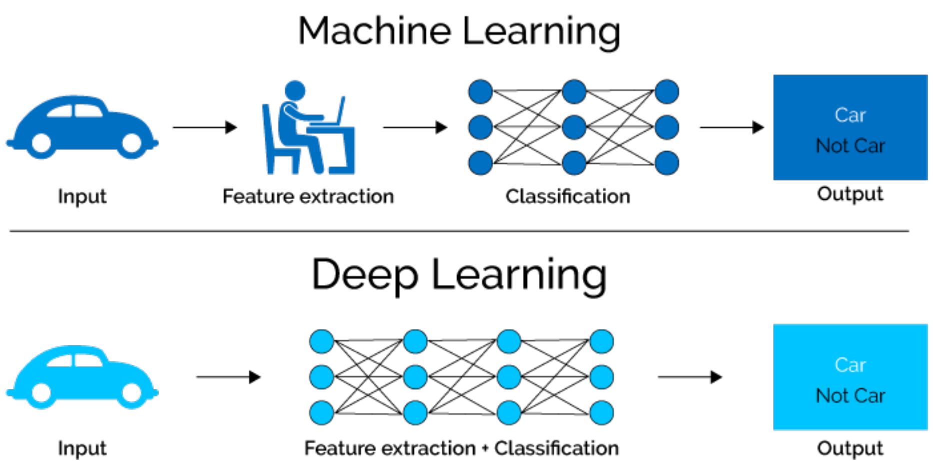


Figure 1.1. Traditional Machine Learning Vs Deep Learning from Wikipedia, the free encyclopedia (2013).

1.3 Problem Statement

In case of a flood, one of the most important aspects to be considered in search operations is the survival time of the victims. In the research conducted by Haegeli, Falk, Brugger, Etter, and Boyd (2011) it was found out that the survival time was even lower than previously predicted and hence precise search operations were necessary. With advancements in technology and hardware, a more promising approach is the use of computer vision and deep learning for precise search operations, especially during a flood. In this research, we have deployed various pre-trained algorithms like YOLOv3, SSD and Mask-RCNN and trained algorithm like YOLOv3 and compared the results of all the algorithms to find the best algorithm that can be deployed for search operations in real-time.

1.4 Scope

The number of natural disasters (hydrological, geophysical, meteorological or climatological) has shown an upward trend in the last five decades. The annual disaster statistical review in its report describes that there have been 345 naturally triggered disasters in 2016 alone and have claimed a countless number of lives which is less than the average number of disasters (376.4) observed during the decade from 2005 to 2015 (Guha-Sapir, Below, & Hoyois, 2016). The average number of deaths (8,733) caused due to natural disasters in 2016 was the second-lowest reported since 2006, where the average number of annual deaths stood at 69,827. Since 2006, a record number of people (565 million) have been affected by the natural disasters and an estimated economic annual damage of USD 100 billion has been reported (Guha-Sapir et al., 2016). Looking at the upward movement in the number of floods and the loss of lives, one of the most important aspects to be considered is the ways to rescue these victims during the disasters.

Disaster search operation can be facilitated using UAV such as drones, which have various benefits viz. small size, low cost of operation, exposure to dangerous environments and can offer a high rate of success with no loss to human lives. These benefits offer an edge over remote sensing techniques like the use of satellites. Unmanned aerial vehicles in the past have been equipped with optical cameras that can be operated in the visible light of spectrum or the infrared light spectrum (in case of low light condition) to capture different spectra of events of interest. The videos and images captured by these sensors can then be processed remotely or locally which is often cumbersome and time intensive. In both cases of local and remote processing, automatic identification of events of interest (i.e. potential disasters) are important for search operations during the disaster while it is happening, taking immediate life-saving actions, either by governmental organizations or humanitarian organizations (Kamilaris & Prenafeta-Boldú, 2018). A modern and more promising technique for object detection is the use of convolutional neural networks for precise search operations. It has been found that deep learning methodology seems to have better performance in image recognition tasks (Guo et al., 2016), in comparison to common techniques used for analyzing images such as SVMs and ANN (Saxena &

Armstrong, 2014). My research study comprises of utilization of these unmanned aerial vehicles equipped with optical sensors for deep learning models to identify victims (in a flood) for the following reasons 1) the identification is fast (i.e. in seconds after the model has learned the problem), 2) these models are flexible (i.e. many events can be identified at once) and 3) scalable (i.e. learning process is continuous and adaptable) in real-time.

1.5 Significance

One of the prime elements of a disaster response is situational awareness – knowing the people who have been affected by the disaster, when, how (the type of disaster) and where (the location). Once a disaster subdues, the humanitarian search organizations carry out assessments to monitor the extent of disaster before the rescue operations can be carried out. One of the key factors to be considered here is the amount of time it takes the rescue teams to assess these disasters which can be utilized to speed the life-saving efforts and help them respond quickly (Al-Khudhairy, 2010). In 2013, one of the most powerful typhoons recorded in human history ("Typhoon Haiyan") made landfall in the Philippines where thousands of humans lost their lives with no proper devices at the disposal for the government (Calantropio, Chiabrando, Sammartano, Spanò, & Losè, 2018). There are various types of sensors that can be put on UAVs such as the thermal sensor or the optical sensors as these sensors can capture high-resolution images and videos in visible light of spectrum and even in the infrared spectrum. Since they capture high-resolution images, they have a tendency of capturing terabytes of data in a single day which is not possible to assess manually and can be a difficult and time-consuming task (Ezequiel et al., 2014). During a disaster in Haiti, an unmanned aerial vehicle (UAV) mission took the analysts more than 25 days to capture an image dataset of 5600 images (Corbane et al., 2010). In Namibia, it took more than 400 hours, approximately 18 days for a team of four analysts to search for wildlife and fauna from an image data set of more than 15,000 images (Fornace, Drakeley, William, Espino, & Cox, 2014). This research study aims to make use of deep learning techniques to make improved decisions for search operations and provide aid to the people stuck in flood real time.

1.6 Research Question

The goal of this research is object detection for search operations in a flood in real-time. To identify objects in real-time the detection speed should be more than 20 frames per second or better and it is one of the prime factors of choosing the model for object detection. This research answers the following question:

- Does the proposed modified version of trained YOLOv3 perform faster (inference time) and accurately (performance metrics) than the existing version of pre-trained models of Mask-RCNN, SSD300, and YOLOv3 to classify specific objects in real-time (20 fps) in flood scenes for search operations?

1.7 Assumptions

- The I/O object detection used in the simulation experiment can reflect the real-world object detection for search operations.
- Specific computer hardware and software used to implement deep learning models do not alter the generalization of the results unless specifically mentioned.
- The execution environment would remain constant and work with equal reliability and efficiency for all the algorithms considered for comparison.
- The flood data set is representative of the future natural disaster flood data set in its data format and size.

1.8 Limitations

- The research study will be tested on systems with GPUs. The models will not be tested on systems with CPUs due to computational constraints.

- The accuracy grade of the performance measurement of various models is limited by precision, recall.
- The data set used in research is limited to hydrological disasters like flash floods, floods, and debris flow.

1.9 Delimitations

- Even though there exist many deep learning models, this research focuses only on three deep learning models that can be deployed in real-time (Mask-RCNN, SSD300, YOLOv3). It does not consider other deep learning algorithms for comparison.
- For higher classification rate on the server end, it is expected that the height and velocity of the drone at which it is being flown should be as minimum as possible.
- The image pre-processing time will not be used as a metric to evaluate the performance of the model in real-time.

1.10 Summary

This chapter provides a brief introduction to the research conducted. It also underlined the scope of our research study in floods, the significance of having a UAV system capable of on the go object detection, the research question along with assumptions, limitations, delimitations of the research study.

CHAPTER 2. REVIEW OF LITERATURE

In this section, we will first discuss the need of search operations in a natural disaster (floods) then review the traditional methods being deployed in search operations like the use of humans, ground robots, thermal imagery and various methods of machine learning: supervised and unsupervised. Later, we will review some of the most influential approaches of deep learning used for object detection, such as the CNN based approach depicted in our study. Here, the architecture, the advantages and the disadvantages of each method will be discussed.

2.1 Background

Natural disasters are catastrophic events that can jeopardize the well-being of people who are incapable of combating the affliction arising from them thereby causing social environment disruption and negative economic impacts (Blaikie, Cannon, Davis, & Wisner, 2014). Natural disasters can be classified into various types such as hydrological (debris flow from runoff, flash-floods due to human activities and floods), geophysical (tsunami, landslide, volcanic eruptions, earthquake), meteorological (sandstorm, hurricane, tropical storm and heavy rainfall) and climatological (wildfire, drought and extreme temperature). In the past 30 years, there has been a significant increase in the material loss caused by these disasters which stands at an order of 100-150 percent (Munich, 2015). Few of the most important factors to be considered while assessing a natural disaster are the availability of resources in disaster areas, the unpredictability and environmental changes (Celik & Corbacioglu, 2010). Unpredictability in the underlying context implies, the severe impact on property and people during these disasters which cannot be forecasted with admissible accuracy (Sutanta, Bishop, & Rajabifard, 2010). Hence, the allocation of adequate resources is not possible in advance. Prediction of damages and movement of people is difficult to forecast with increased dynamic changes in the environment (de Moel et al., 2015).

Following a disaster, a commercial satellite can take close to twenty-four to forty-eight hours to acquire and process the imagery. In the year of 2013, when a powerful typhoon Haiyan made landfall in the Philippines it took more than sixty-four hours for the satellite images of the damage to be made available to the first responders (Ofli et al., 2016). On the contrary, disaster monitoring and rescue operations can be facilitated using UAV which is becoming more and more famous, having various benefits such as small size, low cost of operation, exposure to dangerous environments and high probability of mission success. These and other benefits are explained in related work, either independently (Petrides, Kolios, Kyrkou, Theocharides, & Panayiotou, 2017), (C. Zhang & Kovacs, 2012), or in relation to other remote sensing techniques such as airplanes and satellites (Matese et al., 2015). UAVs have shown the ability to compete with traditional platforms for acquisition (i.e. aircraft and satellite) due to high flexibility offered for operation and high resolution. Since the wake of typhoon Haiyan, several organizations have deployed UAVs and it is expected the number of UAV missions will increase significantly in the years ahead (C. Zhang & Kovacs, 2012).

Preserving human lives is one of the most important aspects when a disaster occurs. In this context, when a disaster strikes, the first 72 hours are the most crucial, which implicates that the search operations must be carried out efficiently and quickly (Wzorek et al., 2006). One of the major challenges during a disaster is lack of situational awareness and communication, forcing the first responder teams to prepare impromptu thereby degrading the efficiency of the search operations. Situational awareness in the underlying context means – knowing who, where, how and when has been affected. Situational awareness can be achieved in the most efficient way through aerial assessment-via UAV networks (Guha-Sapir, Vos, Below, & Ponserre, 2012). With the use of UAVs and computer vision, the first responders can better understand the potential number of people affected by the disaster, the extent of damage caused to the infrastructure, the state of transportation infrastructure and the structures affected by the disaster.

In the past, UAVs have been successfully used in scenarios to detect earthquake-triggered roof holes (S. Li et al., 2015), oil spills and flooding (Luo, Nightingale, Asemota, & Grecos, 2015), and locate victims (Andriluka et al., 2010), etc. A recent report from red cross advocated the use of UAVs to be one of the most powerful technologies for real-time insights in driving search operations. Based on current advancements in the field of UAVs can be used to provide mapping and reconnaissance support, identify survivors stranded in a disaster and redirect the disaster management personnel to reach them in the most optimized way. They are also used for structural assessment (Erdelj & Natalizio, 2016).

In recent times there have been tremendous efforts to forecast and recognize the occurrence of natural disasters to efficiently assess the damage and react in a timely manner by carrying out search operations and restore normalcy. With latest advances in the field of technology, here we describe a vision for leveraging the use of unmanned aerial vehicles along with deep learning technology to amplify the ability of computer vision assisted disaster assessment and response.

2.2 Humanitarian search operations

In the bygone century, natural and man-made disasters have taken the lives of millions of people and caused the destruction of billions of assets around the world. Supernatural events such as the China floods in 1931, the Russian wildfire in 2010 and the Portugal wildfire in 2017 have claimed millions of lives and caused damages worth billions thereby affecting the communities. Man-made disasters such as the civil war, terrorism, engineering hazards, oil spills such as the Adana massacre in 1907, the Ennore oil spill in 2017 have caused even more casualties than those caused by natural disasters. In humanitarian search operations, structural damage is a common sight. They are hindered by a multitude of confined spaces and dangerous conditions which hinder the entry to the cavity that may have survivors stuck beneath them. In such cases, it is strictly prohibited to use heavy machinery for search operation as they risk the life of the victims and the rescuers (Ezequiel et al., 2014). One of the other factors that hamper these operations is the affluence of sensory distractors like false odor, random noise and scattered false readings.

In the past, cameras, advanced sensors, heavy lift equipment, search dogs have been used for humanitarian search operations to find victims. While the cameras with the help of probes could only be used in depths no more than 4 to 6 meters, the dogs even though capable of locating victims buried underground were unable to provide the precise location of the burial of the victim (Rehor, 2007). This has made the humanitarian search operation very difficult as various constraints need to be imposed on the rescue teams. Some of the constraints that need to be addressed have been discussed in the following paragraphs.

2.2.1 Fire and hazardous materials

The pacific rim is one of the most persistent earthquake-prone activity regions that can lead to collateral damage at the disaster site. Disrupted public utilities and emergency services by the personnel can lead to toxic contamination, fires for prolonged hours. With no adequate supply of electricity and water, even the superbly trained firefighters and the search personnel must hold up until the collateral warnings have been taken care of to gain entry to scathing flood struck areas. Even when suitable resources and shelter is at disposal, sensory distress caused by heavy gloves, safety gear, and thick masks can hinder the team's rescue process (Blitch, 1996).

2.2.2 Confined workspace

Void spaces are often created during a structural collapse and they vary in volume and size as in the collapse of Maryland freeway in 1987. This generally creates voids anywhere between 2 to 6 ft in height. Localizing and finding these voids is one of the most important tasks of the search operation in confined spaces (Blitch, 1996; Mills & Navarro, 1995).

2.3 Search operations with the assistance of ground robots

Robot-assisted search operations are well suited to address the limitations of humanitarian search operations. To expedite the search process, several robots can be deployed in response to a disaster. They can navigate through confined spaces and deteriorating unstable structures where dogs cannot make an access and can be used to check the survivor's body temperature and heart rate with the help of probes mounted on it (Snyder, 2001).

Robots can also be used to assess the physical strength of the structure damaged by the flood with minimal intervention of humans which otherwise can take up to three to four critical early hours by engineers (Scassellati, Admoni, & Matarić, 2012). They can also carry radiation, carbon monoxide, pH level, temperature, oxygen sensors on board and conduct hazardous material and atmospheric reading and warn the rescue personnel based on their analysis.

In scientific literature, the use of robots for search operations has been discussed since the 1980s, but none of the robots had been fielded until 2001. With an exponential increase in the speed, miniaturization of sensors and improved processing speed of the microcontrollers, robots have been used in the catastrophic areas since 2001. After the bombing of the Oklahoma City in 1995 (Murphy, 2004), the first real research on search operations by the robots was initiated. At the bombing response, the robots were not used but the suggestions and the response had been taken from the robots. After the bombing of the World Trade Center in 2001, the first documented usage of robots for search operations was recorded. While carrying out these search operations, various robots of different shapes, sizes and capacities were deployed (Snyder, 2001). Here, the primary reason for using these robots was to identify potential hazards and search victims stuck in the debris.

2.4 Search operations using UAVs (Drones)

UAVs have been used in the past to aid and find humans who were critically injured, lost in rivers, mountains, lakes, deserts, and remote setting. Search operations using human aid has been challenging as it either needed search personnel with specialized training or it consumed thousands of man-hours and dollars per year. Humanitarian search operation can be a very slow process if the area of collateral damage is large along with challenging terrain. The use of UAVs can be used to improve the speed at which the victims can be found and the probability of the missing person (Giitsidis, Karakasis, Gasteratos, & Sirakoulis, 2015).

There has been extensive research on human factors using semi-autonomous UAVs. One of the key and foremost aspects that should be considered when using these UAVs for search operations is people needed to operate it. In the work cited by Murphy (2004) it was suggested that a third person could be sometimes useful to monitor the behavior of the person flying the UAV and the sensor operator to facilitate higher situational awareness. Important research was conducted by Cummings (2004); Olsen Jr and Wood (2004) which analyzed the number of humans that could be used to manage the unmanned aerial vehicles. It was suggested that the outreach was limited as information-rich videos from multiple UAVs would be difficult to monitor for search operations.

To address the problem of using semi-autonomous unmanned aerial vehicles; an enhanced autonomous information acquisition and presentation using the unmanned aerial vehicle was suggested by Murphy (2004). The goal of this research was to focus on different interface designs and hardware to cut down on the number of humans needed to perform the task, which included algorithms based on path, altitude and attitude stabilization at low lying plains and path generation.

In more recent work, techniques for search operations using thermal imagery have been suggested. One of the examples uses an infrared camera to map a plan to find a correlation between humans extracted from the infrared camera and a color camera in a static environment using image extraction techniques (Rudol & Doherty, 2008).Kang, Gajera, Cohen, and Medioni (2004) in their research suggested a technique for detecting and tracking moving objects by using a probabilistic framework. Their work was tested on footage collected by an unmanned aerial vehicle, but the authors did not discuss its usability onboard nor the computational requirements needed to identify the humans in a simulated search operation. Keck and Davis (2008) in their research discuss an approach of using a fast screening template using ADA-boosted ensemble classifier to locate a potential person location.

2.5 Machine learning for search operations

Machine Learning is defined as the ability of a machine to learn insights from data and perform tasks thereby reducing human intervention. My research lies in the interplay of assessing the hydrological disasters (flash floods, floods and debris flow) by means of machine intelligence using deep learning.

Machine learning in search operations can be used to prevent life loss and damage to the property when a hydrological disaster strike. Recent advancements in technology have had a major impact on the prediction and forecasting methodologies for search operations. Remotely sensed data over the last two decades has been used to assess the hazards and distinct types of phenomenon (Pradhan, 2010; Pradhan, Hagemann, Tehrany,

& Prechtel, 2014) by use of active and passive sensors. In flood disaster management, Schumann, Neal, Mason, and Bates (2011) showed that mapping was an essential step. In addition to mapping, it was even depicted that geographic information system (GIS) was an important tool to probe the flooding events.

Some of the machine learning developed methods that were used in identification of the flood areas for search operations included quantitative and qualitative techniques such as tree-based models like decision trees (DT) (Tehrany, Pradhan, Mansor, & Ahmad, 2015) , multicriteria evaluation (Cireşan, Meier, & Schmidhuber, 2012) adaptive neuro-fuzzy interface system (Bubeck, Botzen, & Aerts, 2012), classification based models like logistic regression (Pradhan et al., 2014), and hierarchy-based algorithms like analytical hierarchy process (Feng & Wang, 2011). The advantages and disadvantages of the use of statistical machine learning were reviewed by Tehrany et al. (2015). It was found out that these algorithms did not give promising accuracies with an increase in a few predictors as the models were overfitting thus restricting their use for search operations. In South Korea, bivariate probability models were used to map inundated regions, but they had a drawback as it considered the relationship between each independent layer and flood occurrence separately (Feng & Wang, 2011). The use of multivariate logistic regression (Pradhan, 2010) involved finding relationship between the response variable and predictors to predict the area that was inundated by floodwater and then carry out search and rescue operations in a simulated flood area. He noted the use of logistic regression had several advantages: It was able to handle non-linear effects. It was robust as the predictors did not specifically need to have gaussian distribution or have equal variance in the group and the variables could be continuous or discrete. He showed the accuracy of the model by means of area under the area under curve-receiver operating characteristic (AUC-roc) curve and confusion matrix but the impact of classes of each variable was not taken into consideration. Statistical machine learning algorithms like logistic regression and bivariate probability have found much appreciation by the research community for susceptibility mapping of region inundated by flood for search and rescue operations but have major limitations in terms of being deployed in real-time which moved the focus of my research to deep learning.

