

ARTIFICIAL INTELLIGENCE FOR AERIAL INFORMATION RETRIEVAL AND
MAPPING IN NATURAL DISASTERS

A Dissertation

by

YALONG PI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,
Co-Chair of Committee,
Committee Members,

Head of Department,

Wei Yan
Amir H. Behzadan
Youngjib Ham
Michelle Meyer
Robert R. Warden

August 2020

Major Subject: Architecture

Copyright 2020 Yalong Pi

ABSTRACT

Disasters affect every aspect of society, causing significant losses and interruptions to our way of life. Timely and reliable disaster information retrieval and exchange is key to efficiently implementing disaster mitigation, preparedness, response, and recovery. While aerial surveys of disaster-affected areas is one of the most effective ways for disaster reconnaissance, they are still costly, slow, and resource-intensive. The research presented in this dissertation investigates the use of artificial intelligence (AI) to augment current capacities in aerial footage processing, object localization and mapping, and quantification of disaster damage. This framework can provide relatively fast and accurate disaster impact information to first responders, affected people, governments and authorities, non-governmental organizations (NGOs), and other stakeholders, ultimately improving the quality and timeliness of decisions made to increase disaster resiliency. To enable visual recognition of the extent of disaster damage, two fully annotated, multi-class video datasets, Volan2018 and Voaln2019, are created. Convolutional neural network (CNN) architectures including you-only-look-once (YOLO), RetinaNet, Mask-RCNN, and PSPNet which are pre-trained on COCO, VOC, and ImageNet datasets, are then trained and tested on both Volan2018 and Voaln2019 datasets. Several experiments including object detection, projection, mapping, and quantification are carried out. Key performance factors including CNN architecture, viewpoint altitude, pre-trained weights, data balance, projection mechanism, and object sizes are also investigated. Findings of this work are sought to complement current

practices in disaster response, while also laying the foundation for future work in the general area of human-AI partnership for the social good.

DEDICATION

This Dissertation is dedicated to my father, Yexiong Pi (1949-2015). In your memory.

ACKNOWLEDGEMENTS

First, I would like to thank my advisors Dr. Amir H. Behzadan and Dr. Wei Yan. Your guidance, knowledge, and support helped me in my research and in completing this Dissertation. Additionally, I would like to thank other members of my Ph.D. advisory committee, Dr. Youngjib Ham and Dr. Michelle Meyer. Your insightful suggestions helped me expand the perspective of this research. I would also like to thank my colleagues in the Connected Informatics and Built Environment Research (CIBER) Lab, Nipun Nath, Chih-Shen Cheng, Md Nazmus Sakib, and Bahareh Alizadeh. It was a pleasure to work with you all. Last but not least, I would like to thank Ellie, who is the cutest dog in the world, and her owner Brittany Garcia.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor Amir H. Behzadan (research advisor) and Professor Youngjib Ham of the Department of Construction Science, Professor Wei Yan of the Department of Architecture, and Professor Michelle Meyer of the Department of Landscape Architecture and Urban Planning .

All work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from the Department of Construction Science at Texas A&M University.

NOMENCLATURE

AI	Artificial Intelligence
ANN	Artificial Neural Networks
AP	Average Precision
ARC	American Red Cross
B	Boat
BR	Balance Ratio
C	Car
CDC	Centers for Disease Control
CE	Cross Entropy
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CRED	Center for Research on The Epidemiology Of Disasters
CV	Computer Vision
D	Debris
DB	Drone Balanced
DBT	Diversity Balancing Threshold
DL	Deep Learning
DOF	Degrees of Freedom
DR	Damaged Roof
DU	Drone Unbalanced

EM-MDAT	Emergency Events Database
FA	Flooded Area
FAA	Federal Aviation Administration
FCN	Fully Connected Network
FEMA	Federal Emergency Management Agency
FL	Focal Loss
FN	False Negative
FN	Frame Number
FNN	Feedforward Neural Network
FP	False Positive
FPN	Feature Pyramid Network
FPS	Frames per Second
FRCN	Region-Based Fully Convolutional Networks
GAN	Generative Adversarial Network
GCP	Ground Control Points
GDP	Gross Domestic Product
GIS	Geographic Information System
GPS	Global Position System
GPU	Graphics Processing Unit
HB	Helicopter Balanced
HU	Helicopter Unbalanced
IA	Individual Assistance

ILSVRC	Large Scale Visual Recognition Challenge
IMU	Inertial Measurement Unit
IN	Instance Number
IoU	Intersection Over Union
IPF	Instance per Frame
KCF	Kernelized Correlation Filter
KPH	Kilometers per Hour
LiDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
mAP	Mean Average Precision
MD	Mask Drone
MDA	Mask Drone Augmented
MH	Mask Helicopter
MHA	Mask Helicopter Augmented
ML	Machine Learning
MNN	Modular Neural Network
MOSSE	Minimum Output Sum of Squared Error
MPH	Miles per Hour
NGO	Non-Governmental Organization
P	People
P&C	Precision and Confidence

PA	Public Assistance
PDA	Preliminary Damage Assessment
PPM	Pyramid Pooling Module
PROC	Projection from Perspective to Orthogonal Based On Reference Objects' Coordinates
PROS	Projection from Perspective to Orthogonal Based On Reference Objects' Size
PVT	Position, Velocity, and Time
QBT	Quantity Balancing Threshold
R	Road
R-CNN	Region-Based CNN
RFID	Radio-Frequency Identification
RGB	Red, Green, And Blue
RNN	Recurrent Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
RTK	Real-Time Kinematic
SAR	Search and Rescue
SAIPE	Small Area Income and Poverty Estimates
SLAM	Simultaneous Localization and Mapping
SOM	Self-Organizing Map
SoVI	Social Vulnerability Index

SVM	Support Vector Machine
Texas CDBG	Texas Community Development Block Grant
TFR	Temporary Flight Restriction
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
UD	Undamaged Roof
UGC	User-Generated Content
UNISDR	United Nations Office for Disaster Risk Reduction
US&R	Urban Search & Rescue
USNG	U.S. National Grid
UTM	universal transverse Mercator
V	Vegetation
VGI	Volunteered Geographic Information
VOC	Visual Object Classes
YOLO	You-Only-Look-Once

TABLE OF CONTENTS

CHAPTER	Page
ABSTRACT	ii
DEDICATION	iv
ACKNOWLEDGEMENTS	v
CONTRIBUTORS AND FUNDING SOURCES.....	vi
NOMENCLATURE.....	vii
TABLE OF CONTENTS	xii
LIST OF FIGURES.....	xv
LIST OF TABLES	xix
1. INTRODUCTION.....	1
1.1. Disaster management	1
1.1.1. Disaster impact	1
1.1.2. Phases of disaster management	3
1.1.3. Disaster information	6
1.2. Artificial intelligence.....	12
1.2.1. Machine learning and neural network	12
1.2.2. Computer vision and CNN applications.....	15
1.3. Organization of this Dissertation.....	21
2. LITERATURE REVIEW	24
2.1. Hurricane impact	24
2.2. Big data, remote sensing, and disaster information retrieval	25
2.3. CNN architectures	30
2.4. CNN training datasets	32
2.5. GPS-denied object localization and mapping	33
2.6. Social vulnerability	35
3. METHODOLOGY	38
3.1. General methodology	38
3.2. Research challenges	39
3.2.1. Identifying objects of interest in disaster footage	39

3.2.2. Selecting the best CNN architecture	40
3.2.3. Factors influencing model performance	41
3.2.4. Quantifying disaster damage	41
4. BOUDNGING BOX OBJECT DETECTION*	43
4.1. Volan2018 data description.....	43
4.1.1. Data collection and annotation	43
4.1.2. Volan2018 data balancing	50
4.2. YOLO v2 architecture.....	54
4.3. Performance measurement	60
4.4. Experiments and results analysis.....	63
4.5. Summary	71
5. IMAGE SEMANTIC SEGMENTATION	72
5.1. Volan2019 data description.....	72
5.1.1. Data collection and annotation	72
5.1.2. Multiclass data augmentation	77
5.2. Mask object detection.....	84
5.2.1. Mask-RCNN architecture.....	84
5.2.2. Performance measurement	85
5.2.3. Experiments and results analysis.....	87
5.3. Autoencoder segmentation.....	98
5.3.1. PSPNet architecture.....	98
5.3.2. Performance measurement	100
5.3.3. Experiments and results analysis.....	101
5.4. Summary	106
6. PERSPECTIVE TO ORTHOGONAL MAPPING.....	108
6.1. Homography transformation	108
6.2. Moving object projection	111
6.2.1. Experiments.....	111
6.2.2. Data collection and description	114
6.2.3. Projection Based on Reference Objects' Coordinates (PROC).....	117
6.2.4. Projection Based on Reference Objects' Size (PROS)	119
6.2.5. Results and analysis.....	120
6.3. Object tracking and counting	123
6.3.1. CNN-based centroid tracker	123
6.3.2. IoU-based ID assignment using Hungarian algorithm	126
6.3.3. Experiments and results.....	128
6.4. CNN integration and mapping	132
6.4.1. Static frame projection	132
6.4.2. Reference points updating and calibration	135

6.4.3. Experiments and results.....	137
6.5. Summary	139
7. CONCLUSION AND FUTURE WORK.....	142
7.1. Findings and Contributions	142
7.1.1. Disaster-domain multi-class visual datasets.....	143
7.1.2. CNN model selection and influencing factors	143
7.1.3. GPS-denied localization and mapping	145
7.1.4. Social vulnerability indicators.....	146
7.2. Future work	146
REFERENCES.....	150
APPENDIX A. YOLO MODEL PERFORMANCE	180
APPENDIX B. MASK MODEL REGRESSION	192

LIST OF FIGURES

	Page
Figure 1. Disaster management phases.	4
Figure 2. Disaster information flow.	10
Figure 3. Artificial neural networks (ANNs) structure.	13
Figure 4. CNN model outputs: classification, classification and localization, object detection, and instance segmentation.	16
Figure 5. Convolution computation example.	17
Figure 6. Convolutional neural network (CNN) structure.	19
Figure 7. Dissertation methodology.	38
Figure 8. Examples of Volan2018 annotation examples [173].	45
Figure 9. Volan001-006 instance distribution [173].	48
Figure 10. Volan2018 data splitting.	54
Figure 11. YOLO v2 architecture [173].	56
Figure 12. Transfer learning [173].	59
Figure 13. Union (IoU) example [173].	60
Figure 14. TP, FP, and FN prediction examples [173].	62
Figure 15. Precision-recall curve interpretation [173].	63
Figure 16. Precision, recall, and F1 score Model 2 tested on Volan007 for (a) debris, and (b) undamaged roof classes [173].	64
Figure 17. Model2 detection examples [173].	66
Figure 18. Model2 testing results on DB, DU, HB, HU, Volan007, and Volan008 subsets [173].	67
Figure 19. Sample annotations of classes roof (damaged and undamaged), vegetation, car, flooded area, and road in Volan2019 dataset.	75

Figure 20. Volan2019 instance and class distribution per video frame.	76
Figure 21. Selecting video frame candidates for data augmentation based on balance ratio.	78
Figure 22. Examples of image augmentation operations (from left to right: original image, original annotation, annotation overlaid on original image, augmented image, augmented annotation, augmented annotation overlaid on augmented image).	79
Figure 23. Volan2019 augmentation based on BR values.	80
Figure 24. Volan2019 data splitting based on altitude and augmentation based on BR value.	81
Figure 25. Mask-RCNN architecture.	85
Figure 26. Pixel intersection and union example.	86
Figure 27. Pixel level examples of TP, FP, and FN.	87
Figure 28. Sample detection output of Mask1 tested on Volan2019 dataset.	88
Figure 29. Precision-confidence correlation in Mask1 for class undamaged roof (UR) using (a) linear, and (b) polynomial regression.	89
Figure 30. Mask1 detection error matrix.	91
Figure 31. Example of determining overlapping detection based on pixel-level IoU.	92
Figure 32. Mask1 error percentage matrix to identify highly mismatched classes.	94
Figure 33. PSPNet architecture.	100
Figure 34. TP, FP, and TN cases in semantic segmentation.	101
Figure 35. Sample semantic segmentation output of PSPNet1 tested on P2019 dataset.	102
Figure 36. P2019 data augmentation based on balance ratio, and data separation based on altitude.	103
Figure 37. Perspective to orthogonal transformation.	110
Figure 38. PROC and PROS experiments.	113

Figure 39. Orthogonal video 1 (OV1) of experiment 1.....	115
Figure 40. Perspective video 1 (PV1) of experiment 1.	115
Figure 41. Orthogonal video 2 (OV2) of experiment 2.....	116
Figure 42. Perspective video 2 (PV2) of experiment 2.	116
Figure 43. Example of detections of person and cones.	118
Figure 44. Perspective to orthogonal projection.	119
Figure 45. Projection error in X- and Y- axis for PROC and PROS.....	122
Figure 46. Detections, centroids, and distances in event I.	125
Figure 47. Detection, centroids, and distances in event II.	126
Figure 48. Hungarian cost matrix.....	128
Figure 49. Face tracking and counting.	129
Figure 50. Person blocks the cone in PV2.	130
Figure 51. Roof counting based on YOLO model3.	130
Figure 52. Examples of projection input from Volan003 video.	133
Figure 53. Comparison of object projection results.	134
Figure 54. Reference points updating for homography transformation.	136
Figure 55. Example of reference point calibration.....	137
Figure 56. CNN combination and projection.	138
Figure 57. Nonlinearity of homography transformation for close and distant detections	139
Figure 58. Model 1-8 testing results on UB, UD, HB, HU, Volan007 and 008.	180
Figure 59. Mask1 confidence-precision linear regression.	192
Figure 60. Mask1 confidence-precision nonlinear regression (degree 2).	195
Figure 61. Mask1 confidence-precision nonlinear regression (degree 3).	197

Figure 62. Mask1 confidence-precision nonlinear regression (degree 4).200

LIST OF TABLES

	Page
Table 1. Numbers of disasters per type (1998-2017) [6].	2
Table 2. Top 10 disasters for absolute losses 1998-2017 [6].	3
Table 3. Information and work examples for public assistance (PA) [23].	8
Table 4. Individual assistance (IA) information category and detail [23].	9
Table 5. Summary of CNN applications in different domains.	20
Table 6. Saffir-Simpson Hurricane Scale.	24
Table 7. Current aerial disaster detection examples.	28
Table 8. Disaster object classes and applications.	40
Table 9. CNNs test Average precision (AP) and processing time (millisecond) on COCO test-dev [80].	41
Table 10. Volan2018 video name, event location, duration and instance amount per class [173].	46
Table 11. Volan2018 dataset instance per frame (IPF) [173].	50
Table 12. Data balancing results for each video within Volan2018 dataset [173].	53
Table 13. Pre-trained weights, training, and validation combinations of 8 YOLO v2 CNN models [173].	60
Table 14. 8 model mAP (%) tested on 8 testing subsets (* denotes the best performance in each subset) [173].	66
Table 15. Statistical analysis of the influence of viewpoint altitude on model performance ($p < 0.01$) [173].	69
Table 16. Performance and training time analysis [173].	70
Table 17. Description of Volan2019 dataset videos and number of annotated frames (D: UAV, H: Helicopter).	73
Table 18. Number of instances per class in training data and balance ratio reduction for Volan2019 dataset.	83

Table 19. Mask1 average error (%) for different degrees of precision-confidence regression.	89
Table 20. Precision-confidence correlation analysis for different models using linear regression.	90
Table 21. Summary of AP (%) and mAP (%) of Mask-RCNN models and their variations.	96
Table 22. Performance comparison of CNN models for similar classes to Volan2019 dataset.	96
Table 23. Effect of mask size on Mask1 model performance.	98
Table 24. Summary of AP (%), mIoU (%), and accuracy (%) of PSPNet models and their variations.	105
Table 25. Performance (%) of U-Net, SegNet, and PSPNet for similar classes to P2019.	106
Table 26. Data statistics for experiments 1 and 2.	117
Table 27. Comparison of Model-P and Model-O performance.	120
Table 28. Test results of PROC and PROS mapping methods.	122
Table 29. Centroid tracking and IoU based Hungarian tracking tests.	131

1. INTRODUCTION

This Chapter presents an overview of key concepts in disaster management, and reviews the extent of disaster impact and the types of information needed to respond to and recover from natural disasters. Next, various computer vision algorithms together with example applications are introduced. Finally, a short description of the organization of forthcoming Chapters is provided.

1.1. Disaster management

1.1.1. Disaster impact

The term “disaster” refers to a sudden (unexpected) event that results in property damage, human injury or death, and economic loss [1]. A natural disaster is caused by geophysical, meteorological, and hydrological changes which in turn impact humans and the society. Different types of natural disasters include but are not limited to earthquake, volcanic eruption, storm, extreme heat or cold, flood, landslide, drought, and wildfire. Beside natural disasters, there are technological and biological disasters. Examples of technological disasters are industrial (e.g., chemical spill, explosion, gas leak, oil spill) [2] and transportation (e.g., plane, car, and train crashes) accidents [3]. Biological disasters include epidemic diseases, insect infestation, and animal accidents [4].

In the year of 2018 alone, the Center for Research on the Epidemiology of Disasters (CRED) reported 315 natural disasters worldwide which affected 68.5 million people, claimed 11,840 lives and cost \$132 billion in economic damage [5]. The United Nations Office for Disaster Risk Reduction (UNISDR) compiled an emergency events database (EM-MDAT) for the period of 1998-2017, as summarized in Table 1 [6].

According to this report, the economic loss of disasters that occurred in this 20-year period was \$2,908 billion, and approximately 1.3 million people lost their lives to disasters. In the coming years and decades, it is expected that the number and frequency of disasters by on the rise around the world [7]. According to the EM-MDAT 1998-2017 in Table 1, flood (3,148 times) and storm (2,049 times) are the most frequent natural disasters, followed by earthquake (563 times) and extreme temperature (405 times).

Table 1. Numbers of disasters per type (1998-2017) [6].

Disaster type	Occurrence	Percentage (%)
Flood	3,148	43.39
Storm	2,049	28.24
Earthquake	563	7.76
Extreme temperature	405	5.58
Landslide	378	5.21
Drought	347	4.78
Wildfire	254	3.50
Volcanic activity	99	1.36
Mass movement (dry)	12	0.17

As shown in Table 2, the most costly types of disasters are water-related events such as floods and storms, according to EM-MDAT. In particular, hurricanes Harvey, Irma, and Maria in 2017 are the second to fourth most costly disasters in the last two decades with a total loss of \$2,455 billion. Although the absolute economic loss of such large-scale disasters is difficult to quantify due to cascading impact on people’s lives, jobs, education, and health, numbers show that low-income communities around the world suffer disproportionately in natural disasters. During the period from 1998-2017 [6], high-income countries reported disaster related losses of 0.41% of their annual

gross domestic product (GDP), compared to disaster losses recorded in low-income regions reached average 1.8% of their GDP.

Table 2. Top 10 disasters for absolute losses 1998-2017 [6].

Disaster	Date	Regions affected	Total damage (billion \$)
Hurricane Katrina	September, 2005	U.S.	156.3
Hurricane Harvey	August, 2017	U.S.	95.0
Hurricane Irma	September, 2017	U.S., Caribbean islands	80.8
Hurricane Maria	September, 2017	U.S., Caribbean islands	69.7
Hurricane Sandy	October, 2012	U.S., Caribbean islands	53.5
Flood	July-August, 1998	China	44.9
Flood	August, 2011-Janurary, 2012	Thailand	43.4
Hurricane Ike	September, 2008	U.S., Caribbean islands	36.3
Hurricane Ivan	September, 2004	U.S., Caribbean islands, Venezuela	29.9
Hurricane Wilma	October, 2005	U.S., Mexico, Belize, Honduras, Caribbean islands	25.0

1.1.2. Phases of disaster management

Disaster management refers to a series of activities that organize, deploy, and manage available resources to reduce disaster impact [8]. Typically, disaster management consists of four interconnected and collaborative phases of mitigation, preparedness, response, and recovery, as shown in Figure 1, which may involve ordinary citizens and communities, first responders, urban planners, building designers, local and

federal governments, non-governmental organizations (NGOs), insurance firms, and law enforcement agencies [9].

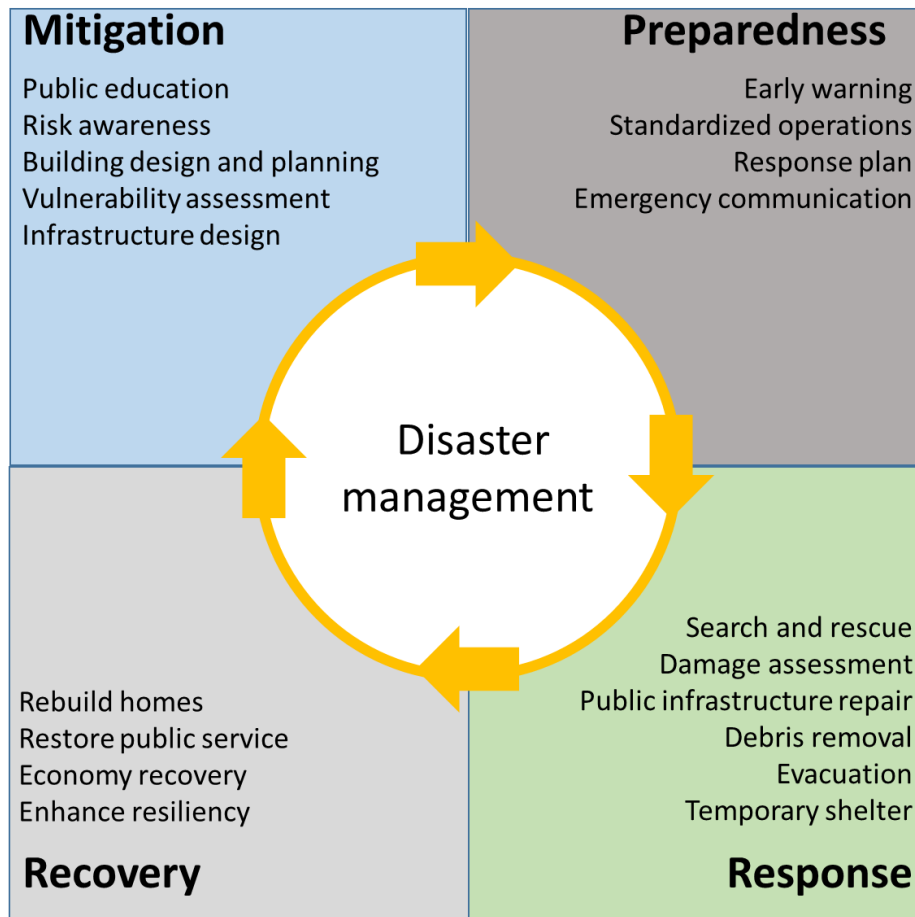


Figure 1. Disaster management phases.

The goal of mitigation and preparedness phases is to ensure physical and social robustness through effective urban planning, establishing building codes, disaster education, disaster prediction system, and environmental protection. Different countries and jurisdictions enforce building codes tailored to their specific geographical region, performance specifications, and building materials, to ensure buildings and facilities can

withstand disaster loads. In Japan, for example, the government has established a nationwide public disaster education system to improve the resiliency of the country as a whole [10]. In the U.S., states and local governments utilize the guidelines of the Federal Emergency Management Agency (FEMA) [11] to design and implement their own mitigation and preparedness plans to cater to the needs and overall wellbeing of their constituents. For example, the Texas Community Development Block Grant (CDBG) action plan [12] defines procedures, cost assessment, and funding resources to cope frequent disaster types such as storm and drought. Florida State mitigation plan regulates the plan maintenance, risk assessment, and funding to address local disasters [13]. In addition to disaster response strategies, researchers are also working on technologies to predict disasters in advance to help people evacuate in a timely and orderly manner. For example, China is building a massive typhoon alarm system to protect the residents of the southern sea region [14]. Similarly, Japan built early warning systems for tsunamis and earthquakes, in order to mitigate those frequent local disasters [15]. In Europe, researchers have developed seasonal water related early warning system to detect water flooding, debris flow, mud flow, and landslides [16].

In the response phase, depending on the severity of the disaster, neighborhoods and local government, state government, FEMA, and NGOs provide assistance to the victims. Such assistance can be in the form of providing search and rescue (SAR) specialties, temporary shelters, transportation, utilities, medical care, financial support, and insurance coverage. Following reconnaissance missions in affected areas, the type

and amount of assistance will be determined and coordinated. For example, an area which has lost gas stations may be in need of immediate gasoline supply.

The last phase of disaster management is the recovery which aims at helping the affected communities and local economy to bounce back to normal conditions. This can be achieved by helping victims rebuild their homes and businesses, family structure, social network, public services, recover from injuries, minimize psychological influences, and restore the economy. The rebuild process also serves to enhance the future resiliency. During this long-term endeavor, it is critical to pay special attention to social equity given the disproportionate extent of damage to vulnerable communities and underserved populations. If done properly and effectively, by the end of the recovery period, communities and local jurisdictions should be sufficiently prepared for the next disaster event.

1.1.3. Disaster information

Among the many factors that affect the success of disaster management, timely access to accurate data describing the severity, quantity, and location of the damage with sufficiently high spatiotemporal resolution is of utmost importance [17, 18]. Such information can significantly improve the quality and timeliness of related decisions with respect to SAR operations [19], evacuation [20], infrastructure repair, zoning improvement, building code amendment, public disaster education [21], debris cleanup, victim resettling, and processing insurance claims [22].

According to FEMA [23] and the American Red Cross (ARC) [24], local or county assessment is the most critical step for both individual assistance (IA) and public

assistance (PA). The information submitted by local or county emergency management needs to be verified by the state or tribal government through preliminary damage assessment (PDA). If the damage exceeds a certain amount, further assistance will be provided by FEMA (Stafford Act declaration [25]). Thereby, damage assessment conducted by local authorities needs to be prompt, accurate, efficient, and consistent.

In addition to IA and PA, the outcome of PDA may also directly inform short-term disaster response, according to FEMA National Urban Search & Rescue (US&R) Response System rescue field operations guide [26]. Information used for US&R range from location and type of impacted areas, people, and debris, as well as building configuration and collapse type. Table 3 lists work types and relevant information for PA operations. For instance, in debris removal, the assessment information includes road location, debris location and amount, debris type (e.g., vegetation, concrete, or steel), and removal cost per cubic yard.

Table 3. Information and work examples for public assistance (PA) [23].

Work type	Work example	Information type
Debris removal	Removal of debris from public or private property and right-of-ways	Road and debris location, debris amount and type, cost
Emergency protective measures	Protecting public health and safety, and property	Infrastructure cracks, failures, or unstable working statues, cost
Roads and bridges	Restoring roads (paved, gravel, and dirt), bridges, and their components	Surface, bases, shoulder, and ditch working conditions, cost
Water control facilities	Flood control, maintenance of fish and wildlife, storm water management	Channel alignment, flood location and amount, cost
Buildings and equipment	Restoring damaged buildings and equipment	Building and equipment working conditions and performance, cost
Utilities	Replacing water storage facilities, treatment plants, and delivery systems, natural gas transmission and distribution facilities	damaged system components, cost
Parks, recreation facilities, and other	Restoring facility needed for erosion control, replanting trees, shrubs, and other vegetation	Location of damaged trees, required labor, cost

Similarly, Table 4 shows information needs for IA operations. According to this Table, damaged homes are counted in clusters (5-100 homes per cluster) with occupancy status, household insurance, cause of damage, and damage degree (destroyed, major, minor, or inaccessible). Notably, homes located on flooded roads are counted as inaccessible, making it necessary to reflect the location of flooded areas and roads in those assessments. Altogether, this information is used to determine if disaster survivors are eligible for FEMA or state assistance. Moreover, insurance companies and NGOs deliver their aid packages based on this information.

Table 4. Individual assistance (IA) information category and detail [23].

Information category	Information detail
Concentration of damage	Damaged house amount by cluster (5 to 100 homes)
Ownership status	single-family residence, multi-family residence, manufactured homes
Occupancy status	Owner, renter
Households with insurance coverage	Homeowners insurance, flood insurance, renters insurance,
Number of homes impacted and degree of damage	Destroyed, major, minor, and inaccessible
Estimated cost of assistance	Uninsured damage to homes
Impact to populations with greater need	The percentage of the population living under poverty thresholds, or 65 years and older
Mass care/emergency assistance-feeding operations information	Number and location of fixed feeding sites (by county), number of vehicles providing mobile feeding
Emergency sheltering information	Shelter with family, dorms, ships, tents

Figure 2 illustrates the traditional information flow in disaster reconnaissance. As shown in this Figure, first responders investigate the damage and status of victims, and implement the most effective rescue plan. Meanwhile, they report the situation to the jurisdiction coordinator and other decision-makers. Based on the available resources, the coordinator either dispenses resources to the victims or reach out for more resources. In the meantime, providers exchange resource information with the responders to better meet the requirement. Lastly, disaster-related information is shared with the public, media, and other stakeholders. In this time-compressed period [27], any latency in information collection and exchange process can lead to drastic delays in delivering assistance. In addition, this process is prone to social inequality due to the presence and influence of social stereotypes and bureaucracy, and the fact that affected communities

are mostly on the receiving end of assistance, rather than being actively involved in the process.

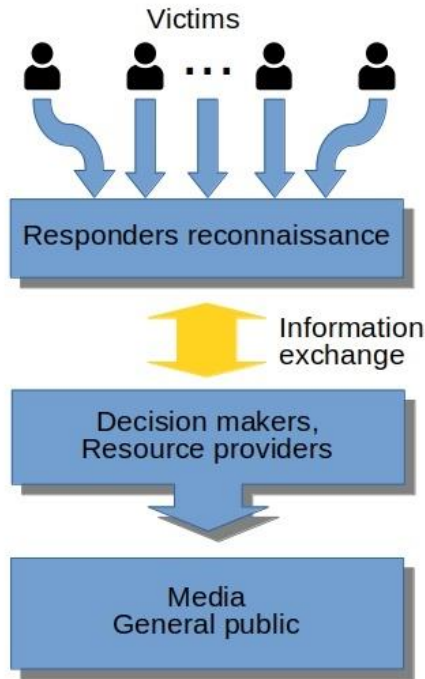


Figure 2. Disaster information flow.

Current PDA methods consist of self-reporting, fly-over, windshield surveys, and door-to-door site assessment. While aerial reconnaissance is more suitable when damage is visible from the air, follow-up ground surveys may be needed to sample and verify the results. Collected data is compiled and presented in a format interpretable by geographic information system (GIS)-based computer tools that use the U.S. National Grid (USNG) for damage quantification, mapping, and analysis. However, this type of assessment requires analysts to manually process a large volume of collected heterogeneous data from many sources.

Fly-over survey (aerial reconnaissance) is one of the most effective methods for locating victims, assessing the overall damage, and documenting the progress of recovery efforts, especially in the aftermath of large-scale natural disasters such as hurricanes, tornados, earthquakes, and wildfires. Traditionally, such reconnaissance is performed using helicopters to observe and record the impact across a large area [28]. In recent years, the advent of unmanned aerial vehicles (UAVs; a.k.a., drones) has created new opportunities to adopt this technology for aerial data collection mainly due to its flexibility, lower cost, and ease of use [29]. According to the U.S. Federal Aviation Administration (FAA) estimation, by the year 2022, there will be more than 2.4 million drones in the U.S. [30], which equates to 1 out of every 140 Americans flying a drone by that year. As FAA safety regulations concerning among other, temporary flight restriction (TFR), are being amended and evolve over time, it is not hard to imagine that the widespread ownership and use of drones can lead to the generation of large volumes of volunteered geographic information (VGI) [31] in natural disasters and during the recovery period. Having said that, regardless of the type of aerial data acquisition platform (helicopter or drone), the current practice is heavily dependent on skilled personnel, heavy manual effort for data cleaning and processing, and expensive equipment for data storage and exchange [32]. In natural disasters, when municipalities and first responders are overwhelmed with limited personnel and large quantities of work to be done, such limitations can hinder their ability to take timely actions based on appropriate data and resulting information, and deliver necessary resources to disaster-affected areas.

In this Dissertation, informed by the limitations of the current practice in disaster management, and the need for timely collection and exchange of disaster information particularly in large-scale disasters, aerial data collection and artificial intelligence (AI) in the form of computer vision are used to extract and map the location and extent of damage to ground objects. Combining the outcome of this work with the promise of widespread crowdsourcing applications in the future, it is envisioned that in the long-term, ordinary citizens will be more meaningfully involved in all phases of disaster management, leading to more transparency, and better social equity and fairness.

1.2. Artificial intelligence

1.2.1. Machine learning and neural network

AI refers to a system that can perceive and interpret external data and independently perform actions to achieve their goals [33]. As a category of AI methods, machine learning (ML) uses algorithms and mathematical models including support vector machine (SVM), decision tree, Bayesian network, and deep learning (DL) in supervised, unsupervised, or reinforcement learning schemes to improve task performance on given data [34]. In supervised ML, data with known input and output is used to optimize the model. The input and output signals could be a combination of arrays, vectors, or matrices. Once the learning process is completed, the model is capable of predicting accurate output with new input data [35]. In unsupervised ML, algorithms take unlabeled data, identify the commonalities in the data, and output separated data clusters [36]. In reinforcement ML, agents are created to complete particular tasks without prior knowledge and by setting up several policies to reward or penalize agents

depending on their performance. The final goal for the agents is to maximize its reward [37].

DL is based on artificial neural networks (ANNs) [38] with multiple hidden layers. Each layer has multiple neurons, as shown in Figure 3. Each neuron takes its input from the previous layer, applies a mathematical calculation with the weight in the connection, and passes the output to the next layer. Based on the network structure, there are several types of ANNs, including feedforward neural network (FNN) [39], recurrent neural network (RNN) [40], convolutional neural network (CNN) [41], and modular neural network (MNN) [42].

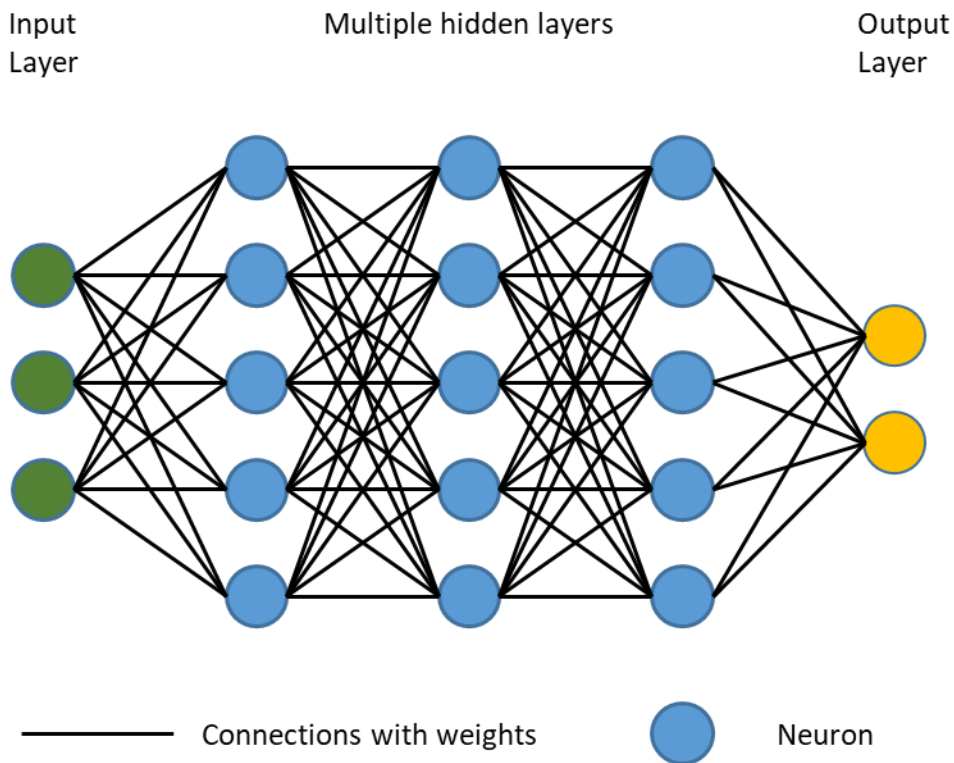


Figure 3. Artificial neural networks (ANNs) structure.

The first ANN proposed by psychologist Rosenblatt in 1958 was designed as a mathematical model to mimic human brain with multiple neurons and recreate brain responses to random environmental stimulations [43]. Since then, many researchers developed other variations of ANN and implemented them in a host of DL applications.

For example, Arel et al. [44] created multiple traffic light control agents using Q-learning [45] (reinforcement learning) to reduce delay times in a simulation environment comprising eight intersections with a steady traffic pattern. Agents controlling traffic lights were rewarded (from -1 to 1) with a reward policy (function) such that reducing delay time would result in a positive reward and vice versa. After 20,000 iterations, agents learned to control the traffic lights collaboratively and outperformed conventional isolated control system.

Unlike typical ANNs, autoencoder networks have the same size of output as the input with the smallest bottleneck layers in the middle of the network [46]. This structure uses unsupervised learning to encode the input (layers to the left of the bottleneck) and reconstruct less noisy representation (layers to the right of the bottleneck) with close resemblance to the input. Autoencoders are used in applications that require noise removal, such as speech recognition [47], image noise control [48], and semantic information extraction [49].

In another study, Amodei et al. [50] presented an end-to-end approach using RNNs to recognize both English and Mandarin in one generative algorithm. Since supervised RNNs use internal memory to store information from previous timestamps and process continuous sequence, they are suitable choices for sequential data such as

speech recognition. Other ANN structures such as FNN and autoencoders are also adopted in voice recognition to reduce input noise and model speech pattern using both supervised and unsupervised learning.

1.2.2. Computer vision and CNN applications

Computer vision (CV) is a field in computer science that aims to interpret visual information embedded in digital imagery (photos and videos) [51]. A grayscale digital image can be described as a matrix (i.e., pixel grid) in which cells (i.e., pixels) contain an integer (ranging from 0 to 255) indicating the level of brightness of that cell [52]. The image resolution is reflected in the dimensions of the matrix. If the image is in color, three overlapping matrices (or channels) corresponding to the three basic colors of red, green, and blue (a.k.a., RGB) can be used to describe the image [53]. Together, these channels represent all visible colors to the human eye. A digital video can be deconstructed to its building blocks (i.e., frames). When played sequentially at a specific speed (expressed in frames per second, or FPS), the output is perceived by the human eye as a continuous scene. The FPS for modern real-time videos is 30 and above [54].

CV methods broadly cover image acquisition, pre-processing, feature extraction [55], detection [56], and segmentation [56]. CNN is one of the most popular forms of neural network architectures that have been successfully used for a variety of CV applications [57] such as image classification (general image content), object detection (object types and locations in an image), and instance segmentation (pixel-level object boundaries in an image). Figure 4 demonstrates examples of CNN outputs in the context of this research. In these examples, a classification CNN takes an input image and

produces a binary output indicating the presence or absence of a damaged roof in the image. By combining this model with sliding boxes, one can try to determine the location (i.e., localize) of the damaged roof. However, since the sliding box method is extremely slow [58], object detection CNNs are instead developed to produce bounding boxes for localizing and classifying multiple objects simultaneously. Finally, segmentation CNNs are capable of predicting pixel-level boundaries (a.k.a., masks) of detected objects with classification information.

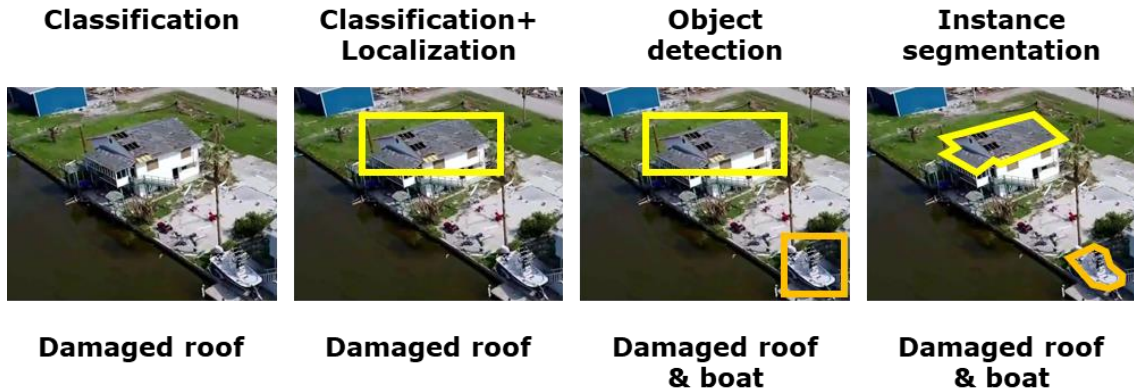


Figure 4. CNN model outputs: classification, classification and localization, object detection, and instance segmentation.

The convolution computation defined in Equation 1 is often implemented in image processing to achieve effects including blurring, sharpening, edge detection, and more [59]. In this Equation, $input(i, j)$ represents the pixel value in the input image, $output(m, n)$ indicates the output pixel value, and K is the filter kernel. The symbol \times denotes convolution computation which multiplies each value in the input matrix with similar local values (weights) in the kernel and then sums them to generate the output.

As the example in Figure 5 shows, a 3×3 kernel is applied on a 5×5 input matrix and is slid over all positions, resulting in a 3×3 output matrix. The sliding interval (e.g., 1 pixel in this example) is termed stride [60]. By definition, convolution computations reduce the size of the input matrix, as evidenced by the example shown in Figure 5. Therefore, to ensure that the output matrix maintains the exact same dimensions of the input matrix, extra pixels (with 0 values) are added on the outside of the output matrix, in a process referred to as padding [61].

$$output(m,n) = \sum_{i,j=-\infty}^{\infty} input(i,j) \times K(m-i,n-j) \tag{1}$$

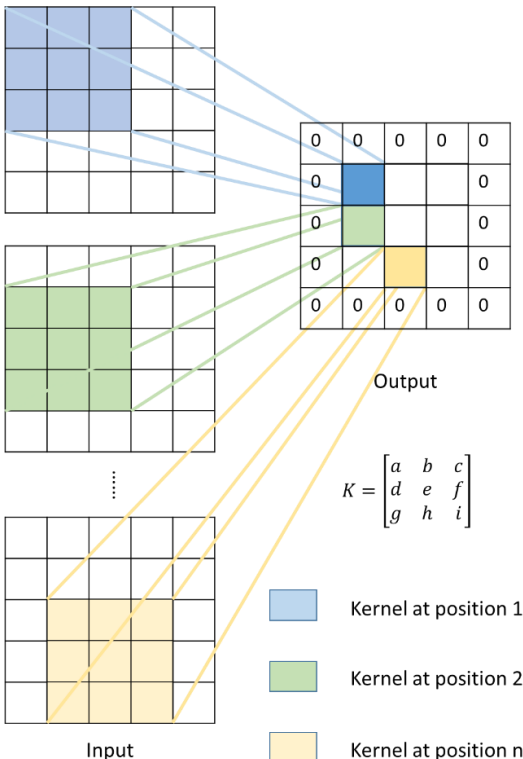


Figure 5. Convolution computation example.

CNN models apply convolution computations on RGB input grid to extract useful features (e.g., object edges, shapes, and patterns), followed by multiple max pooling [62], regression, and optimization calculations, and ultimately output results in the form of image classification, object detection, or instance segmentation. Figure 6 demonstrates a sample CNN architecture which contains multiple max pooling and convolution layers. Each convolution layer consists of several kernels with separate weights. Adjusting the values of each of the many parameters of a CNN (e.g., weights in convolution kernels) is key to successfully understanding input and generating an accurate output. To obtain these weights, a CNN must be first trained on ground truth data, that often comes in the form of a carefully annotated dataset. Multiple iterations of training process lead to minimizing the error between ground truth and prediction. The optimized parameters and layer structure (i.e., CNN architecture) are referred to as a fully trained model. Researchers are developing different CNN architectures, with varying number of layers, kernel sizes, and loss functions, that can handle different input and output formats and diverse object classes, with best possible accuracy and processing speed.

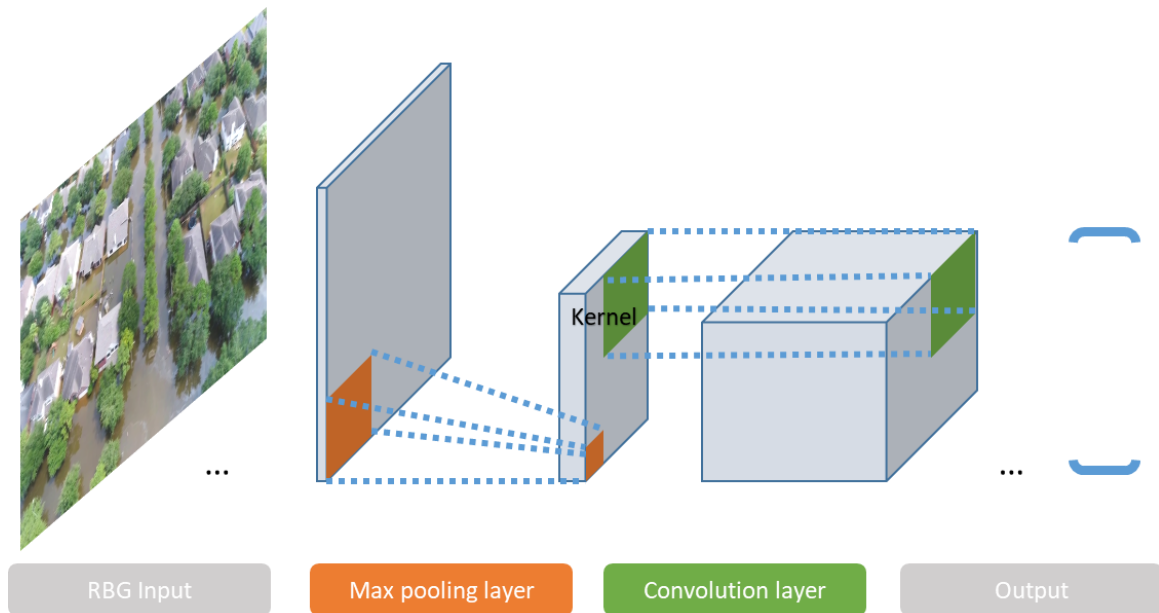


Figure 6. Convolutional neural network (CNN) structure.

The very first CNN “neocognitron” published by Fukushima [63] was designed to recognize the patterns of a given black and white digital image. This CNN output has only 10 categories corresponding to numbers 0 to 9 with probabilities. Later, Lecun et al. [64] proposed a CNN with denser layers and expanded output classes including numbers and all English letters in MNIST dataset. Beyond document analysis, some CNN applications aim at image content search and captioning, facial recognition, and video processing. Table 5 lists examples of such CNN applications with their corresponding learning type, input and output data type, and speed.

Table 5. Summary of CNN applications in different domains.

