



## APPROVAL SHEET

Title of Dissertation: FloodBot: Vision and AI Enabled Flood Detection Systems in Urban Environment

Name of Candidate: Bipendra Basnyat, P.E.

Doctor of Philosophy, 2022

Dissertation and Abstract Approved:



---

Dr. Nirmalya Roy  
Associate Professor  
Information Systems

Date Approved: 04/17/2022

## ABSTRACT

Title of dissertation: FLOODBOT: VISION AND AI ENABLED  
FLOOD DETECTION SYSTEMS  
IN URBAN ENVIRONMENT

Bipendra Basnyat, Doctor of Philosophy, 2022

Dissertation directed by: Dr. Nirmalya Roy  
Department of Information Systems

Flash floods are one of the most commonly occurring natural disasters. However, communities are often ill-prepared for its pre-disaster precautions and post-disaster aftermath. We argue that the technical and economic resources are significant constraints in identifying, assessing, and reducing disaster risks. While other mature flood protection mechanisms exist, they are often expensive and site-specific. Expensive flood detection and control mechanisms are often limited to affluent communities, increasing the risk of flood damage to less affluent areas. Our research develops economically viable, scalable, and mobile flash flood detection systems that reduce disaster risks. We explore various state-of-the-art machine learning models, the Internet of Things (IoT), crowd-sourcing, participatory sensing, and cloud infrastructure to deliver social media-based flash flood detection systems called FloodBot. The FloodBot is a scalable, mobile, and end-to-end mass-deployable alternative flash flood detection system based on vision, sound, and social media content. The FloodBot's vision is enabled by state-of-the-art computer vision (CV) techniques; its auditory capabilities are enabled by acoustic scene classification (ASC) techniques, while conversational AI enables its speech processing capabilities.

In this thesis, we propose novel multimodal deep learning frameworks and cross-domain transfer learning techniques to classify flood severity, perform object recognition and segmentation. We employ pre-trained transfer learning techniques to enhance the accuracy of traditional/hand-crafted models and attain 97% accuracy under an ideal scenario. Specifically, we posit deep learning models such as convolutional neural networks (CNNs), single-shot multi-box object detection (SSDs), and segmentation models for vision-based tasks. We augment the vision by sound (Mel-Spectrogram) based deep learning models and classify environmental sounds pertinent to flood during weather adversaries (low light, bad weather). Our experiments using the sound signal and deep learning models attest that we can classify flood-related sound events with a 78% accuracy, even in adverse weather conditions (heavy rain, strong wind). Finally, we demonstrate how memory-based end-to-end pre-trained language models such as Bidirectional Encoder Representations Transformers (BERT) can enable conversational power and seamlessly integrate the FloodBot into social media.

In summary, this thesis assesses the relevancy of the multimodal information (image, sound, and social media) and integrate them together to deliver proactive notifications such as tweets to reduce damages and prepare the community during natural disasters. We have deployed FloodBot in Ellicott City – a severe flash flood-prone area in Maryland with a live Twitter handle `umbc.floodbot` in collaboration with Howard County Storm Water Management Division and released more than 24 hours of annotated multimodal AI-Ready data (video recordings) in the public domain to foster further research for natural disaster monitoring in the community.

# FLOODBOT: VISION AND AI ENABLED FLOOD DETECTION SYSTEMS IN URBAN ENVIRONMENT”

by

Bipendra Basnyat, P.E.

Dissertation submitted to the Faculty of the Graduate School of the  
University of Maryland, Baltimore County in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
2022

Advisory Committee:

Dr. Nirmalya Roy, Chair/Advisor

Dr. Aryya Gangopadhyay, Co -Advisor

Dr. George Karabatis,

Dr. Maryam Rahnemoonfar,

Dr. Sudipta Sarkar, NASA

© Copyright by  
Bipendra Basnyat  
2022



## Dedication

I dedicate this dissertation to my father Dr. Birendra Bir Basnyat, mother Mrs. Bindu Basnyat, wife Seema Adhikari Basnyat, and my two daughters, Binisa and Stuti Basnyat.

## Acknowledgments

With my head bowed down and touching both your feet, I dedicate this work to the most hard-working man, my inspiration Buwa (father): Dr. Birendra Bir Basnyat. Next, with the same admiration and utmost sincerity to my Mamu (mother) Bindu Basnyat. If it were not for either of you, I would have either never ventured this path or come to a point where I could write the closing chapter of my PhD. This degree in Information Systems is all to your hard work and belief in me. The energy to push the limits comes from observing both of your continued perseverance even at this age. Then to my friend forever, my wife, Seema Adhikari Basnyat, for standing by every decision and supporting our family while I disappeared in pages of this dissertation.

Thank you, Professor Nirmalya Roy, for motivating and guiding me throughout my PhD. endeavor, being there, and supporting every crazy idea around my research. The most rewarding part of my Ph.D. journey was learning and walking alongside Dr. Roy. My co-advisor, Dr. Aryya Gangopadhyay, and his calmness would always inspire me. Dr. Gangopadhyay's way of simplifying things or looking at a complex problem with such ease gives me a notion of what a great learned people in the field should be.

I would also like to thank my other committee members, Dr. Aryya Gangopadhyay, Dr. George Karabatis, Dr. Maryam Rahnemounfar, and Dr. Sudipta Sarkar, for their suggestions and feedback on my work. Their valuable comments and feedback help me shape the final piece of my thesis. I want to express my special

gratitude to Dr. Rahnemoonfar and Dr. Sarkar for sparing their time to review this thesis manuscript.

I am also thankful to all my Mobile and Pervasive Computing fellow researchers at the University of Maryland, Baltimore County. I am indebted to them for providing me with an intellectual aura and make me feel at home. Finally, the National Science Foundation (NSF) and Center for Real-time Distributed Sensing and Autonomy at UMBC for providing assistance and supporting my PhD research projects.

As my father, Dr. Birendra Bir Basnyat, and brother Dr. Bijendra Basnyat inspired me to do a doctorate, I sincerely hope that my work will inspire others, specifically my daughters Binisa and Stuti to continue the tradition of continuing the education till the end and add more PhDs to the family.

Finally, this dissertation brings peace with my dream that I saw two decades ago as a student at University of Dayton Ohio, where I conceived the thought of extending my name to Bipendra Basnyat, P.E., PhD. These are two hard earned acronyms that I will always be proud of.

# Table of Contents

List of Figures	viii
List of Abbreviations	x
1 Introduction	1
1.1 Flood Detection: General Architecture . . . . .	4
1.2 Thesis: Scope . . . . .	6
1.3 The contribution of this Thesis . . . . .	7
1.4 Thesis Organization . . . . .	10
2 Related Work	13
2.1 Multi Modal Data Analysis . . . . .	17
2.2 Image Analytics . . . . .	19
2.3 Environment Sound Analytics . . . . .	24
2.4 Social Media . . . . .	27
3 Deployment	30
3.1 Ideation . . . . .	31
3.2 Prototype & Generations . . . . .	32
3.2.1 General System Architecture . . . . .	33
3.2.2 Gen-1 On-Prem FloodBot . . . . .	34
3.2.3 Gen-2 Edge-Computing FloodBot . . . . .	37
3.2.4 Gen-3 & Beyond . . . . .	42
3.2.5 System Evaluation . . . . .	46
3.3 Conclusion . . . . .	46
4 Data Collection	49
4.1 Community Collaborations . . . . .	50
4.2 Data Collection . . . . .	52
4.2.1 Sensor Data . . . . .	53
4.2.2 Audio Visual Data . . . . .	55
4.2.3 Weather Data . . . . .	57
4.3 Multimodal Temporal Data Fusion . . . . .	58
4.3.1 AI-Ready Data Release . . . . .	60

5	Image Analytics	65
5.1	Shallow Learning . . . . .	67
5.1.1	Feature Extraction . . . . .	69
5.1.2	Data Annotation . . . . .	72
5.1.3	Shallow Learning . . . . .	74
5.2	Deep Learning . . . . .	75
5.2.1	V-M1 Flood Classification . . . . .	78
5.2.2	V-M2 Object Detection . . . . .	79
5.2.3	V-M3 Image Segmentation . . . . .	80
5.3	Model Architectures . . . . .	81
5.4	Implementation . . . . .	89
5.4.0.1	Feature Extraction . . . . .	91
5.4.0.2	Data Annotation . . . . .	92
5.4.0.3	Deep learning Pipelines . . . . .	94
5.5	Experimental Results . . . . .	98
5.6	Metric learning . . . . .	106
5.6.1	V-M4: Similarity Search . . . . .	107
5.6.2	Model Architecture . . . . .	113
5.6.3	Implementation . . . . .	115
5.6.4	Experimental Results . . . . .	117
5.7	Conclusion & Discussion . . . . .	123
6	Environmental Sound Analytics	128
6.1	Model Architecture . . . . .	136
6.2	Implementation . . . . .	136
6.2.1	Data Collection . . . . .	137
6.2.2	Pre-Processing . . . . .	138
6.2.3	Feature Generation . . . . .	139
6.3	Experimental Results . . . . .	142
6.4	Conclusion & Discussion . . . . .	147
7	Social Media Analytics	150
7.1	Social Media Integration . . . . .	152
7.1.1	Social Media Pipeline . . . . .	154
7.1.2	FloodBot in TwitterVerse . . . . .	155
7.2	AI Converse . . . . .	158
7.2.1	Implementation . . . . .	160
7.2.2	AI Conversing Pipeline . . . . .	161
7.2.3	Models . . . . .	165
7.3	Experimental Results . . . . .	170
7.3.1	Results : Social Media Integration . . . . .	170
7.3.2	Results : AI Converse . . . . .	174
7.4	Conclusion & Discussion . . . . .	183

8	Conclusion & Future Work	186
8.0.1	Synopsis . . . . .	188
8.0.2	Image - for Flood Detection . . . . .	190
8.0.3	Audio - for Flood Detection . . . . .	192
8.0.4	Social Media - for Flood Detection . . . . .	195
8.1	Future Work . . . . .	196
	Bibliography	200

## List of Figures

1.1	Flood Detection System - General System Architecture . . . . .	4
1.2	Flood Detection Framework: Scope . . . . .	6
1.3	Thesis Organization . . . . .	11
2.1	Research Analogy . . . . .	17
3.1	Ideation . . . . .	31
3.2	General System Architecture . . . . .	33
3.3	Gen-1 Deployment . . . . .	34
3.4	Gen-2 Deployment . . . . .	37
3.5	Water Level Riser . . . . .	38
3.6	Typical FloodBot Setup . . . . .	42
3.7	Gen-3 - Deployment . . . . .	48
4.1	July 2016 Flood Damage: Ellicott City . . . . .	49
4.2	May 2018 Flood Damage: Ellicott City . . . . .	50
4.3	FloodBot-Watershed/Scope Area . . . . .	51
4.4	FloodBot-Deployment Map . . . . .	52
4.5	FloodBot - Sensor . . . . .	53
4.6	FloodBot Water Depth . . . . .	54
4.7	Hodrick-Prescott Filter . . . . .	55
4.8	Flood-Watch Camera . . . . .	56
4.9	Sample Images Captured by FloodBot . . . . .	56
4.10	Weather Data . . . . .	58
4.11	Multi Modal Temporal Data Fusion . . . . .	64
5.1	Color Histogram as Image Feature . . . . .	70
5.2	Base Image . . . . .	71
5.3	Images for SSIM Base Metrics . . . . .	73
5.4	Deep Learning Models . . . . .	77
5.6	V-M1 CNN Architecture . . . . .	82
5.5	Layer wise Architecture V-M1 . . . . .	83
5.7	Classification & Detection Pipeline . . . . .	95
5.8	Segmentation Pipeline . . . . .	96

5.9	Output: Custom CNN . . . . .	99
5.10	Results: V-M2 . . . . .	100
5.11	Results: Semantic Segmentation . . . . .	103
5.12	Qualitative Result-Deep Lab . . . . .	103
5.13	Siamese Network . . . . .	114
5.14	Deep Learning Pipeline . . . . .	116
5.15	Negative & Positive Image Pair . . . . .	118
5.16	Negative Positive Clustering . . . . .	119
5.17	Triplet Loss - White Anchor . . . . .	120
5.18	Triplet Loss - Black Anchor . . . . .	121
5.19	Empirical Results . . . . .	122
5.20	Model Stacking . . . . .	124
6.1	Sound Event Detection-Approach . . . . .	129
6.2	Flood Detection Pipeline . . . . .	130
6.3	Sound Detection Framework . . . . .	132
6.4	Log Mel-Spectrograms . . . . .	133
6.5	Flood-Watch Recordings . . . . .	137
6.6	Sample Data points & Weather Distribution . . . . .	138
6.7	Weak Label . . . . .	142
6.8	Wind Event Classes . . . . .	144
6.9	Results: Binary Flood Classification . . . . .	145
6.10	Result:Flood Multi Classification . . . . .	146
7.1	FloodBot Tweets : Word Cloud . . . . .	153
7.2	Social Media Integration Pipeline . . . . .	154
7.3	Social Media Integration Components . . . . .	156
7.4	AI Conversing Pipeline . . . . .	162
7.5	Attention Mechanism . . . . .	168
7.6	Longitudinal: System Stability . . . . .	172
7.7	Social Media & Relevancy . . . . .	173
7.8	Floodbot: Q & A System . . . . .	175
7.9	Example 1 Q&A score span visualization . . . . .	179
7.10	Example 2 Q&A score span visualization . . . . .	180
7.11	Question Answering System Using BERT . . . . .	182
8.1	Summary: Flood Detection Approach . . . . .	188
8.2	Synopsis . . . . .	189
8.3	Vision Model Results . . . . .	190
8.4	Weekly Object(Car) Count During Covid '19 . . . . .	192
8.5	Qualitative Analysis: Multimodal Flood Detection . . . . .	194
8.6	Elevated View During Flood . . . . .	196

## List of Abbreviations

DRR	Disaster risk reduction
MLP	Multi Layer Perceptron
CNN	Convolutional Neural Network
ERM	Empirical Risk Management
IoT	Internet of things
IA	Intelligent agent
IaaS	Infrastructure as a Service
DSS	Decision Support System
FSL	Few Shot Learning
WOE	Weight of evidence
MCDM	Multi Criteria Decision Making
ML	Machine Learning
DNN	Deep Neural Network
VQA	Visual Question and Answering
ASC	Acoustic scene classification
SED	Sound event detection
ESC	Environmental sound classification
MVP	Most viable product
STR	Scene text recognition
HP	Hodrick-Prescott
MSE	Mean Squared Error
SSIM	Structural Similarity Index Measure
SSD	Single Shot Detection
RPN	Regional Proposal Networks
CV	Cross-Validation
STFT	Short Time for Fourier Transform
VPS	Virtual Private Server
SUTime	Stanford Temporal Tagger
AED	Acoustic Event Detection

## Chapter 1: Introduction

Flooding destroys livelihoods, depletes valuable resources, and kills people. Numerous efforts are being made to develop a flood warning system with the goal of minimizing or mitigating the damage caused by flood disasters. Previously, flood management and mitigation were regarded as a technical challenge best left to civil engineers. Civil and construction engineering aims to improve community safety and readiness by constructing dams, relocating infrastructure or moving people out of the flood zone. Their procedure is reliable and often life-saving, but it is expensive and out of reach for people with lower socioeconomic status.

However in recent decades, the paradigm has shifted more toward an interdisciplinary approach. Data-driven mitigation and management is emerging as a viable option for resolving flooding issues. The paradigm shift can be attributed to the smart city vision [9]. Proponents of the smart city vision believe that seeing an event occur in real-time can help us better understand its dynamics and create more context-aware applications. They believe data and digital technology improve people's lives. Communities can monitor events remotely and better understand how a calamity, such as a flood affects demand patterns in real-time. They can

then strategies effective mitigation plan. This means that automated data collection systems take the place of human data collectors, minimizing the possibility of human casualties. For instance, remote sensing and image-based solutions such as unmanned aerial vehicles (UAVs) or other noninvasive technologies can collect real-time data for disaster risk reduction (DRR) without the need for human presence at the disaster site. While eliminating the need for manual data gathering saves money, it also generates a large volume of data, necessitating an optimized data/information extraction system.

Breakthrough in sensor technology enabled us to deploy sophisticated sensors and cameras in rugged environments [128]. We can now sense elements of the physical world that were previously unavailable due to advancements in sensor technology and microchips. This is the intersection of the Internet of Things (IoT) with our current understanding of cyber-physical systems. Our capacity for observing and comprehending environmental occurrences has increased significantly, transforming environmental disasters into a data burst.

Even though signal collection from remote systems has become relatively straight forward, information extraction remains a challenging research problem. Extracting the appropriate signal type from a data burst is an iterative process. Relevant information may be concealed and discovered only after they have been properly churned. To that purpose, this thesis also tries to find the most important signal in a flood event from various types of signals. We traverse across distinct signals such as image, sound, and text from social media to identify the most in-

formative signal. We refer to these various data types as modalities, which are characterized by their distinct properties.

Lastly, we can lessen the impact of disasters in two ways: by being informed and prepared ahead of time, or by efficiently managing the chaos afterward. Flooding is a natural calamity, and the timing and manner in which it occurs are by large unpredictable. However, we can be better prepared and informed about the risks and consequences. It is possible to save lives by being well-equipped with tools and technologies to detect and respond to such crises before they occur. Our thesis focuses on the development and implementation of such remote flood monitoring system that can detect floods without the need for human involvement. As a result, human beings are safe from harm. We use images, acoustic, and social media contents to detect floods and categorize their intensity and post information on social media for awareness.

Therefore, this thesis highlights the importance of information system-based inter-disciplinary work in reducing disasters and better preparing the community. We believe it can provide a broader solution and hypothesize that machine learning and computational methods will help life-saving missions. Our approach alleviates the need to extract and examine data manually as the rate of data generation is beyond human comprehension. This thesis seeks to deliver a scalable, mobile, and end-to-end mass-deployable flood detection system based on sight, speech, and hearing. As a result, this thesis addresses common issues associated with smart city deployment, where an automated system captures data but lacks a proper down-

stream system to convert the raw data into usable information. Here we attempt to convert those otherwise non usable raw data into usable information. Simply put, we want to create a system that can successfully translate the vast amounts of raw data collected by sensors into decision-supporting facts. This thesis provides system design approaches for a downstream information extraction system based on real time deployment and data collection.

## 1.1 Flood Detection: General Architecture

A typical flood detection system and its component are shown in Figure 1.1. To help focus and emphasize the scope of this thesis, we will first define the fundamental components of conventional flood detection system. There are two major building blocks of flood detection system, the Environmental Monitoring Unit (EMU) and Decision Support System (DSS)/Alert and Warning System Unit.

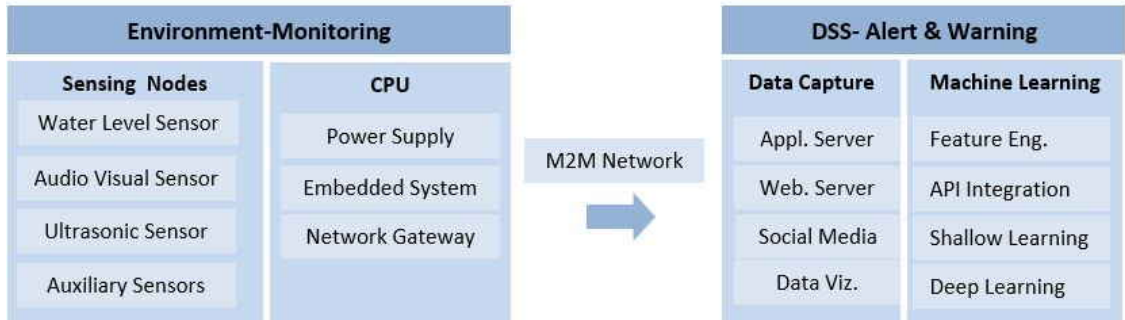


Figure 1.1: Flood Detection System - General System Architecture

### *i) Environmental Monitoring Unit*

Environmental monitoring units (EMU) are a collection of tools and techniques used to monitor and assess an environment. They capture the environmental

parameters (data points) [43]. EMU is a mesh of integrated cyber-physical sensors and data generation devices that interface with their surrounding. The EMUs used in our research are water level sensor, audiovisual (camera), ultrasonic sensor, and other auxiliary ambient sensors.

The water level sensors use a pressurized tube to determine the water level and relay data to the Central processing unit (CPU). The Audio Visual Sensor/Camera is the EMU's second crucial component, as it records temporal video. These two key components produce the data necessary for this thesis. Furthermore, EMU is equipped with solar-powered integrated systems and a networking gateway for data transmission to remote servers.

## *ii) DSS/Alert and Warning System Unit*

The Decision Support System (DSS) is the flood detection system's computational unit. After the EMU transmits data using machine to machine (M2M) technology, the raw data is stored, analyzed and converted to useful information by the DSS/Alert and Warning System Unit.

We further split DSS-Unit into two broad categories. The Data Capturing Unit - a standardized data storage system containing application servers, web servers, and an integration platform for social media. The Data Capturing Unit provides data persistence and other general application need such as web hosting, automation and other system integration. Next is the Machine learning or advanced analytics unit (focus of this thesis) where we use sophisticated algorithms to transform raw data

into usable information.

## 1.2 Thesis: Scope

With a clear understanding of general architecture of conventional flood detection system, we now define the scope of this thesis. Our flood detection systems framework and the focus areas of our work are illustrated in Figure 1.2. The thesis discusses four key topics (boxes): deployment and data collection, various data modalities, the primary purpose of these modalities, and the outcome of multimodal analysis.

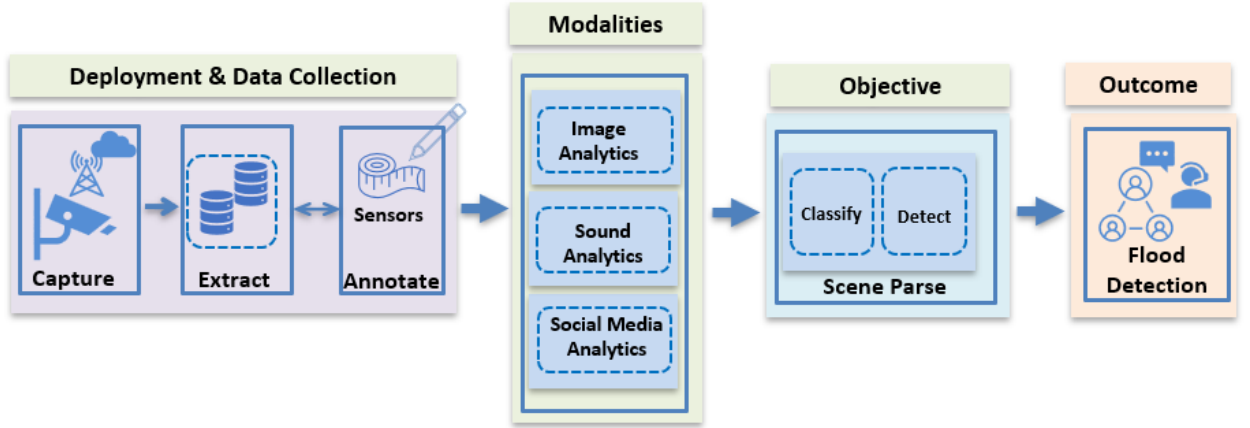


Figure 1.2: Flood Detection Framework: Scope

The first section of the thesis discusses our deployment, data collection, and sensors and how we use sensor data to annotate and validate our multi-modal (image, sound, and social media content) raw data. We then discuss our multi-modal data, experiments and various deep learning techniques, including convolutional neural networks (CNNs) and transfer learning. We can gather information about a

phenomenon or a system of interest from various tools, measurement techniques, experimental setups, and other sources. Because natural processes and environments are so diverse, it is rare for a single acquisition method to provide a comprehensive understanding. The availability of multiple datasets obtained from the same system using different acquisition methods opens up new possibilities for research. It provides context beyond what can be obtained by analyzing each dataset separately. Therefore we motivate our thesis to assess how one can use multi-modal data in parsing flood-related information. We then describe how the curated information (outputs) can be prepared for mass consumption and two-way communication in social media. The thesis concludes with a design of a prototypical functional system for flood detection.

### 1.3 The contribution of this Thesis

This thesis is an applied research performed with a defined objective to understand challenges in designing artificially intelligent flood detection system. Our thesis seeks to deliver a practical system and highlight the lessons learned during the design, development, and implementation of flood detection systems based on deep learning. We start with a simple design and gradually enhance its complexity and capacity. As we investigate and comprehend the components and obstacles, we propose enhancements and solutions for the roadblocks that impede the development of a safe and sound flood detection system.

We posit that disaster-response systems such as ours cannot rely on a single form of signal (data modalities); therefore we test a variety of physical flood detection signals, such as water level data, images, and sound (physical sensors). We infuse the social sensors [76] into the physical sensors to create even stronger flood signals and integrate our system into social media for information dissemination.

Beyond technical merit, this thesis has a broad impact to the community. We took a multidisciplinary approach and integrated ourselves into the community by collaborating with local government, industry partners and most importantly the flood impacted local stakeholders. Our work was also featured by NSF [53, 103] and widely covered by news media [146, 139, 161]. Specific thesis contributions are :

- We iterate through different designs and demonstrate our deployment in diverse setups and flood-prone locations. We discuss the deployments and their underlying hardware and software architectures. The performance of the next-generation (iteration) flood detection system is then improved by observing and mitigating the previously observed challenges.
- We traverse various data modalities (image, sound, and social media) to identify the best signals that can be used to build a robust flood detection system. More specifically, we use deploy various deep learning models and technologies to detect flood signals. We develop novel techniques for annotating data that would otherwise be difficult to obtain strong supervision using weakly supervised learning.

- Rather than being a “Lab-Only Research”, this thesis dealt with various ground-level implementation and real-world challenges. We brought together a diverse group of stakeholders to holistically explore the flood detection problem (for example, industrial partners—vendors, Howard County government—regulators, Old Ellicott City Partnership—flood victims and stakeholders, foreign collaborator—University of Pisa, and so on).
- AI-Ready Data Release - To foster long-term feasibility and continuity of this research and deployments beyond this thesis, we have released **29 hours** of raw data in public domain [28]. **20 hours** of this data has been hand-annotated with a five-minute binning that allows multi-faceted research formulation in various urban settings.

Precisely, this thesis presents result of an experience collected during an end-to-end system design of machine learning based flood detection system called ‘**FloodBot**’. FloodBot is a social media integrated distributed computing system consisting of cameras, networking devices, solar-powered computation unit, and a powerful deep learning server(s) [18, 20, 134, 22].

We use the FloodBot loosely as an generic term to describe different aspect of flood detection and our pertinent focus areas. FloodBot’s scope also vary depending on the experiment setup, modalities and our problem in hand. FloodBot is a live system deployed in Ellicott City that was devastated by major flood in 2016 July and 2018 May [119]. Our system has been live and operational since September

2019 and can be followed on Twitter at [https://twitter.com/umbc\\_floodbot](https://twitter.com/umbc_floodbot) [145] for live data and social media interactions.

## 1.4 Thesis Organization

In this thesis, we developed a multi-modal flood detection system using novel machine learning techniques and assessed them under a variety of weather conditions and data modality (image, sound, and social media). Figure 1.3 illustrates the thesis’s organization and the location of each chapter. First, in Chapter 2, we review the relevant research from these three data modalities. In Chapter 3, we describe several iterations of our deployments. There we discuss several architectures and challenges before settling into the current state of FloodBot. We also highlight our learning from those deployments and our mitigation strategies in Chapter 3.

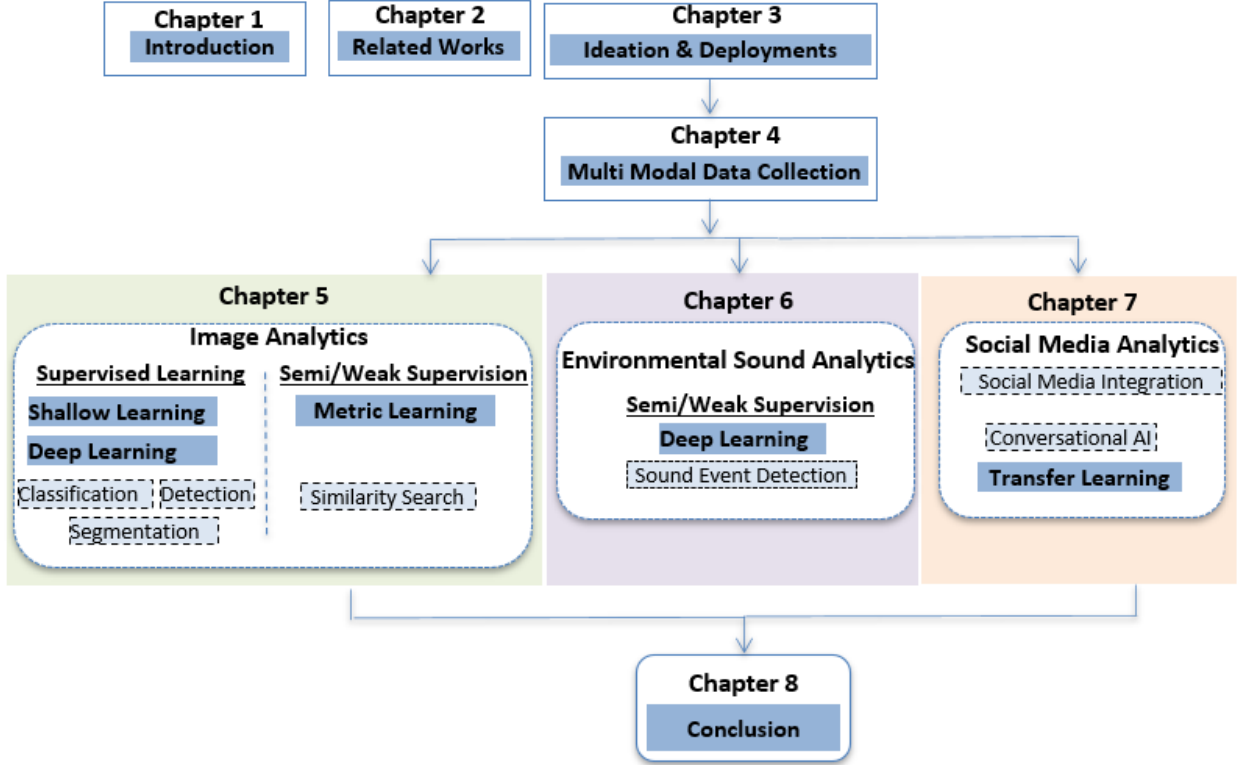


Figure 1.3: Thesis Organization

After describing the deployment and providing pertinent background information in Chapter 1 through 3; we outline our data collection in different modalities in Chapter 4. We describe our experiments and findings across three modalities in Chapter 5 (Image Analytics), Chapter 6 (Sound Analytics), and Chapter 7 (Social Media Analytics) respectively. The chapters have been separated by their data modality. Figure 1.3 shows the organization of our thesis.

In Chapter 5 we describe various experiments and deep learning models for image analytics. We begin with shallow learning progress into to deep learning. We examine the difficulties and fallacies inherent to traditional and shallow learning. We detect objects using a convolutional neural network (CNN) and transfer learn-

ing. We studied a CNN-based model that had been trained on large amounts of labeled data and then tested it on our limited labeled data. To improve flood detection capability, the parameters of the CNN model were fine-tuned utilizing transfer learning techniques. Additionally, we experimented with pre-trained models and used them to recognize objects. Furthermore, we examined flood detection using an image segmentation technique under image analytics. Then, in later part of Chapter 5, we used weak supervision and similarity search technique to compare the flood image to a non-flood image.

In Chapter 6, we switch the data modality from image to sound. We experiment with Mel Spectrogram-based sound signal and train CNN to classify environmental sound events. We demonstrate our experiment results on possibility of floods detection only using sound waves as our signal. Chapter 7 describe our work third data modality (social media) and attempts to transform the Floodbot into a conversation agent. We demonstrate how we have integrated FloodBot into social media agent and assess its information dissemination potential.

Finally, in Chapter 8, we conclude the thesis by proposing possible future research directions. There, we summarize our work and describe the work in progress that will continue beyond this thesis.

## Chapter 2: Related Work

Flood detection is an interesting topic that has attracted the attention of multiple disciplines [39, 137, 92, 74, 75]. Flooding at the watershed scale is a complicated phenomenon and nonlinear dynamic processes not easily represented by simple models [74, 65]. Typically, solving river engineering problems requires a thorough understanding of river flow, such as the flow depth, flow velocity, and flood extent. Hydraulic models predict the flow characteristics by utilizing the governing equations of the flow in motion (mass and momentum conservation principles) [107]. However, depending on their spatial extension, solving such equations can be cost and resource prohibitive. Additionally, modeling two-or three-dimensional river flows using high-resolution topographic data at large scales (national or continental) is nearly impossible [65].

Numerous data-driven models, including bivariate models of frequency ratio, Shannon entropy, the weight of evidence (WOE), and evidential belief function (EBF), have been created and used for flood mapping [127]. However, the success of such models are limited to that specific area (watershed) not transferable to different area. Multi-criteria decision-making (MCDM) models listed above also

rely on expert opinion, introducing significant bias and inaccuracy. The conventional approach, where hydraulic models [65, 75] are used to simulate water-flow and estimate downstream inundation is beyond the scope of this thesis. Instead, we try to understand the data (environmental conditions) contributing to floods using numerical methods and data-driven approaches.

Along with the physics-based hydraulic models, machine learning (ML) techniques have evolved over time [27]. The emphasis on learning from current data and experiences to improve understanding of real-world problems is also growing [97]. ML techniques approaches are particularly advantageous when existing models are incapable of fully capturing the physics in mathematical terms, the computational cost is prohibitively high, or available knowledge about the problems is limited. This has made machine learning approaches extremely useful for evaluating various aspects of water resources engineering, rainfall-runoff estimation, flood susceptibility mapping, and flood prediction e.t.c. [84, 129, 2].