2.6 Deep Learning in the context of machine learning

Conventional machine learning methods were based on shallow-structured learning architectures and were restricted in the ability to process large amounts of data as well as process data in unstructured form (Chen, Lin, Zhao, Wang, & Gu, 2014) as shown in Figure 2.1. In the past, constructing a machine learning structure or a pattern recognition system needed considerable experience and domain knowledge with the expertise to craft a feature extractor that could transform the data into suitable representation or feature support vector to a group and categorize, detect or classify the pattern in the input. He, Zhang, Ren, and Sun (2016) compared deep learning models and statistical machine learning models on equal footing and by using same data pre-processing and augmentation techniques used with convolutional neural networks on traditional statistical machine learning. In his research, he explored the importance of color information and showed how it made a significant difference by using deep learning methodology. He found out that deep learning had an upper edge over traditional statistical learning. The color information, data augmentation, and pre-processing tasks were precisely measured by using this technique.

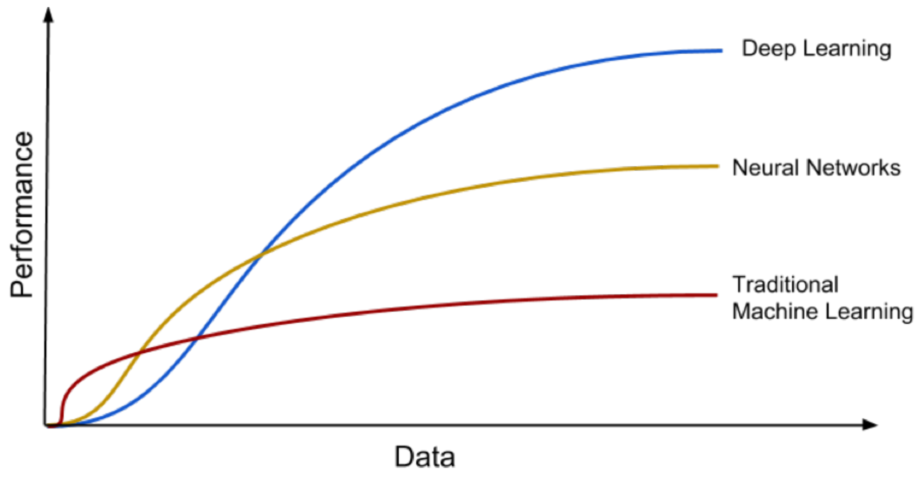


Figure 2.1. Performance graphs of machine learning methods from Armin Wasicek (2018)

Deep learning has been inspired by biological nervous systems and learns from depictions of data with multiple layers of abstraction (H. Li, Zhao, & Wang, 2014). They have been used to discover intricate patterns in large and raw data sets by using the concept of backpropagation to tune the hyperparameters and their representation in each layer.

2.6.1 Neural Network

A neural network comprises of three things just like any other machine learning algorithm, they are the parametric model (shape of the function), the cost function and the optimization approach to fit the structure of the model. The neural networks were proposed in the 1950s and have originated from the perceptron which has one output and more than one input. The structure of a perceptron can be understood from Figure 2.2. The output of the perceptron is a binary value (Jain, Mao, & Mohiuddin, 1996).

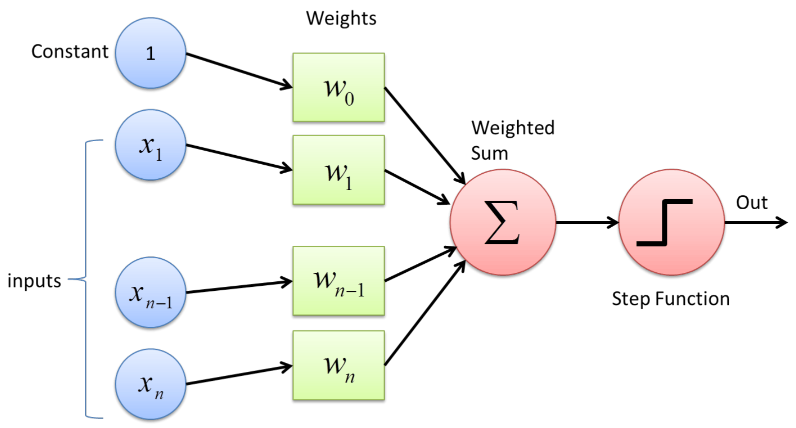


Figure 2.2. Structure of Perceptron from Jose Ariza (2018)

The output of the perceptron will change to 0 or 1 when the summation of the input changes. A small step of change in input won't affect the output. In order to obtain outputs between 0 to 1, activation functions are used. There are prominently four different types of activation functions namely: ReLU, Maxout, Sigmoid, and Softmax. The function is then applied elementwise to the summation of inputs and weights to obtain the output which lies in the range of 0 and 1 (Jain et al., 1996).

2.6.1.1 Structure of a neural network

It is a parametric machine learning algorithm which comprises of multiple layers as depicted in Figure 2.3. It has an input layer; also known as the first layer, an arbitrary number of hidden layers; which are also known as the middle layers, an output layer; also known as the last layer, a set of weight and biases between each layer and an activation function for each hidden layer. The neurons are connected to each other to pass information from one layer to the other layer (G. Zhang, Patuwo, & Hu, 1998).

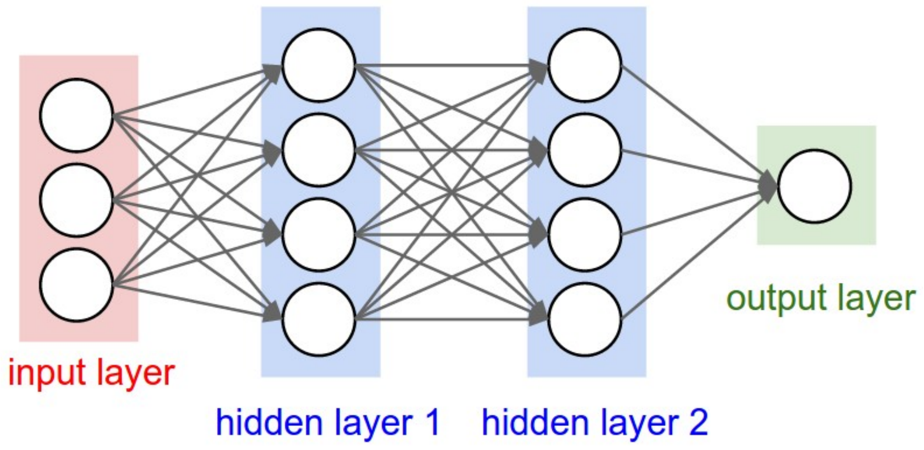


Figure 2.3. A Fully Connected Network from Andrey Kurenkov (2015)

When the network is connected in tandem then the decisions made by the first layer are passed on to the second layer as input. The second layer then can make more smart and improved decisions. The information is then propagated so that the network can make more informed decisions.

<u>2.6.1.2 Cost Function</u>

Neural networks are parametric models where the distribution of the data is defined. Training a neural network involves learning how to perform a task. Every machine learning algorithm learns from its errors between ground truth and predictions and tries to minimize it as much as it can by using the concept of maximized likelihood estimation or ordinary least squares. The prediction error can be computed by using a cost function. In the euclidean space since the default choice of metric is euclidean metric, similarly, in a neural network, the choice for a cost function is information gain which works on the principle of maximum likelihood.

During the training, the neural network tries to minimize the error as much as it can and once the error cannot be reduced any further the training process is said to be completed. There exist a different kind of cost functions such as quadratic functions like RMSE (root mean square error) or MSE (mean square error). Figure 2.4 gives you the visualization of the cost function. The black dot in the bowl-shaped convex function, which is also known as prediction error that tries to attain a minimum value by moving to the bottom of the convex function (G. Zhang et al., 1998).

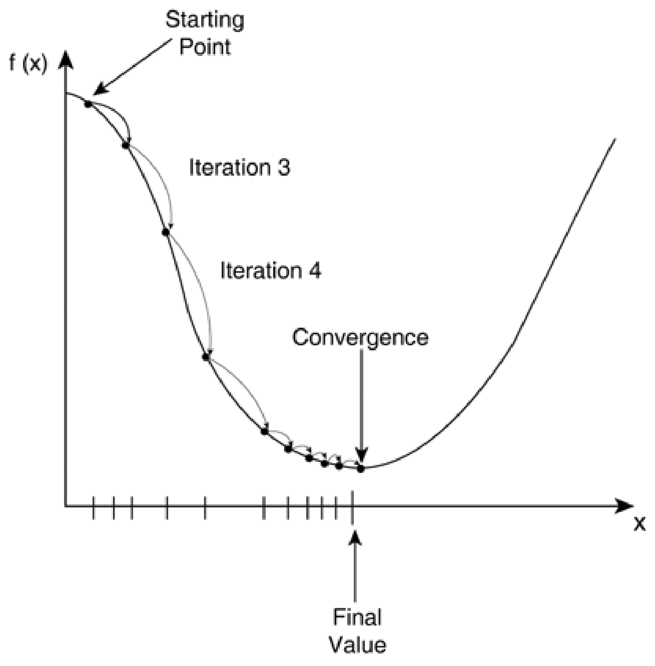


Figure 2.4. 2D representation of a cost function from Conor McDonald (2017)

2.6.2 Convolutional Neural Networks

In the early 1990s, the use of neural networks for handwritten zip code identification was first documented. The convolutional neural networks were first introduced in 1998 but were only showed to be an effective tool for image detection and recognition in 2012 at the image net computer vision competition.

CNN's have brought about a breakthrough in videos and images whereas RNNs have been used for sequential data such as speech and text (Schmidhuber, 2015). For digit recognition, LeCun, Bottou, Bengio, Haffner, et al. (1998) made use of convolutional neural networks. With improved computational speed and faster processing capability it has found applications in the field of CV and ML as they have been said to achieve the start of the art performance for various tasks (Tompson, Goroshin, Jain, LeCun, & Bregler, 2015). Figure 2.5 shows an illustration of a convolutional neural network.

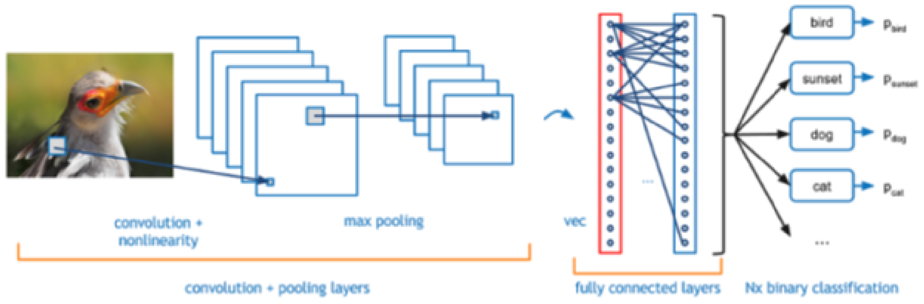


Figure 2.5. Illustration of CNN from Adit Deshpande (2016)

Krizhevsky, Sutskever, and Hinton (2012) first used the deep CNN in the image detection task in 2012. They attained a successful test error of 15.3 percent on training the model on more than one million images with over 1000 classes. Since then there has been extensive research to reduce the test error rate of the convolutional neural networks and recent work got better results by reducing the test error rate to 13.24 percent by training the model to continuously locate and classify objects (Ibrahim, Muralidharan, Deng, Vahdat, & Mori, 2016). (Howard, 2013) in their work found out that object detection could also benefit from convolutional neural networks apart from image classification tasks. Researchers in 2014 shifted their focus to multiple object detection tasks. In such cases, more than one object in the image is to be detected and the system tries to identify the object's location and class. Since then, numerous regions based on convolutional neural networks approaches came out. In recent times, faster object detection and classification algorithms like Mask R-CNN, SSD, YOLOv3 have been crafted by researchers to provide object detection solutions in real-time.

2.7 Mask R-CNN

The underlying structure that is used to build a Mask R-CNN is Faster R-CNN. The architecture leverages two networks in parallel for object detection, one: region proposal network and two: binary mask classifier. In this type of algorithm, the conventional RoI pooling layer has been replaced with more accurate RoI align module. Additionally, a branch from the module is fed in parallel to the two convolutional layers. These layers are responsible for classification and localization. The output from these layers is the masked segmented image, with bounding box surrounding it (He, Gkioxari, Dollár, & Girshick, 2017). The architecture can be understood from Figure 2.6.

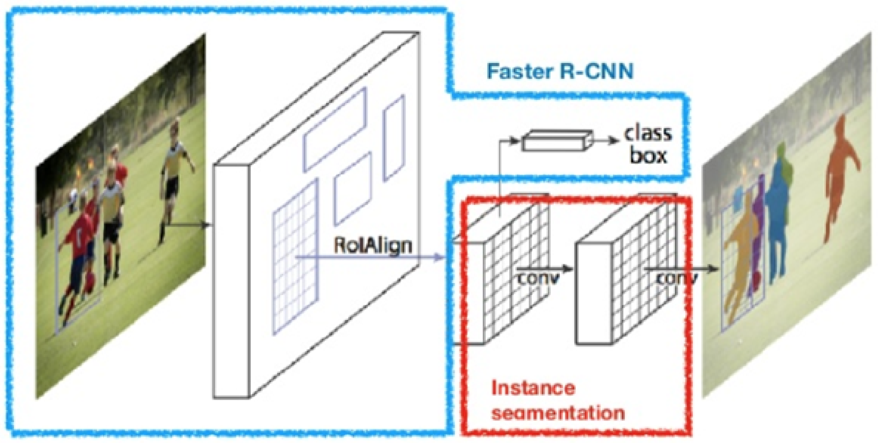


Figure 2.6. Architecture of Mask R-CNN from Chanuk Lim (2017)

The network comprises of Faster R-CNN and a binary mask classifier.

2.7.1 Faster R-CNN

The RCNN and Fast RCNN use selective search method to extract regions of interest that form an object. An object has four regions that the selective search approach intends to identify: textures, scalers, enclosures, and colors. It takes the input image and sub-segments it to generate multiple regions. Large regions are formed by combining similar regions (based on shape, compatibility, size, and texture). Finally, these regions generate RoI. The method is a slow and time-consuming process and cannot be used in real-time on large real-time datasets. A promising approach for on the go object detection involves the use of region proposal networks which generates object proposals based on feature maps Ren, He, Girshick, and Sun (2015). Figure 2.7 illustration shows that Faster R-CNN can be used for on the go object detection as it is much faster than its predecessor algorithms based on region-based proposal method.

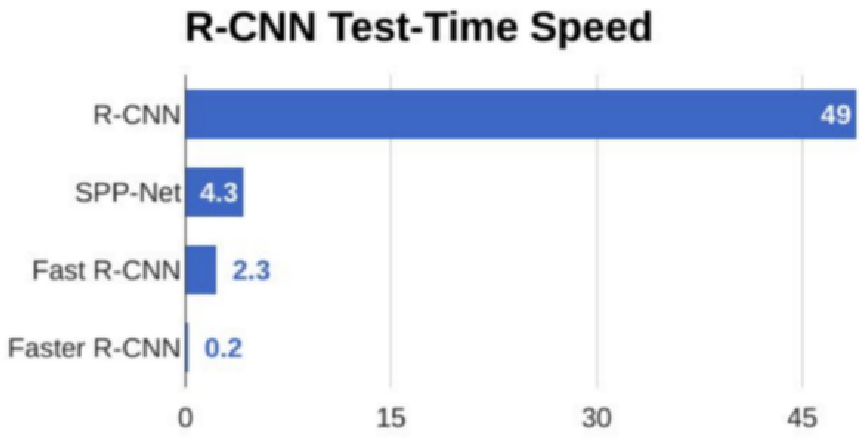


Figure 2.7. Comparison of test time speed of R-CNN based algorithms from Fei-Fei Li (2017)

The Faster R-CNN has two networks: one that extracts feature maps from the CNN for generating region proposal networks and a second network for detecting objects using these proposals. When the region proposal network slides over the image, it generates anchors of different shapes and sizes. Here, the region proposal network ranks the region boxes also known as the anchor and proposes the one most likely containing the objects. The predictions are made for each anchor based on two things: one, the probability that an anchor is an object, two, the bounding box regressor to better fit the object. From here, the anchors are passed through the ROI layer to extract feature maps from the anchor. It is then passed through an FCN which has a soft-max and a linear regressor layer to make predictions of the objects Ren et al. (2015). The architecture of the Faster R-CNN algorithm can be understood from Figure 2.8:

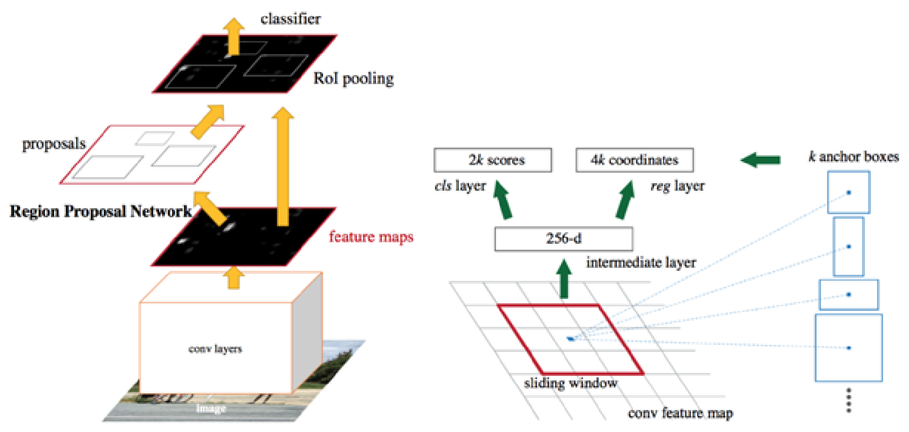


Figure 2.8. Architecture of Faster R-CNN for object detection from Shaoqing Ren (2016)

2.7.2 RoI Align Layer

The RoI Pooling Layer has a negative impact in predicting the masking operations which impacts the classification to a large extent hence it can be used. In a typical RoI pooling layer, a feature map is first extracted by first quantizing it to discrete granularity. It is then divided into spatial bins which are quantized and aggregated by using max pooling. The process of quantization generally inserts misalignment between the extracted features and the Region of Interest and hence it is not robust for classification task as it does not generate accurate masks. To address this problem RoI align layer is used in Mask R-CNN (He et al., 2017).

In RoI Align Layer, no quantization is performed. Here, bilinear interpolation is computed for each RoI bin and the result is later aggregated by using (average or max pooling). This leads to a large improvement in classification tasks thereby reducing the misalignment.

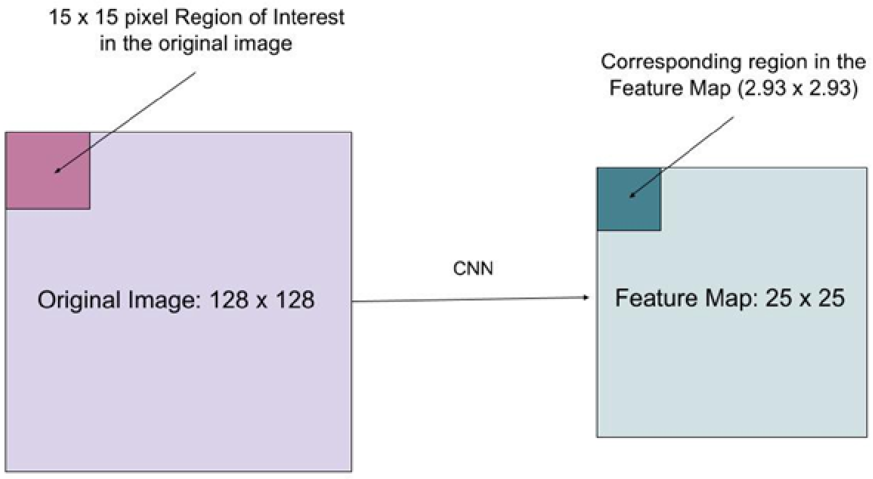

15 x 15 pixel Region of Interest in the original image
Corresponding region in the Feature Map (2.93 x 2.93)
Original Image: 128 x 128
CNN
Feature Map: 25 x 25

Figure 2.9. Accurate mapping of original image to feature maps by using RoI align layer from Lilian Weng (2017)

2.8 Single Shot Multi-Box Detector 300

The R-CNN based algorithms cannot be used for on the go object detection as they face following problems, one: In these networks, training happens in multiple phases, two: the network is too slow at inference time, three: training the data is too long and unwieldy. To address the bottlenecks of the R-CNN based algorithms, SSD detector was introduced (Liu et al., 2016).