CNN application	Learning type	Input	Output	Speed
Facial recognition	Supervised	Facial images or videos	Emotion, identity, facial features (e.g., lips)	>30 FPS
Document analysis	Supervised	Images	Text	N/A
Self-driving	Supervised	First person driving stream	Object location, distance, and meaning (e.g., traffic signs)	>250 FPS
Drug discovery	Supervised	3-D medicine vectors	New 3-D drug models	N/A
Language processing	Supervised	Audio	Text	N/A
Gaming	Reinforcement	Game features	Gaming operation	>30 FPS
GAN	Unsupervised	Images	Images	N/A
Disaster reconnaissance	Supervised	Aerial fly-over visual input	GIS of damage and victim	>30 FPS

In recent years, self-driving vehicles have utilized CNN to process visual input from cameras for recognizing and localizing traffic lanes, traffic lights, pedestrians, road signs, and other vehicles [65, 66]. The output of these CNNs is used to control driving operations such as start, stop, steer, brake, and accelerate. Considering that a driving vehicle needs to make decisions in a split of a second, processing speed is a critical factor in this domain and currently, an FPS of up to 250 has been achieved [67]. Besides cameras input, self-driving vehicles normally integrate multiple sensors such as radar, global position system (GPS), and light detection and ranging (LiDAR) to generate accurate traffic information. Similarly, facial recognition uses CNNs to locate human faces in image or video input, followed by classifying emotion, identity, or facial

features such as lips and eyebrows for various applications including entertainment and security [68, 69].

Although CNNs have convolution kernels that are designed to process two-dimensional matrices, with proper pre-processing such as Fourier transformation [70], CNNs can also process data of one or multiple dimensions. For instance, in speech recognition, time and frequency data input can be transformed into two-dimensional matrices to be processed by CNNs [71]. In medical and chemical discovery, three-dimensional models of molecules are created as training data to automatically generate new chemical combinations [72, 73]. CNNs are also implemented in gaming to create agents that outperform human players. For example, alpha GO uses visual representation of the game features such as location and color of stones to extract gaming information. The game agent further utilizes reinforcement learning to learn strategies by playing with itself for millions of games. To date, alpha GO has defeated the best human GO player by a large lead [74]. Goodfellow et al. [75] proposed a generative adversarial network (GAN) which integrates autoencoder and CNN to create new data with the same statistical attributes of the original training data. Applications for GAN include animation creation [76], video game development [77], and data augmentation [77].

1.3. Organization of this Dissertation

There is a tremendous amount of aerial visual data collected during and after disasters. This data, however, is often processed manually which is a resource intensive and error-prone process, leading to inefficiencies and bottlenecks in disaster management practices. The goal of the research presented in this Dissertation is to utilize

CNNs to automate the processing and mapping of disaster impact data. In particular, two fully annotated video datasets, namely Volan2018 and 2019 are created to train, validate, and test customized CNNs for disaster reconnaissance. Altogether, Volan2018 (bounding box annotation) and Volan2019 (pixel-level annotation) contain 255,458 instances of key disaster damage categories including building roofs (damaged and undamaged), flooded area, car, boat, people, debris, road, and vegetation. The term “Volan” is singular for Volans (short form of *Piscis Volans*), which is a constellation in the southern sky, representing a flying fish that can jump out of the water and glide through the air on wings. Volans was one of twelve constellations created by Petrus Plancius from the observations of Pieter Dirkszoon Keyser and Frederick de Houtman [78].

First, CNN architectures you-only-look-once (YOLO) [79] and RetinaNet [80] are applied to investigate bounding box detection. Next, pixel segmentation architectures Mask-RCNN [81] and PSPNet [82] are implemented to pursue more accurate prediction and mapping. Detailed analysis considering the effect of viewpoint altitude, data balance, pre-trained weights, error matrix, and CNN model selection are carried out to improve the efficiency and reliability of the developed methods. Moreover, predicted objects and their corresponding locations are further projected on a geocoded map to quantify, locate, and communicate the disaster impact with response teams and other stakeholders. Experiments are conducted to investigate moving object projection, object tracking and counting, and reference point updating and calibration. Results suggest that the designed mapping technique is capable of projecting disaster information from

perspective aerial views onto an orthogonal coordinate system with high accuracy and speed.

The remainder of this Dissertation is organized as follows: In Chapter 2, a thorough review of the literature is conducted. Chapter 3 explains the methodology for building the video datasets and training CNN models. Next, bounding box detection experiments and results (Chapter 4) and pixel segmentation detection experiments and results (Chapter 5) are presented. In Chapter 6, the developed mapping technique is described with the help of several experiments. Finally, Chapter 7 will conclude the work conducted in this research with a summary of findings and potential direction for future work.

2. LITERATURE REVIEW

This Chapter reviews hurricane categories, and studies relevant literature in big data remote sensing, artificial intelligence (AI) structures, large imagery datasets, GPS-free mapping techniques, and social vulnerability within the context of natural disasters.

2.1. Hurricane impact

The Saffir-Simpson Hurricane Scale [83] groups hurricanes into five major categories. Table 6 shows the category specifications with a recent example. Category 1-4 are hurricanes that have the wind speed from 119 KPH (74 MPH) to 252 KPH (155 MPH). Category 5 hurricanes have a wind speed greater than 252 KPH (156 MPH) [84].

Table 6. Saffir-Simpson Hurricane Scale.

Hurricane scale	KPH	MPH	Example (year)
Category 1	119-152	74-95	Lorena (2019)
Category 2	154-177	96-110	Humberto (2019)
Category 3	178-209	111-130	Miriam (2018)
Category 4	210-252	131-155	Lorenzo (2019)
Category 5	>252	>156	Dorian (2019)

Hurricanes cause strong winds and heavy rainfall, and can cause catastrophic damage before gradually slowing down and losing energy as they move over land. Hurricanes Harvey, Irma, and Maria in the 2017 hurricane season are among the costliest water-related disasters in recent history which led to \$125 billion in damages [85]. In 2018, two major hurricanes, Florence [86] and Michael [87], struck the U.S. coastal areas and left behind approximately \$50 billion in damages. More recently, in 2019, two category 5 hurricanes, Lorenzo (developed from category 4) [88] and Dorian, along with 16 other named storms affected parts of U.S., Canada, U.K., and west Africa. More

hurricanes are forecasted to impact coastal communities due to a global rise in sea temperatures [89].

Considering the severity of hurricane damage and cascading effects that follow, particularly as experienced by communities along the U.S. Gulf Coast, the main focus of this Dissertation is to explore the extent to which current capacities in disaster management can be augmented with a special focus on hurricane damage.

2.2. Big data, remote sensing, and disaster information retrieval

The digital technology has been reshaping information acquisition, exchange, and delivery for decades. With more prevailing personal mobile devices producing various forms of data such as text, audio, photo, video, blogs, and tweets, numerous data-driven applications are continuously developed [90]. With respect to disasters, big data can help better understand and document vulnerability, degree of preparedness, social equity and risk, as well as the quality of SAR and recovery [91-93]. Previous research has focused on mining social media data during disasters. Craglia et al. [94] mapped VGI using user-generated content (UGC) from Facebook and Twitter. Kim and Hastak [95] extracted disaster information using ML from online social networks. Yuan et al. [96] analyzed semantic meanings in tweets during hurricane Matthew. Goodchild and Glennon [97] crowdsourced disaster information collection to the public. Zou et al. [98] mined tweets during hurricane Harvey to provide an indication of geographical disparities in recovery. Lastly, in order to achieve a faster disaster inspection, a Monte-Carlo searching algorithm for post-disaster search using drones was tested by Baker et al. [99] and found to be more efficient than conventional sweeping search. However,

mining the data contributed by a large number of people often involves issues such as noise, bias, and authenticity.

Aerial remote sensing capacity is crucial in many domains including disaster management, agriculture, ecology, marine industry, mining, construction, transportation, and urban design [29, 100]. CV and 3D photogrammetry modeling are often integrated with aerial surveys to generate GIS data. Radovic et al. [101] explored using YOLO to detect ground airplanes in satellite images. Region-based CNN (R-CNN) and kernelized correlation filter (KCF) were used by Han et al. [102] to build a real-time human tracking system from drones. In lieu of an onboard computer, Narayanan et al. [103] tried high-performance cloud computing to achieve real-time drone detection of everyday objects. Guirado et al. [104] tracked and monitored sea whales from satellite imagery using Faster R-CNN and an in-house whale dataset. Friedel et al. [105] utilized self-organizing map (SOM) of the aerial spectral information to map soil and vegetation in drone footage. On a mining site, Suh and Choi [106] integrated ground control points (GCPs) and drones to produce maps of site terrain with a 14-cm error. Ventuna et al. [107] used drone cameras and GCPs to remodel the nursery environment for coastal fish with a high accuracy (89.1%). Cunliffe et al. [108] used images captured by drones and structure-from-motion (SfM) to produce accurate grain biophysical data. While past work has successfully extracted information from publicly available big data and aerial footage, there is still a lack of research on aerial remote sensing for real-time disaster information extraction.

Table 7 demonstrates examples of previous work in aerial detection of disaster-related object classes. As shown in this Table, Rahnemoonfar et al. [109] proposed a method that uses their own CNN architecture, followed by an RNN to detect flooded area from an aerial angle with 96% pixel accuracy. Ghaffarian and Kerle [110] explored histogram of the oriented gradients to identify debris types in drone and satellite imagery, but did not locate the debris automatically. Li et al. [111] built a CNN using the VGG-19 architecture [112] to quantify and localize damage on a map with 95.5% accuracy from social media images. However, this model only described ground-level damage information since almost all social media images are taken on the ground. Liu et al. [113] compared SVM, FNN, and CNN for wet-land classification from images acquired from drones, and achieved a 88% accuracy in classifying wetland using the best performing CNN model. Chen and Li [114] created a framework to detect building roofs with a 75% precision using Mask R-CNN for post-earthquake recovery and PDA. Rao et al. [115] utilized a fully connected network (FCN) to detect single roads from satellite images with 92.5% accuracy. While this approach provides good accuracy, it costs 6.32 seconds to process one image due to the dense layers in the FCN. Ghaffarian et al. [116] proposed CNN models to update post-disaster reconstruction data over time from OpenStreetMap [117], and extracted information such as building edge, damage, and rebuild progress. While this framework captures the recovery progress with 84.1% precision, it heavily relies on the accuracy of the third party data provided by OpenStreetMap.

Table 7. Current aerial disaster detection examples.

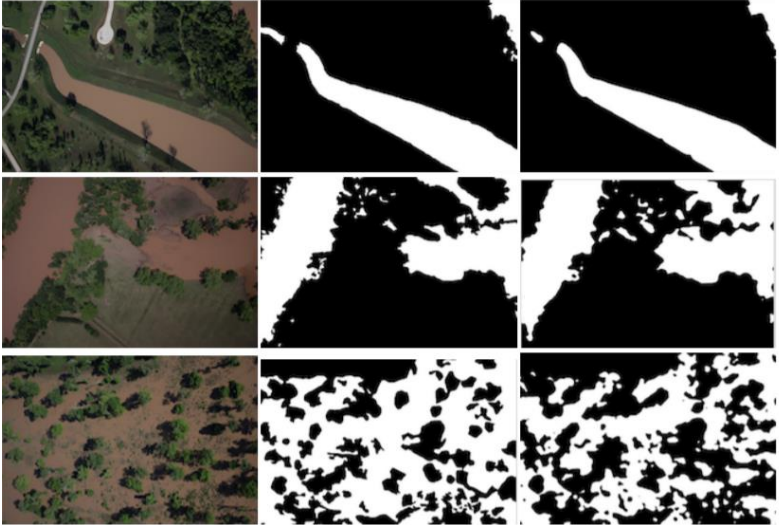
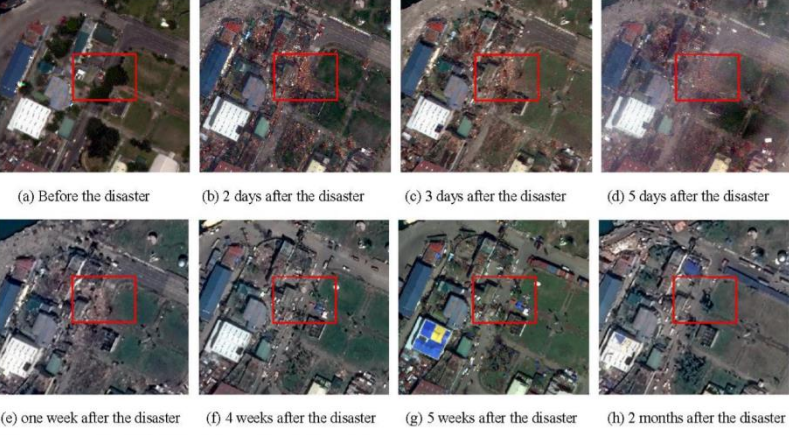
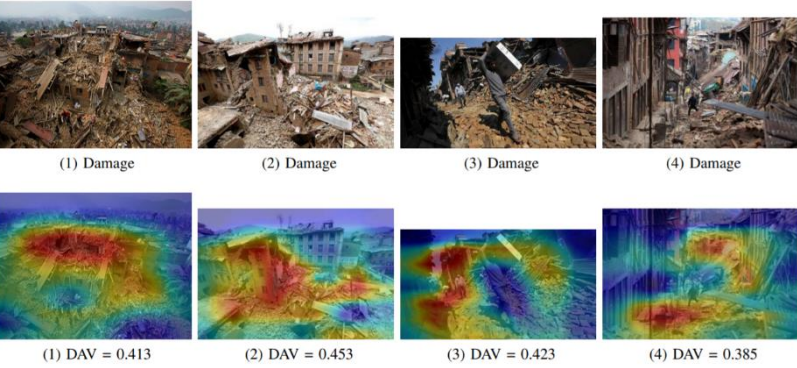
Authors	Targets	Precision (%)	Detection Examples
Rahmemonfar et al.	Flood area	96	
Ghaffarian and Kerle	Debris	95.5	 <p>(a) Before the disaster (b) 2 days after the disaster (c) 3 days after the disaster (d) 5 days after the disaster</p> <p>(e) one week after the disaster (f) 4 weeks after the disaster (g) 5 weeks after the disaster (h) 2 months after the disaster</p>
Li et al.	Ground damage	95.5	 <p>(1) Damage (2) Damage (3) Damage (4) Damage</p> <p>(1) DAV = 0.413 (2) DAV = 0.453 (3) DAV = 0.423 (4) DAV = 0.385</p>

Table 7. Continued.


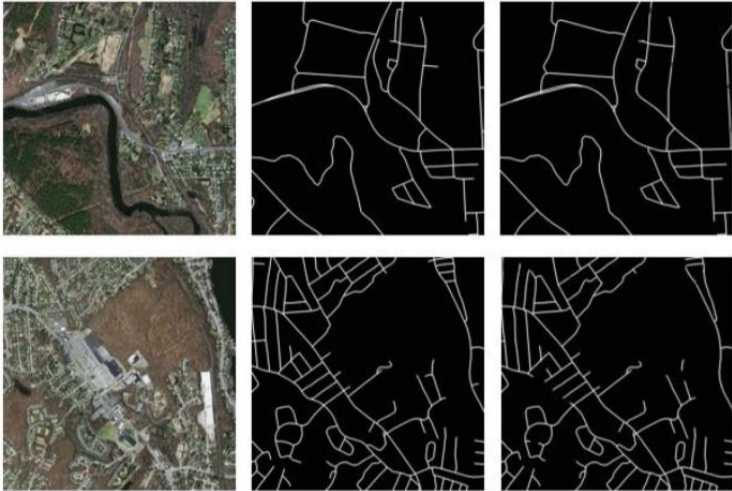
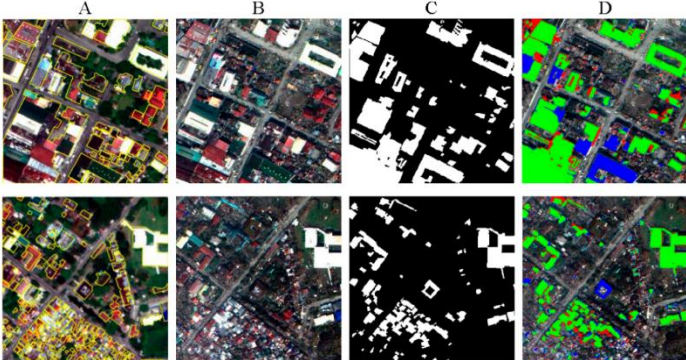
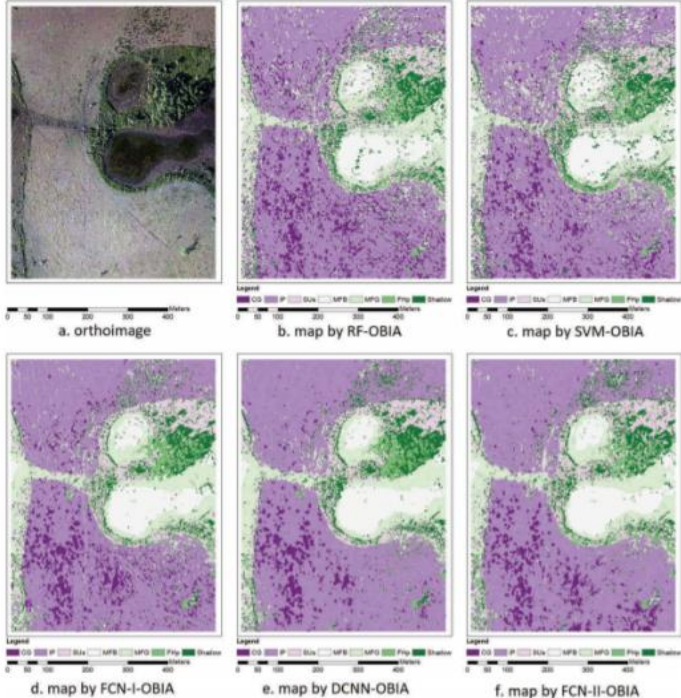
Authors	Targets	Precision (%)	Detection Examples
Chen and Li	Building roof	75.0	
Rao et al.	Road	92.5	

Table 7. Continued.

Authors	Targets	Precision (%)	Detection Examples
Ghaffarian et al.	Building edges/recovery	84.1	
Liu et al.	Wetland	88	

2.3. CNN architectures

With the development of new graphics processing unit (GPU) technology, the application domain of CNN has been rapidly expanding. Alex et al. [118] proposed denser AlexNet with 8 layers (60 million parameters) achieving 47.1% precision on

ImageNet dataset. Simonyan and Zisserman [112] designed VGGNet with 11-19 layers, which outperformed AlexNet by linking more network parameters. GoogLeNet proposed by Szegedy et al. [119] included inception modules that integrate convolution computations and max-pooling, thus outperformed VGGNets. The more complex ResNet [120] architecture linked multiple layers with residual networks leading to a CNN model that can outperform an average human by 3.57%.

However, the above architectures are designed for image classification, and cannot produce output that describes the types (i.e., multiple object classes) and locations of detected objects in the image. As the information carried through CNN layers contain features that lead to more accurate candidate regions proposals, several researchers have focused on more efficient architectures for object detection. For example, R-CNN introduced by Girshick et al. [121] used region proposal and achieves a 53.3% mean average precision (mAP) on VOC dataset. Girshick et al. [122] later introduced Fast R-CNN with region of interest (RoI) to improve prediction speed. Ren et al. [123] proposed region proposal network (RPN) forming Faster R-CNN, which achieved a 73.2% mAP on VOC 2007 datasets [124]. Dai et al. [125] investigated pre-processing input with a 3×3 position-sensitive map, resulting in region-based fully convolutional networks (FRCN) that process images 2.5-20 times faster, with similar precision as Faster-RCNN.

One-stage architectures process images with a higher speed since they generate candidate regions and classify them at the same time, albeit with lower precision. For instance, YOLO introduced by Redmon et al. [126] accomplished 63.4% mAP on VOC

2007 dataset with the processing speed of 45 FPS. Liu et al. [127] proposed single shot multibox detector (SSD) based on MobileNet which achieves real-time speed (more than 30 FPS) and 76.8% mAP on VOC 2007. Lin et al. [80] suggested a CNN architecture called RetinaNet with focal loss function, which can better learn hard examples, yielding 37.8% average precision (AP) on COCO dataset.

Considering pixel segmentation, He et al. [81] created Mask-RCNN which produces pixel-level segmentation by adding one more 1×1 convolution layer on top of the predicted bounding boxes [81]. Badrinarayanan et al. [128] introduced SegNet which consists of encoder and decoder layers with bottleneck layers in between. SegNet is capable of producing pixel level semantic segmentation at 90.40% pixel precision on CamVid dataset. Zhao et al. [82] contributed PSPNet which replaced the bottleneck layer with a pyramid pooling module (PPM) which consist of four parallel convolution network. This PPM is proven to outperform SegNet especially on small objects.

2.4. CNN training datasets

In order to design and test CNN architectures for better performance (i.e., speed, accuracy, output type), it is essential to have large-scale annotated datasets. To this end, several challenges and corresponding datasets were created in the past to solicit innovative solutions. The ImageNet large scale visual recognition challenge (ILSVRC) was created by Russakovsky et al. [129] for image classification with a dataset called ImageNet. ImageNet contains 14,197,122 images with 27 classes, each consisting of several sub-categories. For example, the class sport has subcategories such as racing, contact sports, and water sports. The PASCAL challenge [130] was initiated by

Everingham et al. resulting in dataset called visual object classes (VOC) for object detection. VOC covers 20 classes with 11,530 images containing 27,450 annotated objects. Common objects in context (COCO) challenge was introduced by Lin et al. [131] with a dataset of the same name for segmentation. COCO contains 80 object categories with 330,000 images and 1.5 million object instance segmentation. While ImageNet, VOC, and COCO datasets were originally created for classification, object detection, and segmentation tasks, respectively, the current versions of these datasets support all three tasks.

The majority of large datasets to date portray everyday objects such as animals, computers, furniture, food, and trains, and there are few remote sensing datasets. AID [132] contains 10,000 images retrieved from Google Earth and it is annotated for classification. Bounding box annotated dataset PatternNet [133] consists of 38 classes with images obtained from satellites. Collected from a lower altitude, drone-captured images in Minidrone [134] are annotated for classes people and car mainly for considering intended applications for surveillance. Ritwik et al. [135] built a satellite imagery dataset named xBD consisting of 850,736 building annotations with damage description. However, xBD contains only a single class, building, and lacks other classes. To the author's best knowledge, there is a dearth of large fully annotated aerial visual datasets for applications in disaster information retrieval.

2.5. GPS-denied object localization and mapping

The output (i.e., bounding-boxes, segmentation) of a CNN model can be used to localize, quantify, and map detections for end users, depending on their needs and

expected levels of access to information. To obtain the position of captured images (and subsequently, using this position information to calculate the absolute locations of detected objects), the most trivial solution is to equip the data collection platform (helicopter or drone) with a GPS sensor that continuously transmits the global coordinates of the device. However, GPS data is not always available when timely disaster information is needed the most. In urban settings, for instance, GPS signals can be intermittent due to signal loss or multipath effect (i.e., signals reflected off buildings or other obstacles) [136]. During times of major disasters, it has been reported that real-time kinematic (RTK) tower failures caused by disasters heavily reduced the accuracy of GPS signal down to 16 feet or worse [136]. Additionally, if visual data collection is crowdsourced, some users may not have the will or knowledge to share their GPS location (due to privacy issues, for example). In other domains such as defense, indoor navigation, and underwater exploration where operations are carried out in unfamiliar or unfriendly environments, open access to GPS signals may not be even possible. Therefore, enabling GPS-denied localization and mapping is of major importance in the work conducted in this Dissertation.

To obtain the position, velocity, and time (PVT) information in GPS-denied environments, some researchers have used alternative data sources such as inertial measurement unit (IMU) readings, stereo (depth) cameras, laser scanner, Wi-Fi signals, radio-frequency identification (RFID), and sonar [137-144]. Chao et al. [145], for example, used monocular cameras to build visual simultaneous localization and mapping (SLAM) for drone navigation. Pestana et al. [146] utilized OpenTLD tracker to enable

drones to follow moving objects. Rajeev et al. [147] proposed indoor (GPS-denied) drone navigation using augmented reality (AR) and a pre-built 3D model. Bachrach et al. [148] integrated a depth camera with regular RGB cameras to generate instant 3D paths for navigation. Scaramuzza et al. [149] designed a visual-inertial multi-drone system with cameras and IMUs to autonomously fly and map the environment. While previous work has investigated GPS-denied navigation, the challenge of aerial object localization and mapping using only visual information (i.e., vision-based navigation) has to a large degree remained unexplored.

2.6. Social vulnerability

Social vulnerability refers to the susceptibility of a certain group of people to disaster impact [150]. Burton et al. [151] framed this concept through the lens of social, economic, and physical conditions that make people and communities vulnerable to natural hazard events. Blaikie et al. [152] defined social vulnerability as a social or personal characteristic of people and their ability to anticipate, cope with, and recover from hazard impact. Cutter et al. [153] introduced social vulnerability index (SoVI) to describe and analyze the county-level social vulnerability. Collectively, the main components of social vulnerability research include the identification of influential variables, interrogation of the impact of these variables and their relationships, and exploring approaches to reduce social vulnerability.

Admittedly, certain population groups are more vulnerable than others. For example, studies conducted on the social impact of hurricane Katrina revealed that New Orleans communities with low-income and limited education recovered at a slower pace

[154]. Low-income households, that are less likely to have sufficient disaster insurance or safeguards, are often underfunded during the rebuild and recovery process which is primarily funded by insurance claims and bank loans. In contrast, high-income families possess more resources to recover from damages; some can even relocate to their second home that is not impacted by the disaster [155].

Gender and race are two more important factors in social vulnerability. Due to the fact that women are disproportionately responsible for raising children, taking care of elderly family members, and doing housework, they may experience more anxiety in the aftermath of a major disaster [156]. Moreover, women are at the disadvantaged position in the post-disaster job market and are less likely to earn enough income to pay for recovery expenses [157]. Studies show that black and Hispanic families on average deal with more housing problems than white population groups [158], and many underrepresented and minority groups live in unsafe buildings and experience sheltering problems after disasters [159]. The agreed upon influential factors include socioeconomic status, gender, race and ethnicity, age, commercial development, employment, rural or urban location, residential property infrastructure, occupation, family structure, education, population growth, medical service, social dependence, and special need population [160-165].

Understanding the disparity of these factors is critical because any or a combination of them could be precursors to social vulnerability. For example, high-income groups who enjoy a better socioeconomic status live in more developed residential properties with better infrastructure, making them less vulnerable and more

resilient to disaster impact. Conversely, vulnerable groups with low income and limited education, often do not have enough insurance, and lack risk awareness and resources to recover, making them more susceptible to disasters. Since disaster risks are not borne equally by all members of society but are imposed disproportionately on already vulnerable social and economic groups [153, 166-169], research and outreach activities that enable public participation and engagement in disaster resiliency practices can potentially address some of these social vulnerabilities [170], and promote trust-building between ordinary people, authorities, and disaster response agencies.

3. METHODOLOGY

This Chapter covers the general methodology designed and pursued in this research, including the study of disaster impact, data preparation, model development and implementation, and the analysis of results. It also presents four research challenges that this research aims to address.

3.1. General methodology

With the development of personal recording and sharing technologies with mobile connectivity, more first responders, NGOs, and ordinary citizens are generating large amounts of data describing the aftermath of natural disasters including hurricanes and floods. The research presented in this Dissertation attempts to collect, annotate, experiment with, and analyze these data (particularly aerial imagery) using CNNs, as demonstrated in the general methodology in Figure 7.

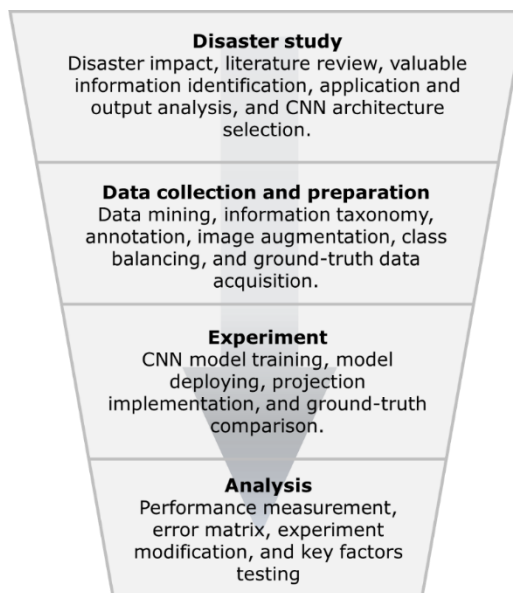


Figure 7. Dissertation methodology.

For data collection, web mining and keyword search is used to obtain videos from the public domain (e.g., YouTube). As explained later, this raw data needs accurate annotation for successful CNN training, validation, and testing. The type of data annotation (e.g., bounding box or pixel segmentation annotation) depends on the choice of CNN architecture. Data balancing in the form of up-sampling, down-sampling, augmentation, and class separation are also carried out to improve model performance and create more generalizable output. Given the fully prepared data (training, testing, and ground-truth data), multiple CNN models are trained and tested. Prediction results are then compared with ground-truth data to evaluate model performance, and several iterations are used to modify and retrain the CNN models to achieve better performances. Key factors such as pre-trained weights, error matrix, and data balance will be analyzed which vary from one CNN architecture to the next.

This general methodology can be extended to other disaster types as long as the required information is perceivable from aerial angle. Although the visual appearance of damage varies from one type of disaster to the next, the designed techniques for data pre-processing, data balancing, model training and testing, and viewpoint projection and mapping are to a large extent generalizable to other disasters.

3.2. Research challenges

3.2.1. Identifying objects of interest in disaster footage

In order to produce proper datasets to train CNNs, a total of 22 aerial videos depicting the aftermath of previous hurricanes are mined from the web. After examining

the video contents, a list of objects of interest is created. Table 8 shows that most aerial videos contain scenes that depict flooded areas building roofs (both damaged and undamaged), cars, debris, vegetation, people, and boats. In addition to visibility in aerial footage, other key criteria for selecting these classes for video data annotation include application to disaster response, and potential for future research. For example, the location of people not only does help first responders in SAR operations, but may also guide decisions with respects to evacuation and sheltering, resource delivery, and recovery efforts.

Table 8. Disaster object classes and applications.

Object classes	Potential application
Flooded area	Search and rescue, wayfinding, storm surge mapping, aid delivery, insurance claims
Building roofs (damage, undamaged)	Damage information map, construction repair, search and rescue, insurance claims, debris removal
Car	Search and rescue, insurance claims, property rescue
Debris	Cleanup, damage information map, construction repair, search and rescue
Vegetation	Cleanup, repair
People	Search and rescue, resource deployment, victim relocation
Boat	Insurance claims, search and rescue

3.2.2. Selecting the best CNN architecture

In order to quantify the performance of a CNN model, the trade-off between accuracy (expressed in terms of average precision or AP) and speed (expressed by FPS in generated output) must be considered. Table 9 lists examples of CNN architectures and their performance when tested on COCO test-dev data [131]. Ideally, the desired model for disaster information extraction should be able to operate in real-time (i.e., 30

FPS or higher). On the other hand, slower but more precise models are also worthy of investigation since the slow speed may be overcome through the use of proper post-processing techniques. The selection of CNN architecture also depends on the desired output types, i.e., classification, object detection, or segmentation.

Table 9. CNNs test Average precision (AP) and processing time (millisecond) on COCO test-dev [80].

CNN architecture	AP (%)	Time (ms)
YOLOv2 [79]	21.6	25
SSD321 [127]	28.0	61
R-FCN [125]	29.9	85
DSSD513 [127]	33.2	156
FRCN [125]	36.2	172
RetinaNet-101-800 [80]	37.8	198

3.2.3. Factors influencing model performance

In line with the previous challenge, it is critical to understand the influence of both internal and external factors on model performance, and prediction quality and generalizability. Internal factors include those that are inherent to the CNN architecture (e.g., pre-trained weights), while external factors describe input data distribution (i.e., class balance), drone viewpoint altitude, and object sizes. To address this challenge, several CNN models are trained and tested on different data combinations, and targeted data augmentation strategies will be utilized to investigate and isolate the effect of each factor on model performance.

3.2.4. Quantifying disaster damage

In the disaster aftermath, as summarized in Table 3 and Table 4, cost estimation is an important consideration for deployment of resources to the affected areas. For example, debris removal may cost close to 30% of the total cost of disaster response [171]. Timely debris removal is also necessary for public health and to reduce the load on sanitary systems [172]. Therefore, in addition to detect the location of disaster damage, this Dissertation aims to enable the quantification of such damage to provide estimates of cleanup and recovery cost and time. To achieve this, transforming visual information from drone's perspective view onto an orthogonal mapping system for easy geometric calculation is necessary. The key challenge in achieving this is how to perform such operations in the absence of geolocation data, or when such data is sparsely available. Therefore, this research explores the possibility of using only visual information (without relying on drone's GPS coordinates) to localize and map detected disaster damage.

4. BOUNDING BOX OBJECT DETECTION*

This Chapter presents bounding box object detection by describing the work completed on dataset preparation and balancing, CNN model architecture, performance measurement, experiments conducted, and analysis of results.

4.1. Volan2018 data description

4.1.1. Data collection and annotation

The first dataset created in this research is named Volan2018 [173], which contains 8 videos (Volan001-008) obtained from YouTube using web mining by searching keywords such as “hurricane”, “damage”, “aerial”, “drones”, and “aftermath”. Volan2018 contains eight aerial videos (with 1280×720 resolution, 30 FPS) from different hurricanes that occurred during the 2017-18 hurricane seasons (including hurricanes Harvey, Maria, Irma, and Michael). Together, this video dataset covers locations in the city of Houston (Texas), southern areas of Texas (i.e., Port Aransas, Holiday Beach, and Rockport), Puerto Rico (U.S. territory), Big Pine Key, Mexico Beach, and St Joe Beach (all in Florida). As part of pre-processing, frames containing watermarks in Volan001 and 002, and black margins in Volan008 were removed. These videos are extracted frame by frame and then annotated using DarkLabel [174] for the following object classes: damaged roof, undamaged roof, debris, vegetation, car, and flooded area. One object in one frame is defined as one instance so that the pixel

*Part of this Section is reprinted with permission from “Convolutional neural networks for object detection in aerial imagery for disaster response and recovery” by Yalong Pi, Nipun Nath, and Amir Behzadan, 2020. Advanced Engineering Informatics, 43, 101009, Copyright [2020] by Elsevier.

coordination values (with the most upper left point of the frame serving as origin) of the bounding boxes are obtained for training. Annotating consecutive frames in a video using DarkLabel can cost one annotator approximately 2 seconds per frame, although this varies based on the content of the frame and the number and diversity of objects of interest that must be annotated. For example, to annotate a 10-minute long video (18,000 frames), 10 hours of work is expected. Figure 8 illustrates annotation samples for each class following the annotation strategy below:

- **Flooded area:** Flooded areas in Volan2018 are widespread and connected. Bounding boxes corresponding to flood can thus cover the most portion of a frame (including other class instances). To avoid this, flooded area is split by drawing multiple small bounding boxes that cover only the flooded area without including other objects.
- **Debris:** Small-sized debris are ignored in annotation since small amounts of debris will most likely not turn into major obstacles during disaster operations. Large-sized debris that could impede transportation or cause damage is annotated by drawing bounding boxes around the edges. Similarly, visible destructions to infrastructure and splintered houses are annotated as debris.
- **Cars:** Bounding boxes are drawn around the edges for each car, individually.
- **Vegetation:** Large boxes are drawn covering the tree cluster if those boxes do not cover any other class instance. Otherwise, one separate box is drawn for each tree. Lawn is not classified as vegetation for the purpose of this annotation.

- Damaged roof: Bounding boxes are drawn to cover the entire roof (include undamaged part), but do not include elements below the roof (e.g., walls).
- Undamaged roof: Bounding boxes are drawn to reach the roof drip edge (containing skylight, chimney, etc.), but do not include elements below the roof (e.g., walls).

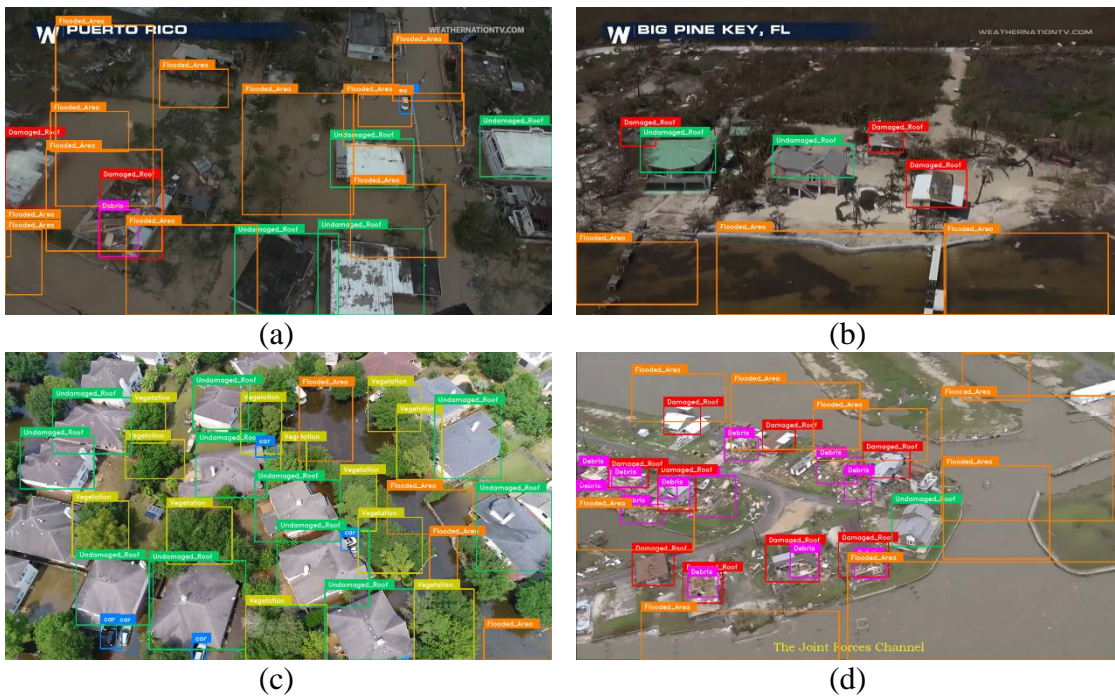


Figure 8. Examples of Volan2018 annotation examples [173].

All 8 videos Volan001-008 with location, duration, and hurricane name listed in Table 10. According to FAA [175], drones should fly at or below 300 feet which is the altitude limit set by most drone manufacturers. On the other hand, most helicopters fly at 1000 feet and above [176]. Considering the viewpoint altitude, Volan001, 002 and 003 are grouped as drone dataset (D), while Volan004, 005, and 006 are grouped as

helicopter dataset (H). Also, Volan007 (drone video) and Volan008 (helicopter video) from hurricane Michael, which are different from Volan001 through Volan006, serve as completely unseen test videos to validate the real-world applicability of this work beyond common testing and validation practices in computer vision. While in traditional computer vision, the same (already available) dataset is split into training and testing portions, here the goal is to train a CNN model in advance using past disaster data, and then test it on new disaster data captured from a different time and location.

Table 10. Volan2018 video name, event location, duration and instance amount per class [173].

Dataset	Number	Hurricane	Duration (s)	Location	Flooded Area	Undamaged Roof	Damaged Roof	Car	Debris	Vegetation
Drone (D)	001	Harvey	84	Southern TX	1,015	1,814	1,457	1,046	2,678	123
	002	Harvey	72	Houston, TX	1,572	1,174	871	612	1,653	0
	003	Harvey	1,333	Houston, TX	38,480	37,661	296	24,710	0	38,976
Helicopter (H)	004	Irma	88	Big Pine Key, FL	1,916	1,481	2,217	666	2,928	597
	005	Maria	219	Puerto Rico	3,774	4,003	3,416	3,129	3,986	2,005
	006	Harvey	317	Rockport, FL	8,975	7,544	7,112	1,389	11,412	0
Unseen	007	Michael	59	St Joe Beach, FL	0	1,657	1,779	896	1,712	246
	008	Michael	14	Mexico Beach, FL	0	410	339	0	410	0

In this Chapter, models are first trained and tested on Volan001 through Volan006 that cover six different locations in three hurricanes during the 2017 season (hurricanes Harvey, Maria, and Irma). Next, the trained models are then tested on Volan007 and Volan008 captured from the 2018 season (hurricane Michael) at two completely different locations (St Joe Beach and Mexico Beach, Florida) to assess the scalability and generalizability of the trained models to new situations.

The instance distribution of Volan001 through 006 is shown in Figure 9. In this Figure, each colored line represents one class, the x-axis indicates the frame number, and the y-axis indicates the number of instances. This distribution shows the variation of visible objects on the ground in camera's viewpoint as the vehicle (drone or helicopter) flies over different areas. For example, in Volan005, a helicopter flies over flooded areas in frame 1 through 360, followed by areas that are not flooded in frames 361 through 1,170, and then over flooded areas again in frames 1,171 through 2,460. A similar visual analysis can be performed for all classes in all six videos.



Figure 9. Volan001-006 instance distribution [173].

In order to quantify the class/instance diversity in Volan2018 dataset, two indicators are created to describe the instance frequency, namely, instance number (*IN*), i.e., total amount of instance per class in each images; and frame number (*FN*), i.e., the amount of frames that contain more than one particular class. Next, instance per frame (*IPF*), i.e., average instance amount per class in each frame is calculated by Equation 2.

$$\text{Instance Per Frame (IPF)} = \frac{\text{Instance Number (IN)}}{\text{Frame Number (FN)}} \quad (2)$$

All the *IPF* per class in each video are listed in Table 11 and average *IPF* for each classes considering all the videos (all the locations) is calculated in the last column. Comparing the *IPF* values for the same class across different videos (taken from different locations or at different times) reveals the significance of a certain type of damage or hazard across time (temporal scale) and locations (spatial scale), as viewed in aerial imagery taken by drone or helicopter. For example, Volan003 contains more flooded areas than other videos, while the location represented in Volan005 has suffered less damage (as indicated by the fewer number of damaged roofs) than other locations. The last column in the table lists the average *IPF* value for the entire Volan2018 dataset. Evidently, the most frequent classes in Volan2018 are flooded area (5.73), undamaged roof (4.68), and vegetation (3.15) whereas debris, car, and damaged roof do not appear frequently. The average *IPF* can assists future data collection by directing more attention to classes that are underrepresented (lower *IPF* values), thus helping balance the dataset.

Table 11. Volan2018 dataset instance per frame (IPF) [173].

	001	002	003	004	005	006	007	008	Average IPF
Flooded Area	5.72	5.84	3.72	3.49	10.35	5.25	-	-	5.73
Undamaged Roof	2.94	5.33	12.94	2.64	6.22	3.60	1.34	2.42	4.68
Damaged Roof	1.34	3.94	1.00	2.84	3.49	3.83	1.63	2.27	2.54
Car	3.44	1.33	2.72	1.04	3.83	1.88	1.00	-	2.18
Debris	2.26	1.93	-	3.15	1.70	2.81	2.01	3.92	2.54
Vegetation	2.54	-	8.86	1.43	1.49	-	1.42	-	3.15

4.1.2. Volan2018 data balancing

Initially, a model is trained on the whole Volan2018 dataset resulting in a trial model. However, this model is overfitting on Volan003, i.e., it performs very well only on testing data from Volan003 video but not others. As shown in Table 10, Volan3 contributes a large portion of images in the training data, hence the trial model is heavily biased towards Volan003. In order to overcome this imbalanced data problem, data pre-processing methods include up-sampling (augment rare data) and down-sampling (reduce frequent data) are often adopted [177]. Considering up-sampling Volan2018 will result an even larger size of data with images focusing on Volan001-006, which will lead to overfitting. In other words, poor performance on unseen data (e.g., Volan007 and 008). Down-sampling is implemented to balance Volan2018 by removing images from the data pool.

In order to reflect the data balance level, balance ratio (BR) calculated using Equation 3 is introduced, where $i_{n,k}$ indicates how many instances of class n frame k contains, the total number of classes is c , and the total amount of frames is f .

Moreover, $\bar{N} = \frac{\sum_{n=1}^c \sum_{k=1}^f i_{n,k}}{c}$ denotes the average instances amount for class c in f frames. By definition, the higher BR implies the a less balanced dataset. The most balanced dataset should have BR value at 0. This BR can be applied to describe the balance level of the whole Volan2018 or individual video dataset (e.g., Volan001). Table 12 shows the BRs for Volan001-006, suggesting the most balanced video is Volan006 with BR 0.36 and the most unbalanced video Volan003 with BR 0.79.

$$\text{Balance Ratio (BR)} = \frac{\sum_{n=1}^c |\sum_{k=1}^f i_{n,k} - \bar{N}|}{c \times \bar{N}} \quad (3)$$

Down sampling in the content of this experiment indicates removing images with frequent classes from Volan001-006 thus balance the classes among them. In order to implement under-sampling, two parameters are defined in the follow. Diversity balancing threshold (*DBT*) expressed in Equation 4 where c_k represents the class types in frame k . By definition, *DBT* is in the range from $\min_{1 \leq k \leq f} (c_k)$ which is the minimum class types frame k has, to $\max_{1 \leq k \leq f} (c_k)$ which is the maximum class types in frame k . Quantity balancing threshold (*QBT*) defined with Equation 5. Here, $i_{n,k}$, n , and k , are similar to the definition in Equation 4, c indicates the total class types, and f is the total frame number.

$$\min_{1 \leq k \leq f} (c_k) \leq \text{Diversity Balancing Threshold (DBT)} \leq c_{\max} = \max_{1 \leq k \leq f} (c_k) \quad (4)$$

$$\min_{1 \leq n \leq c} (\min_{1 \leq k \leq f} i_{n,k}) \leq \text{Quantity Balancing Threshold (QBT)} \leq \max_{1 \leq n \leq c} (\max_{1 \leq k \leq f} i_{n,k}) \quad (5)$$

Frame k in one video (frame $0\dots f$) is removed if $c_k < DBT$, i.e., frame k contains less class of c_k than the DBT value. While balancing Volan001-006, DBT is set equal to 1, i.e., frames containing no instances of any class are removed. Considering the remaining frames, frame k is removed if $i_{n,k} > QBT$, i.e., k contains more instances of class n . All the possible QBT values are implemented and the BR values and each operation is recorded. Eventually, the set of the remaining frames corresponding to the lowest BR (i.e., the most balanced) is saved with the BR_{min} . This procedure saves images with more diverse classes. Table 12 below shows down sampling results listing the initial BR, initial frames, minimum BR, minimum frames, and the final selected frames from each video. In this table, it is observable that the amount of the minimum frames for each subset is different. For instance, the minimum frame amount of Volan003 in subset D is 3,084 but for Volan002 it is 925. In order to select images representing different locations equally, the lowest values of the minimum frames in each subset (D and H) are selected, i.e., 925 for D and 519 for H. For each remaining video (Volan001 and 003) in D, 925 frames are randomly selected from the minimum frames. These selected frames with the 925 frames from Volan002 form a new subset named drone balanced (DB). Similarly, subset helicopter balanced (HB) is created by selecting 519 frames from Volan004-006. Data balancing of Volan003 has led to the largest frame removal (39,065 frames) with the greatest BR drop of 0.55.