With its continuous surveillance of the flooding river, our thesis's FloodBot deployment generates many audiovisual recordings, sensor readings, and weather data. As a result, we have collected abundance of complex granular and raw incomprehensible data. Data thus collected cannot be translated into a mathematical expression or model to adequately represents the events (flood). We are also unable to use these raw data and draw a mathematical model or solvable equation to detect the flood event [65].

Complex environments, such as ours, are frequently modeled using parameter estimation and pattern recognition techniques, [46]. Parameter estimation [143] attempts to identify and model nonlinear and complicated interactions between inputs and outputs; draw inferences and generalizations; and uncover underlying correlations, patterns, and so on.

As a result, this thesis attempts to model the environment and feed our multi-modal raw data as an input in order to find a function or model that can detect pertinent patterns associated with flood/flood causing events. In other word, we cast the problem of flood detection into a to parameter estimation problem or inverse modeling (let the environment be unknown function). The idea is then to find a system that solves an optimization problem by maximizing or minimizing an objective/cost/fitness function to find the best model parameters [4] within a permissible range (referred to as solution space) (for example, residual sum of squares).

To perform the parameter estimation, we need powerful algorithms that can find patterns during inverse environmental modeling (estimating parameters). Machine learning (ML) techniques have made significant contributions to the evolution of prediction systems that provide more accurate and cost-effective solutions [99] in estimating parameters from such complex data set. One such approximation tool is Artificial neural network (ANN).

ANN has been shown to capture the non-linearity and detect patterns successfully in various applications [5, 149, 110] or data representation. ANN are computer

systems comprised of interconnected nodes that function similarly to human brain neurons. ANN learn by detecting the hidden relationships within data through pattern recognition [29]. Pattern recognition is a type of data analysis that employs machine learning algorithms to detect patterns and regularities in data automatically. ANN can learn and model nonlinear and complicated interactions between inputs and outputs, generating inferences and generalizations and uncovering hidden links, patterns, and predictions [110]. Due to its excellent approximation capabilities [85], ANN has been applied in a variety of areas, including modeling and identification of complex and nonlinear systems and optimization and automation [108, 24]. However ANN fails to scale in complex problem setting. Due to the slow learning speed, over-fitting, and constraints on local minima, determining critical parameters such as training algorithms, activation functions, and hidden neurons is relatively tedious [123].

Deep learning is used to circumvent this limitation of ANN by stacking multiple ANN layers and transforming them to DNN. DNN is type of an artificial neural network (ANN) with multiple layers between the input and output layers. Compared to its predecessor ANN, DNN is a highly scalable machine learning technique that can learn directly from raw data without manual feature engineering [25, 57, 121]. DNN have been widely used for representation learning in recent years and have demonstrated superior performance in a variety of applications, including image classification, object detection, and speech recognition [25, 121, 57]. Thus inspiring us to experiment with multiple DNN architectures and follow their approaches.

We show the research analogy and our motivation based on related work in Figure 2.1. We use computational methods and parameter estimation technique to understand/simulate the flood causing environment (sensor and audio visual recordings FloodBot) and deploy multiple DNNs architectures to help us parse/model the flood related data representation.

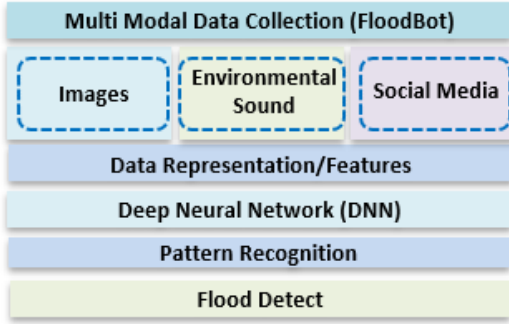


Figure 2.1: Research Analogy

We are specially interested to learn flood related data patterns (parameters) hidden in three data modalities (Image, Sound and Social Media) captured by Floodbot. Our work therefore focuses in utilizing these data modalities as input and various types of DNN as an approxi-

mator. We exploit the representational power of deep neural network (DNN) across three data modalities.

## 2.1 Multi Modal Data Analysis

FloodBot is envisioned to converse and inform users about the flood situation remotely. As a result, it is responsible for two fundamental tasks: first, comprehend the flood signal, and second, posses linguistic features necessary to communicate. These two tasks falls under overarching study area of Visual Question and Answering. A VQA system takes image and open-ended natural language question as input

and returns a natural language answer as the output [10]. VQA is a vast and complicated research area that is frequently regarded as the pinnacle of current Artificial Intelligence capability. Automatic image understanding and artificial intelligence are complex because of time, hardware, software, data volume, etc., and are not possible at a smaller scale. Therefore, instead of making this thesis a full VQA problem, we limit our work into understanding the building blocks VQA in context of natural disasters (flood).

FloodBot captures various data that can supplement the VQA task. For example, images from the video enables us to identify physical objects, while audio (e.g., the sound of pouring rain) can provide additional context required to detect flood. FloodBot’s social media presence, data curation and content parsing can provide context and link physical sensors with human and their feelings during the disaster. [70]. Stitching these three modalities and information helps us design a robust flood detection system.

Multimodality is a very natural concept for living creatures. External and internal sensors, sometimes referred to as "senses," are used by living creatures to detect and discriminate between signals. Humans can detect and discriminate between signals, communicate, cross-validate, disambiguate, and make numerous decisions in a dynamic and constantly changing internal and external environment.

Multi-modal data fusion and analytics is done for a variety of reasons. For example to obtain a more unified picture and global view of the system, improve

decision making, exploratory research, or answer specific system questions (flood detection). They can also be used to trace common vs. distinct elements across modalities or time, and to extract knowledge from data in general. Here, we will look at setups in which a phenomenon (flood) is observed using multiple instruments, measurement sensors, or acquisition techniques. The setup in which one has access to data obtained from multiple modalities is referred to as multimodal [79]. Complementarity is a crucial property of multimodality because each modality adds value to the overall setup that cannot be deduced or obtained from any of the other modalities in the setup. Our multi-modal signal analysis reduces the risk of uni-modal data failure and widens our ability to detect flood-causing environment further. To that end, the following section reviews related work in each of the data modalities. We evaluate the work in three areas: first for image analytics, then the environmental sound, and finally social media.

## 2.2 Image Analytics

Image recognition and analytics is a branch of Computer Vision (CV) and artificial intelligence. CV aids computers in recognizing locations, people, objects, and other elements in images. But before a computer vision model can be successful at a specific task, such as flood detection, it must first perform fundamental tasks such as image recognition, classification, or detection. The success of image recognition is primarily attributed to a deep neural network architecture known as convolutional

neural network (CNN) [121].

Our thesis explores various CNN based architectures [60, 147, 120]. Convolutional neural networks introduced by [82] replaced the fully connected layers with discrete convolutions and drastically reduced the parameters compared to its predecessor multi layer perceptron (MLP). The thesis' guiding principle of teaching machines to understand images and their content is at the heart of deep learning and computer vision research. Some of the groundbreaking architecture that has advanced the research area are listed chronologically LeNet [82], alexNet [77], Visual Geometry Group of the University of Oxford [132], GoogLeNet the winner of ILSVRC 2014 [142] , and the ResNet [60]. The majority of our work in Chapters 5 is based on a variant of CNN. We employ a custom CNN for flood detection that is loosely based on the LeNet [82] architecture, and ResNet [60] variant for object detection.

Object detection is the task of detecting instances of objects of a certain class within an image. We use object detection technique to determine what is nearby the flooded stream. In Chapter 5, we start by finding if a person or a vehicle is within the flood zone or classify the severity of the flood using CNN. However, that approach is unable to establish the object's vicinity to a flooding stream. As a result, we further examine a more sophisticated computer vision concept known as semantic segmentation [23]. The semantic segmentation in pixels enables us to compute and infer the proximity of increasing water levels to nearby objects [23]. As with the alarm system on an autonomous driving vehicle [30], our implementation can alert

authorities when the water level rises and the impending danger to surrounding items reaches a certain threshold .

Semantic image segmentation is also called pixel-level classification. It is the task of clustering parts of the image together which belong to the same class. Object detection deals with finding different objects in an image and classify them into one category [166]. Previously computer vision task was performed using low-level image abstraction such as HOG, edge, pixel histograms [144]. However, these methods lacked scalability until a CNN-based breakthrough method was proposed by [56] called U-Net [120]. U-Net is the most used encoder-decoder based famous models. U-Net constructs the encoder-decoder structure for semantic segmentation. U-Net is an improvised version of fully connected network (FCN)-based segmentation model [120]). Like the FCN, U-Net first creates convolutions and restores the image in the later part of the network, learning feature representation along the way. Our implementation closely relates to medical imaging [120] work where computer vision and deep learning model extracts clinically relevant information from medical images.

All the experiments discussed so far are based in supervised learning. In supervised learning, the algorithm ‘learns’ from the training examples by generating predictions and adjusting for the correct response iteratively. While supervised learning models are more accurate than unsupervised learning models, they need manual data labeling before training. For instance, in the flood detection problem, a supervised learning model can predict whether or not there is a flood based on sample training data. This creates two significant challenges for the scalability and

generalizability of our solution approach.

To start, we want to make the device portable, which means that the detection should work on images (places/environments) that the model has never seen before. Second, the time and resources required to label the data are considerable. As a result, we experiment with less data hogging and more flexible deep learning architectures for unsupervised or semi-supervised learning. On the other hand, Un-supervised learning models operate autonomously to discover the inherent structure of unlabeled data. They still require some level of human intervention to validate output, but far less than supervised learning. Our experiments in the supervised setting know that deep supervised learning performs well when large amounts of label data are available. However, there is a massive amount of unlabeled data in the real world. Every day, FloodBot generates hundreds of videos requiring supervised learning labeling.

In the later part of Chapter 5 we explored the possibility of limiting our need to manually annotate the data or evaluating how to reduce the need for significant annotation using semi-supervision techniques. Self-supervised learning is a variation of unsupervised learning, where we exploit unlabeled data to obtain labels. There are no explicit annotations or class labels associated with the data. We use unlabeled data to get some labels and induce a supervised learning model on unlabeled data. Specifically, we design supervised tasks called pretext or auxiliary tasks, which can learn meaningful representations [72]. Thus, the model becomes ready to solve a downstream task such as a classification or semantic segmentation or any other

supervised learning task.

Therefore, we aim to understand the work in a semi-supervised setting and increase FloodBot’s vision parsing capability to generalize more. Specifically we looked at Radford, et al.[7] new visual concept called Contrastive Language-Image Pre-training (CLIP). CLIP builds on previous work on zero-shot transfer, natural language supervision, and multimodal learning. Because of our mutli-modal data we found CLIP’s technique of jointly training an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples relevant to our work.

In Chapter 5 we also experimented with zero-shot learning, which is expected to generalize to unseen object categories in image classification. After analyzing different state-of-the-art works in contrastive learning, we based our experiment on similarity search after Schroff et al.’s Siamese network design [59]. Similar to their model FaceNet [125], which directly learns a mapping from face images to a compact Euclidean space. We assess the similarity of our flood images (positive and negative pairings) using the similarity index calculated via metric learning [153]. Metric learning compares the proximity of two vector space representations with the closer distance resulting in similar items and vice versa [152].

While vision-based flood detection has had success, the main issue arises when the system loses visibility. Low lighting combined with inclement weather can cause a loss of vision [52], which is frequently a precursor to heavy rain and flooding. As

a result, we propose using sound as a alternative signal to assess the likelihood of detecting floods. Furthermore, sound signals have a smaller data volume (byte) , reducing the footprint of each deployment. Next we look at literature and work that are related to us and guided us in designing sound based flood detection.

## 2.3 Environment Sound Analytics

We enable devices to make sense of their environment by analyzing the environment’s sound signals. This work is part of a larger investigation area from machine listening related to computational auditory scene analysis [33, 14]. Machine listening systems perform similar processing tasks to the human auditory system. They are part of a broader research theme that connects fields such as machine learning, robotics, and artificial intelligence [12].

Acoustic scene classification (ASC) is the task of assigning a semantic label to an audio stream that identifies its the environment from the sound recording [12]. Sawhney[124] proposed the first reported work in the ASC area in a 1997 technical report from the MIT Media Lab. ASC is also related to psychoacoustic/psychological studies aimed at understanding the human cognitive processes that enable humans to understand acoustic scenes [93]. However, the context of our work and this thesis is focused on computational auditory scene analysis (CASA) [33], a more focused algorithmic approach tries to identify the sources of acoustic events in a sound clipping and closely related to event detection, and classification techniques [47]. ASC

aims to identify and label temporal regions containing single events of a specific class. ASC have been employed in surveillance systems [117], and speech analysis through segmentation of acoustic scenes[69].

Sound event detection (SED) is a subfield of ASC that aims to automatically detect the occurrence of target sound events from audio signals capturing an acoustic scene, such as a vacuum cleaner, thunderstorm, birds sound, or a dog barking. The surge in publications in the Detection and Classification of Acoustic Scenes and Events [3, ?, 1] indicates an increase in interest in SED. Since its first competition in 2013, the field has seen many novel research ideas, focusing on deep learning-based ASC algorithms.

Most SED algorithms build upon deep neural networks, specifically the convolutional neural network (CNN). As input to the network, either fixed two-dimensional signal transformations such as Mel spectrograms [66], or raw one-dimensional audio samples are used (end-to-end learning) [170]. Until the recent uprise in deep learning, traditional methods Gaussian mixture model (GMM) [164] was used to classify/detect acoustic events. Other conventional methods were hidden Markov model proposed by [95] and support vector machine [116].

Chapter 6 describes our work in Environmental sound classification (ESC) a task under SED. The objective here is to determine what is occurring in an audio stream and when it occurs. In practice, the objective is to determine the temporal locations of distinct sounds within an audio source [96]. Such detection systems has

been successfully used in many practical applications[96, 158], such as robotic hearing, smart home[13], audio monitoring systems, soundscape assessment, etc. ESC is a more challenging research problem than the regular musical note detection or speech recognition problem [130]

So far, we have only discussed physical sensors (image + sound); however, we found that social sensing is equally relevant and robust in event detection during [16, 137]. We argue that social sensing data contain a wealth of information about spatial interaction and semantics that outperforms the detection capability of physical sensors [89]. Our deployment site is in a busy commercial area, so we could infer if a mass event is happening by mining social media posts. The analysis of social media content related to crises focuses primarily on data types such as images, geolocation, videos, text, and so on. However, the majority of this work has centered on images and geolocation in the context of crisis management. [76, 109, 70, 136].

When used properly, social media can be a goldmine of information. Social media analytics [165] needs a strong natural language processing (NLP) capability often provided by deep learning language models. In Chapter 7, we develop language models and aid the Floodbot’s physical sensing capability with social sensing capability through natural language processes. We describe one such experiment where FloodBot is perceived as a conversational AI agent. We coined the term **FloodBot** after infusing the social sensing capability to otherwise just physical sensor (image + sound).

## 2.4 Social Media

Designing and implementing chatbots is not a straightforward task. It involves a specific architectural design and an appropriate goal/task-oriented algorithm with the domain-specific dataset [35]. Chatbot customization demands and development technologies are evolving rapidly with current industrial needs. There have been various recent enhancements, new features, and in-depth studies that are reported in this domain [6]. We carefully selected to review certain relevant work in this area of chatbot design. Well-designed and executed chatbots play a crucial role in the real world and could be an essential tool to improve user engagement and experience between humans and the served domain overall. FloodBot is designed to converse and inform about the situation remotely.

The implementation of ChatBot highly depends on its knowledge bank, also known as corpus. The content of the information that a chatBot is trained on provides knowledge and context to the ChatBot for conversation. A successful chatBot implementation needs tonnes of datasets and standardized evaluation metrics to track and compare their performance [63]. However, collecting data to train data-driven dialogue systems is difficult and time-consuming. Here, we list some of the famous corpora used to build chatbot. *The WikiQA* [163], *Question Generation as a Competitive Undergraduate Course Project data* by [138], *Ubuntu Dialogue Corpus* [91], *MultiWOZ* [34], *Cornell Movie-Dialogues Corpus* [45], *Stanford Question Answering Dataset (SQuAD)* [114] and *bAbI* [155].

We experimented with multiple non domain-specific Question and Answering corpus (Q&A) from the list above. After performing the complexity and relevancy analysis on these datasets, we used Facebook’s bAbI-QA [155] for question answering for our work in Chapter 7, and text understanding as our seed corpus. This dataset comprises a set of contexts, with multiple question-answer pairs available based on the contexts. We infused more data from physical sensor readings into the same dataset and in the same format.

FloodBot is a type of chatbot envisioned to respond naturally to human queries. Though FloodBot is uniquely designed to converse around potential hazards around flood-prone areas, it uses similar components as any other chatbot. FloodBot is a task-oriented Chatbots are built to accomplish specific tasks like making restaurant reservations [31, 73, 140]. Most goal-oriented bot like ours are implemented using a modular system, where the bot often has access to an external database on which to inquire about information to accomplish the task [55]. Some other groundbreaking work in the dialogue system is proposed [140, 148, 126, 150, 115, 155].

The primary problem in all categories covered previously has been developing a unified model that examines signals (image, sound, and social media) in a single location or situation. Somehow, research is deepening in each domain but fragmented. Therefore, we try to test multi-modal data and model development to harness the full potential of the deep learning data domain. This hinders the use of deep learning in delivering end-to-end problem solutions. Therefore, in Chapter 4, 6 and 7 and we tried to implement these together and propose a joint multi-modal

learning technique to solve a real-world problem.

In Chapters 4 and 5, we incorporate work from the field of image analytics; in Chapter 6, we incorporate work from the audio analysis. Chapter 7 then discusses textual and social media-related projects. We presented studies in this chapter that motivated us to conduct experiments in multiple modalities. After examining numerous successful works in the field of deep learning [42, 58, 121, 82, 11, 75] from a variety of disciplines, we employ these concepts to develop our flood detection system.

## Chapter 3: Deployment

This chapter’s primary objective is to present FloodBot’s system design, iterations, and subsequent learning and enhancements. We began our study in 2017 [15] and have tested and refined our design ever since. We upgraded the technology from a standalone physical sensor to a social media-enabled FloodBot [22]. To reach there, we constructed multiple prototype devices and collected real-time data to deliver advanced alerts for citizen safety.

One of the primary contributions of this thesis is to develop a viable cyber-physical system capable of remotely monitoring/sensing the flooding situation. Once we have reliable and most viable product (MVP) in place, we can extend our systems to serve as a community watchdog, for example, through social media integration. This chapter takes a chronological look at those design attempts and deployments. We then discuss the complexities and challenges inherent in deploying disaster risk reduction IoT systems, a.k.a FloodBot. We document the complexities and challenges of establishing a fully functional and dependable IoT-based flood detection system (FlodBot) for the community.

### 3.1 Ideation

In the early stages of our research, we spent a significant amount of time trying to build a simple hardware/software platform for flood detection. We carried out several micro-experiments and designed boards and circuit-level flood detection systems. These experiments helped us discover about the challenges and fragility of these devices, as well as their limitations in being able to work in a challenging environment.

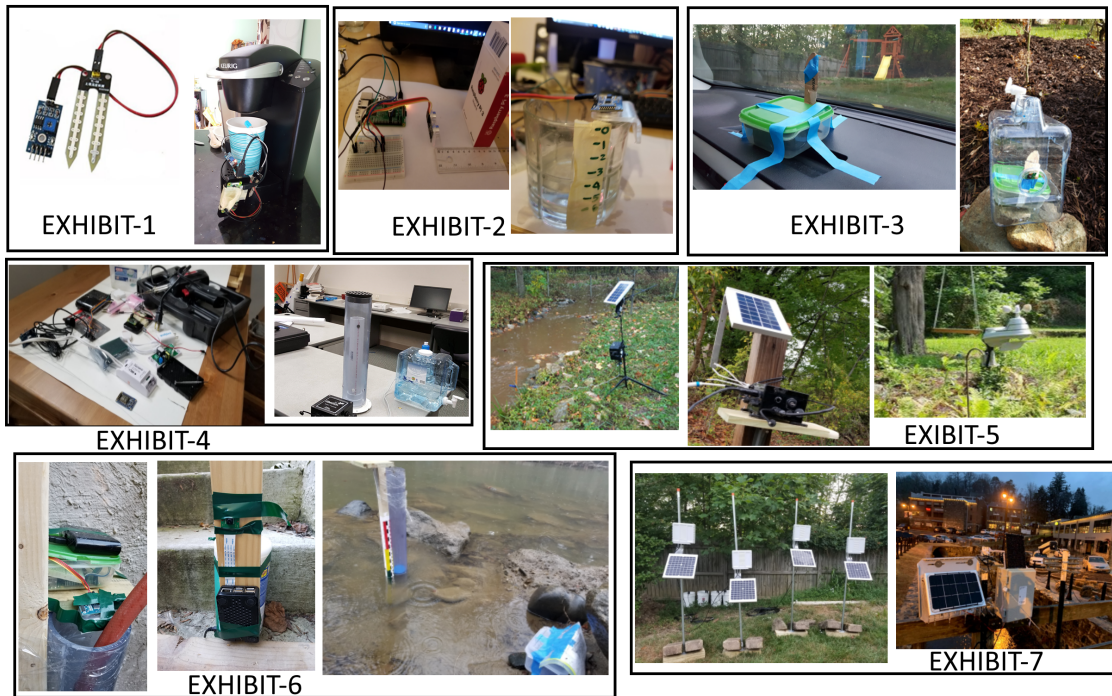


Figure 3.1: Ideation

The units needed to be waterproof whereas severe rain and extreme weather integral part of our deployment. This possessed a major challenge around the scalability. Further, there was always a difficulty with the power supply. The cost of pow-

ering and reusing the system (solar + batteries) would outweigh the sensor/board and computing cost. We show some of those micro experiments in Figure 3.1 and list out their main capability, sensors used and the objective etc. in Table 3.1.

Table 3.1: Ideation & Experiments

Experiment	Capability	Sensor	Objective
Exhibit-1	Senses the moisture and sends email	Moisture Sensor	Server side Technology & Automation
Exhibit-2	Measures the Depth of Liquid	Ultrasonic Sensor	Notify when water level has reached threshold
Exhibit -3	Image Capture	Camera	Weather Monitoring
Exhibit-4	Networking & Data Transport	Wifi, Zigbee,Float	Flood Detect & underground sensing
Exhibit-5	Monitor & Alert Water Depth	Gen-1	Detect water threshold & notify
Exhibit-6	Vision Based Detection	Gen-2	Scene Text Recognition
Exhibit-7	Vision,Sound & Text Based Detection	Gen-3	End to End System-FloodBot

Nonetheless, we revisit these units to remind ourselves of their challenges and the lessons we learned from these deployments. Following the ideation phase, it became clear that we should focus our research on either hardware and its stability or software and its optimization. We decided to dive deep into software development and analytics and more toward advanced analytics (Deep Learning).

### 3.2 Prototype & Generations

From the ideation phase deployments and trials, we discovered that the most time-consuming and crucial aspect of IoT development is its reliability, physical stability, and data quality [38]. To that end, we evaluated the design, development, and deployment of those Internet of things (IoT) systems in severe environments and with a high degree of heterogeneity. We further put the FloodBot’s IoT components through a stress test to determine their reliability in severe conditions.

In subsequent sections, we describe the deployment specifics, their architectures, and the problems and motives that led us to iterate into the next generation. We refer to our past deployments and their ecosystem as their ‘Generations’ to represent different genre of hardware and software. Each deployment had its own set of benefits and drawbacks. We gradually acquired knowledge, corrected errors, and continued to the next stage. We experimented with alternative technologies and underlying processing units (genre) and proceed into the next generation, starting from on-premise server-based computing, to edge computing, and then to the cloud computing. Even though there were other deployments and experiments we only focus on the three most significant iterations (generations).

### 3.2.1 General System Architecture

Our deployments are all based on the similar technology and architecture. They follow traditional client-server architecture, clients are Environmental Monitoring Unit (EMU) deployed on flood detection site and remote server Decision Support System (DSS).

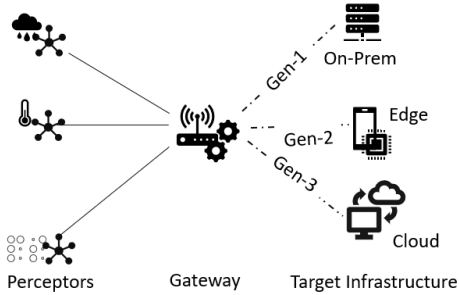


Figure 3.2: General System Architecture

They all use a three-tiered Internet of things architecture consisting of a perception layer, a network/gateway layer, and an application layer [113]. These layers are depicted in Figure 3.2.

Although the intra-device communica-

tion protocols, IoT units, and target infrastructure slightly change in each generation (Gen) deployment, their overall data flow and operation remain identical. The data endpoint for Gen-1, Gen-2, and Gen-3 deployments is an on-premise server, an edge computing unit, and the cloud, respectively.

### 3.2.2 Gen-1 On-Prem FloodBot

This was the first deployment outside our lab and first real world deployment. Our initial Waspnote Plug & Sense Unit deployment was based on ‘On-Premise IoT Architecture’. The Waspnote Plug & Sense included an inbuilt SD (Secure Digital) card with a maximum storage capacity of 2 GB. The solar panel charged the unit’s battery.

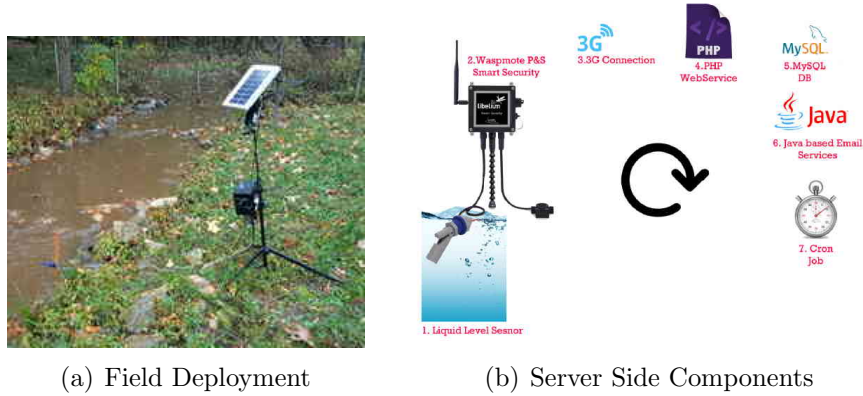


Figure 3.3: Gen-1 Deployment

This system was deployed in May 2018 and remained operational throughout the summer and December 2018. We relocated the equipment and evaluated its agility and reliability under different scenarios, locations, and weather conditions.

This deployment provided abundant learning opportunities. Table 3.2 summarizes the capability of Gen-1 and its technology stacks.

Table 3.2: Gen-1 IoT Deployment Summary

Gen-1 On-Prem IoT	
Network Protocol	3G
Perceptor	Flood Level Threshold Sensor
The application layer	Custom Code (LAMP, JAVA)
Major Capability	Binary Data and Email Trigger

Once we were able to operate the device, it was adequately reliable. The server performed the critical function of this unit, and hence the unit itself is less prone to failure at the IoT node.

#### *i) Capabilities*

This unit’s primary feature is to send a binary trigger to the server when it crosses the unit’s predefined flood threshold. The circuit is closed when the water level reaches the mechanical float mechanism, and the unit transmits a binary signal. The signal then triggered a sequence of server-side applications, including database inserts, SMS alerts, and email notifications.

#### *ii) Limitations*

Next we list limitations observed in this deployment.

- *Data Limitations:* This device is only capable of sending a binary signal when the preset flood threshold is reached.

- *Limited Predictive Capability:* Real-time flash flood detection requires a

predictive model which can forecast future flood level in the area. However, the data limitation capability of this system does not accommodate the predictive power well. This deployment would be able to create at most a *Rule Based* engine.

- *Cost:* Given a limited data ability for our application, the device is also cost-prohibitive. The equipment cost us almost two thousand dollars.

- *International Device:* Being an imported device connecting to the American network service providers is a challenge.

### *iii) Lessons Learned*

It took a long time and many attempts to get Gen-1 to work. Gen-1 was the first time we deployed and tested a cyber-physical system outside a controlled laboratory. We needed to ensure that the device was safe and not prone to physical damage or theft, which is counter intuitive for our research and the remote monitoring. Although the device was marketed as plug-and-play, it required extensive specialized coding to function. We were required to research and comprehend a wide range of hardware, software, and networking components proprietary to the vendor [86].

We based our second-generation (Gen-2) deployment on our previous experience with Gen-1 deployment. We built an in-house lab unit for this iteration and attempted to address the issues seen in Gen-1. In addition, we experimented with a cost-effective solution and more robust data.

### 3.2.3 Gen-2 Edge-Computing FloodBot

Our objective for Gen -2 was to automate the process of flash flood detection and categorize flood levels using computer vision and image processing techniques. We required a time-variant image capture system that could store and track rising water levels in a stream. Therefore, we developed an end-to-end in-house prototype and validated its performance in a challenging environment.

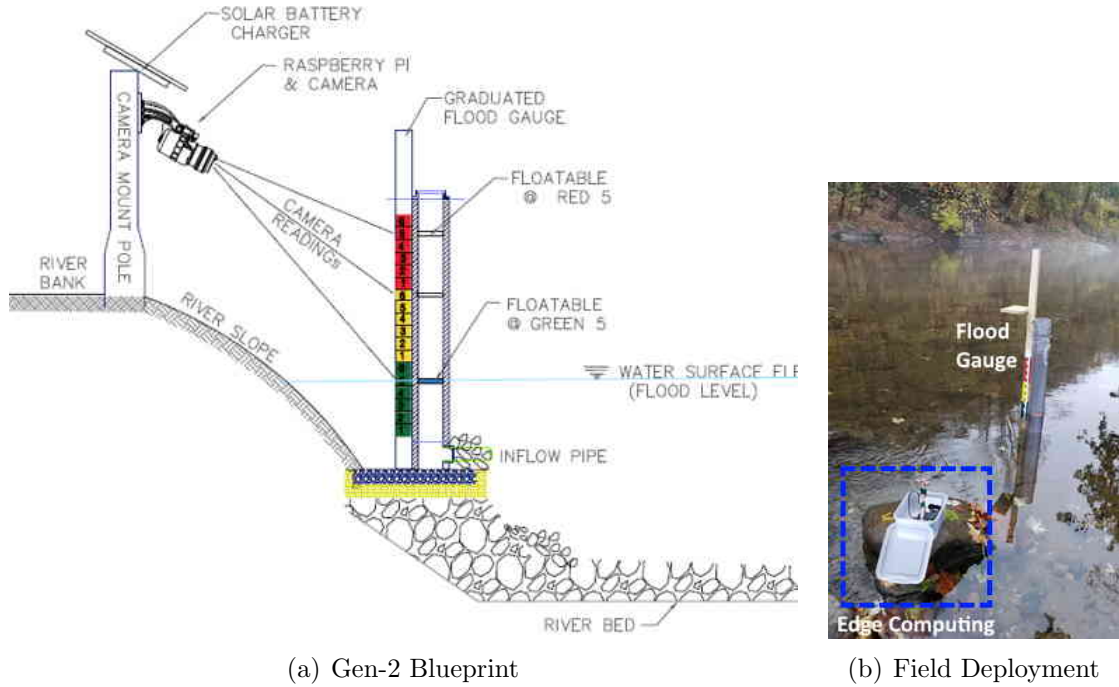


Figure 3.4: Gen-2 Deployment

For this, we devised two distinct units located adjacent to one another: the physical unit - riser structure, and the computational unit - edge computer. The physical unit/riser structure is a flood gauge (ruler-like) color-coded labels. The computation unit/edge computer is a solar-powered Raspberry Pi [54] with a camera. The edge device captures the images and stores them for post-processing.

### *i) Capabilities*

The guiding principle for this deployment is based on scene text recognition (STR)[131]. Scene text recognition is the task of recognizing character sequences in natural scenes. The microcomputer (Raspberry Pi) and camera are powered by a battery connected to a solar cell. The camera takes pictures at preset time intervals. The edge computer(Raspberry Pi) was provisioned with of three primary components/modules. The first module in the pipeline is the image Acquisition unit, the second module is the image pre-processing unit, and the third is the machine learning unit.

### **a) Image Acquisition Unit**

In comparison to other fields of computer vision research, such as traffic sign detection or address identification, our flood detection task lacked vast amounts of labeled data and flood detection systems. As a result, we had to create our training data meticulously by deploying the unit for an extended period and labeling the data. To do this, we developed a color-coded calibrated scale for the Flood Gauge/Riser Structure and collected training data over three months. Next we describe the components of the system and their capabilities.

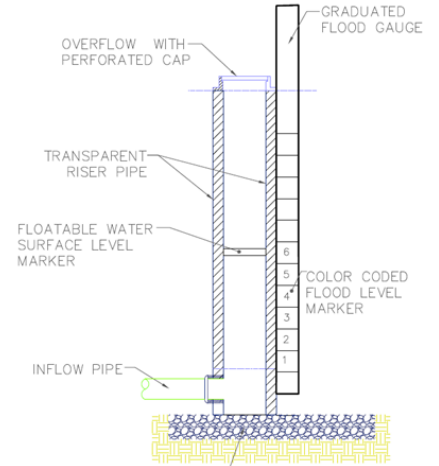


Figure 3.5: Water Level Riser

- *Water Level Riser Structure* The Water Level Riser Structure was constructed entirely from components obtained at a local hardware store. The setup and components level detail is shown in Figure 3.5. This unit comprises a clear hollow riser (6-inch High density pipe), an inflow pipe (2-inch PVC) , a floatable water marker (floating pool noodles), and a top emergency spillway (perforated riser cap). The riser's bottom has an inflow pipe from where water enters the system. The floatable inside the riser structure, rises with depth of stream. The second critical component of this accessory is the blue-colored floatable. The floatable water level indicator used to observe rise and fall of the water level in the riser. A Graduated Flood Gauge is attached to the Water Level Riser Structure. This is a color-coded Water Level Scale with three color-coded regions and a 1-inch demarcation. The scale used in this prototype can measure Water-surface difference up to 18-inches.

#### ***b) Edge Computing***

For edge computing we used following three components. Their capabilities and uses are described accordingly.