A typical CNN for object detection contracts the feature map and grows the depth as it penetrates to the deeper layers. The shallow layers enfold smaller receptive field while the deeper layers are responsible for the construction of more abstract features and cover larger receptive fields. Thus, deeper layers are used to predict big objects while shallow layers are used to predict small objects.

The SSD-300 architecture has been derived from VGG-16 architecture with FCN layers discarded from it. The VGG-16 architecture can be used for image classification task as it offers high performance and can be used for improving the accuracy by transfer learning. The features here are extracted by reducing the size of each layer of the input and using a set of convolutional layers (Liu et al., 2016).

The underlying architecture of the SSD has VGG net as base net where the first feature map is generated from the VGG net layer with a depth of 512 and size of 38x38. For every point, there exist 512 channels and the feature map of size 38x38 can be used to map the complete image using this. Image classification can be done using the features in 512 channels and various labels can be predicted at every point. As the network moves forward, the size of the feature map is reduced to 19x19 which can now detect larger objects as it can cover the bigger receptive field. The last layer just has one point which is used for detecting big objects (Liu et al., 2016).

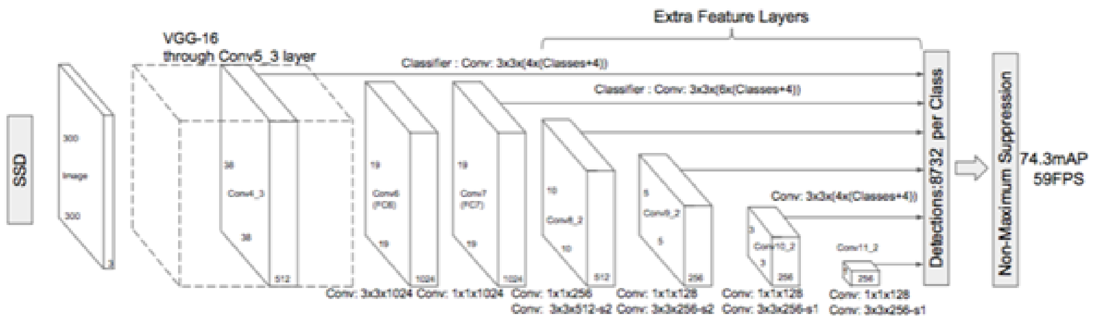


Figure 2.10. Architecture of Single Shot Multi-Box Detector from Wei Liu (2016)

2.9 You Only Look Once (YOLOv3)

The pre-existing object detection framework is based on RPN which involves refining the bounding boxes, rescoring the bounding boxes based on other objects and eliminating duplicate detections. Here, each individual component needs to be trained separately as the pipeline is complex and slow. An alternate approach to detect objects is using you only look once (YOLO). Multiple bounding boxes can be detected by a single convolutional network. Here are some of the benefits of using YOLO over other models.

Firstly, it can run at 45 fps and can process streaming video in real-time which makes it a very fast object detection network. Also, it does not need any batch processing and can run at 150 fps on a Titan X GPU. The mean average precision achieved by this network is twice as much as the traditional networks. Secondly, it doesn't make use of a sliding window and RPN technique to make predictions of the class. During the training period, it sees the entire image and encodes the contextual information of the appearance as well as the class. YOLO suffers from some trade-offs as it struggles to precisely localize small objects and thus the start-of-art performance is not guaranteed.

2.9.1 Network Design of YOLOv3

Initially, the YOLO model was implemented on the PASCAL VOC dataset where initial layers of the CNN were used for feature extraction and the FCN layers were used for prediction of coordinates and probabilities. The network has 24 convolutional layers along with 2 FCN layers. It uses a 1x1 reduction layer from the GoogleNet followed by 3x3 convolutional layer. The architecture of the YOLO network has been depicted in Figure 16. Here, the task is to predict (7,7,30) tensor. The spatial dimension of the network is reduced by using a CNN network and it has 1024 channels. The algorithm performs regression using the 2 FCN layers to make predictions. The objects with high prediction score are kept while the remaining ones are discarded.

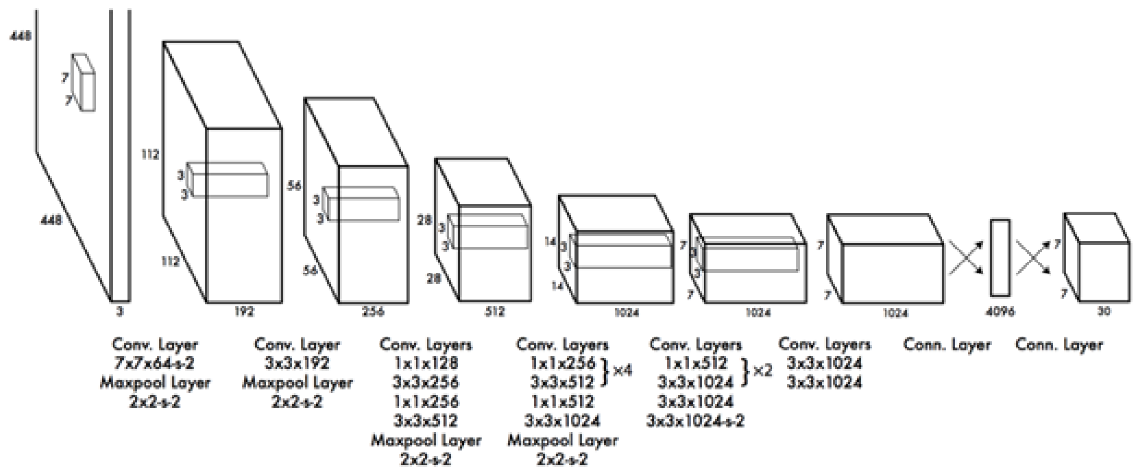


Figure 2.11. Architecture of YOLO from Joseph Redmon (2016)

Even though YOLO can be used for real-time object detection, the model has high localization errors and recall thus reducing the overall accuracy of the model. A better version of YOLO; YOLOv2 and YOLOv3 have been found to improve the accuracy of the model while making it faster. YOLOv2 can achieve a mean average precision (mAP) of 76.8 at 67 fps outperforming pre-existing traditional algorithms like Faster R-CNN and SSD. YOLOv2 tries to address the problem of accuracy by improving localization and recall while maintaining accuracy.

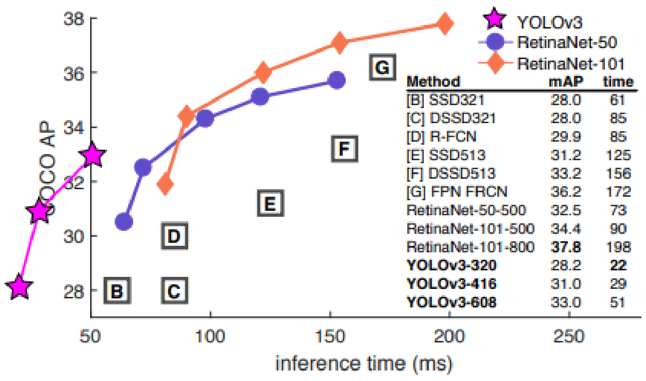
- Batch Normalization: It eliminates the need for regularization and leads to significant improvement in convergence. By doing this, an improvement of 2 percent mAP is obtained in the pre-existing YOLO model. Dropout from the model is removed by using batch normalization which prevents overfitting.
- High-Resolution classifier: The original YOLO network trains the network at 224x224 and followed by 448 for object detection. Meanwhile, the YOLOv2 network tunes the network at 448x448 with only 10 epochs. This improves the mAP by up to 4 percent.

- Convolution with anchor boxes: The traditional object detection algorithms use convolutional layers to predict offsets and confidence for anchor boxes. YOLO, on the other hand, is used for prediction of coordinates of the bounding boxes using FCN layers on top of feature extractor. Predicting offsets is better than predicting coordinates as the task of learning is simplified and made easier, thus we remove the FCN layers from YOLO and predict bounding boxes using anchor boxes.

On using YOLOv2 for Object Detection in real-time it was found by the researchers that YOLOv2 often struggled with small object detection even with a 19-layer architecture. The YOLOv2 lacked one of the most important features that are stapled to most state of art algorithms: The residual block, Upsampling, and No skip connections. All these problems were addressed by YOLOv3. The YOLOv3 is based on Darknet-53 architecture that has been trained on ImageNet. For detection task, another 53 layers have been stacked on top of it, which makes the object detection task a little slow at 30 frames per second, but it has an edge over YOLOv2 for the following reasons:

- It is better at detecting small objects: In YOLOv3 the 52x52 layer is responsible for detecting the small objects whereas the 13x13 layer is responsible for the detection of large objects. Meanwhile, the 26X26 layer is responsible for the detection of medium objects. In YOLOv3 the upsampled layers concatenated with previous layers are responsible for the detection of small objects, a feature that is not seen in YOLOv2 (Redmon & Farhadi, 2018).

- More bounding boxes per image: YOLOv3 predicts bounding boxes at 3 different scales and thus it can predict more bounding boxes than YOLOv2. For instance, for a 416x416 resolution image, the YOLOv2 can predict only 13x13x5 boxes which is equivalent to 845 but the YOLOv3 can detect 10,647 boxes for the same image (Redmon & Farhadi, 2018).



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 2.12. Inference time comparison of YOLOv3 with other models from Joseph Redmon (2017b)

YOLOv3 performs at par with other states of art detectors like Retina Net. It is also better than SSD and its variants. Here’s a comparison of performances right from the paper.

2.10 Summary

One of the most important factors influencing a person's survival following a natural disaster is the rate at which they are rescued. With increased disasters (natural or man-made) in the last five decades and traditional methods of search operations (humanitarian, ground robot) the rescue rate is incredibly low which needs to be addressed. With advancements in technology and hardware, a more promising way of search operations during a disaster is the use of deep learning and photogrammetry for precise search and rescue operations to maximize the rescue rate and reduce casualties. The literature review first discusses the traditional methods of search and rescue operations which later shifts the focus to current trend of machine learning and deep learning for object detection in real time.

CHAPTER 3. METHODOLOGY

In this chapter, we will discuss the framework and the proposed methodology used in this research. The methodology includes details on the experimental setup, the hardware platform, the parameters on which the models will be assessed and steps on how the results will be validated through this research.

3.1 Research Framework

The aim of this research is to address the traditional methods of search operations using deep learning. Using the concept of object detection, classes (objects) can be identified in real-time with precise geo-location using the concept of photogrammetry. This location can then be shared with the rescue teams to carry out rescue operations and therefore minimize the downtime (usually 48-72 hours post floods) it takes to carry out these operations. The study focuses on answering the following research question: Does the proposed modified version of trained YOLOv3 perform faster (inference time) and accurately (performance metrics) than the existing version of pre-trained models of Mask-RCNN, SSD300 and YOLOv3 to classify objects (person, car) in real-time (20 fps) in flood scenes for search operations?

3.2 General Methodology

This section talks about the methodology used to address the research question. Figure 3.1 and Figure 3.2 represent the overall procedure of object detection in flood videos using drones for the real-time search operation.

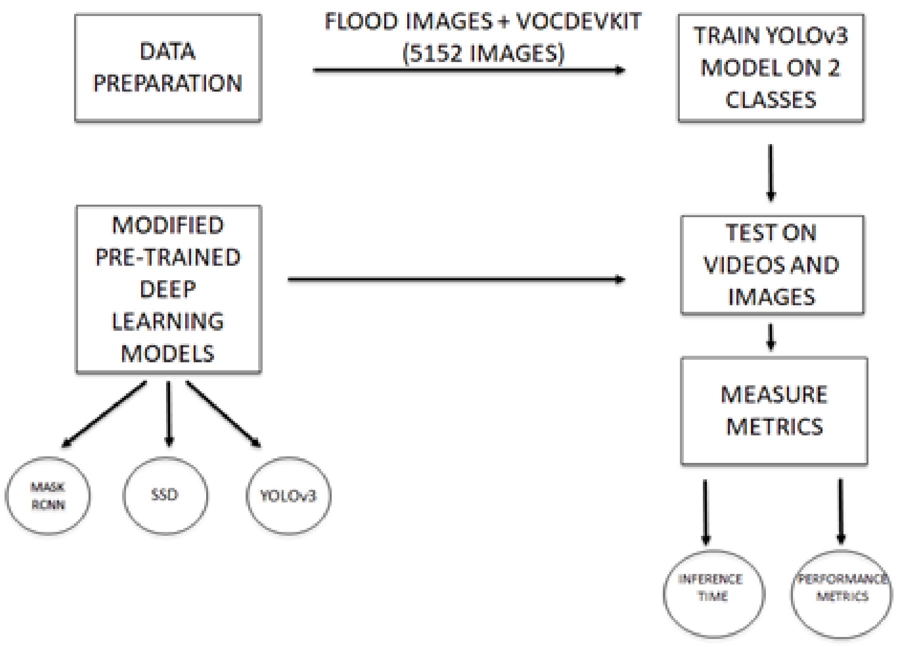

DATA PREPARATION
FLOOD IMAGES + VOCDEVKIT (5152 IMAGES)
TRAIN YOLOv3 MODEL ON 2 CLASSES
MODIFIED PRE-TRAINED DEEP LEARNING MODELS
TEST ON VIDEOS AND IMAGES
MEASURE METRICS
MASK RCNN
SSD
YOLOv3
INFERENCE TIME
PERFORMANCE METRICS

Figure 3.1. Workflow architecture for object detection using pre-trained and modified trained YOLOv3 model

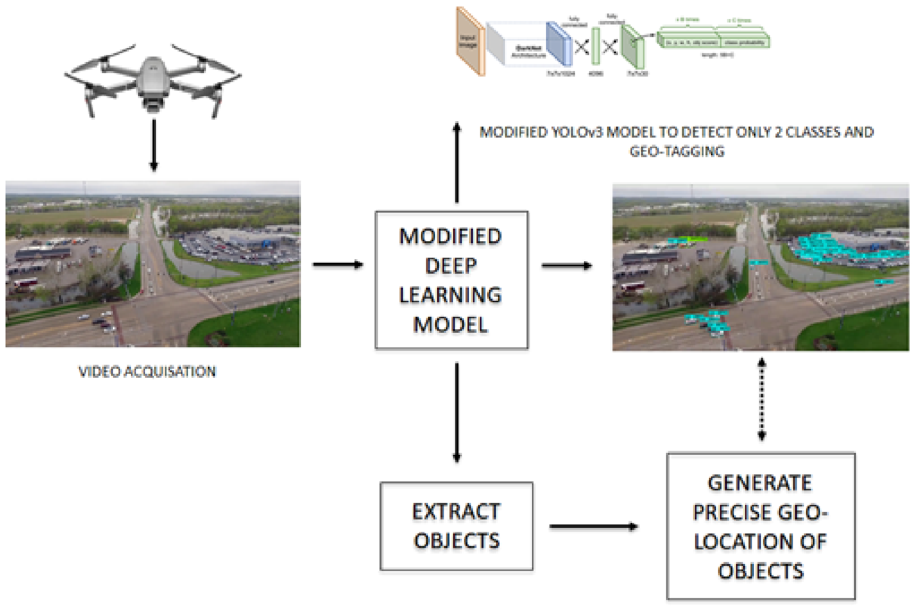

MODIFIED YOLOv3 MODEL TO DETECT ONLY 2 CLASSES AND GEO-TAGGING
VIDEO ACQUISATION
MODIFIED DEEP LEARNING MODEL
EXTRACT OBJECTS
GENERATE PRECISE GEO-LOCATION OF OBJECTS

Figure 3.2. A visual illustration of object detection using modified trained YOLOv3

3.3 Modifications to YOLOv3

Section 2.9 gives a detailed explanation of the YOLOv3 algorithm for object detection. To summarize the YOLOv3 algorithm: It is based on Darknet-53 architecture and makes detections at three different scales which are obtained by down sampling the dimensions of the input image by 32,16 and 8. It is better at detecting small objects which help in improving the accuracy of the algorithm and it is capable of predicting more bounding boxes than YOLOv2. In the study done by Joseph Redmon (2017) the YOLOv3 algorithm was able to process images at 30fps with a mAP of 57.9 percent in 51ms inference time. The overall performance of the model dropped significantly. Hence in order to improve the performance of the model, the YOLOv3 model is trained to detect only 2 objects in a flood scenario i.e. inference time and performance metrics are taken into consideration for comparison with pre-trained models.

The proposed version of YOLOv3 model is trained on 5152 images. Before training the YOLOv3 model, the labels of choice (person, car) have been labeled for every image using the software LabelImg as shown in Figure 3.3.

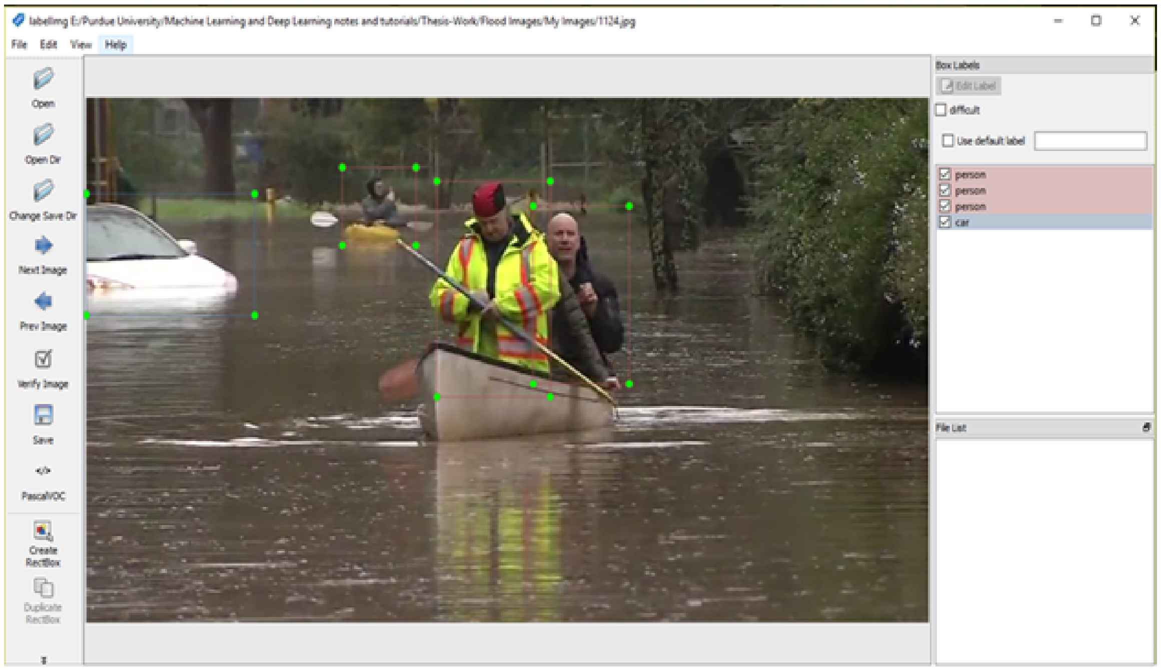


Figure 3.3. A visual illustration of object detection using modified trained YOLOv3

Table 3.1. Hyperparameter values for YOLOv3

Hyperparameter	Value
Batch	64
Momentum	0.9
Decay	0.0005
Epochs	5000
Learning Rate	0.001
Max Batches	40100
Saturation	1.5
Exposure	1.5
Hue	0.1
Subdivision	8
Width	416
Height	416
Channels	3

The images were then trained on only 2 classes (person, car) with hyperparameter values as stated above.

3.4 Object detection using pre-trained models

Since the purpose of object detection here is mainly to carry out search operations, it is expected that the models identify humans in real-time with maximum accuracy. Training a neural network is a time-intensive task and it is expected that pre-trained models be used for search and operations for on the go object detection. To train a neural network from scratch requires training the neural network with random weights. The weights are saved once the training is completed. To perform similar object detection tasks, it is intended to use the weights that can be trained and be optimized on the new dataset rather than training the dataset again. The process of using random weights from the training network is known as pre-training.