Table 12. Data balancing results for each video within Volan2018 dataset [173].

Viewpoint	Video	Initial BR	#Frames @ Initial BR	BR_{min}	#Frames @ BR_{min}	#Frames Selected
D	Voaln001	0.49	2,520	0.36	1,045	925
	Voaln002	0.51	2,160	0.47	925 (min.)	925
	Voaln003	0.79	39,990	0.24	3,084	925
H	Voaln004	0.60	6,570	0.29	519 (min.)	519
	Voaln005	0.64	2,640	0.24	3,136	519
	Voaln006	0.36	9,510	0.20	2,813	519

Frames in dataset D and H that overlap with frames from DB and HB are excluded resulting in two unbalanced datasets: drone unbalanced (DU) and helicopter unbalanced (HU). Figure 10 demonstrates the data splitting scheme. In total, Volan2018 dataset is divided into subsets DU, HU, DB, HB, Volan007 and Volan008. Unseen videos Volan007 and 008 are reserved to test the proposed framework on real-world practices, i.e., training CNN models on disaster data that have happened and deploying them on newly happening disasters. Each subset from Volan2018 is further split into three parts training (60%), validation (20%), and testing (20%) by random selection.

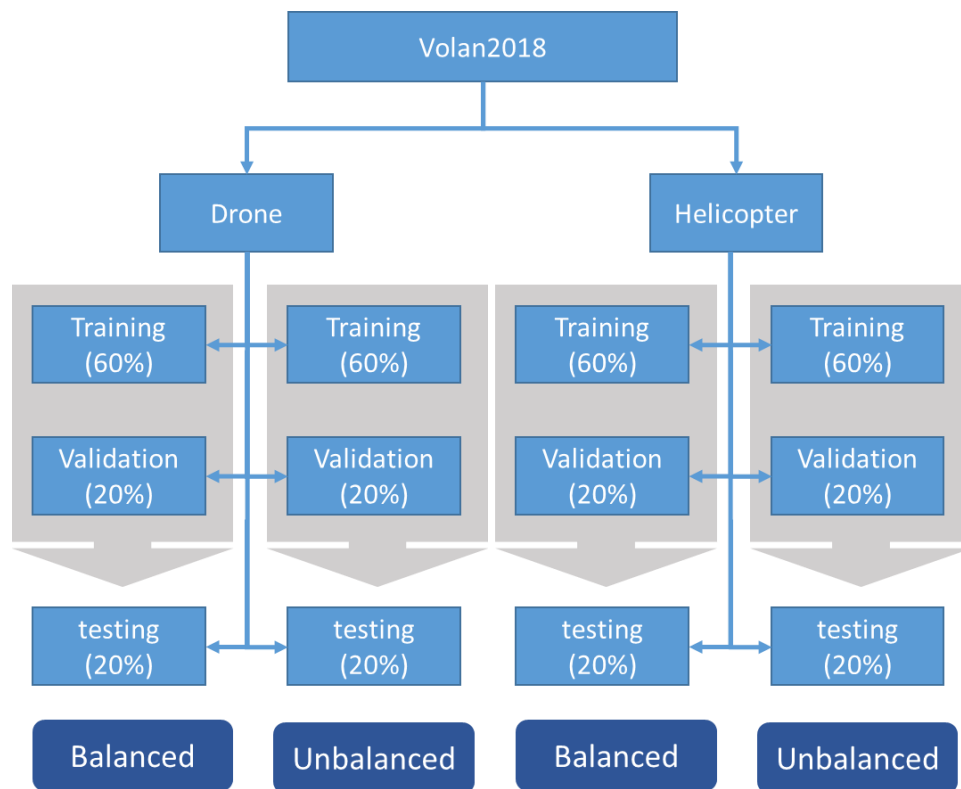


Figure 10. Volan2018 data splitting.

4.2. YOLO v2 architecture

YOLO v2 [79] is a CNN model that takes RGB imagery as input, and outputs predictions in form of target objects' classes and their coordinates in the image. Figure 11 demonstrates the structure which is comprised of 23 layers including convolution and max pooling layers, each with kernel, normalization, and activation functions. A convolution layer has kernels with parameters in each kernel cell, and extracts features such as shape and color from the input. A convolution layer has kernels with parameters in each kernel cell, and extracts features such as shape and color from the input. Each kernel applies convolution computations by a stride to cover the entire image and

outputs one channel. Taking the first convolution layer in Figure 11 as an example, after applying 32 kernels with stride 1 on the input ($416 \times 416 \times 3$), the output is a 32 channel matrix. Each layer computes the input and then passes the result on to the next layer until the last output layer. Altogether, the model divides the input image into a 13×13 grid, and at the output layer each grid predicts 5 anchor boxes. Each anchor box contains x and y coordinates, width, height, confidence value (confidence with which the anchor box contains an object), and probability for each class. Since this model predicts six classes, in each anchor box, the output corresponding to a single image (i.e., video frame) contains $13 \times 13 \times 5 \times (5+6) = 9,295$ predicted values.

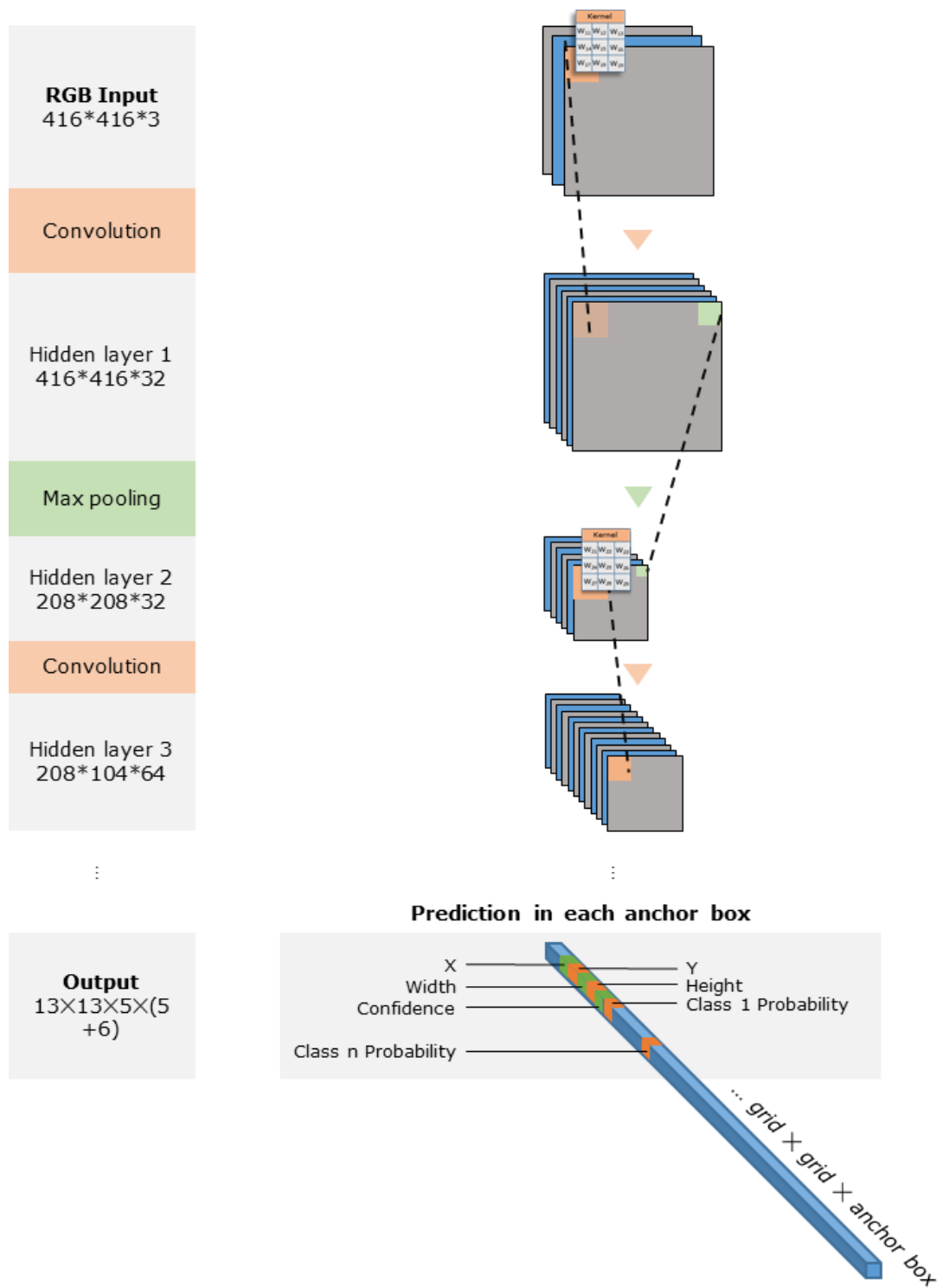


Figure 11. YOLO v2 architecture [173].

The key for a model to produce the correct prediction is the weights in the kernels. Training CNN models involves backpropagation, a process during which kernel weights are updated. In each iteration, the difference between ground truth and prediction is calculated by loss function, L , as laid out in Equation 6. In this Equation, x_i , y_i , w_i , h_i , c_i , and p_i represent predicted x and y coordinates of the center of the box, width, height, objectness score, and class probability, respectively, while \hat{x}_i , \hat{y}_i , \hat{w}_i , \hat{h}_i , \hat{c}_i , and \hat{p}_i represent ground truth x and y coordinate of the center of the box, width, height, objectness score, and class probability, respectively, and λ_{box} represents the box scale factor. The function $BC(\hat{a}_i, a_i)$ (Equation 7) represents the binary cross-entropy where \hat{a}_i and a_i are ground truth and predicted values, respectively. During training, backpropagation updates the weights systematically using loss function and regression function to yield better prediction until the optimum point.

$$\begin{aligned}
L = & \lambda_{box} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{i,j}^{obj} [BC(\hat{x}_i, x_i) + BC(\hat{y}_i, y_i)] + \lambda_{box} \sum_{i=0}^{S^2} \\
& \sum_{j=0}^B \mathbf{1}_{i,j}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{i,j}^{obj} [BC(\hat{c}_i, c_i)] + \sum_{i=0}^{S^2} \\
& \sum_{j=0}^B \mathbf{1}_{i,j}^{noobj} [BC(\hat{c}_i, c_i)] + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbf{1}_{i,j}^{obj} [BC(\hat{p}_i, p_i)]
\end{aligned} \tag{6}$$

$$BC(\hat{a}_i, a_i) = -a_i \log(\hat{a}_i) - (1 - a_i) \log(1 - \hat{a}_i) \tag{7}$$

Transfer learning [178] is a model training scheme that leverages the pre-trained weights on large dataset such as VOC [124] and COCO [131], followed by re-training the model on costumed dataset (often smaller). Since the pre-trained weights already have the capability to extract key features (edge, color, and pattern), transfer learning is more effective than training only on costumed data. Therefore this experiment adopts

transfer learning using the YOLO v2 pre-trained weights on VOC and COCO as shown in Figure 12. This approach also requires fine-tuning of the last layer to match the class number due to Volan2018 targets on 6 classes whereas COCO and VOC covers 80 and 20 classes, respectively. After loading the pre-trained weights, only the last layer of the model is updated on the training data with the rest 22 layers frozen. Learning rate is set at 10^{-3} with epoch (iteration) set as 25. Next, the frozen layers are released to let all the weights to update with a 10^{-4} leaning rate. When the loss L does not drop within 3 epochs, the learning rate is reduced by 5×10^{-5} . Validation loss is calculated to check and monitor the performance during training by deploying the intermediate trained weights on validation data for every epoch. Provided that validation data is not included in training, validation procedure therefore simulates the deployment on unseen data. If validation loss L does not drop in 10 consecutive epochs, the training is terminated and the best weights are saved at the local minimum to avoid overfitting. Fully trained CNN models are tested on the testing data to evaluate prediction accuracy.

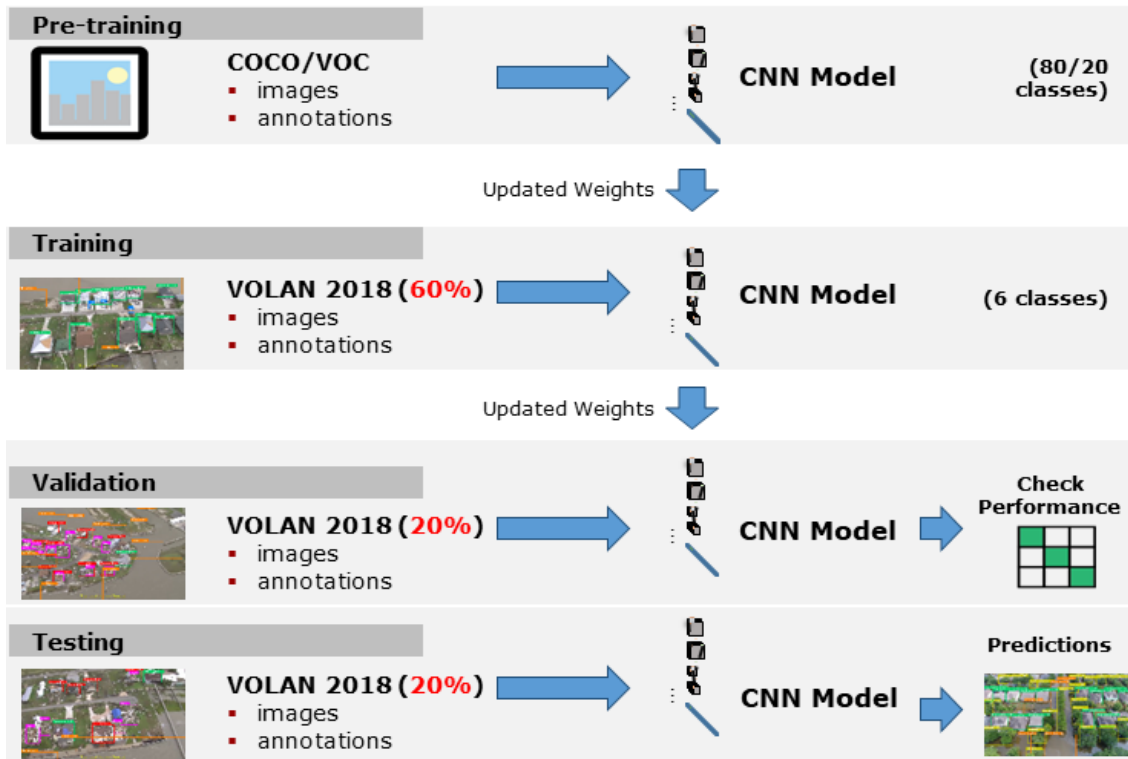


Figure 12. Transfer learning [173].

As previously stated, Volan2018 is split into DB, DU, HB, and HU, each subset is divided into training, validation, and testing portions. In order to investigate the influence of the pre-trained weights, models for four subsets are trained using transfer learning based on pre-trained weights from VOC and COCO. Altogether, 8 YOLO v2 models are trained based on different combinations to investigate the factors including viewpoint altitude, pre-trained weights, and data balance. All the fully trained CNNs are tested on all the testing portions and Volan007 and 008. Table 13 lists the numbered models with their pre-trained weights, training, validation, and testing data. Please note, Volan007 and 008 are not included in the training or validation steps in order to test the performance in the real-world situation.

Table 13. Pre-trained weights, training, and validation combinations of 8 YOLO v2 CNN models [173].

Model	Pre-trained weights	Trained on	Validated on	Tested on
1	COCO	DB	DB	(for all models) DB, DU, HB, HU, Volan007, Volan008
2	VOC	DB	DB	
3	COCO	DU	DU	
4	VOC	DU	DU	
5	COCO	HB	HB	
6	VOC	HB	HB	
7	COCO	HU	HU	
8	VOC	HU	HU	

4.3. Performance measurement

As shown in one detection example in Figure 13, provided with the ground truth bounding box and the predicted box of the same class, the intersection over union (IoU) is calculated with Equation 8. Here, *intersection area* indicates the pixel area where the prediction and ground truth boxes overlap, and union area is the combined area of prediction and ground truth boxes. By definition, the IoU value of any given prediction ranges from 0 to 1. While IoU = 100% is the most ideal result, current CNN researches have not achieved 100% for all detections. Hence in this work, a detection is considered as successful while $\text{IoU} \geq 50\%$, which is a common practice in VOC challenge.

$$\text{Intersection over Union}(\text{IoU}) = \frac{\text{Intersection Area}}{\text{Union Area}} \quad (8)$$

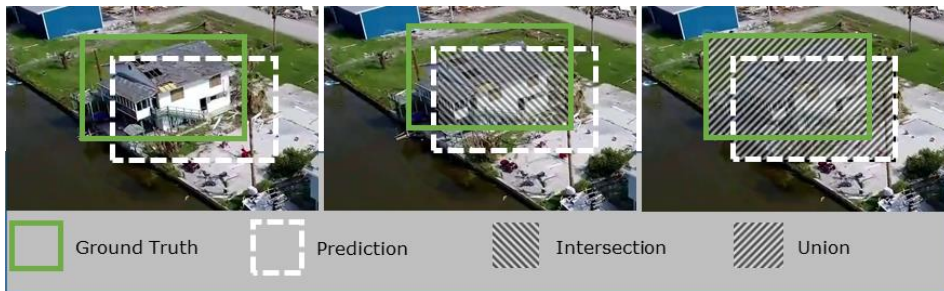


Figure 13. Union (IoU) example [173].

As Figure 14 (a) shows, a successful detection ($\text{IoU} \geq 50\%$) is defined as a true positive (TP). In addition, there could be two types of wrong detection as shown in Figure 14 (b) and 14 (d); false positive (FP) when there is no ground truth but a detection is produced, and false negative (FN) when the ground truth is present but the model does not detect it. Cases such as that shown in Figure 14 (c) can be classified as both FP and FN when the IoU is less than 50%. Of note, true negative (TN) cases that occur when there is no ground truth and the model indicates no object in the image, are not considered. Subsequently, precision, recall, and F1 are calculated by Equations 9, 10, and 11. By definition, precision is calculated by dividing the total number of TP cases by the total number of detection boxes. In other words, precision indicates the percentage of correct detections. The recall value, on the other hand, is computed by dividing the total number of TP cases by the total number of ground truth boxes, therefore, describing the percentage of correctly detected ground truth cases.

$$\textit{precision} = \frac{TP}{TP + FP} \quad (9)$$

$$\textit{recall} = \frac{TP}{TP + FN} \quad (10)$$

$$F1 = 2 \times \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}} \quad (11)$$

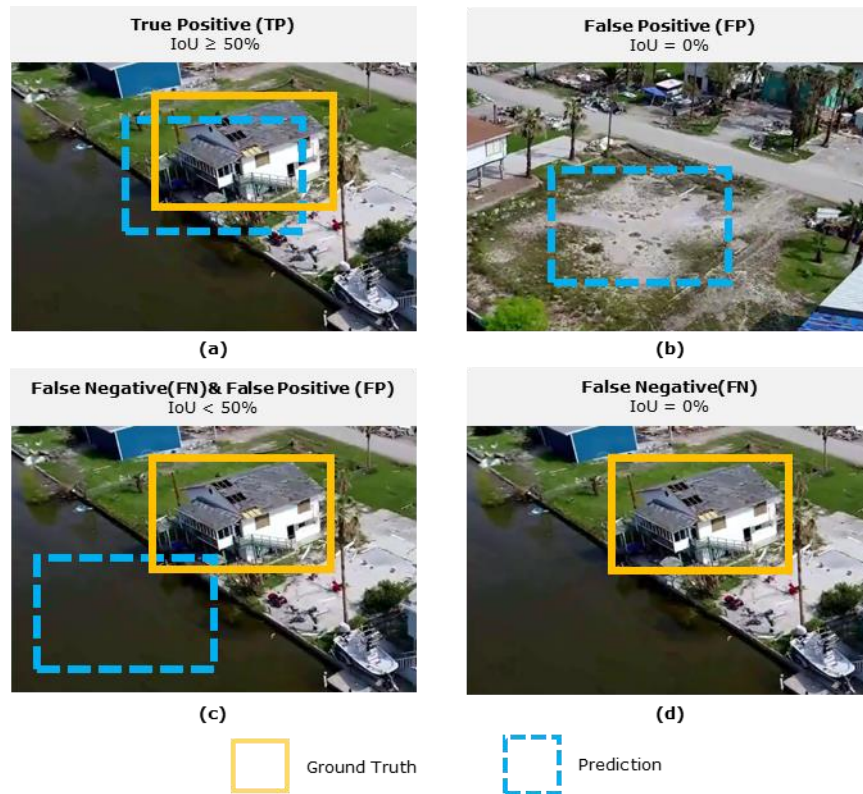


Figure 14. TP, FP, and FN prediction examples [173].

Precision-recall curve is one of the common methods to evaluate the CNN performance. For each detection, the precision and recall are computed and reflected on a graph with X-axis indicating the recall value and Y-axis representing the precision value. The area below the precision-recall curve is defined as mean average precision (mAP). As shown in Figure 15, the best model produces all TP detection and no FP and FN, i.e., all the ground truth are correctly detected (mAP=100%). On the contrast, the worst model with mAP 0 % cannot produce any TP, i.e., all detections are either FP and FN.

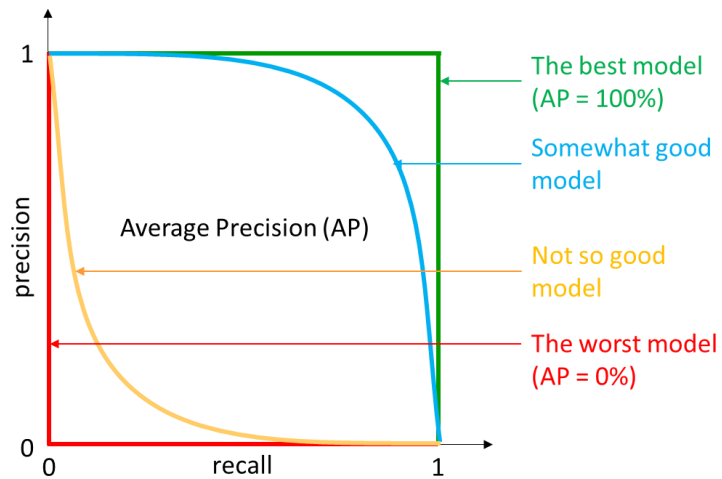


Figure 15. Precision-recall curve interpretation [173].

4.4. Experiments and results analysis

In the prediction stage, threshold (i.e., minimum class probability) is commonly used to eliminate low-confidence detection boxes and produce human understandable output. For example, a model with the threshold value of 0.1 only outputs detections with class probability higher than 0.1. To investigate the suitable threshold for real-world applications, model2 (trained on DB based on weights from VOC) is tested on Volan007 video, and record precision, recall, and F1 score for threshold values ranging from 0.0 to 0.9 with 0.1 increments. Sample results for classes debris and undamaged roof are shown in Figure 16. According to this Figure, larger threshold values lead to lower recall and higher precision, indicating that the model produces fewer detections, therefore, fewer false positives and more false negatives. Theoretically, the best threshold is at the point where precision, recall, and F1 score converge. However, in cases where the object of interests are highly valuable or not detecting them may have

severe consequences (e.g., trapped people on the roof of a flooded home), high recall is necessary even with low precision. On the other hand, different information users may have different opinions about the value of different objects, thus leading to varying perceptions of precision and recall values. Besides, different classes and test data have different corresponding precision, recall and F1 curves. In conclusion, the selection of precision and recall, essentially the threshold value, primarily depends on the needs and expectations of the end user.

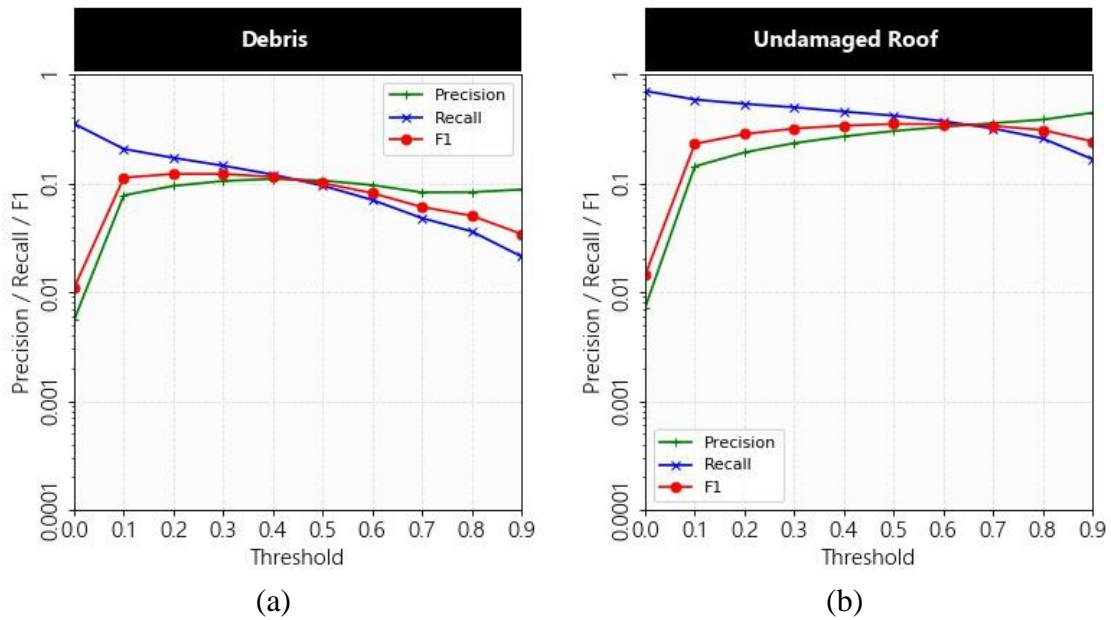


Figure 16. Precision, recall, and F1 score Model 2 tested on Volan007 for (a) debris, and (b) undamaged roof classes [173].

The results of testing all 8 CNN models on DB, DU, HU, HB, Volan007, and 008 with threshold 0 (all detections) are shown in Table 14. As indicative by these results, models trained on drone videos tend to perform better when tested on drone videos, while models trained on helicopter videos tend to perform better when tested on

helicopter videos. This observation supports the influence of viewpoint altitude on model performance. The best mAP for D and H set is model3 (74.48% mAP, trained on DU based on COCO weights and tested on DU) and model7 (80.69% mAP, trained on HU based on COCO weights and tested on HB), respectively. These promising results support that automatically extracting information from aerial views is feasible.

Although training and testing images in D and H are different, they are still extracted from similar videos that closely resemble one another (e.g., same disaster, or same location). In practice, the test video could be captured from an utterly different altitude, camera, flying speed, disaster event, location, time, and lighting condition. A robust model must perform adequately in detecting ground objects when applied to a completely new (unseen and drastically different) video footage. Therefore, trained models are also tested on Volan007 (drone video) and Volan008 (helicopter video) that were not previously seen by the models. It can be seen in Table 14, that models pre-trained on VOC and trained on the balanced data tend to perform better than other models. For example, for drone footage, model2 (trained on DB based on VOC weights) performs best (24.50% mAP) on Volan007 video whereas for helicopter footage, model6 (trained on HB with VOC weights) performs best (13.88% mAP) on Volan008 video. Figure 17 displays detection examples of model2 tested on randomly selected input.

Table 14. 8 model mAP (%) tested on 8 testing subsets (* denotes the best performance in each subset) [173].

Testing	model1	model2	model3	model4	model5	model6	model7	model8
DB	67.37	65.30	71.84	69.66	5.81	4.54	6.85	4.36
DU	41.17	40.00	74.48*	72.23	6.79	5.02	7.34	6.37
HB	4.59	4.50	3.83	3.74	61.37	52.07	80.69*	79.73
HU	2.92	2.40	2.27	1.72	31.86	25.83	78.72	76.92
Volan007	18.17	24.50*	16.67	12.57	1.26	1.65	1.24	0.62
Volan008	11.91	6.40	2.16	1.62	12.07	13.88*	12.30	10.43



Figure 17. Model2 detection examples [173].

Figure 18 displays the precision-recall curves for the best model (i.e., Model 2) tested on DB, DU, HB, HU, Volan007 and 008 testing data. For similar data, this model 2 achieved 65.30% and 40.00% mAP on testing data from DB and DU, respectively. For individual classes performances, class debris, damaged roof, and vegetation have the highest AP of 81.77%, 53.27%, and 93.39% while tested on DB, respectively. Looking at the performance on unseen data, Model 2 has 24.50% mAP. Particularly, class vegetation has the highest AP of 67.59%, followed by undamaged roof with 32.81%, and debris with an AP of 17.58%. In conclusion, the best model is capable of accurately

predicting object classes including vegetation, undamaged roof, and debris. The testing precision-recall graphs of all other models are stated in displayed the Appendix A.

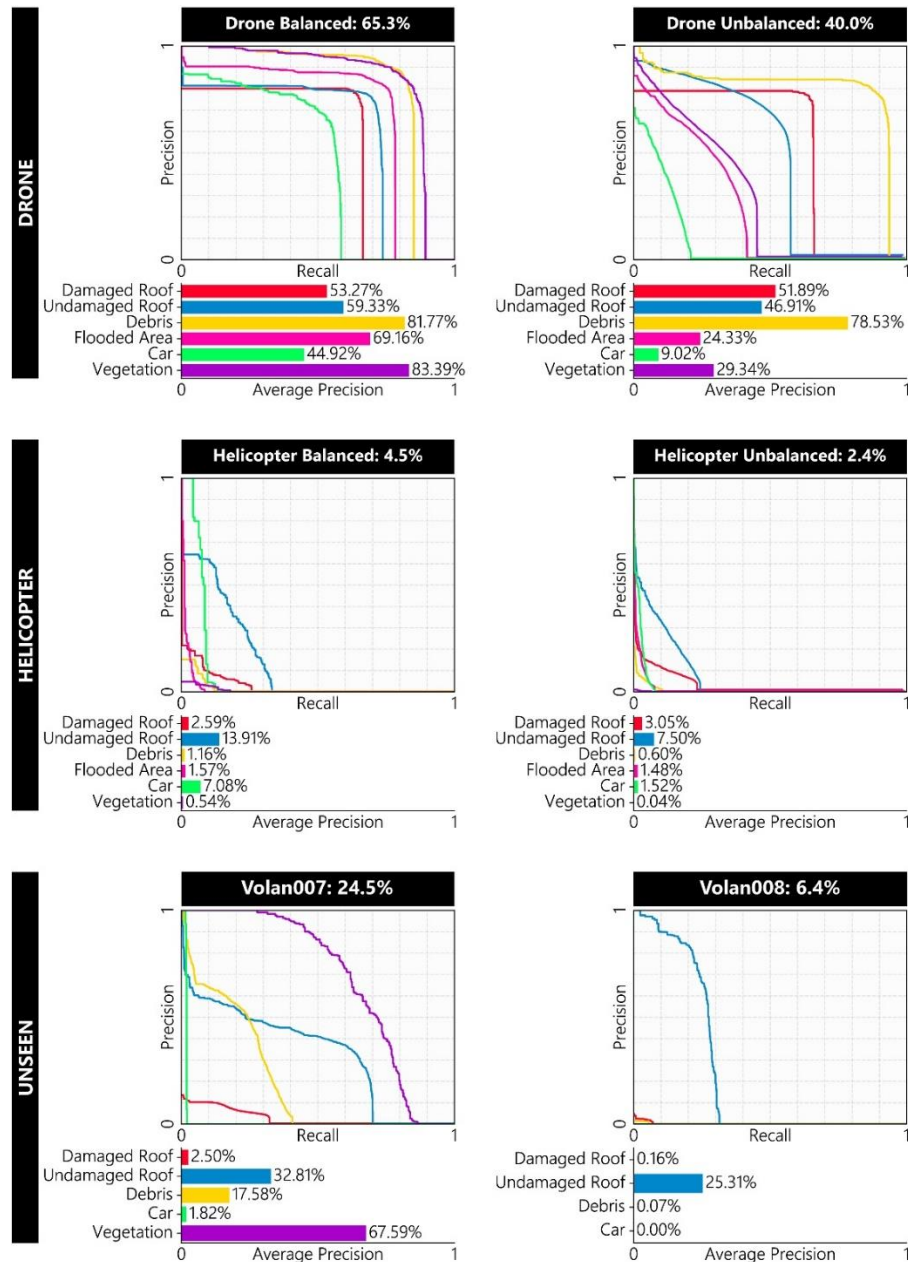


Figure 18. Model2 testing results on DB, DU, HB, HU, Volan007, and Volan008 subsets [173].

The effect of three key factors, namely the viewpoint altitude (low for drone vs. high for helicopter), data balancing (balanced vs. unbalanced), and pre-trained dataset (i.e., COCO vs. VOC) on model performance is investigated. Table 15 summarizes the results of statistical analysis performed on all eight models with respect to viewpoint altitude. For each trained model, a comparison is made between the performance of that model, when tested on Volan007 (drone) and Volan008 (helicopter) videos. For each comparison listed in Table 15, the general hypothesis is that training and testing on videos captured from the relatively same altitude yields best results. In other words, a model trained on D (H) dataset tends to show a statistically better performance when tested on D (H) dataset. Mathematically, this can be expressed by the null hypothesis H_0 that there is no difference in the mAP between the results when the mode is tested on Volan007 (u_{007}) and Volan008 (u_{008}), i.e., $u_{007} = u_{008}$. The alternative hypothesis or H_A is that $u_{007} > u_{008}$ (for models trained on drone video, i.e., Models 1, 2, and 3) and $u_{008} > u_{007}$ (for models trained on helicopter video, i.e., models 4, 5 and 6). For each comparison, the confidence level is set at 99%, test data sample is produced by randomly selecting two-thirds of the entire frames in each video (Volan007 or Volan008), and mAP is measured and averaged over 50 iterations. Next, a two-sample one-tail t -test is run to evaluate the significance for each pair. Considering the first row of Table 15 for instance, results show that model1 (trained on D) performs with an average mAP of 18.20% when tested on Volan007 (captured by drone), and an average mAP of 11.74% when tested on Volan008 (captured by helicopter), and this difference in mAP is

statistically significant at 99% confidence level. The same trend is observed for all eight models (training and testing on video taken from relatively same altitude yields better results). However, cross training and testing, i.e., model trained on D dataset and tested on H dataset or vice versa, leads to very low accuracy. Therefore, it can be concluded that the viewpoint altitude is a critical factor that must be considered when building and deploying CNN models.

Table 15. Statistical analysis of the influence of viewpoint altitude on model performance ($p < 0.01$) [173].

Model	Test data	Viewpoint	mAP (%)	STDV	<i>t</i> -value	Hypothesis
1	Volan007	Drone	18.20	0.520	74.27	$u_{007} > u_{008}$
	Volan008	Helicopter	11.74	0.317		
2	Volan007	Drone	24.54	6.372	252.73	$u_{007} > u_{008}$
	Volan008	Helicopter	4.26	0.367		
3	Volan007	Drone	18.45	0.485	263.97	$u_{007} > u_{008}$
	Volan008	Helicopter	0.15	0.017		
4	Volan007	Drone	12.69	0.544	137.75	$u_{007} > u_{008}$
	Volan008	Helicopter	1.69	0.130		
5	Volan007	Drone	1.278	0.064	-303.28	$u_{008} > u_{007}$
	Volan008	Helicopter	12.08	0.241		
6	Volan007	Drone	1.66	0.07	-145.31	$u_{008} > u_{007}$
	Volan008	Helicopter	14.01	0.591		
7	Volan007	Drone	1.22	0.042	-325.86	$u_{008} > u_{007}$
	Volan008	Helicopter	12.37	0.236		
8	Volan007	Drone	0.63	0.022	-224.80	$u_{008} > u_{007}$
	Volan008	Helicopter	10.50	0.307		

The BR parameter was introduced to measure the extent of data balance in individual dataset. Table 16 demonstrates the influence of BR balancing and pre-trained

weights when tested on unseen data. The proposed data balancing method excludes video frames from the training dataset and select same amount of data representing different locations equally. As a result, the training time reduced 14 to 17 times compare to training without this method. In particular, when pre-trained on COCO [131], reducing the size of training dataset affects slightly on performance (1.5% increase for D and 0.23% decrease for H subset). On the other hand, by pre-training on VOC [124], data balancing not only does decrease the training time, but also improves the performance by 11.89% and 3.54% for D and H subsets, respectively. In this analysis, training time is based on Intel Xeon E5-2680 v4 2.40GHz 14-core CPU, 128GB RAM, and NVIDIA K80 (12 GB) GPU [179]. Overall, it is observed that models trained on balanced training subset with pre-trained weighs from VOC [124] (i.e., Model 2 from the D subset, and Model 6 from the H subset) outperform other models.

Table 16. Performance and training time analysis [173].

Model	Train data	Test data	Pre-trained on	Data balance	# Training frames	mAP (%)	Training time (hr)
1	D	Volan007	COCO	B	1,665	18.17	6.93
3			COCO	U	25,137	16.67	121.79
2			VOC	B	1,665	24.50*	7.67
4			VOC	U	25,137	12.57	118.98
5	H	Volan008	COCO	B	934	12.07	2.63
7			COCO	U	10,297	12.30	39.04
6			VOC	B	934	13.88*	2.52
8			VOC	U	10,297	10.43	37.33

4.5. Summary

With the goal of leveraging CNNs to extract valuable information from aerial disaster visual data, a list of valuable object classes was first created. Followed by data-mining the internet to acquire data that contains those classes resulted in a bounding box annotated dataset named Volan2018. This dataset consisted of 8 videos from various hurricanes at different locations in different time. A novel approach using BR, DBT, and QBT to balance and select most equally representative data was proposed and tested. In order to investigate the influence factors including viewpoint altitude, data balancing, and pre-trained weights, 8 CNN models based on YOLO v2 architecture were trained, validated, and tested on Volan2018 subsets. Results showed that CNN models tend to perform better on data that is collected at the similar viewpoint altitudes and perform poorly on data from different altitudes. The proposed data balancing approach was capable of improving the CNN model performance on unseen data while reducing training time by up to 17 times. Model2 trained on drone balanced data using pre-trained weights from VOC achieved the best performance of 24.50% mAP on unseen data. While training and testing on similar data (Volan001-006), best model3 (trained on DU based on COCO weights) accomplished mAP of 74.48% and model17 (trained on HU based on COCO weights) achieved a mAP of 80.69%. These result suggested the proposed framework has the potential to extract accurate information in real-time, thus was capable of assisting disaster information retrieval.

5. IMAGE SEMANTIC SEGMENTATION

This Chapter presents image semantic segmentation by describing the work completed on dataset preparation and balancing, semantic segmentation CNN model architecture, performance measurement, experiments conducted, and analysis of results.

5.1. Volan2019 data description

5.1.1. Data collection and annotation

The second dataset, Volan2019 (under Creative Common license), is web-mined from YouTube by searching key words including: “disaster”, “hurricane”, “drone”, “aerial”, and “aftermath” Altogether, Volan2019 contains 16 videos covering locations in Florida, Texas, Massachusetts, South Carolina, California, and the British Virgin Islands. Table 17 details the corresponding disaster event, recording vehicle, and video duration of all the videos, which are numbered from Volan201-216. The majority of the Volan2019 dataset reflects hurricanes Michael, Harvey, Irma, Matthew, and Florence during the 2017-2018 hurricane seasons. California wildfire Tubbs in 2017 is also included as Volan213. All frames in the videos are extracted resulting in an image pool of 112,050 numbered images following the same sequence as the videos. Given that annotating one image costs approximately 20 minutes (37,516 hours to annotated all images), every 1 out of every 128 frames (or 1 frame in every 4.3 seconds) is extracted for further consideration. This operation downsizes the dataset to 875 images without losing much content diversity. Table 17 documents the total extracted frame number from each video.

Table 17. Description of Volan2019 dataset videos and number of annotated frames (D: UAV, H: Helicopter).

Video	D/H	Date	Length (s)	Location	Disaster event	# Frames
201	D	Oct 11, 2018	78	Mexico Beach, Florida	Hurricane Michael	18
202	D	Aug 29, 2017	163	Friendswood, Houston, Texas	Hurricane Harvey	30
203	D	Jun 23, 2018	431	British Virgin Islands, UK	Hurricane Irma	101
204	D	Dec 29, 2016	119	Orlando, Florida	Hurricane Matthew	28
205	D	Aug 27, 2017	73	Houston, Texas	Hurricane Harvey	17
206	D	Oct 16, 2018	39	Beacon Hill, Boston, Massachusetts	Hurricane Michael	9
207	H	Sep 18, 2018	431	South Carolina	Hurricane Florence	67
208	D	Oct 14, 2018	50	Beacon Hill, Boston, Massachusetts	Hurricane Michael	12
209	H	Aug 31, 2017	1335	Rockport Texas	Hurricane Harvey	312
210	H	Oct 11, 2018	18	Mexico Beach, Florida	Hurricane Harvey	4
211	D	Nov 20, 2018	61	St Joe Beach, Florida	Hurricane Michael	14
212	D	Nov 6, 2017	215	Fountain Grove, California	Wildfire Tubbs Fire	50
213	D	Sep 4, 2017	387	Bear Creek, Houston, Texas	Hurricane Harvey	76
214	D	Aug 27, 2017	199	Houston, Texas	Hurricane Harvey	46
215	D	Nov 17, 2018	312	Unknown	Unknown/Tents	73
216	D	Oct 15, 2018	87	Panama City, Florida	Hurricane Michael	18

Next, all the selected images are split into training (60%), validation (20%), and testing portions (20%). Note that the burnt houses in Volan212 (California Tubbs wildfire) are annotated but excluded due to the fact that their appearances are drastically different from hurricane damages. However, the remaining annotations from Volan212, e.g., car and vegetation, are used for model training and testing. Labelbox [180] is employed to annotate every image with classes stated in Table 8, namely roofs (damaged and undamaged), car, people, boat, flooded area, debris, vegetation, and road. As examples shown in Figure 19, pixel level masks are drawn corresponding to the instances. In doing so, individual objects (e.g., damaged roof, undamaged roof, car, boat, people) are annotated separately and bulk objects (e.g., flooded area, vegetation, road, debris) are marked as one instance. For example, there are 6 roofs, 1 flooded area, 7 vegetation, and 1 car in Figure 19 (a). Bulk but continuing classes are annotated separately, e.g., there are two instances of flooded areas in Figure 19 (b). Similarly, Figure 19 (c) shows 67 undamaged roof, 2 car, 46 vegetation, and 33 flooded area instances, and Figure 19 (d) contains 12 undamaged roof, 2 car, 3 flooded area, and 21 vegetation instances. Compared to Volan2018, the annotation in this dataset is more accurate especially for bulk objects, i.e., they are not divided into smaller bounding boxes.

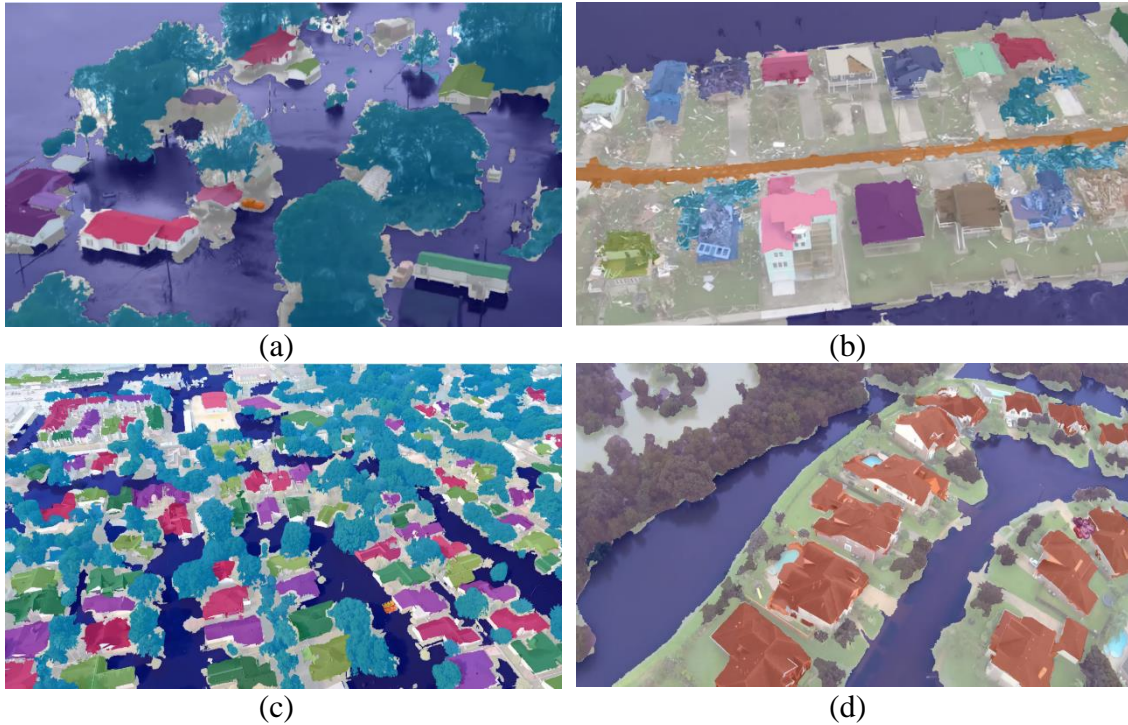


Figure 19. Sample annotations of classes roof (damaged and undamaged), vegetation, car, flooded area, and road in Volan2019 dataset.

Figure 20 illustrates the instance distribution, where the X-axis represents the image number, the Y-axis indicates different classes, and the instance amount for each class goes upwards in the Z-axis. In addition, the bar chart at the bottom of the figure indicates the percentage of frames categorized by their original videos, which represents different locations and disasters. Observably, video Volan203 (12%) and 209 (36%) contribute the larger portion of frames. In total, Volan2019 contains 7,457 undamaged roof (UD), 1,637 flooded area (FA), 501 damaged roof (DR), 815 car (C), 2,994 boat (B), 662 people (P), 540 debris (D), 762 road (R), 5,191 vegetation (V). In consistency as Volan2018, class flood area, vegetation, and undamaged roof are the frequent classes, as oppose to damaged roof and debris.