- *Camera and Micro Computer* The most crucial task is capturing time-variant images of rising/receding water levels in the stream, so we used a raspberry pi a micro-computer and its camera as portable image capturing unit. The raspberry pi and a camera takes a time-lapse picture with one minute interval.

- *Software Stack* The Micro Computer (Raspberry Pi) runs on Linux Operating system called Debian. Images and the meta-data are transmitted and stored in the Linux based Centos 6 server with Apache, MySQL and PHP installed in our lab.

- *Server Side Processing* Field deployed Raspberry Pi transmits the data periodically to the server via HTTP push. Then our final process is to automate the flash flood categorization task. We implemented popular clustering techniques K-Means to separate the most dominant colors. We then used digit recognition techniques to identify the flood level in a stream. The capabilities of Gen-2 is summarized in Table 3.3.

Table 3.3: Gen-2 IoT Deployment Summary

Gen-2 Edge Computing IoT	
Networking	On Device Storage
Perceptor	Camera Unit
The application layer	Pre-Trained Deep Learning Model
Major Capability	Real time Flood Detection on Edge

## *ii) Limitations*

This device is constructed locally using commodity hardware (pipes, gauges, overflow structures) and software (computing models, image data storage, and processing). It presented unique problems and learning opportunities. Some of the significant challenges and shortcomings are highlighted below.

- *Stability:* The major challenge in this design and deployment is the physical stability of the unit. Figure 5.19(b) shows the precarious physical stability of this deployment.

- *Reliability:* Lose wire connections, dangling parts, and stand-alone camera reduced the reliability of this unit.

*Safety:* The units will also be hard to deploy in harsh flooding or unforeseeable

scenarios. It would be easily swept away during flooding and misplaced.

- *Accessibility:* The camera is unable to capture clear images in extreme/foggy weather and at night.

This study was centered on the challenging field of computer vision and presented us with excellent research prospects. Apart from the shortcomings mentioned previously, the primary limitation of this equipment is its inability to detect floods at night. The computer vision technique described above relies on the camera's ability to determine color and digits, and so the device would be unable to determine flood level at night.

### *iii) Lessons Learned*

Through this deployment, we validated the fundamentals of cyber-physical device development and understood its sophistication. The construction of this complete package comprises hardware (physical devices), IoT sensor circuit components, and deep learning. The project depended on various engineering disciplines, including mechanical/civil, electronics, and software.

Here, we also explored on how to automate flash flood detection via scene-text recognition technique. Though the setup was prototypical, we laid a foundation for future expansion into this work. With the challenges addressed, this deployment proved that image could be a reliable source for flood detection. We perfected our networking, data transfer, and some pre-cursory work needed in future deployments. Much of current FloodBot design ideas are expansion of the in-house flood detector (Gen-2).

We sought to address concerns identified in Gen-1 and Gen-2 with our third generation deployment -Gen-3. With a solid understanding on the constraints imposed by past deployments, we can now envision a fully functional system. The overall setup and deployment of this system, as well as the sensors, are depicted in Figure 3.6.

### 3.2.4 Gen-3 & Beyond

Gen-3 FloodBot is envisioned to be a cyber-physical device capable of both physical and social sensing [89]. The term sensing implies two distinct characteristics of the data. First one is generated by the our hardware/software components such as water level reading, video captures, weather data etc. [165].

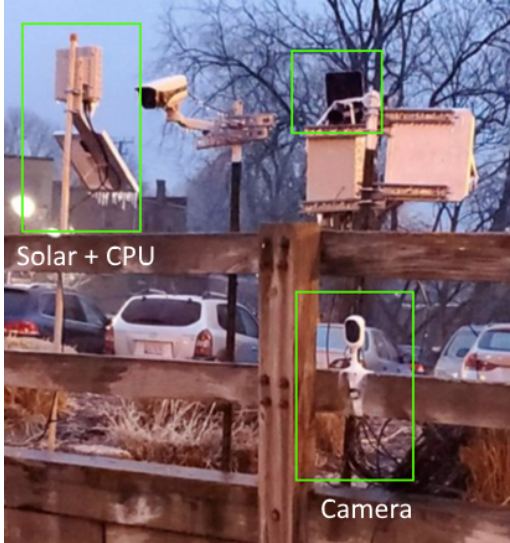


Figure 3.6: Typical FloodBot Setup

Second, where human act as a sensor by creating social interaction data (often in social media) when they either consume/curate the flood related data to propagate further into ts in social media. We also envision the social media integrated social sensing capability as contemporary (cool) need for FloodBot. When a disaster strikes, people often seek information from social media platforms,

and post the contents back into social media. FloodBot is expected to live and play

active role in community via its enhanced communication and data propagation capability.

Beyond contemporary, the device should be durable, economically viable, and stable. Thus, our build and deployments under during Gen-3 provides significant contributions to this thesis. As we encountered physical damages and washing away of our underwater sensors, we inclined our research more towards the over-the-air (contact less) sensor modalities.

We examined the current trend of using and interacting with technology today and attempted to simulate flood detection accordingly. Rather than constructing a standard data gathering and wireless sensor network, we envision FloodBot as a state of the art community member, contributing to the creation and consumption of knowledge in the same way that humans do. As a result, we equipped Gen-3 FloodBot with cutting-edge features such as AI-Conversation [137, 20, 18].

### ***i) Capability***

The setup for Gen-3 FloodBot is depicted in Figure 3.6. Gen-3 deployments consists of two types of sensors. The first sensor is a pressure-based water level indicator, while the second is a camera unit. Several such sensors are located along the Ellicott City streams. The installation is based on a native client-server architecture. Each node consists of camera(s), solar-powered CPU with a network gateway, and other ambient sensors, including a water level sensor (at the bottom/under flowing water).

- *State of the art:* The sensor technology, water proof enclosures, reliable network components and embedded system makes this sensor state of the art.

- *Redundancy:* We deploy more than one unit in one location (Figure 3.6) i.e. more than one sensors (often 2 cameras, devices) to create redundancy and collect multi-modal data.

- *Interfacing:* Unit provides two easy ways of data interfacing. We can either use application programming interface (API) to read the raw data reading or web interfaces to download them. Table 3.4 shows capabilities of Gen-3.

Table 3.4: Gen-3 IoT Deployment Summary

Gen-3 Cloud Computing IoT	
Network	Cloud Connected 3G
Perceptor	hydro-static level sensor
The application layer	Cloud hosted API
Major Capability	Water Level, Images and Social Media Posting

## ii) Limitations

FloodBot [145] is an online and live system operating in Howard County, Maryland, since 2018. We have gained substantial understanding of the unit’s functionality and associated issues. While we are always learning through the live Gen-3, the following are some of the problems we have faced thus far.

- *Initial Setup:* It took several months initially to get the device operational due to internal hardware failures and network connectivity issues and even bureaucratic challenges (fund, permissions etc).

- *Proprietary Device:* The device and its perceptor (CPU and other units) are all proprietary to the vendor, therefore we have limited access, visibility and modification capabilities to the unit's internal operation.

- *Stability:* Flash flood event(s), repeatedly washed away few of our preceptors (pressure transducer); thus, this unit is also susceptible to physical stability, specifically to the underwater 'contact' sensor units.

- *Clogging & Debris:* The sensor unit is submerged in the water and often gets clogged by siltation and other debris. The debris also causes connectivity issues and data loss.

### *iii) Lessons Learned*

As with prior deployments, Gen-3 also provided abundant learning. We notice that even the advanced, complex systems as Gen-3, some of the fundamental underlying issues with WSN still remains. For instance, the washing away of our Gen 3 sensor, vandalism, and network connectivity reliability are all ongoing difficulties to any outdoor wireless networking system or cyber-physical system. However, we believe that Gen-3 deployment is a major milestone for us and will provide possibilities for multi-faceted research directions. The unit is completely integrated with a camera, enabling remote viewing and validation of the data.

### 3.2.5 System Evaluation

We score the systems and their success based on evaluation criteria discussed by Fahmideh et al. and Maoling and et al. [51, 162]. Based on their criteria, we selected four evaluation metrics to assess the quality of our deployment generations.

- *Performance* to measure the device capacity from parallelism, their ability to query the unit from both multiple user interfaces and Operating Systems.

- *Modifiability* to measure the ability and flexibility to make changes from both hardware and software in the deployment.

- *Reliability* to measure the overall reliability of the system.

- *Availability* to measure the capability of usage and execution of the software developed during intervals of time. The results are shown in Table 3.5.

Table 3.5: Evaluation Metrics

Evaluation Report Card				
IOT	<i>Performance</i>	<i>Modifiability</i>	<i>Reliability</i>	<i>Availability</i>
Gen-1	Low	Low	High	High
Gen-2	Medium	High	Low	Limited
Gen-3	High	Low	High	High

## 3.3 Conclusion

Despite complex communication networks, Gen-1 deployment was feasible for the execution time duration. Since the Gen-1 was mainly a plug-and-play device, it

was less adaptive to changing requirements for scalability and customization. Gen-1 performance was limited due to its inability to support numerous users or distributed environments.

The availability of the Gen-2 deployment was limited, as it was developed and installed in a precarious state, not suitable for harsh environmental/weather conditions. The overall performance of the Gen-2 system was average, as the data collected and analyzed with this system were not up to par due to the device's physical instability, insufficient image data, and lack of a night vision camera.

Thus far, the Gen-3 deployment appears to be promising compared to prior deployments. We have real-time access to the data via the API call and dashboard. The sensors and equipment are very efficient and connected to the internet. The Gen-3 system's performance is superior to that of the previous generations due to the better data quality, system stability, network connectivity, and data complexity. Rest of the thesis now only discussed the data collection and describes our work performed under Gen-3. We show various deployment locations and pertinent components of Gen-3 sensors in Figure 3.7 .



(a) Deployment Location-1



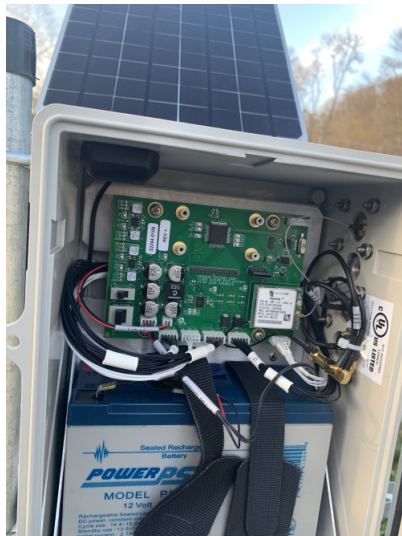
(b) Deployment Location-2



(c) Deployment Location-3



(d) Deployment Location-4



(e) Gen-3 - CPU



(f) Gen-3 - Water Sensor

Figure 3.7: Gen-3 - Deployment

## Chapter 4: Data Collection

This chapter’s primary objective is to document our data collection efforts, motivation, and community impact. First, we present the context of our work and its relevance to everyday real-life and life-threatening situations. We then discuss our collaborative community activities and highlight ongoing work in this area that fosters sustainability of our work beyond this thesis. This thesis gave us a unique opportunity to engage with the community and consider how to effectively solve a real-world problem. Our study began as a result of the flood and subsequent damage to the community in July 2016 [118, 156, 159].



Figure 4.1: July 2016 Flood Damage: Ellicott City

While our study was still in the ideation, the town was affected by a second flood in May 2018 [157, 160]. These disasters wreaked devastation in the area and resulted in property loss. Figures 4.1 and 4.2 depict the damage caused by these two floods.

The flood mostly devastated a historic commercial district, costing several

small company owners their lives and forcing them to flee town or close their doors [119].



Figure 4.2: May 2018 Flood Damage: Ellicott City

Therefore, we envision our research to be scalable, long-term, and have a lasting influence on the community. We worked with various community members and took a holistic approach to the flood detection problem rather than limiting ourselves to a lab prototype(research). In the next section of this chapter, we describe our such multiple collaborations. These collaborations are ongoing aspects of our research and community impact that will continue beyond this thesis.

## 4.1 Community Collaborations

Following the flood devastation in the area, UMBC/Mobile Pervasive & Sensor Computing Lab (our research team) partnered with the Howard County Department of Public Works and their hardware vendors [68, 112] .

In addition, the Howard County Department of Public Works [67] provided us with potential flood spots all across the county. Owing to the two floods that wreaked havoc in the Old Ellicott City neighborhood, we concentrated our data collection efforts in that area. Figure 4.3 shows the entire county and their identified flood prone locations (red dots). The purple highlighted area in Figure 4.3 depicts

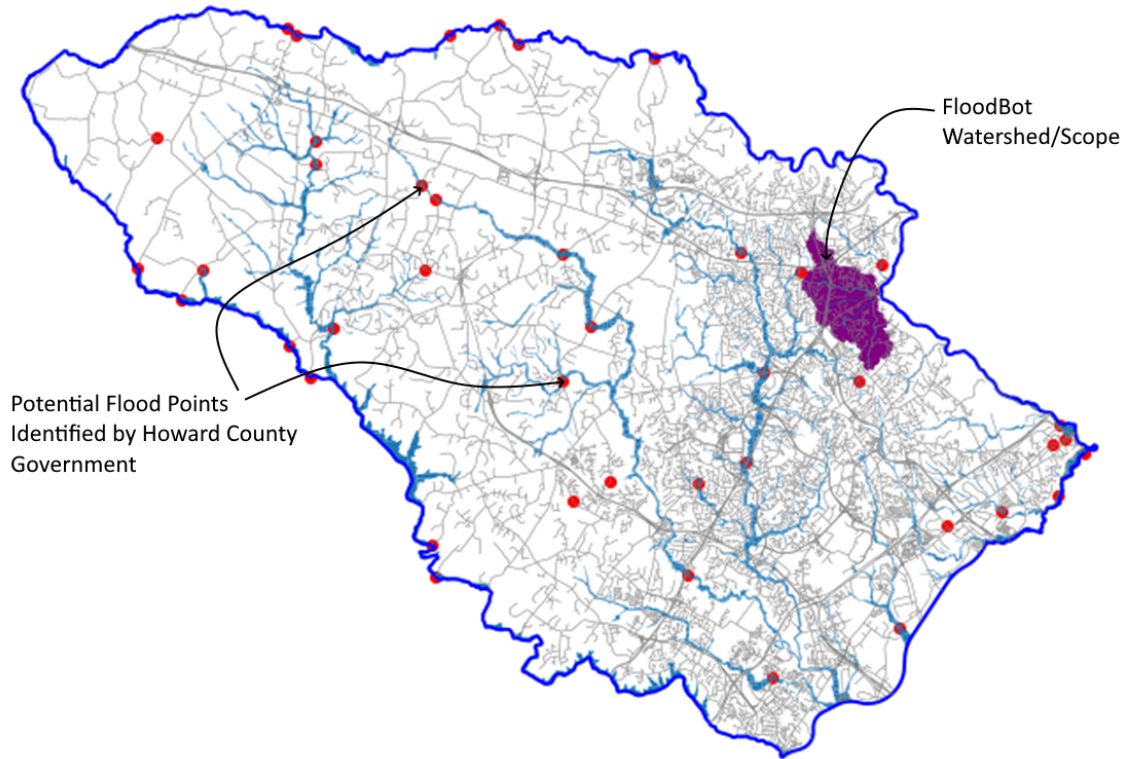


Figure 4.3: FloodBot-Watershed/Scope Area

the scope of deployment and flood detection researched as part of this thesis.

This is an ongoing collaborative project between UMBC, Howard County, and other hardware-software companies initially supported through the Department of Homeland Security’s Flood Mitigation Program [48] and National Science Foundation (NSF) research grant [104]. The locations of FloodBot’s data accessibility, sensor nodes within purple area above (Howard County Maryland and Old Ellicott City area) are shown in Figure 4.4. We also independently bought sensors from another vendor Evigia [50] and deployed at the same location to facilitate cross-company/device resilience research and built redundancy.



Figure 4.4: FloodBot-Deployment Map

Our research and deployments have contributed to the emergence of an entirely new and noble study direction. In addition to the scope of this thesis, we also experiment with various techniques for integrating the cyber-physical system with social media [133, 134, 135, 136]. The readings and images from the device are automatically tweeted using the `umbc_floodbot` [145] Twitter handle.

## 4.2 Data Collection

As described in Chapter 1.2 and depicted in Figure 1.2 on page 6, this thesis relies on three data modalities for image analytics, sound analytics and social media. This section describes how we enrich and annotate multi-modal data using a sensor. We describe the sensor data, image data, sound data, and then the textual (social media) data and their use in our research.

### 4.2.1 Sensor Data

This thesis began as a physical sensor investigation into effective flood sensor methods. In Chapter 3.1, we discussed our deployment and ideation experiments. After evaluating several in-house sensor systems, we discovered that sensor design is a separate field of study. As a result, we collected water level data using cloud-connected “off the shelf” sensors already integrated in Gen-3 Deployment. Gen3-FloodBot has a pneumatic pressure sensor to measure the water-depth (flood level) depicted in Figure 4.5.



Figure 4.5: FloodBot - Sensor

The sensor is a pressure sensor (also called pressure transducers or pneumatic pressure sensors). Pneumatic pressure sensor is used to measure and monitor compressed air/gas pressure levels in a system using a variety of electronic devices. These pressure sensors are transducers, which provide an electrical signal proportional to the measured output pressure. The transducer(sensor) is connected to the CPU for data storage and transmis-

sion. It performs liquid level measurement by having the sensor submerged under the water surface. The pressure sensor measures the water’s equivalent hydrostatic

pressure above the sensor, converted into liquid depth.

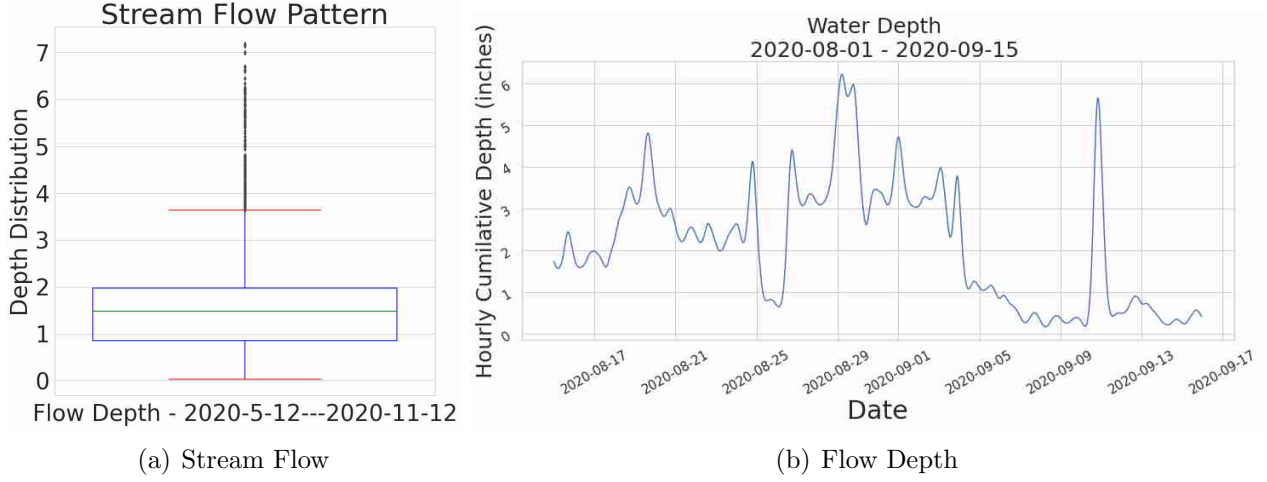


Figure 4.6: FloodBot Water Depth

We have collected depth of water in the stream through out the deployment period. Here we show some general statistics around the depth recorded during our deployment. We show sample reading from one month (2020-05-12 to 2020-09-24) in Figure 4.6 .

## Data-smoothing

The sensors' depth readings are too erratic to be used as a signal. Except in the case of a sudden flood, the water level gradually increases or recedes. As a result, the early spikes and peaks are primarily due to sensor anomalies. We smooth sensor readings with a linear filter while maintaining the depth trend. We employ the Hodrick-Prescott (HP) filter, which is frequently used to smooth time series. Hodrick-Prescott is a method for estimating a time series' cyclical and trend components. The function computes the time series' cyclical and movement components

using a frequency cut-off or smoothing value. We filter the raw water level reading using the smoothness parameter  $\lambda = 100$ . The effect of the HP filter on the readings of our water level from one of the month is shown in figure 4.7.

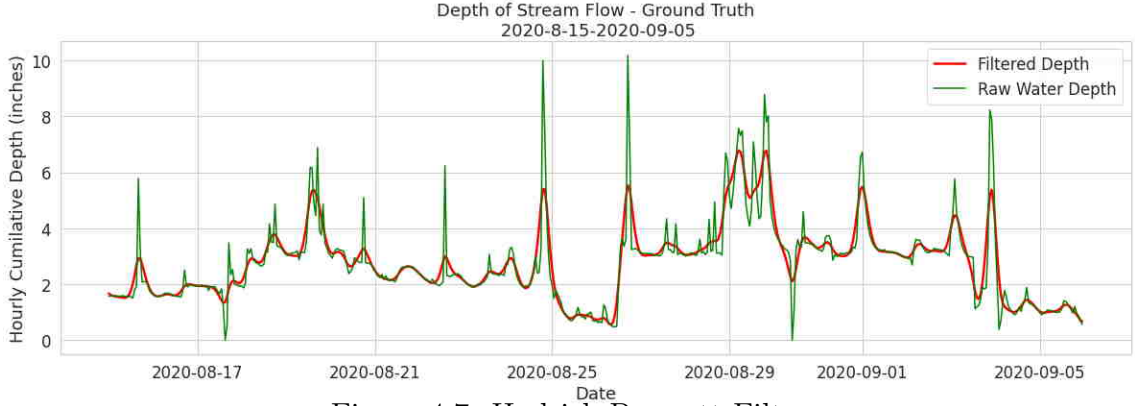


Figure 4.7: Hodrick-Prescott Filter

#### 4.2.2 Audio Visual Data

The FloodBot implementation starts with real-time images acquisition. These images are transferred into the cloud for pre-processing required in any computer vision pipeline. We automate the image pre-processing using Open CV [32] and Python. The Flood Image database contains images captured in various weather conditions for more than two years. Figure 4.8 shows the sample images captured during various ambient conditions from the testbed.

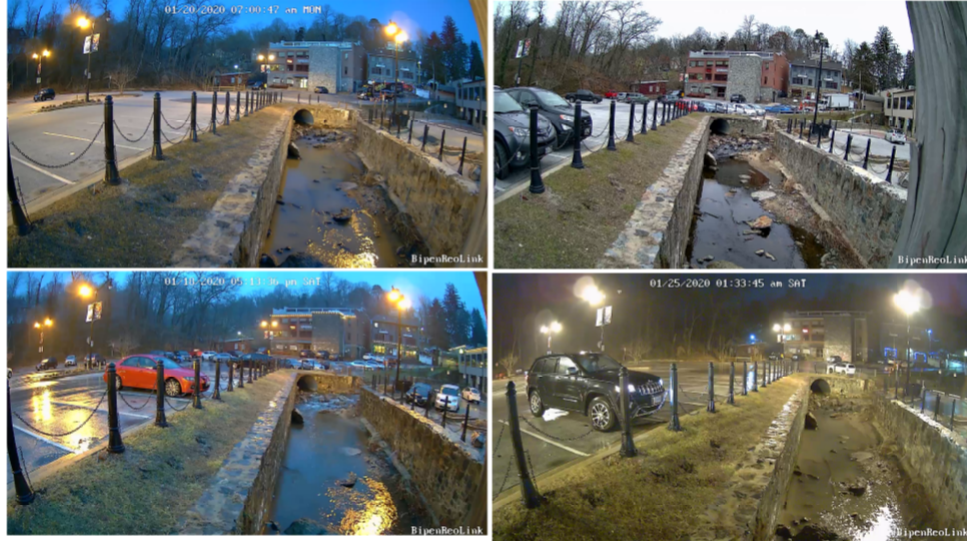


Figure 4.8: Flood-Watch Camera

We collected thousand of such images from the each FloodBot camera node [68, 112]. We programmed the FloodBot to use a smart file naming convention. As depicted in Figure 4.9, the filenames of images are stored using a standard date YYYY-MM-DD-HH-MM-SEC. This naming convention gives us two valuable pieces of information: one-it makes sure the images are unique (time base constraint). We can imperatively see the day and time of that image, another important aspect for temporal data parsing need.



Figure 4.9: Sample Images Captured by FloodBot

Since temporal information is preserved in the filename, we can easily connect other data elements using temporal join. We also have a video camera deployed in the same location. We can extract image frames from the video for the image-to-image comparison and audio signals from the same video to use it as another flood detection signal.

### 4.2.3 Weather Data

We also collected the weather data from a public weather application interface during the same deployment time frame. We find the weather in the area at the time we captured those images. The weather data API allows us to extract weather at that particular location based on our camera’s latitude and longitude. We collect and store these records by minutes in our database. From the weather API, we extract many ambient weather elements.

Day	FileName	Img_Time	temperature	humidity	windSpeed	visibility
5/16/2020	20200516152104.jpg	5/16/2020 15:21	54.29	0.87	7.79	10
6/24/2020	20200624012310.jpg	6/24/2020 1:23	72.77	0.85	6.44	10
10/11/2020	20201011080816.jpg	10/11/2020 8:08	51.31	0.89	7.56	10
10/4/2020	20201004220847.jpg	10/4/2020 22:08	53.84	0.7	5.8	10

These include a *summary*, *icon*, *Precipitation Intensity*, *Precipitation Probability*, *temperature*, *Apparent Temperature*, *Dew Point*, *humidity*, *pressure*, *wind speed*, *wind gust*, *wind bearing*, *cloud cover*, *UV index*, *visibility*, *ozone*, *Precipitation Type*, *Precipitation accumulation*. FloodBot is provisioned with real-time

weather integration. Our current FloodBot Twitter [145] automatically tweets the weather condition every 6 hours using that weather data.

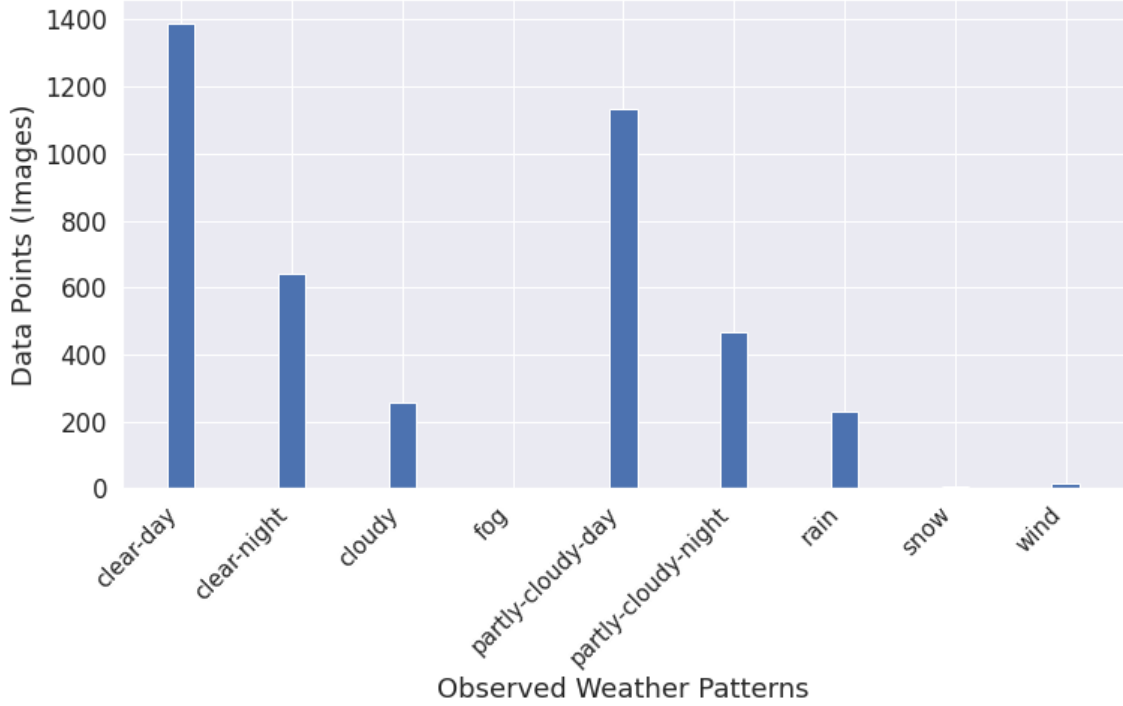


Figure 4.10: Weather Data

### 4.3 Multimodal Temporal Data Fusion

Temporal data are sequences of a single type of data, most commonly numerical or categorical values, and multivariate or composite information. We seek to extract implicit, non-trivial, and potentially useful information from our various data sets by mining temporal data (image, sound, weather and social media). Therefore, during the data collection we take extract precaution to save all the timestamps or perform time based annotation (eg. image frame naming convention using frame time).

Table 4.1: Multimodal Data

Parameter	Values
Weather_Time	10/29/2020 16:00
FileName	20201029160625.jpg
Img_Time_Stamp	10/29/2020 16:06
pressure_water_level	2.78
dpth_trend	7.18978
time	1603987200
summary	Rain
icon	rain
precipIntensity	0.0534
precipProbability	0.71
temperature	43.4
apparentTemperature	40.22
dewPoint	43.31
humidity	1
pressure	1008.1
windSpeed	5.35
windGust	11.39
windBearing	117
cloudCover	0.99
uvIndex	2
visibility	7.424
ozone	260
precipType	rain
precipAccumulation	0

As a result, we take extra precautions during data collection to save all timestamps or perform time-based annotation (e.g., image frame naming convention using frame time).

Once the data is collected, we use a temporal join to bring all of these heterogeneous sources together (image captured time, social media posting time, weather timestamp, depth recorded timestamp, and so

on) and store it in our database for machine learning. With all of the data points stitched together, we now have multidimensional data that can represent the site condition and flood situation in greater detail. Figure 4.11 shows our sample temporal multi-modal data. The following chapters use temporally fused data to discover temporal patterns and regularities across modalities, beginning with image, then sound, and finally social media. The temporal data stitching techniques enable our thesis’s next set of data explorations.

### 4.3.1 AI-Ready Data Release

Research only matters if it is seen and used, and open-access publishing has been shown to increase readership and citations. We strongly believe that many researchers can build insights from our work’s underlying data. Furthermore, collecting data, formulating models and writing code are resource-intensive. We hope our research continues beyond this thesis and encourage future researchers to spend time pressing research questions rather than redundant work on a model or dataset development. Therefore, we have released all our AI - Ready data collection and relevant code in the public domain and made them accessible in GitHub [28].

#### *i) Raw Data*

Our data collection is extensive and versatile, and it may be applied to a variety of experimental settings, not just flood detection. We show the monthly data availability in Table 4.2. As part of this thesis, we are sharing a **3831** video recording that lasts **29 hours**. Since FloodBot captured the video, we have static image frames and audio snippets. Additionally, we provide the code for extracting the visuals and audio elements included in video frames to do various experiments. There are *1,553,370*

Table 4.2: Raw Data

Month	Video
202001	355
202002	735
202003	700
202004	513
202005	387
202008	4
202009	2
202010	12
202011	41
202012	29
202101	34
202102	15
202103	5
202104	14
202108	7
202109	29
202110	21
202111	230
202112	342
202201	214
202202	141

static image frames to be extracted from the  $1072\ W \times 1920\ H$  video recording. Similarly,  $1,638,525$  extractable audio frames are recorded at  $127313$  maximum bit rate. In the background, the recording captures varied weather patterns, extreme stream conditions, and exciting urban events such as weddings, social gatherings, and concerts.

## *ii) Annotated Data*

During our research, we experimented with the data extensively and published it at various venues. To that end, the data is richly annotated and ready for the next level of analysis and problem formations. We show two sample annotations in Table 4.3 and in Table 4.4.

Table 4.3: Classification & Detection

Identification		Classification	Detection	
Video	Frame_Id	Flood Level	Person	Vehicle
202001181134	202001181134f_0.jpg	No Flood	0	10
202001181136	202001181136f_0.jpg	No Flood	0-2	10
202001190018	202001190018f_3.jpg	No Flood	0	5-10
202001191101	202001191101f_4.jpg	No Flood	0-2	10

- *Classification and Detection*

Table 4.3 shows the annotation primarily used for our Image Analytics. We use the annotated data to verify model accuracy. The video frames are annotated (Identified) with their timestamp and labeled accordingly. For example,  $V\_f\_i$  would represent the time in minutes, the frame, and the frame number from that F. We find this nomenclature very informative to have a temporal sense of data during analysis and scene reconnaissances. The following two sets of columns are our class labels

where we can frame the problem either into multi-class classification  $\{Flood, No\ Flood, Severe\ Flood\}$ . The last column is our annotation, where we have estimated the count of objects in range. Note that the FloodBot captures images from the parking lot; therefore, the actual counts are not possible. We use these ranges to verify the risk associated with persons or vehicles in a broader sense and use object detection to validate these counts.

- *Contextual Information*

Table 4.4 shows descriptive annotation primarily used during environmental sound analysis and support the social media curation. These data have been beneficial in generating the Tweets and annotating the sound clippings when we assume that the vision-based system has failed or is not performing well.

Table 4.4: Contextual Information

Identification	Context			Natural Language
Video	Visibility	Weather	Time	Description
202001181134	Clear	Cloudy	Morning	calm and quite morning time.
202001181136	Clear	Cloudy	Morning	Flood Image is blocked by wooden post.
202001190018	Night	Cloudy	Night	Street lights are on. pretty empty parking lot
202001191101	Clear	Sunny	Morning	parking full of cars. people standing by the stream

Out of the total raw **29 hours** of video recorded, we have annotated 20 hours of video recordings that translate into *1,031,314* of annotated image frames. Furthermore, all of these image frames are temporally joined with weather data to their **5 minutes** time binning; hence we can see the temporal weather pattern and

stream flood variation with a minor detail (our thesis) or formulate other urban data analytics problems.