When trying to pre-train a network with large data set one common way to do so is by removing weights of FCN layers and replacing it with random weights. There are several advantages to using a pre-trained network. First, in order to obtain good accuracy on the test sets, it requires training a neural network which is not always available in researching field. The color patterns, circles, arcs, and edges are common to all objects and we can learn a variety of information and basic features from these data sets. Fine-tuning can be done when a pre-trained network is to be applied to a small data set. Second, ample time can be saved by using pre-trained networks. To train a neural network from scratch we can take weeks to months even if one wants to train these networks on GPUs with multiple cores. A small revision in the neural network could cost multiple hours of time which is not a feasible thing to do. In my experiments in order to save time I have used pre-trained models. Below is the description of the pre-trained models I used for my research:

3.4.1 ResNet-101 for Mask R-CNN

In the past, state of art CNN architectures like AlexNet had only 5 convolutional layers, the GoogleNet had 22 convolutional layers and the VGG Network had 22 layers. As the network goes deeper, its performance starts degrading rapidly as these networks are hard to train because of the vanishing gradient problem. A way to tackle the problem was to introduce a short-cut connection that skipped one or more layers as shown in Figure 3.4. The introduction of the shortcut connection helps skip one or more layers forming the bases of ResNet. A lower training error and a faster convergence were observed in the ResNet.

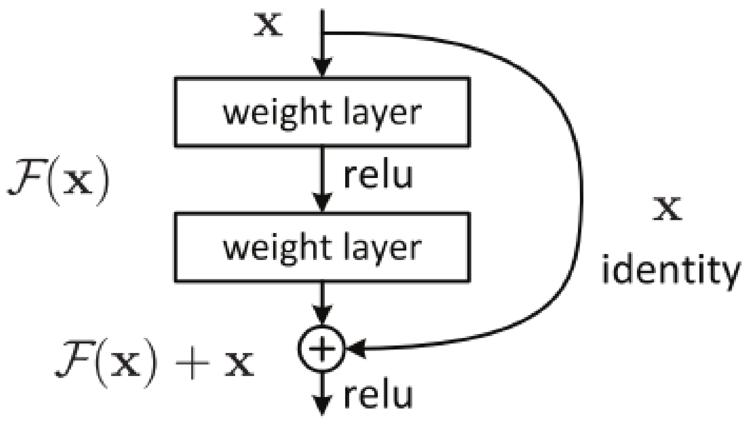


Figure 3.4. Building Block for ResNet-101 from Kaiming He (2015)

3.4.2 Darknet-53 for YOLOv3

The Darknet-53 model used in YOLOv3 is a hybrid approach between Darkent-19 and the residual network. It has 53 convolutional layers and so it is called as Darknet-53. The nertwork uses successive 3X3 and 1X1 convolutional layers along with shortcut connections. The network is more powerful and efficient than ResNet-101 and Darknet-19. Darknet is 1.5 times faster than ResNet-101 and can perform state-of-art classifications with more speed.

Table 3.2. Comparisons of underlying backbone used in research from Joseph Redmon (2017a)

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19	74.1	91.8	7.29	1246	171
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	77.6	93.8	29.4	1090	37
DarkNet-53	77.2	93.8	18.7	1457	78

3.5 Photogrammetry - Mapping detected objects from algorithms to GPS location

The science of making measurements from photographs is known as photogrammetry. By using the concept of photogrammetry, the exact positions of the surface points can be estimated. The distance between two points on a photographic image plane can be determined if the scale on the image is known (Mikhail et al., 2001). Using the concept of photogrammetry, precise search and rescue operation can be carried out. The geo-location of the center of the image can be calculated by using the concepts,

- Ground Sample Distance.
- Haversine Formula
- Elevation from the ground

3.5.1 Ground Sample Distance

It is the distance between the center of two points on the earth surface. The concept of the ground sample distance can be understood in figure 3.5. To calculate the GSD the specifics of the camera sensor of the drone must be known Drone Blocks (2013).

The equation to calculate the GSD is given by,

$$D_s = \frac{sw * al * 100}{fl * iw} \tag{3.1}$$

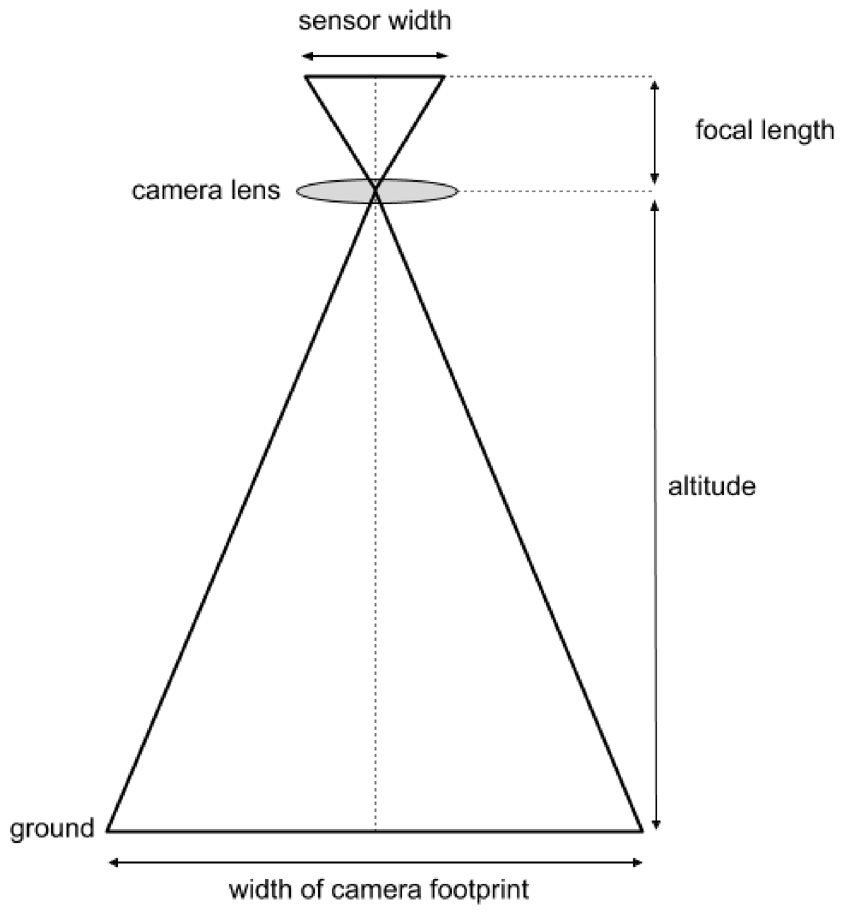


Figure 3.5. Concept of Ground Sample Distance from Drone Blocks (2013)

where,

sw - sensor width of the sensor fitted on the drone

al - altitude of the drone from the ground

fl - focal length

iw - image width

Once the Ground Sample Distance has been calculated, the width of the camera footprint needs to be calculated. It is given by the following formula,

$$W_f = D_s * W_p \tag{3.2}$$

where,

D_s - Ground Sample Distance

W_f - Width of camera footprint

W_p - Pixel Width of the Image

3.5.2 Haversine Formula

The haversine formula is used for mapping distance between two points on the sphere given their latitude and longitude. Concurrently, the haversine formula can even be used to find the geo-location of the second point provided the geo-location of the first point and the distance between two points are known. By using haversine formula, the objects can be mapped in real-time and precise location of objects can be shared with the search and rescue team. The formula is given by,

$$\theta = \frac{d}{r} \tag{3.3}$$

$$hav(\theta) = hav(\tau_2 - \tau_1)\cos(\tau_1)\cos(\tau_2)hav(\lambda_2 - \lambda_1) \tag{3.4}$$

$$hav(\theta) = sin^2\frac{\theta}{2} \tag{3.5}$$

r - radius of the sphere

d - distance between two points on the sphere

θ - central angle between two points

τ_2, τ_1 - latitude of point 2 and latitude of point 1

λ_2, λ_1 - longitude of point 2 and longitude of point 1

From this formula, the latitude and the longitude of the center of the image can be calculated.

3.5.3 Elevation from the ground

By the use of the Google Elevation API, the elevation on the ground can be determined by using a simple HTTP request. Below is the code:

```python
import http.client
from pygeocoder import Geocoder
import pandas as pd
import numpy as np
import simplejson
import urllib

def get_elevation(lat, lon):

    base_url = 'https://maps.googleapis.com/maps/api/elevation/json'
    url_params = "locations=%.7f,%.7f" % (lat, lon)
    url = "%s?%s" % (base_url,  url_params)
    result = simplejson.load(urllib.urlopen(url))
    elevation = result["results"][0]["elevation"]

    return (elevation)


if __name__ == '__main__':

    lat = -29.26399994
    lon = -61.02799988
    elevation = get_elevation(lat, lon)
    print(elevation)
```

Listing 3.1: Extraction of elevation from ground based on latitude and longitude

3.6 Assumptions made in the research study

Here, it has been assumed that the metadata from the drone (GPS coordinates) can be easily extracted for every frame which then could be used to estimate the GPS coordinates and the distance of humans in a flood. To demonstrate how the metadata could be used for precise search and rescue operation in a flood, the flight log of DJI drone was extracted from the website `http://app.airdata.com/share/yLhFyj`. Here, the metadata (drone latitude, drone longitude, time, altitude, speed, voltage, compass heading, battery percentage) was being recorded every 100 milliseconds. Based on this metadata

Table 3.3. Flight log of DJI Drone

Time (ms)	Datetime (utc)	Latitude (deg)	Longitude (deg)	Altitude (m)
42000	8/28/2017 9:39:48	55.588606	38.932124	123.7562
42100	8/28/2017 9:39:48	55.589939	38.935163	123.7562
42200	8/28/2017 9:39:49	55.589984	38.935193	123.7564
42400	8/28/2017 9:39:49	55.589997	38.935207	123.7566
42600	8/28/2017 9:39:50	55.590156	38.935248	123.7570
42700	8/28/2017 9:39:50	55.59114	38.935295	123.7590

the drone's latitude, longitude for every image could be then used to map the human location by the concept of photogrammetry for precise search operations. Currently, in my research, the GPS coordinates of the drone have been hardcoded due to lack of availability of GPS drone data. Table 3.3 gives an illustration of the metadata of the drone flight.

3.7 Research Type

This is qualitative research with the aim to study the YOLOv3 thoroughly and investigate many ways by which the speed (inference time) and the accuracy (performance metrics) of the algorithm can be improved while preserving the quality of the algorithm at the same time. The results are based on a statistical analysis of the running time and performance metrics of the pre-trained models like Mask-RCNN, SSD300, YOLOv3 and running time and performance metrics of the proposed version of the trained YOLOv3.

3.8 Evaluation Criteria

A Deep Learning model can be evaluated based on two parameters:

- Inference time
- Performance Metrics

3.8.1 Inference time

Inference time is one of the key parameters that should be considered while assessing a model's performance. By using this metric along with the performance metric, we have tried assessing the model that can be deployed in real-time for a typical search operation in a flood.

3.8.2 Performance Metrics

The results of object detection can be verified by using two criteria's, overall recall and overall precision. Figure 3.1 depicts the overall precision and recall. Precision means the identification of values that are actually correct." Recall means identification of actual positives correctly." The formula, of precision and recall, are given by,

$$Precision = \frac{TP}{TP+FP} \tag{3.6}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.7}$$

3.9 Experimental setup of computational resources on google colaboratory

For my research, I have used google colaboratory, a jupyter notebook environment that supports Python 2.7 and Python 3.6. google colaboratory is a research tool for machine learning, deep learning education, and research. The service doesn't require any setup to use and runs entirely on the cloud. With colaboratory, codes can be written, executed and shared as it gives free access to powerful computing resources from the web browser. The codes are executed in a virtual machine but if the machines are kept idle for too long, the machines are recycled, and the notebook should be executed again. Here are the hardware specifications of the system I will be using for my research:

- GPU: 1xTesla K80 , 2496 CUDA cores, compute 3.7, 12GB (11.439GB Usable) GDDR5 VRAM
- CPU: 1xsingle core hyper threaded (1 core, 2 threads) Xeon Processors @2.3Ghz (No Turbo Boost) , 45MB Cache
- Disk: 320 GB (HDD)

For my analysis of deep learning models, I have used the Tesla K80 GPU with all other hardware configurations intact and have found serious improvement in computation speed.

3.10 Summary

Here, we first discussed the research framework and the general methodology of how we have set up the experiment for object detection and geo-location mapping of the object to the drone by using the concept of photogrammetry. Later in the chapter, we discussed evaluation metrics methods like inference time, performance metrics like precision and recall followed by the experimental setup we used during our research.

CHAPTER 4. EXPERIMENT RESULTS

In this chapter, we will present the results of the experiments we conducted in this research. First, we will discuss the procedure in which data has been collected and prepared in our research. Next, we will introduce the data sets like COCO that have been used to label our objects followed by the implementation methodology discussed in Chapter 3. In the end, the results have been compared that best suits our needs.

4.1 Data collection and preparation

The dataset of the flood was scraped from google manually. More than 1700 images were extracted on querying floods from the search engine. The search queries have images dating as old as of June 2000. The dataset collected from google has been collected over a span of over 20 years. Over 200 images and 10 videos were fit (test set) on the models (pre-trained and trained) to test the inference timing and the performance metrics of these models for a precise search operation in real-time.

4.2 Annotation of objects using COCO (Common Objects in Context)

In the original work done by He et al. (2017); Redmon and Farhadi (2018) state art of performance was obtained when the models were fit on COCO data set, hence in order to obtain same results COCO annotations have been used. Object recognition can be divided into three categories to address the following: semantic scene labeling, object classification, and object detection. The MS COCO data set was designed for detection, segmentation and scene labeling of objects occurring in a natural context. The data set has few categories but more instances per category which can help in precise 2D localization. Additionally, the contextual information is aided by the number of labeled instances per image (Lin et al., 2014). Some of the key features of the COCO data set are:

- 328,000 images with 2,500,000 labeled instances

- 82 object categories with more than 5000 labeled instances
- Overall 91 object categories
- Object Segmentation
- 5 captions per image
- Super pixel stuff segmentation
- Recognition in context

The MS COCO dataset has been used for annotation of all the detected classes in pre-trained models Mask R-CNN, SSD-300, and YOLOv3.

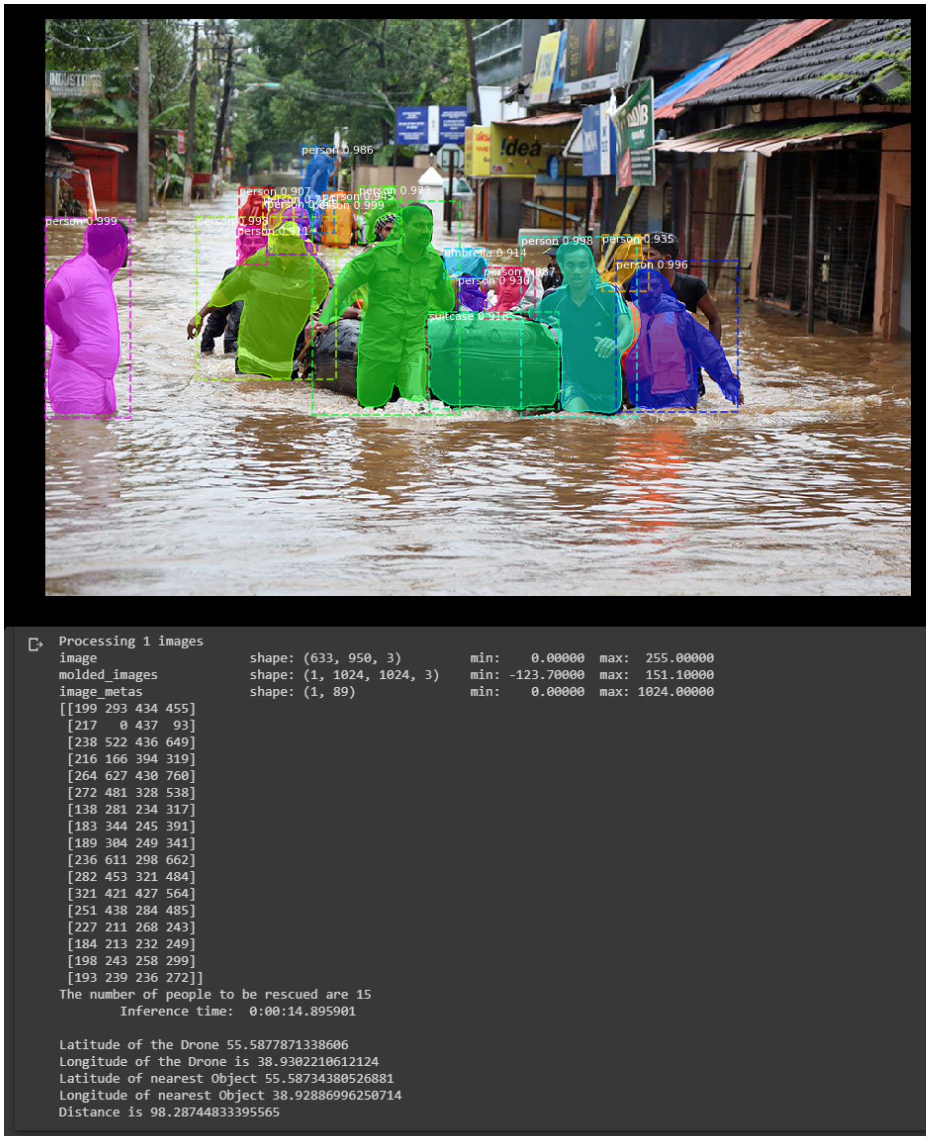
4.3 Implementation details and results

4.3.1 Pre-trained Mask-RCNN model on 80 classes

As inference time and prediction metrics are very critical for any deep learning model, in this section, the pre-trained Mask-RCNN model fit on 80 classes (COCO annotations). The inference time and performance metrics for the pre-trained are then compared with the proposed YOLOv3 model for both videos and images for precise search operations of humans in floods.

In this section, the Mask-RCNN configuration file and pre-existing weights were downloaded from `https://github.com/matterport/Mask_RCNN`.The Mask-RCNN model is based on ResNet-101 architecture. The classes (annotations) were retrieved from the COCO website `https://github.com/cocodataset/cocoapi` for annotation purpose once the masked objects were identified on the pre-trained Mask-RCNN model.

For every frame in case of videos/image in the Mask-RCNN, the bounding box coordinates of the objects were first extracted as shown in Figure 4.1. Based on the bounding box image coordinates of the objects, the distance from the drone to the object along with GPS coordinates of the object were estimated using the concept of photogrammetry (ground sample distance and haversine formula) for precise search operation of the objects in floods. Figure 4.1 shows the GPS coordinates of the object and distance estimation from the drone to the object obtained by us for every image in the Mask-RCNN model.

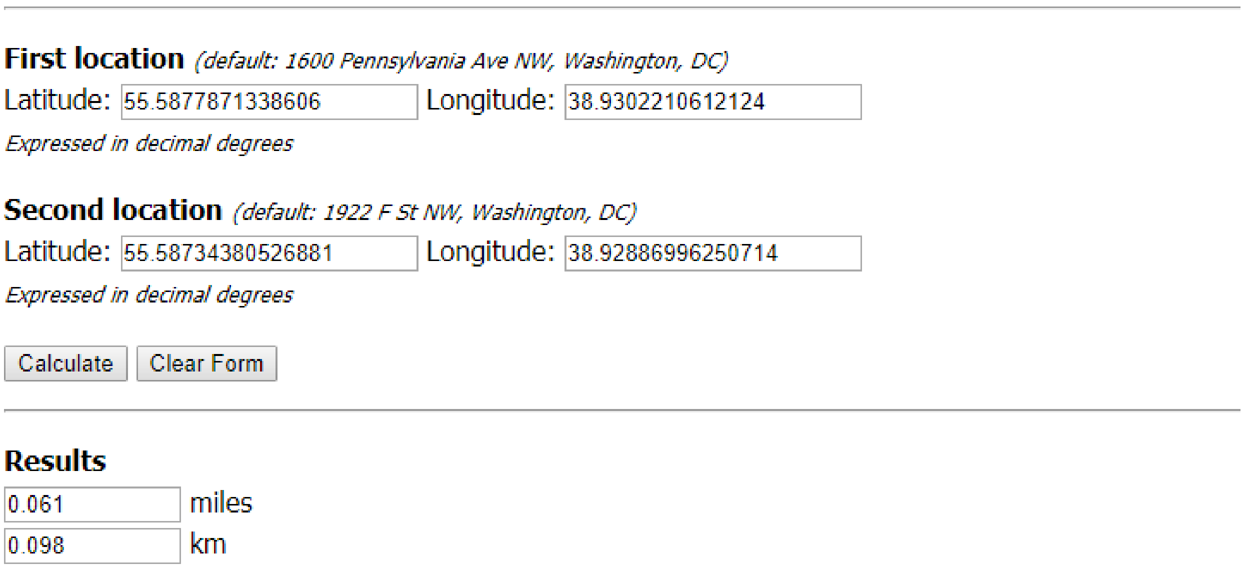


```
Processing 1 images
image                    shape: (633, 950, 3)         min:    0.00000  max:  255.00000
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.10000
image_metas              shape: (1, 89)               min:    0.00000  max: 1024.00000
[[199 293 434 455]
 [217   0 437  93]
 [238 522 436 649]
 [216 166 394 319]
 [264 627 430 760]
 [272 481 328 538]
 [138 281 234 317]
 [183 344 245 391]
 [189 304 249 341]
 [236 611 298 662]
 [282 453 321 484]
 [321 421 427 564]
 [251 438 284 485]
 [227 211 268 243]
 [184 213 232 249]
 [198 243 258 299]
 [193 239 236 272]]
The number of people to be rescued are 15
        Inference time:  0:00:14.895901

Latitude of the Drone 55.5877871338606
Longitude of the Drone is 38.9302210612124
Latitude of nearest Object 55.58734380526881
Longitude of nearest Object 38.92886996250714
Distance is 98.28744833395565
```

Figure 4.1. GPS estimation of objects using Mask R-CNN model

The distance between the GPS coordinates of the object and the GPS coordinates of the drone was further validated on the haversine calculator available on `https://andrew.hedges.name/experiments/haversine/` with GPS coordinates of the two points provided in the calculator as shown in Figure 4.2.