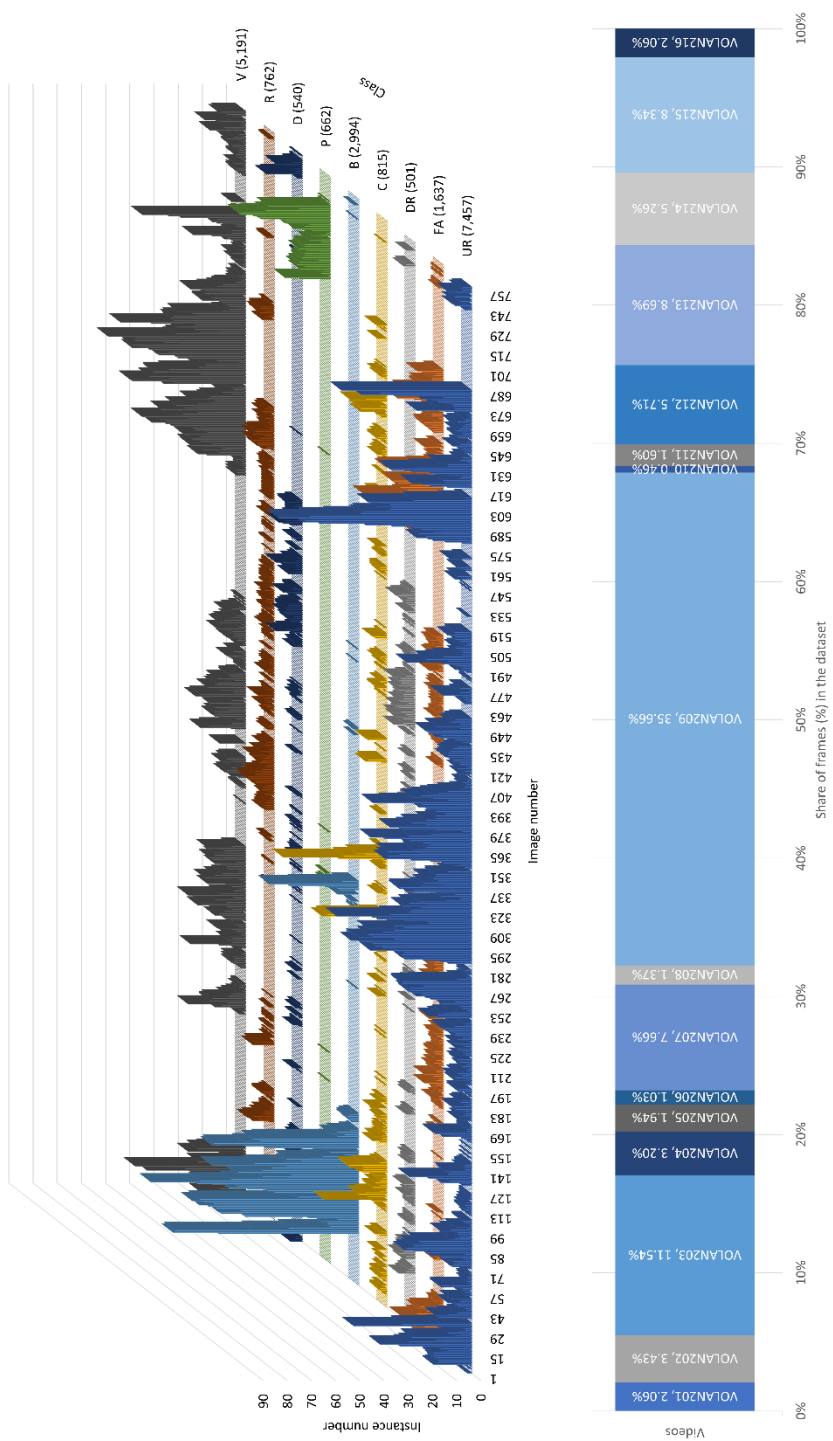


Figure 20. Volan2019 instance and class distribution per video frame.

5.1.2. Multiclass data augmentation

As shown in Figure 20, an important implication of the annotated Volan2019 is not balanced, i.e., certain classes (a.k.a., majority classes) such as UR and V have much more instances than other classes (a.k.a., minority classes) such as DR and C). This is reflected as the distribution peaks in Figure 20. In Subsection 4.1.2, balance ratio (BR) is introduced to quantify the degree of the balance level, followed by down-sampling Volan2018 resulting in a BR drop from 0.79 to 0.24. To put the BR value in perspective, the BR is 0.87 for Volan2019, 0.84 for COCO, and 0.54 for VOC. According to Huh et al., an effective image size per classes should be greater than 500 in order to pursue accurate results [181]. Considering that the total amount of images in Volan2019 (only 875 images) is relatively small, down sampling Volan2019 may further shirk the dataset losing the necessary information. Therefore, up sampling is implemented for the minority classes with the goal to achieve a lower BR value for the entire dataset. To this end, images that are relatively unbalanced with minority classes are chosen to augment. The reason behind is that augmenting images will simultaneously increase the instance amount for all the existing classes, which undermines the balancing purpose. To start, the dominating class for each image is determined by selecting the classes which comprise most instances for each image. For example, Figure 21 (a) represents an image that has class P as the dominating class with BR 1.03. Figure 21 (b) shows FA, V, and D as dominating classes for a image with a BR 0.74. The image corresponding to Figure 21 (a) is more unbalanced than Figure 21 (b), hence it is more suitable for data augmentation to increase the instance amount for class P.

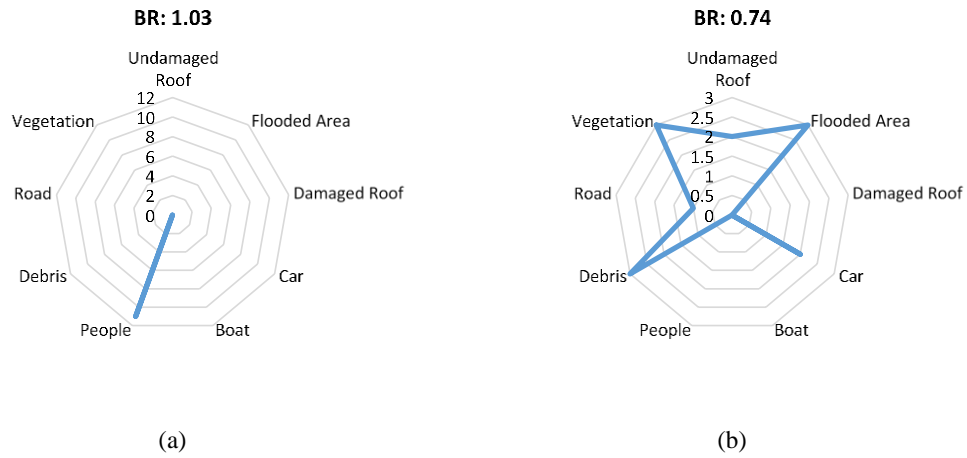


Figure 21. Selecting video frame candidates for data augmentation based on balance ratio.

In order to simulate practical circumstances, augmentation effects including zoom in, motion blur, pixel drop out, add cloud, and color equalization (different time during the day) are considered. For example, motion blur simulates images taken while the recording vehicle is moving at a high speed, which often happens in practice. These 5 different effects form an operation bag with 32 possible combinations, e.g., zoom in with add clouds. Figure 22 illustrates the augmentation example which contains original image, original annotation, annotation overlaid on original image, augmented image, augmented annotation, augmented annotation overlaid on augmented image (from left to right).

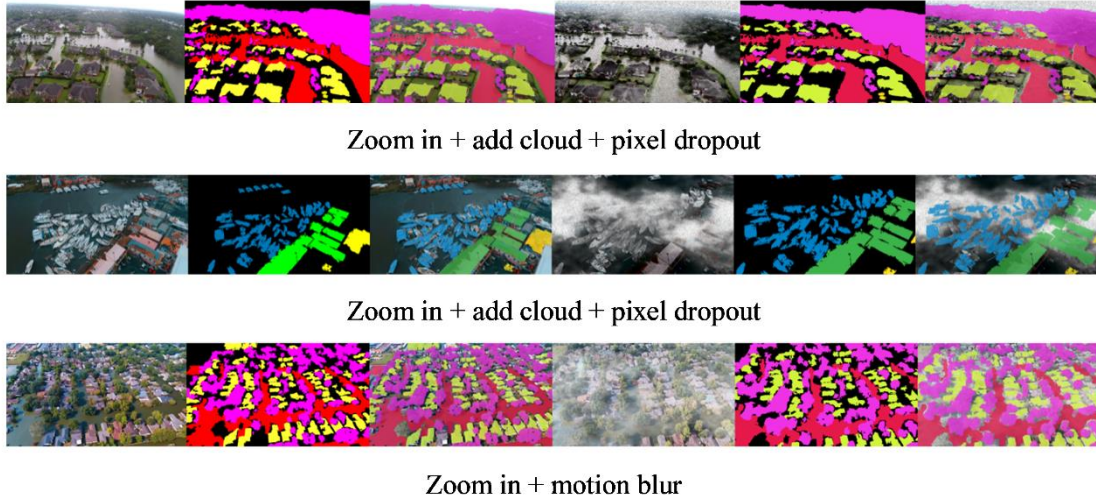


Figure 22. Examples of image augmentation operations (from left to right: original image, original annotation, annotation overlaid on original image, augmented image, augmented annotation, augmented annotation overlaid on augmented image).

Next, in order to augment the minority classes (e.g., D) without increasing the amount of majority classes (e.g., UR), images are selected by the BR threshold and dominating class. Considering images (I) with one certain class (C) as dominating class, images with BR that are higher than the BR threshold are augmented N times. Here, N indicates random selections from the bag of augmentation (32 combinations). All the possible BR thresholds and classes are used to augment the entire training data to achieve the best results. Afterwards, images and operations corresponding to the lowest total BR of are used to generate the final augmented data. In theory, augmenting with an infinite N will produce the most balance data yet impossible. Thus, considering the total operation possibilities, Volan2019 training data is augmented with maximum $N = 32, 16,$ and 8 (indicating 32, 16, and 8 maximum augmentation operations, respectively)

resulting in augmented datasets named VA32, VA16, and VA8, respectively. The maximum N is referred to as degree of augmentation. The resulting 3 models trained on VA32, 16, and 8 are named Mask1 (VA32), Mask1 (VA16), and Mask1 (VA8) as shown in Figure 23.

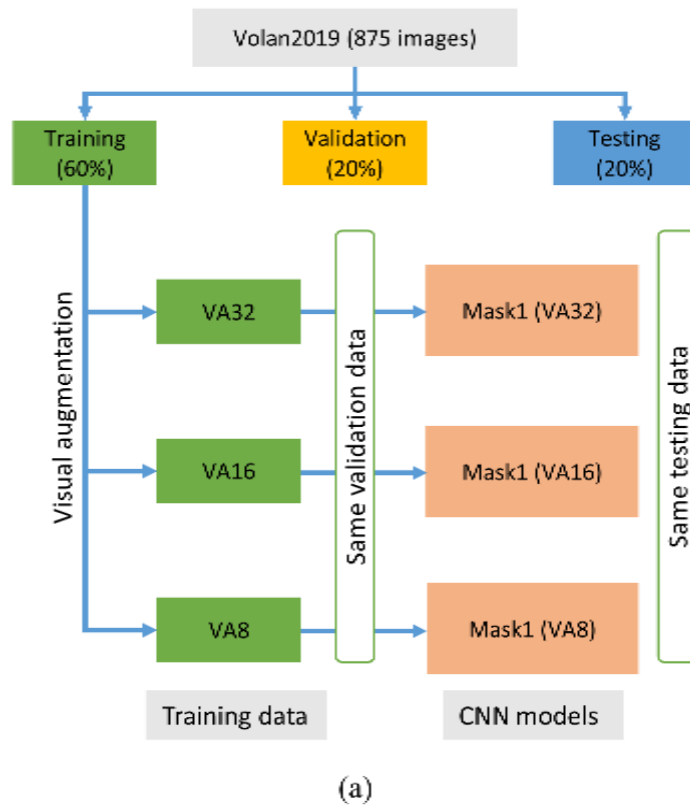


Figure 23. Volan2019 augmentation based on BR values.

Based on the conclusion in Section 4.4, view-point altitude of the recording vehicle is a main factor of the CNN models' performance. Therefore, Volan2019 is also split into two subsets MD (corresponding to mask drone) and MH (corresponding to

mask helicopter) based on the altitude as illustrated in Figure 24. Both MD and MH are split into three subsets: training (60%), validation (20%), and testing (20%). Similar to Volan2019 augmentation, each training subset of MD and MH subset is further augmented as shown Figure 24 in using the balancing strategy based on the BR. MD training subset is augmented with maximum degree 32, 16, and 8 resulting in drone augmented MDA32, MDA16, and MDA8, respectively. The training subset of MH is augmented resulting in augmented MHA32, MHA16, and MHA8 (indicating maximum 32, 16, and 8 degree augmentation applied). Validation and testing data are not augmented to evaluate the influence of BR based augmentation.

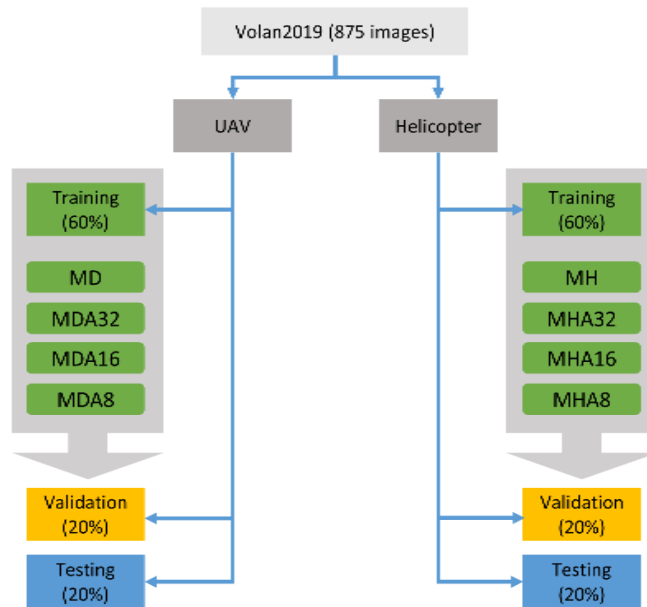


Figure 24. Volan2019 data splitting based on altitude and augmentation based on BR value.

Table 18 displays the model names with the training data (with and without augmentation), instance amount per class, and dataset BR. Model Mask2, 3, and 4 have less classes based on the confusion level between classes as later will be stated in Subsection 5.2.3.2. Notably, data augmentation for Mask2-4 is implemented only for the included classes. In order to keep the consistency of the evaluation, model Mask1-4 are tested on the same Volan2019 testing data. MaskD and its variation as well as MaskH and its variations are tested on MD testing and MH testing subsets, respectively. According to this table, augmentation is able to reduce the BR value resulting in more balanced data. Moreover, higher degree augmentation leads to more balanced data expect the training data of Mask4 has only 2 classes.

Table 18. Number of instances per class in training data and balance ratio reduction for Volan2019 dataset.

Model (training data)	UR	FA	DR	C	B	P	D	R	V	BR
Mask1 (9 classes)	4,363	1,017	276	494	1,948	282	290	432	3,025	0.87
Mask1 (VA32)	6,490	1,915	2,258	2,099	2,172	1,789	2,780	1,583	4,225	0.40
Mask1 (VA16)	5,824	1,735	1,357	1,773	2,008	1,496	1,877	1,115	3,905	0.48
Mask1 (VA8)	5,043	1,553	769	1,086	1,976	1,387	1,001	758	3,656	0.57
Mask2 (5 classes)	4,363	-	276	494	1,948	282	-	-	-	0.91
Mask2 (VA32)	4,608	-	836	978	1,965	843	-	-	-	0.62
Mask2 (VA16)	4,832	-	826	907	1,964	829	-	-	-	0.65
Mask2 (VA8)	4,770	-	676	894	1,962	838	-	-	-	0.67
Mask3 (3 classes)	4,363	-	-	-	1,948	282	-	-	-	0.66
Mask3 (VA32)	4,363	-	-	-	1,948	4,860	-	-	-	0.32
Mask3 (VA16)	4,363	-	-	-	1,948	645	-	-	-	0.59
Mask3 (VA8)	4,363	-	-	-	1,948	645	-	-	-	0.59
Mask4 (2 classes)	-	-	276	494	-	-	-	-	-	0.28
Mask4 (VA32)	-	-	2,197	494	-	-	-	-	-	0.63
Mask4 (VA16)	-	-	2,197	494	-	-	-	-	-	0.63
Mask4 (VA8)	-	-	993	526	-	-	-	-	-	0.31
MaskD (D)	2,080	689	129	339	1,856	369	151	221	2,502	0.85
MaskD (MDA32)	3,448	1,267	972	1,427	1,961	1,146	1,801	965	4,155	0.41
MaskD (MDA16)	2,931	1,131	576	1,170	1,921	1,055	1,002	630	3,505	0.47
MaskD (MDA8)	2,677	1,133	495	929	1,902	1,253	842	562	3,227	0.46
MaskH (H)	2,320	292	187	78	39	4	170	245	673	1.37
MaskH (MHA32)	2,808	950	1,063	392	611	4	939	506	1,061	0.79
MaskH (MHA16)	2,587	734	748	238	332	4	710	403	829	0.92
MaskH (MHA8)	2487	636	537	169	171	4	540	343	721	1.08

5.2. Mask object detection

5.2.1. Mask-RCNN architecture

Mask-RCNN is capable of producing instance segmentation at 5 FPS with high accuracy (32.0 % COCO-AP). Figure 25 illustrates the architecture which takes RGB input and outputs segmentation (mask) in each predicted box with an autoencoder. In detail, the input is processed by a common CNN (ResNet in this experiment) but without flattening the high dimensional output. Next, a feature pyramid network (FPN) is applied on 4 of the CNN hidden layers resulting in P2-P5 feature maps. One more map P6 is calculated by max pooling P5. Meanwhile, the input goes into a separate region proposal network (RPN) which is responsible for generating candidate prediction boxes with objectness (i.e., the chance that one box contains target objects). Normally, RPN downsizes the input by a factor of 30-40, followed by each output pixel corresponding to a large region of interest (RoI) in the input. A 3×3 kernel is applied on each FPN map to predict the objectness score and bounding box. Altogether, P2-P6 feature maps are concatenated to a list of numbers and the top 2,000 (default) proposals are selected for further consideration. This 2,000 proposals, with other parameters, are passed down separately to box and mask heads, which are responsible for box prediction and mask segmentation. For the box head, the concatenated feature maps are fully connected with the output to generate the class probability (also include background) and the offset values of the bounding boxes. For mask head, an autoencoder with several encode and decode kernels, is responsible to generate pixel masks in the predicted boxes from the featured maps. The resulting mask with each pixel containing a confidence value

represents the likelihood of objectness. The mask threshold is set at 0.5 [81], i.e., if a pixel is predicted with a confidence higher than 0.5, it belongs to that object. Finally the output includes bounding boxes with class probabilities and binary masks in each box.

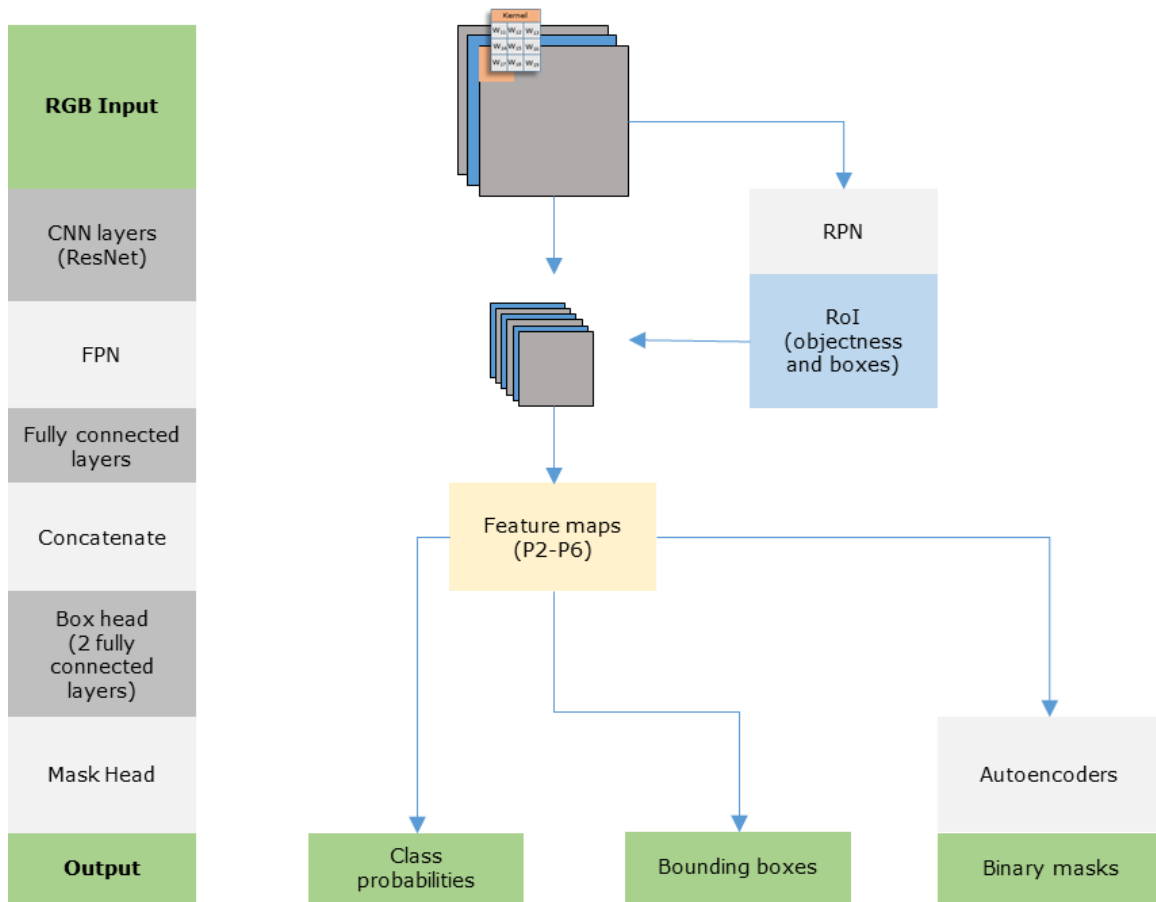


Figure 25. Mask-RCNN architecture.

5.2.2. Performance measurement

Similar to the bounding boxes measurement, intersection over union (IoU), precision, recall, and mAP are calculated at pixel level. As shown in Figure 26, the pixel wise overlapping area of the ground truth annotation and predicted area is calculated as

intersection, and union area of them are computed as union. Next, IoU for each detection is calculated with Equation 8 in Section 4.3. Based on the IoU, Figure 27 illustrate cases including true positive (TP), true negative, and two scenarios of false negative (FN). Similar to bounding boxes detection evaluation, the TP ($\text{IoU} > 50\%$), FP ($\text{IoU} = 0\%$), and FN (pixel $\text{IoU} < 50\%$, or $=0\%$) cases are saved to calculate the precision and recall value using Equation 9 and 10 in Section 4.3. Finally, the mAP which is the area under the precision-recall curve is calculated to describe the overall performance.



Figure 26. Pixel intersection and union example.

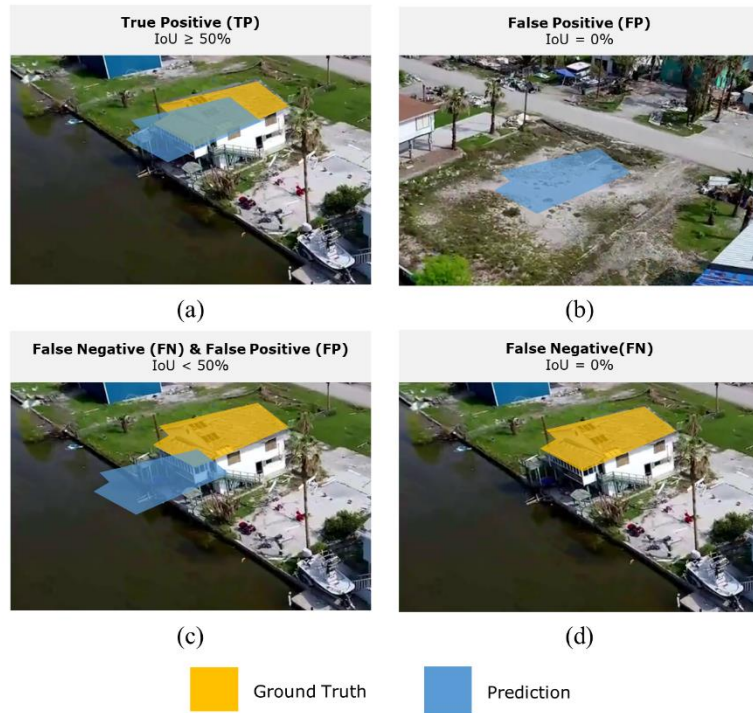


Figure 27. Pixel level examples of TP, FP, and FN.

5.2.3. Experiments and results analysis

5.2.3.1. Prediction confidence and precision correlation

Model Mask1 trained on the entire Volan2019 training subset is based on Mask-RCNN architecture with pre-trained weight that is trained on the COCO dataset. While training, the learning rate is set as 10^{-3} and decreased by 10^{-1} if there is no loss drop monitored within 3 epochs. The training of Mask1 is terminated if the validation loss does not drop after 10 epochs. After the model is fully trained, Mask1 is tested on the testing subset (154 images) as examples shown in Figure 28. According to this table, Mask1 has the mAP of 25% with class P has the highest AP of 55%, followed by class B at 46% and UR at 43%.

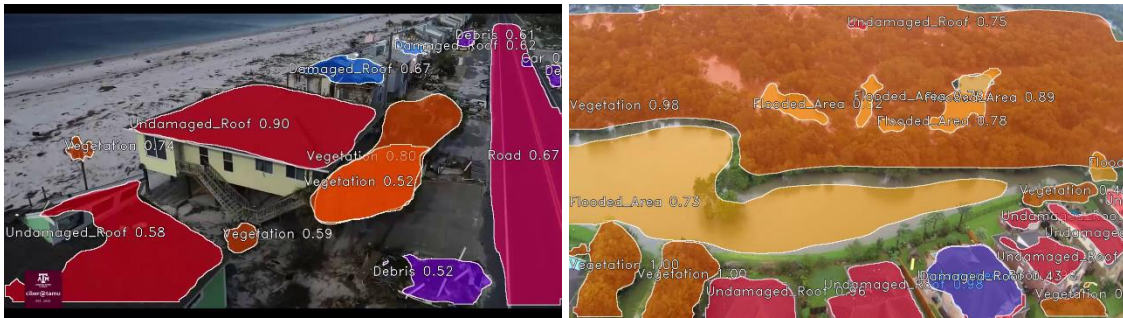


Figure 28. Sample detection output of Mask1 tested on Volan2019 dataset.

However, there is no ground truth to compare when deploying the models in practice (e.g., newly happening hurricane). Therefore, metrics including AP and mAP are not available due to the lack of the ground truth. Given the model produces confidence with prediction as shown in detection examples in Figure 28, it is possible to predict the accuracy only based on the confidence value to solve this problem. To this end, an analysis of the correlation between prediction precision and confidence is carried out. Considering Mask1, prediction for each image per class is compared individually with the ground truth resulting in paired precision and confidence (P&C) values. The Pearson R value of this P&C data is measured and reported in Table 20. Moreover, this P&C data is further split into three parts: regression (60%), validation (20%), and testing (20%). Regression and validation data are used to perform regression with degree 1 (linear), 2, 3, and 4 in order to predict the accuracy based on a given confidence value. Next, the predicted accuracy is compared with testing data and the errors for each class as reported in Table 20. As an example of UR shown in Figure 29 (a), the P&C values are plotted and used to conduct linear regression resulting in a function of confidence F .

The function F is later used to predict the testing portion of the P&C data to determine the detection precision solely based on confidence value. After comparing the predicted precision and ground truth precision, the error is calculated as 1.19%. Figure 29 (b) shows the P&C regression at degree 2 with 1.17% error. Observably, P&C regression with higher degrees does not necessarily produce more accurate precision prediction.

Table 19. Mask1 average error (%) for different degrees of precision-confidence regression.

Degree	UR	FA	DR	C	B	P	D	R	V
1 (linear)	1.19	1.41	1.43	1.44	0.45	3.49	0.89	3.87	0.99
2	1.17	1.47	1.49	1.44	0.38	3.6	0.88	3.83	1.1
3	1.17	1.49	1.50	1.44	0.39	3.32	0.88	3.77	1.14
4	1.19	1.65	1.48	1.43	0.31	3.77	0.87	3.65	0.99

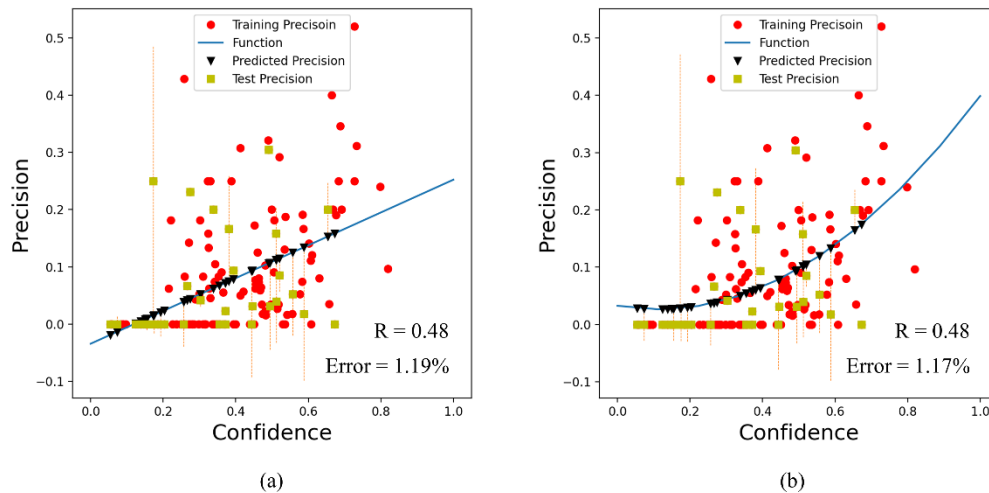


Figure 29. Precision-confidence correlation in Mask1 for class undamaged roof (UR) using (a) linear, and (b) polynomial regression.

The same linear regression and testing analysis is performed for Mask2, 3, and 4, which have less classes compared to Mask1. Results listed in Table 20 show that there is

always a positive correlation between confidence and precision, i.e., high confidence predictions are associated with high precision. This correlation is valuable in determining prediction quality when testing CNN models on unseen data (e.g., newly happening disasters). Furthermore, the error between the predicted (from regression) and ground truth precision falls in the range of 0.45-4.28% indicating this method is capable of producing an approximate precision without the ground truth data.

Table 20. Precision-confidence correlation analysis for different models using linear regression.

		UR	FA	DR	C	B	P	D	R	V
Mask1	Pearson R	0.48	0.46	0.40	0	0.75	0.5	0.03	0.53	0.39
	Error (%)	1.19	1.41	1.43	1.44	0.45	3.49	0.89	3.87	0.99
Mask2	Pearson R	0.48		0.14	0.03	0.74	0.42			
	Error (%)	1.91		0.14	0.1	0.48	1.45			
Mask3	Pearson R	0.44				0.88	0.41			
	Error (%)	2.36				0.29	0.93			
Mask4	Pearson R				0.26			0.54		
	Error (%)				1.27			4.28		

5.2.3.2. Error percentage matrix and class separation

Confusion matrix is a common method of presenting model performance in binary and multi-class classification tasks. In this matrix, rows and columns correspond to predicted and actual classes [182], and misclassified cases are compared and benchmarked against correctly classified cases. In semantic segmentation, Li and Nevatia [183] used pixel-level confusion matrix (a.k.a., detection error matrix in this Dissertation) for image region recognition. Past work in object detection has also adopted the notion of error matrix. For instance, Peren-Hernandez et al. [184] created

such matrix from detection results using one-versus-one strategy. In a document analysis study, an error matrix was created through comparing each ground truth (e.g., table, image, and text) with the prediction [185]. Figure 30 shows the detection error matrix generated by comparing each prediction from Mask1 model tested on the Volan2019 testing subset (175 images) with all ground truth annotations contained in the same image. A sample frame is shown in Figure 31, which shows that only when the pixel IoU is greater than 50%, the detection is considered an overlapping detection. Subsequently, in the detection error matrix of Figure 30, the value of the cell corresponding to this detection-ground truth pair is increased by 1. Following this approach, it should be noted that unlike the traditional confusion matrix, the values in the detection error matrix, do not necessarily add up to the total number of ground truth cases; rather, they represent the number of all overlapping (IoU>50%) cases.

		Ground truth								
		UR	FA	DR	C	B	P	D	R	V
Prediction	UR	80	11	11	0	0	1	0	1	1
	FA	22	64	2	0	0	0	0	4	2
	DR	30	1	20	0	1	4	0	0	0
	C	10	0	2	3	2	5	0	0	0
	B	5	1	5	0	19	1	0	1	0
	P	2	0	0	0	0	13	0	0	0
	D	12	1	10	0	1	2	7	0	2
	R	3	14	0	0	1	0	0	8	1
	V	4	0	0	1	0	3	2	0	99

Figure 30. Mask1 detection error matrix.

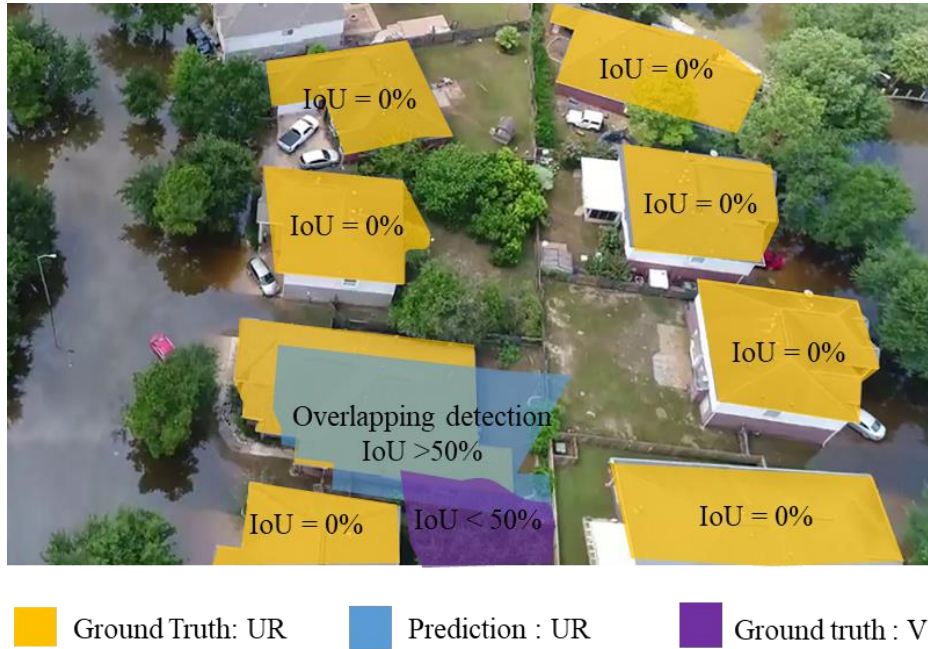


Figure 31. Example of determining overlapping detection based on pixel-level IoU.

Given that the prediction confidence values reflect prediction quality, a 9×9 percentage error matrix is created, as shown in Figure 32 using a similar approach but considering detection confidence instead of pixel-level IoU. This time, the confidence value $confidence_k$ of an overlapping detection is added to the value of the cell corresponding to the detection-ground truth pair. For example, when a ground truth UR overlaps ($>50\%$) with a detected UR, the prediction confidence will be added to the cell $\{UR, UR\}$. After taking into account all predictions, the cumulative error matrix score (EMS) of each cell is computed using Equation 12. Here, n is the total number of the overlapping cases for one cell, which is noted as set $S = \{confidence_1, confidence_2, \dots, confidence_n\}$. For each cell, error percentage (EP) is

further calculated by Equation 13, in which, c is a set of all classes $\{UR, FA... V\}$, and EMS' denotes the value of the cell corresponding to a particular ground truth class.

$$\text{Error Matrix Score (EMS)} = \sum_{k=1}^n \text{confidence}_k \quad (12)$$

$$\text{Error Percentage (EP)} = EMS \div \sum_{EMS' \in c} EMS' \times 100\% \quad (13)$$

As shown in Figure 32, the error percentage matrix reflects the misclassification between classes. For instance, detections for ground truth DR is often predicted as UR with high confidence (33.96%). Class P is mostly wrongly predicted as DR (6.75%) and C (8.27%). Class R is often misclassified as FA (23.79%). Furthermore, low performance classes including FA, D, R, and V are bulk classes, i.e., they are spread out and continuous as shown in Figure 28. The approach, which Mask-RCNN models produce bounding boxes followed by segment pixels inside them, is not suitable for these bulky classes. In other words, a bounding box across the whole image is required to produce segmentation for large FA, R, D, and V instances. Therefore, an autoencoder is built specifically for these bulk classes as will be explained in the following section. On the other hand, individual classes including UR, DR, C, B, and P are separated based on the error matrix to avoid confusion resulting in the following models:

- Mask2 trained on data only including classes UR, DR, C, B, and P (all individual classes).
- Mask3 trained on data solely covering classes UR, B, and P.
- Mask4 trained on data that contains classes DR and C only.

		Ground truth								
		UR	FA	DR	C	B	P	D	R	V
Prediction	UR	71.42	7.75	33.96	0.00	0.00	0.85	0.00	3.33	0.35
	FA	7.43	83.15	1.46	0.00	0.00	0.00	0.00	23.79	2.04
	DR	11.30	0.30	39.09	0.00	3.22	6.75	0.00	0.00	0.00
	C	2.47	0.00	1.61	60.68	2.44	8.27	0.00	0.00	0.00
	B	0.88	0.13	8.59	0.00	88.67	0.85	0.00	5.82	0.00
	P	0.66	0.00	0.00	0.00	0.00	76.60	0.00	0.00	0.00
	D	2.39	1.17	15.29	0.00	2.57	1.89	90.49	0.00	0.31
	R	0.40	7.50	0.00	0.00	3.10	0.00	0.00	67.06	0.94
	V	3.05	0.00	0.00	39.32	0.00	4.79	9.51	0.00	96.36

Figure 32. Mask1 error percentage matrix to identify highly mismatched classes.

5.2.3.3. Results and analysis

As stated in Subsection 5.1.2, the training data for Mask2-4 is augmented with the proposed BR augmentation method resulting in three augmented (denoted with VA 32, VA16, and VA8 corresponding to maximum degree at 32, 16, and 8, respectively) variation dataset for each. Table 21 displays the performance of Mask1-4 and their variations tested on the same Volan2019 testing subset (only existing classes are considered). Mask3 produces the highest AP for class UR (46.25%) and P (59.16%). Mask3 (VA8) has the best performance for class B with an AP of 64.01%. Mask4 achieves the highest AP of 19.26% for C. Mask4 (VA32) is able to detect DR with an AP of 39.68% which is the largest improvement (25.72%) compared to Mask1 among all classes. It is observable that data augmentation and class separation based on error

matrix improves the performance of individual classes but not bulk classes. Table 21 also contains the testing results of MaskD and its variations are tested on the testing subset of MD, as well as MaskH and its variations are tested on the testing subset of MH. Compared to the benchmark model Mask1, data augmentation and separation based on altitude improves the performance of the bulk classes. For example, Mask H achieves 25.28% AP for class FA, while MaskD (MDA8) has the highest AP for classes D (29.51%), R (10.01%), and V (26.07%). Altogether, the highest mAP is achieved by Mask3 (VA8) proving data augmentation and class separation (based on error matrix and altitude) is an efficient way to boost the model's performance.

To compare the performance of this research work with the state-of-the-art models such as Mask-RCNN and Faster-RCNN, shared classes between Volan2019 and large datasets (e.g., COCO and VOC) such as car, people, boat, and plants are selected to compare. Although all the images from large datasets are taken from the ground view-point, as oppose to Volan2019 images which are recorded from aerial view, those classes should share similarities. Table 22 lists the testing results of Mask-RCNN tested on COCO and Faster-RCNN tested on VOC regarding the shared classes. According to this table, the proposed framework is capable of detecting classes people, boat, and vegetation (plant) with comparable accuracy. However, the performance of class car in this research is lower than its counterpart, due to the relatively small car sizes in Volan2019 compared to COCO and VOC.

Table 21. Summary of AP (%) and mAP (%) of Mask-RCNN models and their variations.

Model	UR	FA	DR	C	B	P	D	R	V	mAP
Mask1	42.85	20.51	13.96	15.08	46.03	55.00	3.35	7.11	20.90	24.98
Mask1 (VA32)	37.72	13.70	39.35	8.25	53.93	32.57	8.03	2.55	21.51	24.18
Mask1 (VA16)	37.22	13.96	32.89	7.34	51.25	38.02	7.85	2.68	21.15	23.59
Mask1 (VA8)	39.90	16.26	29.57	8.64	56.17	43.16	5.64	2.18	24.86	25.15
Mask2 (5 classes)	42.23		25.26	15.89	58.16	38.33				35.97
Mask2 (VA32)	42.03		35.50	10.36	60.56	42.31				38.15
Mask2 (VA16)	42.70		37.49	11.27	60.53	44.13				39.22
Mask2 (VA8)	41.48		38.40	10.54	60.05	48.24				39.74
Mask3 (3 classes)	46.25				48.86	59.16				51.42
Mask3 (VA32)	41.22				58.95	37.51				45.89
Mask3 (VA16)	40.91				62.94	48.23				50.69
Mask3 (VA8)	41.76				64.01	48.85				51.54
Mask4 (2 classes)			24.60	19.26						21.93
Mask4 (VA32)			39.68	10.57						25.12
Mask4 (VA16)			39.34	11.74						25.54
Mask4 (VA8)			39.68	10.57						25.12
MaskD	37.45	13.99	7.66	7.95	50.63	44.42	1.85	4.93	22.21	21.23
MaskD (MDA32)	36.88	12.66	21.90	12.88	44.84	41.33	3.27	4.24	21.64	22.18
MaskD (MDA16)	37.47	15.16	19.40	8.40	45.32	45.07	2.57	4.68	22.05	22.24
MaskD (MDA8)	37.82	19.73	25.18	15.42	45.51	41.30	29.15	10.01	26.07	27.80
MaskH	44.66	25.28	22.72	7.07	2.77	0.00	1.97	3.98	5.91	12.71
MaskH (MHA32)	45.22	24.32	20.18	6.33	17.08	0.00	4.54	4.14	6.10	14.21
MaskH (MHA16)	42.66	21.39	22.67	7.07	11.53	0.00	5.22	2.58	7.26	13.38
MaskH (MHA8)	44.93	18.33	18.75	4.01	6.41	0.00	8.18	2.79	6.22	12.18

Table 22. Performance comparison of CNN models for similar classes to Volan2019 dataset.

Model	Train & test data	Car	Boat	Person	Plant (vegetation)	mAP
This work	Volan2019	19.26	64.01	59.16	26.07	51.54
		Mask4	Mask3 (VA8)	Mask3	MaskD (MDA8)	
Mask R-CNN	COCO	49.10	N/A	34.80	N/A	58.10
Faster R-CNN	VOC	78.20	53.20	69	30.10	66.90

In order to investigate the influence of the object size, another analysis of Mask1 testing results is conducted based on the prediction pixel sizes. For each class, all ground truth pixel areas are measured to form a set T . The top 25% and bottom 25% of T are selected as *upper* and *lower* threshold. Next, all the predictions and ground truth masks are considered as portion large (L) if the mask area is greater than *upper*; portion medium (M) if mask size is between *upper* and *lower*, and small (S) portions if the mask pixel areas smaller than *lower*. For each portion (bin), precision, recall, and AP are calculated and recorded. Beside the results of bin L, M, and S, Table 23 also lists the results of large and medium portion combined (L+M) as well as the total performance bin T . Compared to the benchmark (bin T), the large portion (bin L) has the highest AP for class UR (40.59%), FA (47.08%), DR (28.27%), B (44.46%), D (4.36%), R (14.21%), and V (28.40%). The medium portion (bin M) produces the highest performance for class C (14.74%) and P (66.34%). The reason behind the low performance of class C is that the images contain cars are taken at a relatively high altitude, resulting in smaller pixel sizes even in bin L compared to other classes. On the other hand, the small portion (bin M) yields to a much lower accuracy, indicating that the large pixel area is associated with higher performance. Lastly, the combined portion (bin $L+M$) shows a higher performance than the benchmark (bin T) implies filtering the prediction to select large objects can help improve the model performance.

Table 23. Effect of mask size on Mask1 model performance.

Bin		UR	FA	DR	C	B	P	D	R	V
Large (L)	# instances (ground truth)	369	78	29	42	171	38	38	43	281
	# instances (prediction)	734	305	95	272	252	87	303	108	579
	AP (%)	40.59	47.08	28.27	12.41	44.46	34.87	4.36	14.21	28.40
	Precision (%)	31.88	18.36	14.74	6.62	38.10	25.29	1.65	13.89	23.49
	Recall (%)	63.41	71.79	48.28	42.86	56.14	57.89	13.16	34.88	48.40
Upper (pixel area)		3,709	25,263	16,631	807	2,804	2,247	8,338	21,699	7,977
Medium (M)	# instances (ground truth)	737	156	57	81	341	74	74	86	560
	# instances (prediction)	1517	965	390	298	692	208	791	443	1,681
	AP (%)	39.70	18.01	9.94	14.74	42.87	66.34	3.14	7.18	19.97
	Precision (%)	32.23	6.74	6.15	11.07	32.66	30.29	2.15	4.74	13.44
	Recall (%)	66.35	41.67	42.11	40.74	66.28	85.14	22.97	24.42	40.36
Lower (pixel area)		929	812	3,774	384	719	1,081	1,377	2,953	786
Small (S)	# instances (ground truth)	370	77	28	41	170	37	37	42	281
	# instances (prediction)	1,341	640	333	64	307	213	434	424	1,065
	AP (%)	13.88	1.21	4.31	1.44	8.21	11.84	0.15	0.33	3.91
	Precision (%)	11.48	2.19	2.70	3.13	14.33	4.69	0.46	0.47	4.79
	Recall (%)	41.62	18.18	32.14	4.88	25.88	27.03	5.41	4.76	18.15
L+M	AP (%)	46.73	26.85	15.46	18.06	50.78	60.84	3.29	8.56	23.39
	Precision (%)	34.72	9.69	8.87	9.46	37.67	30.85	2.10	6.90	16.67
	Recall (%)	70.58	52.34	49.43	43.55	69.26	79.82	20.35	29.2	44.72
All (T)	AP (%)	42.85	20.51	13.96	15.08	46.03	55.00	3.35	7.11	20.90
	Precision (%)	27.51	7.23	6.73	9.78	34.45	20.83	1.77	4.62	13.71
	Recall (%)	66.94	44.37	48.25	37.80	63.20	71.14	18.12	26.32	40.64

5.3. Autoencoder segmentation

5.3.1. PSPNet architecture

PSPNet stands for pyramid scene network which takes images as inputs and outputs semantic segmentations by using an encoder-decoder structure, known as

autoencoders. Figure 33 illustrates the PSPNet architecture details: the input image matrices first goes through a series of kernels and max pooling layers which produces denser matrices with smaller dimensions. Generally, this encoder process is implemented by using established CNN architectures such as ResNet or VGG-16. Unlike common bottleneck layers between encoder and decoder structures, PSPNet has a pyramid pooling module which consists of multiple 1×1 kernels and average pooling layers at different scales such as 2×2 or 8×8 which result in several feature maps. The output of the all the feather maps is concatenated along one dimension. The following up-sampling structure uses a series of reversed CNNs to output a single channel mask with the same dimension as the input. In the output, each pixel will be classified in terms of belong or not belong to one class by SoftMax (i.e. binary prediction) functions. Altogether, this mask contains numerical values indicating different classes (0, 1, 2, ...) including the background (0). Compare to FCN, SegNet, and other segmentation architectures, the pyramid pooling module in PSPNet reduces the information loss thus outperforms others in complex tasks.