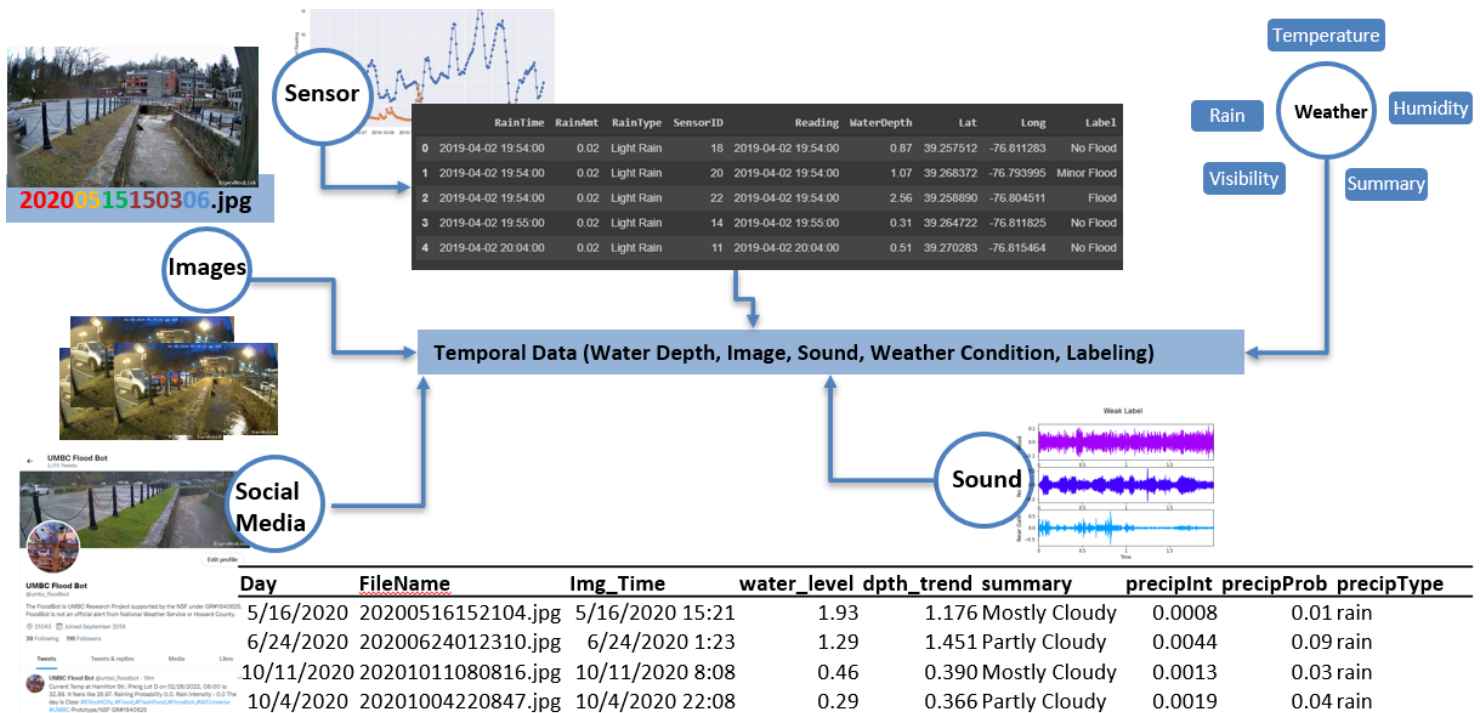


Figure 4.11: Multi Modal Temporal Data Fusion

## Chapter 5: Image Analytics

Over the last decade, there has been tremendous success in the field of computer vision [121]. Computer vision output has been successfully applied in various real-world applications [78, 44]. We can use computer vision to develop a complete information propagation and processing system, such as FloodBot. As a result, we leverage the power of computer vision models to detect objects and classify flood severity in images. This chapter details our experiments and analytics performed in FloodBot’s first data modality (images).

We first start the experiments, as a supervised learning where we are given a pair of the dataset, feature vectors, and corresponding labels. We pose our problem as a multi-class classification problem and divide the input images into three classes ‘*No Flood*’, ‘*Minor Flood*’, and ‘*Flood*’ conditions based on the stream flow conditions observed in the images. We start with traditional image analysis and shallow learning for flood detection before progressing into more complex (deep learning) methods. The goal of the classifier is to find the best model that can map a feature to its corresponding label. We are given the task of building a machine learning system that will classify them into one of their classes. Such supervised classifiers

require labeled data (feature/label pairs) to train.

## Supervised learning

In supervised learning, we are given a data set  $\{x_i, y_i\}_{i=1}^N$  where  $X$  and  $Y$  are their feature and label spaces. We usually assume  $X$  is in a high dimensional space  $x_i \in R^d$  &  $y_i \in R^c$ ;  $C = \{1, \dots, M\}$ . The label space are comparatively smaller and finite as compared to the feature space.

### SUPERVISED LEARNING

- **Problem Definition**

Given a set of images:  $P_{(FloodBot\ Images)} P = \{x_i, y_i\}_i^N$  Find:  $f(\cdot)$

that maps  $f^*(\cdot) = X \rightarrow \mathcal{Y} \in C\}_0^K$

- **Goal**

$$f^*(\cdot) = \operatorname{argmin}_{h(\cdot) \in \mathcal{H}} \frac{1}{|D_{TR}|} \sum_{(\mathbf{x}, y) \in D_{TR}} \ell(\mathbf{x}, y | h(\cdot)),$$

- **Approach**

*Cross Entropy Loss*  $\mathcal{L}$  :

$$\mathcal{L}(\hat{y}, y) = \frac{1}{N} \sum_1^N y_i \cdot \log(\hat{y}_i)$$

Here, we assume the labels are in a finite field. For instance, we can approach our research problem of identifying a flooding stream as a binary classification problem where  $y_i \in \{0, 1\}$  or in a multi-class classification setting, we assume the labels are  $y_i \in \{C\}$ . Alternatively, we can also define the supervised learning problem as

a conditional probability distribution of two sample spaces  $P(y|x)$ .

$$f_w : R^d \rightarrow R; f_w(x_i) \approx y_i \quad \forall i \quad (5.1)$$

## 5.1 Shallow Learning

Until the recent upsurge of the deep learning-based solution, image analysis and feature understanding were primarily done using traditional computer vision [121, 24]. Traditional image processing approach involves interconnected steps such as segmentation, feature extraction and classification often done manually [62]. In certain circumstances, deep learning is unnecessary, as traditional Computer Vision (CV) algorithms may handle a problem far more efficiently and with fewer computational resources. Deep learning in computer vision tasks requires a large amount of data, frequently millions of records. For example, the PASCAL VOC [49], Dataset has 500K images with 20 object categories, ImageNet [121], has 1.5 million images with 1000 object categories, and Microsoft Common Objects in Context (COCO) [87] has 2.5 million photos with 91 object categories. Even though deep learning might be the go-to tool for computer vision these days, We believe that traditional image analysis is still valuable.

Traditional approaches can be used when data is not large or high computational power are unavailable. Deep learning requires considerable computing power, time, precision, input qualities, and quantity, among other things, which are not

always readily available. As a result, we first explore flood detection problem using traditional image analysis. We implement computer vision tasks using hand-crafted global features extraction from FloodBot captured images and evaluate them using traditional shallow supervised learning techniques. We treat our problem in its simplest form (shallow learning) and begin with a supervised learning frame work where we are given a a pair of labeled dataset.

Formally, the problem can be therefore be stated as:

Given observations  $\mathcal{X} = \{x_1, \dots, x_N\} \subseteq R^R$  and Labels  $\mathcal{Y}^* = \{y_1^*, \dots, y_N^*\} \subseteq R^R$  of a specific flood category such that  $y_n^*$  represents the ground truth flooding condition of  $x_n$ . Therefore, the classifier’s task is to learn a mapping function  $f_w$  such that  $f_w$  is able to map  $(x_n \mapsto y_n^*)$  and generalize to previously unseen observations. Here, we assume the observations  $x_n$  and their labels  $y_n^*$  to be vectors in some high-dimensional space  $R^R$ .

The observations  $x_n$  corresponds to a spatio-temporal data representation or a single data instance (image and other features) captured by FloodBot. For the ground truth labeling  $\mathcal{Y}$  we experiment with various automated class labels and also manually annotate the validation set. More specifically,  $Y \in C = \{FloodClass\}$ . We have separated training images explicitly therefore, the model has partial learning of the full dataset.

### 5.1.1 Feature Extraction

The term “feature extraction” refers to the process of transforming raw data into numerical features that can be processed while retaining the dataset’s information. Also, raw data are often unusable by machine learning models. Extracted feature are believed to produce better and accurate results than directly applying machine learning to raw data.

Image feature extraction can be performed on two levels: local and global. Global features describe the image as a whole to generalize the entire object, whereas local features describe image patches (critical points in the image). Local feature extraction compares critical points (detectors) in the same image using, e.g., affine covariant features, a difference of Gaussian, etc. Some of the commonly used local feature descriptors are SIFT (Scale Invariant Feature Transform), SURF (Speeded Up Robust Features), ORB (Oriented Fast and Rotated BRIEF), BRIEF (Binary Robust Independent Elementary Features) etc.

Some examples of global descriptors are shape Matrices, Invariant Moments (Hu), Histogram Oriented Gradients (HOG), and Co-HOG, etc. These are real-valued numbers (integers, float, or binary). Local features describe the image patches (critical points in the image) of an object. We chose global descriptor color, texture and shape to use as feature for the classification task.

## Color

Based on the heuristic image analysis, we noticed that our images show similarities during different times and weather conditions, including heavy rain. Therefore we decided to use the color histogram as one of our features. Though the color histogram technique is a straightforward and low-level method, it has shown promising results in practice for image indexing and retrieval tasks [105].

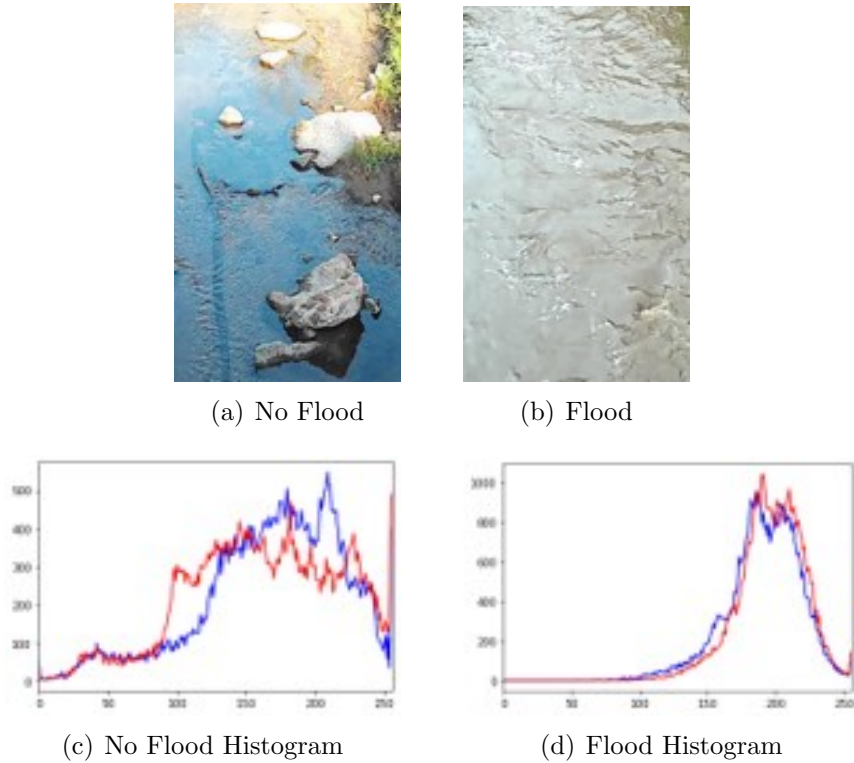


Figure 5.1: Color Histogram as Image Feature

A color is represented by a three-dimensional vector corresponding to a color space position, the hue-saturation-value (HSV) and red-green-blue (RGB). We chose an RGB color space. We show one sample image from our dataset in Fig-

ure 5.1. It is apparent in Figure 5.1 that the distribution histogram for the “No Flood” image is wider, with pixel intensity picking around 200. In the case of the Flooded images, the histogram range is less and peaks around 225.

## Texture

In image processing, the texture is a function of spatial variation of the pixels’ brightness intensity. The texture is the primary term used to define objects or concepts of a given image [80]. A surface is a set of texture elements or texels occurring in some regular or repeated pattern. The texture is considered as a statistical property where similar structures repeat [36]. Texture analysis plays a crucial role in computer vision cases such as object recognition, surface defect detection, pattern recognition, medical image analysis, etc.

Using texture as a feature allows us to understand the statistical and structural difference of pixel alignments in an image. We see a pattern in our images to our images into their classes such as Flood and No-Flood. The image texture gives us useful information about the image’s content, the objects in the image, the background context, back-

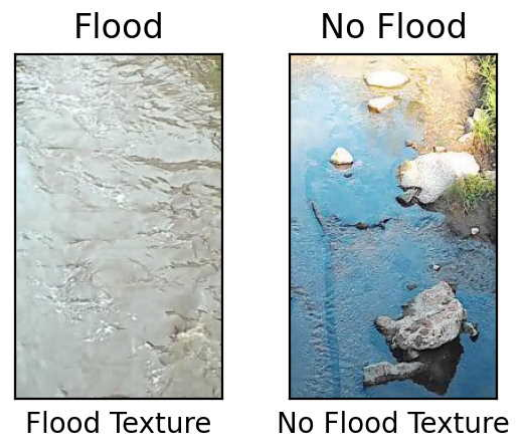


Figure 5.2: Base Image

ground, and so on. E.g., when the stream is in a low-flow condition, the riverbed and stones are visible, creating a more textured surface, whereas when the stream is completely flooded. This example shown in Figure 5.2 . We use structural (Local Binary Pattern) and Statistical (Co-occurrence Matrix, Energy Entropy) as our feature extractor. The statistical methods use spot localization of pixel values. Statistical image analysis involves analyzing the texture of images performs a series of statistical calculations on the lightness intensity distribution functions of pixels [80].

### 5.1.2 Data Annotation

To annotate data we use the structural similarity index measure (SSIM) [151, 101] between base image and incoming image. The SSIM index evaluates a test image  $X$  with respect to a reference image  $Y$  to quantify their visual similarity. SSIM evaluates the quality of  $X$ , with respect to  $Y$ , by computing a local spatial index that is defined as follows.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (5.2)$$

Given an image, the goal of an image similarity feature is to find other “similar” images. We used similarity index to annotate our training data. We use two extreme image as our base image and annotate the test images based on their proximity to the base image. First we fixed the two base images (Flood and No Flood) and computed their SSIM value against the fixed all black pixel images of same size. All three images are shown in Figure 5.3.

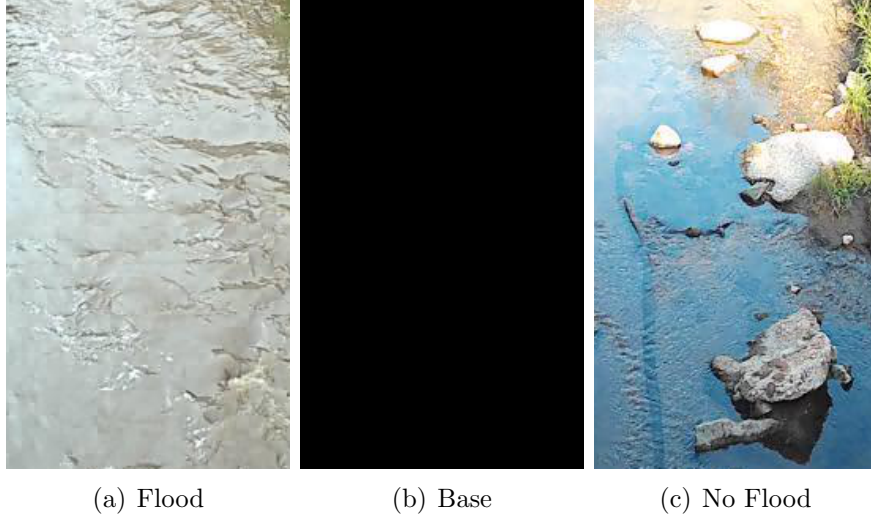


Figure 5.3: Images for SSIM Base Metrics

Here we use two images shown in Figure 5.2 to compare the similarity score using the mean squared error (MSE). The mean squared error (MSE) estimates the average squares of the errors—that is, the average squared difference between the estimated values and the actual value.

SSIM is used to compare two images, especially from their quality perspective. However, it can also be used as a metric to compare two images. To create our training-labeled dataset, we compared the incoming image vector similarity index with both of the images and used those features to label images into one of their class *flood* and *No Flood*. For example, the MSE produced by two flood images should be closer than their contrasting pairs. We show our example input dataset in table 5.1 below. Here  $\{\text{histogram, haralick moments, floodcomp, nofloodcomp}\} \in \{Y\}$ . As observed in the ‘Flood Base’ and ‘No Flood Base’ the SSIM scores are in range of 6000 and above for the non flooded stream images. All remaining images are then categorized similarly and our training data is generated. We save

Table 5.1: Sample Data: Supervised Learning

Histogram	haralick	moments	Flood Base	No Flood Base	label
1.88	7478.94	0.00	2144.86	7708.33	flood
2.14	6145.18	0.00	1592.86	6132.83	flood
2.14	8940.25	0.00	7287.06	4711.10	no_flood
1.75	4135.11	0.00	8904.78	5433.42	no_flood
1.82	4898.44	0.00	2708.60	2918.53	no_flood
4.52	7082.86	0.00	2112.21	3205.62	no_flood

manually annotated images for validation.

### 5.1.3 Shallow Learning

In this section, we walk through three shallow learning used to classify flood . We labeled our dataset into one of the classes as described earlier. We then split the dataset in training and test set with 70% and 30% of examples of each, respectively. With the datasets ready, we can run machine learning such as *LogisticRegression*, *K-nearest neighbors*, *Linear Discriminant Analysis*, *Gaussian Naive Bayes* and *Support Vector Machine*.

#### i) K-Nearest Neighbour(KNN)

KNN classifies data points based on the most similar points. It is non-parametric, i.e., it does not assume data to come from a normal distribution. The model is made up entirely of the data given to it. The algorithm works by finding the distance between the vector distance between points. KNN computes the distance between each data point and the test data. It then finds the probability of these points being similar to the test data and classifies it based on which points share the highest probabilities.

## ii) Logistic Regression(LR)

Logistic regression (LR) is a statistical method similar to linear regression. Usually, LR finds an equation that predicts an outcome for a binary variable,  $Y$ , from one or more response variables,  $X$ . However, unlike linear regression, the response variables can be categorical or continuous, as the model does not strictly require continuous data. To predict the output class, LR uses the log odds ratio rather than probabilities and an iterative maximum likelihood method rather than least squares to fit the final model.

## iii) Naive Bayes (NB)

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. This extension of naive Bayes is called Gaussian Naive Bayes. We use Gaussian distribution because data can be reconstructed using only two-parameters  $\mu$  and  $\sigma$ . LDA is another supervised learning technique that computes the directions i.e., linear discriminant that will represent the axes that maximize the separation between multiple classes.

## 5.2 Deep Learning

Low-level image features discussed here (shallow learning) have been shown to be effective representations for a variety of high-level visual recognition tasks, including object and scene classification. However, as visual recognition tasks become more difficult, the semantic gap between low-level feature representation and their

capability to represent/understand scene widens. The low performance of shallow learning is often termed as ‘Semantic Gap’ [167] unable to be closed due to the deficiency of shallow models. The term ‘semantic gap’ is widely used to refer to the disconnect between image representations (information stored in raw pixel) and image recognition objectives (detect/classify object in image). Shallow learning fails to generalize and the learning of these models is sub-par for our problem of flood detection. Therefore we need to find a model that has a better learning algorithm or help us close the semantic gap. In next section we experiment with various forms of DNN into models specifically the CNN.

Surveillance systems like FloodBot generate abundance of images, and videos. The development of search applications and algorithms to perform semantic analysis on these images and videos can provide more relevant search results and summaries and often beyond capacity of shallow learning.

Deep learning methods have grown in popularity as a result of their ability to outperform traditional (i.e. shallow) machine learning methods and extract features from raw data with little or no preprocessing [83]. Convolutional neural networks (CNNs), a type of deep neural network, have demonstrated significant performance gains in the area of object detection and scene classification [167] specifically closing the semantic gap. Convolutional neural networks (CNNs) [78] is particularly well suited for image pattern recognition, the precursory for classification [81, 102].

Rather than employing a single model and increasing its complexity, we divide

our problem (scene parsing/understanding) into individual components. The variants of the deep learning model used in our image analytics are depicted in Figure 5.4. We investigate the concept of scene classification in this work by performing two types of high-level image recognition tasks. First, we examine image classification techniques and categorize the flood using deep learning (CNN) patterns using (V-M1).

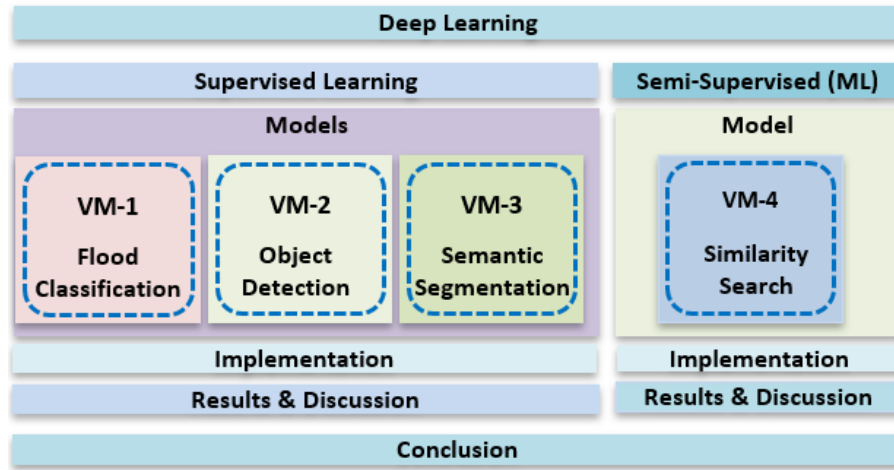


Figure 5.4: Deep Learning Models

Following that, we employ various types of feature learning to detect the presence of objects in the scene (V-M2). We combine the two elements(classification and detection) to serve as FloodBot’s eye on flood severity and objects in harm’s way. We then add semantic segmentation to our DNN model set to understand the vicinity and proximity (V-M3).

Deep learning can perform better at learning the data representation than shallow learning however supervised learning needs a large amount annotation data. Extracting features from the image is a resource intensive task and subjective. Get-

ting good-quality training data is another challenge in machine learning because data labeling is tedious. Therefore, we explore reducing the need of annotation by framing our problem as a semi-supervised learning and look at Semi-Supervised metric learning (VM-4).

### 5.2.1 V-M1 Flood Classification

The goal of this work is to adapt a CNN network to classify different types of flooding. Traditional flood classification based on hydraulic models lacks scaling and generalization [39, 137, 75]. For flood categorization, we propose a customized CNN network. Rather than defining a set of features by hand, we created a fully automated neural-based machine learning system for extracting discriminative features from training data and perform classification. Our method is easily adaptable to a variety of other fields, not just flood classification. Furthermore, because flood images lack distinguishing visual patterns and we have relatively small data, we adapted the standard CNN design to address these challenges. To avoid over-fitting, we used data re-sampling and random neural node drop-out to reduce the number of parameters in the CNN model. To enable flexible experimentation, we used Google Colab Platform to implement various neural networks.

This is our domain-specific Flood Categorization Model. In order to classify the current flooding situation at the site. This is supervised learning based on annotated data developed in for training image dataset were provided using the

current rainfall intensity .

### 5.2.2 V-M2 Object Detection

For complete image understanding, just classifying different images is not enough; instead, we need to estimate the concepts and locations of objects contained in that image. Object detection is a branch of computer vision and image processing that is used to detect instances of semantic objects of a particular class (such as humans, buildings, or car) in digital images and videos. Object detection enables the identification and localization of items in an image or video. We can identify, localize and count count objects in a scene, determine and track their specific locations, and precisely label them

Object detection is the process of identifying the instance of the class to which the object belongs and estimating its location by outputting the bounding box surrounding the object. Detecting a single instance of a class of objects in an image is called single class object detection, whereas detecting the classes of all objects in an image is called multi-class object detection. We developed a multi-class object detection model to detect objects in the images captured by FloodBot.

### 5.2.3 V-M3 Image Segmentation

Object detection identifies what is in the vicinity of the flooding stream. In our previous work, we answered whether a person or a car was in the flood vicinity. However, that work was unable to define the proximity of the object in the context of a flooding stream. Semantic segmentation helps determine the relations between objects and the context of things in an image. With this work and semantics segmentation of objects detected in the area, we can further enhance FloodBot’s power to quantify the risk metrics. For example, a car detected a few hundred feet away from the flood zone might be less risky than the one parked right next to the flooding stream. The pixelated semantic segmentation allows us to compute and infer the closeness of increasing water levels with the vicinity objects. Just like autonomous driving vehicle’s alarm system, our implementation can warn the authorities when the flood level is rising and imminent risk for nearby objects reaches some threshold.

Semantic segmentation/scene parsing has been an active research area of the computer vision community since early 2000. However, J. Long and et al’s [90] convolutional neural networks breakthrough work perform end-to-end segmentation of natural images is considered as the major milestone in the area. Semantic segmentation has been successfully used for applications such as face recognition [56], number plate identification, detecting road signs, and satellite image analysis.

Our previous work [18, 20] has shown FloodBot’s success in performing computer vision task of object detection and classification. There, we trained FloodBot

to identify objects in the vicinity and perform flood level classification using object detection, classification, and transfer learning. This work extends the flood detection technique to a more robust, sophisticated, and descriptive computer vision task of semantic segmentation.

Semantic segmentation helps determine the relations between objects and the context of objects in an image. With this work [19] and semantics segmentation of objects detected in the area, we can further enhance FloodBot’s power to quantify the risk metrics. For example, a car detected a few hundred feet away from the flood zone might be less risky than the one parked right next to the flooding stream. The pixelated semantic segmentation allows us to compute and infer the closeness of increasing water levels with the vicinity objects. Just as in the case of an autonomous driving vehicle’s alarm system, our implementation can warn the authorities when the flood level is increasing and imminent risk for nearby objects.

### 5.3 Model Architectures

In this section we describe model architecture of various DNN used as a part of our thesis. We designed two CNN models for classification: i) Custom: CNN where we designed and experimented with different layers of CNN and ii) pre-trained model (Mobilenet V2) for object detection and implemented image segmentation iii) Image Segmentator.

#### *i) Custom CNN: Classifier*



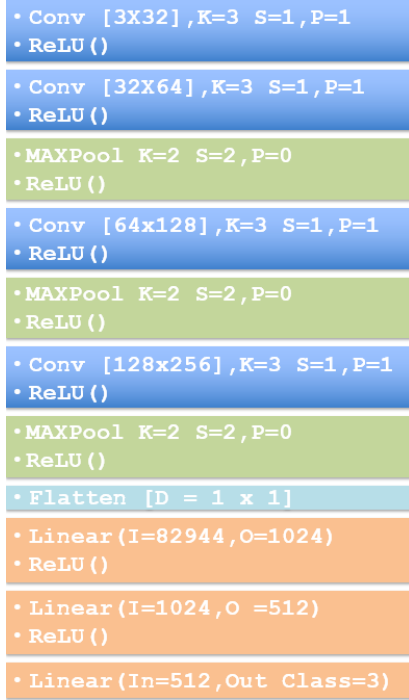


Figure 5.5: Layer wise Architecture V-M1

This gives us the same number of feature maps as input in the output, but these are much smaller. The pooling layer reduces the number of parameters and calculations in the network and improves the efficiency of the network, reducing over-fitting. We infuse non-linearity into our data by using one of the most common nonlinear functions called ReLU (Rectified Linear Units). It is defined by  $ReLU(x) = \max(0, x)$ . The ReLU correction layer replaces all negative values received as inputs by zeros and as an activation function.

Finally, the most commonly used layer is the fully connected layer. The objective of fully connected layer is to identify or detect the final output categories, Flood Classification in V-M1.

Here we summarize our V-M1 Hyper Parameters. We experiment with the following four hyperparameters:

- The number of filters  $K = 3 \times 3$ .
- The size F filters: each filter has dimensions  $F \times F \times D$  pixels.
- The S is Stride ( $S = 1$  move the window one pixel at a time ).
- The Zero-padding P:  $P=0$ . For each input image of size  $W \times H \times D$ , the pooling layer returns a matrix of dimensions  $WcHcDc$ , where: Choosing  $P=F-1/2$  and

$S=1$  gives feature maps of the same width and height as those received in the input.

We show the arrangement and high-level organization of the CNN layer in Figure 5.6 and the actual model parameters in Figure 5.5.

### *ii) Pre-Trained : CNN Classifier*

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [169]. Transfer learning has been used successfully to save resources and improve efficiency when training new models. Transfer learning expedites model training when limited labeled datasets are available, harnessing pre-trained learned parameters. In Deep learning, the first few layers are trained to identify features of the task. During transfer learning, you can remove last few layers of the trained network and retrain with fresh layers for target job. The process of keeping early layers and altering last few layers is called fine tuning. Fine-tuned learning experiments require a bit of learning, but they are still much faster than learning from scratch [98]. Additionally, they are more accurate compared to models trained from scratch.

Transfer learning, therefore, alleviates the need for a large set of labeled training data for every new model. Since few parameters need altering, many variations can be performed, improving the efficiency of machine learning development and easing deployment for multiple models. We used pre-train model [122] for transfer learning. MobileNetV2 is a light weight pre-trained convolution neural network

---

**Algorithm 1:** Cross Domain Transfer Learning

---

**Input:** Real Time Images from Flood Potential Site

**Output:** Flood Label: ***Flood, No Flood, Minor Flood***

- 1: Get Rainfall Intensity at  $t_i$
  - 2: Label training image  $\mathcal{Y}_i$  into one of the class-Labels based on rainfall intensity
  - 3: Start with the PreTrained Model(s) of choice
  - 4: Freeze the base Model and transfer learned parameters to custom head model
  - 5: Align output of the Base Model to required vector length
  - 6: Add a drop out and a fully connected (dense) layer + Softmax Classifier
  - 7: Validate Model Performance
  - 8: Save Model Weights for Real Time Image Classification
- 

and has been claimed to surpass its predecessors. We resized our input images to MobileNetV2 expected size of  $\{128 * 128 * 3\}$ .

The MobileNetV2 was trained on millions of images from the Imagenet[121] dataset. MobileNetV2 serves as a feature extractor for our classification model. Their base model consists of more than 100 layers and over 2.5 million parameters and trained over millions of images. We freeze the base model and add a new custom head for our classification. If we do not freeze the base model then the new run would recompute initial weight and parameters hence defeating the purpose of using such a powerful pre-trained model. The last layer of MobileNetV2 outputs  $\{4*4*1280\}$  tensor. The output size is not useful for our classification problem. We treat the output from MobileNetV2 as our source domain. MobileNetV2 provides a very efficient mobile-oriented model that can be used as a base for many visual recognition tasks [122].

Table 5.2: Cross Domain TL Model - V-M1

Layer (type)	Output Shape	Layer
MobileNetV2 Layers	..	..
out_relu (ReLU)	(4 x 4 * 1280)	Existing
Global Average Pooling	1280	New
Fully Connected Layer	3	New

We flatten the output to fit our classification problem. In order to flatten the output without losing the weights, we use global average layer. This layer functions similar to max pooling layer where the highest pixel values are transferred to the next layer. Instead, the Global average layer averages the weights there by maintaining the semantics of learned feature weights. Global average layer reduces the output from the base model to a vector size of 1280. We then add a drop out, and a fully connected layer to the base model, and create our prediction model for three classes. We use Softmax as an activation function. Table 5.2 summarizes addition of custom layers and their types. Algorithm 7 outlines our approach in using Cross Domain Transfer Learning.

### *iii) Object Detector*

Our object detection model Vision-Model 2 (VM-2) is based on a method proposed by Wei Liu et al. [88] called Single Shot Detection (SSD) for detecting objects in images using a single deep neural network. SSD allows us to detect multiple objects within an image with a single shot (one pass) instead of approaches based on regional proposal networks (RPNs). R-CNN series require two shots (two-pass): first for generating region proposals and second for detecting the objects in the proposal region. Thus, SSD is much faster compared with two-shot RPN-based

approaches [88].

The SSD architecture consists of a single convolutional network that learns to predict and classify bounding box locations in a single pass. As a result, SSD can be trained from end to end. The SSD network is comprised of a base architecture (MobileNet-V2) and a series of convolutional layers. SSD generates a fixed-size collection of bounding boxes and scores for the presence of object class instances within those boxes. This is a vanilla grade SSD implementation of the Single Shot Multi box Object detection used to detect objects in the vicinity. SSD generates scores for the presence of object category by a drawing and bounding box around the detected object. We randomly selected image frames and ran them through the V-M2.

#### *iv) Image Segmentator*

We use two types of deep learning model to perform the binary segmentation task of identifying the island. Our first deep learning model is a standard U-Net architecture model, based on U-Net: Convolutional Networks for Biomedical Image Segmentation [120]. Our second deep learning model is a variation of the U-Net model with added dropout for model correction. We also test and validate Flood-Bot's segmentation pipeline using pre-trained models such as FCN, Deeplab, and U-Net.

#### *a) U-Net Image Segmentor*

One of the most famous models in this class is U-Net. U-Net constructs the encoder-decoder structure for semantic segmentation. U-Net is an improvised ver-

sion of fully connected network (FCN)-based segmentation model [120]). Like the FCN, U-Net first creates convolutions and restores the image in the later part of the network, learning feature representation along the way.