First location *(default: 1600 Pennsylvania Ave NW, Washington, DC)*

Latitude: 55.5877871338606 Longitude: 38.9302210612124

Expressed in decimal degrees

Second location *(default: 1922 F St NW, Washington, DC)*

Latitude: 55.58734380526881 Longitude: 38.92886996250714

Expressed in decimal degrees

Results

0.061	miles
0.098	km

Figure 4.2. Validation of distance obtained in Figure 4.2 using Haversine Formula

The experiment was conducted on a test set of 10 videos of 2 minutes each and on 200 images separately. On fitting the Mask-RCNN model on batches of 10 images for 200 images, it was found that the inference time taken by 200 images on google colaboratory varied between 96.5s to 99s with a median of 98.2 and an outlier on 95.5s. Similarly, when the Mask-RCNN model was fit on 10 flood videos of 2 minutes each, the inference time varied between 2800s to 2600s with a couple of outliers at 3500s and 1750s as depicted in the boxplot in Figure 4.3.

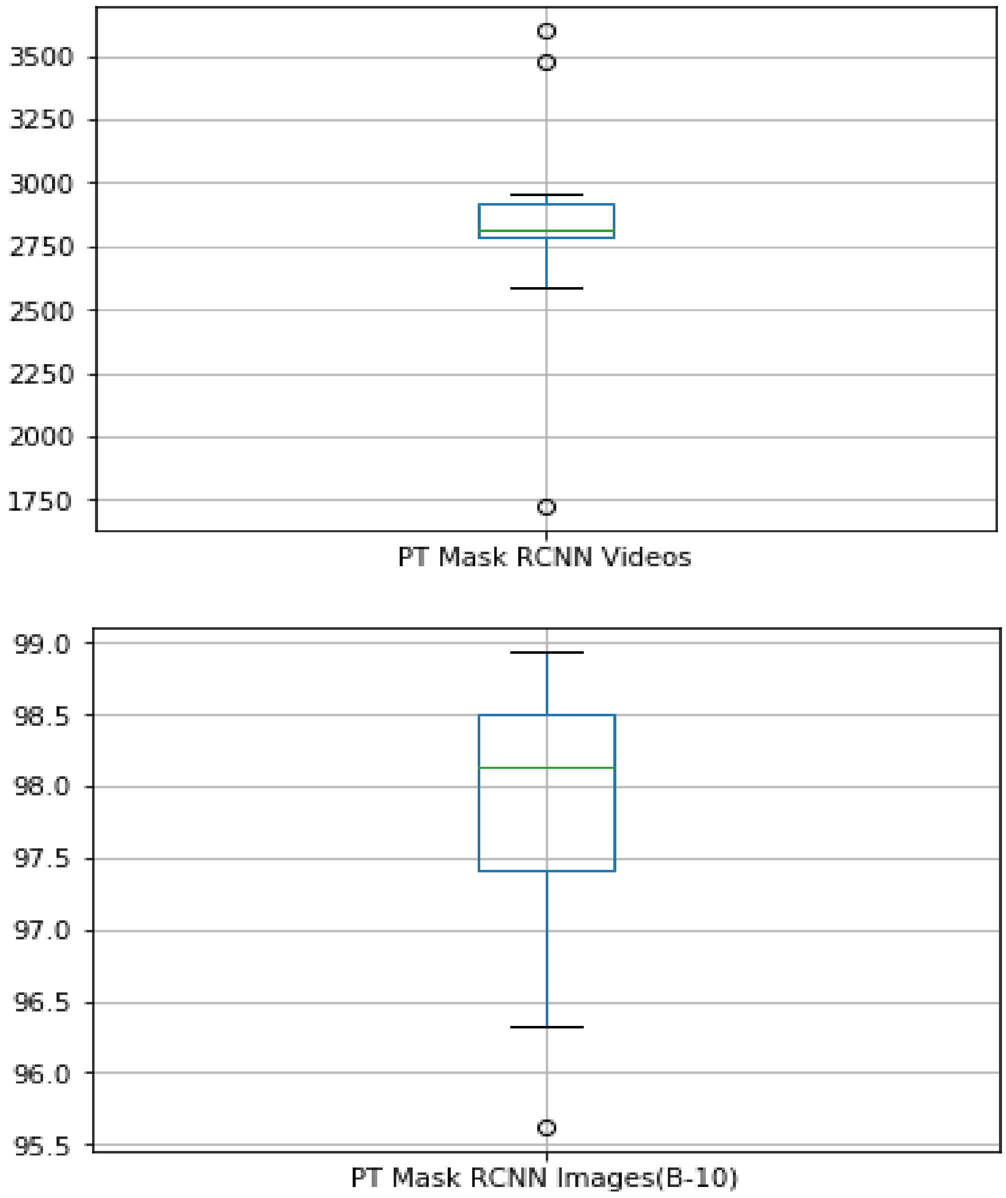


Figure 4.3. Inference time for pre-trained Mask R-CNN model

The performance metric parameters, precision, and recall for the Mask-RCNN model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.5 for images and 0.2 for videos. If the IoU was more than a threshold, then a true positive was confirmed. If IoU was less than the

Table 4.1. Pre-trained Mask-RCNN confusion matrix - Images

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 834$	$FN = 96$	$834 + 96$
	(0)	$FP = 90$	$TN = 303$	$90 + 303$
	Total	$729 + 70$	$86 + 438$	1323

Table 4.2. Pre-trained Mask-RCNN confusion matrix - Videos

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 3104$	$FN = 4124$	$3104 + 4124$
	(0)	$FP = 1278$	$TN = 1730$	$1278 + 1730$
	Total	$3104 + 1278$	$4124 + 1730$	10236

threshold, a false positive was confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. For images and videos for Mask-RCNN, as depicted in the confusion matrix, there were a considerable number of false negatives and false positives which played a crucial role in the determination of precision and recall values. The precision and recall curve for pre-trained Mask-RCNN model has been depicted below.

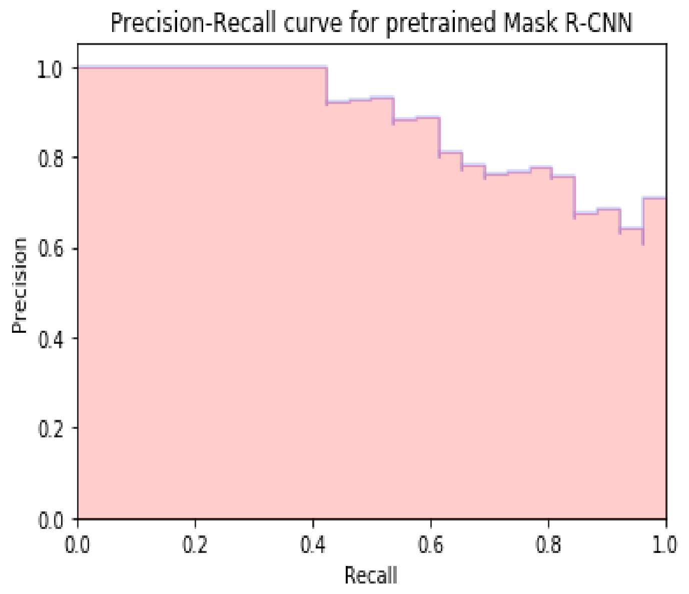


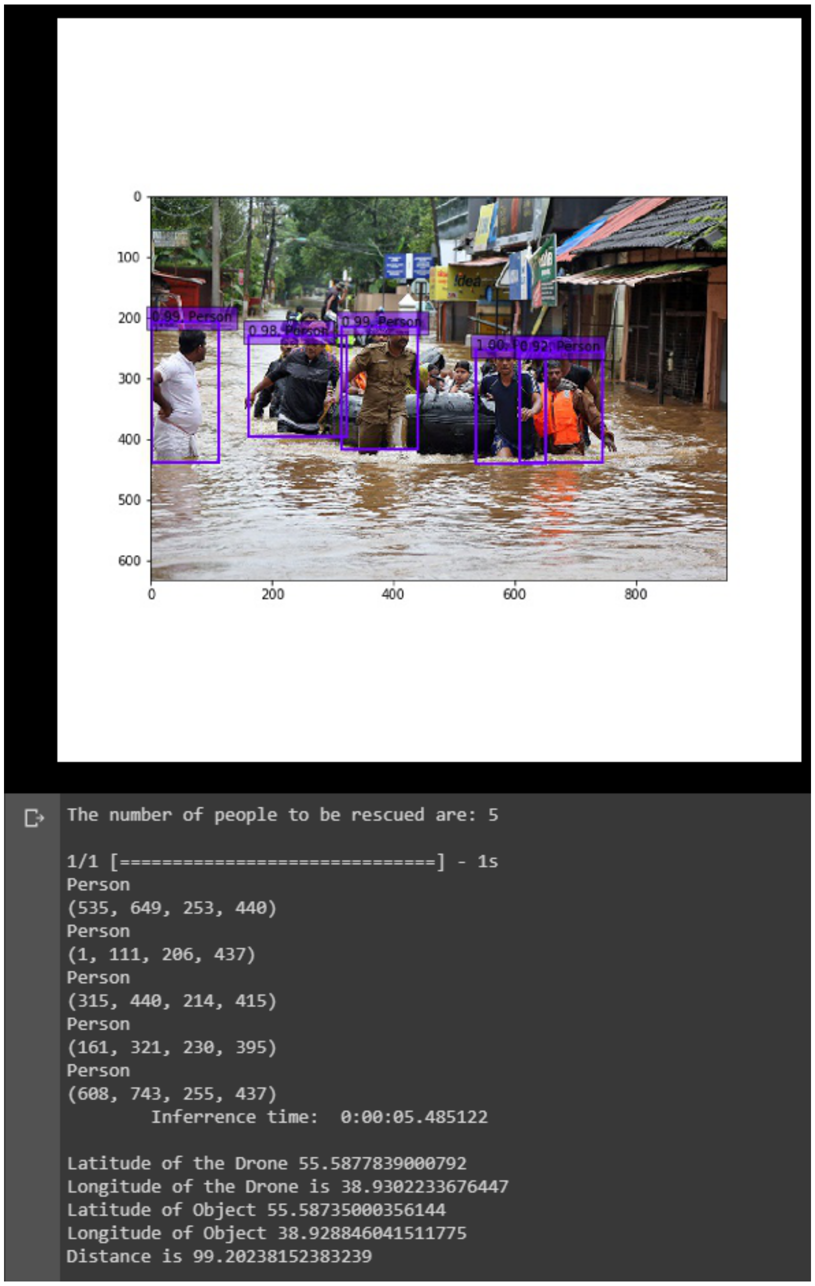
Figure 4.4. Precision Recall curve for pretrained Mask-RCNN

4.4 Pre-trained SSD300 model on 80 classes

As inference time and prediction metrics are very critical for any deep learning model, in this section, the pre-trained SSD300 model fit on 80 classes (COCO annotations). The inference time and performance metrics for the pre-trained data were then compared with the proposed YOLOv3 model for both videos and images for precise search operations of humans in floods.

In this section the SSD300 configuration file and pre-existing weights were downloaded from `https://github.com/pierluigiferrari/ssdkeras`.The SSD300 architecture is based on VGG16 architecture. The classes (annotations) were retrieved from the COCO website `https://github.com/cocodataset/cocoapi` for annotation purpose once the predictions were made on the pre-trained SSD300 model.

For every frame in case of videos/image in the SSD300, the bounding box coordinates of the objects were first extracted as shown in Figure 4.4. Based on the first bounding box image coordinates of the person, the distance from the drone to the object along with GPS coordinates of the object were estimated using the concept of photogrammetry (ground sample distance and haversine formula) for precise search operation of the humans in floods. Figure 4.5 shows the GPS coordinates of the object and distance estimation from the drone to the object obtained by us for every image in the SSD300 model.



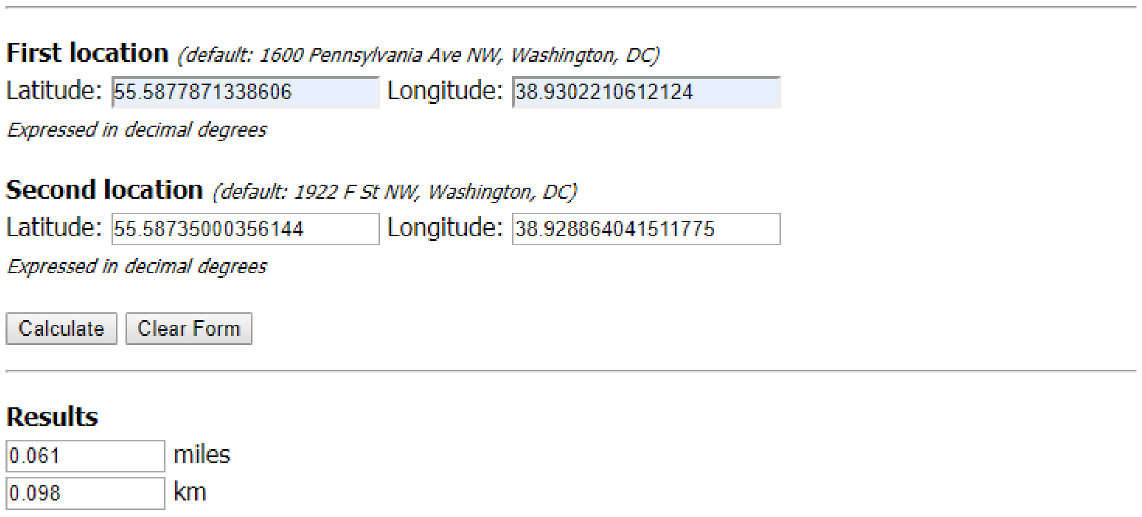
```
The number of people to be rescued are: 5

1/1 [==============================] - 1s
Person
(535, 649, 253, 440)
Person
(1, 111, 206, 437)
Person
(315, 440, 214, 415)
Person
(161, 321, 230, 395)
Person
(608, 743, 255, 437)
        Inferrence time:  0:00:05.485122

Latitude of the Drone 55.5877839000792
Longitude of the Drone is 38.9302233676447
Latitude of Object 55.58735000356144
Longitude of Object 38.928846041511775
Distance is 99.20238152383239
```

Figure 4.5. GPS estimation of objects using pre-trained SSD300 model

The distance between the GPS coordinates of the object and the GPS coordinates of the drone was further validated on the haversine calculator available on `https://andrew.hedges.name/experiments/haversine/` with GPS coordinates of the two points provided in the calculator as shown in Figure 4.5.



First location *(default: 1600 Pennsylvania Ave NW, Washington, DC)*
Latitude: 55.5877871338606 Longitude: 38.9302210612124
Expressed in decimal degrees

Second location *(default: 1922 F St NW, Washington, DC)*
Latitude: 55.58735000356144 Longitude: 38.928864041511775
Expressed in decimal degrees

Results
0.061 miles
0.098 km

Figure 4.6. Validation of distance obtained in Figure 4.5 using Haversine Formula

The experiment was conducted on a test set of 10 videos of 2 minutes each and on 200 images separately. On fitting the pre-trained SSD300 model on batches of 10 images for 200 images, it was found that the inference time taken by 100 images on google colaboratory varied between 68s to 62s with an outlier on 63.5s. Similarly, when the SSD300 model was fit on 10 flood videos of 2 minutes each, the inference time varied between 1020s and 550s with a median on 800s as depicted in the boxplot in Figure 4.6.

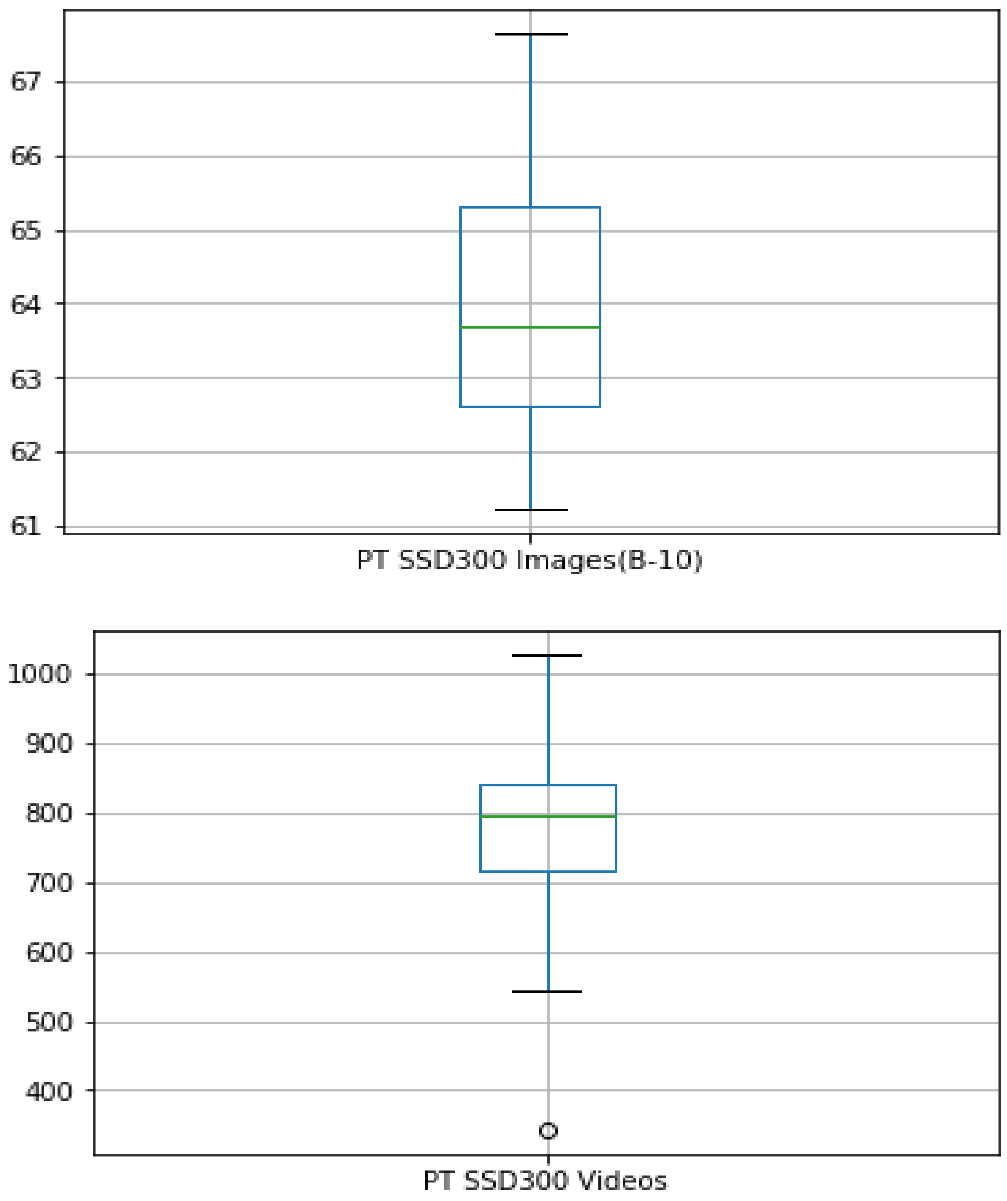


Figure 4.7. Inference time for pre-trained SSD300 model

The performance metric parameters, precision, and recall for the SSD300 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.5 for images and 0.2 for videos. If the IoU was more than a threshold, then a true positive was confirmed. If IoU was less than the

Table 4.3. Pre-trained SSD300 Confusion Matrix - Images

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 316$	$FN = 736$	$316 + 736$
	(0)	$FP = 58$	$TN = 213$	$58 + 213$
	Total	$316 + 58$	$736 + 213$	1323

Table 4.4. Pre-trained SSD300 Confusion Matrix - Videos

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 978$	$FN = 5862$	$978 + 5862$
	(0)	$FP = 1366$	$TN = 2030$	$1366 + 2030$
	Total	$978 + 1366$	$5862 + 2030$	10236

threshold, a false positive was confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. For images and videos for SSD300, as depicted in the confusion matrix, there were a considerable number of false negatives and false positives which played a crucial role in the determination of precision and recall values. The precision and recall curve for pre-trained SSD300 model has been depicted below.