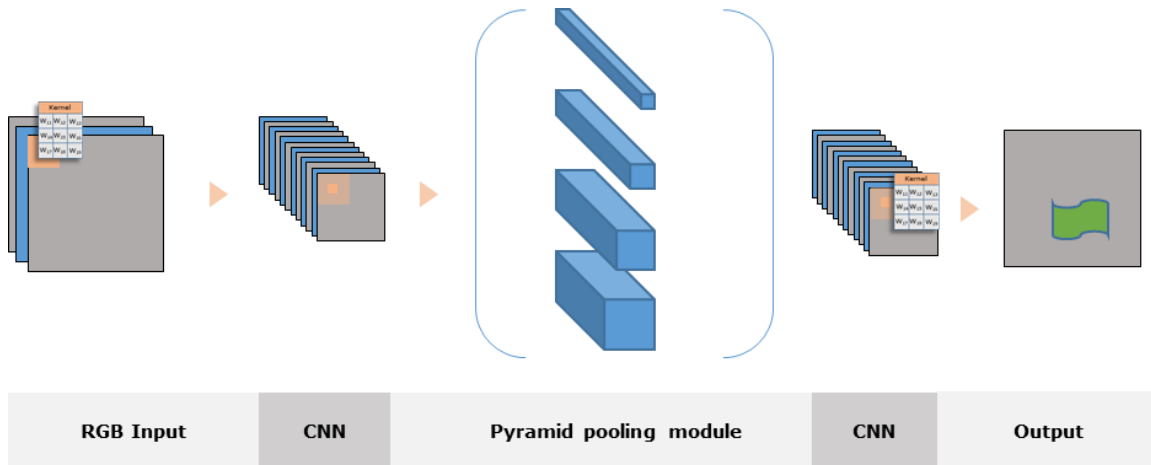


Figure 33. PSPNet architecture.

5.3.2. Performance measurement

In autoencoder segmentation (e.g., PSPNet), slightly different from object detection (e.g., YOLO or Mask-RCNN), the pixel level intersection over union (pixel IoU) and accuracy is often reported [186]. Given each pixel is predicted as belonging or not belonging to one particular class, each pixels is categorized into cases of true positive (TP, i.e., correct predicted pixel), true negative (TN, i.e., background), false positive (FP, i.e., wrong predicted pixel), and false negative (FN, i.e., missed ground truth pixel). Figure 34 shows an example (5×5 square pixel image) of the measurement of comparing each predicted pixel (25 in total) with the ground truth pixel resulting in TP, TN, FP, and FN cases. For every class, pixel IoU is calculated with Equation 14 by dividing TP case pixels over the sum of TP, FP, and FN cases. The mean pixel IoU (mIoU) across all classes is used to describe the overall performance of the model. Accuracy is computed by TP and TN cases divided by all the pixels in the image as in Equation 15. To give a perspective, the mIoU for FCN, SegNet, and PSPNet tested on

ImageNet scene parsing 2016 dataset are 29.39%, 21.64%, and 41.68%, respectively [82].

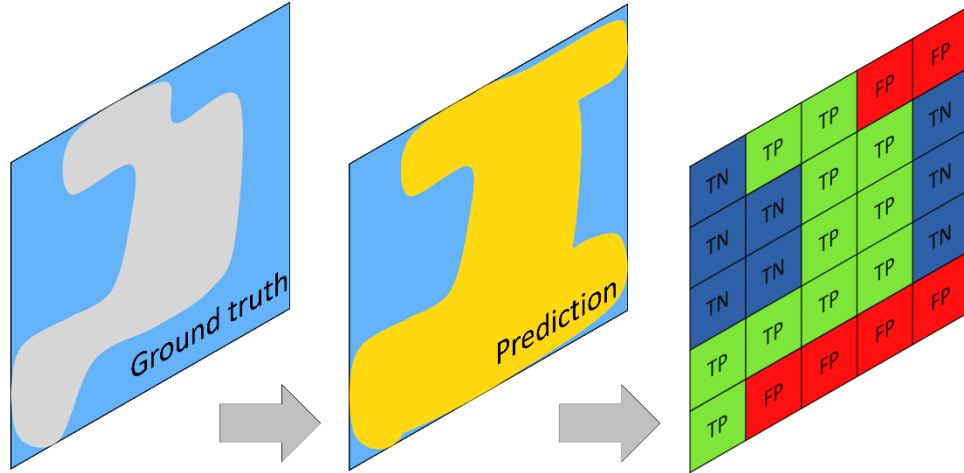


Figure 34. TP, FP, and TN cases in semantic segmentation.

$$IoU = \frac{TP}{TP + FP + FN} \quad (14)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (15)$$

5.3.3. Experiments and results analysis

As stated previously, models based on PSPNet architecture are trained in order to detect bulky classes including D, V, FA, and R. Therefore, only the annotations for those classes are extracted from Volan2019 excluding class UR, DR, C, P, and B. This processed dataset is named as PSPNet2019 (P2019), followed with the same separation scheme, dividing P2019 into training (60%), validation (20%), and testing (20%) portions. Model PSPNet1, using ResNet50 as the backbone architecture, is trained and validated on P2019 with pre-trained weights on VOC. As shown in Table 24, this model

achieves pixel mIoU of 31.26% and 73.92% accuracy. Figure 35 displays the detection examples which qualitatively better represent the bulky objects.

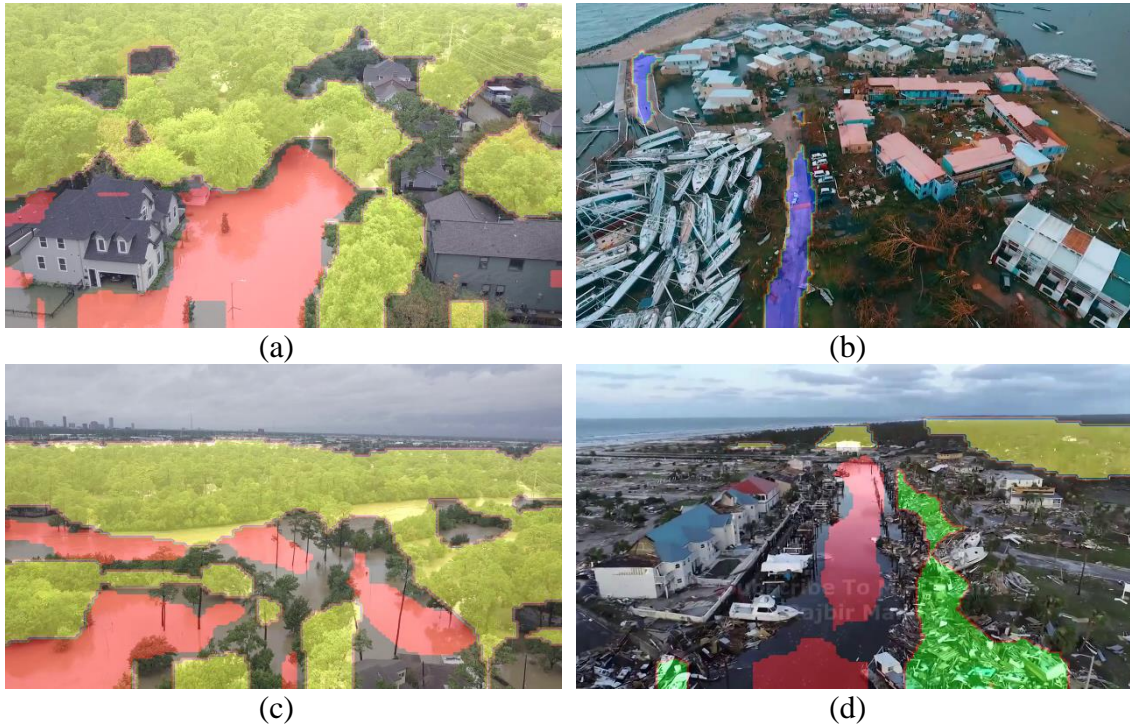


Figure 35. Sample semantic segmentation output of PSPNet1 tested on P2019 dataset.

Similar BR-based data augmentation is implemented on P2019 training subset resulting in augmented datasets P2019 augmented at degree 32 (PA32), P2019 augmented at degree 16 (PA16), and P22019 augmented at degree 8 (PA8). With the same validation data, pre-trained weights, and backbone architecture, models PSPNet1(PA32), PSPNet1(PA16), and PSPNet1(PA8) are trained on PA32, PA16, and PA8, respectively. In order to investigate the influence of pre-trained weights and network depth, two models PSPNet2 and 3 are trained. Compared with PSPNet1, the

difference is that PSPNet2 uses pre-trained weights on Cityscapes [187] as oppose to PSPNet1 is pre-trained on ImageNet. PSPNet3 uses ResNet 101 as backbone which is denser (more filter weights in layers) compared to PSPNet1 used ResNet50. Similar to Volan2019, P2019 is also split into two groups: drone images (PSPD) and helicopter (PSPH) followed by training models PSPNetD and PSPNetH on the corresponding training subsets (60%). Each training subset is augmented based on the explained augmentation using degree of 32, 16, and 8 resulting in dataset PDA32, PDA16, PDA8, (for drone) PHA32, PHA16, and PHA8 (for helicopter) as in Figure 36. Follow the same name convention for Mask-RCNN, the trained models are named PSPNetD (PDA32), PSPNetD (PDA16), PSPNetD (PDA8), PSPNetH (PHA32), PSPNetH (PHA16), and PSPNetH (PHA8).

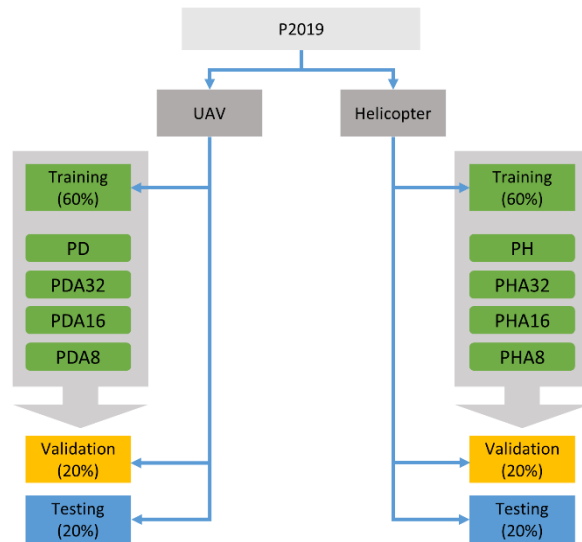


Figure 36. P2019 data augmentation based on balance ratio, and data separation based on altitude.

Table 24 shows the training and testing data specifications as well as the performance readings including IoU for each class, mIoU, and pixel accuracy for all the PSPNet models. Comparing PSPNet1 and its augmented variations, augmenting minority classes generally increases the IoU of those classes, e.g., IoU of class D increased from 2.24% to 8.26%. Similarly, class R has the pixel IoU improvement from 8.32% to 9.53%. On the other hand, majority classes typically have decreases after augmentation. For instance, the IoU of class V decreased from 38.94% to 5.02%. The comparison made between PSPNet1 and PSPNet3 is to identify the effect of network depth. Although ResNet101 is deeper than ResNet50, the IoU of small size classes (class D) does not improve compared to the benchmark model (PSPNet1). For the effect of pre-trained weights (comparing PSPNet2 and PSPNet1), pre-training the model on Cityscapes does not improve the IoU of the shared classes such as flooded area and road significantly, as well as in other three classes. Also, the mIoU remains almost the same (31.26% compared to 28.98%). Training and testing models on subsets PSPD and PSPH based on elevation improves the IoU of class R, however leads to lower IoU for FA and D (in both subsets). Considering splitting data and augmentation, the best performance for FA is 38.58% (PSPNetH), class R at 25.86% (model PSPNetD), and class V has the IoU of 38.94%. The overall best performing model is PSPNetD (PDA8) with mIoU of 32.17%. This proves the proposed data augmentation is capable of improving the performance of most of the classes.

Table 24. Summary of AP (%), mIoU (%), and accuracy (%) of PSPNet models and their variations.

Model	Training	BG	FA	D	R	V	mIoU	Accuracy
PSPNet1 (ResNet50)	PSP2019	70.81	35.97	2.24	8.32	38.94	31.26	73.92
PSPNet1 (PA32)	PA32	69.58	18.92	8.26	9.53	5.02	22.26	72.37
PSPNet1 (PA16)	PA16	70.05	29.85	7.28	4.08	3.49	23.23	73.7
PSPNet1 (PA8)	PA8	59.85	25.25	4.68	6.27	10.06	23.23	72.37
PSPNet2 (Cityscapes)	PSP2019	70.95	30.30	0.73	7.87	35.04	28.98	73.2
PSPNet3 (ResNet101)	PSP2019	71.22	14.54	1.58	13.91	37.37	27.73	73.26
PSPNetD	PSPD	67.24	29.66	0.00	16.5	37.92	31.25	70.19
PSPNetD (PDA32)	PDA32	66.38	22.36	3.57	10.63	22.4	25.06	66.64
PSPNetD (PDA16)	PDA16	67.69	27.9	0.79	24.9	30.71	30.4	67.81
PSPNetD (PDA8)	PDA8	67.95	30.59	1.48	25.86	34.96	32.17	70.75
PSPNetH	PSPH	69.35	21.7	1.53	18.67	18.61	25.97	77.01
PSPNetH (PHA32)	PHA32	76.83	33.65	9.08	17.68	12.94	28.4	74.18
PSPNetH (PHA16)	PHA16	76.57	31.03	0.46	12.95	15.73	27.34	75.63
PSPNetH (PHA8)	PHA8	77.08	38.58	0.21	9.92	17.94	28.75	76.82

In order to compare this research work with the state-of-the-art autoencoders, Table 25 shows results of Unet [188], SegNet [128], and PSPNet tested on ADE20K [189]. Similar to COCO and VOC, ADE20k contains images taken from ground view, and have classes that are similar to flood, road, and vegetation (grass). The comparison shows that the series of PSPNet models in this research are comparable with the benchmarks in terms of mIoU and accuracy.

Table 25. Performance (%) of U-Net, SegNet, and PSPNet for similar classes to P2019.

Model	Train & test data	Water (flooded area)	Road	Grass (vegetation)	mIoU	Accuracy
This work	PSP2019	38.58	25.86	38.94	32.17	77.01
		PSPNetH (PHA8)	PSPNetD (PDA8)	PSPNet1	PSPNetD (PDA8)	PSPNetH
Unet	ADE20K	20.40	41.40	52.70	28.50	71.32
SegNet		58.40	63.00	62.80	21.64	71.00
PSPNet		n/a	n/a	n/a	21.64	76.35

5.4. Summary

In order to produce pixel level segmentation of aerial visual input, Volan2019 was created containing 875 pixel-level annotated images with classes including building roof (damaged and undamaged), road, car, boat, people, vegetation, debris, and road. CNN architectures Mask-RCNN was employed (using transfer-learning) to train, validate, and test models on Volan2019. Particularly, correlation between prediction confidence and precision was analyzed, and results showed a positive relation between confidence and precision. This provides practical value for disaster response practitioners with a mathematical tool to evaluate the detection quality especially during newly happening disaster when the ground truth is not available. Based on the positive correlation between confidence and precision, an error matrix reflecting the detection confusion was created. This matrix was further used to guide class separation lead to a series of Mask-RCNN models each trained and tested on subsets with focuses of different classes (less confused). Additionally, a novel data augmentation strategy based on data balance level was created and tested. Altogether, with class separation and data

augmentation, the best performing Mask-RCNN model achieved a mAP of 51.54%. Furthermore, bulk classes including flood area, vegetation, debris, and road were selected to build datasets (training, validation, testing, and augmentation) for PSPNet models. The testing results showed autoencoders not only able produced better quality output for bulk objects but also achieved considerable pixel mIoU and accuracy. In particular, the best performing PSPNet model achieved a 32.17% mIoU and 77.01% accuracy. Altogether, these results implied the proposed framework was capable of extracting pixel level segmentation from aerial imagery for rapid PDA and quantification in the aftermath of disasters.

6. PERSPECTIVE TO ORTHOGONAL MAPPING

This Chapter reviews the homography transformation method used in this study to convert the pixel coordinates of detected objects in perspective aerial views to orthogonal coordinates used in mapping applications. It also presents object tracking and counting techniques that do not rely on drone's positional data, and tests these techniques on disaster footage collected by a drone.

6.1. Homography transformation

The outputs of object detection and semantic segmentation (described in Chapters 4 and 5) are bounding boxes and masks (e.g., roofs, debris, car). However, these detected objects with bounding boxes or masks cannot be directly used in mapping systems such as United States national grid (USNG), which allows rapid assessment by describing the damage, performing analysis, and illustrating impact. Therefore, this research designs a framework to project the quantitative assessment onto an orthogonal GIS map for the end users. In order to transform the perspective view from drone's or helicopter's local frame onto an orthogonal GIS maps, information describing the viewpoint's geolocation (in this case, global coordinates of the drone camera) is often needed at all times. However, access to drone's global position may not be always possible (e.g., in GPS-denied environments) especially during disastrous environment. Moreover, it is likely that position meta-data is not collected when crowdsourcing the data collection, or some users do not will share location information due to privacy issues or lack of knowledge about geolocation meta-data. Therefore, this Chapter present

an approach that projects perspective information onto a GIS map solely relying on visual input.

Homography transformation is an established projection method in computer vision which is commonly used for camera calibration [190], image rectification [191], and aerial mapping [192]. In homography transformation, any point on a given input plane can be projected on an output plane in space by a 3×3 homography matrix H , as written in Equation 15. Matrix H is determined up to scale, thus i in Equation 15 is normalized to 1 resulting in H having 8 degrees of freedom (DOFs), variables $a, b, c, d, e, f, g,$ and h . Input and output matrices are defined by Equation 16, in which (X_{input}, Y_{input}) and (X_{output}, Y_{output}) denote coordinates in the input and output systems, and w indicates the scale factor in the output coordinates system. The final output coordinate $(X'_{output}, Y'_{output})$ is computed by Equation 17.

$$output = H \times input, H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (15)$$

$$input = \begin{bmatrix} X_{input} \\ Y_{input} \\ 1 \end{bmatrix}, output = \begin{bmatrix} X_{output} \\ Y_{output} \\ w \end{bmatrix} \quad (16)$$

$$X'_{output} = \frac{X_{output}}{w}, Y'_{output} = \frac{Y_{output}}{w} \quad (17)$$

In order to calculate the 8 variables for matrix H , four pairs of reference points with known input and output in both coordinate systems are required. As shown in Figure 37, any given point (include the reference points) in perspective view from drones or helicopters can be projected onto a real-world orthogonal map by H . For pixel coordinates, with the most upper left pixel of the image serving as the origin, rows

parallel to the X-axis, and columns parallel to the Y-axis, the pixel coordinates of the centroid of the detected bounding box is determined. The real-world position of the object, on the other hand, refers to its location in the Cartesian grid system (global coordinates on Earth). Reference points marked as (x_1, y_1) and (x'_1, y'_1) ; (x_2, y_2) and (x'_2, y'_2) ; (x_3, y_3) and (x'_3, y'_3) , and (x_4, y_4) and (x'_4, y'_4) are shown in Figure 37. For each paired reference points, orthogonal coordinates (X_{output}, Y_{output}) are computed based on perspective coordinates (X_{input}, Y_{input}) using Equation 18, resulting in Equations 19 and 20. Repeating this procedure for the other three reference points results in a total of 8 equations that can be used to solve variables $a, b, c, d, e, f, g,$ and h , for matrix H . Using H , any given point (x_5, y_5) in the perspective view can be projected on the orthogonal map (x'_5, y'_5) using Equation 17.

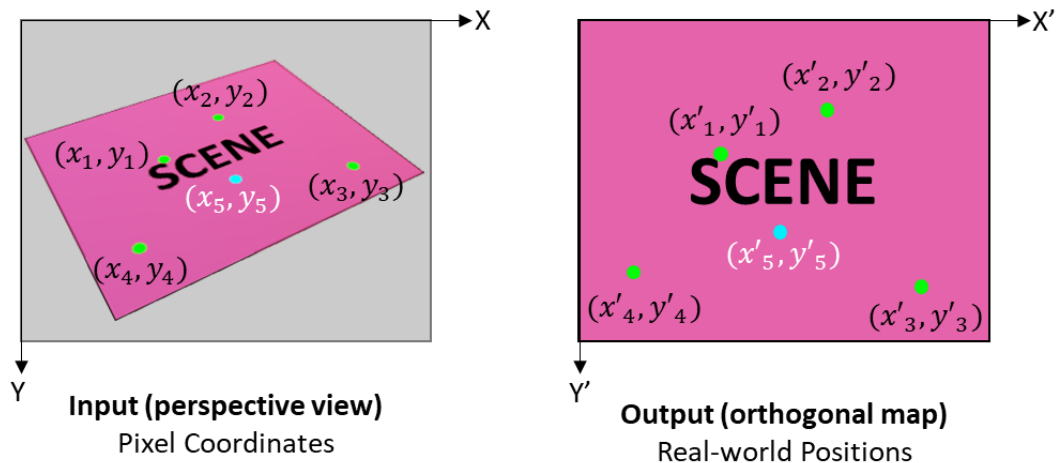


Figure 37. Perspective to orthogonal transformation.

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix}_{\text{output}} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_{\text{input}} \\ y_{\text{input}} \\ 1 \end{bmatrix} \quad (18)$$

$$x'_{\text{output}} = \frac{ax_{\text{input}} + by_{\text{input}} + c}{gx_{\text{input}} + hy_{\text{input}} + 1} \quad (19)$$

$$y'_{\text{output}} = \frac{dx_{\text{input}} + ey_{\text{input}} + f}{gx_{\text{input}} + hy_{\text{input}} + 1} \quad (20)$$

6.2. Moving object projection

6.2.1. Experiments

Compared to most one-stage CNNs, RetinaNet [80] is able to produce more accurate predictions with slightly longer processing time. Due to its focal loss (FL) function differentiates background and foreground examples, the network applies larger weight updates for TP cases and makes less weight changes for the misclassifications (mostly background) during training. Equation 24-26 express the FL calculation. In Equation 24, p indicates the probability of one prediction, TP means this prediction is a true positive. In Equation 25, cross entropy (CE) is calculated based on the output p_t from the previous step. Lastly, FL is computed with Equation 26 where r is the focusing parameter which adjusts the weight change for negative examples such as background. Therefore, the experiment in this Chapter selects RetinaNet as the main architecture to implement.

$$p_t = \begin{cases} p, & \text{if } TP \\ 1 - p, & \text{otherwise} \end{cases} \quad (24)$$

$$CE(p_t) = \begin{cases} -\log(p_t), & \text{if } TP \\ -\log(1 - p_t), & \text{otherwise} \end{cases} \quad (25)$$

$$FL(p_t) = -(1 - p_t)^r \log(p_t) \quad (26)$$

In this chapter, the problem of calculating the real-world position and orthogonal mapping of the detected objects is approached from two angles: (i) projection from perspective to orthogonal based on reference objects' coordinates (PROC), and (ii) projection from perspective to orthogonal based on reference objects' size (PROS). Figure 38 illustrates the workflows of PROC and PROS approaches. In this Figure, four traffic cones are used as reference objects, and the goal is to calculate the orthogonal position of a moving person from drone-captured perspective views. In the PROC approach, first, Model-P is trained on a perspective video (PV1) and tested on another perspective video (PV2) to predict the pixel coordinates of the reference objects in each video frame. Next, from the known real-world positions of the reference objects, the real-world position of the object, i.e., $(x'_{\text{r}}, y'_{\text{r}})$, is calculated in PV2. While the PROC approach relies on the real-world positions of reference objects, the PROS approach uses the size of reference objects. The pixel coordinates of reference objects and ToI are obtained as before by using Model-P. However, the reference objects' real-world positions are calculated differently. In essence, another CNN model, Model-O, is trained on an orthogonal video (OV1) and tested on another orthogonal video (OV2) to predict the boxes of the reference objects. Since in the PROS approach, the actual sizes of reference objects are known, the ratio between the sizes of the predicted boxes in

pixel and their real sizes is used to transform the pixel coordinates of these objects to real-world positions. Next, the real-world position of the object is calculated in PV2 using Equations 15 through 20.

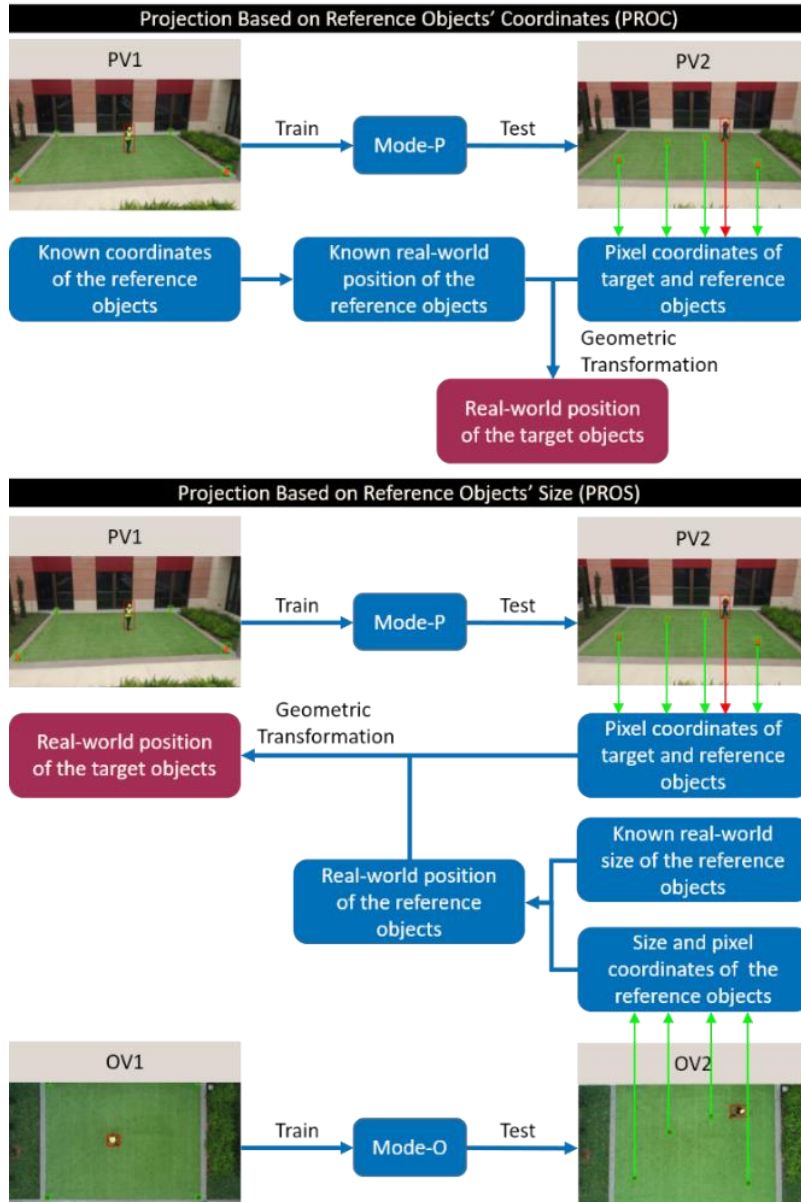


Figure 38. PROC and PROS experiments.

6.2.2. Data collection and description

To train and test RetinaNet CNN models that can automatically detect reference objects and ToIs, two experiments are conducted. Experiment 1 involves collecting perspective video PV1 (Figure 40) and orthogonal video OV1 (Figure 39), which provide the training data for the perspective model (Model-P) and orthogonal model (Model-O). Experiment 2 contains two videos, PV2 (Figure 42) and OV2 (Figure 41), that serve as testing data.

The objective of both experiments is to project a moving object captured in drone's perspective view into real-world (orthogonal) coordinates, using four reference points. Therefore, each experiment contains a continuously moving person and four traffic cones (reference objects). The cones and the person are located on a turf (333 in \times 426 in). To obtain ground-truth information, one drone (Parrot Anafi) records the scene from an aerial angle pointing the camera vertically downward (videos OV1 and OV2). Another drone (Parrot Bebop 2) records the same scene from a perspective angle and an arbitrary altitude (videos PV1 and PV2). The cones' positions in Experiment 1 (PV1 and OV1) are different from Experiment 2 (PV2 and OV2). Also, the person changes outfit between the two experiments. With the upper left corner of the turf designated as the origin of the real-world coordinate system, the real-world positions of the four cones are shown in Figure 3 through 6. For example, cones in Experiment 1 are placed at coordinates (0, 0), (0, 333), (426, 0), and (426, 333), whereas in Experiment 2, they are placed at coordinates (56, 263), (148, 138), (258, 95), and (356, 277).

The timestamps of PV2 and OV2 videos are carefully synchronized, leaving a total of 4,149 frames for projection. To train and test the CNN models for detection of cones and person, all collected videos are manually annotated with DarkLabel [174], frame by frame, as shown in Figure 37 through 40.



Figure 39. Orthogonal video 1 (OV1) of experiment 1.

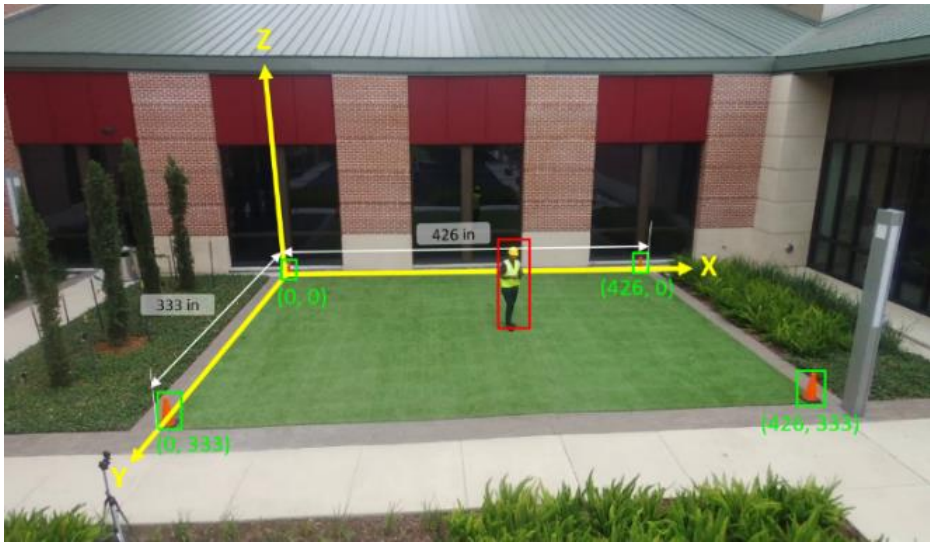


Figure 40. Perspective video 1 (PV1) of experiment 1.



Figure 41. Orthogonal video 2 (OV2) of experiment 2.



Figure 42. Perspective video 2 (PV2) of experiment 2.

In these Figures, green rectangles represent cones, and red rectangle represents person. Description of the experimental data is shown in Table 26. The ground-truth real-world positions of the person are retrieved from the manually annotated OV2 video. All frames within PV1 and OV1 are used for training while the frames in PV2 and OV2

are used for testing. For the training of Model-P and Model-O, the learning rate is set to 10⁻⁴ and batch size is set as 4.

Table 26. Data statistics for experiments 1 and 2.

Total Number	Experiment 1		Experiment 2	
	PV1	OV1	PV2	OV2
Total frames	8,873	9,915	5,261	5,814
Person instances	8,813	8,223	4,492	4,773
Cone instances	33,362	32,892	19,650	19,092

6.2.3. Projection Based on Reference Objects' Coordinates (PROC)

Model-P is responsible for detecting the pixel coordinates of cones and person in PV2. Figure 43 shows an example of detection of cones (yellow boxes) and person (white box). All detected boxes are associated with a confidence level predicted by the model. The four cones and one person that are detected with the highest confidence levels are selected for further analyses. Next, pixel coordinates, i.e., the center point of the bottom edge of each detection box (marked with dots in Figure 43) are determined as input for projection calculation. From the calculated pixel coordinates of four cones and their corresponding real-world positions, the person's real-world position is calculated (using Equations 15 through 20), and projected on the orthogonal map shown in Figure 8. Also, the annotated person's position in OV2 is projected on the same coordinate system which serve as ground-truth.



Figure 43. Example of detections of person and cones.

As expected, there may be situations when the PROC approach fails to project, e.g., when Model-P cannot detect at least four cones in the frame, because they are either not within the camera view or obstructed by other objects. In this experiment, such frames are removed from analysis. The number of skipped frames, presented as the percentage of total number of frames, is termed frame loss (Equation 27) which describes how efficiently the model utilizes the input video data. Moreover, due to the noise in data and/or detection error, sometimes the detected boxes of the same object (e.g., cone) in two consecutive video frames could appear in two considerably different locations. Moreover, some false detections appear significantly away from the ground truth. These outliers are removed by comparing the current and previous frame pixel coordinates, i.e., X- and Y- coordinates of each objects are treated as a timeseries data.. For example, consider a set of consecutive frames $C = \{c_t; t = 1, 2, \dots, n\}$, 15 previous consecutive data points $C' = \{c_t, c_{t-1}, \dots, c_{t-14}\}$ are selected and its mean c'_{mean} and standard deviation c'_{STD} are calculated. c_t indicates the coordinates in the current frame.

When $c_t - c'_{mean} \geq k * c'_{STD}$ ($k = 2$ and 3 are considered), frames corresponding to c_t are skipped. After removing the outlier predictions, the average projection error is recalculated.

$$frame\ loss\ (FL)\ \% = \frac{\#\ of\ skipped\ frames}{total\ \# \ of\ frames} \times 100 \quad (27)$$

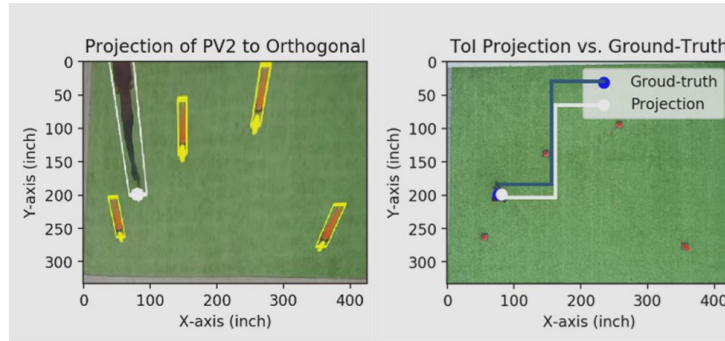


Figure 44. Perspective to orthogonal projection.

6.2.4. Projection Based on Reference Objects' Size (PROS)

As described earlier, the difference between PROC and PROS approaches is in the method for acquiring the reference objects' real-world positions. In the PROS approach, it is assumed that the size of an object in pixel units in the orthogonal image will represent a constant physical size throughout the entire video. Based on this assumption, from the known physical size of the target object, its actual position (in inches) can be calculated solely based on the model's predictions in the orthogonal view. In the experiments conducted, the actual dimensions of the traffic cone are known to be 10 inches (both length and width). However, due to the noise in data and/or human error, the annotated training boxes are slightly bigger than the actual size of the cones. From the training data (OV1), the width of the detection boxes for cones is calculated to be

14.56 inches on average. Therefore, it is assumed that the detected boxes for the same cones in the test data (OV2) would also have the same average physical size. During the test, Model-O is applied to the first 30 frames (i.e., 1 second) of OV2 to obtain the sizes and coordinates of the boxes representing cones in pixel units. Next, knowing that the size of the boxes are 14.56 inches in real-world, coordinates are scaled accordingly to obtain the real-world positions of the boxes. The latter part (orthogonal projection) of the PROS approach is similar to the PROC approach.

6.2.5. Results and analysis

Model-P and Model-O are tested on PV2 and OV2 to evaluate their performance, as shown in Table 27. Overall, Model-O produces 51% mAP which is worse than 97% for Model-P. One reason for this disparity is that from orthogonal view (in Model-O), the person and cones appear less distinctive compared to perspective view (in Model-P). For individual classes, Model-P detects class person with 99.21% average precision (AP); however, Model-O produces only 45.89% AP. Similarly, for class cone, Model-P produces 96.11% detection AP, compared to 51.97% AP for Model-O.

Table 27. Comparison of Model-P and Model-O performance.

Model	mAP	AP (Cone)	AP (Person)
P	97.66%	96.11%	99.21%
O	48.93%	51.97%	45.89%

Projection results obtained from PROC and PROS approaches are summarized in Table 28, and illustrated in Figure 45. For both PROC and PROS, the remaining frames

after removing non-projectable frames (because the Model-P detected less than four cones or one person while tested on PV2) is 3,735 with 9.98% frame loss for both projections. The Euclidian distance (in inches) between the person's real-world and projected positions is reported as projection error in each frame. The average projection error (APE) is calculated as the mean of frame-by-frame errors in all projectable frames. As shown in Table 5, the PROS approach achieves an APE value of 15.39 inches, which is slightly better than PROC's APE of 17.18 inches. Next, the projected person's real-world X- and Y- coordinates are compared with the corresponding ground-truth coordinates, frame by frame, and the errors are divided by the total length along each axis (i.e., 426 inches for X-axis and 333 inches for Y-axis) to present the error as a percentage. The errors in PROC approach are 9.52 inch (2.23%) along X-axis and 11.79 (3.54%) along Y-axis, which is slightly outperformed by PROS approach with errors of 8.62 inches (2.02%) along X-axis and 10.41 inches (3.12%) along Y-axis.

Furthermore, two scenarios are considered for removing the outliers: $k = 2$ and $k = 3$. For $k = 3$, after removing outliers, a total of 1,493 (FL = 64.02%) and 1,488 (FL = 64.14%) frames remain in PROC and PROS, respectively, and the overall APE is improved to 13.86 inches (for PROC) and 11.86 inches (for PROS). When removing outliers with $k = 2$, calculated APEs (overall, X-, and Y-) for both approaches are greater than the case of $k = 3$, but smaller FL (26.63% for PROC and 26.71% for PROS) is achieved. Errors along X- and Y-axis for individual frames are documented in Figure 45. In general, it is observed that removing outliers leads to higher frame loss but smaller

APE in both approaches. Nonetheless, for all cases, the average error along any dimension is less than 4% which indicates the robustness of the proposed methods.

Table 28. Test results of PROC and PROS mapping methods.

	Method	Direct projection	Outlier ($k = 3$)	Outlier ($k = 2$)
Remianed Frame # (frame loss %)	PROC	3,735 (9.98%)	3,044 (26.63%)	1,493 (64.02%)
	PROS	3,735 (9.98%)	3,041 (26.71%)	1,488 (64.14%)
Overall APE (inch)	PROC	17.18	14.35	13.86
	PROS	15.39	12.31	11.85
X-axis APE (inch and %)	PROC	9.52 (2.23%)	7.11 (1.66%)	6.53 (1.53%)
	PROS	8.62 (2.02%)	5.89 (1.38%)	5.37 (1.26%)
Y-axis APE (inch and %)	PROC	11.79 (3.54%)	10.89 (3.27%)	10.87 (3.26%)
	PROS	10.41 (3.12%)	9.55 (2.86%)	9.50 (2.85%)

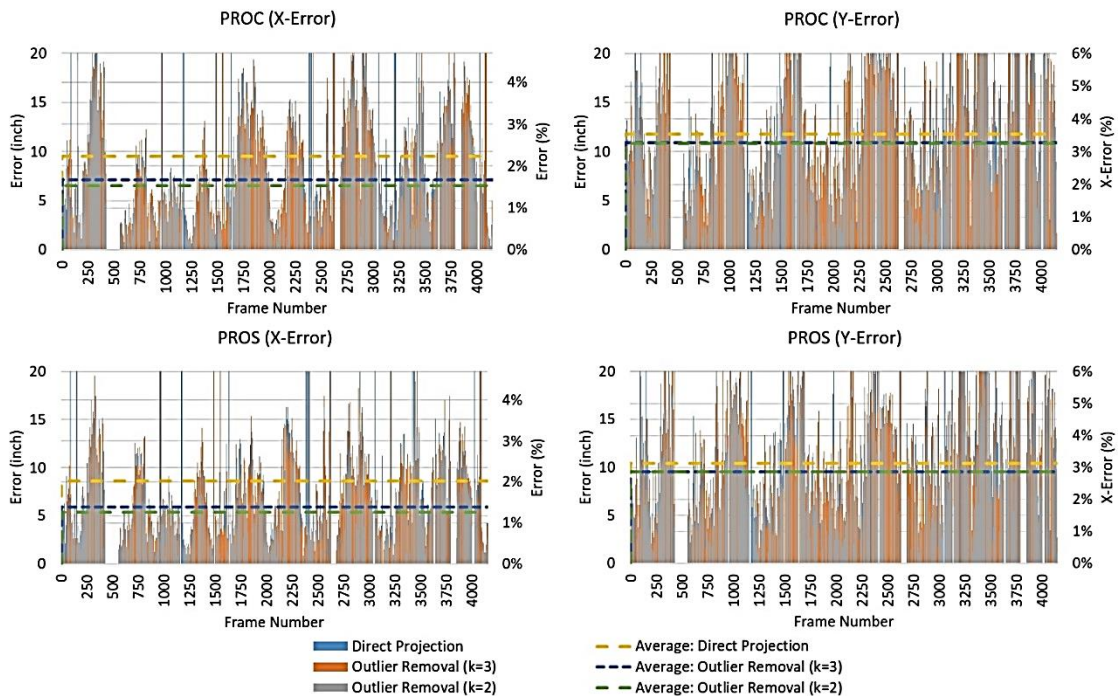


Figure 45. Projection error in X- and Y- axis for PROC and PROS.

6.3. Object tracking and counting

6.3.1. CNN-based centroid tracker

In computer vision, object tracking entails estimating the state and location of an object in the sequential frames based on the initial state [193, 194]. Furthermore, object tracking can be used to identify and differentiate multiple objects in the frames, thus allowing object counting [195]. Several object trackers have been previously studied. For example, boosting tracker uses a binary classifier that continuously updates possible object positions for the next frame, making it suitable for tracking changes in object rotation or illumination [196]. KCF map the image input with circulant matrices, followed by diagonalizing it with fast Fourier transformation. Research has shown that KCF outperforms boosting tracker on both accuracy and speed [197]. Similarly, minimum output sum of squared error (MOSSE) is a high speed tracker capable of processing images at 669 FPS. The correlation filter in MOSSE is trained on data that is processed by FFT to generate the next frame status [198]. In addition to these object trackers, several researches have investigated the use of sequential information from the videos. Zhang et al. [199] designed a FCN combined with long short term memory (LSTM) networks to count vehicles from city cameras, and achieved mean absolute error (MAE) of 4.21. Bagautdinov et al. [200] built an RNN to localize and identify activities of people in videos. Fan [201] investigated emotion recognition in videos using a combination of CNN and RNN achieving 70.74% accuracy. While these methods successfully utilize information in successive frames, they require specific time-related datasets. The CNN models utilized in this Dissertation can detect objects in each frame

and output object locations in the form of bounding boxes (Chapter 4) or pixel classifications (Chapter 5). This information may serve as the initial states for the purpose of object tracking. As a drone or helicopter flies over a disaster-affected scene, objects on the ground may move in or out of the camera view. Therefore, for the purposes of tracking and counting, it is important to identify existing as well as newly appeared objects without interruption, while avoiding double-counting the same object in multiple video frames.

Centroid tracking relies on the Euclidean distance between the object centroids in initial and sequential frames [202]. Through registering and unregistering objects, this technique is capable of differentiating old and new objects. For example, let's consider event I in Figure 46, and assume that the detection model has initially marked the bounding boxes (colored in blue) of two objects, shown with ID:1 and ID:2, together with their centroids. In the next frame, as a result of the moving objects or camera viewpoint, the locations of these two detections in the frame also moves, resulting in two new bounding boxes (colored in orange) with corresponding sequential centroids. When applying centroid tracking to a high-FPS video, it is safe to assume that the relative movements of objects (change of centroid location of each object) in successive frames is relatively small. Mathematically, this can be expressed as the Euclidean distance D from the initial centroid to the sequential centroid of the same object being smaller or larger than distance D' (the distance from an object's initial centroid to sequential centroids of other objects). In Figure 46, if $D < D'$, instead of registering the two new

detections in event I (colored in orange) with new IDs, they will be associated with old ID:1 and ID:2, resulting in the total object count of 2 (unchanged from the initial frame).

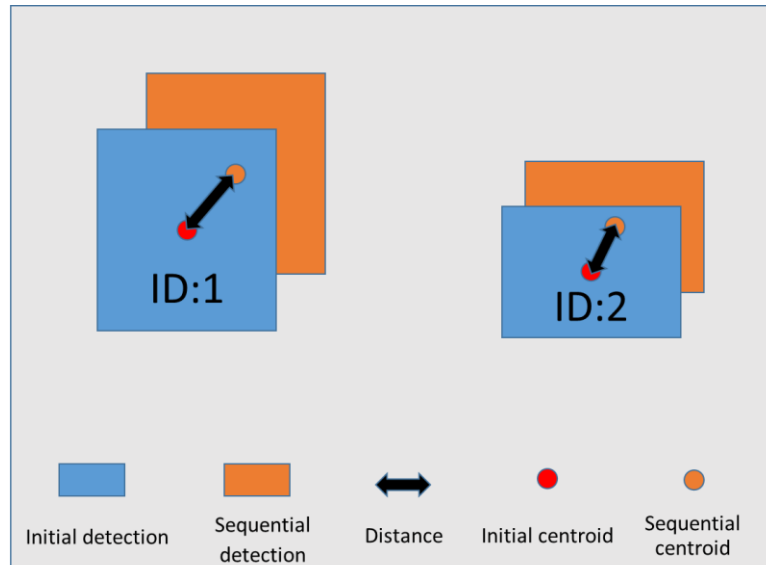


Figure 46. Detections, centroids, and distances in event I.

Now, consider event II in Figure 47, and assume that the detection model has initially marked the bounding boxes (colored in blue) of two objects (ID:1 and ID:2), and later detected a third object. Since the Euclidean distance D from this object's initial centroid (colored in green) to the centroid of the other two objects is large enough, this detection is registered as a new object with ID:3. Repeating the same logic in all future frames should allow for old and new objects to be reasonably tracked and counted. For objects that move outside the camera view (either permanently or temporarily), their IDs will be deregistered after a certain number of successive frames, a.k.a., frame memory or

N , which can be adjusted for each specific application, camera properties, speeds, and the preference.