U-Net [120] is a encoder-decoder type networks that are composed of two sub-networks: an encoder and a decoder. The encoder network’s objective is to extract features from an image and reduce the representation’s dimensionality. It is composed of a contracting path and an expansive path. The contracting path is typical of a convolutional network’s architecture. It is composed of two 3x3 convolutions (unpadded convolutions) that are applied repeatedly, each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for down sampling. The model doubles the number of feature channels with each downsampling step. Each step in the expansive path consists of an upsampling of the feature map, a 2x2 convolution (“up-convolution”) that halves the number of feature channels, concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. Cropping is necessary because each convolution results in the loss of border pixels. A 1x1 convolution is used in the final layer to map each 64-component feature vector to the desired number of classes. The network contains a total of 23 convolutional layers.

Our implementation closely relates to medical imaging [120] work where computer vision and deep learning model extracts clinically relevant information from medical images. The FloodBot, is deployed in a famous historic and shopping

district with lots of pedestrian activities and outdoor restaurant. The stream-our testbed, runs through this shopping district. To that end, we provide a qualitative analysis of the images captured by FloodBot, perform real-time semantic segmentation on them using today’s state of the art the deep learning semantic models to identify imminent risk.

Pre-trained models are excellent at segmenting objects that were part of their training set. However, there is no pre-trained object detection model or segmentation model for our specific problem: to detect stream rocks and walls or islands/landmass. To that end, we analyze images from the stream bed during and after rainfall to understand stream morphology using semantic segmentation. The computer vision and semantic segmentation of the stream morphology element such as boulders, rocks riverbed walls allow for real-time reconnaissance of flood level in the stream. They can be perceived as a spatiotemporal fingerprint of the stream bed before and after the flooding event. We can then use detection (presence or non) of some anchor objects to classify the flood condition.

## 5.4 Implementation

In this section, we describe our deployment detail and data collection process. The deployment of our system follows a native client-server architecture. The clients are cameras and other sensing nodes deployed on the ground. The server houses application and technology stacks with deep learning packages and infrastructure

for offline training and online inference, and a web and database server for message storage and propagation. The deployment locations are chosen based on their hydrological significance and hazard potential. The systems have been online since November 2019 and are collecting real-time flood data from the streams in Ellicott City, Maryland [22].

In this work we explored scene parsing pipeline for flood detection. We build, train, and validate image classification, detection and segmentation pipeline from the flood-prone stream. Our approach is practical, scalable, and portable for flood monitoring. For multimodal data fusion, we stitch heterogeneous multimodal data streams to create a better representation of flood causing the environment, thereby increasing the reliability of our deep learning model frameworks. This work can also create a framework for data amputation and anomaly detection. Our temporally connected multimodal data source enables missing data imputation and anomaly detection. FloodBot’s mission is to complete the inference from its segmentation models in near real-time. Therefore we explore the various state-of-the-art image segmentation models and assess them based on latency, accuracy, and suitability for our problem.

We have connected the data collection systems to the social media platform *Twitter*, which broadcasts the readings regularly via the Twitter handle *umbc\_FloodBot* [134]. The readings (Image and water depth) from the location are tweeted every six hours a day. This work and past tweets can be traced in Twitter using <sup>1</sup> [18, 17, 16].

---

<sup>1</sup>[https://twitter.com/umbc\\_FloodBot](https://twitter.com/umbc_FloodBot)

Some of the foundation work required for this work, such as deployment details and underlying system design, are discussed in our Chapter 3 and Chapter 4.

#### 5.4.0.1 Feature Extraction

We have collected the weather data synchronous to the images. Thus we programmed database queries and python package to label the image into one of the classes based on recorded rainfall intensity during the image frame timestamp. For example, a nice sunny day with no rain recorded would yield ‘No Flood.’ Light rain would cause minor floods. Significant rainfall would classify an image as ‘Flood’. We categorize the flood images into three classes **No Flood**, **Minor Flood** and **Flood**. The classes are based on turbulence, turbidity and observed high velocity of the flowing water predominantly caused by rainfall intensity.

In contrast to the shallow learning, we do not manually extract features from the image. The designed network extracts features automatically and learns their importance on the output by applying weights to its connections. We feed the raw image to the network, and as it passes through the network layers, the network identifies patterns within the image. Therefore, neural networks perform dual functionality of feature extractors and classifiers are end-to-end trainable, as opposed to traditional ML models that use handcrafted features.

#### 5.4.0.2 Data Annotation

We labeled 15,686 images in our training set. We used the automated rule based bulk Image annotation technique discussed above to label 322 images as ‘Flood’, 2440 as ‘minor flood’ and 13,125 as ‘No Flood’ images. This a highly imbalanced dataset so we re-sampled the images to balance the class distribution uniformly. To achieve a more balanced dataset we recreated the images from ‘flood’ class by copying, altering label data (up-sampling to 500 images) and randomly sampled images from other two classes (down sampling to 500 images).

One of the main challenge in machine learning is availability of labeled data set. Once we have a spatiotemporal data, labeling becomes somewhat trivial. Specially, we can label our training data in a a rule-based approach. We use two approaches in creating our label dataset for supervised learning. First we tried the rainfall intensity as our indicator to label the images then we use actual water level depth as our label to classify current flooding condition on site.

##### ***i) Rainfall Intensity Based-Image Annotation***

We programmed FloodBot to query the image database label the image into one of the classes based on recorded rainfall intensity during the image frame timestamp. For example, a nice sunny day with no rain recorded would yield ‘No Flood.’ Light rain would cause minor floods. Significant rainfall would classify an image as ‘Flood’. We categorize the flood images into three classes *No Flood*, *Minor Flood* and *Flood*.

---

**Algorithm 2:** Image Annotation-Rain Intensity

---

**Input:** Rainfall Intensity  $\mu_i, i$ , Image:  $\text{Img}_{ts}$   
**Output:** Image Class Label {No Flood, Minor Flood, Flood}  
**if** *Rain Int*  $i < \mu$  *Rain Int* **then**  
    | No Flood;  
**else if** *Rain Int*  $i > 80\% \mu$  *Rain Int*  $\& \text{Rain Int} \leq 90\% \mu$  *Rain Int* **then**  
    | Minor Flood;  
**else**  
    | Flood ;  
**end**

---

The classes are based on turbulence, turbidity and observed high velocity of the flowing water predominantly caused by rainfall intensity. The algorithm to create image, label pair is shown in algorithm 2.

**ii) Depth Based-Image Annotation**

For improved flood accuracy, we also looked at labeling images using depth recorded (filtered). The depth threshold were determined based on observer mean, median and their quartile range.

---

**Algorithm 3:** Image Annotation-Water Level

---

**Input:** Water Level  $d$ , Image:  $\text{Img}_{ts}$   
**Output:** Image Class Label {No Flood, Minor Flood, Flood}  
**if** *depth*  $< 2.25$  **then**  
    | No Flood;  
**else if** *depth*  $> 2.25$  *and* *Depth*  $\leq 4$  **then**  
    | Minor Flood;  
**else**  
    | Flood ;  
**end**

---

Once the data are annotated we are ready to perform machine learning and train/evaluate our classification model.

### 5.4.0.3 Deep learning Pipelines

This section describes our end-to-end architecture for orchestrating the input and output of data to and from a machine learning model (described earlier). The pipeline is composed of the following components: raw data input, feature extraction, output, the machine learning model and its parameters, and prediction output. Our thesis is centered on designing and implementing a deep learning pipeline. The multimodal data representation learning is attributed to various orchestrations that we implemented and tested. Following the physical sensor, the performance and effectiveness of FloodBot are primarily determined by the deep learning models and pipelines.

FloodBot is a real-world deployment and required the use of complex, multi-step pipelines. We examined each step, compared various libraries and run times, and devised (selected) specialized hardware for the pipeline’s construction. Design decisions have a significant impact on both the cost and performance our system. Therefore, the pipeline remains fairly similar and we experiment (alter) deep learning architectures.

#### *i) Classification V-M1 Detection V-M2*

To observe the current situation of potential flood areas and infer other contextual information, and converse about potential risk we used two deep vision based one for classification and one for the object detection. Two parallel deep learning vision models observe the flooding condition and scene. The first one is a Flood

detection model called **V-M1 (Vision Model1)**. The second model is a Single-shot Multi-box object detection model called **V-M2 (Vision Model2)**. Figure 5.7 shows our pipeline used for V-M1 and V-M2.

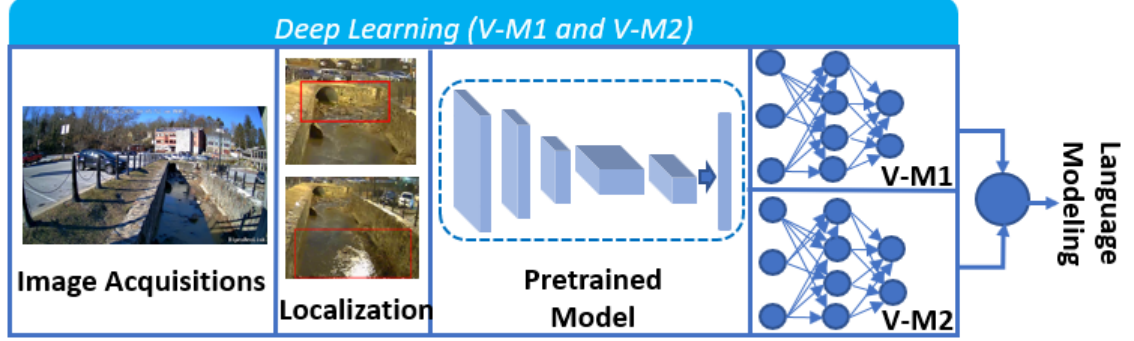


Figure 5.7: Classification & Detection Pipeline

## ii) Segmentation V-M3

In this section, we briefly summarize our segmentation pipeline and approach to the scene parsing problem. The pipeline used in image segmentation are depicted in Figure 5.8. We implemented two deep learning models; save their model weights, and use real-time inference and object detection for semantic segmentation. We use two semantic segmentation models to perform the binary segmentation task of identifying the island.

Our first deep learning model is a standard U-Net architecture model, based on U-Net: Convolutional Networks for Biomedical Image Segmentation [120]. Our second semantic segmentation model is a variation of the U-Net model with added dropout for model correction. FloodBot is integrated with its own Twitter account [22, 134].

FloodBot is automated to Tweets site image, weather condition and water level reading from the site every six hours. We use the sensor data in 5.8 c to compare the results and post it to social media. After receiving the images, we label these images to track visible landmass in the stream. We call the visible landmass around the river bed as “Island”. The white mask on the images is an island and indicator of a non-flooding stream.

We collected images from the site in various weather conditions throughout the deployment. Figure 5.8(a) shows images captured from the deployment site, the segment labeling from the training data set, and binary pixel mapping.

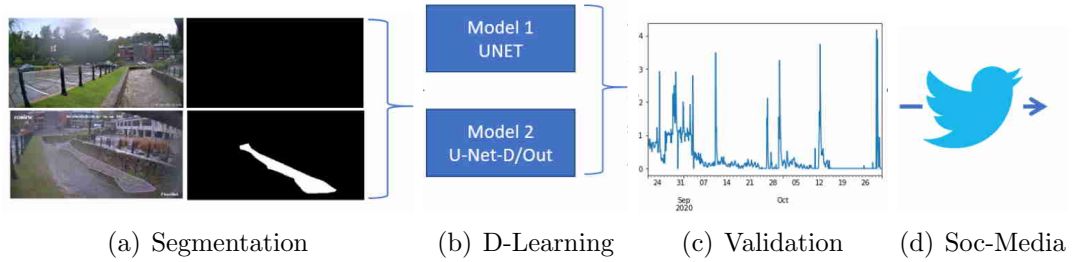


Figure 5.8: Segmentation Pipeline

We run two deep learning models to parse the images and segment the landmass as shown in Figure 5.8(b). Floodbot is also equipped with water level sensors. We use sensor reading as a model validation parameter. The previously established water level threshold reading Figure 5.8(c) allows us to categories the flood level into various flood classes such as base-flow condition, minor flooding, or extreme flooding in the area.

We use two metrics to validate the segmentation tasks: intersection over union

(IoU) equation 5.3a and Dice Index ( $D$ ) equation 5.3b for mean average precision at the different thresholds. Validation Metrics:

$$IoA(A, B) = \frac{(A \cap B)}{(A \cup B)}, \quad (5.3a)$$

$$D = \frac{2 \cdot |A \cap B|}{2 \cdot |A \cap B| + |B \setminus A| + |A \setminus B|}, \quad (5.3b)$$

$$S_i(t) = \frac{TP_i(t)}{TP_i(t) + FP_i(t) + FN_i(t)} \quad (5.3c)$$

IoU is a simple ratio that shows how close the pixel is found between training and prediction regions. The second and more stringent metric is  $D$ , which also penalizes the score by including actual positive samples. We then calculate the average precision for each image and evaluate the model performance as depicted in equation 5.3c. We compute the IoU for each of the land mass predicted by the model against the ground truth or the labeled image. We then calculate if this mask fits at a range of IoU thresholds. At each threshold, we calculate the precision across all our training images.

To train our model, we collected the stream images and labeled them with their landmass. The masked image is a binary black and white pixelated images. The white masking as shown in Figure 5.8 (a) is the landmass, and complete black signifies the complete inundation of the area, i.e., no landmass is visible on site. We collected 4000 such images from various weather, time of the day, and rain intensity patterns. The trained model can detect the presence or absence of these elements from the incoming frame. We trained the model for 50 epochs with a learning rate

of 0.0001 and used binary cross-entropy as our loss function to compute the training and validation loss. Finally, the results from the validated model will be propagated to Social Media using FloodBot’s tweeter handle Fig 5.8(d).

## 5.5 Experimental Results

This section describes experiment, evaluation and our methodology and approach for experiments performed under shallow learning and deep learning pipeline. We present the experiments results from different model in same order as earlier, i.e classification, detection and segmentation.

### *i) Shallow Learning Results*

We compared our classification results with four shallow learning using manually extracted features (1) LogisticRegression; (2) K-Nearest neighbors; (3) Linear Discriminant Analysis; (4) Gaussian Naive Bayes and ; (5) Support Vector Machine. As shown in 5.3 SVM achieved the best classification performance.

Table 5.3: Summary of Shallow Learning

ML	Test Accur.	Train Accur.
LogisticRegression	0.49	0.44
K-Nearest neighbors	0.67	0.65
Linear Discriminant Analysis	0.53	0.48
Gaussian Naive Bayes	0.51	0.46
Support Vector Machine	0.70	0.66

We show the result from best performing model (SVM) in the Table below.

### *ii) Classification Results*

Table 5.4: SVM-Best Performing Model

Class	precision	recall	f1-score	Support
flood	0.8	0.7	0.75	130
minor_flood	0.73	0.39	0.51	119
no_flood	0.55	0.86	0.67	126

Due to the resource constraints, we sampled 500 images instead of 15,686 from each of the flood categories and extracted low-level image features (1 channel-pixel values) from them and ran through following models. The inference algorithm enables FloodBot to detect the Flooding condition in real-time and provide output to the downstream components.



Figure 5.9: Output: Custom CNN

#### a) Custom CNN

It took us about 2.5 hours to train the model on these images. Figure 5.9 shows the result from one of our validation set. In the given example, the model correctly classified the image set into  $\{No\ Flood, Minor\ Flood\ and\ Flood\}$  from respectively. Using the non-transfer learning based models we show that the SVM based RBF model is only able to achieve 53% accuracy where as the CNN based model achieves 67%.

### b) Transfer Learning

We trained The transfer learning based V-M1 for 100 epochs with custom head. We train V-M1 offline with thousands of images captured by the FloodBot camera. Then during the inference, we use trained V-M1 to classify the image into respective flood classes. The model is able to achieve accuracy of 97.18%. Results from our experiment is shown in Table 5.5.

Table 5.5: Results: Classification

Model	Type	Parameters	Accuracy
SVM	RBF Kernels		53%
CNN	4 Layer	relu, softmax	67%
V-M1	Transfer Learning	Glb Avg Pooling,relu,Dropout	97%

### iii) Object Detection Results

From the dataset collected over a month, the V-M2 model identified 90 distinct objects such as  $\{Car, Wheel, House, Tree, Building, Vehicle, Land\ vehicle, Tire, Window, Plant, Bench, Truck\}$ .

Since the camera is pointing to a parking lot area and images are triggered by the motion mostly caused my moving vehicles, the most of the objects detected in 2000 randomly sampled images are those of car and trees.



Figure 5.10: Results: V-M2

Percentage wise summary of most and least occurring objects detected from 2000 input image frames by V-M2 is shown in Table 5.6.

Table 5.6: Most and Least Detected Objects

<b>Objects:</b>	Car	Tree	vehicle	Watercraft	Bicycle
<b>Percentage:</b>	20.79%	17.74%	16%	0.55%	0.40%

#### *iv) Segmentation Results*

In segmentation, our main goal is to observe the current situation of potential flood areas, infer other contextual information, and use it to strengthen FloodBot’s flood reconnaissance capability. We implement two deep learning models; save their model weights, and use real-time inference and object detection for semantic segmentation. For inference, we store our trained model’s weight and let the model detect the presence or absence of landmass in the incoming image-frame. The presence/detection of landmass the model informs us that the stream is in its base flow condition, and there is no imminent flood risk area.

#### **a) Image Segmentation - UNET**

The proposed U-Net models are trained in supervised learning, which requires ground truth pairing with input masked images. Therefore, the performance of the U-Net critically depends on the training data. Figure 5.11 shows the results of the U-Net in two different image set. In Fig 5.11 (a) the land mass is comparatively regular in shape and hence the predicted land mass also follows the contour closely. In the second image set Figure 5.11 (b), the land mass is very irregular and hence the model is unable to predict the segmentation mass properly. The IoU or Dice

Index in later image will be lot lesser compared to the one above. Nevertheless, mere detection of island in flood zone is enough to gauge the flood state. The solution also alleviates the need to deploy expensive sensors to the ground. A properly trained model can also be used as a base model for transfer learning into relatively hard to deploy area.

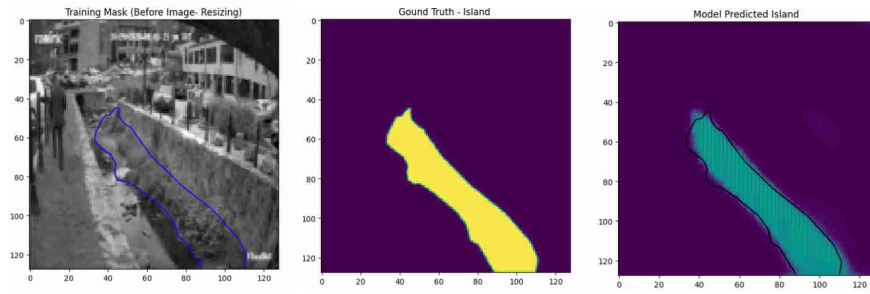
Table 5.7: U-Net Result

Metric	Model		Sample Sizes			
	U-Net	U-NetD/O	Train	Test	Valid	Epoch
IoU	0.76	0.82	4000	400	100	50
Dice	0.79	0.84	4000	400	100	50

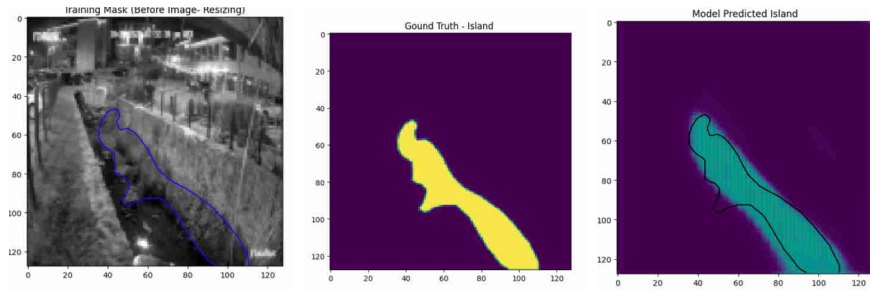
We only labeled 4000 training image which holds limited information about the site. Therefore, a large scale and meaningful data set with distinct characteristics is crucial for training. The results from these two deep learning model are discussed below. This section describes in detail about our custom image segmentation model and their performance metric and validation metrics.

## b) Deep Lab

We experimented with a pre-trained model called Deep-Lab[40] to evaluate performance on our dataset. The results from Deep-Lab model called as Deep-Lab is depicted in Figure 5.12. We show two extreme condition and evaluate the image segmentation result qualitatively.



(a) Training Results



(b) Training Results

Figure 5.11: Results: Semantic Segmentation



(a) Kids Playing in the River Bed



(b) Man Taking Picture During Extreme Flood

Figure 5.12: Qualitative Result-Deep Lab

In Figure 5.12 (a) there are kids playing on the stream bed, while one of

them is trying to climb on the stream-wall. The segmentation model is able to identify all those (kids) human bodies based on Deep Lab's PASCAL-Person-Part training. They report that their model is trained with person part annotations such as Head, Torso, Upper/Lower Arms and Upper/Lower Legs, resulting in six person part classes. Not just person class, the model is able to identify correct orientation of humans in that picture (pose estimation). Finally in Figure 5.12 (b) where a man is taking picture of Flood along the river. Only parameter that needs change from images captured is to set the image sizing to original model size and pixel normalizers. The average inference time to these images for Deep-Lab is The Average 0.024s well within allowance for FloodBot's real-time image segmentation task.

Our experiment shows that we can detect landmass in the stream that comes with the following properties: (1) it can be used as a validation technique for flood zone during pre and post rainfall events; (2) it is transferable to any other location and site with minimal setup. Our results demonstrate that the end-to-end U-Net can provide superior performance and has the ability to accurately segment the images in various weather condition.

Using a pre-trained model such as Deep-Lab in association with a custom model can greatly empower the site reconnaissance needed from the disaster-prone area. We argue that a power full pre-trained model running in tandem with a smaller model can create a useful and life-saving tool. We use the physical sensor on the ground to validate the flood condition (island detection), but we have not

effectively used the information from the sensor reading in improvising our Deep learning model performance. The model performance could be improved by passing the sensor reading into the segmentation pipeline.

Until now, we showed results from various supervised learning to detect the Flood condition. We first used shallow learning techniques to generate image feature and tried to classify them. We then progressed into more challenging deep learning model for classification and object detection. As seen by the results the shallow learning performs badly. Even the best performing model- SVM in our case is only 70% accurate as shown in Table 5.4. Therefore we need to find a model that has a better learning algorithm. Shallow learning fails to generalize and the learning of these models is sub-par for our problem of flood detection.

We now move into models that can perform better at learning the data representation. Also extracting features from the image is a resource intensive task and subjective. Getting good-quality training data is one of the biggest problems in machine learning because data labeling is a tedious.

All our model developed/discussed so far, shallow learning to the most sophisticated image segmentation model, they all exhibit one problem, needing a close supervision. In other words, they need to have ample labeled data to perform well. We have a limited dataset and compute resource compared to other benchmark computer vision tasks such as Imagenet [121, 49, 87] etc. so it is hard for us to assure that the model will generalize else where.

Since all the classification models are trained as three-class classification models, the deep learning models are capable to categorize only in one of those class. For example, if we were to increase the categories from three to four, we have to re-start from beginning We have to re-classify and re-label and re-train, re-build our models. That means our previous model needs re-training from scratch.

Recalling that FloodBot’s task is to generate content for downstream conversational AI, we list the following shortcomings in FloodBot’s current capability and address those in the next section. We see all of these shortcomings as a way to widen FloodBot’s scene parsing capability and more generalize approach. In the next section, we expand FloodBot and its scene parsing capability into a more relaxed learning environment of semi/supervised learning. In other words, we re-frame the learning paradigm from classification to similarity search.

## 5.6 Metric learning

Classification are often performed as a supervised learning. That means a particular deep learning model can train  $\mathcal{K}$  classes. During the training phase model would have seen  $n$  samples belonging to  $k$  classes. The trainer in Classifications setting can only classify the model into one of the  $k$  classes. That means the model is limited in scalability. Every time we add a new class, we need to re-label the dataset and re-train the neural network. For example in our problem set, we notice that the stream bed changes its vegetation with the season and also after major

flooding event. Therefore, rather than posing our problem as a strict supervised classification problem we now pose the problem as a similarity search problem.

To compare and contrast two objects or entity we need distance (in vector space). Developing a distance function for a task is called the metric learning problem. This problem has been shown to be useful when used with nearest-neighbor methods and other methods that use distances or similarities. The fact that the supervised information is a function of the ideal distance (or similarity) is key to distinguishing the methods we experiment in this section.

In databases, search engines, and a number of other applications, similarity searches are common. Had our problem been limited to retrieving data recorded by sensors then it would have been a simple matter of utilizing a query language—for example, locating readings greater than or within a certain threshold. Our problem of detecting flooding stream is complex and much more dynamic and challenging. The question we need to answer *-is this an image of flood?* is hard and changes with weather, time of the day, stream morphology etc. Therefore, we try to answer that question by grouping all the Flood like images (data representation) in one place and see how closely they align with a new data instance.

### 5.6.1 V-M4: Similarity Search

In the machine learning area, similarity search is modeled as the problem of entity matching, which aims at identifying whether two entities closely resemble

each other. Different types of models are trained to find such entity pairs. Recently, deep learning techniques have been extensively adopted in identifying image semantic similarity, where images are mapped into low-dimensional continuous vector space. They use embedding techniques to find matched entities. The building blocks of deep learning for image similarity measurement are mainly Convolutional Neural Network (CNN) [5] and distributed representation learning [10]. Intuitively speaking, contrastive representation learning is “learning by comparing”. The goal of contrastive learning is to pull “similar” samples together and push “dissimilar” samples away. We are inspired by the latest work [41] where authors trained the Siamese network for Contrastive Learning of Visual Representations. The authors used many image augmentation techniques and allowed the model to discriminate between similar and different images assuming that the same image’s augmentation preserves its representation. They used  $X_i \Rightarrow$  Data Augmentation; crop resize, flip, intensity. But we are fortunate that we get such augmentation because of our natural image. Our image dataset automatically brings spatiotemporal data, natural data augmentation, and source variant augmentation. Hence our model is by default trained in additional data augmentation.

### • Distance Metric

The main component of the contrastive learning is about finding distance between two vectors. We use both Euclidean distance and Cosine Similarity as our distance metric. The formula used to compute these distance are shown in equations 5.4 and 5.5 .

$$\begin{aligned} \mathbf{d}_{eclud} &= \sqrt{\sum_{i=1}^n (h_i - h_j)^2} \\ &= \frac{x \cdot y}{\sqrt{x \cdot x} \sqrt{y \cdot y}} \end{aligned} \quad (5.4)$$

$$\begin{aligned} \mathbf{d}(\mathbf{x}_i, \mathbf{x}_j) &= \cos(\theta) = \frac{x_i \cdot x_j}{\|x_i\| \|x_j\|} = \\ &= \frac{\sum_{i=1}^n x_i x_j}{\sqrt{\sum_{i=1}^n (x_i)^2} \sqrt{\sum_{i=1}^n (x_j)^2}} \end{aligned} \quad (5.5)$$

### • Few Shot - Learning

Deep learning models are known to rely on large amounts of labeled data during training. That's one of the reasons they work so well. However, that also poses significant limitations. Specifically on research like ours, where we only have a few samples to train the model. A traditional deep learning model may not generalize well in that scenario. The goal here is to train a deep learning model capable of rapidly generalizing given limited training data. The trained model should also generalize to new tasks with only a few samples or perhaps no samples at all. Therefore, the objective is to enable models to perform under practical settings where data annotation could be infeasible and expensive. Furthermore, if a new class gets added over time will make the model useless. Since, Few shot learning aims to solve this restriction posed by traditional deep learning models, we use Few shot learning after exhausting the supervised learning approach. If we denote  $x$  an image with its corresponding label  $y$  then FSL is defined as:

$$D_{train} = (x_i, y_i)_{i=1}^I \text{ where } y \in Y \ll X \quad (5.6)$$

Few Shot learning is also defined as  $N - Way - K - Shot$  where  $K$  is labeled dataset from  $N$  total classes and  $N < K$ . In case of Zero Shot learning  $K = 0$ :

$$D_{train} = \left\{ (x, y, (a(y)|x \in X^s, y \in Y^s, a(y) \in A)) \right\} \quad (5.7)$$

where  $X^s$  is set of image/features from seen class  $Y^s$ ,  $a(y)$  is a semantic embedding for class  $y$ .

$$D_{test} = \left\{ (x, y, (a(y)|x \in X^u, y \in Y^u, a(y) \in A)) \right\} \quad (5.8)$$

where  $X^u$  is set of image/features from unseen class  $Y^u$ ,  $a(y)$  and  $Y^u \cap Y^s = \emptyset$

Few-shot recognition by learning to compare query images against few-shot labeled sample images. First an embedding module generates representations of the query and training images. Then these embeddings are compared to determines if they are from matching categories or not [141] .

## Problem Overview

Supervised learning needs lot of label data set to perform well. We try to approach our problem in semi supervised learning where we need limited label data. Our goal is to get best results with limited or no data annotation. Rather than training the models to learn the classification task, we try to learn data representation. Learning powerful representations of different types of data these are also called embeddings of data from unlabeled  $x$ . For example we are interested in image classification problem (flood detection), instead of directly using  $x$  in classifier we

use the representation of  $x$

$$\overset{data}{X} \longrightarrow \overset{Repr}{f}(\mathbf{x}) \quad (5.9)$$

## METRIC LEARNING

- **Problem Definition**

Given a set of images  $\mathcal{P}$  where  $p \in \mathcal{R}^d$  For a query image  $p_i \in \mathcal{P}$  We can locate  $p_i^+$  and  $p_i^-$

such that  $p_i^+ \in \mathcal{P}$  is more relevant than  $p_i^- \in \mathcal{P}$

Alternately  $\text{sim}(p_i, p_i^+) > \text{sim}(p_i, p_i^-)$

$$\overset{data}{X} \longrightarrow \overset{Repr}{f}(\mathbf{x})$$

- **Goal**

Learn a Similarity Function  $\mathcal{S}_w$

$$\mathcal{S}_w = \text{sim}(p_i, p_i^+) > \text{sim}(p_i, p_i^-) + \mathbf{m}, \forall p_i, p_i^+, p_i^- \in \mathcal{P}$$

- **Approach**

Minimize Loss  $\mathcal{L}$  or maximize the distance  $\mathbf{d}$  (**Contrastive Loss**)

1. *Pairwise*  $\mathcal{L}_p$  :

$$\mathcal{L} = \begin{cases} d(p_i^+, p_i^-) & \text{if } \text{PositivePair} \\ \max(0, \mathbf{m} - d(p_i^+, p_i^-)) & \text{if } \text{NegativePair} \end{cases}$$

2. *Triplet*  $\mathcal{L}_t$  :

$$\mathcal{L}(x_a, x_p, x_n) = \max(0, \mathbf{m} + d(x_a, x_p) - d(x_a, x_n))$$

Such problems are approached as two steps process, first: find data repre-

sentation and second evaluate the data representation and try group similar data representation into same groups (cluster). If we are able to learn the embedding or the representation as per equation 5.9 then the hope is that we expect to see samples coming from different labels to somehow cluster together. We approach the problem as if there is limited labeled data pair (semi supervision). We hypothesize that being able to classify and learn from the unlabeled or limited labeled dataset is a far more challenging and rewarding undertaking to increase an artificial agent's efficiency and re-usability in detecting the flood Condition.

The problem can be formulated as follows:

Given observations  $\mathcal{X} = \{x_1, \dots, x_N\} \subseteq R^R$  of a specific flood category, the classifier's task is to learn a mapping  $x_n \mapsto y(x_n) \in R^R$  where  $y(x_n)$  is data representation in some embedding space that matches the unknown ground truth stream condition  $y_n \in R^R$  as close as possible. Depending on the availability and use of limited label data we can subdivide our problem as semi-supervised/weakly-supervised or self-supervised. There are many kinds of semi-supervised learning algorithm, we look at one that learn the representation using similarity index between data points. One data instance can represent an infinite-state in the real world, and there may not be a clear demarcation between classes. To solve the ambiguity, we pose an inductive bias into that search space. For both supervised and semi/self-supervised settings, we assume that each observation  $x_n$  corresponds to only one class, i.e.  $\mathcal{Y} = \{y_1, \dots, y_M\} \subseteq R^R$  of representative images corresponding to that class.

### 5.6.2 Model Architecture

Though, similarity search is a fundamental machine learning problem recently there have been many advanced work in the area. Siamese Neural Network and FaceNet [125] is one of such model and inspiration for our research work. FaceNet is a facial recognition system developed at Google by Florian Schroff and et.all [125]. We inspire our experiment under same FaceNet architecture. The FaceNet study describes two distinct architectures: Inception Model Architecture based on GoogLeNet and Deep CNN Based on Zeiler and Fergus Network Architecture [?]. Our work is based on Inception Model Architecture Based on GoogLeNet which has 20 times fewer parameters (around 6.6 million to 7.5 million) and  $5\times$  fewer FLOPS (around 500 million to 1.6 billion). FLOPS is a commonly used unit of measurement for computer performance that needs floating-point computations. Larger FLOPS helps Model's accuracy improves but is resource-intensive. A network with fewer FLOPS operates more quickly and consumes less memory but has a poorer accuracy.

Given an image captured by FloodBot, the model extracts high-quality information from it and predicts a 128-element vector representation of these features, referred to as an embedding. Our model is a convolutional deep neural network trained using a triplet loss function that promotes vectors with the same identity to align together (minimize distance) and segregate vectors with different identities (maximize distance).

The model accepts a 160 x 160 x 3 flood image as input and generates a

128-element face embedding vector. The embedding provides information about the significant features of a flooding image. Then, the model determines which class label from the training flooding embedding has the smallest  $L2$  distance to the target flood embedding.

The most important part of our approach lies in the end-to-end learning of the whole system. To this end we employ the triplet loss .We strive for an embedding  $h_i$ , from an image  $x_i$  into a projected feature space  $Z$  such that the squared distance between all flood images, independent of imaging conditions, of the same scene is small, whereas the squared distance between a pair of flood images from different scene is large.