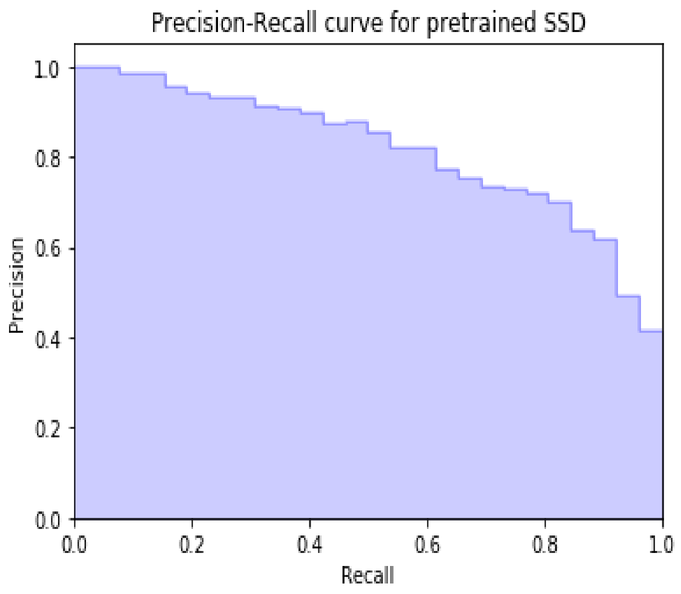


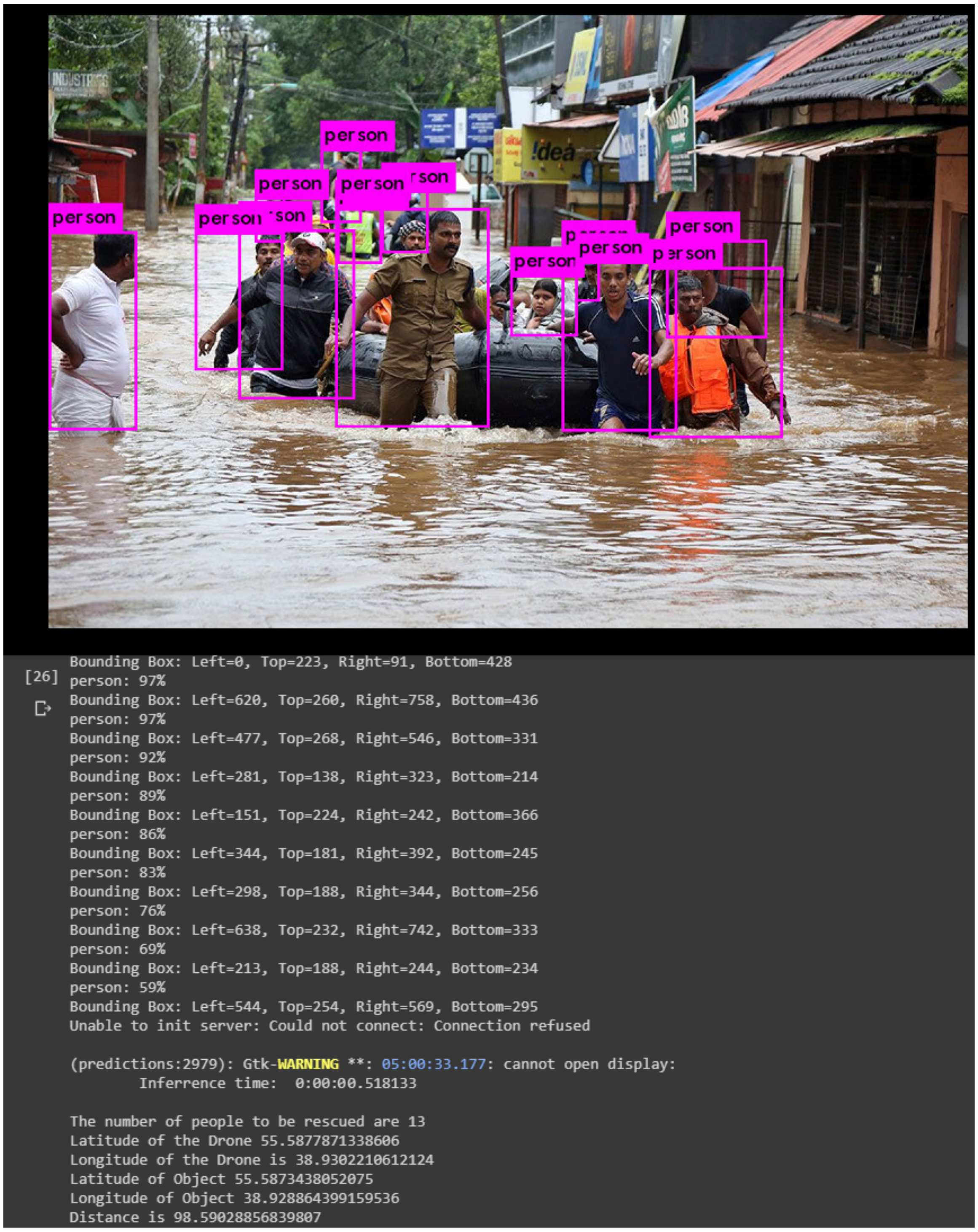
Figure 4.8. Precision Recall curve for pretrained SSD300

4.5 You Only Look Once (YOLOv3)

As inference time and prediction metrics are very critical for any deep learning model, in this section, the pre-trained YOLOv3 model fit on 80 classes (COCO annotations). The inference time and performance metrics for the pre-trained are then compared with the proposed YOLOv3 model for both videos and images for precise search operations of humans in floods.

For the pre-trained YOLOv3 model, the configuration file and pre-existing weights were downloaded from `https://pjreddie.com/darknet/yolo/`. The YOLOv3 model is based on Darknet-53 architecture. The classes (annotations) were retrieved from the COCO website `https://github.com/cocodataset/cocoapi` for annotation purpose once the predictions were made on the pre-trained YOLOv3 model.

For every frame in case of videos/image in the YOLOv3, the bounding box coordinates of the objects were first extracted as shown in Figure 4.8. Based on the bounding box image coordinates of the objects, the distance from the drone to the object along with GPS coordinates of the object were estimated using the concept of photogrammetry (ground sample distance and haversine formula) for precise search operation of the humans in floods. Figure 4.7 shows the GPS coordinates of the object and distance estimation from the drone to the object obtained by us for every image in the YOLOv3 model.



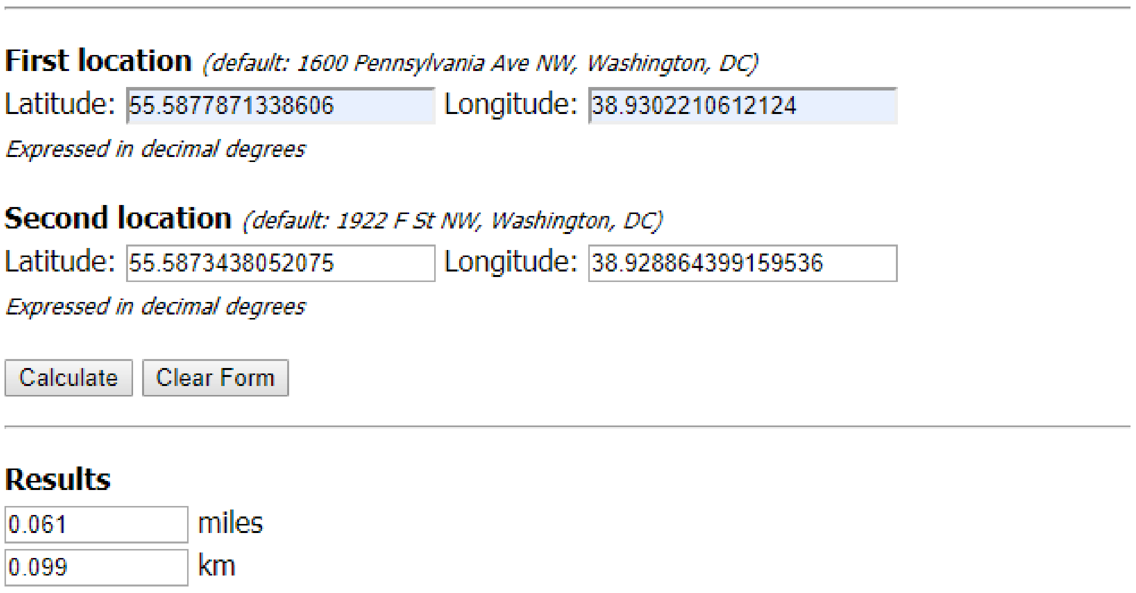
```
     Bounding Box: Left=0, Top=223, Right=91, Bottom=428
[26] person: 97%
     Bounding Box: Left=620, Top=260, Right=758, Bottom=436
     person: 97%
     Bounding Box: Left=477, Top=268, Right=546, Bottom=331
     person: 92%
     Bounding Box: Left=281, Top=138, Right=323, Bottom=214
     person: 89%
     Bounding Box: Left=151, Top=224, Right=242, Bottom=366
     person: 86%
     Bounding Box: Left=344, Top=181, Right=392, Bottom=245
     person: 83%
     Bounding Box: Left=298, Top=188, Right=344, Bottom=256
     person: 76%
     Bounding Box: Left=638, Top=232, Right=742, Bottom=333
     person: 69%
     Bounding Box: Left=213, Top=188, Right=244, Bottom=234
     person: 59%
     Bounding Box: Left=544, Top=254, Right=569, Bottom=295
     Unable to init server: Could not connect: Connection refused

     (predictions:2979): Gtk-WARNING **: 05:00:33.177: cannot open display:
             Inferrence time:  0:00:00.518133

     The number of people to be rescued are 13
     Latitude of the Drone 55.5877871338606
     Longitude of the Drone is 38.9302210612124
     Latitude of Object 55.5873438052075
     Longitude of Object 38.928864399159536
     Distance is 98.59028856839807
```

Figure 4.9. GPS estimation of objects using pre-trained YOLOv3 model

The distance between the GPS coordinates of the object and the GPS coordinates of the drone was further validated on the haversine calculator available on `https://andrew.hedges.name/experiments/haversine/` with GPS coordinates of the two points provided in the calculator as shown in Figure 4.8.



First location (default: 1600 Pennsylvania Ave NW, Washington, DC)
Latitude: 55.5877871338606 Longitude: 38.9302210612124
Expressed in decimal degrees

Second location (default: 1922 F St NW, Washington, DC)
Latitude: 55.5873438052075 Longitude: 38.928864399159536
Expressed in decimal degrees

Results
0.061 miles
0.099 km

Figure 4.10. Validation of distance obtained in Figure 4.8 using Haversine Formula

The experiment was conducted on a test set of 10 videos of 2 minutes each of resolution 1080p, 720p and 360p and on 200 images separately. On fitting the pre-trained YOLOv3 model on batches of 10 images for 200 images, it was found that the inference time taken by 200 images on google colaboratory varied between 39s to 34s with a median on 36.5s as depicted in boxplot in Figure 4.9.

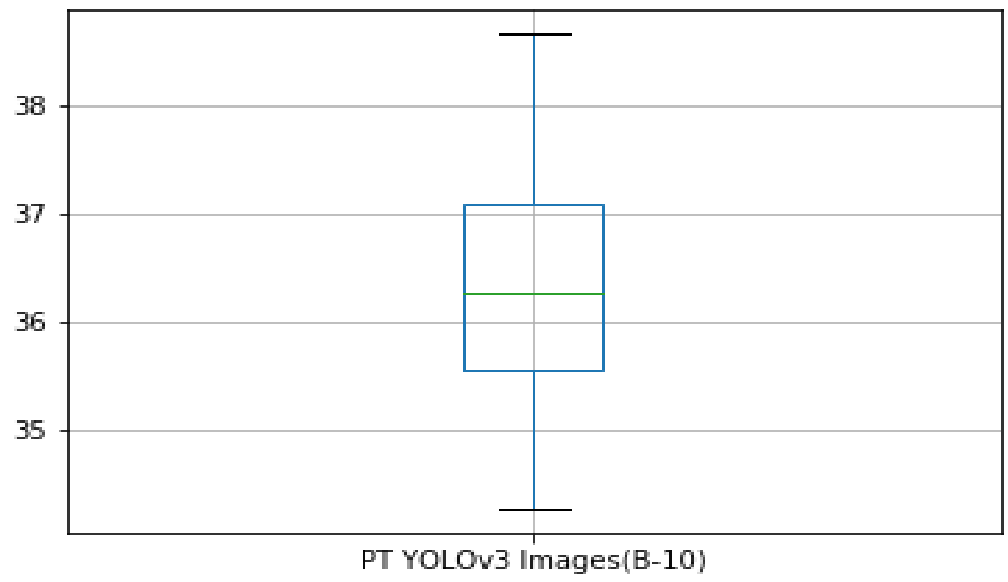


Figure 4.11. Inference time for pre-trained YOLOv3 model

The performance metric parameters, precision, and recall for the YOLOv3 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.5 for images. If the IoU was more than threshold, then a true positive was confirmed. If IoU was less than threshold, a false positive was confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed.

Table 4.5. Pre-trained YOLOv3 Confusion Matrix - Images

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 728$	$FN = 274$	$728 + 274$
	(0)	$FP = 60$	$TN = 261$	$60 + 261$
	Total	$728 + 60$	$274 + 261$	1323

4.5.0.1 Pre-trained YOLOv3 on videos of resolution 1080p

On fitting the pre-trained YOLOv3 model on videos of resolution 1080p it was found out that the inference time varied between 178s and 140s with a median on 170s and an outlier on 250s as depicted in the boxplot in Figure 4.10.

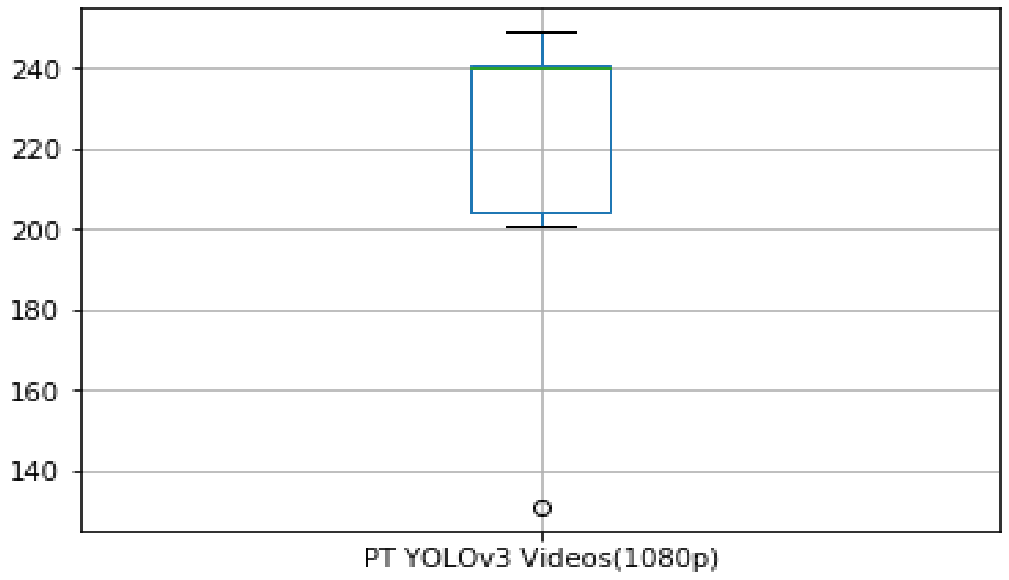


Figure 4.12. Inference time for pre-trained YOLOv3 model of resolution 1080p

The performance metric parameters, precision, and recall for the YOLOv3 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.2 for videos. If the IoU was more than threshold, then a true positive was confirmed. If IoU was less than threshold, a false positive was

Table 4.6. Pre-trained YOLOv3 Confusion Matrix - Videos of resolution 1080p

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 5095$	$FN = 2728$	$5095 + 2728$
	(0)	$FP = 583$	$TN = 1830$	$583 + 1830$
	Total	$5095 + 583$	$2728 + 1830$	10236

confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. For videos, as depicted in the confusion matrix, there were considerable number of false negatives and false positives which played a crucial role in determination of precision and recall values. The precision and recall curve for pretrained YOLOv3 has been depicted below.

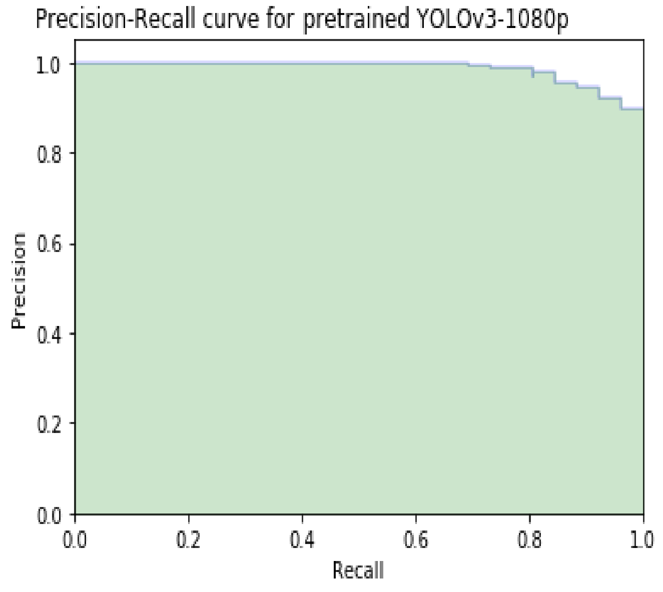


Figure 4.13. Precision Recall curve for pretrained YOLOv3-1080p

4.5.0.2 Pre-trained YOLOv3 on videos of resolution 720p

On fitting the pre-trained YOLOv3 model on videos of resolution 720p it was found out that the inference time varied between 200s and 240s with an outlier on 120s as depicted in the boxplot in Figure 4.11. The inference time of videos of resolution 720p was 4s less than the videos of resolution 1080p.