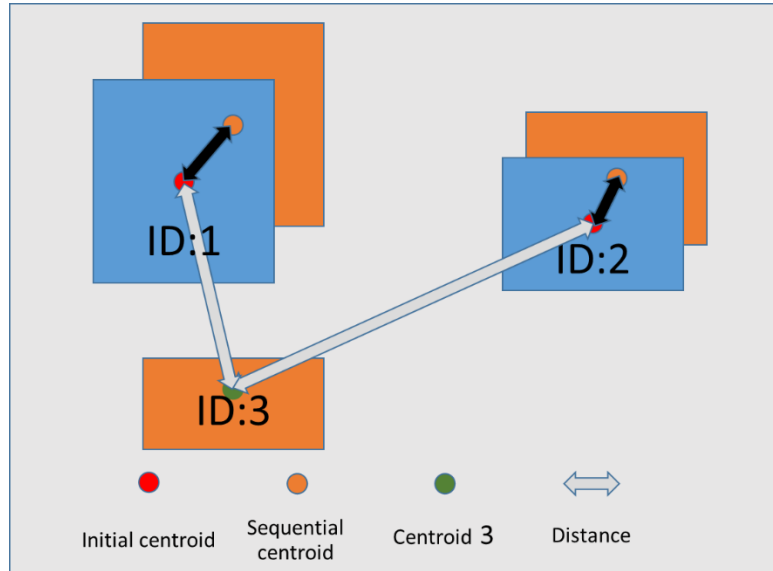


Figure 47. Detection, centroids, and distances in event II.

6.3.2. IoU-based ID assignment using Hungarian algorithm

Besides using the Euclidean distance and assigning IDs for the closest pairs, other methods such as IoU score can be utilized for object tracking. Bewley et al. [203] integrated Hungarian algorithm (IoU-based) and Kalman filter, and created SORT tracker which achieved 33.4% tracking accuracy on multiple object tracking dataset (MOT). Hungarian algorithm is a linear assignment algorithm that minimizes the total cost based on the assumed bipartite graph [204]. Figure 48 displays the cost matrix in the object tracking setting with rows indicating detections in the previous frame and columns representing detections in the next. By initializing an ID to every detection in

the first frame, the followed tracked detections will maintain their unique IDs. In the cost matrix, cells contain IoU values produced by comparing individual detections between the pervious and next frames. Given this matrix, row and column reduction is implemented to find the zero-cost path in the matrix. In row reduction, each non-zero cell in the first row will be subtracted with the minimum value in that row. A similar operation is carried out in all remaining rows. Similarly, for column reduction, the minimum value of each column is subtracted from each cell in that column. The resulting matrix may have to be augmented if the minimum number of lines that can cover all zeros is greater than the matrix dimension (rows and columns). This augmentation is done by subtracting the minimum cell value (remaining cells not covered by the lines) from all the remaining cells. Ultimately, detection pairs in the previous and next frames that correspond to the zeros in the cost matrix are assigned with the same IDs. The remaining unassigned detections in the next frame will be marked with new IDs. In this research, both centroid tracker and Hungarian tracker are shown to assign unique IDs to objects and counting the number of objects belonging to a particular class, such as cars or damaged roofs, without reliance on drone's geolocation (i.e., GPS data).

	1	2	...	j
ID: 1	IoU_{11}	IoU_{21}	IoU	IoU_{j1}
ID: 2	IoU_{12}	IoU_{22}	IoU	IoU_{j2}
...	IoU	IoU	IoU	IoU
ID: i	IoU_{1i}	IoU_{2i}	IoU	IoU_{ji}

Figure 48. Hungarian cost matrix.

6.3.3. Experiments and results

To test the validity of the both centroid tracking method, a face recognition CNN with 87.48% AP proposed by Li et al. [205] is used to detect and count human faces in an office space. The video is taken by a laptop camera and contains three faces which frequently appear and disappear in the video. Frame memory (N) is set at 50, implying that if an object (in this case, face) remains outside the camera view for 50 successive frames, its ID will be deregistered. Figure 49 illustrates selected frames from the office video. In Figure 49 (a), the first human face is detected and registered as ID:0. Next, two more faces appear in the camera in Figure 49 (b), which are subsequently detected and registered as ID:1 and ID:2. In Figure 49 (c), ID:0 and ID:2 faces move out and stay out of sight, followed by ID:0 face moving back into the camera view in less than 50 frames (i.e., frame memory, N), and ID:2 face moving back after 50 frames, as shown in Figure 49 (d). As a result, while face ID:0 maintains its original ID, the face that was initially registered as ID:2 is now assigned a new ID:3, thus detected as a new face. The outcome of this small-scale experiment is shown in Table 29 (first row).



Figure 49. Face tracking and counting.

In order to compare the centroid tracker and Hungarian tracker, the next set of experiments involve actual drone footage and are conducted to investigate the correlation between object detection performance and the reliability of object tracking. Figure 50 shows a video frame from experiments described in Subsection 6.2.1, with the goal of counting the number of people and cones. In video PV2, the person blocks the view of the cones in multiple frames, and also moves out and in the camera view several times. Figure 51 shows the output of YOLO model 3 (Section 4.4) with an AP of 78.87% for detection of undamaged roofs, applied to a segment from Volan003 video (Section 4.1). The AP for each model and the deviation between the true number of objects (i.e., ground truth) and the output of object counting using the previously introduced centroid tracking technique is reported in Table 29.



Figure 50. Person blocks the cone in PV2.

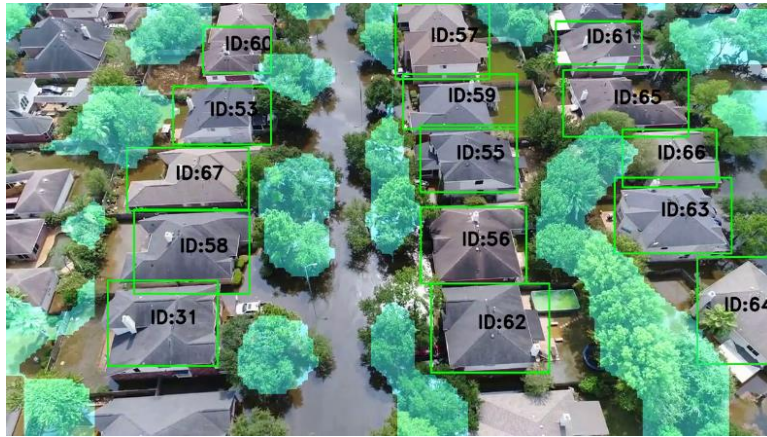


Figure 51. Roof counting based on YOLO model3.

Considering the centroid tracking method, it can be seen from Table 29 that CNN models with higher AP produce better results. For example, Model-P with the highest AP results in zero deviations from the ground truth count of cones and person. On the other hand, Model-O has the lowest APs of 51.97% (cone detection) and 45.89% (person detection) among all models listed in Table 30, which lead to a higher number of deviations from the ground truth count (e.g., 8 for person). Finally, counting undamaged

roofs with YOLO model 3, the deviation from the ground truth count is 4. In conclusion, the designed centroid tracking technique is capable of assigning unique IDs and tracking detected objects using the information in sequential frames. It is also found that higher object detection precision generally leads to better object counting and tracking. On the other hand, the IoU-based Hungarian tracking method is not always better than the centroid object tracker. As Table 29 shows, this method only outperforms centroid tracking when Model-O is used to count cones. This could be explained considering that when the CNN model fails to detect the same object in consecutive frames, the Hungarian algorithm keeps assigning new IDs to that object, resulting in over-counting. Compared to centroid tracking, Hungarian tracking produces larger deviations in all experiments. It can thus be said that centroid tracking is more suitable for object counting in this study.

Table 29. Centroid tracking and IoU based Hungarian tracking tests.

Model	Experiment	AP (%)	Ground truth	Centroid tracking: count (deviation)	Hungarian algorithm: count (deviation)
Face	Face in office	87.48	3	4 (+1)	N/A
Model-P	Cones in PV2	99.21	4	4 (0)	7 (+3)
Model-O	Cones in OV2	51.97	4	5 (+1)	4 (0)
Model-P	Person in PV2	96.11	1	1 (0)	1 (0)
Model-O	Person in OV2	45.89	1	9 (+8)	12 (+11)
YOLO model3	Roof counting	78.87	62	66 (+4)	171 (+109)

6.4. CNN integration and mapping

6.4.1. Static frame projection

Two example classes are used to demonstrate the technique for projecting detected objects on a 2D orthogonal map in a real disaster setting. Four building roofs are marked as reference points with their real-world positions obtained from Google Earth, and all other detected building roofs and two cars are projected from drone's perspective view to a 2D map following the universal transverse Mercator (UTM) coordinate system. The input drone video (Volan003) is from Houston, TX, and covers zone 15R of the UTM system from Volan2018. Two example frames of this video in which detected undamaged roofs (including reference points), flooded areas, and cars are marked is shown in Figure 52. Model 2 which is pre-trained on VOC and re-trained on drone balanced subset (Section 4.1) is applied on frame #12210 and #12374.

Considering frame #12210 in Figure 52 (a), from the real-world positions of all undamaged roofs, those corresponding to the reference points 1-4 are used to project all other detections. In Figure 53 (a), these four points are marked with numbered circles, and all other detected undamaged roofs are marked with rectangles numbered 5-14. Real-world positions of these undamaged roofs serve as ground truth to measure the projection error. Figure 53 (b) shows the frame projection on UTM system. For each instance of undamaged roof, the Euclidean distance between the ground truth and projection is measured. Each projection is then paired with the closest ground truth, and the corresponding Euclidean distance is recorded as error. The frame-level projection

error is then calculated using Equation 29, in which n is the total number of detected objects in that frame. In addition, the absolute discrepancies along Easting and Northing axes are determined, and the average Easting and Northing distances of all pairs in one frame are defined as Easting-error and Northing-error, respectively.

$$frame\ error = \frac{1}{n} \cdot \sum_{i=1}^n error_i \quad (29)$$



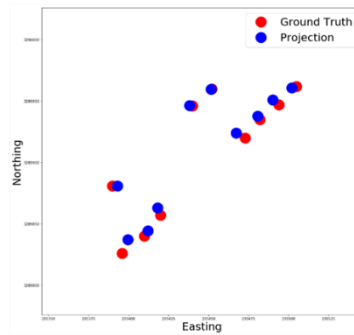
Figure 52. Examples of projection input from Volan003 video.

Projecting undamaged roofs onto a 1-second long segment between frames #12210 and #12240 (drone moving to the left in 30 consecutive frames) of Volan3 video yields a frame error of 7.18 meters, and an average Easting-error and Northing-error of 3.41 meters and 7.28 meters, respectively. Considering the size of the drone’s visible area in the frame ($173.95 \times 130.53 \text{ m}^2$), this projection error translates to approximately 1.9% Easting-error and 5.5% Northing-error, indicating the high accuracy of the designed technique for transforming perspective drone views to orthogonal maps. As shown in Figure 52 (b), two car instances are additionally detected in frame #12374 of Volan003 video, which are also projected along with building roofs, as shown in Figure

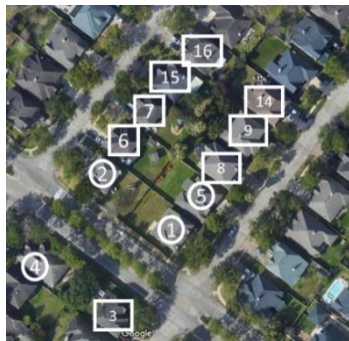
53 (c) and 50 (d). In this projection, detected roofs 1, 2, 4, and 5 are selected as reference points, yielding a 7.79-meter (or 5%) frame error considering both classes. It must be noted that the change of viewpoint (as a result of drone moving in the scene) may cause any or all of reference points (i.e., building roofs) in one frame to fall out of sight, in which case, new reference points are needed to have an uninterrupted projection. For instance, comparing Figure 53 (a) and (c), reference point 3 is replaced with a new reference point 5, using an ad-hoc reference point selection through tracking and updating reference points on the fly.



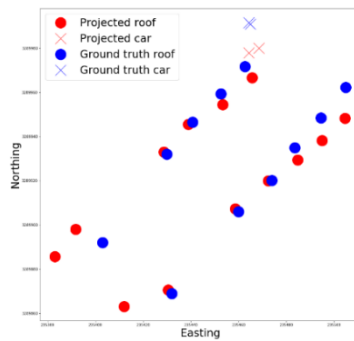
(a)



(b)



(c)



(d)

Figure 53. Comparison of object projection results.

6.4.2. Reference points updating and calibration

As examples in the previous Subsection demonstrated, when covering large areas from the air, one or more of the four known reference points initially selected could move out of or back into the camera view. Therefore, it is necessary to have a contingency for updating and calibrating the required number of reference points on the fly. Figure 54 shows a frame from Volan003, taken by a drone in Houston, TX in 2017 during hurricane Harvey, with scene I and scene II marked with transparent colors representing the view from a drone camera at two different times. Similar to the previous example, undamaged roofs are selected as reference points. In Figure 54, all roofs are numbered from 1 to 19 and only the real-world positions of roofs 2, 3, 5, and 6 are obtained from Google map. In scene I, detected roofs 2, 3, 5, and 6 together with their real-world positions serve as initial four reference points. Next, using the homography transformation described in Section 6.1, drone's perspective view is transformed to an orthogonal coordinate system, and roofs 1, 4, 7, 8, 14, 15, and 16 are projected on the UTM system. Once the drone camera moves to scene II, roofs 2, 3, and 5 are no longer in the view. However, the remaining detected and mapped roofs, i.e., 7, 8, 14, 15, and 16 from scene I can now serve as new reference points. After selecting four new reference points from roofs in scene II, all other roofs detected in this scene can be projected on the UTM system. The application of this technique to successive video frames guarantees that regardless of how the position and angle of the aerial camera changes, detected objects can be mapped in an orthogonal coordinate system.



Figure 54. Reference points updating for homography transformation.

It is likely that this method of ad-hoc updating of reference points results in accumulated errors since projection is based on the location of detected bounding boxes, which is subject to the accuracy of object detection model. Over time, relying on previous projections to update the location and count of objects may lead to large random errors. A potential remedy to this problem is to implement calibration while updating the reference points. As shown in Figure 55, to achieve best calibration results, instead of projecting the centroids of building roofs, detected bounding boxes are compared with the ground truth polygons in the UTM system. For this specific example, this ground truth information is obtained from the U.S. building footprint GitHub repository [200]. Inside the visible area, each projection is compared with neighboring ground truth objects to measure the IoU. The 4 ground truth roofs (marked with color dots) with the highest IoU are considered as successful projections and are therefore selected as reference points for the next frame. After determining the four reference

points in the orthogonal view, the corresponding pixel coordinates of these roofs are selected using the object tracking technique mentioned in Subsections 6.3.2.

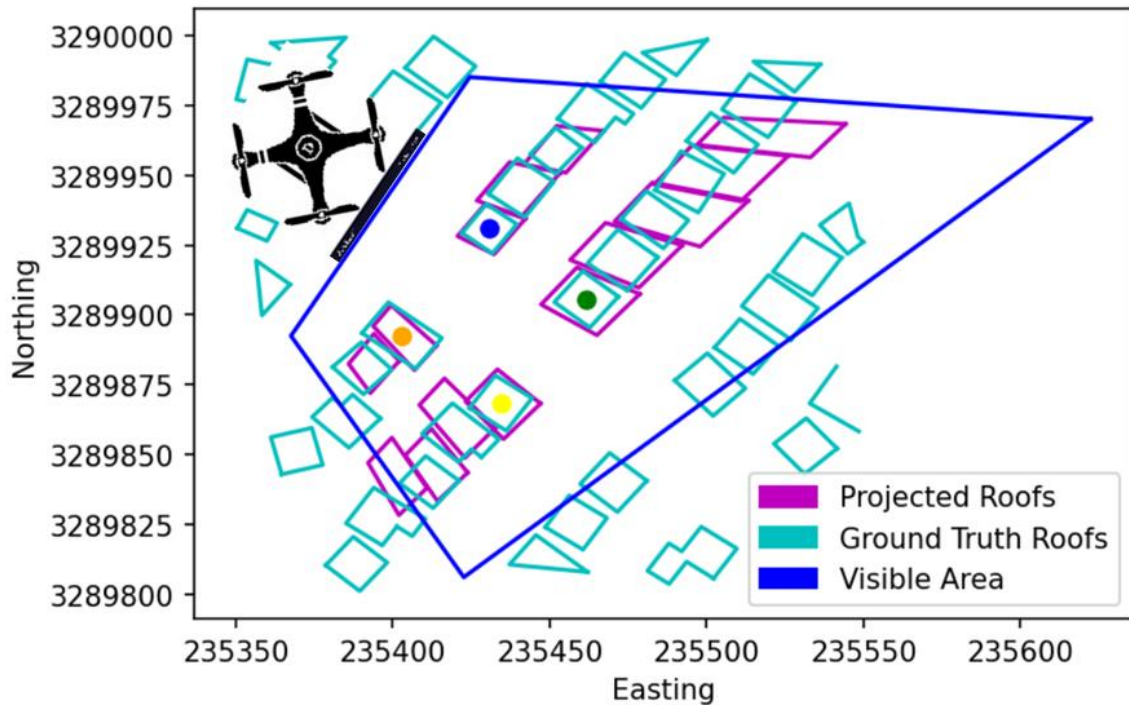


Figure 55. Example of reference point calibration.

6.4.3. Experiments and results

In order to test the performance of the reference point calibration technique, YOLO model 3, trained on Volan2018 drone unbalanced data (Chapter 4) is applied to 100 successive frames from Volan003 video. Autoencoder model PSPNetH (PHA8) (Chapter 5) is also applied on the same frames to generate flood detections, as shown in Figure 56 (a). As mentioned earlier, the ground truth roof map is obtained from the U.S. building footprint dataset [206]. Four initial reference points in the first frame are manually selected, and calibration is applied to project flood detections frame by frame,

as shown in Figure 56 (b). Colored lines between these two figures represent the correspondence between the paired reference points. From this Figure, the area covered in flood water is measured to be 4,725 square meters (approximately 5,651 square yards). If the depth of floodwater is known (which is beyond the scope of this research), a volumetric estimate of flood can be obtained, which is an extremely useful information for flood response, mitigation, and cleanup activities. The same calculation can be applied to other classes such as debris and damage roof, with unit cost, to estimate the operation cost.

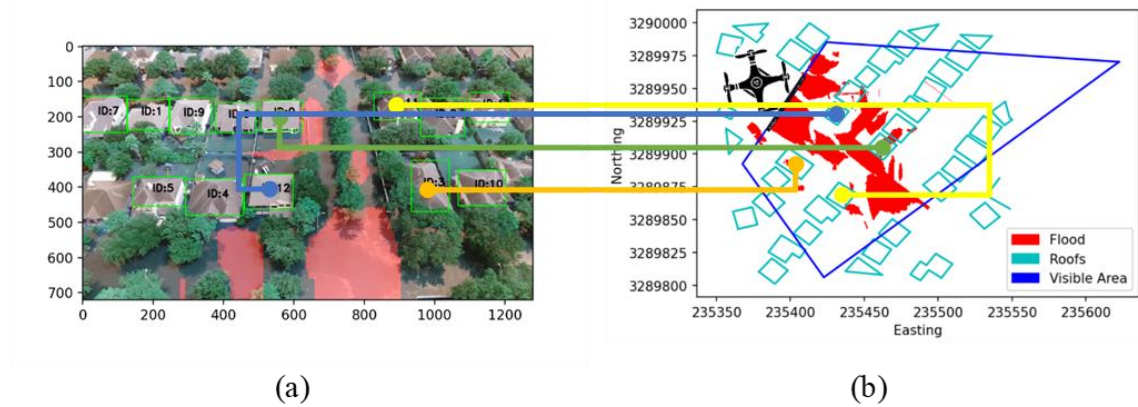


Figure 56. CNN combination and projection.

It must be noted that while homography transformation by itself does not introduce a new source of error, its output quality is directly proportional to the accuracy of object detection. As evidenced by the examples above, the perspective to orthogonal mapping process is not a linear operation. In other words, since objects closer to the camera viewpoint appear closer to the bottom section of the video frame, a detection error in the upper section of the input image results in a larger mapping error than a

detection error in the bottom section of the input image. This concept is demonstrated using an experiment in Figure 57 (a). In this example, the goal is to project areas marked as Detection1 (closer object) and Detection2 (farther object) onto an orthogonal map. Both Detection1 and Detection2 are isosceles trapezoids of the same size (100-pixel upper base, 200-pixel bottom base, and 200-pixel height). Figure 57 (b) shows the outcome of projecting these two detections onto a map with UTM coordinates. The area of Detection2 projection is calculated as $1,117\text{m}^2$, compared to the smaller 278 m^2 area of Detection1 projection, which validates that the perspective to orthogonal projection is nonlinear. Considering this effect, in order to filter out large errors in the homography transformed map, users could choose to map only portions of each frame that contain detections of closer objects, or calculate errors based on the output of the homography transformation if ground truth orthogonal information is available.

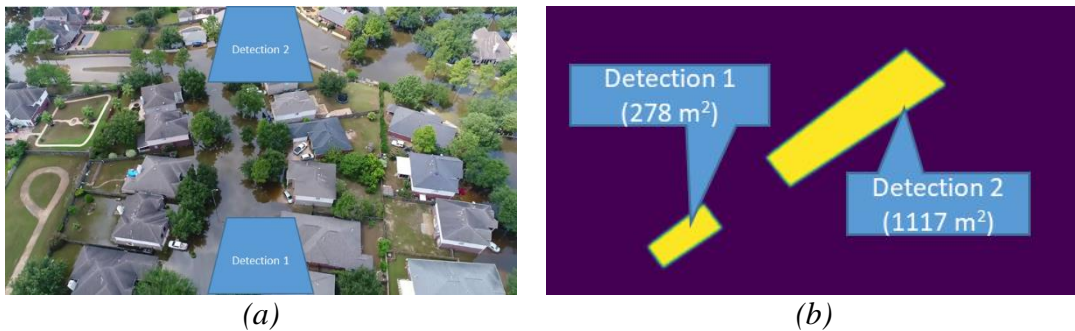


Figure 57. Nonlinearity of homography transformation for close and distant detections

6.5. Summary

In this Chapter, a viewpoint transformation technique based on homography method was introduced to convert object detections (bounding boxes, masks, pixel

classifications) from drone-captured perspective views into the world coordinates of orthogonal maps (i.e. UTM system). In addition, two tracking methods, namely centroid tracking and Hungarian algorithm were utilized to track object movements and count them in successive video frames. The need for this operation was justified through describing practical examples of how timely and reliable collection and delivery of disaster impact information could add value to disaster management. Several experiments were conducted to track and count objects (i.e., human faces in an office, person and cones in a courtyard, and building roofs and cars in drone footage). Results suggest that objects can be mapped with high accuracy. In order to address the limitation of reference point selection and the problem of error accumulation, a calibration mechanism was developed to update reference points on the fly and as the camera viewpoint is moving. An experiment was conducted using a segment of Volan003 (from Volan2018 dataset) video, which resulted in an orthogonal map of flooded areas, and an accurate estimate of the land area covered in floodwater.

It must be noted that for better results and to improve the quality of decision-making, the output of the mapping procedure must be used in combination with other geocoded information available from the same area. For instance, without knowing if a projected flooded area depicts a flooded neighborhood or simply corresponds to an existing pond or lake, it is difficult to make an informed decision about the severity of flood and the level of resources that need to be deployed. To this end, it is recommended that all detection projections are further superimposed onto street maps that contain the boundaries of existing bodies of water (e.g., rivers, ponds, lakes) and shorelines. This

will allow the user to distinguish an open body of water from flood, and determine the areas with excess water that need immediate attention. Another use case is when there is a need to differentiate debris origins, e.g., whether a debris field belongs to a building that used to stand in a particular location, or it has been washed away and brought to this location from another site.

7. CONCLUSION AND FUTURE WORK

This Chapter provides a summary of research findings and contributions of this Dissertation to the body of knowledge and practice, as well as offers potential directions for future work in this field.

7.1. Findings and Contributions

The success of disaster mitigation is highly influenced by the extent to which first response teams have access to timely and accurate damage assessment information. Existing methods of damage assessment still rely heavily on manual processing of large volumes of heterogenous data, and are thus inefficient and resource intensive. This Dissertation investigated the use of artificial intelligence in the form of CNNs to augment current disaster management capacities through automatically extracting and mapping high-value disaster impact information in aerial footage.

In particular, two in-house aerial video datasets, Volan2018 (box labeled) and Volan2019 (pixel labeled), were created. Several types of CNN models were then trained on these datasets to carry out visual recognition, object localization, mapping, and damage quantification. The primary theoretical contributions of this research include a thorough description of the methods to build and annotate disaster-related datasets, investigation key factors that influence CNN model performance, and techniques for GPS-denied object localization and mapping. From the practical perspective, this work introduced two fully annotated disaster visual datasets and offered practical suggestions for implementing the proposed methods in the field. Moreover, the trained CNN models laid the foundation for future research in AI-enabled disaster mitigation. The following

paragraphs describe the scientific contributions of this research to the body of knowledge and practice.

7.1.1. Disaster-domain multi-class visual datasets

A key challenge (Chapter 3) in CNN model training and testing, and quantifying model performance in making reliable predictions is the lack of fully annotated visual datasets in the disaster management domain. The choice of CNN architecture determines the type of annotation (box labeled vs. pixel labeled). As described in Chapter 4, the following steps were taken to create Volan2018 and Volan2019 datasets:

- Create a list of objects of interest (relevant to the problem domain) from the review of literature surveys, focus groups, or field observations.
- Collect available visual data (videos or photos) from the internet, public and private sources (to the extent possible), and other individuals (after securing the necessary approval).
- Annotate data by marking the types and locations of objects, while paying special attention to the intended output type of the CNN model (i.e., classification, bounding box, or segmentation), as it determines the level of detail in the annotation.

Together, Volan2018 and Volan2019 datasets include 64,265 annotated video frames and cover different geographical locations.

7.1.2. CNN model selection and influencing factors

In order to improve the CNN model performance, this research took on the challenge of investigating several performance influencing factors (Chapter 3). CNN

models including YOLO [79], RetinaNet [80], PSPNet [82], and Mask-RCNN [81] were trained using transfer learning (pre-trained on COCO [131], VOC [124], and ImageNet [129]) and retrained on Volan2018 and Volan2019 datasets. These architectures had different layers, loss functions, and annotation requirement, thus making it important to select CNNs based on metrics such as speed, accuracy, and output format. It was found that individual object classes such as car, people, building roof, and boat were suitable for object detection CNNs and bulk classes including flood, debris, vegetation, and road were better predicted using autoencoders. Other key influencing factors that were investigated in this Dissertation are listed below:

- Camera viewpoint altitude: CNN models trained and tested on videos captured from the relatively similar altitude yield best results, while cross-training and testing leads to very low accuracy.
- Pre-trained weights: Models pre-trained on VOC and re-trained on balanced data yield best results when tested on both drone and helicopter footage.
- Data balance: While tested on completely unseen footages, data balancing (down-sampling) improves model performance. Moreover, the suggested balance ratio (BR) can be used for data augmentation to improve the performance of minority classes.
- Object size: it is proven that there is a correlation between object size and the CNN performance. In particular, for any given class, larger objects tend to yield higher AP of prediction.

- Confidence and precision: A positive correlation between prediction confidence and precision exists. This correlation is of extreme value when predictions are to be made on new footage with no ground-truth information. While no ground-truth information prevents the direct calculation of precision, prediction confidence is independent from ground-truth information, and thus can be obtained directly from the model output. Confidence measurements can be therefore used to indirectly estimate prediction precision. Furthermore, this correlation can be used as an objective metric to guide class separation (based on confusion) among multiple CNN models to maximize performance of each model.

7.1.3. GPS-denied localization and mapping

Due to the inherent misalignment between perspective and orthogonal views, the output of the CNN models may not be readily useful for assessing the extent and quantifying the damage. To address this challenge (Chapter 3), this Dissertation introduced a homography-based mapping framework, and it was shown that CNN detections can be projected from perspective views onto orthogonal maps, given only the real-world positions of four initial reference points (e.g., ground landmarks). A centroid-based object tracking technique based on CNN was integrated with reference point updating to enable object counting and GPS-denied mapping in consecutive video frames using vision-only data. Beyond the immediate scope of this research, this method can be used in other applications where GPS signals are absent or unreliable including indoor construction, underwater navigation, battlefield, and space exploration. Once

CNN detections are mapped in an orthogonal coordinate system, basic geometric calculations can be done to obtain quantity estimate such as length (e.g., of flooded road) or area (e.g., of flooded area or debris field).

7.1.4. Social vulnerability indicators

As stated in Section 2.6, social vulnerability varies over time and space, among different social groups, based on diverse natural environments. The outcome of this research can lead to creating affordable and scalable solutions for producing fast and accurate disaster damage assessment based on VGI shared by ordinary people. When juxtaposed with social vulnerability data, these loss maps can draw more attention to communities that are disproportionately affected by natural disasters. Such vulnerability data include the Centers for Disease Control (CDC) SoVI map [207], small area income and poverty estimates (SAIPE) map [208], and related information layers from the United States Census Bureau [209]. Since the degree of generalizability and the scalability of output (i.e., performance) of the proposed approach rely primarily on the quality of collected training data, targeted data collection from communities that are historically impacted the most in natural disasters should be considered as a strategy. The above vulnerability maps can be used to this end to ensure collecting diverse and statistically representative data, and to eliminate the implicit bias along socio-economic lines.

7.2. Future work

In exploring a new domain of AI application in this Dissertation, namely aerial reconnaissance for disaster response and mitigation, several assumptions were made, and

a number of limitations were encountered which could be further investigated in future studies. Firstly, datasets Volan2018 (Chapter 4) and Volan2019 (Chapter 5) contain in total 22 different disasters (mainly hurricanes) that took place in various locations (mainly in North America). However, given the global footprint (type, number, frequency, severity) of natural disasters, these datasets cover only a small fraction of a much larger pool of data. Therefore, a potential direction for future work is to collect, curate, and build larger datasets, possibly using crowdsourcing, in order to support more robust CNN model training and testing for disaster management. In addition to hurricanes, new data should expand to other types of disasters such as earthquakes and landslides, which leave behind visually different damage patterns, calling for specialized datasets to train, validate, and test customized CNN models. As stated in Chapter 6, ground truth information such as flood location and area, and volume and type of debris are necessary to evaluate the performance of disaster information retrieval systems that result from this work. However, this ground truth data is generally rare and hard to collect. To this end, a potential future research direction is to gather and benchmark the scarce ground truth data in support of technology acceptance and development.

In Chapter 5, the concept of class separation (based on detection error) among multiple CNN models was introduced to maximize the performance of CNN models. More research can be carried out along these lines by, for example, creating stacked or stepped architectures that tie together multiple smaller CNN models, and use intermediate learned features from one model as input to other model(s), thus increasing the inference efficiency in cases where classes tend to be confused or when there is

limited training data. Moreover, with better computing capacities, Bayesian ensemble learning [210] can be utilized, without compromising computational speed, to optimize posterior detections and achieve better overall performance by combining the output of individual CNN models.

Additionally, performance measurement metrics such as mAP and IoU that are well-established and widely used in the computer vision may not be good indicators of how useful the developed CNN models are when deployed for disaster response. To this end, work needs to be done to establish and document target mAP or pixel IoU standards for application development in disaster management by developing human performance benchmarks such as processing speed and accuracy in similar scenarios, thereby providing ideal performance goals for computer vision application development, and introducing realistic benchmarks to compare the performance of CNN models with what humans can achieve. Several researches have attempted to benchmark the performance of AI with that of a human. For instance, Chen et al. [211] compared CNNs with humans in a handwritten character recognition task, and concluded that CNN models yielded 0.6% less error than humans. Stallkampa et al. [212] conducted an experiment to examine the performance of a human and CNN model in traffic sign recognition using the German traffic sign recognition benchmark (GTSRB) dataset. In their work, the best CNN model achieved a 99.46% correct classification rate, outperforming the best performing human group. Taigman et al. [213] compared the performance of deep learning and human in face recognition tasks, and suggested that the deep learning method could closely approach human level performance. It is important to note,

however, that in all such work, the ground truth information is also generated by human annotators, and as such, a detection accuracy that can approach or outperform human capability is acceptable for practical applications. In the context of disaster management, similar benchmarks can be determined by recruiting professionals to identify and locate objects. In doing so, special attention must be paid to the tradeoff between detection speed and accuracy (i.e., while a CNN model may produce less accurate detections, it certainly outperforms humans in speed by an order of magnitude), and their importance to the intended outcome (faster detection for immediate search and rescue vs. more accurate detection for long-term mapping and recovery). However, at present, these objective metrics do not exist in the field of disaster management.

REFERENCES

- [1] R. W. Perry, "What is a disaster?," in *Handbook of disaster research*. New York, NY, USA: Springer, 2007, pp. 1-15.
- [2] L. Weisæth, "A study of behavioural response to an industrial disaster," *Acta Psychiatrica Scandinavica.*, vol. 80, pp. 13-24, 1989.
- [3] J. Levinson and H. Granot, *Transportation disaster response handbook*. Cambridge, MA, USA: Academic Press, 2002.
- [4] G. Ackerman and J. Giroux, "A history of biological disasters of animal origin in North America," *Revue scientifique et technique-Office international des Epizooties.*, vol. 25, no. 1, p. 83, 2006.
- [5] Center for Research On The Epidemiology Of Disasters. "Natural Disasters 2018," https://emdat.be/sites/default/files/adsr_2018.pdf. (accessed May. 20, 2020).
- [6] Center for Research On The Epidemiology Of Disasters and United Nations Office for Disaster Risk Reduction. "Economic Losses, Poverty & DISASTERS 1998-2017," https://www.cred.be/sites/default/files/CRED_Economic_Losses_10oct.pdf. (accessed May. 12, 2020).
- [7] J. H. Seinfeld and S. N. Pandis, *Atmospheric chemistry and physics: from air pollution to climate change*. John Wiley & Sons, 2016.
- [8] W. N. Carter, *Disaster management: A disaster manager's handbook*. 2008.

- [9] P. Van Oosterom, S. Zlatanova, and E. Fendel, "Geo-information for disaster management," *Springer Science & Business Media.*, 2006.
- [10] K. Shiwaku, A. Sakurai, and R. Shaw, "Disaster resilience of education systems: Experiences from Japan," New York, NY, USA: Springer, 2016.
- [11] Federal Emergency Management Agency. "Hazard Mitigation Plan Requirement." <https://www.fema.gov/hazard-mitigation-plan-requirement>. (accessed May 10, 2020).
- [12] Texas General Land Office. "State of Texas CDBG Mitigation (CDBG-MIT) Action Plan: Building Stronger for a Resilient Future." <https://recovery.texas.gov/files/hud-requirements-reports/mitigation/mitigation-ap.pdf>. (accessed May. 12, 2020).
- [13] Florida Department of Emergency Management. "Enhanced State Hazard Mitigation Plan, State of Florida." https://www.floridadisaster.org/globalassets/dem/mitigation/mitigate-fl--shmp/shmp-2018-full_final_approved.6.11.2018.pdf. (accessed May. 5, 2020).
- [14] D. Liu, L. Pang, and B. Xie, "Typhoon disaster in China: prediction, prevention, and mitigation," *Natural Hazards.*, vol. 49, no. 3, pp. 421-436, 2009.
- [15] M. Hoshihara and T. Ozaki, "Earthquake Early Warning and Tsunami Warning of the Japan Meteorological Agency, and Their Performance in the 2011 off the Pacific Coast of Tohoku Earthquake," *Early warning for geological disasters.*, Springer, pp. 1-28, 2014.

- [16] L. Alfieri, P. Salamon, F. Pappenberger, F. Wetterhall, and J. Thielen, "Operational early warning systems for water-related hazards in Europe," *Environmental Science & Policy.*, vol. 21, pp. 35-49, 2012.
- [17] T. J. Cova, "GIS in emergency management," *Geographical information systems.*, vol. 2, pp. 845-858, 1999.
- [18] B. Tomaszewski, *Geographic information systems (GIS) for disaster management*. London, UK: Routledge, 2014.
- [19] A. Meissner, T. Luckenbach, T. Risse, T. Kirste, and H. Kirchner, "Design challenges for an integrated disaster management communication and information system," presented at The First IEEE Workshop on Disaster Recovery Networks (DIREN 2002), 2002.
- [20] K. Uno and K. Kashiya, "Development of simulation system for the disaster evacuation based on multi-agent model using GIS," *Tsinghua Science and Technology*, vol. 13, no. S1, pp. 348-353, 2008.
- [21] B. Wisner, "A review of the role of education and knowledge in disaster risk reduction," 2006.
- [22] A. Zerger and D. I. Smith, "Impediments to using GIS for real-time disaster decision support," *Computers, environment and urban systems*, vol. 27, no. 2, pp. 123-141, 2003.
- [23] Federal Emergency Management Agency. "Damage Assessment Operations Manual."

- <https://www.fema.gov/media-library-data/1459972926996-a31eb90a2741e86699ef34ce2069663a/PDAManualFinal6.pdf>. (accessed May. 10, 2020).
- [24] American Red Cross. "American Red Cross Damage Assessment." <http://www.resiliencenw.org/2012files/LongTermRecovery/DisasterAssessmentWorkshop.pdf>. (accessed May. 10, 2020)
- [25] K. Bea, "Federal Stafford Act disaster assistance: Presidential declarations, eligible activities, and funding," library of congress Washington DC congressional research service.
- [26] Federal Emergency Management Agency. "National Urban Search & Rescue (US&R) Response System rescue field operations guide." https://www.fema.gov/pdf/emergency/usr/usr_23_20080205_rog.pdf. (accessed May. 10, 2020).
- [27] R. B. Olshansky, L. D. Hopkins, and L. A. Johnson, "Disaster and recovery: Processes compressed in time," *Natural Hazards Review*, vol. 13, no. 3, pp. 173-178, 2012.
- [28] L. Ozdamar, "Planning helicopter logistics in disaster relief," *OR spectrum*, vol. 33, no. 3, pp. 655-672, 2011.
- [29] S. M. Adams and C. J. Friedland, "A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management," presented at 9th International Workshop on Remote Sensing for Disaster Response, 2011.

- [30] Federal Aviation Administration. "Fact Sheet – Federal Aviation Administration (FAA) Forecast Fiscal Years (FY) 2017-2038."
https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22594. (accessed May. 10, 2020)
- [31] B. Haworth and E. Bruce, "A review of volunteered geographic information for disaster management," *Geography Compass*, vol. 9, no. 5, pp. 237-250, 2015.
- [32] F. Ofli, P. Meier, M. Imran, C. Castillo, D. Tuia, N. Rey, J. Briant, P. Millet, F. Reinhard. And M. Parkan, "Combining human computing and machine learning to make sense of big (aerial) data for disaster response," *Big data*, vol. 4, no. 1, pp. 47-59, 2016.
- [33] S. J. Russell and P. Norvig, "Artificial intelligence: a modern approach," *Pearson Education Limited*, 2016.
- [34] E. Alpaydin, *Introduction to machine learning*. Cambridge, MA, USA: MIT press, 2020.
- [35] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in Proc. 3rd International Conference on Computing for Sustainable Global Development., New Delhi, India: IEEE, 2016.
- [36] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *Computing Research Repository*, 2012.
- [37] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," 2018, arXiv:1805.00909.

- [38] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 251-260, 2013.
- [39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proc. The thirteenth international conference on artificial intelligence and statistics., Sardinia, Italy: AISTATS, 2010.
- [40] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, arXiv:1506.00019.
- [41] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352-2449, 2017.
- [42] J. M. Murre, *Learning and categorization in modular neural networks*. Psychology Press, 2014.
- [43] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [44] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128-135, 2010.
- [45] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279-292, 1992.
- [46] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in

- Proc. The conference on empirical methods in natural language processing.,
Scotland, UK: Association for Computational Linguistics, 2012.
- [47] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in Proc. Interspeech., Lyon, France: ISCA, 2013.
- [48] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371-3408, 2010.
- [49] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in Proc. IEEE Conference on Computer Vision and Pattern Recognition., Honolulu, Hawaii: IEEE, 2017.
- [50] D. Amodei et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in Proc. International conference on machine learning., New York, NY, USA: ICML, 2016.
- [51] C. Blehm, S. Vishnu, A. Khattak, S. Mitra, and R. W. Yee, "Computer vision syndrome: a review," *Survey of ophthalmology*, vol. 50, no. 3, pp. 253-262, 2005.
- [52] E. T. Lin and E. J. Delp, "A review of data hiding in digital images," in Proc. PICS., 1999.
- [53] C. Pohl and J. L. Van Genderen, "Review article multisensor image fusion in remote sensing: concepts, methods and applications," *International journal of remote sensing*, vol. 19, no. 5, pp. 823-854, 1998.

- [54] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in Proc. 2012 IEEE Conference on Computer Vision and Pattern Recognition., Providence, RI, USA: IEEE, 2012.
- [55] M. Nixon and A. Aguado, *Feature extraction and image processing for computer vision*. Academic Press, 2019.
- [56] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.
- [57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc. Advances in neural information processing systems., Stateline, NV, USA: NIPS, 2012.
- [58] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in Proc. 2008 IEEE conference on computer vision and pattern recognition., Anchorage, AK, USA: IEEE, 2008.
- [59] V. Podlozhnyuk, "FFT-based 2D convolution," *NVIDIA white paper*, vol. 32, 2007.
- [60] C. Guo, Y.-l. Liu, and X. Jiao, "Study on the influence of variable stride scale change on image recognition in CNN," *Multimedia Tools and Applications*, vol. 78, no. 21, pp. 30027-30037, 2019.
- [61] M. Dwarampudi and N. Reddy, "Effects of padding on LSTMs and CNNs," 2019, arXiv:1903.07288.

- [62] G. Toliás, R. Sivic, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," 2015, arXiv:1511.05879.
- [63] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193-202, 1980.
- [64] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.
- [65] M. Bojarski et al., "End to end learning for self-driving cars," 2016, arXiv:1604.07316.
- [66] Tesla. "Tesla Auto Pilot Introduction." <https://www.tesla.com/autopilot>. (accessed May. 10, 2020).
- [67] M. Yang, S. Wang, J. Bakita, T. Vu, D. Smith, J. Anderson, and J. Frahm, "Re-thinking CNN Frameworks for Time-Sensitive Autonomous-Driving Applications: Addressing an Industrial Challenge," in Proc. 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Montreal, Canada: IEEE, 2019.
- [68] J. Li et al., "Facial expression recognition with faster R-CNN," *Procedia Computer Science*, vol. 107, pp. 135-140, 2017.
- [69] G. Ozbulak, Y. Aytar, and H. K. Ekenel, "How transferable are CNN-based features for age and gender classification?," in Proc. 2016 International

- Conference of the Biometrics Special Interest Group (BIOSIG)., Darmstadt, Germany: IEEE, 2016.
- [70] L.-H. Wang, X.-P. Zhao, J.-X. Wu, Y.-Y. Xie, and Y.-H. Zhang, "Motor fault diagnosis based on short-time Fourier transform and convolutional neural network," *Chinese Journal of Mechanical Engineering*, vol. 30, no. 6, pp. 1357-1368, 2017.
- [71] D. Palaz and R. Collobert, "Analysis of cnn-based speech recognition system using raw speech as input," Idiap, 2015.
- [72] I. Wallach, M. Dzamba, and A. Heifets, "AtomNet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery," 2015, arXiv:1510.02855.
- [73] Z. Zhou, X. Li, and R. N. Zare, "Optimizing chemical reactions with deep reinforcement learning," *ACS central science*, vol. 3, no. 12, pp. 1337-1344, 2017.
- [74] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.
- [75] I. Goodfellow et al., "Generative adversarial nets," in Proc. Advances in neural information processing systems., Montreal, Canada: IEEE, 2014.
- [76] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "Ganimation: Anatomically-aware facial animation from a single image," in Proc. The European Conference on Computer Vision (ECCV)., Munich, Germany: CVF, 2018.

- [77] A. Ghosh, B. Bhattacharya, and S. B. R. Chowdhury, "Sad-gan: Synthetic autonomous driving using generative adversarial networks," 2016, arXiv:1611.08788.
- [78] E. Dekker, "Early explorations of the southern celestial sky," *Annals of science*, vol. 44, no. 5, pp. 439-470, 1987.
- [79] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in Proc. The IEEE conference on computer vision and pattern recognition., Columbus, OH , USA: IEEE, 2017.
- [80] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in Proc. The IEEE international conference on computer vision., Venice, Italy: IEEE, 2017.
- [81] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in Proc. The IEEE international conference on computer vision., Venice, Italy: IEEE, 2017.
- [82] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in Proc. IEEE conference on computer vision and pattern recognition. , Columbus, OH , USA: IEEE, 2017.
- [83] L. Kantha, "Time to replace the Saffir - Simpson hurricane scale?," *Eos, Transactions American Geophysical Union*, vol. 87, no. 1, pp. 3-6, 2006.
- [84] National Aeronautics and Space Administration. "What Are Hurricanes?" <https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what-are-hurricanes-k4.html>. (accessed May. 10, 2020).

- [85] A. Benfield. “ Weather, climate and catastrophe insight: 2017 annual report.” <http://thoughtleadership.aonbenfield.com/Documents/20180124-ab-if-annual-report-weather-climate-2017.pdf>. (accessed May. 11, 2020).
- [86] National Center for Environmental Information. “Assessing the U.S. Climate in 2018” <https://www.ncei.noaa.gov/news/national-climate-201812>. (accessed May. 11, 2020).
- [87] National Hurricane Center. “Tropical Cyclone Report Hurricane Michael.” https://www.nhc.noaa.gov/data/tcr/AL142018_Michael.pdf. (accessed May. 10, 2020).
- [88] National Hurricane Center. “Tropical Cyclone Report Hurricane Lorenzo” https://www.nhc.noaa.gov/data/tcr/AL132019_Lorenzo.pdf. Accessed (accessed May. 10, 2020).
- [89] K. E. Trenberth, L. Cheng, P. Jacobs, Y. Zhang, and J. Fasullo, "Hurricane Harvey links to ocean heat content and climate change adaptation," *Earth's Future*, vol. 6, no. 5, pp. 730-744, 2018.
- [90] W. Fan and A. Bifet, "Mining big data: current status, and forecast to the future," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1-5, 2013.
- [91] S. Choi and B. Bae, "The real-time monitoring system of social big data for disaster management," in *Proc. Computer science and its applications.*, Cebu, Philippines: Springer, 2015.
- [92] T. Papadopoulos, A. Gunasekaran, R. Dubey, N. Altay, S. J. Childe, and S. Fosso-Wamba, "The role of Big Data in explaining disaster resilience in supply

- chains for sustainability," *Journal of Cleaner Production*, vol. 142, pp. 1108-1118, 2017.
- [93] V. Chang, "Towards a Big Data system disaster recovery in a Private Cloud," *Ad Hoc Networks*, vol. 35, pp. 65-82, 2015.
- [94] M. Craglia, F. Ostermann, and L. Spinsanti, "Digital Earth from vision to practice: making sense of citizen-generated content," *International Journal of Digital Earth*, vol. 5, no. 5, pp. 398-416, 2012.
- [95] J. Kim and M. Hastak, "Social network analysis: Characteristics of online social networks after a disaster," *International Journal of Information Management*, vol. 38, no. 1, pp. 86-96, 2018.
- [96] Y. Faxi, L. Rui, and M. Ouejdane, "An information system for real-time critical infrastructure damage assessment based on crowdsourcing method: a case study in Fort McMurray," in Proc. International Conference on Sustainable Infrastructure, New York City, NY, USA: ASCE, 2017.
- [97] M. F. Goodchild and J. A. Glennon, "Crowdsourcing geographic information for disaster response: a research frontier," *International Journal of Digital Earth*, vol. 3, no. 3, pp. 231-241, 2010.
- [98] L. Zou, N. Lam. S. Shams. H. Cai. M. Meyer, S. Yang, K. Lee, S. Park, M. Reams, "Social and geographical disparities in Twitter use during Hurricane Harvey," *International Journal of Digital Earth*, pp. 1-19, 2018.