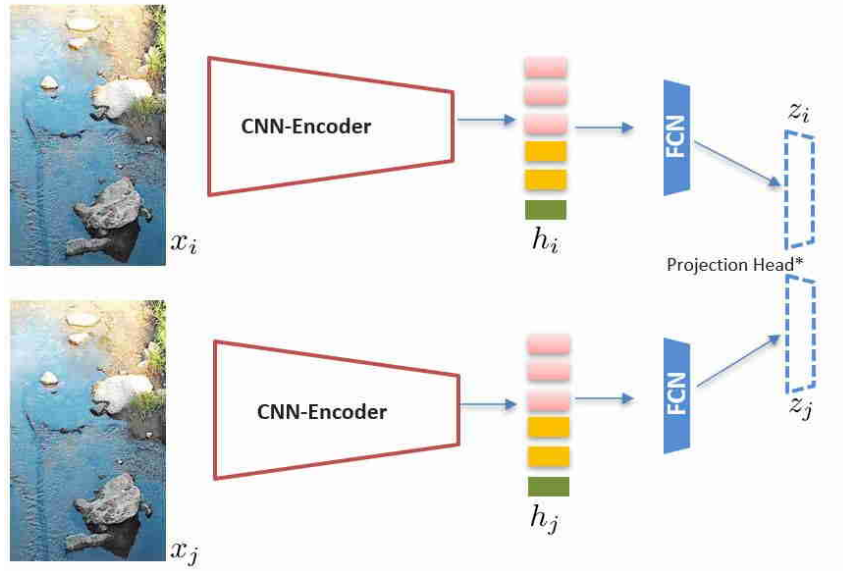


Figure 5.13: Siamese Network

A Siamese neural network comprises twin networks that receive distinct inputs but are linked at the top by an energy function. This function calculates a metric (distance) between each side's highest level feature representation Figure 5.13. Our typical model is a Siamese convolutional neural network composed of  $L$  layers with

$N_l$  units each, where  $h_{1,l}$  signifies the hidden vector in layer  $l$  for the first twin and  $h_{2,l}$  denotes the same for the second twin.

In the first  $L - 2$  layers, we employ rectified linear (ReLU) units and sigmoidal units in the remaining layers. The model comprises a series of convolutional layers, each of which uses a single channel and filters of varying sizes with a fixed stride of one. The number of convolutional filters is specified as a multiple of 16 optimizes performance. The network activates the output feature maps using a ReLU function, optionally followed by max-pooling with filter size and stride of 2.

### 5.6.3 Implementation

We train neural network such that when the images are different they would give get a low similarity score and when the two images are same they would have high similarity score. Therefore, the task for our model is not classification but we are training the neural network to output a continuous number for range of similarity score. Low similarity score can have its own range of scores and high similarity can have its own range of scores. A neural network trained would be trained using two image pair. If the distance between two images is larger than a threshold then it means that they are not similar

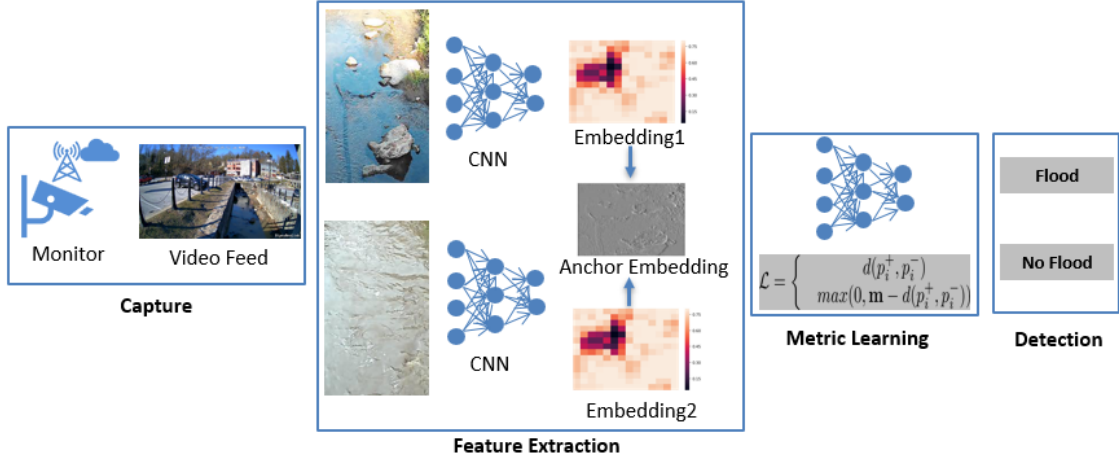


Figure 5.14: Deep Learning Pipeline

## Metric learning Pipeline

The proposed Metric learning Pipeline for Flood detection system is depicted in Figure 5.14. The pipeline is composed of four major components: data collection, feature extraction, metric learning using Triplet loss and the, and detection output. We capture video from site deployed camera system.

Next we use two identical (Siamese), a deep neural network (CNN) to learn generative domain (flood) -specific features. The proposed model is trained using a triplet loss function adjusted for learning feature embeddings. During Metric learning we compute the distance (margin) between embedding-pairs.

To train such a similarity scoring network, we use a particular type of network that passes two images in parallel, hence the name Siamese. This means that we pass images through a CNN and obtain their embedding in some low dimension with out changing model parameters.

### 5.6.4 Experimental Results

The most challenging part of this work is to create similar image pair or anchor selection. We experimented with various techniques to find best matching images for training purpose. For example, our preliminary thought was to find images by time of the day or pair images based on ground weather etc. These all measures did not yield better contrasting pair. We then resort to the Water Sensor depth based image pairing. That seemed to produce best result.

#### Anchor Selection

Once the pairing technique is formalized. Next step is to find the best anchor images to compare distances. Initially, we tried just to get differential distance between negative and positive pair. The distance thus found was not separator. The idea behind training Siamese network and learning metric distance is to be able to find a linearly separable representation in low dimensions. We use following algorithm 4 to find the best distance separator.

##### ***i) Best Margin 2-Way Anchor***

For this experiment we picked two images one from the negative class (no flood) and one from the positive (flood). Our expectation is that all the images that are close to ‘flood instance’ will have similar distance and vice versa. The negative and positive image pair is depicted in Figure 5.15. We use embedding of the image

---

**Algorithm 4:** Metric Learning - 2 Way Anchor

---

**Input:** Base Images:  $p_i^+$ ,  $p_i^-$ ,  $img_i$

**Output:** Distance:  $img_{i=1}^n \leftarrow d_i \{dist\}$ ,  $m$

- 1: Find best anchor images
  - 2: Pass the image ( $img_i$ ) through image encoder
  - 3: Get image embedding ( $img_i$ ) in low dimension.
  - 4: Compare  $d_i$  with base image  $\{p_i^+, p_i^-\}$
  - 5:  $d_{pos_i} \leftarrow \text{sim}(d_i, p_i^+)$
  - 6:  $d_{neg_i} \leftarrow \text{sim}(d_i, p_i^-)$
  - 7: Evaluate best Margin
- 

and tried to cluster them using two distance metrics (cosine similarity and euclidean distance).

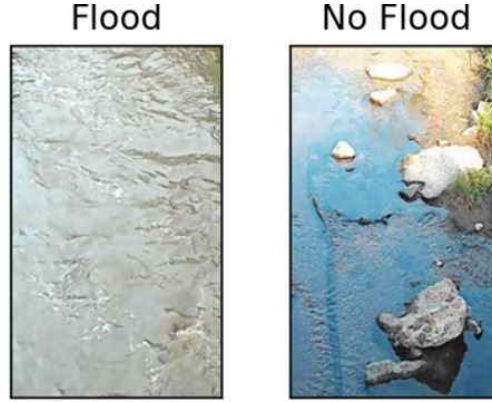


Figure 5.15: Negative & Positive Image Pair

We started with an assumption that the images would be able to auto align (cluster) together based on their proximity with similar images (flood to flood and no-flood to no Flood). We passed our images through the pre-trained ResNet18 [60] model and extracted their feature representation (embedding) as a CNN output. We then experimented with cosine similarity and euclidean distance as our distance metric to find their similarity or cluster the learned representation.

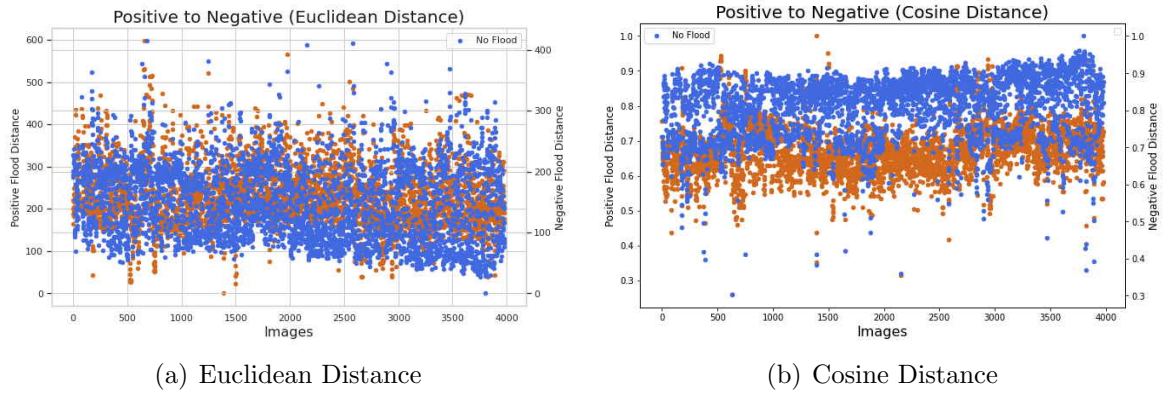


Figure 5.16: Negative Positive Clustering

The distance plot is shown in image below in Figure 5.16. Our hope was to yield a linearly separable plane for shallow classifier. However, as shown in Figure 5.16 (a) we observed that in case of Euclidean distance as a metrics completely inseparable. We note that Euclidean distance being just a magnitude of distance (no direction of vector), it makes sense that they are equally distance from each other. Next we used the cosine distance 5.16 (b) as our distance metric, the clustering seems somewhat improved but still not linear separability. Just using negative and positive image pair to separate the flood images was inconclusive, therefore we experimented with the Triplet loss.

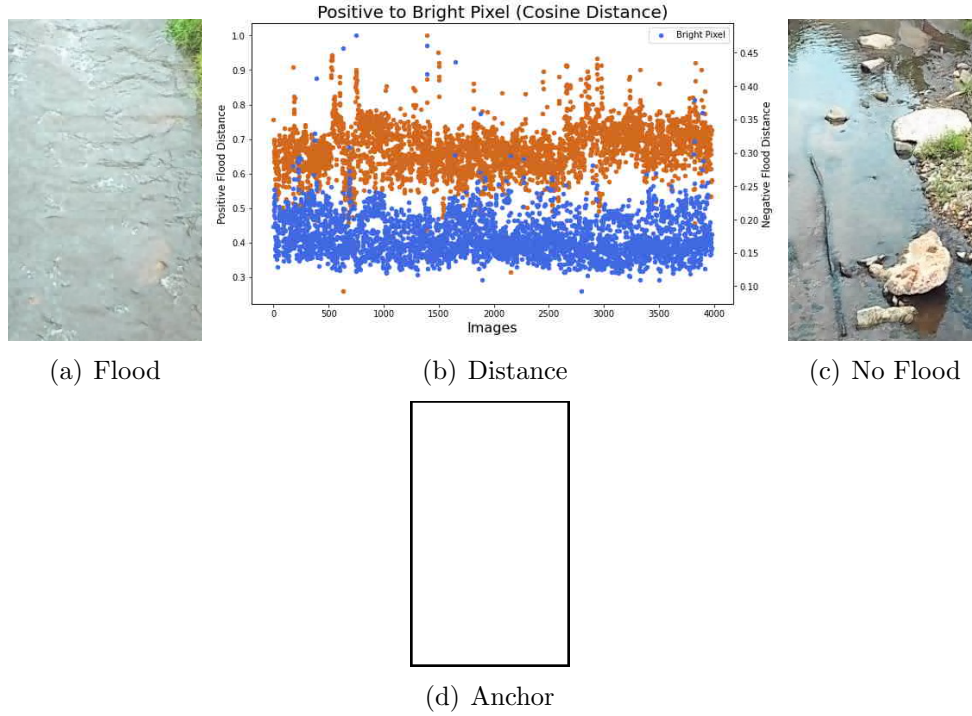


Figure 5.17: Triplet Loss - White Anchor

### *ii) Triplet Loss - White Anchor*

The goal of Triplet Loss is to create triplets consisting of an anchor image, a positive image (that is similar to the anchor image), and a negative image (which is dissimilar to the anchor image). First we tried fixing white image as an anchor and tried to find the distance between the white anchor and positive and negative pair. The distance plot is shown in Figure 5.17 and white image. Upon further evaluation of the clustered images, we observed that the white background was detecting (pulling over exposed images together). The triplet loss using white distance metrics was not able to detect the flood similarities.

### *iii) Triplet Loss - Black Anchor*

With this setup, we find the best margin and image pair for our contrastive learning model. As seen in Figure 5.18 the best separator (image anchor pair) seems

---

**Algorithm 5:** Metric Learning - Triplet Loss

---

**Input:** {ref Images:  $p_i^+$ ,  $p_i^-$ ,  $p^{blk}$ ,  $p^{wht}$ }, {Image:  $img_i$ }

**Output:** Distance:  $img_{i=1}^n \leftarrow d_i \{dist\}$

- 1: Find best anchor images
  - 2: Pass the image ( $img_i$ ) through image encoder
  - 3: Get image embedding( $img_i$ ) in low dimension.
  - 4: Compare  $d_i$  with four ref image  $\{p_i^+, p_i^-, p^{blk}, p^{wht}\}$
  - 5:  $d_{pos_i} \leftarrow \text{sim}(d_i, p_i^+)$
  - 6:  $d_{neg_i} \leftarrow \text{sim}(d_i, p_i^-)$
  - 7:  $d_{pos_i} \leftarrow \text{sim}(d_i, p^{blk})$
  - 8:  $d_{neg_i} \leftarrow \text{sim}(d_i, p^{wht})$
  - 9: Evaluate best Margin
- 

to be Negative (no-flood) with Black pixel. We attribute this wide margin to the intensity subtraction happening in their embedding. We used the metric distance learned above to separate our positive and negative pair for training. We use algorithm 6 to define our positive and negative samples.

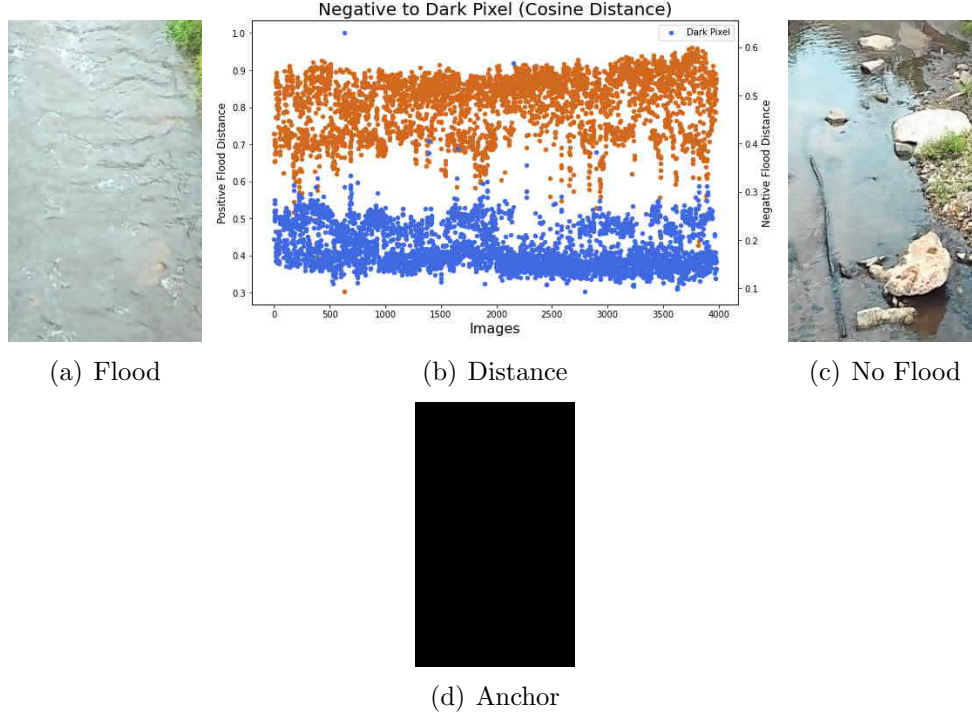


Figure 5.18: Triplet Loss - Black Anchor

---

**Algorithm 6:** Similarity Threshold

---

**Input:** cosine distances  $d$ , Image:Img<sub>ts</sub>  
**Output:** Image Similarity :  $\{ pos_{pair}, neg_{pair} \}$   
**if**  $No_{FldCosDist} \geq 0.8$  **then**  
    | *Negative Pair*;  
**else if**  $Fld_{CosDist} > 0.55$  and  $No_{FldCosDist} \leq 0.68$  **then**  
    | *Positive Pair*;

---

With the optimized distance threshold ready, we performed the model training. We experimented with various CNN formulations and found out that best encoding is obtained when we use pre-trained model as our backbone, specifically, the ResNet18 [61]. We believe that the pre-trained model perform better than custom CNN because the model has been trained in large dataset and for longer time. In other words the embedding capacity of the model is dependent on the volume and disparity of images the model has seen. The last layer of ResNet 18 yields 512 length vector from a d gray scale image of  $100 \times 100$  image.

Figure 5.19 shows the result from our empirical analysis. We randomly chose images from our corpus and tested images against unseen images. Our experiment shows that we can perform similarity search and detect if the images captured by FloodBot is closer to Flood or non -Flood.

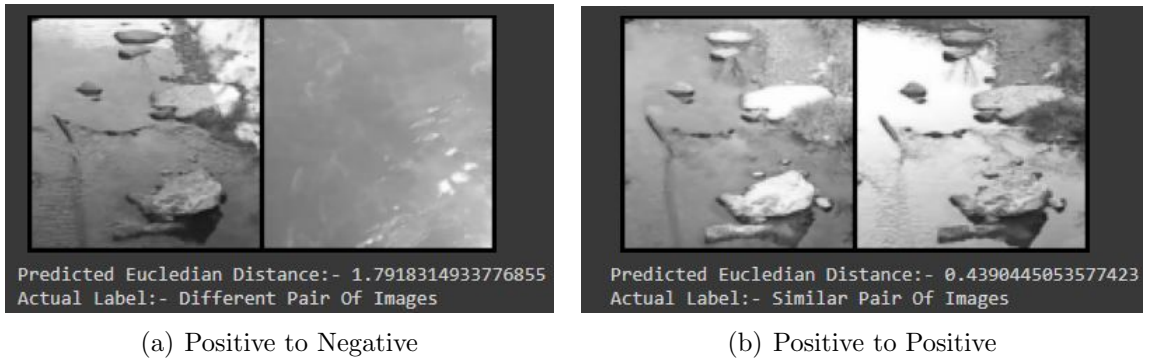


Figure 5.19: Empirical Results

We note that the distance between correctly classified negative pair seems to be closer than the distance between negative pair.

## 5.7 Conclusion & Discussion

In this chapter, we presented our learning and experiment in Flood Detection using deep learning. We first started with shallow learning then used deep learning for classification and object detection and use segmentation and landmass detection from the stream as yet another flood detection techniques. As we grow and build the Flood detection capabilities for FloodBot. Other than the shallow learning deep learning did not require feature extraction. Model learned the representation based on our image data. We believe this approach is valuable. We have evaluated FloodBot's image segmentation power with limited labeled data and a less sophisticated U-Net model in this work. Adding drop out to the U-Net does not significantly change the model learning parameters; this only allows for model smoothing and reduces over fitting.

We detect the flood conditions and identify objects in harm's way by stacking deep learning models such as a convolutional neural network V-M1 and object detection-V-M2 and prepare the Question Answering using Language Model-LM (Chapter 7). However, the models described above provide discrete information not directly useful for the content generation and consumption for FloodBot. We need to organize the extracted data and provide readable content for FloodBot. We use

the Model stacking technique shown in Figure 5.20 to deliver content and discuss this technique as an example on how to coherently bring leanings from these model into useful consolidated one piece information. We deploy two parallel deep learning

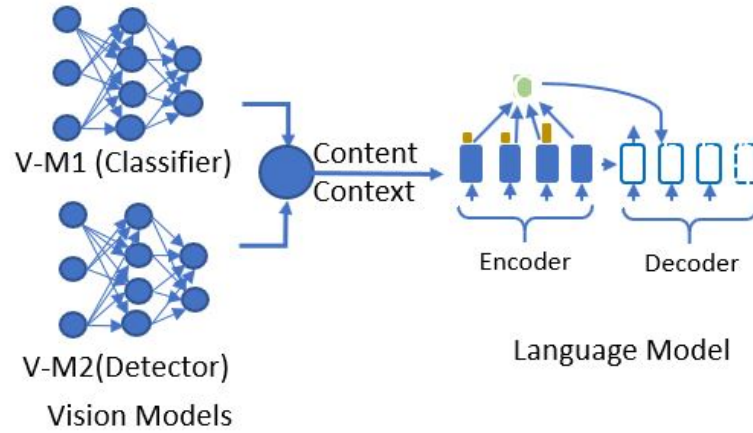


Figure 5.20: Model Stacking

vision models to observe the flooding condition and scene. Flood detection model called **V-M1 (Vision Model1)** is our first model. The second model is a Single-shot Multi-box object detection model called **V-M2 (Vision Model2)**. We later feed the image contents to a knowledge base of our artificially intelligent FloodBot and explore its AI-Conversing power using end to end memory network. We also showcase the power of cross-domain transfer learning and model fusion techniques.

While the initial premise of FloodBot is to detect flood conditions from a given image, the work does not stop there. FloodBot is expected to parse the scene around it. Therefore, detection of flood in the incoming image is its first step. Ultimately, the language model receives the augmented scene parsing content. We show the model stacking and The setup of three deep learning models in Figure 5.20.

We combine and show results from from V-M1 and V-M2 in Table 5.8 for one of the sample image frame (e.g.20200516152104.jpg). This is one of the sample image frames form our full dataset. The output from V-M1 i.e, current Flooding condition as ‘*Minor Flood*’ is the output from our V-M1 and detected objects {building, Car etc.} are the output of V-M2. Once we have these two elements together (frame by frame and at a given time), the FloodBot has current contextual information to converse and answer queries about real time hazard potential.

Table 5.8: Joint Model Learning V-M1 & V-M2

frame.id	summary	precipIntensity	humidity	V-M1	V-M2	
				<i>Flood_Label</i>	<i>Object</i>	<i>Total_Object</i>
20200516152104.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Building	2
20200516152104.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Car	90
20200516152104.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Person	1
20200516152104.jpg	Possible Light Snow and Windy	0.0257	0.81	Minor Flood	Street light	6

Here we also discussed our deployment of a real-time flood monitoring. We collect, annotate and visually parse images from potentially hazardous areas. We detect the flood conditions using a convolutional neural network (CNN) both custom (hand crafted CNN model) and pre-trained model. It is well perceived fact that computer vision work there needs to be lot of low level image pre-processing before achieving acceptable results from shallow learning model such as SVM often not attainable.

The object detection count and score from the vision model V-M2 is probabilistic and somewhat heuristic in nature. We argue that the ultimate use of this kind of application is either to have a sense of emergency or try to estimate the possible damage. Thus the object detection and count measured are mostly for

estimation purposes. During disaster and life saving events the timely information provided by our model could be more valuable than trying to get to the correct count.

We are able to detect flood using either of the two supervised flood classification models (shallow + deep learning) model and combine it with object detection model to get a sense of risk around the site. However, we still see some limitation to the FloodBot’s scene parsing capability. The scope of object detection stops at identifying what is in the vicinity of the flooding stream. Now FloodBot can answer if there is a person or car, buildings in the flood zone. It is able to define the proximity of the object in the context of a flooding stream. For example, there might be 50 Cars parked in the parking lot, only 10 might be too close to the stream and hence in actual harms way. To answer such question we deployed flood detection technique using land mass detection in the stream. We train V-M1 offline with thousands of images captured by the FloodBot camera. Then during the inference, we use trained V-M1 to classify the image into respective flood classes. The inference algorithm enables FloodBot to detect the Flooding condition in real-time and provide output to the downstream components.

During our qualitative analysis, we observed that weather conditions create a barrier to the flood detection model. Fog and heavy rain impede the model’s inference quality. While there have been great success with vision-based flood detection, the main challenge arises when the FloodBot loses visibility. Low illumination adverse weather conditions can cause visibility loss, often precursory to heavy rain

and flood events. Therefore, we seek to improve the flood detection probability by exploiting sound as a signal in next work.

## Chapter 6: Environmental Sound Analytics

Sound signals can carry a large amount of information about the environment and surroundings. Humans can easily perceive the sounds associated with scenes such as forests, offices, kitchens, etc., and recognize sources of sounds that contribute primarily to the background, such as birds, people, utensils, etc. The ability to extract this information automatically has enormous potential in several applications, such as searching for multimedia based on its audio content, making context-aware mobile devices, robots, cars, etc. They can be further enhanced into an intelligent monitoring system to recognize activities in their environments using acoustic information [94]. Intelligent monitoring can be crucial during a natural disaster, including floods, earthquakes, hurricanes, snowstorms, and severe thunderstorms.

Flooding is one of the most severe yet common forms of natural disaster. Therefore, there is a great deal of interest in preventing the damages caused by the flood. An early accurate estimation of the flood can lead rescue teams to help the people affected areas mitigate the damage of flooding. Sound-based intelligent monitoring could be one such tool for early detection. In this work, we describe one such system developed to extract information in the context of a flooding stream. We

show how information extraction from the surrounding can detect the environmental conditions related to the flood event.

Historically, flood detection has been done using hydrology and hydraulic model from civil engineering. Such models are site-specific, resource-intensive, and lack generalization. Automated processes are quick, cost-effective, and monitored from a distance compared to traditional methods, which require labor, in-person visits, and longer processing time [71]. We further argue that the detection model proposed in this work is compact, economical, and scalable. We foresee these systems being widely applicable not just in disasters but also in many other applications.

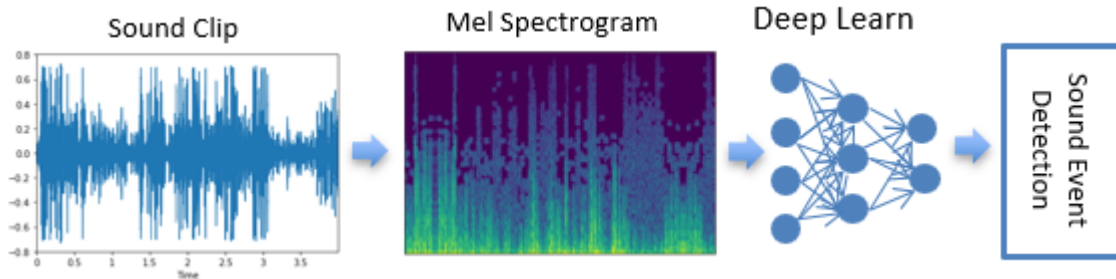


Figure 6.1: Sound Event Detection-Approach

A broader goal of this work is to produce a context-aware computer system capable of processing real-world sound scenes of moderate complexity. The system should prepare an abstract representation of the sources in the sound and classify unseen data. Simply put, the systems should perform Sound event detection (SED). Typical SED framework is shown in Figure 6.1. The objective of SED is to automatically identify the sound events, i.e., to associate distinctive concepts or label for sound segments. SED aims to detect each sound event’s onset and offset times in an audio recording and associate a label with each of these events [170]. Sound events

are good descriptors for an auditory scene as they help describe and understand human and social activities.

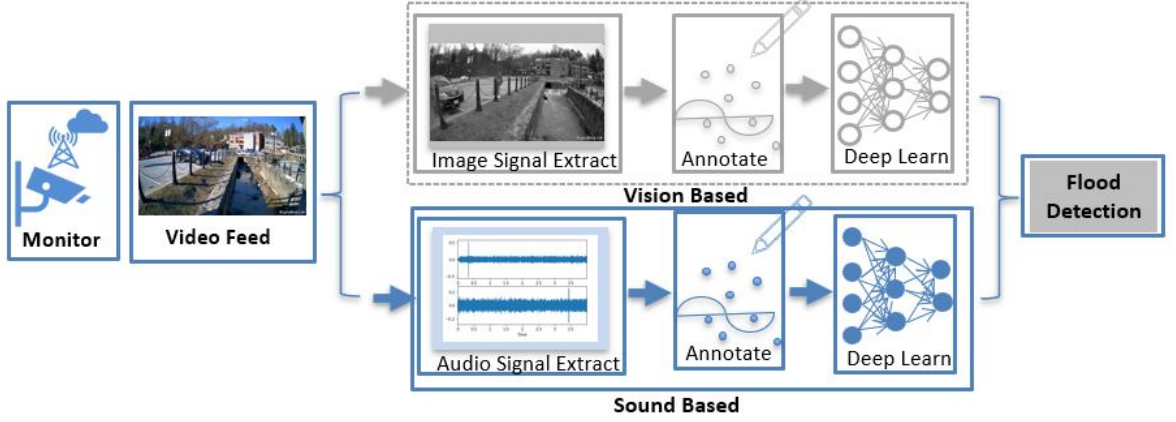


Figure 6.2: Flood Detection Pipeline

The major components and the flood detection pipeline are shown in Figure 6.2. The top part of the pipeline diagram in Figure 6.2 shows the vision-based flow (Chapter 5), and the bottom pipeline depicts the work described in this chapter. This work extends some of previously deployed Flood detection System using machine vision techniques. The major components and the flood detection pipeline are shown in Figure 6.2. Flood detection paradigm in our previous work [18, 22, 20, 19] was to use various techniques of computer vision. The flood detection was primarily done using images and computer vision models until this work. The top part of the pipeline diagram in Figure 6.2 shows the vision-based flow, and the bottom pipeline depicts the work described in this paper.

SED systems like ours are usually designed for specific tasks or environments. There are several challenges in extending the detection system to handle multiple settings and a large set of events. The most prominent challenge is the presence

of multiple sound signals embedded within the event (e.g., heavy wind and rain). Therefore, we propose reducing the search space by providing the context in the sound event detection in the same manner as humans do [100]. Therefore, we have limited our experimental design to a given context of adverse weather such as flooding or detecting the wind conditions in a natural environment.

Unlike humans, computers cannot readily perform SED without prior knowledge and pattern recognition capabilities. To that end, we transform the raw sound signals into their time-frequency representation (Mel-Spectrogram) then employ deep learning models to learn the mapping between this representation and the target sound events. The high-level approach to sound event detection is shown in Figure 6.1. We envision SED-based models as auxiliary systems deployed as a trigger to more resource-intensive high accuracy models. Such systems can be easily integrated with a network of a distributed system consisting of cameras, networking devices, and solar-powered field deployed sensors (FloodBot). FloodBot can observe the current situation of potential flood areas, infer relevant contextual information, and detect the potential risk using sound signals.

Large-scale weather-related sound recognition system is essential for flood detection. It helps us quantify the possibility of flooding in the vicinity with ongoing weather patterns. Heavy flooding is accompanied by heavy sound event episodes such as thunderstorms, gusting winds, and the high-pitched sound of flowing water. The abundance of surveillance systems deployed in urban settings provides an extensive dataset that could be harnessed to build SED models. However, because

recordings spanning hundreds of hours need to be carefully analyzed and categorized, flood detection and weather categorization remain almost an impossible task to be done manually. Therefore it seems natural to look at ways to automate the process. We collect video feed from the site, generate features to detect different flood-related sounds, and evaluate their performance.

## Problem Overview

We pose the Sound Detection Problem as a supervised classification problem where a set of training recordings  $\mathcal{M}$  and labels:  $\{c_m\}_{m=1}^M \{s_m\}_{m=1}^M$  is provided. Let  $\{\gamma_q\}_{q=1}^Q \in \mathcal{Q}$  possible categories  $\in \{\text{Low Wind, High Wind}\}$ . Each label  $c_m \in \mathcal{Q}$ , and we define a set  $\Lambda_q = \{m : c_m = \gamma_q\}$  that identifies the signals belonging to the  $q^{th}$  class. We train the model offline to learn the statistical properties of different classes and test them against unlabelled recordings  $s_{new}$ . Let  $\mathcal{D}$  be the length of each frame,  $s_{n,m} \in R^D$  indicates the  $n^{th}$  frame of the  $m^{th}$  signal.

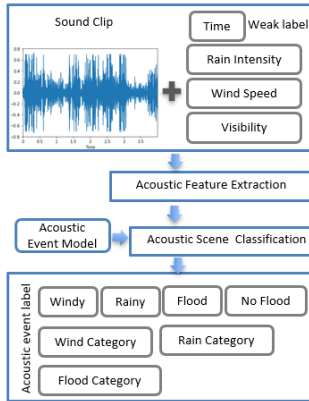


Figure 6.3: Sound Detection Framework

Typically,  $\mathcal{D}$  is chosen so that the frames duration is about 4 seconds. Instead of using wave form in the time domain, we extract a sequence of features through a transform  $\mathcal{T} : \mathcal{T}(s_n, m) = x_n, m$ , where  $x_n, m \in R^K$  indicates a vector of features of dimension  $K$ . Since  $K \ll \mathcal{D}$ .  $\mathcal{T}$  also causes dimensionality reduction. Let  $x_n, \Lambda_q$  indicate the features extracted from the signals belonging to the  $q^{th}$  category. The function  $\mathcal{S} : \mathcal{S}(\{x_n, \Lambda_q\}) = \mathcal{M}$  learns the

parameters of a statistical model  $\mathcal{M}$  that describes the global properties of the training data. Once the training phase has been completed, and a model  $\mathcal{M}$  has been learned, the transform  $\mathcal{T}$  is applied in the test phase to a new unlabelled recording  $s_{new}$ , leading to a sequence of features  $x_{new}$ . A function  $\mathcal{G} : \mathcal{G}(x_{new}, \mathcal{M}) = c_{new}$  is then employed to classify the signal, returning a label in set  $\{\gamma_q\}_{q=1}^Q$ . We show the graphical representations steps described above in Figure 6.3. The sound clips are labeled using weakly supervised techniques the passed through the deep learning models to categorize the sound and event association.