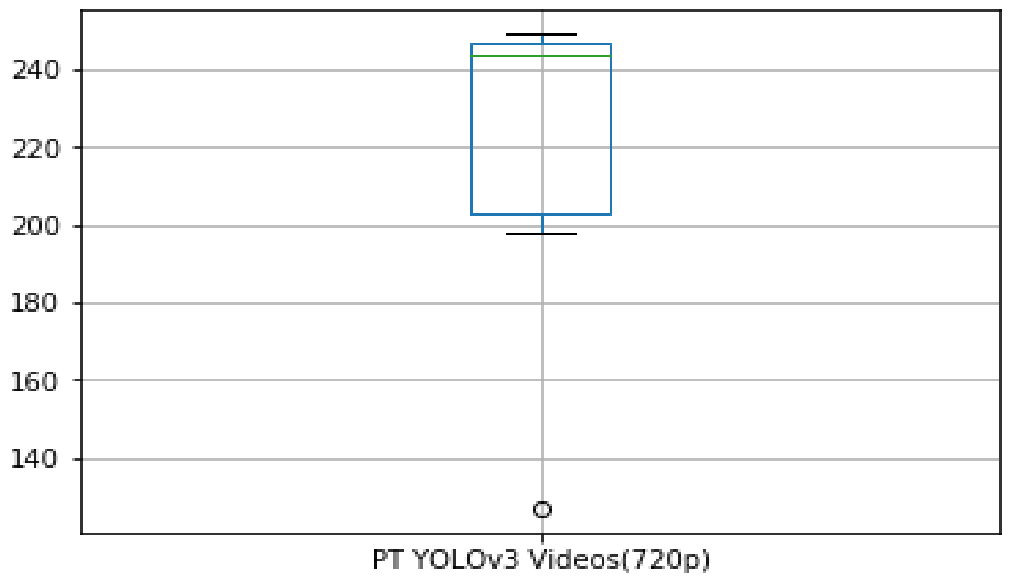


Figure 4.14. Inference time for pre-trained YOLOv3 model of resolution 720p

The performance metric parameters, precision, and recall for the YOLOv3 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.2 for videos. If the IoU was more than threshold, then a true positive was confirmed. If IoU was less than threshold, a false positive was confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. For videos, as depicted in

Table 4.7. Pre-trained YOLOv3 Confusion Matrix - Videos of resolution 720p

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 4823$	$FN = 2871$	$4823 + 2871$
	(0)	$FP = 621$	$TN = 1921$	$621 + 1921$
	Total	$4823 + 621$	$2871 + 1921$	10236

the confusion matrix, there were considerable number of false negatives and false positives which played a crucial role in determination of precision and recall values. The precision and recall curve for pre-trained YOLOv3 for videos of resolution 720p has been depicted below.

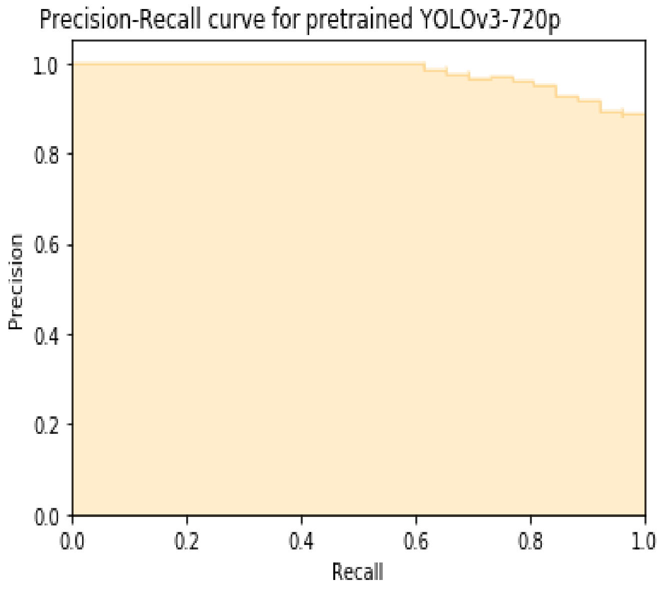


Figure 4.15. Precision Recall curve for pretrained YOLOv3-720p

4.5.0.3 Pre-trained YOLOv3 on videos of resolution 360p

On fitting the pre-trained YOLOv3 model on videos of resolution 360p it was found out that the inference time varied between 180s to 228s with an outlier at 110s as depicted in the boxplot in Figure 4.12. In terms of inference time, the videos of resolution 360p was faster (20s) than the videos of resolution 1080p.

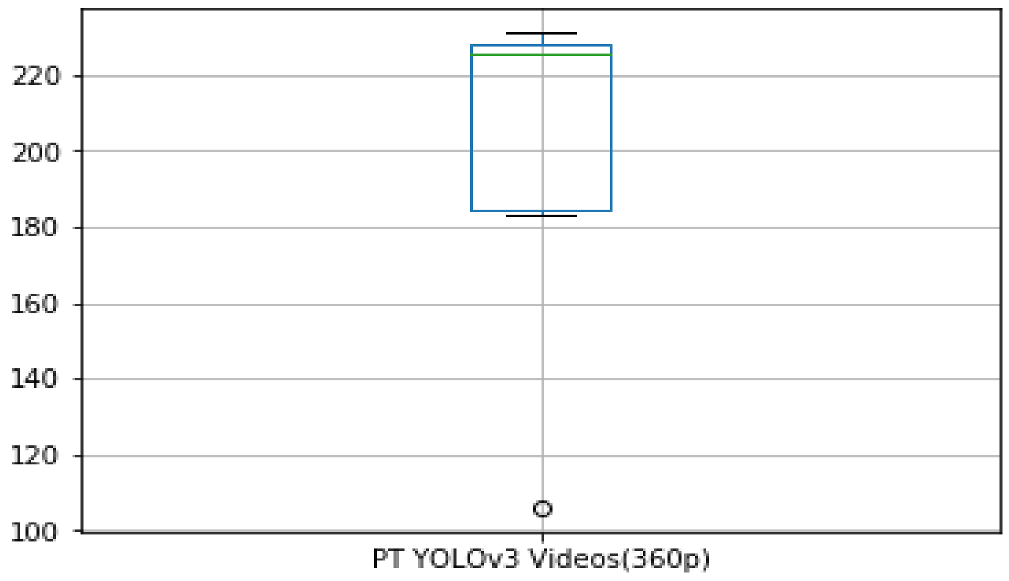


Figure 4.16. Inference time for pre-trained YOLOv3 model of resolution 360p

The performance metric parameters, precision, and recall for the YOLOv3 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.2 for videos. If the IoU was more than threshold, then a true positive was confirmed. If IoU was less than threshold, a false positive was confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. For videos, as depicted in

Table 4.8. Pre-trained YOLOv3 Confusion Matrix - Videos of resolution 360p

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 4517$	$FN = 2989$	$4517 + 2989$
	(0)	$FP = 1027$	$TN = 1709$	$1027 + 1709$
	Total	$4517 + 1027$	$2989 + 1709$	10236

the confusion matrix, there were considerable number of false negatives and false positives which played a crucial role in determination of precision and recall values. The precision and recall curve for pre-trained YOLOv3 for videos of resolution 360p has been depicted below.

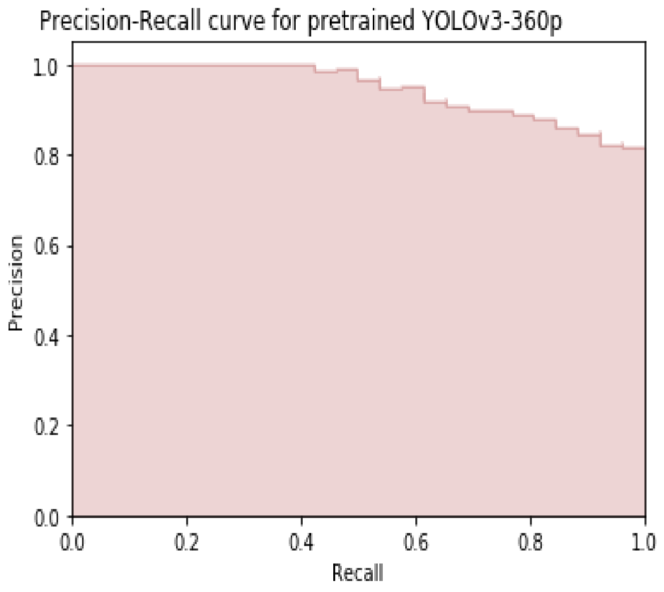
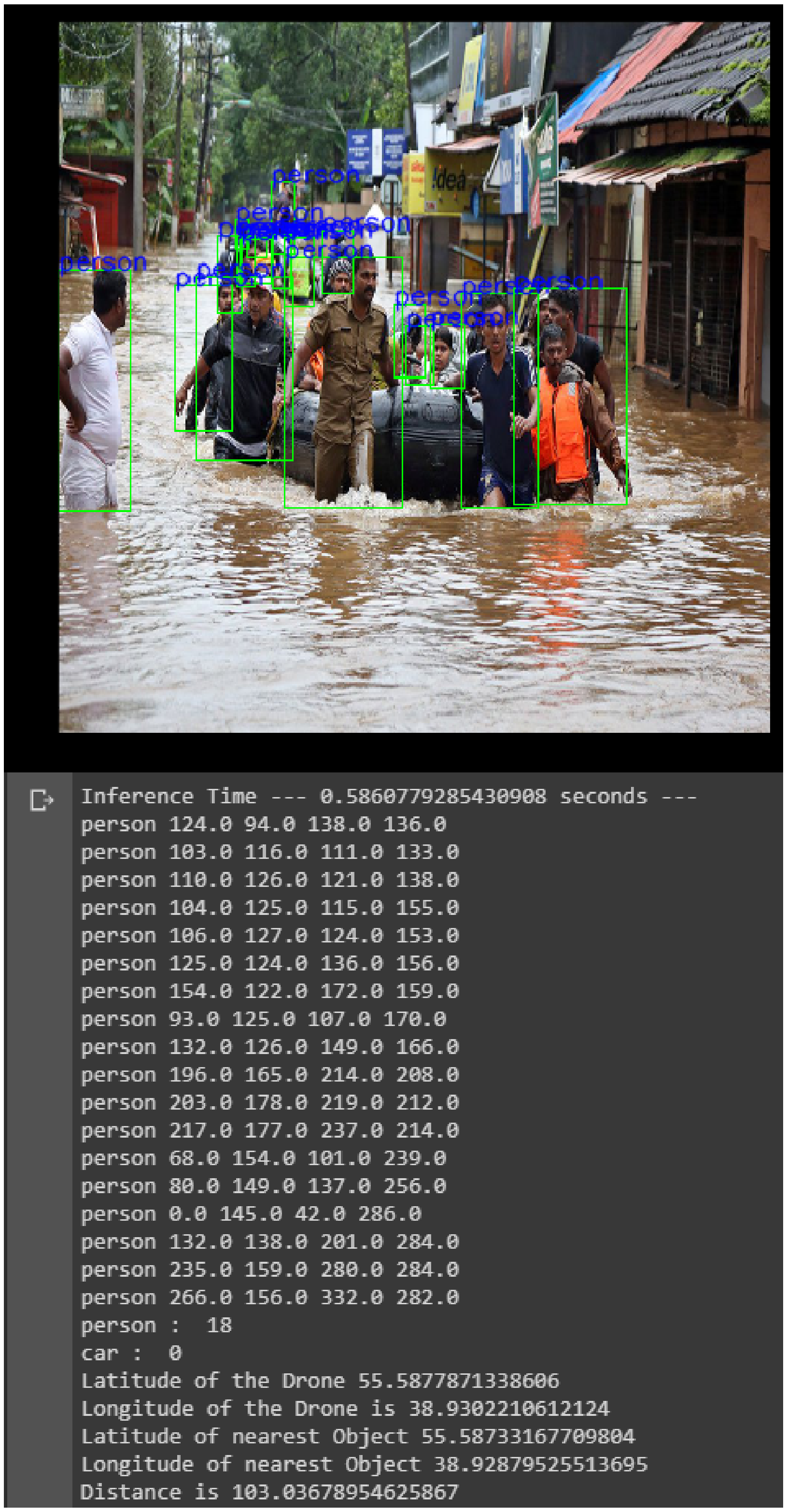


Figure 4.17. Precision Recall curve for pretrained YOLOv3-360p

4.6 Modified trained YOLOv3 model on 2 classes

The YOLOv3 was trained on a dataset of 5152 images downloaded from google. The VOC development kit dataset (4952 images) contained bounding box annotations of the objects (person, car). The dataset had 5227 instances of person and 1541 instances of car in 5152 images. The remaining 200 flood images were collected manually from google and annotated using LabelImg. The entire dataset was kept for training. Next, the Darknet-53 model was downloaded from `https://pjreddie.com/darknet/yolo/`. The configuration file was modified to be trained and tested for only 2 classes (person, car). The labels.txt file was modified in such a way that it could detect only two classes. The intent of compiling the model from scratch was to find a trade-off between performance metrics and inference time and compare these results with the pre-trained model for precise search and rescue operations.

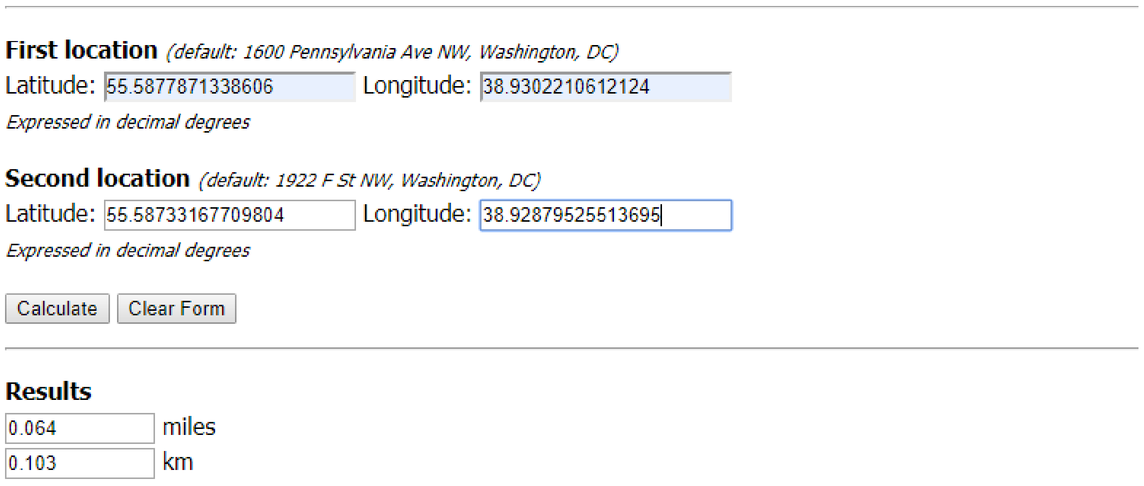
For every frame in case of videos/image in the YOLOv3, the bounding box coordinates of the objects were first extracted as shown in Figure 4.13. Based on the bounding box image coordinates of the objects, the distance from the drone to the object along with GPS coordinates of the object were estimated using the concept of photogrammetry (ground sample distance and haversine formula) for precise search and rescue operation of the humans in floods. Figure 4.13 shows the GPS coordinates of the object and distance estimation from the drone to the object obtained by us for every image in the YOLOv3 model.



```
Inference Time --- 0.5860779285430908 seconds ---
person 124.0 94.0 138.0 136.0
person 103.0 116.0 111.0 133.0
person 110.0 126.0 121.0 138.0
person 104.0 125.0 115.0 155.0
person 106.0 127.0 124.0 153.0
person 125.0 124.0 136.0 156.0
person 154.0 122.0 172.0 159.0
person 93.0 125.0 107.0 170.0
person 132.0 126.0 149.0 166.0
person 196.0 165.0 214.0 208.0
person 203.0 178.0 219.0 212.0
person 217.0 177.0 237.0 214.0
person 68.0 154.0 101.0 239.0
person 80.0 149.0 137.0 256.0
person 0.0 145.0 42.0 286.0
person 132.0 138.0 201.0 284.0
person 235.0 159.0 280.0 284.0
person 266.0 156.0 332.0 282.0
person :  18
car :  0
Latitude of the Drone 55.5877871338606
Longitude of the Drone is 38.9302210612124
Latitude of nearest Object 55.58733167709804
Longitude of nearest Object 38.92879525513695
Distance is 103.03678954625867
```

Figure 4.18. GPS estimation of objects using trained YOLOv3 model

The distance between the GPS coordinates of the object and the GPS coordinates of the drone was further validated on the haversine calculator available on `https://andrew.hedges.name/experiments/haversine/` with GPS coordinates of the two points provided in the calculator as shown in Figure 4.14.



First location *(default: 1600 Pennsylvania Ave NW, Washington, DC)*
Latitude: 55.5877871338606 Longitude: 38.9302210612124
Expressed in decimal degrees

Second location *(default: 1922 F St NW, Washington, DC)*
Latitude: 55.58733167709804 Longitude: 38.92879525513695
Expressed in decimal degrees

Results
0.064 miles
0.103 km

Figure 4.19. Validation of distance obtained in Figure 4.12 using Haversine Formula

The experiment was conducted on a test set of 10 videos of 2 minutes of resolution 1080p and on 200 images separately. On fitting the trained YOLOv3 model on batches of 10 images for 200 images, it was found that the inference time taken by 200 images on google colaboratory varied between 5s to 6.6s with a median on 5.8s. The inference time for trained YOLOv3 model was found out to be smaller than the inference time for the pre-trained model.

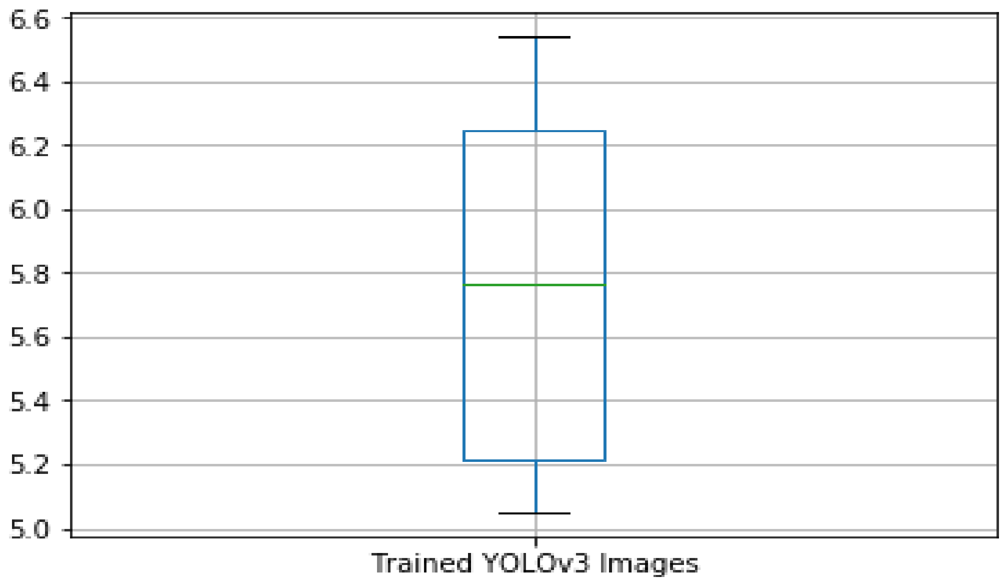


Figure 4.20. Inference time for trained YOLOv3 model

The performance metric parameters, precision, and recall for the YOLOv3 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.5 for image. If the IoU was more than a threshold, then a true positive was confirmed. If IoU was less than threshold, a false positive was confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. The precision and recall for trained YOLOv3 model for 200 images were found out to be 0.9800 and 0.94.

Table 4.9. Trained YOLOv3 Confusion Matrix - Images

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 736$	$FN = 40$	$736 + 40$
	(0)	$FP = 15$	$TN = 532$	$15 + 532$
	Total	$736 + 15$	$40 + 532$	1323

4.6.0.1 Modified trained YOLOv3 on videos of resolution 1080p On fitting the trained YOLOv3 model on videos of resolution 1080p it was found out that the inference time inference time varied between 200s and 240s with a median on 230s and an outlier on 130s as depicted in the boxplot in Figure 4.16.