- [99] C. A. B. Baker, S. Ramchurn, W. T. Teacy, and N. R. Jennings, "Planning search and rescue missions for UAV teams," in Proc. Twenty-second European Conference on Artificial Intelligence., Netherlands: IOS Press, 2016.
- [100] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs)," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281-330, 2015.
- [101] M. Radovic, O. Adarkwa, and Q. Wang, "Object recognition in aerial images using convolutional neural networks," *Journal of Imaging*, vol. 3, no. 2, p. 21, 2017.
- [102] S. Han, W. Shen, and Z. Liu, "Deep Drone: object detection and tracking for smart drones on embedded system," Stanford University, 2012.
- [103] P. Narayanan, C. Borel-Donohue, H. Lee, H. Kwon, and R. Rao, "A real-time object detection framework for aerial imagery using deep neural networks and synthetic training images," in Proc. Signal Processing, Sensor/Information Fusion, and Target Recognition XXVII., Orlando, FL, USA: International Society for Optics and Photonics, 2018.
- [104] E. Guirado, S. Tabik, M. L. Rivas, D. Alcaraz-Segura, and F. Herrera, "Automatic whale counting in satellite images with deep learning," 2018.
- [105] M. J. Friedel, M. Buscema, L. E. Vicente, F. Iwashita, and A. Koga-Vicente, "Mapping fractional landscape soils and vegetation components from Hyperion satellite imagery using an unsupervised machine-learning workflow," *International journal of digital earth*, vol. 11, no. 7, pp. 670-690, 2018.

- [106] J. Suh and Y. Choi, "Mapping hazardous mining-induced sinkhole subsidence using unmanned aerial vehicle (drone) photogrammetry," *Environmental Earth Sciences*, vol. 76, no. 4, p. 144, 2017.
- [107] D. Ventura, M. Bruno, G. J. Lasinio, A. Belluscio, and G. Ardizzone, "A low-cost drone based application for identifying and mapping of coastal fish nursery grounds," *Estuarine, Coastal and Shelf Science*, vol. 171, pp. 85-98, 2016.
- [108] A. M. Cunliffe, R. E. Brazier, and K. Anderson, "Ultra-fine grain landscape-scale quantification of dryland vegetation structure with drone-acquired structure-from-motion photogrammetry," *Remote Sensing of Environment*, vol. 183, pp. 129-143, 2016.
- [109] M. Rahnemoonfar, R. Murphy, M. V. Miquel, D. Dobbs, and A. Adams, "Flooded area detection from UAV images based on densely connected recurrent neural networks," in Proc. IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium., Valencia, Spain: IEEE, 2018.
- [110] S. Ghaffarian and N. Kerle, "towards post-disaster debris identification for precise damage and recovery assessments from UAV and satellite images," *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2019.
- [111] X. Li, D. Caragea, H. Zhang, and M. Imran, "Localizing and Quantifying Damage in Social Media Images," in Proc. 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)., Barcelona, Spain: IEEE, 2018.

- [112] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, arXiv:1409.1556.
- [113] T. Liu, A. Abd-Elrahman, J. Morton, and V. L. Wilhelm, "Comparing fully convolutional networks, random forest, support vector machine, and patch-based deep convolutional neural networks for object-based wetland mapping using images from small unmanned aircraft system," *GIScience & remote sensing*, vol. 55, no. 2, pp. 243-264, 2018.
- [114] M. Chen and J. Li, "Deep convolutional neural network application on rooftop detection for aerial image," 2019, arXiv:1910.13509.
- [115] Y. Rao, W. Liu, J. Pu, J. Deng, and Q. Wang, "Roads Detection of Aerial Image with FCN-CRF Model," in Proc. 2018 IEEE Visual Communications and Image Processing (VCIP)., Taichung, Taiwan, 2018.
- [116] S. Ghaffarian, N. Kerle, E. Pasolli, and J. Jokar Arsanjani, "Post-disaster building database updating using automated deep learning: An integration of pre-disaster OpenStreetMap and multi-temporal satellite data," *Remote sensing*, vol. 11, no. 20, p. 2427, 2019.
- [117] OpenStreetMap. "Open street map" <https://www.openstreetmap.org>. (accessed May. 10, 2020).
- [118] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc. Advances in neural information processing systems., Lake Tahoe, NV, USA: NIPS, 2012.

- [119] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proc. IEEE conference on computer vision and pattern recognition., Boston, MA, USA: IEEE, 2015.
- [120] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE conference on computer vision and pattern recognition., Las Vegas, NV, USA: IEEE 2016.
- [121] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE conference on computer vision and pattern recognition, Columbus, OH , USA: IEEE, 2014.
- [122] R. Girshick, "Fast r-cnn," in Proc. IEEE international conference on computer vision, Boston, MA, USA: IEEE, 2015.
- [123] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Proc. Advances in neural information processing systems, Vancouver, Canada: NIPS, 2015.
- [124] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303-338, 2010.
- [125] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in Proc. Advances in neural information processing systems., Barcelona, Spain: NIPS, 2016.

- [126] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE conference on computer vision and pattern recognition., Las Vegas, NV, USA: IEEE, 2016.
- [127] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. Berg, "SSD: Single shot multibox detector," in Proc. European conference on computer vision., Amsterdam, Netherlands: Springer, 2016.
- [128] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481-2495, 2017.
- [129] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211-252, 2015.
- [130] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98-136, 2015.
- [131] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, L. Zitnick, "Microsoft coco: Common objects in context," in Proc. European conference on computer vision., Zurich, Switzerland: Springer, 2014.
- [132] G. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, X. Lu, "AID: A benchmark data set for performance evaluation of aerial scene classification,"

- IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3965-3981, 2017.
- [133] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 197-209, 2018.
- [134] M. Bonetto, P. Korshunov, G. Ramponi, and T. Ebrahimi, "Privacy in mini-drone based video surveillance," in Proc. 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)., Ljubljana, Slovenia: IEEE, 2015.
- [135] R. Gupta, R. Hosfelt, S. Sajeev, N. Patel, B. Goodman, J. Doshi, E. Heim, H. Choset, and M. Gaston, "xBD: A Dataset for Assessing Building Damage from Satellite Imagery," 2019, arXiv:1911.09296.
- [136] National Coordination Office for Space-Based Positioning and Timing. "Official U.S. government information about the Global Positioning System (GPS) and related topics" <https://www.gps.gov/systems/gps/performance/accuracy/>. (accessed May. 10, 2020).
- [137] D. Magree and E. N. Johnson, "Combined laser and vision-aided inertial navigation for an indoor unmanned aerial vehicle," in Proc. 2014 American Control Conference., Portland, OR: IEEE, 2014.
- [138] G. Balamurugan, J. Valarmathi, and V. Naidu, "Survey on UAV navigation in GPS denied environments," in Proc. 2016 International Conference on Signal

Processing, Communication, Power and Embedded System (SCOPES)., ankt Goar, Germany: IEEE, 2016.

- [139] C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for unmanned aerial vehicle using ARM-based stereo vision pre-processing system," in Proc. 2015 International Conference on Unmanned Aircraft Systems (ICUAS)., Denver, CO, USA: IEEE, 2015.
- [140] J. M. Barrett, M. Gennert, W. Michalson, M. Audi, J. Center, J. Kirk, and B. Welch, "Development of a low-cost, self-contained, combined vision and inertial navigation system," in Proc. 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)., Woburn, MA, USA: IEEE, 2013.
- [141] A. Chambers, S. Scherer, L. Yoder, S. Jain, S. Nuske, and S. Singh, "Robust multi-sensor fusion for micro aerial vehicle navigation in GPS-degraded/denied environments," in Proc. 2014 American Control Conference., Portland, OR, USA: IEEE, 2014.
- [142] D.-Y. Gu, C.-F. Zhu, J. Guo, S.-X. Li, and H.-X. Chang, "Vision-aided UAV navigation using GIS data," in Proc. 2010 IEEE International Conference on Vehicular Electronics and Safety., QingDao, China: IEEE, 2010.
- [143] R. Mebarki, J. Cacace, and V. Lippiello, "Velocity estimation of an UAV using visual and IMU data in a GPS-denied environment," in Proc. 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)., Linköping, Sweden: IEEE, 2013.

- [144] A. Soloviev, "Tight coupling of GPS, laser scanner, and inertial measurements for navigation in urban environments," in Proc. 2008 IEEE/ION Position, Location and Navigation Symposium, Monterey, CA, USA: IEEE, 2008.
- [145] C. Wang, T. Wang, J. Liang, Y. Chen, Y. Zhang, and C. Wang, "Monocular visual SLAM for small UAVs in GPS-denied environments," in Proc. 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou, China: IEEE, 2012.
- [146] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli, "Vision based gps-denied object tracking and following for unmanned aerial vehicles," in Proc. 2013 IEEE international symposium on safety, security, and rescue robotics (SSRR), Linköping, Sweden IEEE, 2013.
- [147] S. Rajeev, Q. Wan, K. Yau, K. Panetta, and S. S. Aghaian, "Augmented reality-based vision-aid indoor navigation system in GPS denied environment," in Proc. Mobile Multimedia/Image Processing, Security, and Applications 2019., Baltimore, MD, USA: International Society for Optics and Photonics, 2019.
- [148] A. Bachrach, S. Prentice, R. He, P. Henry, A. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320-1343, 2012.
- [149] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, and L. Kneip, "Vision-controlled micro flying robots: from system design to autonomous navigation

- and mapping in GPS-denied environments," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 26-40, 2014.
- [150] S. R. Singh, M. R. Eghdami, and S. Singh, "The concept of social vulnerability: A review from disasters perspectives," *International Journal of Interdisciplinary and Multidisciplinary Studies*, vol. 1, no. 6, pp. 71-82, 2014.
- [151] I. Burton, *The environment as hazard*. Guilford press, 1993.
- [152] P. Blaikie, T. Cannon, I. Davis, and B. Wisner, *At risk: natural hazards, people's vulnerability and disasters*. Routledge, 2005.
- [153] S. L. Cutter, B. J. Boruff, and W. L. Shirley, "Social vulnerability to environmental hazards," *Social science quarterly*, vol. 84, no. 2, pp. 242-261, 2003.
- [154] C. Finch, C. T. Emrich, and S. L. Cutter, "Disaster disparities and differential recovery in New Orleans," *Population and Environment*, vol. 31, no. 4, pp. 179-202, 2010.
- [155] M. A. Meyer, "Social capital and collective efficacy for disaster resilience: Connecting individuals with communities and vulnerability with resilience in hurricane-prone communities in Florida," Ph.D. dissertation, Colorado State University, 2013.
- [156] E. Enarson and M. Fordham, "Lines that divide, ties that bind: Race, class, and gender in women's flood recovery in the US and UK," *Australian Journal of Emergency Management*, vol. 15, no. 4, p. 43, 2000.

- [157] B. H. Morrow and E. Enarson, "Hurricane Andrew through women's eyes," *International Journal of Mass Emergencies and Disasters*, vol. 14, no. 1, pp. 5-22, 1996.
- [158] W. G. Peacock, S. Van Zandt, Y. Zhang, and W. E. Highfield, "Inequities in long-term housing recovery after disasters," *Journal of the American Planning Association*, vol. 80, no. 4, pp. 356-371, 2014.
- [159] A. Fothergill, E. G. Maestas, and J. D. Darlington, "Race, ethnicity and disasters in the United States: A review of the literature," *Disasters*, vol. 23, no. 2, pp. 156-173, 1999.
- [160] S. Zahran, S. D. Brody, W. G. Peacock, A. Vedlitz, and H. Grover, "Social vulnerability and the natural and built environment: a model of flood casualties in Texas," *Disasters*, vol. 32, no. 4, pp. 537-560, 2008.
- [161] C. Burton and S. L. Cutter, "Levee failures and social vulnerability in the Sacramento-San Joaquin Delta area, California," *Natural Hazards Review*, vol. 9, no. 3, pp. 136-149, 2008.
- [162] A. Dwyer, C. Zoppou, O. Nielsen, S. Day, and S. Roberts, "Quantifying social vulnerability: a methodology for identifying those at risk to natural hazards," 2004.
- [163] M. K. Lindell and R. W. Perry, *Communicating environmental risk in multiethnic communities*. Sage Publications, 2003.

- [164] B. H. Morrow and W. G. Peacock, "Disasters and social change: Hurricane Andrew and the reshaping of Miami," *Hurricane Andrew: Ethnicity, gender and the sociology of disasters*, pp. 226-242, 1997.
- [165] W. G. Peacock, "Hurricane mitigation status and factors influencing mitigation status among Florida's single-family homeowners," *Natural Hazards Review*, vol. 4, no. 3, pp. 149-158, 2003.
- [166] C. Rojahn, L. Johnson, V. Cedillos, T. O'Rourke, T. P. McAllister, and S. L. McCabe, "Increasing Community Resilience Through Improved Lifeline Infrastructure Performance," 2019.
- [167] D. S. Thomas, B. D. Phillips, W. E. Lovekamp, and A. Fothergill, *Social vulnerability to disasters*. CRC Press, 2013.
- [168] S. Van Zandt, W. G. Peacock, D. W. Henry, H. Grover, W. E. Highfield, and S. D. Brody, "Mapping social vulnerability to enhance housing and neighborhood resilience," *Housing Policy Debate*, vol. 22, no. 1, pp. 29-55, 2012.
- [169] W. N. Adger, "Vulnerability," *Global environmental change*, vol. 16, no. 3, pp. 268-281, 2006.
- [170] P. Berke, J. Cooper, D. Salvesen, D. Spurlock, and C. Rausch, "Building capacity for disaster resiliency in six disadvantaged communities," *Sustainability*, vol. 3, no. 1, pp. 1-20, 2011.
- [171] L. G. Luther, S. Resources, and I. Division, "Disaster debris removal after Hurricane Katrina: Status and associated issues," Congressional Research Service, 2006.

- [172] N. Hirayama, T. Shimaoka, T. Fujiwara, T. Okayama, and Y. Kawata, "Establishment of disaster debris management based on quantitative estimation using natural hazard maps," *Waste Manag Environ*, vol. 140, pp. 167-178, 2010.
- [173] Y. Pi, N. D. Nath, and A. H. Behzadan, "Convolutional neural networks for object detection in aerial imagery for disaster response and recovery," *Advanced Engineering Informatics*, vol. 43, p. 101009, 2020.
- [174] DarkLabel. "Image Labeling and Annotation Tool."
<https://darkpgmr.tistory.com/16>. (accessed May. 10, 2020).
- [175] Federal Aviation Administration. "Recreational Flyers & Modeler Community-Based Organizations." https://www.faa.gov/uas/recreational_fliers/. (accessed May. 10, 2020).
- [176] Federal Aviation Administration. "Low Flying Aircraft Complaints"
https://www.faa.gov/about/office_org/field_offices/fsdo/atl/local_more/media/nl_owfly.pdf. (accessed May. 10, 2020).
- [177] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220-239, 2017.
- [178] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in Proc. IEEE conference on computer vision and pattern recognition., Columbus, OH, USA: IEEE, 2014.

- [179] Texas A&M High Performance Research Computing. "Terra: A Lenovo x86 HPC Cluster" <https://hprc.tamu.edu/wiki/Terra:Intro>. (accessed May. 10, 2020).
- [180] LabelBox. "Labelbox: The leading training data solution" <https://labelbox.com/>. (accessed July. 20, 2019)
- [181] M. Huh, P. Agrawal, and A. A. Efros, "What makes ImageNet good for transfer learning?," 2016, arXiv:1608.08614.
- [182] S. Deepak and P. Ameer, "Brain tumor classification using deep CNN features via transfer learning," *Computers in biology and medicine*, vol. 111, p. 103345, 2019.
- [183] Y. Li and R. Nevatia, "Key object driven multi-category object recognition, localization and tracking using spatio-temporal context," in *Proc. European Conference on Computer Vision.*, Marseille, France: Springer, 2008.
- [184] F. Pérez-Hernández, S. Tabik, A. Lamas, R. Olmos, H. Fujita, and F. Herrera, "Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance," *Knowledge-Based Systems*, p. 105590, 2020.
- [185] D. Augusto Borges Oliveira and M. Palhares Viana, "Fast CNN-based document layout analysis," in *Proc. IEEE International Conference on Computer Vision Workshops.*, Salt Lake City, UT, USA: IEEE, 2017.
- [186] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *Proc. International symposium on visual computing*, Las Vegas, NV, USA: Springer, 2016.

- [187] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, W. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in Proc. IEEE conference on computer vision and pattern recognition., Columbus, OH, USA: IEEE, 2014.
- [188] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Proc. International Conference on Medical image computing and computer-assisted intervention., Munich, Germany: Springer, 2015.
- [189] K. Yang, K. Wang, L. Bergasa, E. Romera, W. Hu, D. Sun, J. Sun, R. Cheng, T. Chen, and E. Lopez, "Unifying terrain awareness for the visually impaired through real-time semantic segmentation," *Sensors*, vol. 18, no. 5, p. 1506, 2018.
- [190] R. Sukthankar, R. G. Stockton, and M. D. Mullin, "Smarter presentations: Exploiting homography in camera-projector systems," in Proc. Eighth IEEE International Conference on Computer Vision., Vancouver, Canada: IEEE, 2001.
- [191] C. Loop and Z. Zhang, "Computing rectifying homographies for stereo vision," in Proc. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1999., Fort Collins, CO, USA: IEEE, 1999.
- [192] Z. Zhang and A. R. Hanson, "3D reconstruction based on homography mapping," Proc. ARPA96, 1996.
- [193] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in Proc. IEEE conference on computer vision and pattern recognition, 2013, Portland, OR, USA: IEEE, 2013.

- [194] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.
- [195] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.
- [196] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. European conference on computer vision.*, Marseille, France: Springer, 2008.
- [197] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583-596, 2014.
- [198] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA: IEEE, 2010.
- [199] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras," in *Proc. IEEE International Conference on Computer Vision*, 2017., Venice, Italy: IEEE, 2017.
- [200] T. Bagautdinov, A. Alahi, F. Fleuret, P. Fua, and S. Savarese, "Social scene understanding: End-to-end multi-person action localization and collective activity recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition.*, Honolulu, HI, USA: IEEE, 2017.

- [201] Y. Fan, X. Lu, D. Li, and Y. Liu, "Video-based emotion recognition using CNN-RNN and C3D hybrid networks," in Proc. 18th ACM International Conference on Multimodal Interaction., Tokyo Japan: ACM, 2016.
- [202] Y. Bar-Shalom, P. K. Willett, and X. Tian, Tracking and data fusion. YBS publishing Storrs, CT, USA:, 2011.
- [203] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in Proc. International Conference on Image Processing (ICIP)., Phoenix, AZ, USA: IEEE, 2016.
- [204] H. W. Kuhn, "The Hungarian method for the assignment problem," Naval research logistics quarterly, vol. 2, no. 1 - 2, pp. 83-97, 1955.
- [205] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in Proc. IEEE conference on computer vision and pattern recognition., Boston, MA, USA: IEEE, 2015.
- [206] Microsoft. "USBuildingFootprints"
<https://github.com/microsoft/USBuildingFootprints>. (accessed May. 10, 2020).
- [207] H. A. Tarling, "Comparative Analysis of Social Vulnerability Indices: CDC's SVI and SoVI®," 2017.
- [208] United States Census Bureau. "Small Area Income and Poverty Estimates (SAIPE)" https://www.census.gov/data-tools/demo/saipe/#/?map_geoSelector=aa_c. (accessed May. 10, 2020).

- [209] United States Census Bureau. "Interactive Maps"
<https://www.census.gov/programs-surveys/geography/data/interactive-maps.html>.
(accessed May. 10, 2020).
- [210] J. O. Berger, "Statistical decision theory and Bayesian analysis." Springer
Science & Business Media, 2013.
- [211] L. Chen, S. Wang, W. Fan, J. Sun, and S. Naoi, "Beyond human recognition: A
CNN-based framework for handwritten character recognition," in Proc. Asian
Conference on Pattern Recognition (ACPR)., Beijing, China: IEEE, 2015.
- [212] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer:
Benchmarking machine learning algorithms for traffic sign recognition," Neural
networks, vol. 32, pp. 323-332, 2012.
- [213] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "Deepface: Closing the gap
to human-level performance in face verification," in Proc. IEEE conference on computer
vision and pattern recognition, Columbus, OH, USA: IEEE, 2014.

APPENDIX A. YOLO MODEL PERFORMANCE

This Appendix reports the performance of all trained YOLO v2 models (model1-8) on all available test data from Volan2018 dataset. It is worth to note, while testing on unseen dataset (Volan007 and 008), only the existing classes are considered to calculate the precision and recall values.

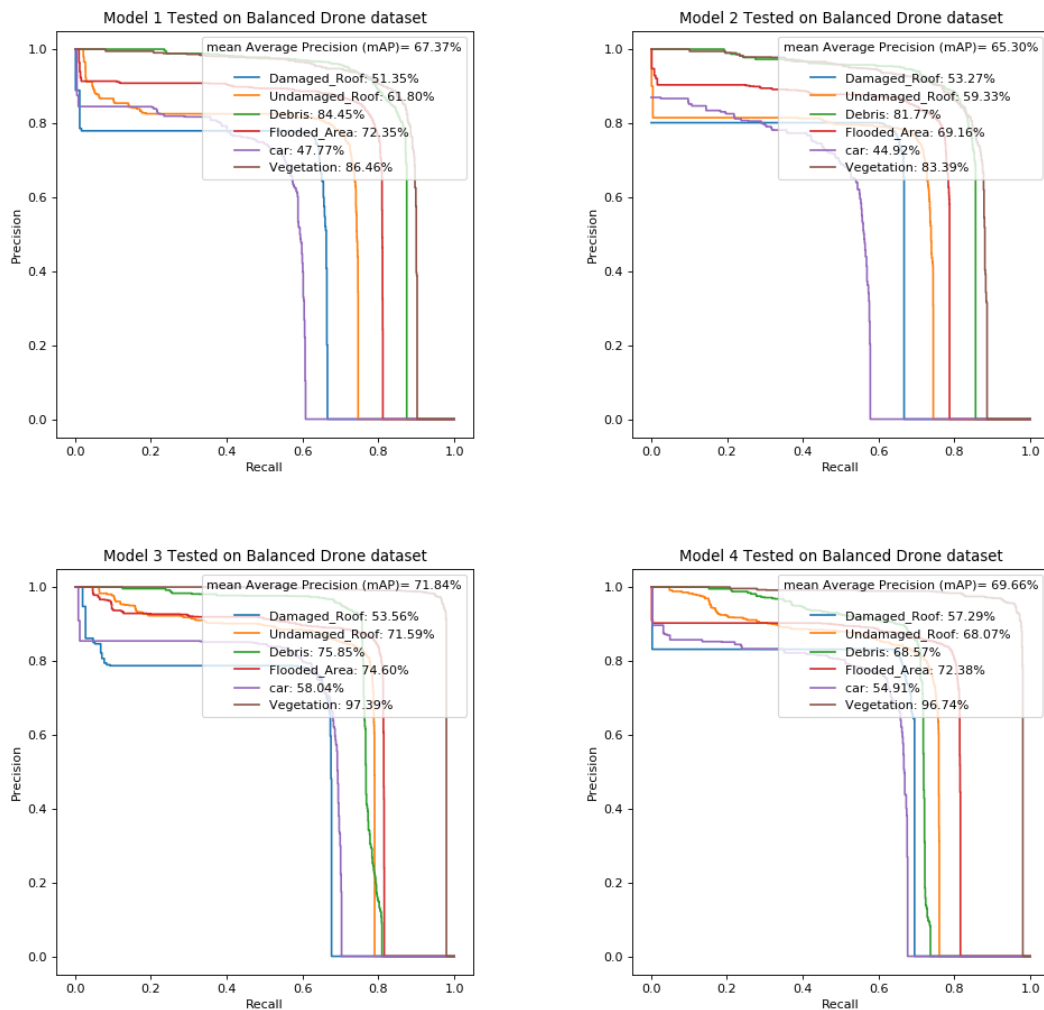


Figure 58. Model 1-8 testing results on UB, UD, HB, HU, Volan007 and 008.

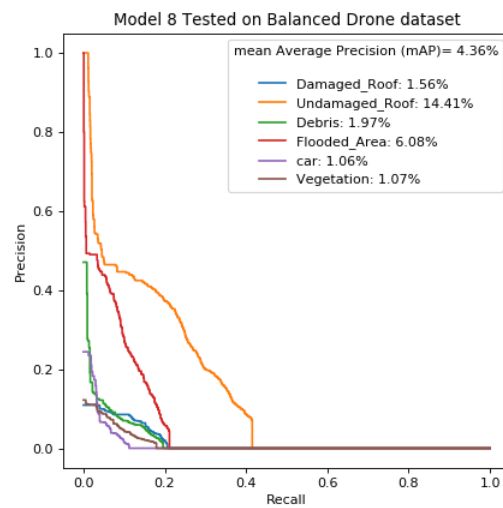
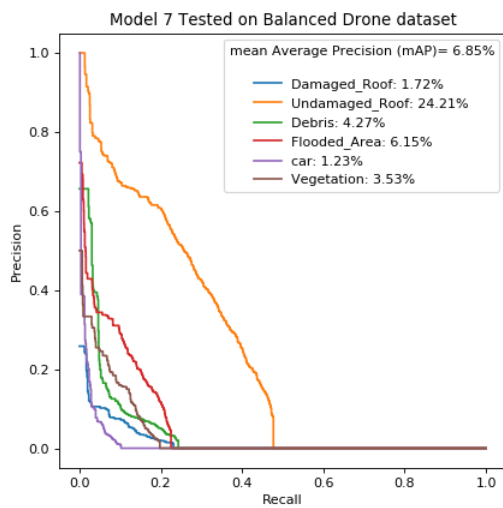
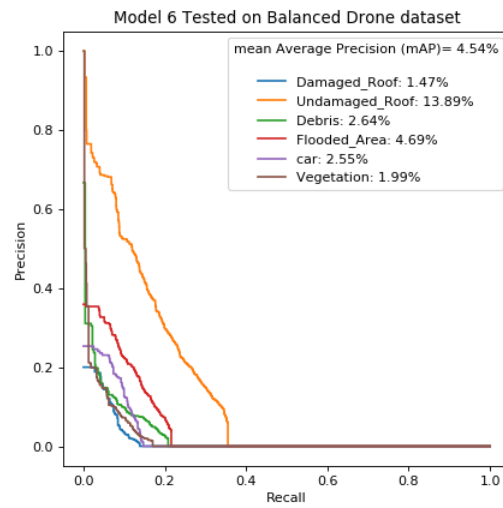
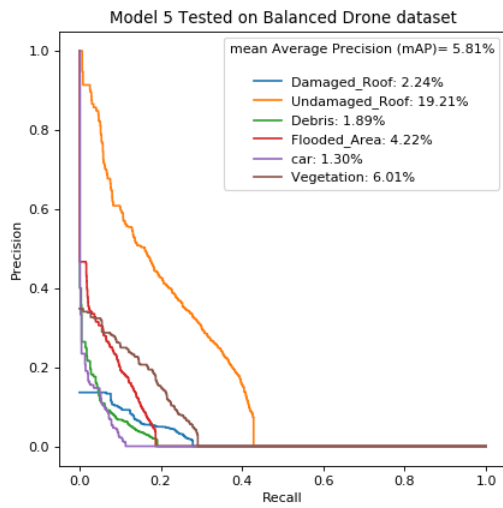


Figure 58. Continued.

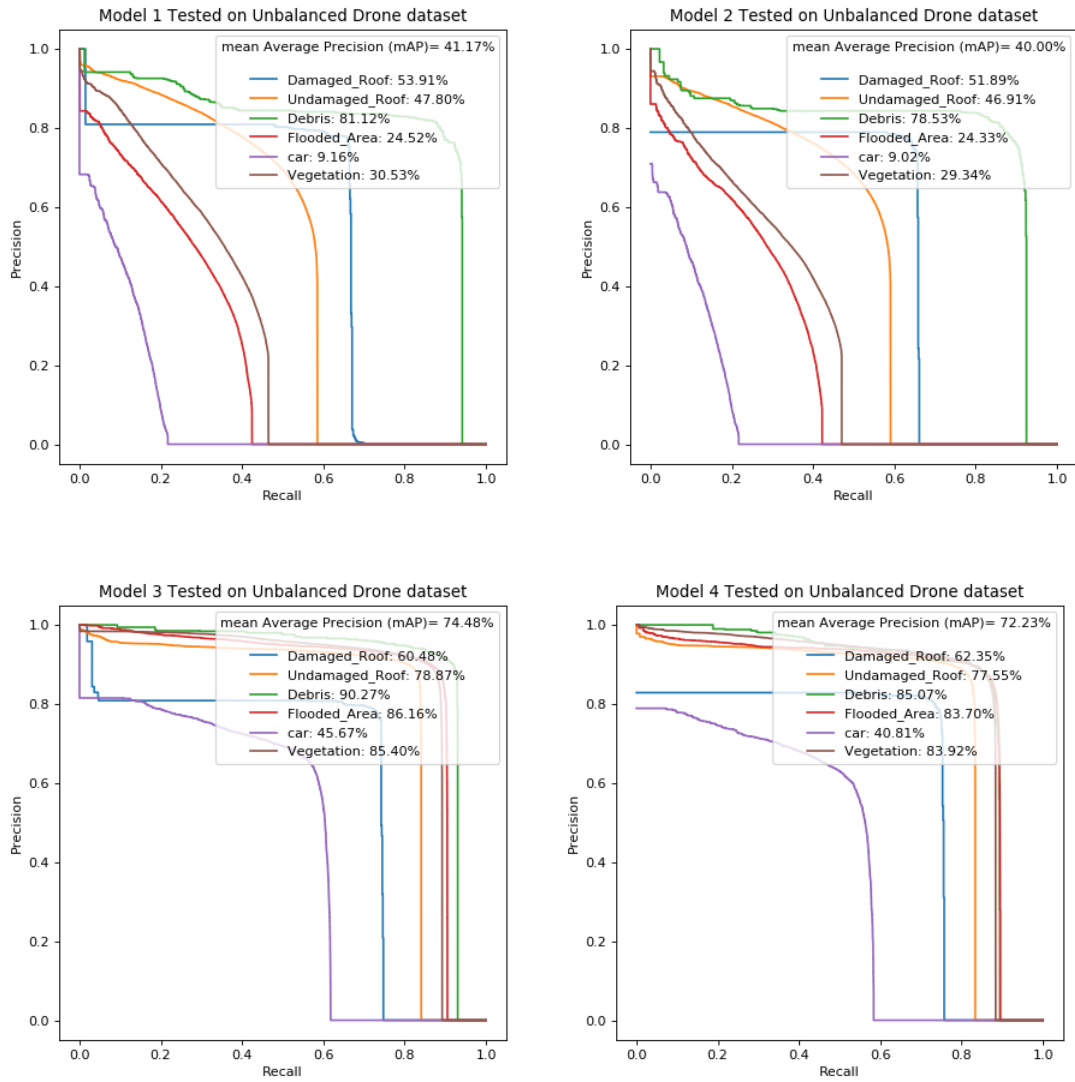


Figure 58. Continued.

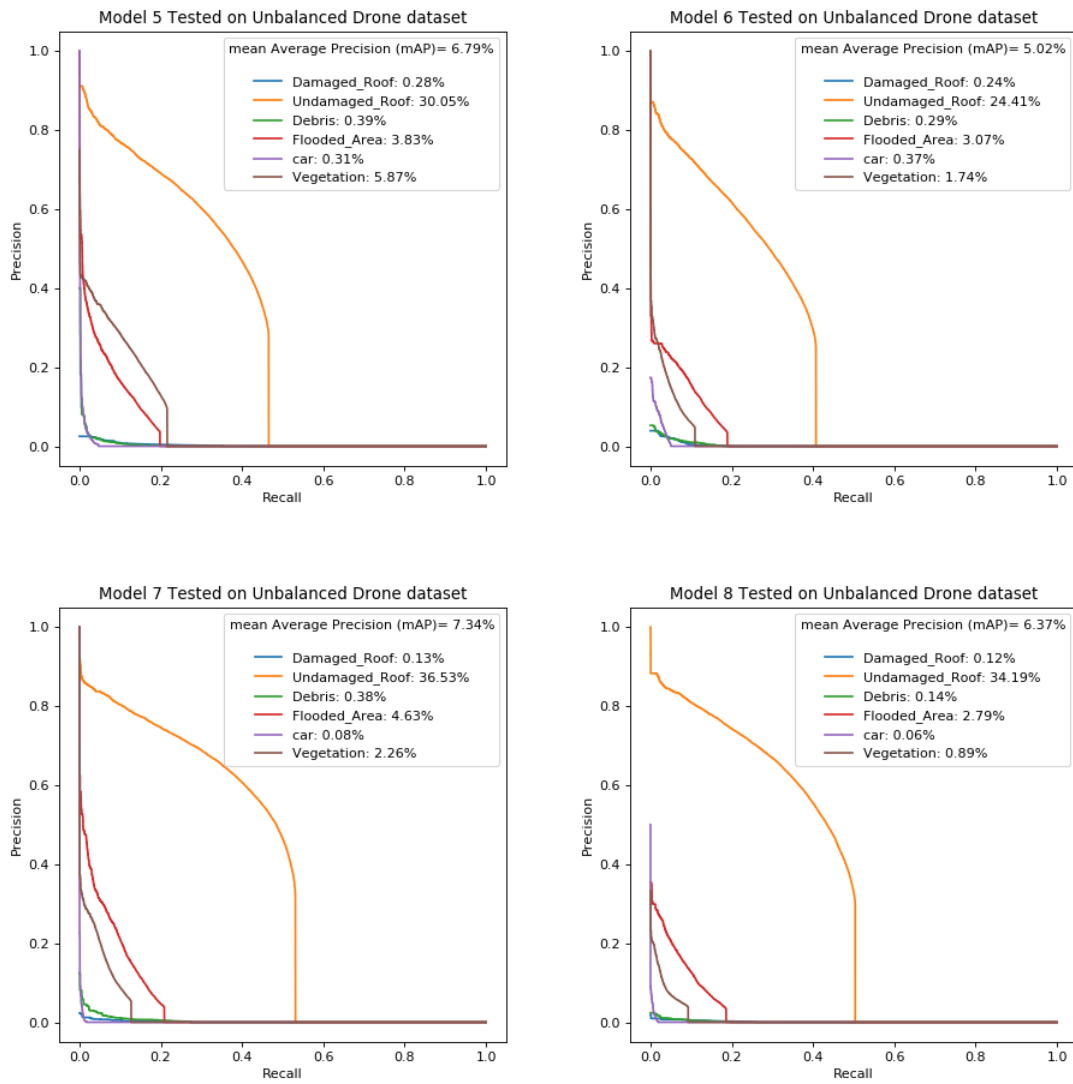


Figure 58. Continued.

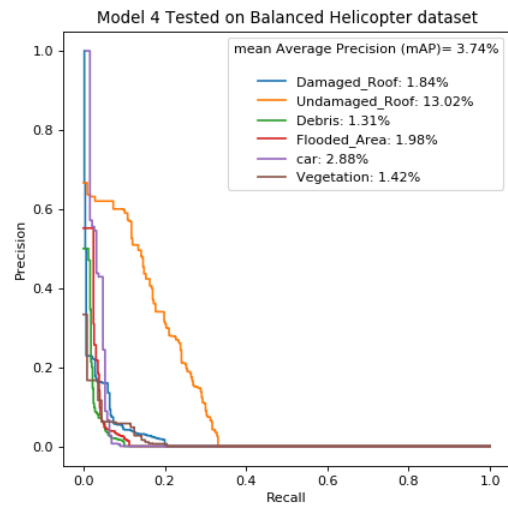
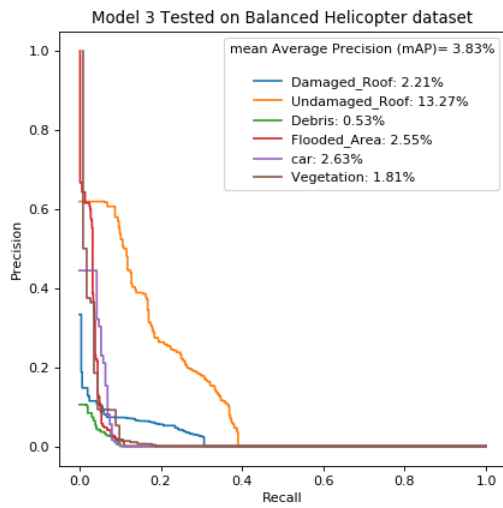
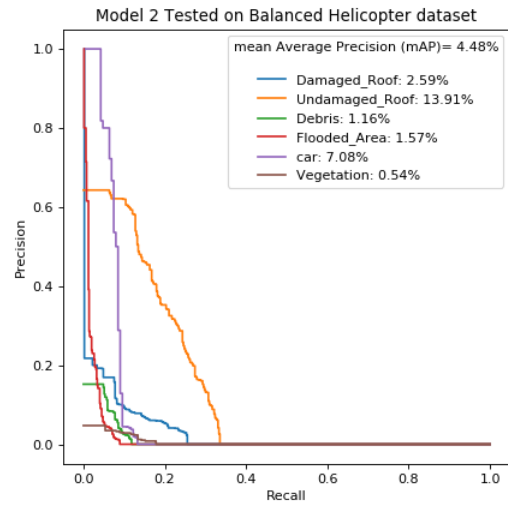
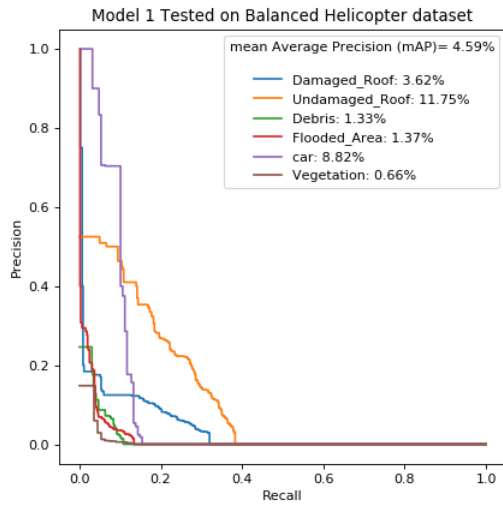


Figure 58. Continued.

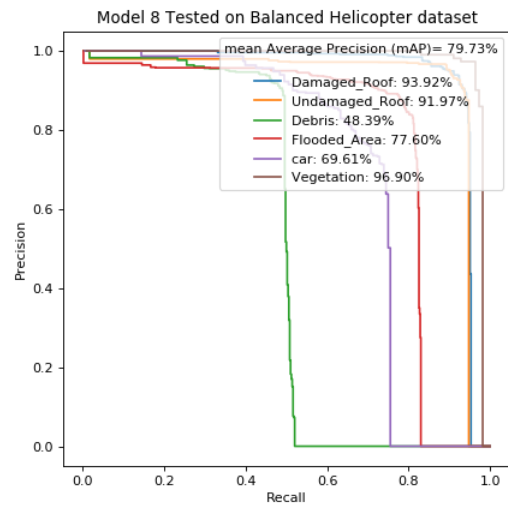
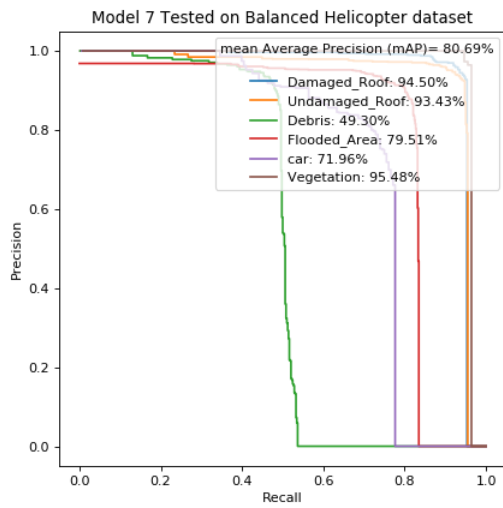
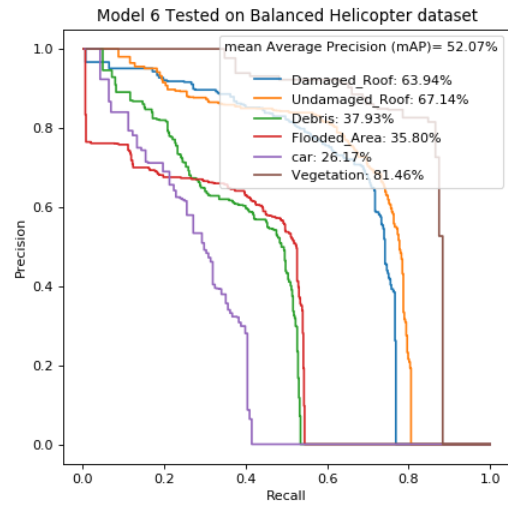
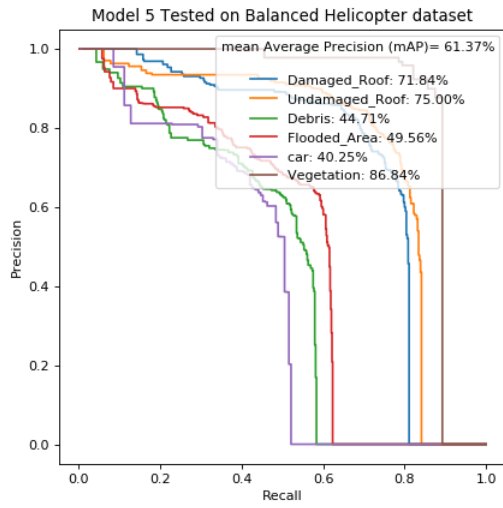


Figure 58. Continued.

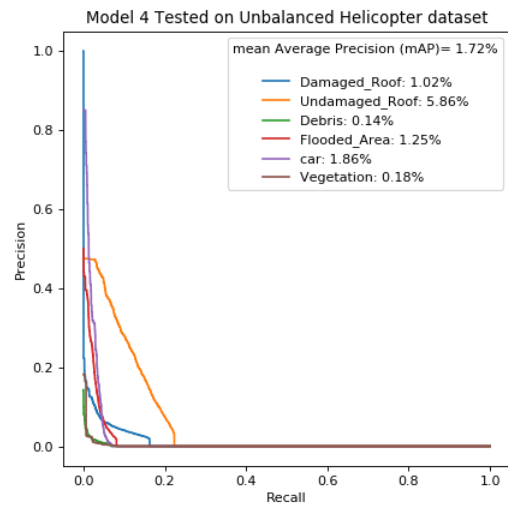
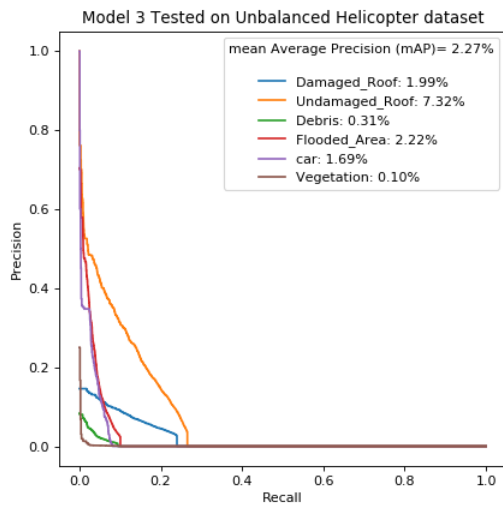
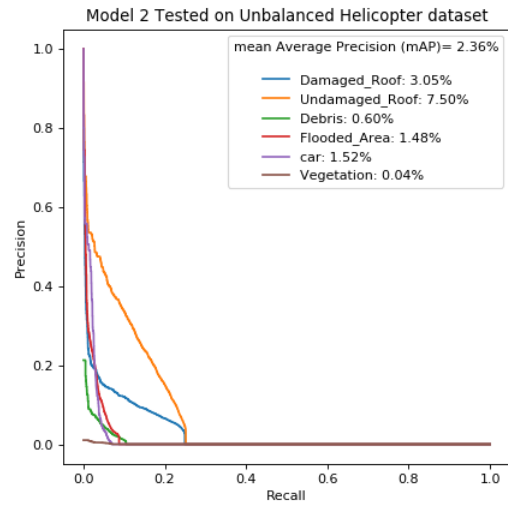
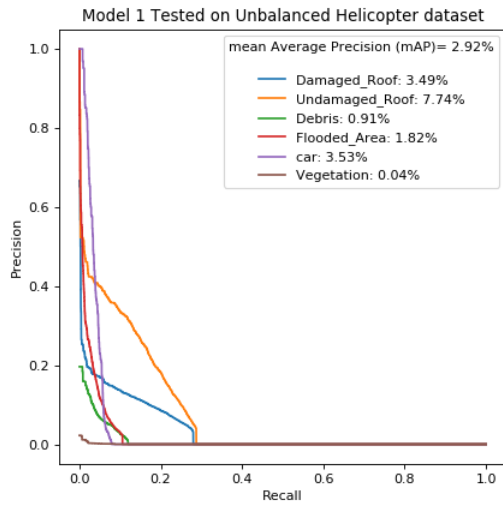


Figure 58. Continued.