### • Mel-Spectrogram

Most audio-based machine learning and deep learning, in general, are based on sound data represented in Mel spectrogram, Log -Mel spectrogram, and Mel Frequency Central Coefficients(MFCC) [117, 111, 130].

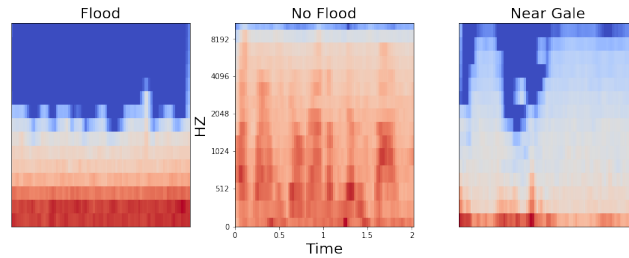


Figure 6.4: Log Mel-Spectrograms

We use the log Mel spectrogram to train the deep neural network in our experiments. For other types of machine learning, such as Gaussian mixture models and hidden Markov models, an additional preprocessing step to generate Mel frequency central coefficients (MFCC) is performed. In Figure 6.4, we show some examples of Mel-spectrograms. The spectrograms depict how the sound pattern differs in dif-

ferent events. It is observed that the typical “No Flood” spectrograms event seems normally distributed versus the Windy (Near Gale) sound spectrogram. We extract the audio features and transform them into feature images, so there are three channels like traditional color images. Spectrograms capture both time and the frequency associated with the raw sound wave for a better data representation based on short time for Fourier transform (STFT). STFT preserves the time domain data as a long vector computed using the wave’s length (time), the sampling rate, and window size. Spectrograms are represented in hertz, which does not perceive the well human auditory system. We convert the linear frequency scale to the logarithmic Mel-scale and pass-through filter bank to get the eigenvector. These eigenvalues can be roughly expressed as signal energy distribution on the Mel-scale frequency. The output is saved as an image shown in Figure 6.4 to train the convolutional neural networks for their pattern recognition.

### *i) Incomplete Supervision*

For incomplete supervision, we use human annotation on a limited data set. This type of annotation is a sub-type of weak supervision called Incomplete Supervision. Incomplete supervision involves the situation where we have a small amount of labeled data and abundant unlabeled data. The labeled to unlabeled data ratio is insufficient to train a good learner.

Formally, the task is to learn  $f : (\mathcal{X} \mapsto Y)$  from a training data set  $\mathcal{D} = \{(x_1, y_1), \dots, (x_l, y_l), x_{l+1}, \dots, x_m\}$ , where there are  $l$  number of labeled training examples (i.e. those given with  $y_i$ ) and  $u = m - l$  number of unlabeled instances. For

the experiment we manually labelled 20% of the video frames by replaying them at our lab.

## ii) *Inexact Supervision*

Inexact supervision is the process of annotating the dataset with imprecise certainty. Some supervision information is given for the inexact supervision but not as exact as desired. A typical scenario is when only coarse-grained label information is available. For example, in the problem of sound detection, the objective is to build a model to predict the type of sound event such as heavy wind, rain, etc.

However, that same sound can also be associated with other sounds, such as a flowing river. One video frame captured by the system can have many sound event patterns, and the sound wave extracted could represent more than one event. Even with human annotation, there could be ambiguity and subjectivity.

Layers	Output Size
<b>Input: 1 x 128 x 431</b>	
3x3,32,BNx2	1 x 128 x 431
3x3,32,BNx2	32 x 128 x 431
3x3,32,BNx2	64 x 64 x 215
3x3,32,BNx2	128 x 32 x 107
3x3,32,BNx2	256 x 16 x 53
Dense	500
Dropout-18	0.5
Dense	3
<b>Output : 3</b>	

Table 6.1: CNN Architecture

Formally, the task is to learn  $f : (\mathcal{X} \mapsto \mathcal{Y})$  from a training data set  $\mathcal{D} = \{(X_1, y_1), \dots, (X_m, y_m)\}$  where  $\mathcal{X}_i = \{\mathbf{x}_{i1}, \dots, \mathbf{x}_{i, m_i}\} \subseteq \mathcal{X}$  is called a bag,  $x_{ij} \in \mathcal{X} (j \in \{1, \dots, m_i\})$  is an instance,  $m_i$  is the number of instances in  $\mathcal{X}_i$ , and  $y_i \in \mathcal{Y} = \{\mathcal{Y}, \mathcal{X}\}$ .  $\mathcal{X}_i$  is a positive bag, i.e.  $y_i = \mathcal{Y}$  if there exists  $x_{ip}$  that is positive, while  $p \in$

$\{1, \dots, m_i\}$  is unknown. The goal is to predict labels for unseen bags. This is called multi-instance learning [168].

## 6.1 Model Architecture

The network we used consists of ten convolutional layers and two fully connected layers. For hyperparameters, we chose the learning rate as 0.002, dropout as 0.5, and batch size as 16. Full CNN Layers and their input-output parameters are shown in Table 6.1. Since we are focused on detecting the readiness and quality of data representation, we use the same CNN architecture for all experiments and focus more on experiment variations than model variations. We resize the mel-spectrogram images to 128 x 431 in size and then pass the image to a CNN for image classification.

## 6.2 Implementation

This section describes our implementation, starting with the data collection, preprocessing, feature generation, and data annotation before describing the deep learning model.

### 6.2.1 Data Collection

We drive our process based on three data sources: the field camera’s video, the real-time weather data API, and the manually labeled dataset for verification. Data is assimilated into their databases and made available for deep learning models, as described below. Figure 6.6 summarizes the total dataset and the observed weather pattern during the video recordings.

#### i) Flood Audio Database

The FloodBot implementation starts with real-time video acquisition from the deployment. These videos are transferred into the cloud for pre-processing. The Flood video database contains videos captured in various weather conditions for more than a year. Figure 6.5 shows the sample images captured during various ambient conditions from the testbed.



Figure 6.5: Flood-Watch Recordings

#### ii) Weather Database

We also collected the weather data from a public weather application interface during the same deployment time frame. We found the weather in the area when we

captured those images. The weather data API allows us to extract weather at that particular location based on the latitude and longitude of our camera. We collect and store these records by minutes in our database.

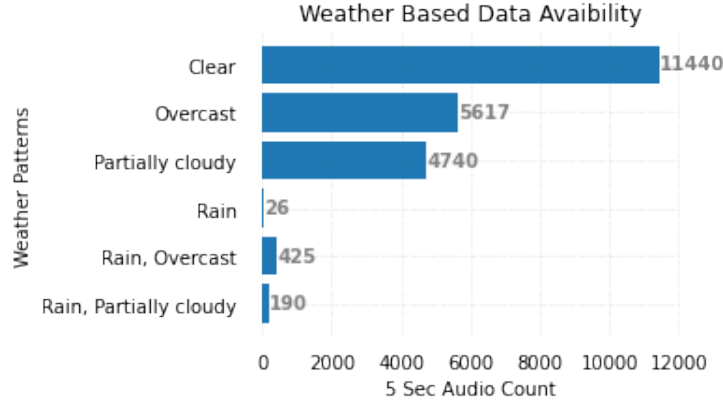


Figure 6.6: Sample Data points & Weather Distribution

### iii) Temporal Sound Data

We establish a temporal join between a video’s recording time and weather timestamp to collect the weather condition of the site to our audio database. With a video to weather time tracking, we can visually observe the weather condition at the site (through video feed) and from data released by meteorological weather stations via their APIs. Both video feed and weather data have been rounded to their nearest five-minute time binning to capture and assign weather reading to as many videos as possible.

## 6.2.2 Pre-Processing

The Average video clipping size captured by our system is 30 seconds and hence the audio frame too. We first extract the sound from a video entirely and

then shorten it to 4-sec frames each. Our preliminary experiments found that a 4-sec frame length is optimal from data storage volume and representation without cluttering them with uncorrelated sounds. We also maintain the splits and their source to locate them in different folds for cross-validations.

### 6.2.3 Feature Generation

Feature generation is essential in analyzing and finding relations between different things. The models cannot directly understand audio data provided to convert them into an understandable format feature extraction. It is a process that explains most of the data but is primarily understandable by the machine. We describe the feature generation steps in Algorithm 7.

---

**Algorithm 7:** Feature Generation

---

**Input:** Video Frames

**Output:** Sound Features, Sound Event Representation

- 1: Capture Video at  $t_i; t_i \in \{\text{Natural Weather Occurrences}\}$
  - 2: Extract Audio from Sample  $\mathcal{Y}_i$
  - 3: Split the sample into  $y_i$  where  $t_s = 4$  sec
  - 4: Compute Mel-Spectrogram  $\mathcal{M}_i$  for each  $y_i$
  - 5: Normalize  $\mathcal{M}_i$
  - 6: Persist Normalizes Image features for Deep Learning
- 

Therefore, feature extraction is precursory for any classification, prediction, and or recommendation algorithms. Creating a feature is crucial yet challenging because it involves numerous preprocessing steps. We discuss the steps involved in generating features in detail in this section. Audio data usually have complex features, so it is necessary to extract useful features to recognize the audio. The

Mel-Spectrogram is one of the efficient methods for audio processing, and 8 kHz sampling is used for each audio sample. We used a Python package called Librosa for data processing with parameters (n fft =1024, hop length = 512, n mels = 128). Then we use Librosa’s power to db function to convert the power spectrum (amplitude square) to decibel (DB) units.

**i) Data Augmentation**

Data Augmentation is a compelling method used to represent the data better. The augmented data is expected to be a more comprehensive set of possible data points, thus minimizing the probability of disparity between the training and validation set. As seen from Figure 6.6, the data of our interest is quite limited. Naturally, adverse weather such as rain is comparatively less than the clear regular days.

			We show an example of a skewed ratio of majority
Category	Fold	Count	to minority samples, e.g., 11K clear days versus 641
Low Wind	7	2070	rain event-related data points. This is a classic ex-
Low Wind	9	227	ample of class imbalance. We used data Augmen-
High Wind	1	640	tation techniques called an oversampling solution
High Wind	7	635	to alleviate the problem. We use a hybrid of over-
Low Wind	1	2102	sampling and under-sampling methods to achieve

Table 6.2: Sample Data Distribution (Folds)

a balanced dataset for training and testing. During the over-sampling process, instances from the minority class (641 samples) were randomly duplicated (via replication). We also randomly remove sample data points from the majority class during the under-sampling process.

Cross-Validation (CV) is a machine learning technique used when the available datasets are limited or not balanced like our case. Therefore, instead of training a fixed model only once with a train/test split, we iterated through several models on different data portions and validated the model on hold-outs. K-Fold is one common CV approach that we have implemented in our experiments.

First, we create a well-defined model is by normalizing data, selecting features, tuning parameters, as explained in the earlier sections. We split the clippings (4-sec sliding windows) into their folds from their full-length audio. A 40-Sec audio clip, for example, would be divided into ten 4-sec clippings, each clip belonging into its fold, ensuring that clipping from the same full-length sample would never be on the same fold. Sample data points and their cross-validation fold counts are shown in Table 6.2.

## **ii) Data Annotation**

Supervised learning tasks such as classification and regression has achieved great success in various Machine learning problem. The success of supervised learning is due to the availability of many training examples, each corresponding to an event/object. Most successful techniques, such as deep learning, require ground-truth labels for an extensive training dataset. In many tasks, however, it can be challenging to attain substantial supervision information due to the high cost of the data labeling process. Thus, machine learning problems are often re-framed into weak supervision [168]. We use two types of Weak Supervision processes, incomplete supervision and inexact supervision.

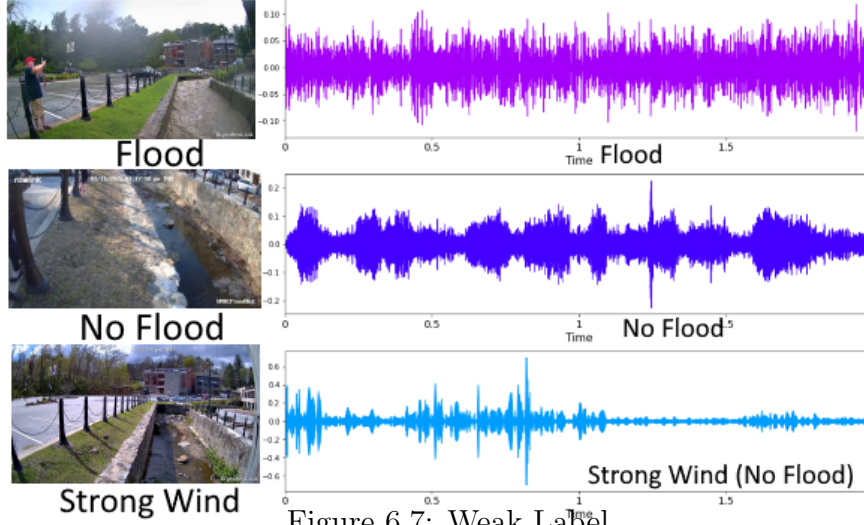


Figure 6.7: Weak Label

We show sample output of Weak (Inexact Supervision) based data annotation example in Figure 6.7. We use the temporal join (time-based) in-exact supervision to bulk label our training dataset. One 4-sec frame can only belong to one sound event detection at a given time. Unable to detect that class during test/validation is regarded as a miss-classified data point.

### 6.3 Experimental Results

The experimental setup is shown in Table 6.3. We first start with a baseline model where the objective is to detect the presence of "Gusting Wind Sound" in FloodBot captured videos. Once the audio is extracted and pre-processed as described earlier. We use the Weak Supervision to annotate the event.

Table 6.3: Experiment Design

ID	Experiment	Criterion	Type
1	Wind Event	12 MPH Wind	Binary
2	Flood Event	24-H Cum.Rain	Binary
3	Wind Category	Beaufort Measures	MultiClass
4	Rain Category	24-H Cum.Rain Range	Multiclass

We envision that the outcome of this model can be used to enhance flood detection capacity when other (vision-based) approaches either fail or have difficulty detecting the flood. We also argue that the model outputs could serve as a triggering event to detect or enable resource-intensive flood detection hardware and software. To that end, we designed four experiments and progressed their complexity, and evaluated their performance.

## Binary Detection

We divided the experiments into two categories; Binary detection to detect the presence or absence of prevalence signal. We use heavy wind sound and cumulative rainfall amount to detect the wind and flood events under the binary classification experiments described below.

### *i) Wind Detection*

We train the model to detect high-speed wind sound in the video for this experiment. We define this experiment as a binary classification problem. The recorded wind speed varies from 0 MPH to 34.2 MPH. The mean wind speed was 7.9 while the 25% quartile is at 5 MPH and 75% (Q3) at 11.8 MPH. Therefore, we use the binary classification of wind versus no wind at a 5 MPH threshold (No Wind/Wind) to classify wind sound in the video. We also use 11 MPH as another threshold to detect the wind speed (High Wind/Low Wind).

We show the classes and their distribution in Figure 6.8. Our goal with this ex-

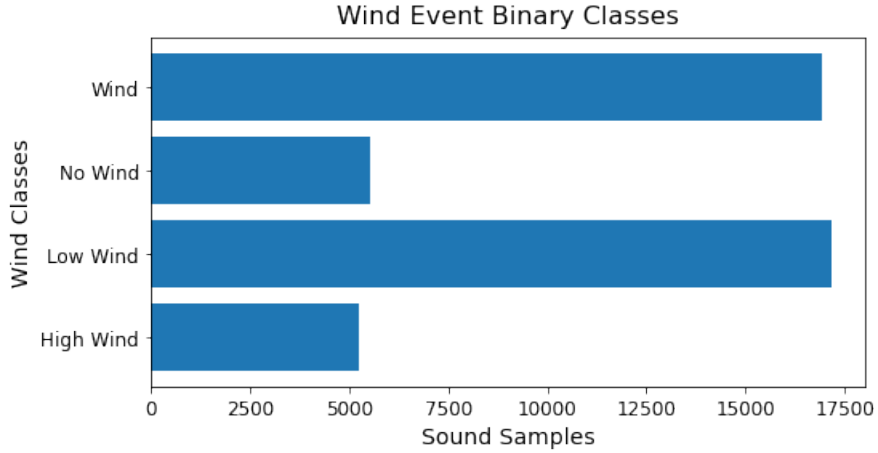


Figure 6.8: Wind Event Classes

perimental setup is to evaluate the preliminary natural sound event detection (wind speed) capability of our deep learning model(s), which is often directly associated with rainfall.

We show that the model performed well in binary detection of the Wind Event. We are motivated by the fact that detection of Wind gust is an important first step in deploying the flood detection. It is also imperative that heavy rain events and thunderstorms often produces the heavy gusts and vice versa.

### *ii) Flood Detection*

Flood Event Categorization is probably the most challenging setup from all experiments. Unlike wind speed, flooding is not an instantaneous event and often takes a prolonged period of rain to generate a flood. Therefore, we first computed the 24-hour cumulative rainfall from five-minute precipitation recorded by weather data. It is standard practice to estimate flooding events based on 24-hour cumulative rain. A threshold of 0.6 inches of recorded rainfall over 24-hour cumulative precipitation seems to be a reasonable threshold when the stream shows significant flow pattern

changes to be classified into flooding versus not flooding stream. We show the results from the binary flood class in Figure 6.9.

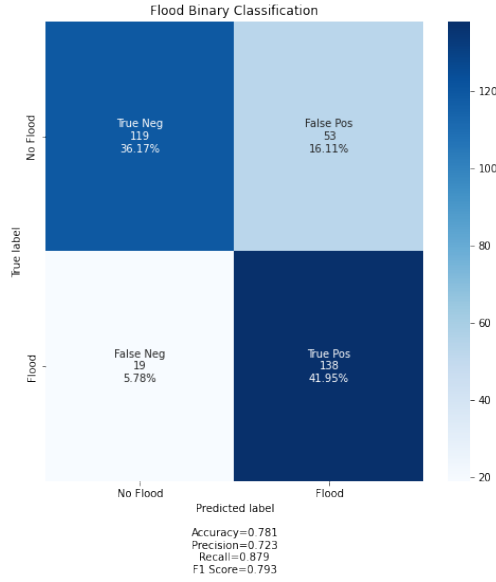


Figure 6.9: Results: Binary Flood Classification

88% and F1 Score of 0.79.

## Multi class Detection

We further increase the complexity of our experiment by turning the binary classification into multi-class classification. We explore the model performance for three classes for Flood categories and four for Wind speed.

### *i) Multi class Flood Event*

We use similar computation and inexact supervision techniques to label our

Next, we experiment with a binary flood detection problem. The faster the water flows, the larger the sound it makes. We correlated the sound intensity recorded to a water velocity and delineated our classes. Furthermore, the observed pattern of the sound wave during calm conditions is smooth, long, and continuous versus exhibiting high frequency and energy during fast flow flooding. The overall accuracy is 78% with a precision of 0.72% and recall

data and evaluate the flood detection models. After removing a few outliers of rain event data, we categorize the flood into three classes: Flood, Minor Flood, and No Flood. The threshold is based on 24 hours cumulative yielding no flood for less than 0.40 inches of rain, minor flood for above 0.40 inches but less than 2 inches. Above 2 inches of cumulative rain over 24 hours is categorized as flood-producing rain. After the bulk annotation, we manually verified a few hundred images and agreed with their distribution. We were able to achieve overall accuracy of 0.76 for three class

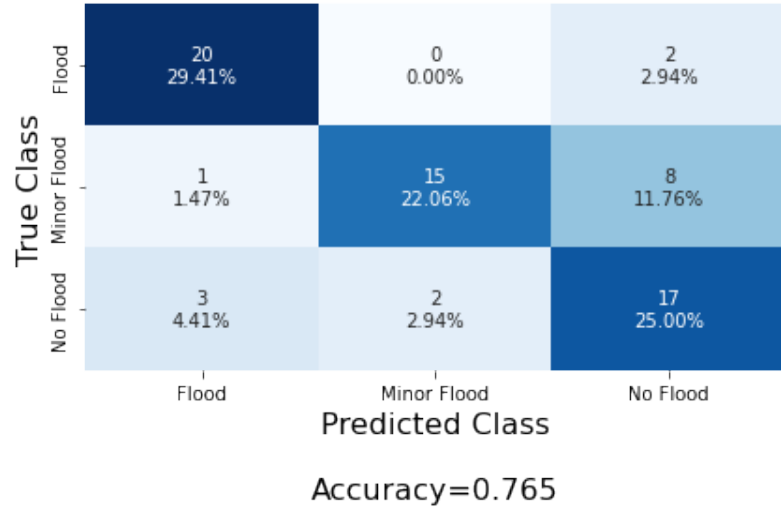


Figure 6.10: Result:Flood Multi Classification

flood detection problem and show the result in Figure 6.10.

Though the precision and recall are less important metrics in context of multi-class classification, we generalize the multi-class problems by summing over rows / columns of the confusion matrix. Specifically, Precision is computed as  $\text{Precision}_i = \frac{M_{ii}}{\sum_j M_{ji}}$ . Recall is computed as  $\text{Recall}_i = \frac{M_{ii}}{\sum_j M_{ij}}$ . Where  $i$  is each sample and  $j$  is total dataset.

## *ii) Wind Speed Categorization*

Rather than identifying the windy or calm conditions, we try to classify the sound event into one of the Beaufort wind scale measures. The Beaufort wind scale measures wind speed according to the wind’s impact on the land and sea. The measure developed by Sir Francis Beaufort in 1805 remains a widely used system to measure wind speed today. We experimented with all 12 Beaufort classes such as Calm, Light air, Light breeze, Gentle Breeze, etc. However, the classification was too granular for acceptable results. Therefore, we created four classes out of the range. The model achieved overall accuracy as shown in Table 6.4.

Table 6.4: Results

Experiment	Accuracy	Recall	Precision	F1 Score
Wind Event	0.72	0.76	0.78	0.77
Flood Event	0.78	0.72	0.80	0.79
Flood Multi Class	0.76	0.77	0.78	0.76
Wind Category	0.70	0.78	0.76	0.75

## 6.4 Conclusion & Discussion

Table 6.4 shows the mean classification accuracy achieved by the proposed CNN. In all experiments, proposed CNNs achieved a mean accuracy of above 0.70% and reached 0.78%, the highest for flood sound detection setup. We did not change the model architecture except reducing batch sizes if the sample sizes were smaller. We opted for the design choice to evaluate the readiness and usability of 2D representations of the audio signal as input. Our preliminary experiments find that the data set can predict the sound events on par with state-of-the-art models where the

trend is still around 80% accuracy.

While there has been a great success with vision-based flood detection, the main challenge arises when the system loses visibility. Low illumination adverse weather conditions can cause visibility loss, often precursory to heavy rain and flood events. Therefore, we proposed this method to improve the flood detection probability by exploiting sound as a signal. Sound signals are also lighter in terms of data volume, thereby reducing the footprint of each deployment unit.

Motivated by the idea of Federated Learning, where data is allowed to remain on distributed devices while a central/global model is trained and shared from locally trained (on-device) model updates. Just in time detection and triggering of SED to either enable flood Computer vision techniques and image-based solutions may have peaked in performing the task at hand, for example, flood detection in our setting. However, vision-based inference requires considerable processing power, especially for real-time data-intensive applications. Cloud computing and inference are a way around this constraint, but that is not always an option. Flood detection is often done where there might be no internet or data transportation possibility. Cloud cost, network bandwidth, and detection time lag impede mission-critical deployment like ours.

However, running computer vision in the cloud heavily limits real-time computer vision applications. We argue that Sound Event detection can be one way to alleviate that problem without investing in computer vision deep learning resources.

Furthermore, relying only on vision-based solutions restrains the detection capability to the camera angle, illumination, and image frame compositions. On the other hand, the sound-based signal is lightweight and more palatable for edge devices.

We presented our work in environmental sound classification for flood event detection using CNN and a Mel-spectrogram. The experiment result and our field-collected real-time datasets show that the proposed experiments can achieve acceptable performance in sound event recognition. We present how the sound detection technique could be used as a supplemental trigger or precursory evaluation toolkit before deploying the relatively resource-intensive models such as vision models. The ultimate goal of our research is to deploy flood detection models into microcomputers, embedded systems and move towards edge computing. Federated learning provides us the means to reach that ultimate goal. Federated learning allows us to detect the flood on edge devices and compute the in-situ flooding condition without data transfer to the remote server or heavy computational unit.

## Chapter 7: Social Media Analytics

This chapter’s primary objective is to discuss FloodBot’s social media integration. We have discussed our approach and design for the physical sensor thus far (image, sound, sensors). This chapter discusses various approaches and emphasizes the critical role of social media during natural disasters. Additionally, we discuss our ongoing FloodBot system design, automation process, and meta-analysis approach for validating user engagements, among other things. In the second half of this chapter, we discuss our work in making FloodBot a conversing agent and describe it in context of Chatbot design.

We begin this section by recognizing Alexa, Google Assistant, Siri, Cortana, and Bixby as extended family members. They have seamlessly woven their presence into our life and aided us in our daily chores. The situation we live in today can be perceived as the realization of the noble vision of ubiquitous computing presented by Mark Weiser in 1991 [154]. The term Internet of Things (IoT) that was originally coined by Kevin Austin at MIT in 1999, is now a household commodity. IoTs integrate a large number of technologies and connect things or objects to the internet, generating or collecting the data along the way. WSNs powered by the IoTs

can collect and transmit data through the use of machine-controlled social media curators (bots) or their human counterparts via social media posts.

IoT can collect data and propagate them either by machine-controlled social media curators (bots) or their human counterparts via their social media posts. The union of social media posts and the IoTs generated data fusion is extending the IoT world into social media world. Social media integration into vanilla grade IoTs makes the cyber physical system. IoT and bots are a valuable member of ubiquitous computing society; however, the main challenge in their successful integration lies in their reliability and resiliency, especially when in need. We argue that the impediments around wider acceptance of IoTs and their transformation to become social IoTs lie in their weakness posed by i) creating temporally insignificant messaging or annoying users with pre-programmed non-contextual information, ii) missing a two-way communication and human-like conversing capability and iii) inability to improve the quality of the data and augmented information produced by IoTs.

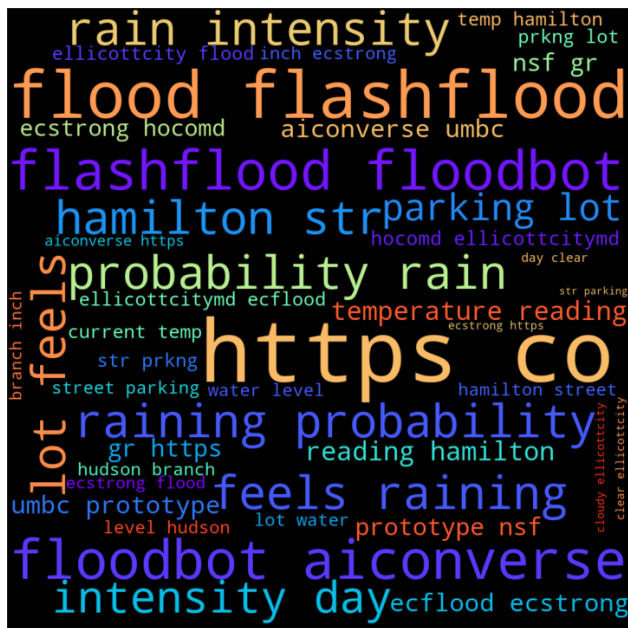
Our thesis proposes scalable and contextually aware deep learning architecture that can ingest data from multiple sources and assess the severity of an emergency event. We envision an automated artificially intelligent system that can look at various data stream and decide upon adversity of the area. In addition to sleek architecture design, our system will be backed up by qualitative, quantitative measures and machine-learned data validation techniques. Finally, we see our model as a hands-off cyclic AI unit where the data from multiple sensors (social, physical) are ingested, augmented and disseminated back to the community as value-added

information. To reach that goal, we looked at FloodBot's we have two objectives : first the system had to be integrated into social media (to generate and consume) social media content. Next, it had to possess artificially intelligent communication capability to converse with human. We group these two work under two sections i) Social Media Integration and ii) AI Converse.

## 7.1 Social Media Integration

Mobile phones are the most widely used means of information dissemination due to their increasing ubiquity and ease of communication. Nowadays, social media acts as a glorified microphone service, disseminating near-real-time events. When people face a natural crisis or disaster, they immediately turn to social media to post about their safety or help requests. Similarly, crisis responders use social media to assess situations efficiently and allocate available resources. Thus, any automated middle-tier application capable of extracting, validating, and inspecting data would benefit both parties.

Wireless Sensor Networks (WSNs) and cyber-physical systems are robust, cost-effective, adaptable for remote monitoring. They facilitate faster data transfer, including high-resolution information and processing, allowing further analysis and timely alerts for disaster management applications. However, these system lacks the human touch or feeling around the unfolding events. The cyber-physical systems often serve as a data point for complex analytics unable to parse human sentiments.



Twitter and other social media platforms provide valuable data to surveillance systems. Nowadays, social media platforms such as Twitter, Facebook, and YouTube provide many information dubbed social data. Individuals share their news and perspectives on social media and discuss with their friends, rela-

tives, and other members of their virtual network. They respond to these events positively, negatively, or neutral on that post. These data can be sensed and analyzed to identify patterns of human behavior in cities for the benefit of urban authorities and citizens, particularly during natural disasters, for traffic forecasting, urban planning, and social science, among other applications.

We use Twitter account with screen name `umbc_floodbot` to propagate real time information from the site . In the subsequent section we describe our design and integration in more detail. Framework describes high level services and component involved in automating the FloodBot. Next we describe our implementation detail and show results from our meta analysis on the system.

### 7.1.1 Social Media Pipeline

We describe the pipeline for Social Media Integration using Figure 7.2. There are four major building blocks in our social media integration pipeline. We describe their function and setup briefly.

#### *i) Data Collection*

These are our physical sensors capturing field data. The physical sensor deployed on field has two sensors attached to it. Water Level Measuring unit to measure the stream flow depth and Camera unit to record video.

#### *ii) Storage & Manage*

Our data collected on site are raw and the device are not powerful enough to perform compute locally. Therefore, the site collected data are shipped and stored in our remote VPS (Virtual Private Server) for cloud compute.

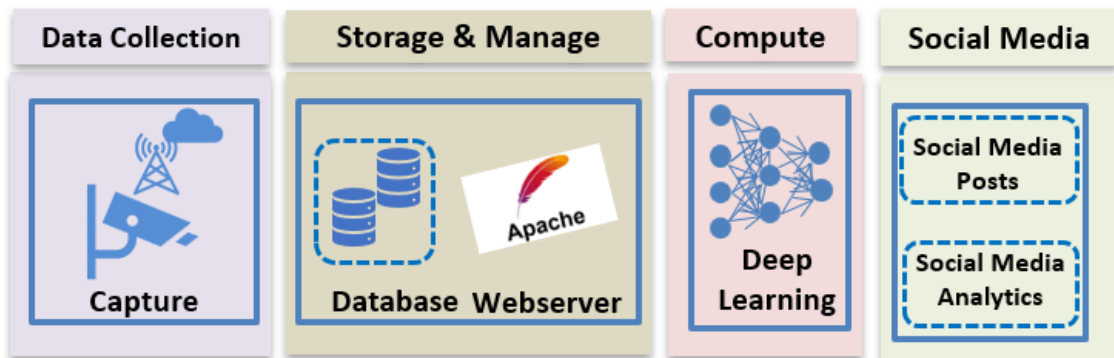


Figure 7.2: Social Media Integration Pipeline

We have provisioned a VPS (Virtual Private Server). VPS is a hosting service that uses virtualization technology and provides us with dedicated (private) server like environment at a fraction of cost. VPS service allows us to build the system

perfectly suitable to our research need. We provisioned the with CentOS-7.6 64 bit Operating System, has a 40 GB storage (Hard Disk). We installed Apache web service for web hosting and mySQL database for database storage.

### *iii) Compute*

Since, the compute and Deep learning model needs GPU we opted for GPU CPU computational resources provided by Google Colab. Since computer vision tasks requires substantially large memory we subscribed to their pay as you go plan and provisioned our compute to run on on GPU either K80 or T4 with 16GB GPU Memory and 2 CPU Cores, 12 GB RAM and storage Disk Space 358 GB. Virtualization and compute provided by GPU helped us iterate and improve our Deep Learning research and complete this thesis.

#### 7.1.2 FloodBot in TwitterVerse

We have integrated FloodBot into Twitter Universe (TwiterVerse) and propagate real time information from site 24 x 7. Twitter was a good choice for this research for two reasons: i) there has been a lot of work done on Twitter accounts and their usage during a natural disaster [58], and ii) the Twitter API makes it easy to post, retrieve, and analyze data. In times of disaster, people need to get information quickly. Therefore, Twitter was also a good choice as messages are forwarded or (“retweeted”), it is possible to see how quickly information spreads in real-time.

Twitter users usually post messages to enhance situation awareness. In addition, tweets are trustworthy and possess valuable information content, which backs up the idea that Twitter has a good chance of being useful in situations where a quick emergency response is needed.

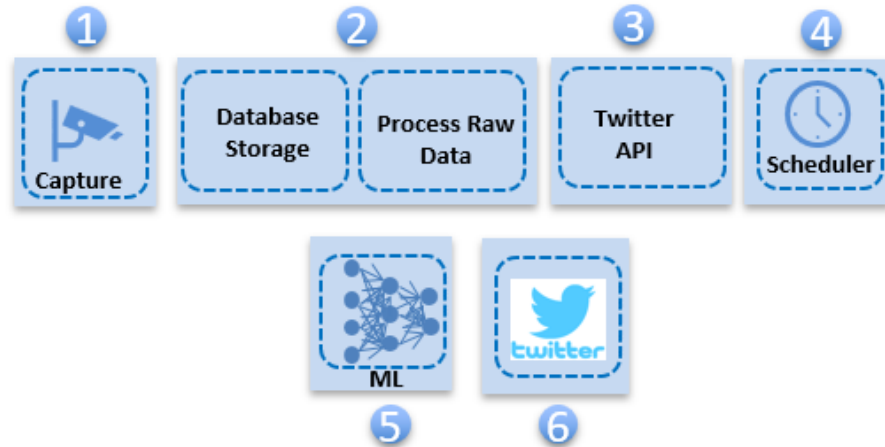


Figure 7.3: Social Media Integration Components

FloodBot generates the data via sensor recordings or as an output from Deep learning (compute) resources. Those information as posted into our Twitter account as Social media postings. We then perform retrospective analysis on usage looking for patterns relevant to our research (flood detection) and explore human aspects/connectivity the social media (Tweet posts) that our system creates. We have automated Twitter account to post the images, weather conditions and often output from Deep Learning analysis in every six hours. The components designed to automate the FloodBot is shown in Figure 7.6.