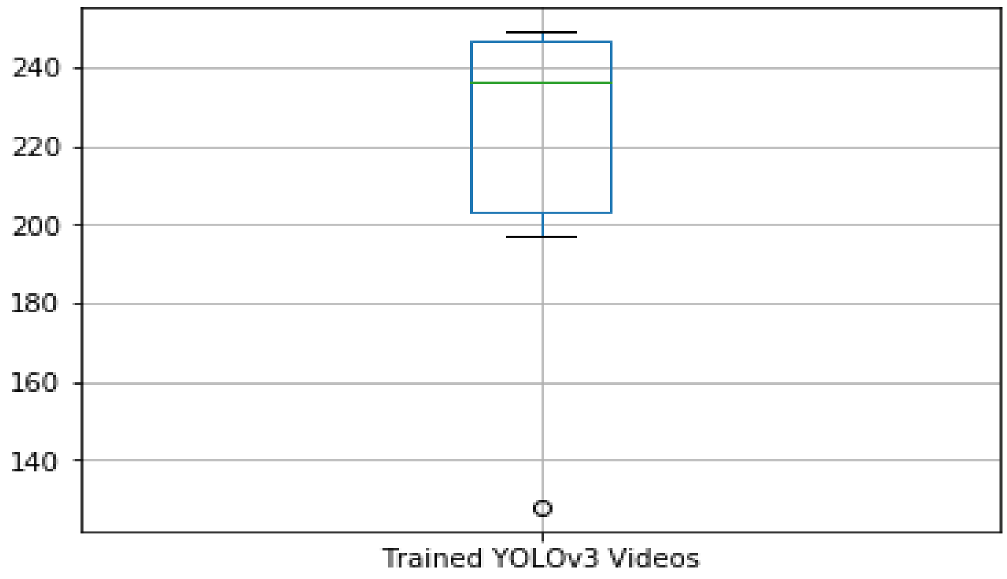


Figure 4.21. Inference time for trained YOLOv3 model of resolution 1080p

The performance metric parameters, precision, and recall for the YOLOv3 model were found based on the threshold set in the algorithm. For every object, the IoU (Intersection over Union) was computed and compared with the threshold value we set in the algorithm. The threshold was set at 0.2 for videos. If the IoU was more than threshold, then a true positive was confirmed. If IoU was less than threshold, a false positive was

Table 4.10. Trained YOLOv3 Confusion Matrix - Videos of resolution 1080p

		Predicted		
		(1)	(0)	Total
Actual	(1)	$TP = 4895$	$FN = 408$	$4895 + 408$
	(0)	$FP = 147$	$TN = 4786$	$583 + 1830$
	Total	$5095 + 583$	$2728 + 1830$	10236

confirmed. For no objects detected in the images/frames, a true negative was confirmed otherwise a false negative was confirmed. Once the TP, FP, FN, and TN was found out for the images and the videos, a confusion matrix was computed. For videos, as depicted in the confusion matrix, there were considerable number of false negatives and false positives which played a crucial role in determination of precision and recall values. The precision and recall for modified trained YOLOv3 for videos of resolution 1080p has been depicted below.

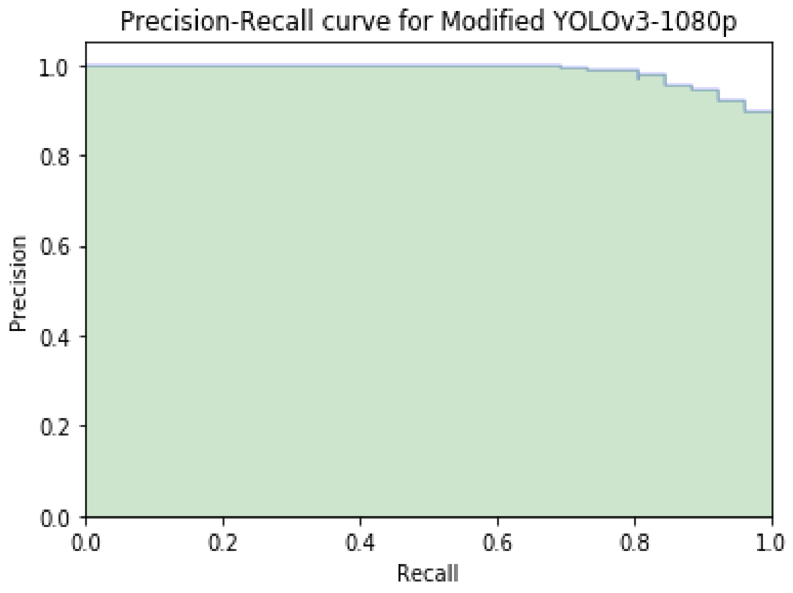


Figure 4.22. Precision Recall curve for modified YOLOv3-1080p

4.7 Results and Analysis

The original version of pre-trained models like Mask-RCNN, SSD300, YOLOv3 and the modified version of trained YOLOv3 were tested on 200 images and 10 videos of length 2 minutes each and the performance metrics and inference time were compared. From the values obtained on executing these algorithms it was found out that, the modified version of YOLOv3 outperformed in terms of performance metrics (precision and recall) but the inference time only improved slightly. The pre-trained YOLOv3 model was also tested on videos of different resolutions like 360p, 720p and 1080p. Based on the values obtained, it can be concluded that, the modified version of YOLOv3 had better model accuracy and very less loss as depicted in the graph below and thus could be used in real time for search operations as it out performed over the pre-trained models.

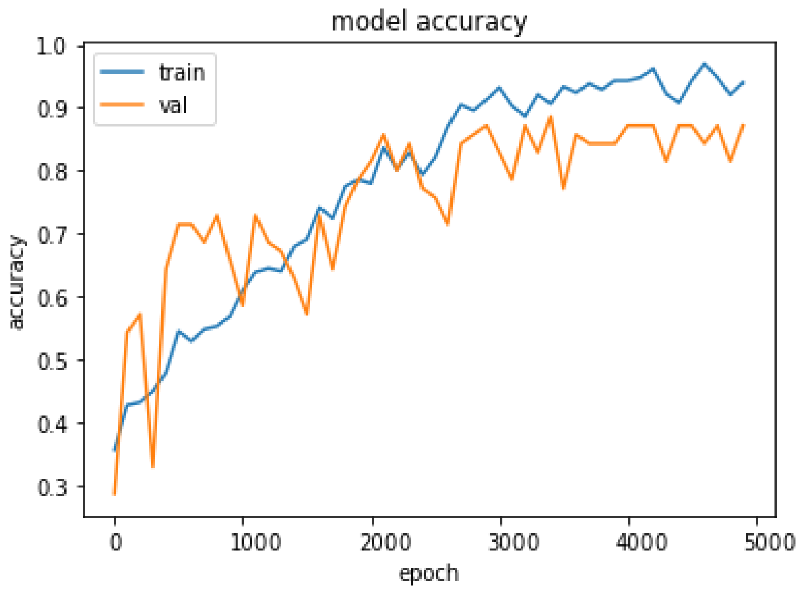


Figure 4.23. Accuracy of Modified YOLOv3-1080p

Table 4.11. Comparison of results

	Inference Time (sec)	Precision (max val:1)	Recall(max val:1)
PT Mask R-CNN-1080p	2780	0.7083	0.4294
PT SSD300-1080p	800	0.4172	0.1429
PT YOLOv3-1080p	240	0.8973	0.6512
PT YOLOv3-720p	242	0.8859	0.6268
PT YOLOv3-360p	222	0.8147	0.6017
Modified YOLOv3-1080p	238	0.9708	0.9230

*PT - Pretrained

4.8 Summary

Here, we first discussed the procedure of data collection and preparation followed by class annotation we used in our research. Later in the chapter, we have discussed and validated results of various deep learning models we obtained during our research on pre-trained models like Mask-RCNN, SSD300, YOLOv3 and modified version of trained YOLOv3 on parameters like inference time and performance metrics.

CHAPTER 5. CONCLUSION AND FUTURE PLAN

This chapter briefly explain the research conducted. It also concludes the results of the research and last includes relevant discussion and future direction.

5.1 Conclusion

The principal focus of the thesis was to modify the YOLOv3 algorithm to detect persons and cars for search operation in flood videos and compare it with pre-trained models like Mask-RCNN, SSD300, YOLOv3 for search operations in real time. Traditional means of search operations are time-intensive and can be a major bottleneck in search operations. This study focused to solve the issues of real time search operations using the concept of deep learning and photogrammetry.

Literature states that Mask-RCNN, SSD300, YOLOv3 algorithms are one of the faster algorithms and can detect objects in real time. Till today there has been plethora of work done on making these algorithms fast (in terms of inference time) and more accurate (in terms of performance metrics). One of the famous researches in the area of deep learning has been done by Redmon and Farhadi (2018). The researchers have in the past worked on deep learning models to detect multiple objects but none of them focused on detecting specific objects and measuring the performance of the algorithm based on these parameters. The thesis focuses on detection of specific objects (person, car) in a flood in real time with focus on algorithm parameters like inference time and performance metrics. Chapter 4 gives details on the results obtained by executing pre-trained models like Mask-RCNN, SSD300 and YOLOv3 and the modified YOLOv3 on the same set of inputs. The performance metrics like precision and recall and inference time were observed when these algorithms were executed. It was found out that the performance metrics (precision and recall) of the modified algorithm improved significantly while there was slight change in the inference time of the model thus concluding that it could be used to make search operations in real time in a flood.

5.2 Future Plan

A future direction to this study is to investigate deeper into why the inference time of the model didn't improve significantly and why the performance metrics of the algorithm was still failing to give impressive numbers. The current experimentation uses only 2D information for detection and segmentation. In the future, a study can be conducted where lidar data can be used in the modified algorithm for detection. The use of lidar data can only help improve the overall accuracy of the model by mapping more TP and TN over FP or FN. The research study even focuses on the inference time of the algorithm to make object detections. A study can be conducted to improve the inference time of the algorithm and ensuring that the performance metrics of the algorithm is not being hampered.

REFERENCES

Adit Deshpande. (2016). *Illustration of convolutional neural network.* Retrieved from `https://adeshpande3.github.io/A-Beginner-27s-Guide-To-Understanding-Convolutional-Neural-Networks/` ([Online; accessed July 20, 2016])

Al-Khudhairy, D. H. (2010). Geo-spatial information and technologies in support of eu crisis management. *International Journal of Digital Earth*, *3*(1), 16–30.

Andrey Kurenkov. (2015). *A fully connected network.* Retrieved from `https://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/` ([Online; accessed December 24, 2015])

Andriluka, M., Schnitzspan, P., Meyer, J., Kohlbrecher, S., Petersen, K., Von Stryk, O., ... Schiele, B. (2010). Vision based victim detection from unmanned aerial vehicles. In *2010 ieee/rsj international conference on intelligent robots and systems* (pp. 1740–1747).

Armin Wasicek. (2018). *Performance graphs of machine learning methods.* Retrieved from `https://www.sumologic.com/blog/machine-learning-deep-learning/` ([Online; accessed October 11, 2018])

Blaikie, P., Cannon, T., Davis, I., & Wisner, B. (2014). *At risk: natural hazards, people's vulnerability and disasters*. Routledge.

Blitch, J. G. (1996). Artificial intelligence technologies for robot assisted urban search and rescue. *Expert Systems with Applications*, *11*(2), 109–124.

Bubeck, P., Botzen, W. J., & Aerts, J. C. (2012). A review of risk perceptions and other factors that influence flood mitigation behavior. *Risk Analysis: An International Journal*, *32*(9), 1481–1495.

Calantropio, A., Chiabrando, F., Sammartano, G., Spanò, A., & Losè, L. T. (2018). Uav strategies validation and remote sensing data for damage assessment in post-disaster scenarios. In *International archives of the photogrammetry, remote sensing and spatial information sciences* (Vol. 42).

Celik, S., & Corbacioglu, S. (2010). Role of information in collective action in dynamic disaster environments. *Disasters*, *34*(1), 137–154.

Chanuk Lim. (2017). *Architecture of mask r-cnn.* Retrieved from `https://www.slideshare.net/windmdk/mask-rcnn` ([Online; accessed August 14, 2017])

Chen, Y., Lin, Z., Zhao, X., Wang, G., & Gu, Y. (2014). Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, *7*(6), 2094–2107.

Cireşan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745.*

Conor McDonald. (2017). *Cost function.* Retrieved from `https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220` ([Online; accessed November 27, 2017])

Corbane, C., Saito, K., Dell'Oro, L., Gill, S., Piard, B., & Huyck, C. (2010). & eguchi, r.(2011). *A comprehensive analysis of building damage in the January*, *12*, 997–1109.

Cummings, M. L. (2004). *Designing decision support systems for revolutionary command and control domains*. Unpublished doctoral dissertation, University of Virginia Charlottesville, VA.

de Moel, H., Jongman, B., Kreibich, H., Merz, B., Penning-Rowsell, E., & Ward, P. J. (2015). Flood risk assessments at different spatial scales. *Mitigation and Adaptation Strategies for Global Change*, *20*(6), 865–890.

Drone Blocks. (2013). *Haversine calculation.* Retrieved from `https://learn.droneblocks.io/courses/droneblocks-math-with-drones/lectures/2908697` ([Online; accessed December 23, 2013])

Erdelj, M., & Natalizio, E. (2016). Uav-assisted disaster management: Applications and open issues. In *2016 international conference on computing, networking and communications (icnc)* (pp. 1–5).

Ezequiel, C. A. F., Cua, M., Libatique, N. C., Tangonan, G. L., Alampay, R., Labuguen, R. T., ... others (2014). Uav aerial imaging applications for post-disaster assessment, environmental management and infrastructure development. In *2014 international conference on unmanned aircraft systems (icuas)* (pp. 274–283).

Fei-Fei Li. (2017). *Comparison of time speed of r-cnn.* Retrieved from `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf` ([Online; accessed May 10, 2017])

Feng, C.-C., & Wang, Y.-C. (2011). Giscience research challenges for emergency management in southeast asia. *Natural hazards*, *59*(1), 597.

Fornace, K. M., Drakeley, C. J., William, T., Espino, F., & Cox, J. (2014). Mapping infectious disease landscapes: unmanned aerial vehicles and epidemiology. *Trends in parasitology*, *30*(11), 514–519.

Giitsidis, T., Karakasis, E. G., Gasteratos, A., & Sirakoulis, G. C. (2015). Human and fire detection from high altitude uav images. In *2015 23rd euromicro international conference on parallel, distributed, and network-based processing* (pp. 309–315).

Guha-Sapir, D., Below, R., & Hoyois, P. (2016). Em-dat: the cred/ofda international disaster database.

Guha-Sapir, D., Vos, F., Below, R., & Ponserre, S. (2012). *Annual disaster statistical review 2011: the numbers and trends* (Tech. Rep.). Centre for Research on the Epidemiology of Disasters (CRED).

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, 27–48.

Haegeli, P., Falk, M., Brugger, H., Etter, H.-J., & Boyd, J. (2011). Comparison of avalanche survival patterns in canada and switzerland. *Cmaj*, *183*(7), 789–795.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 2961–2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*.

Ibrahim, M. S., Muralidharan, S., Deng, Z., Vahdat, A., & Mori, G. (2016). A hierarchical deep temporal model for group activity recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1971–1980).

Jain, A. K., Mao, J., & Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *Computer*(3), 31–44.

Jose Ariza. (2018). *Structure of perceptron.* Retrieved from `https://swapps.com/blog/introduction-to-machine-learning-and-pytorch/` ([Online; accessed November 17, 2018])

Joseph Redmon. (2016). *Architecture of yolo.* Retrieved from `https://arxiv.org/pdf/1506.02640.pdf` ([Online; accessed May 9, 2016])

Joseph Redmon. (2017a). *Backbone comparison.* Retrieved from `https://pjreddie.com/media/files/papers/YOLOv3.pdf` ([Online; accessed May 9, 2017])

Joseph Redmon. (2017b). *Inference time comparison.* Retrieved from `https://pjreddie.com/media/files/papers/YOLOv3.pdf` ([Online; accessed May 9, 2017])

Kaiming He. (2015). *Resnet.* Retrieved from `https://arxiv.org/pdf/1512.03385.pdf` ([Online; accessed December 10, 2015])

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Disaster monitoring using unmanned aerial vehicles and deep learning. *arXiv preprint arXiv:1807.11805*.

Kang, J., Gajera, K., Cohen, I., & Medioni, G. (2004). Detection and tracking of moving objects from overlapping eo and ir sensors. In *2004 conference on computer vision and pattern recognition workshop* (pp. 123–123).

Keck, M., & Davis, J. W. (2008). 3d occlusion recovery using few cameras. In *2008 ieee conference on computer vision and pattern recognition* (pp. 1–8).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, H., Zhao, R., & Wang, X. (2014). Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *arXiv preprint arXiv:1412.4526*.

Li, S., Tang, H., He, S., Shu, Y., Mao, T., Li, J., & Xu, Z. (2015). Unsupervised detection of earthquake-triggered roof-holes from uav images using joint color and shape features. *IEEE Geoscience and Remote Sensing Letters*, *12*(9), 1823–1827.

Lilian Weng. (2017). *Roi mapping.* Retrieved from `https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html` ([Online; accessed December 31, 2017])

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37).

Luo, C., Nightingale, J., Asemota, E., & Grecos, C. (2015). A uav-cloud system for disaster sensing applications. In *2015 ieee 81st vehicular technology conference (vtc spring)* (pp. 1–5).

Matese, A., Toscano, P., Di Gennaro, S., Genesio, L., Vaccari, F., Primicerio, J., … Gioli, B. (2015). Intercomparison of uav, aircraft and satellite remote sensing platforms for precision viticulture. *Remote Sensing*, *7*(3), 2971–2990.

Mills, C., & Navarro, M. (1995). Urban search and rescue task force overview. In *Proceedings of the 6th international symposium on rescue dogs.*

Munich, R. (2015). Natcatservice loss events worldwide 1980–2014. *Munich Reinsurance, Munich.*

Murphy, R. R. (2004). National science foundation summer field institute for rescue robots for research and response (r4). *AI Magazine*, *25*(2), 133–133.

Ofli, F., Meier, P., Imran, M., Castillo, C., Tuia, D., Rey, N., … others (2016). Combining human computing and machine learning to make sense of big (aerial) data for disaster response. *Big data*, *4*(1), 47–59.

Olsen Jr, D. R., & Wood, S. B. (2004). Fan-out: measuring human control of multiple robots. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 231–238).

Petrides, P., Kolios, P., Kyrkou, C., Theocharides, T., & Panayiotou, C. (2017). Disaster prevention and emergency response using unmanned aerial systems. In *Smart cities in the mediterranean* (pp. 379–403). Springer.

Pradhan, B. (2010). Flood susceptible mapping and risk area delineation using logistic regression, gis and remote sensing. *Journal of Spatial Hydrology*, *9*(2).

Pradhan, B., Hagemann, U., Tehrany, M. S., & Prechtel, N. (2014). An easy to use arcmap based texture analysis program for extraction of flooded areas from terrasar-x satellite image. *Computers & geosciences*, *63*, 34–43.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Rehor, M. (2007). Classification of building damage based on laser scanning data. *The Photogrammetric Journal of Finland*, *20*(2), 54–63.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

Rudol, P., & Doherty, P. (2008). Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. In *2008 ieee aerospace conference* (pp. 1–8).

Saxena, L., & Armstrong, L. (2014). A survey of image processing techniques for agriculture.

Scassellati, B., Admoni, H., & Matarić, M. (2012). Robots for use in autism research. *Annual review of biomedical engineering*, *14*, 275–294.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Schumann, G. J.-P., Neal, J. C., Mason, D. C., & Bates, P. D. (2011). The accuracy of sequential aerial photography and sar data for observing urban flood dynamics, a case study of the uk summer 2007 floods. *Remote Sensing of Environment*, *115*(10), 2536–2546.

Shaoqing Ren. (2016). *Architecture of faster r-cnn for object detection.* Retrieved from `https://arxiv.org/pdf/1506.01497.pdf` ([Online; accessed January 6, 2016])

Snyder, R. G. (2001). Robots assist in search and rescue efforts at wtc. *IEEE Robotics and Automation Magazine*, *8*(4), 26–28.

Sutanta, H., Bishop, I., & Rajabifard, A. (2010). Integrating spatial planning and disaster risk reduction at the local level in the context of spatially enabled government.

Tehrany, M. S., Pradhan, B., Mansor, S., & Ahmad, N. (2015). Flood susceptibility assessment using gis-based support vector machine model with different kernel types. *Catena*, *125*, 91–101.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 648–656).

Wei Liu. (2016). *Architecture of ssd.* Retrieved from `https://arxiv.org/pdf/1512.02325.pdf` ([Online; accessed December 29, 2016])

Wikipedia, the free encyclopedia. (2013). *Atomic force microscopy.* Retrieved from `https://en.wikipedia.org/wiki/File:Atomic_force_microscope_block_diagram.svg` ([Online; accessed April 27, 2013])

Wzorek, M., Conte, G., Rudol, P., Merz, T., Duranti, S., & Doherty, P. (2006). From motion planning to control-a navigation framework for an autonomous unmanned aerial vehicle. In *21th bristol uav systems conference.*

Zhang, C., & Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture*, *13*(6), 693–712.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, *14*(1), 35–62.