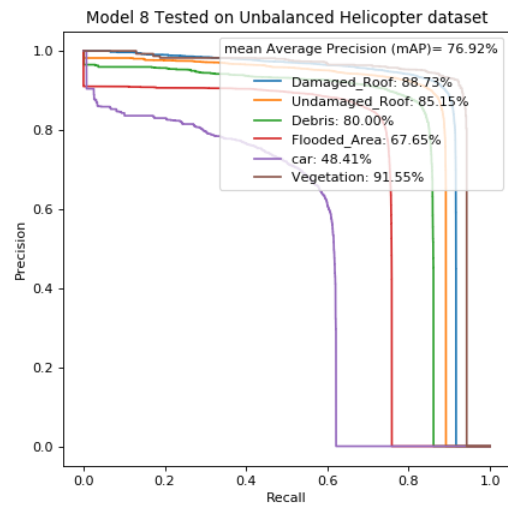
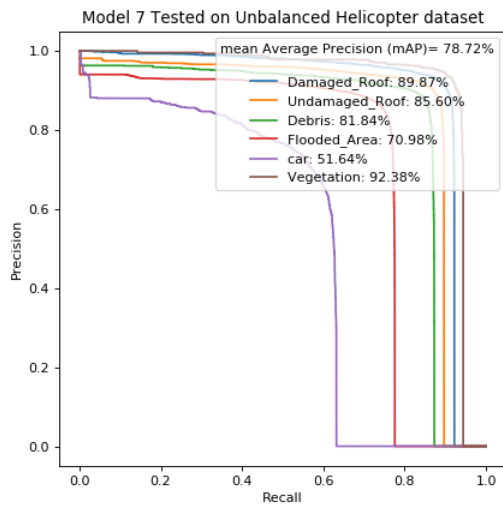
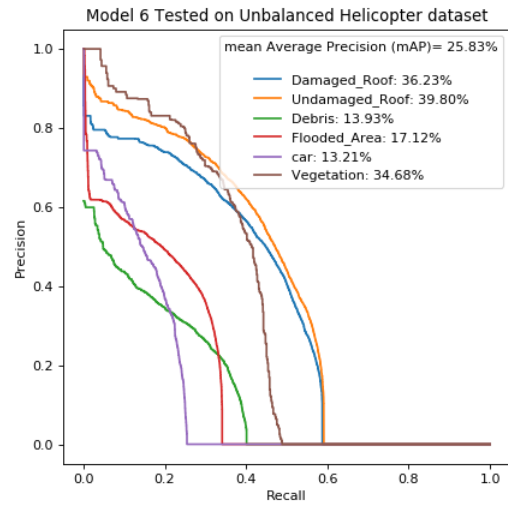
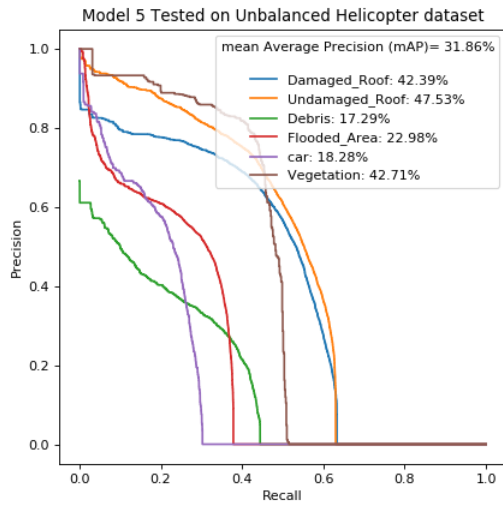


Figure 58. Continued.

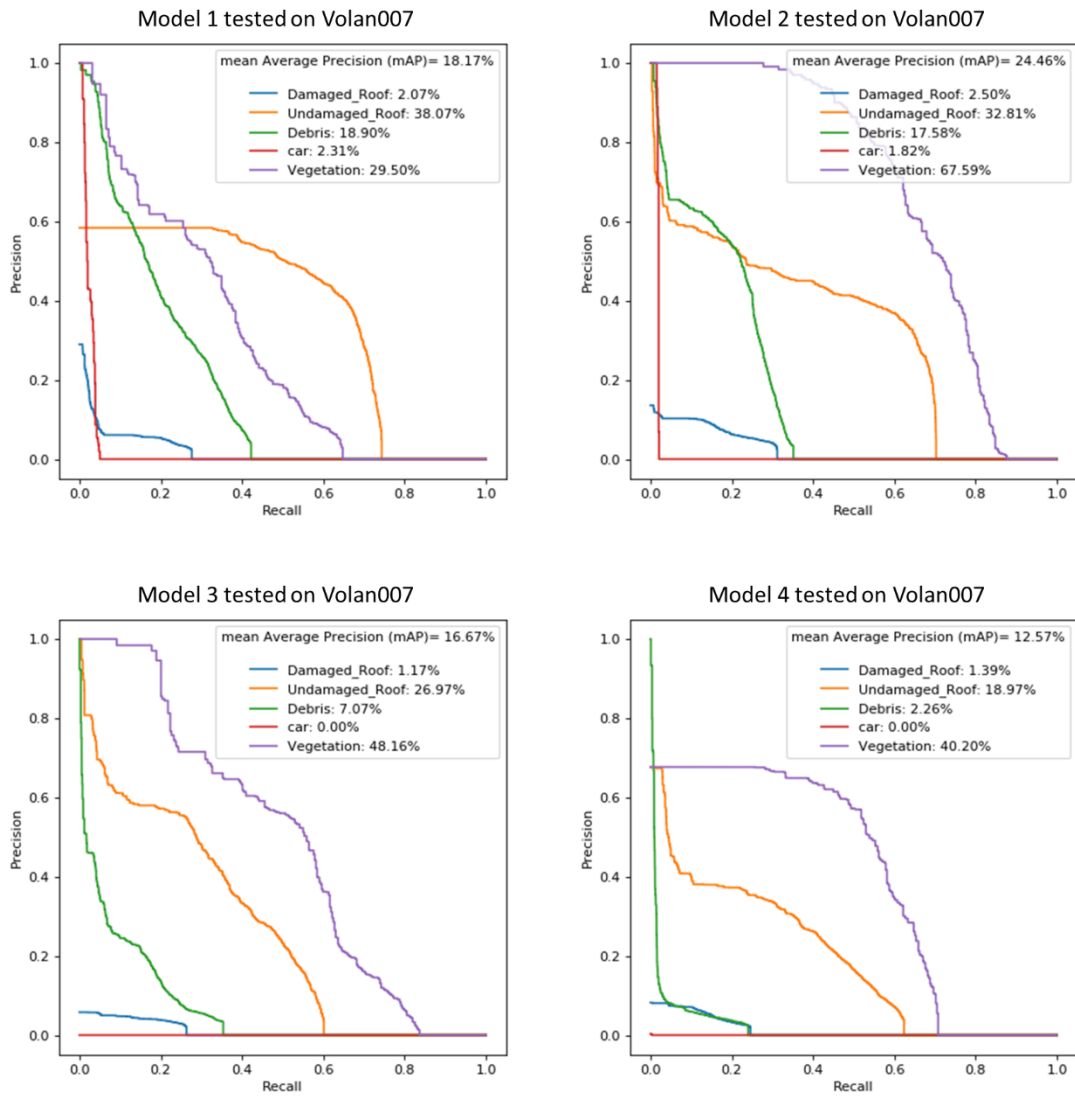


Figure 58. Continued.

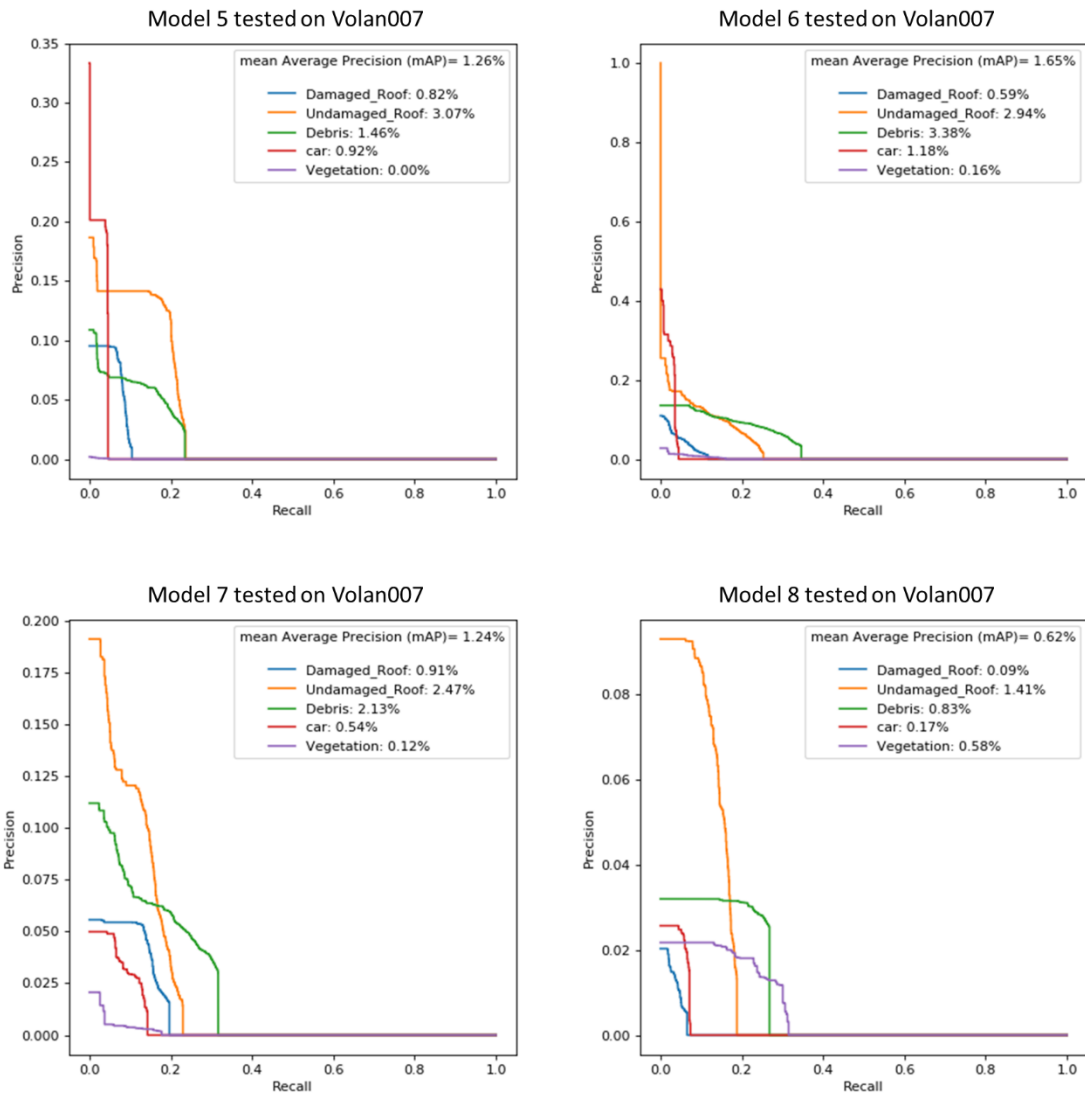


Figure 58. Continued.

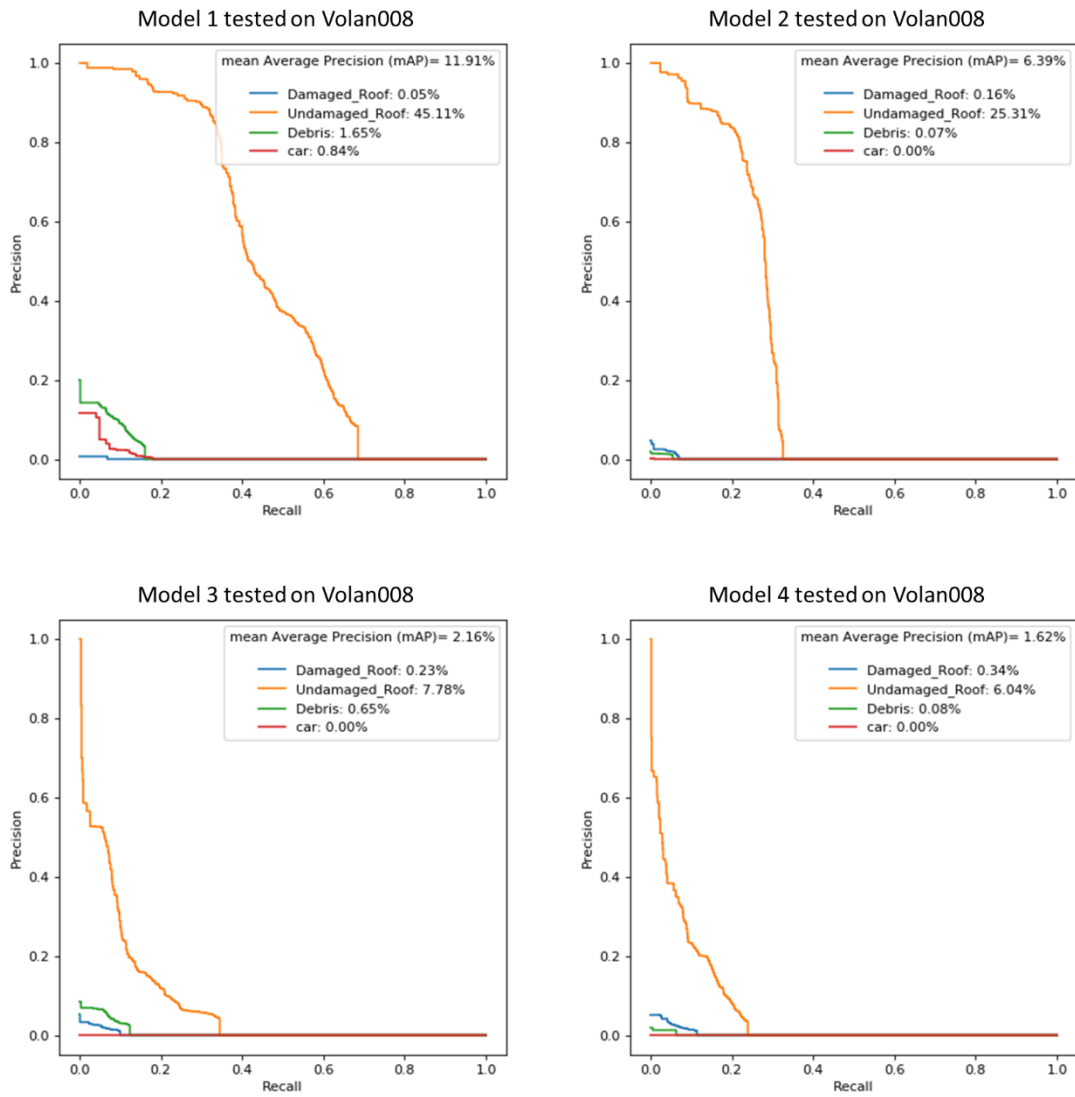


Figure 58. Continued.

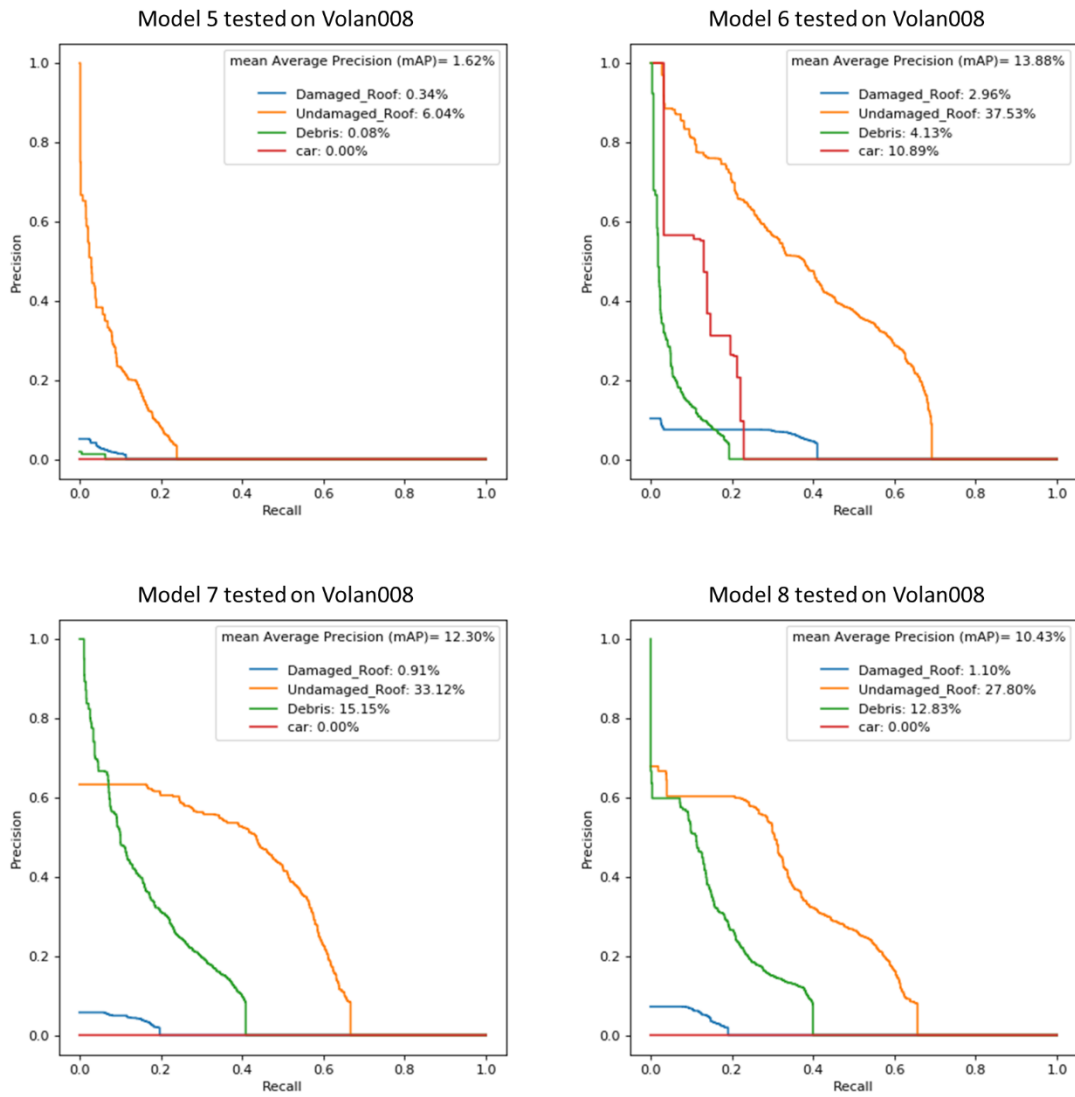


Figure 58. Continued.

APPENDIX B. MASK MODEL REGRESSION

This Appendix reports the regression results of model Mask1 for degrees of 1, 2, 3, and 4. Errors are measured as described in Chapter 5. It can be seen that there is always a positive correlation between the prediction confidence and precision. Moreover, some classes experience performance improvement as the regression nonlinearity increases.

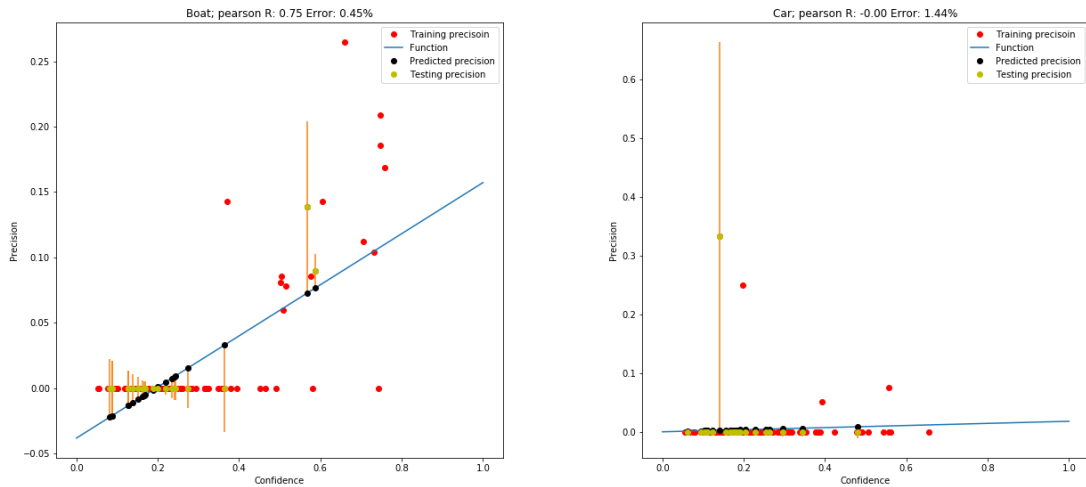


Figure 59. Mask1 confidence-precision linear regression.

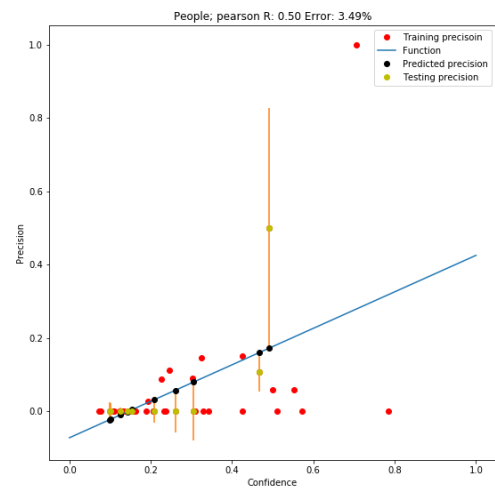
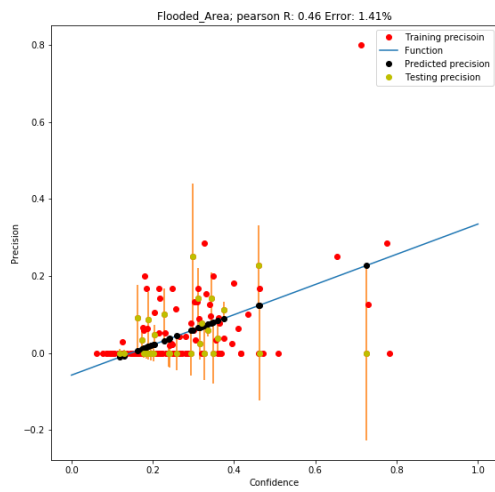
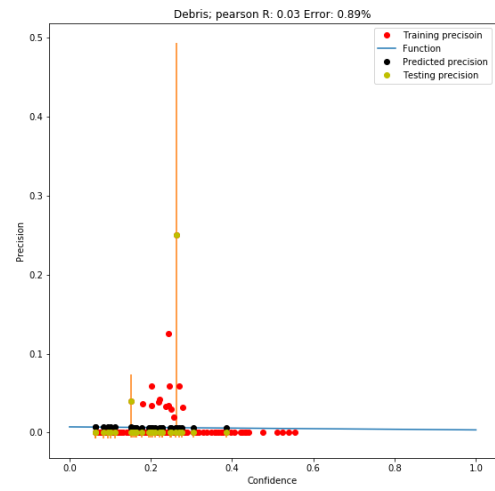
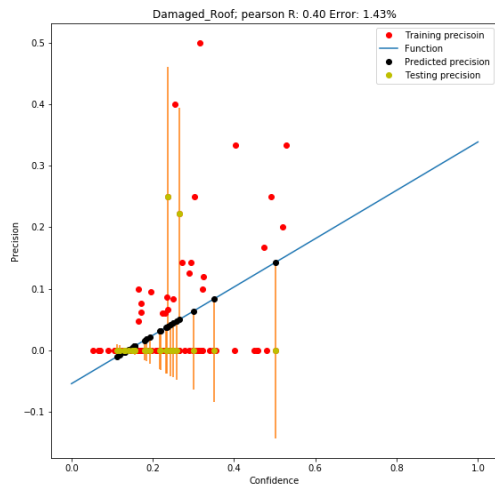


Figure 59. Continued.

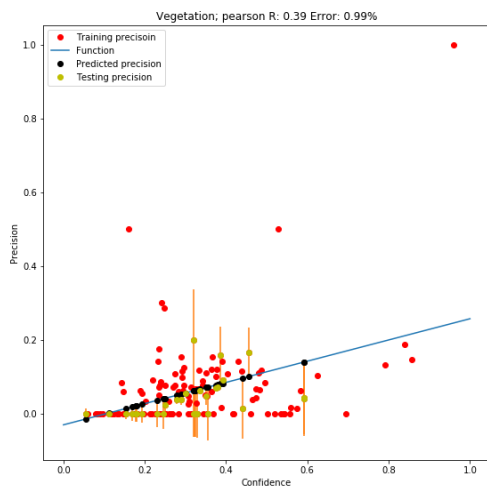
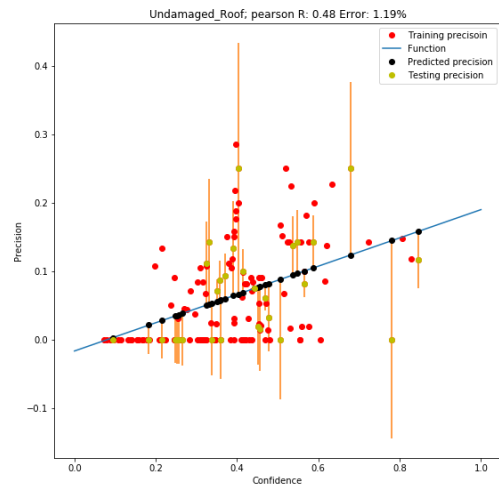
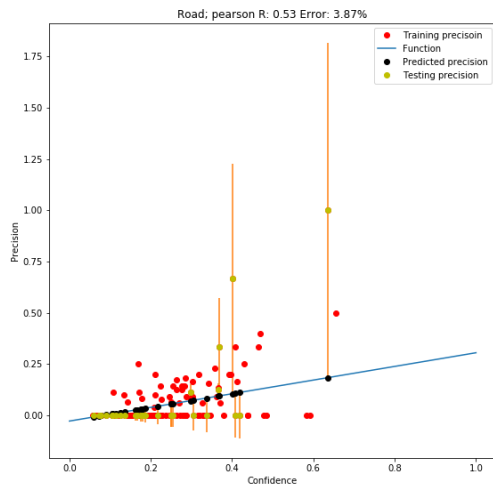


Figure 59. Continued.

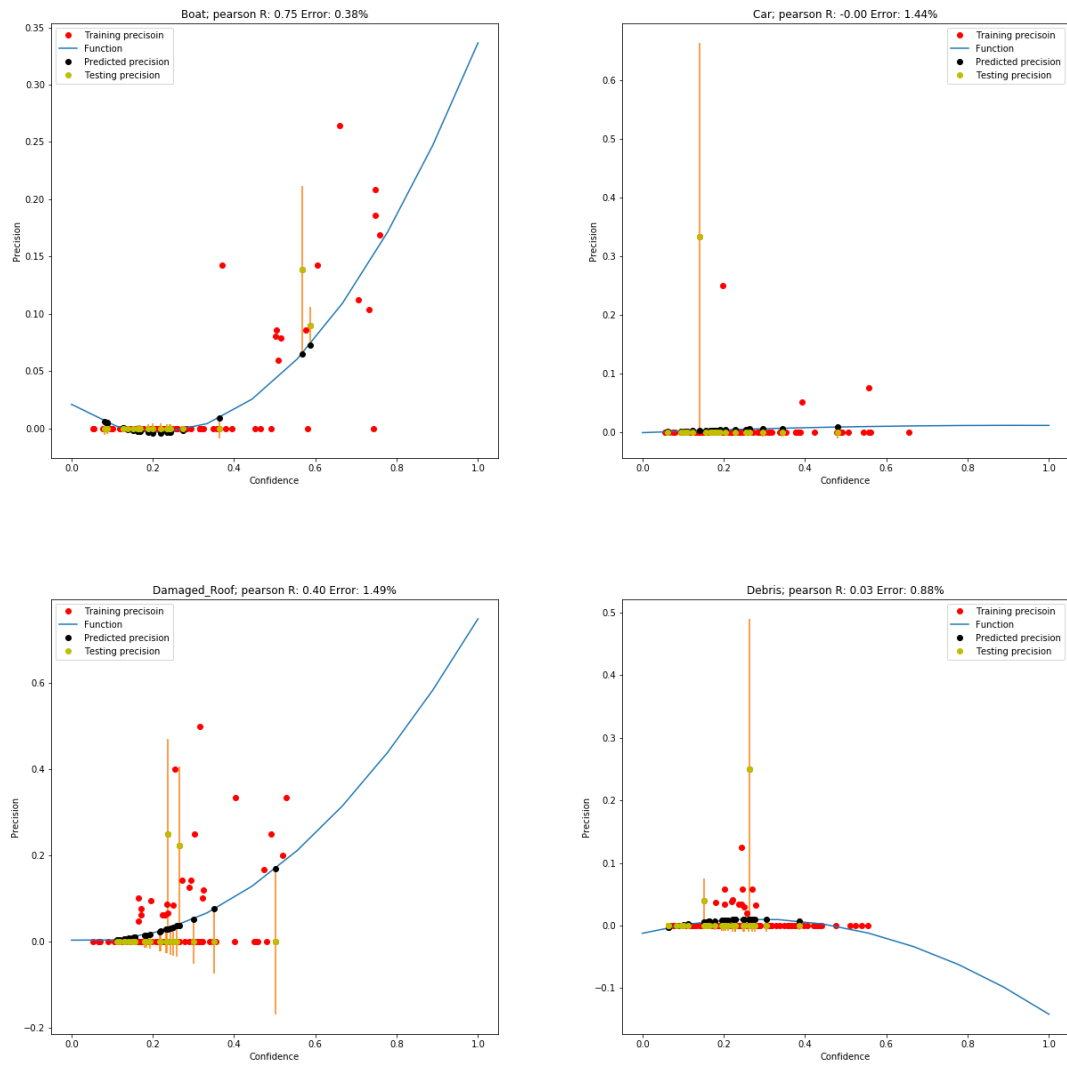


Figure 60. Mask1 confidence-precision nonlinear regression (degree 2).

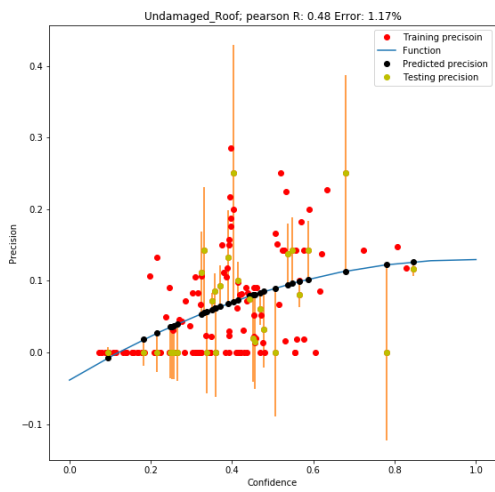
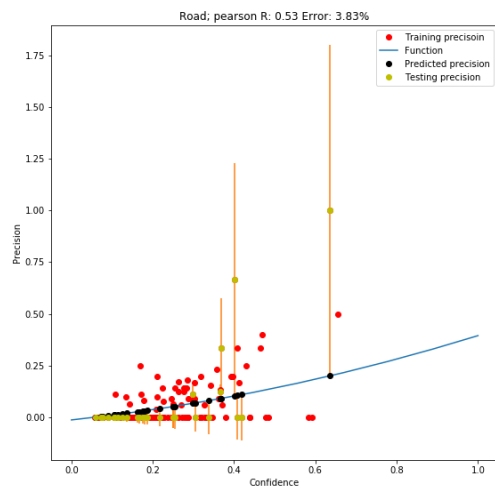
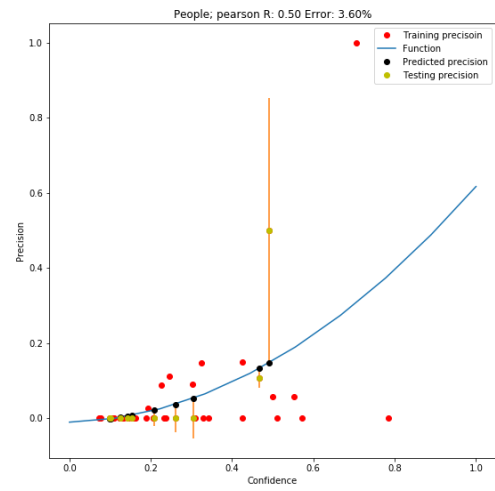
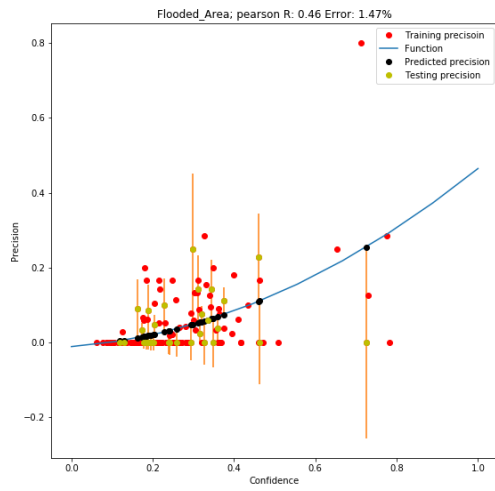


Figure 60. Continued.

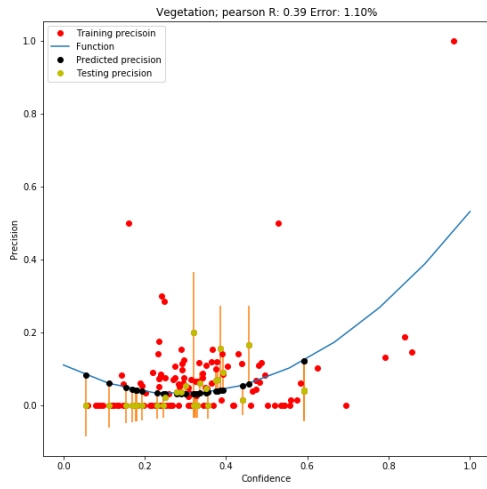


Figure 60. Continued.

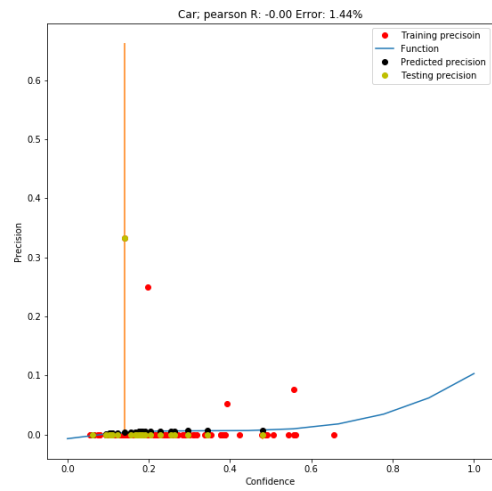
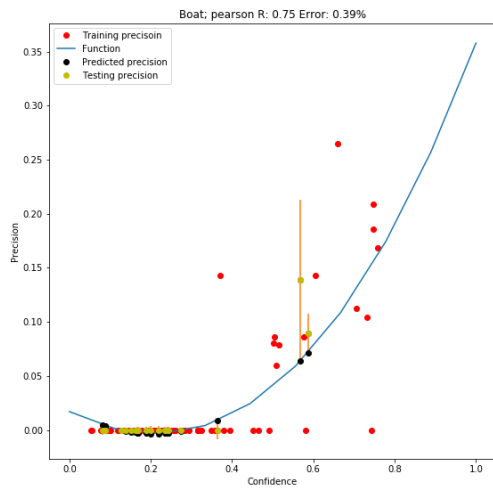


Figure 61. Mask1 confidence-precision nonlinear regression (degree 3).

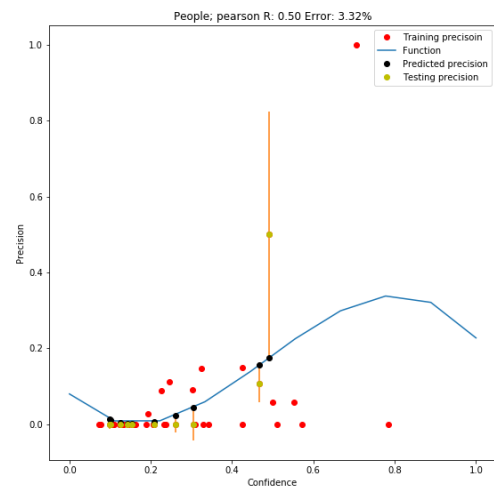
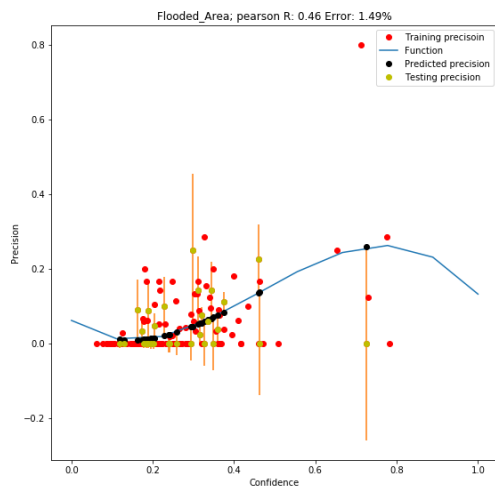
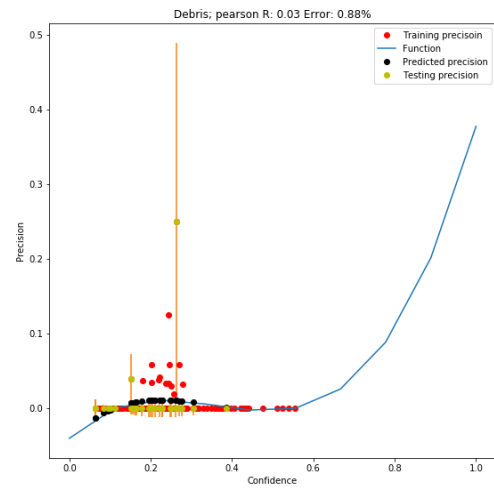
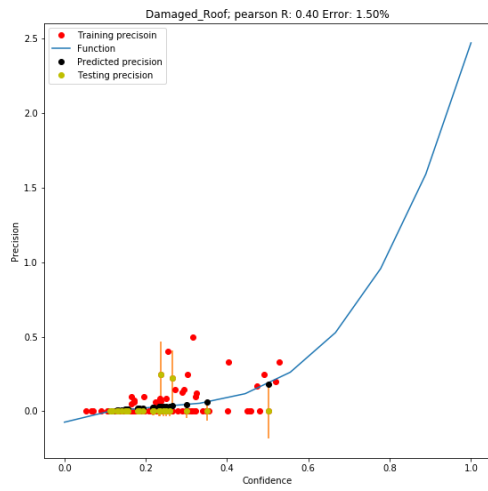


Figure 61. Continued.

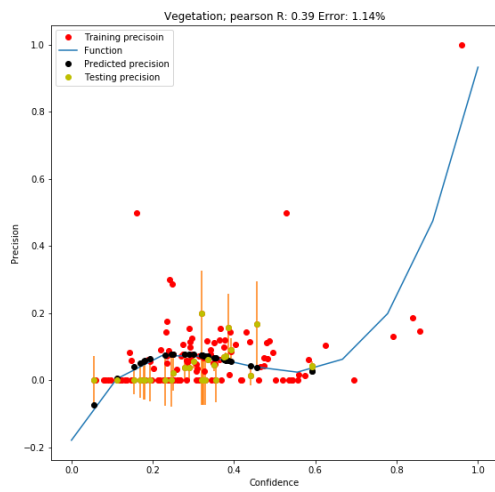
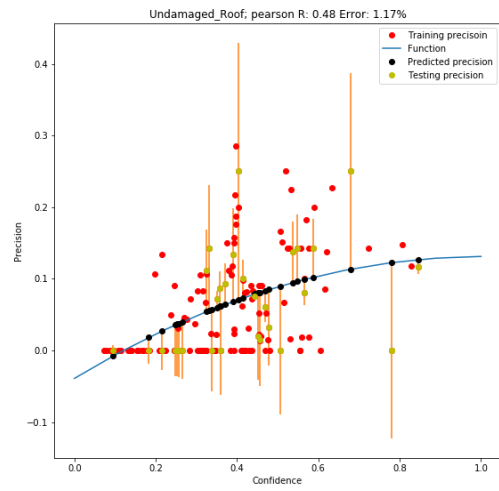
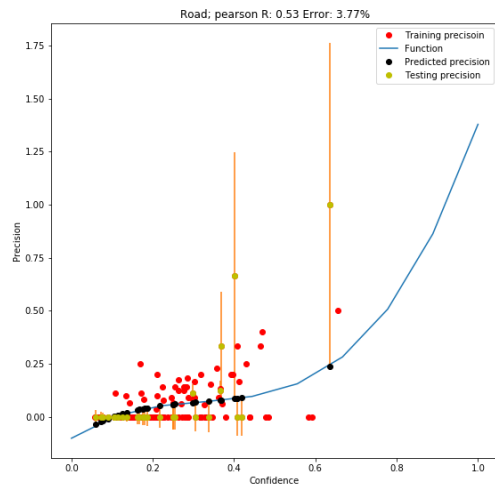


Figure 61. Continued.

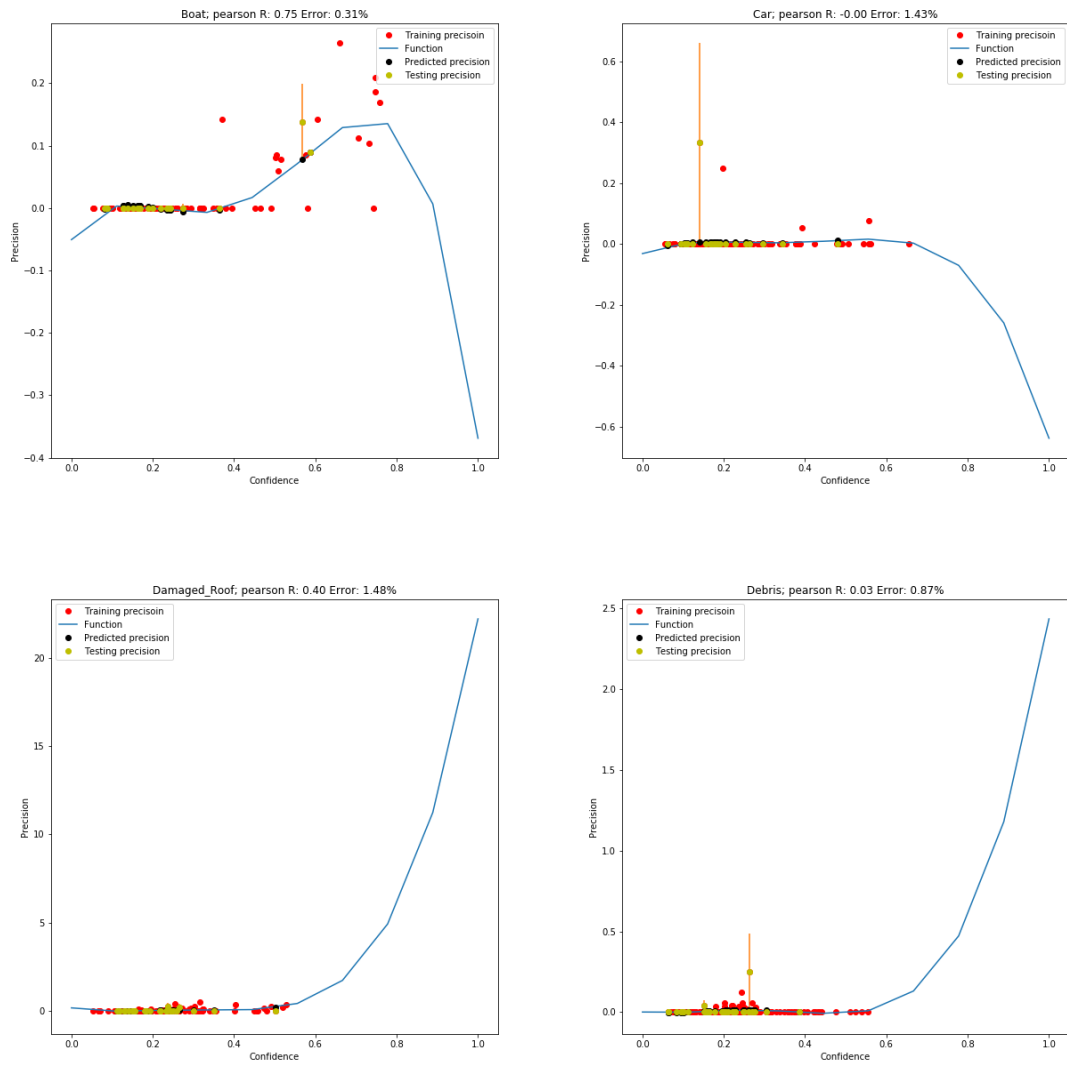


Figure 62. Mask1 confidence-precision nonlinear regression (degree 4).

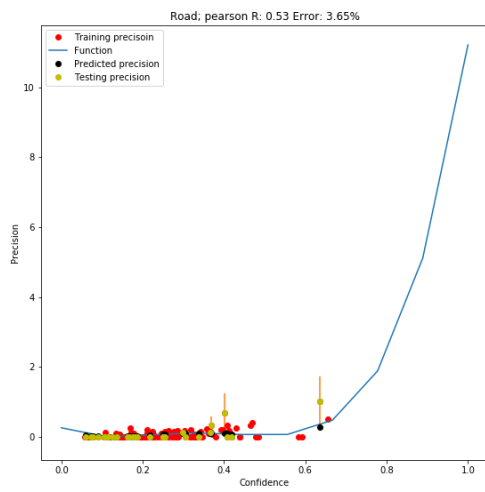
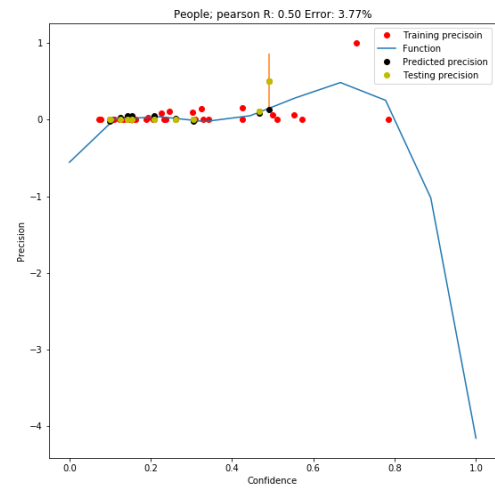
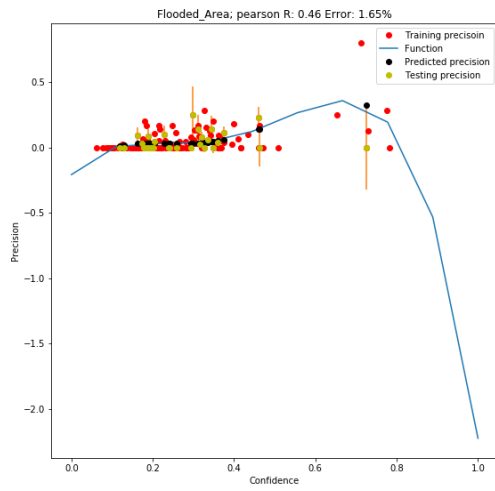


Figure 62. Continued.

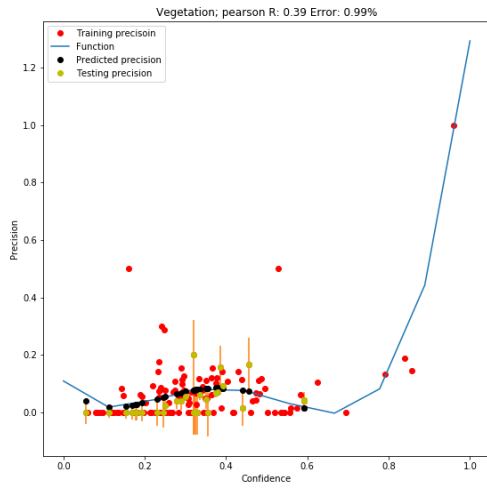


Figure 62. Continued.