We enumerate through the components that automate the Twitter posting every six hours.

1. Capture : Site Deployed Camera and Sensors

2. Storage : Database Storage & Raw Data Processing (Image, weather data and, sensor integration, make content ready for posting)
3. Twitter API : Integration with Twitter API (OAuth2 validation Social Media Curating)
4. Scheduler : FloodBot is automated to tweet every 6 hour.
5. ML : Machine Learning
6. Twitter : Social Media posting

We have created parameter based Tweet generation algorithm, that takes relevant information such as weather, time, expected rainfall etc. We show example Tweet thus created below.

*Current Temp at Hamilton Str. Prkng Lot D on 02/16/2022, 06:00 is: 24.74.  
It feels like 18.39. Raining Probability 0.0. Rain Intensity : 0.0 The day is  
Partly Cloudy #EllicottCity, #Flood, #Flashflood, #FloodBot, #AIConverse,  
#UMBC Prototype/NSF GR#1640625.*

Once FloodBot connects to social media, we want to investigate its communication capabilities. More specifically, in real-time and using the artificial knowledge we have collected thus far (image, sound, sensor data, etc.). FloodBot requires natural language processing and capabilities to respond effectively to the user. It must also consider additional elements like context, memory, intelligent comprehension, prior experience, and customize the user experience.

We therefore explore the range of conversational interfaces and concentrate on strengthening the FloodBot's communication power using artificial intelligence. We

investigate the design and deployment of conversational interfaces for AI Conversing. We employ text-based conversational software agents with a deep learning capability and explore FloodBot’s capabilities to hold a conversation and manage the discussion to accomplish a goal (disseminate relevant information from crisis location).

## 7.2 AI Converse

People use many similar terms to describe conversational artificial agents (AI) such as a bot, chatbot, chatterbox, dialogue system, personal assistant, etc. However, the underlying fundamental AI powering these systems do not differ much. In this work, we discuss the design, development, and deployment experiences of one such conversational AI deployed at a potential flood hazard area. FloodBot is a goal-oriented chatbot. As opposed to the general chat-oriented dialog system (chatbot), the purpose of FloodBot is to maintain small-talk with a human user. In other words, FloodBot is a domain-specific goal-oriented dialogue system that is meant to assist in resolving practical, real-life problems, primarily to inform and answer queries about potential flood hazards in the area.

The last decade has seen an overwhelming amount of success in the area of computer vision. The output from the computer vision has been effectively used in many real-world applications. Building a computer vision model entails plenty of complexity and is extremely powerful, yet the output and adaptation of the vision-based model are somewhat limited. The output of computer vision can be a valuable

asset in developing an end to end information propagation and processing system. So we harness the power of computer vision model and use its output to design an easy to understand conversational AI or dialogue system.

Various researches in computer vision analyze natural disaster data. The main themes of these works are post-disaster meta-analysis and the ability to make sense of the losses using damaged pictures and artifacts. Given the disaster has already happened, the usefulness of such a system is inadequate. Such analyses are often performed in a specific disaster, which further limits the re-usability and generalization of that approach. Here, we use “Flood Risk” as our test-bed, but the method and the system we design are easily transferable to other disasters.

IoTs and various sensors have been deployed widely to monitor the natural conditions and imminent risks such as floods, earthquakes, and landslides. However, the endpoint of these systems is often a database server or cloud storage. The data and valuable information that is gathered, often get buried behind the technological complexity. The database brings some structure to the data, but they still lack easy accessibility. Hence, there are many exciting works in trying to build a natural language interface to read the data easily [8] from the database. A natural language interface to a database is a system that allows the user to access information stored in a database by typing requests expressed in some natural language (e.g., English). Therefore, in this work, we also propose such Natural Language Processing and interface units between humans and computers to allow effective communications. We argue that a conversational agent could be an enabler that assists in reducing

technological complexity.

Though the use and adaptation of early warning systems are on the rise, their integration with today’s automatic speech recognition system and voice assistance is limited. We have seen how personal assistance such as Alexa, Google Assistant, Siri, Cortona, and Bixby have become an integral part of our lives. However, they can only answer specific queries about general things. There is still a scarcity of the AI-powered chatbots, mainly designed to disseminate information about the disaster at the grass-root level. We postulate that people would be interested to know about the potential danger in the area by asking a simple question. They would rather get short and relevant information instead of having to read a news article or scan through social media feeds.

### 7.2.1 Implementation

In this section, we propose an end to end goal-oriented conversational AI agent that can provide contextual information from a potential hazard site. We posit the conversational agent as a FloodBot capable of seeing, sensing, assessing hazard condition, and ultimately conversing about them. We present our domain-specific FloodBot design-solution and learning-experience from the real-time deployment in a flash flood devastated city that uses state-of-the-art deep learning models. As described in earlier chapters, we specifically used computer vision and pertinent natural language processing technologies to empower the conversation power of the

FloodBot. To deliver such practical and usable AI, we chain multiple deep learning frameworks and create a human-friendly question-answer based dialogue system.

Using flash flood detection as our problem domain, we present a real-time flood monitoring chatbot design. We assimilate contextual information and infuse the conversational AI power of FloodBot using various deep learning models such as a convolutional neural network (CNN), single-shot multi-box object detection (SSD), and Deep Bidirectional Encoder Representation for Transformers (BERT) based language model. We seek to solve the complexity problem by allowing users to ask a simple question and get the information at their fingertips (mobile devices) or with their smart speaker. It also allows users to retrieve relevant information easily. We show the Framework of AI Conversing FloodBot system in Figure 7.4 and then describe our first-hand experience from the real-world deployment of the FloodBot.

### 7.2.2 AI Conversing Pipeline

This section describes the Pipeline for FloodBot and its relevant components. FloodBot is a network of a distributed system consisting of cameras, networking devices, and solar-powered field deployed sensors. Floodbot aims to observe the current situation of potential flood areas, infer relevant contextual information, and converse about the potential risk. We broadly categorize the components of FloodBot into three significant parts: (a) The monitoring unit, (b) the Learning component, and

(c) The AI Conversing component.

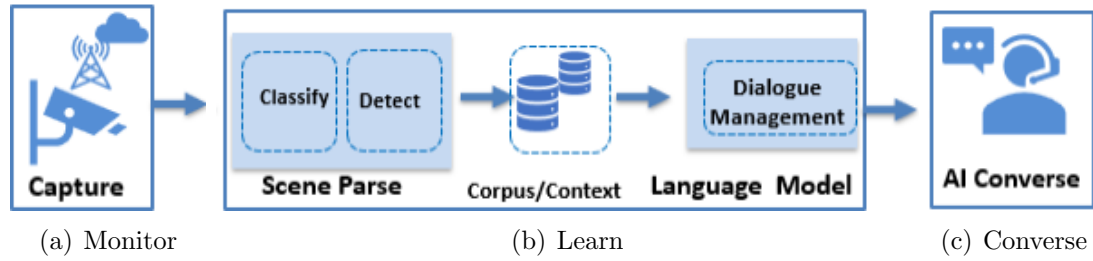


Figure 7.4: AI Conversing Pipeline

### i) The Monitoring unit

The first component is the remote monitoring camera unit connected to the cloud via 4G/LTE network as shown in Figure 7.4 (a). This is camera overseeing stream and capturing still images and video at different times of the day. The camera's frame rate is dynamically adjusted to increase during severe weather conditions to capture more granular data. We have deployed a camera unit connected to the cloud. The camera is programmed to capture videos at pre-scheduled time intervals and transmit data to the cloud.

### ii) Learning Component

The second, more innovative and research-oriented component is the learning component. This component learns from the data that we captured during the monitoring phase along with the three distinct deep learning strategies depicted in Figure 7.4 (b). To achieve an AI conversing chatbot, we implement two different types of information extraction paradigm. The first is the scene Parsing modules, and the second is the Language Model. Information obtained from the location parsing mod-

ules is persisted in the database layer and enriched with dialogue corpus. The next component in the FloodBot pipeline is the language modeling or (NLU/NLP) unit, which prepares FloodBot to converse with humans via chat(typing) or through spoken languages. The Natural understanding unit provides the intelligence required for human-level conversing.

- **Scene Parsing** Scene parsing classifies the severity of flood events and provides meaningful, semantic labels to the scene by identifying object components of the images. Given an image, our algorithm classifies the severity of flood events and provides meaningful, semantic labels to the scene by identifying object components of the images.
- **Corpus/Context** Corpus /Context explores the relationship between FloodBot linguistics characters generated primarily from the scene parsing output. The context for FloodBot is provided by scene parsing content infused with other existing corpora.
- **Language Model** Language model provides the glue to AI Conversing capability. The language model provides the complete capabilities of natural language processing, including language generation and language understanding. This component handles unstructured inputs as a human spoken language (natural language). It converts them into a structured form that a machine can understand and perform a specific action.

Language models are composed of four sub-modules: (1) *Natural Language Un-*

*derstanding (NLU)* module to identify user intents and extract associated information, (2) *State tracker* to track the dialogue state and capture critical information in the conversation, (3) *Dialogue policy* to select the following action based on the current state, and (4) *Natural Language Generation (NLG)* module for converting agent actions to human language responses.

### iii) The AI Conversing Component

Finally, the third component of the FloodBot framework is the AI Conversing component. This unit primarily understands and transforms the human spoken or textual language to machine language and translates the requested information as answers back to the human language for the humans to understand. FloodBot is an AI-powered chatbot that can automate conversations between people and machines (trained model) to automate any other such as Q&A, notifications, event reporting, or risk tracking. The endpoint of our implementation is an *AI conversing*-chatbot. FloodBot is a user interface for computers and humans to communicate. Further expansion of AI Converse can easily lead to a more human-friendly speech recognizable bot system that is trendy nowadays.

We seek to solve the complexity problem by allowing users to ask a simple question and get the information at their fingertips (mobile devices) or with their smart speaker. It also allows users to retrieve relevant information easily. In this work, we seek to turn FloodBot into Practical AI by feeding parsed scene contents from our vision models. Deep Learning frameworks can sometimes be perceived merely as a research problem. We mask the sophisticated computational units and

deep learning model and present FloodBot as a friendly question answering chatbot system. We then use Transfer Learning technique that allows quick and effective deep learning model development.

Thus in this work, we propose a cross-domain transfer learning in a rather uncharted domain of flood detection. We use pre-trained vision and language models and fine-tune them to infer ongoing flood potential and severity. Finally, the contextual Q&A-Data Assimilation facilitates the question and answering power, we also infuse multiple data sources and created a stand-alone elastic knowledgebase for FloodBot.

### 7.2.3 Models

In Chapters 5, we trained computer vision models to identify the objects and Flood severity. We then store the trained vision model’s information into a repository. We then add a natural language processing transformers model to prepare the inference data and build conversational power for FloodBot using the vision model’s repository. During inference, we backtrack the pipeline. We start by querying the repository with a human-understandable question. Our main task during inference is to translate the questions asked by humans into machine-understandable queries and extract information from the repository.

Our approach is somewhat similar to commonly developed in visual VQA or image captioning problems. However, our approach differs from those models on

how we treat the output of the vision models. In a regular image captioning, or VQA tasks, models often use low-level pixel value information and translate them to build the semantic relationship found in that image frame. In contrast, we use higher-level extraction from the vision models and provide input to the language model. We also conceive our method as a modular and detachable pipeline of two powerful models (the vision and the language).

### i) End to End Memory Network

Recurrent neural networks are extensions of the standard feed-forward multi-layer perceptron networks, where the inputs and outputs are sequences instead of individual observations. These networks suffer from a longer computation time and the problem of vanishing gradients. A simple RNN takes over memory from many previous steps, and hence their implementation becomes cumbersome and resource-intensive [26]. For a chatbot, recalling information from the distant past is not critical. A chatbot is essentially a question and answering system, so it only needs to remember the last statement and the flow. Hence we opted for a variation of the encoder-decoder model called End to End Memory Network [140]. Memory networks help chatbots capture the most important fact from the provided information. Algorithm 8 outlines our end-to-end memory learning.

In a memory network, the input sentences are broken down into word vectors and treated as a bag of words (*Input memory representation*). Thus, each sentence (fact) becomes a group of individual vectors  $\{x_i\}$ . Formally, given an input sentence  $\mathcal{X}$  containing words  $\{x_1, x_2, \dots, x_i\}$ , memory networks vectorize the words and store

---

**Algorithm 8:** End to End Memory Network

---

**Input:**User Queries about the event and scene context

**Output:**Chatbot Response(s): Answer

- 1: Collect and enrich Question and Answering Corpus
  - 2: Encode Question and Answer into Word Embedding
  - 3: Compute Memory Vector from the word Embedding
  - 4: Multiply memory vector with word embedding
  - 5: Perform dot product memory and softmax them to find the most relevant fact
  - 6: Repeat {2,3,4,5} for each question  $q_i$
  - 7: Compute the context vector  $c_i$
  - 8: Perform stacked softmax on output and questions internal memory vector
  - 9: Return the most relevant answer
- 

them in memory vectors  $\{m_i\}$  of dimension  $d$  computed by embedding each  $\{x_i\}$  of dimension  $d$ . This results an embedding matrix  $\mathcal{A}$  of size  $(d * V)$ . The exact same procedure is repeated for the question sentence  $q$  set yielding another embedding matrix  $\mathcal{B}$  of the same size as  $\mathcal{A}$ . The internal state of the queries is vectorized into  $u$ . The only remaining part is finding the most relevant answer-sentence to the encoded question. This is measured using the cosine distance or dot product between memory vector  $\{m_i\}$  and question's internal state vector  $u$ .

$(p)_i = \text{Softmax}(u^T m_i)$  where

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(x_j)}$$

and  $(p)_i$  is the probability vector. Similarly the output of the sentences are represented into  $c_i$  and into a matrix  $\mathcal{C}$ . The resulting response vectors from the output memory  $o$  is summed over the transformed inputs  $c_i$ .

$$o = \sum_{i=1}^i p_i * c_i$$

Finally, to get the prediction for a single question/answer case, the sum of the output vector  $o$  and the input embedding  $u$  is passed through a final weight matrix  $\mathcal{W}$  (of size  $V * d$ ) and a softmax to produce the predicted label:

$$o = \text{softmax}(\mathcal{W})(o + u)$$

A very related yet more complex method is proposed by [11] called attention model. The working of the encoder-decoder model/attention model is depicted in Figure 7.5. At a high level, the attention mechanism helps the next step in recurrent neural networks find the most relevant words based on the dot product (distance metric) and softmax over it. The concept of context vector is analogous to the memory vector in end-to-end memory network [140].

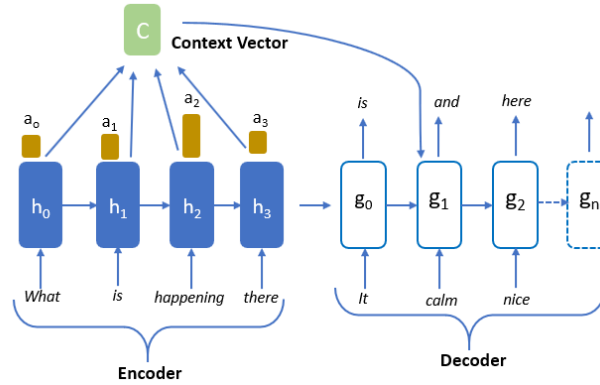


Figure 7.5: Attention Mechanism

The attention mechanism ensures that the words we want to focus on are kept as is, and the unnecessary words are discarded.

## ii) BERT for Q&A

It is a multi-layer bidirectional Transformer encoder based on [147] and two

major model sizes are BERTBASE (12 Layers, 768 Hidden nodes, 12 Attention head, Total 110M Total parameters) and BERTLARGE (24 Layers, 1024 Hidden nodes, 16 Attention head, 340M Total Parameters). BERT has been used to solve various Natural language problems. We specifically use the *Fine-Tuning of BERT: For the question answering task*. It is fine-tuned on SQuAD (Stanford Question Answering Dataset), contains 100k crowdsourced question/answer pairs from Wikipedia. It predicts the answer text range/span in the context data given to the system. It computes the embedding  $A$  and  $B$  for question and context, respectively, then tokenize every word. Finally, it calculates the start score ( $S$ ) and end score ( $E$ ) of the answer span by the probability of being the start word at location  $i$  can be computed as a dot product between  $T_i$  (the final hidden vector for the  $i_{th}$  input token) and  $S$  followed by a softmax over all of the words in the paragraph.

$$P_i = \frac{e^{S.T_i}}{\sum_j e^{S.T_j}}; S.T_i + E.T_j \quad (7.1)$$

For the end of the answer, span score uses a similar formula and for maximum scoring  $j \geq i$  used for prediction. The model also take no-answer into consideration by comparing null span score  $s_{null} = S.C + E.C$  to the best non-null span score  $\hat{s}_{i,j} = \max_{j \geq i} S.T_i + E.T_j$  and predict a non-null answer when  $\hat{s}_{i,j} > s_{null} + \tau$  by selecting the threshold  $\tau$  which maximizes the F1 score on test set. It finally uses the sum of log-likelihoods of the correct start and end positions for the training objective. We exploit the scoring mechanism to identify the key-answer term from our input text.

## 7.3 Experimental Results

We looked at experiment results under two broader group. First is around the Social Media Integration where we analyze the potential of FloodBot as a social media agent. Next we present results from the converting the FloodBot into a AI conversing chatbot.

### 7.3.1 Results : Social Media Integration

We aim to scale FloodBot and integrate it as an information propagator into social media. To that end, we wanted to build a product that users find as a necessary component of their life or task. Thus, gauging user engagement is critical for determining whether people value FloodBot. We wanted to see FloodBot’s social media presence and usage pattern. User engagement, also known as product engagement, is a catch-all term that refers to any interaction users have with FloodBot. The primary goal is to measure the leading indicators of whether users find FloodBot valuable enough to continue using it. One of the most often utilized metric for determining user or customer engagement is temporal engagement indicator (i.e. daily, monthly, or quarterly active users).

FloodBot and its ubiquitous computing have enabled the collection of a large amount of data from physical and social sensors (humans), such as data shared by people about their interactions with media posts (Tweets). However, the study

that looks at media usage (directly on data generated) by a disaster monitoring system appears limited. To that end, we investigate the usage pattern and attempt to comprehend and characterize the temporal trends of social media usage through usage analysis.

We go even further and look at the relevancy of social media posting (tweets) to understand user engagement. We explore Twitter data, such as impressions, engagements, and engagement rate for each tweet, were gathered for tweets made during this period from the FloodBot Twitter account. The term “impressions” refers to the number of times a tweet is displayed in a user’s timeline or as part of a search result.

Our tweeter account is setup as an information dissemination platform therefore we wanted to test if the use pattern shows relevancy to our objective. As of February 2022, the UMBC FloodBot Twitter account (@umbc floodbot) has over 300 tweets and 112 followers. The number of times a user interacts with a tweet is referred to as an engagement. Engagement occurs when a user clicks on a tweet in any location, including retweets, replies, follows, likes, links, cards, hashtags, embedded media, username, profile photo, or Tweet expansion. Each tweet’s engagement score was computed as the number of engagements divided by the number of impressions.

### **i) System Stability & Reliability**

We present result in this section from two perspective. First the longitudinal success of our system and seconds its reliability. As discussed in earlier Chapters

we have iterated through various devices and seem to have found system that is reliable. Figure 7.6 shows that our system is live and operating constantly since January 2019 till date.

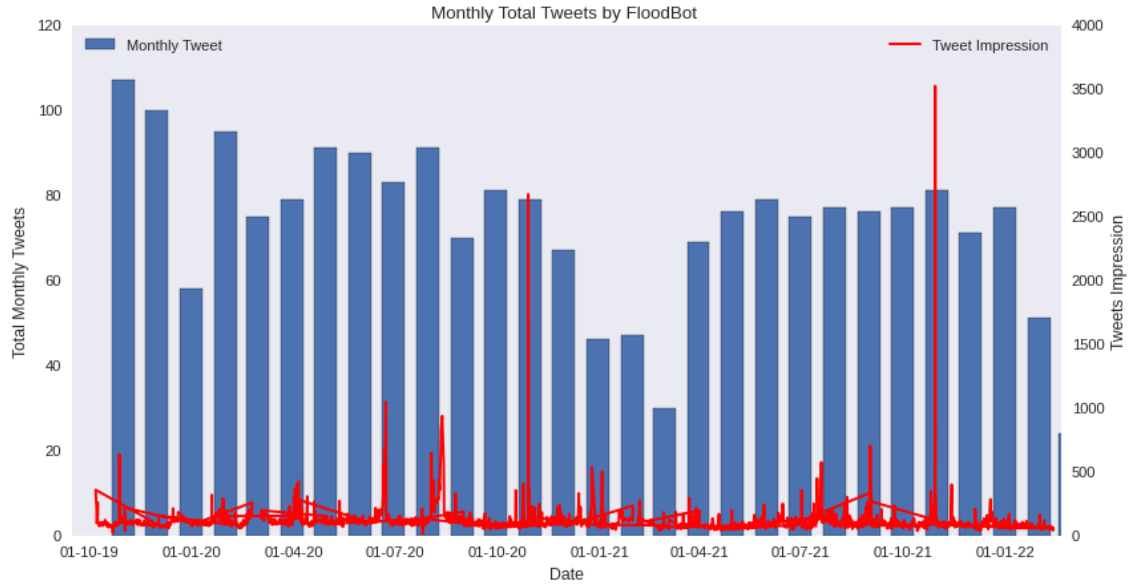


Figure 7.6: Longitudinal: System Stability

FloodBot has been tweeting at a constant rate of around 50 tweets per month. There are numerous integration for automated Tweeting (image captures, weather information gathering, tweet auto-generation, scheduler, etc.). The constant rate (bar plots) of data publication and social media data curation gives us high confidence in the system’s physical and technical stability. Aside from a few manual Tweets from our mobile app, the majority of the Tweets were generated by the system. It’s encouraging to see that the system worked even when the weather was harsh (storm, windy, snow, etc.)

## ii) Use Pattern & Relevancy

We are interested to see how and when does FloodBot get used more to un-

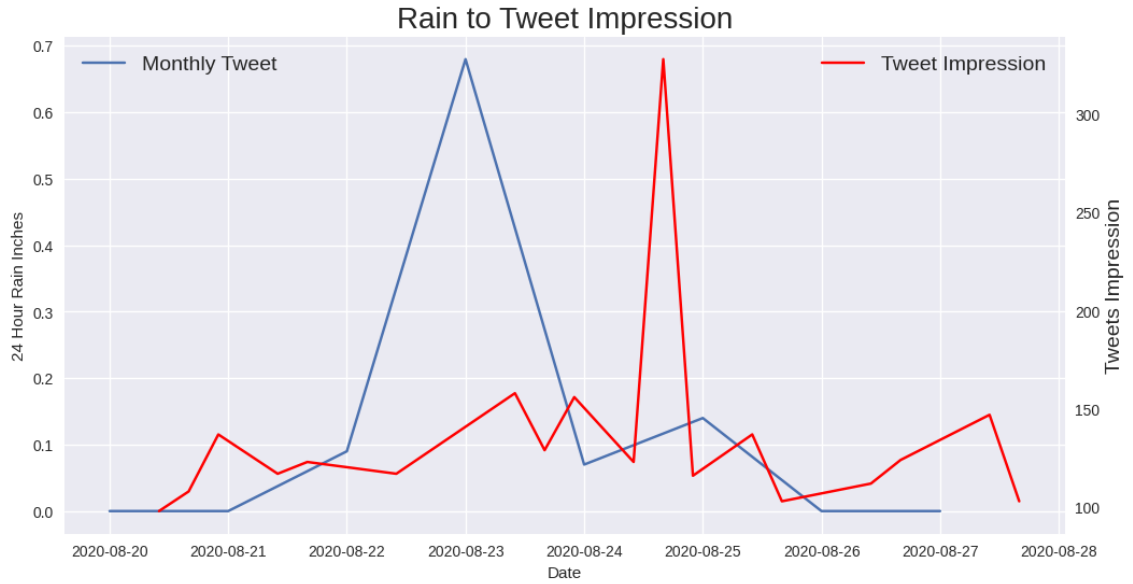


Figure 7.7: Social Media & Relevancy

derstand the user pattern and relevancy of deployments. We use Tweet impressions as a metric to gauge Use Pattern & Relevancy. Tweet Impressions are the total number of times a Tweet has been seen. This includes metrics such as when the message appeared on the timelines of followers, when it appeared in search, and when someone liked the Tweet. It excludes instances when someone saw the Tweet via an embed on a website or a text preview. It only counts if the message is viewed on the Twitter app.

In Figure 7.6 the red lines (spikes) are Tweet impressions during our entire deployment (January 2019 till date). The sudden spikes are the days when Twitter impression have substantially increased. Incidentally, these are also the days when it was heavily raining and the flood possibility in the areas have escalated.

We show more detailed view of same analysis in Figure 7.7. When the rainfall depth has peaked around August 22 2020 (rainfall depth of 0.7 inches in 24 hour),

the impression peaks up to 350 in next 24 hour. Note that the heavy rainfall day prior increases the weariness in the local (FloodBot’s followers) and are logging into to assess the situation. This is another encouraging fact that FloodBot is slowly integrating itself into the society and people come to our application during relevant days (high flood potential).

### 7.3.2 Results : AI Converse

#### i) End to End Memory Network

We motivate the results of chatbot implementation by recalling the basic type of chatbot system, a **Closed Domain Retrieval Chatbot**. These chatbots can only converse around information that is stored in the database. The learning and outcomes from our vision models and output is depicted in Figure 7.8. We have broken down the domain knowledge required by the FloodBot into their own sub domains. The FloodBot can respond to the sub domains  $\{Time, Weather, Flood Risk, and Objects at Risk\}$ . For example the answer to “Is Ellicott City Flooding” can be easily answered by ‘yes/no question’ based on the Flood Risk within that time domain. The overall requirement of the chatbot implementation does not change, so we still need to break the user request into two main parts: utterance and intent. In the examples discussed above, utterance is the sentence/phrase and the user intent is to find the current flooding risk. With this background we now digress into actual deep learning based Chatbot Implementation Results.

Our chatbot implementation is based on end to end memory network [155]. We infused additional data into the bAbI corpus and retrained our model. The end to end memory is built using encoder decoder functions from Keras Sequential model API. The model was trained for 120 epochs, with each epoch limited to event narrative max length of 75 words.


	<b>Object + Flood Conditions</b> <i>{ Time: year:2020, month:2 day:2,day of week: Sunday 2 hour:15}</i> <i>{ Weather : Possible Light Snow ,Windy}</i> <i>{ Flood Risk: Minor Flood}</i> <i>{ Object/Risk : Building:2, Car: 90, Person:1, Street light: 6,Wheel: 1}</i>	<b>Queries</b> <ul style="list-style-type: none"> <li>● <i>Is Ellicott City Flooding</i></li> <li>● <i>What's the weather now?</i></li> <li>● <i>What/Who are at risk?</i></li> <li>● <i>Is their a flooding risk?</i></li> <li>● <i>What is happening and Ellicott City now?</i></li> </ul>
---	--	--

Figure 7.8: Floodbot: Q & A System

The maximum query length was set to 6 words. We trained our model with 10,000 event narrative which consists of 9000 original stories from bAbI and 1000 new ones from our flood related narrative. We tested our model on the same composition i.e 900 from original corpus and 100 is flood related. We maintained the original batch-size of 32. We were able to achieve 89% accuracy after infusing our data.

Table 7.1: Memory Network Comparison

<b>Model :</b>	BaseLine(bAbI)	After Data Fusion
<b>Accuracy:</b>	98.55%	89%

The original (baseline) model without the data fusion achieves 98.55% accuracy. While there is a reduction in the original accuracy the achieved accuracy is still within acceptable range. We believe that the reduction in accuracy is due to increase in vocabulary size and disjoint data sets. The accuracy could increase if we

use the dataset from same domain with similar context.

Next, we present two question and answer result from our FloodBot in Q&A Demo section. The first question is an example from data fusion from NOAA’s website and the second example is from the the joint learning of vision models infused into Q & A knowledgebase corpus. FloodBot answers based on the decoder portion of End to End Memory network and word embedding. In the second, test scenario, the answer provided by the FloodBot is a result of model fusion from three deep learning model stack.

### Chatbot Q&A Demo

**Story:** {‘Six’, ‘inches’, ‘water’, ‘.’ ‘Flooding’, ‘reported’, ‘in’, ‘town’}

**Question:**{ ‘How’, ‘deep’, ‘is’, ‘water’, ‘?’}

**FloodBot’s Answer:** {Six}

---

**Story:** {‘Minor’, ‘Flood’ ‘.’ , ‘Ninety’, ‘cars’ ‘Two’ ‘Building’}

**Question:**{ ‘Is’, ‘there flood risk’, ‘in’, ‘Ellicott City’, ‘?’}

**FloodBot’s Answer:** {yes-minor}

### ii) BERT for QA

FloodBot is a *Closed Domain Retrieval Chatbot*. These kinds of chatbots can only converse around information that is stored in the database. Goal-oriented bots and dialogue systems are designed to categorize the incoming questions into one of the predefined intents. In this work, we have devised the FloodBot’s intent into six

categories:  $\{Temporal, Risk\ Metrics, Weather\ Condition, Flood\ Risk, Unknown, Greetings\}$ . Our model first categorizes questions into one of the given intent before searching for the answer. To process the incoming question and reply properly, FloodBot needs to possess three natural language capabilities (enablers) i.e., NLU (Natural Language Understanding), NLP (Natural Language Processing), and NLG (Natural Language Generation).

Luckily, the computational linguistic area has made great advancement, so such computations are built into a single package. We use **BERT for Q&A** a multi-layer bidirectional Transformer encoder-decoder model [147] to achieve this task. The following section presents our qualitative result from the FloodBot dialogue system. We evaluated FloodBot’s performance on each of the six intents described earlier to accomplish four major tasks (goals). We present examples of question/answer conversations and analysis around these four tasks.

#### **a) Greetings**

To provide a pleasant starting point, we infused the corpus with some greetings and common phrases. People often start by asking the general question before testing the bot’s other functions. We show some sample greetings in the example dialogue below.

Human: *Hello!*

FloodBot: *Hi*

Human: *How are you.*

FloodBot: *I am fine, How can I help you?*

This kind of question answering power is possible because of our general Q&A greetings. This dialogue system lets users inquire about the Ellicott city’s historical or current weather, Flood, damage, mobility, detours, and emergency notice. BERT’s pre-trained model and transfer learning allow us to ask and get very contextual information back. We show some examples of dialogue between FloodBot and the BERT output model below. Some examples are explained by the detailed visualization of the specific question/context pair that includes identifying the answer span.

In example 1, we provided contextual information about Ellicott City and asked our FloodBot a question. We received the answer as “*Howard County, Maryland*” as depicted in Figure 7.9. The blue and orange lines represent the probability of start and end score, respectively, in Figure 7.9. The highest positive scores are selected for the predicted answer text span. The model extracted the highest start score at word token “Howard” and highest end score at word token “Maryland” and provide the correct answer as “Howard County, Maryland.” The model is tuned to pick up the highest end score, given that the end score should be greater than the start score to avoid getting a negative or wrong answer.

**Example 1:** *Human:* Where is Ellicott City?

*FloodBot:* Howard County, Maryland

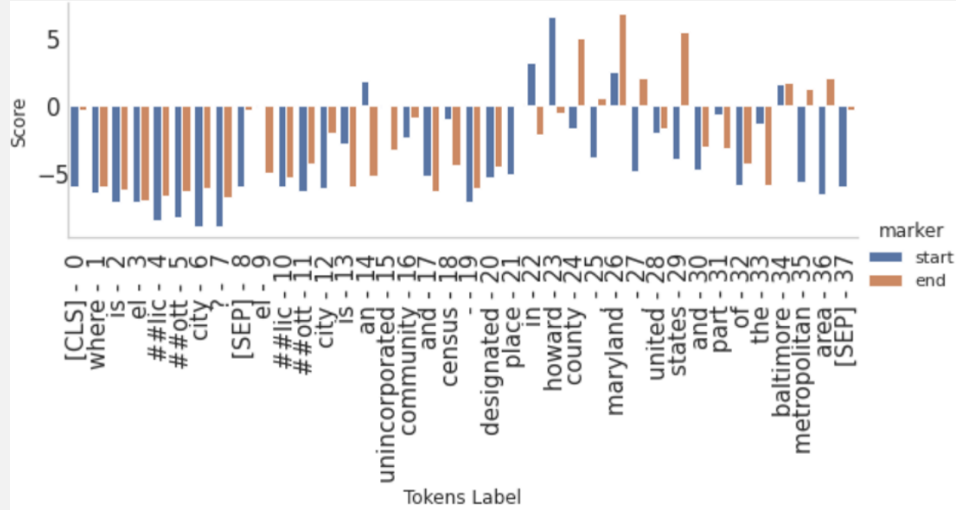


Figure 7.9: Example 1 Q&A score span visualization

In example 2, we have two different dates and numbers of cars presented in a specific parking lot. This information is coming from our object detection vision model. We posed this question along with the context, as shown in example 2, and the model selected the answer for the specific date asked in the question. We obtained the reply as “14 cars”, and the calculation for start/end score is visualized in Figure 7.10. The model selects the start score at token “14” and end score at token “cars,” which is a definite and accurate answer from the given context. The model is capable of answering the specific question from the very little information that includes temporal components.

**Example 2:** Human: How many cars are there on 2020-1-25?

FloodBot: 14 Cars

Human: How deep is the stream on 2020-5-20?

FloodBot: 1.6 ft

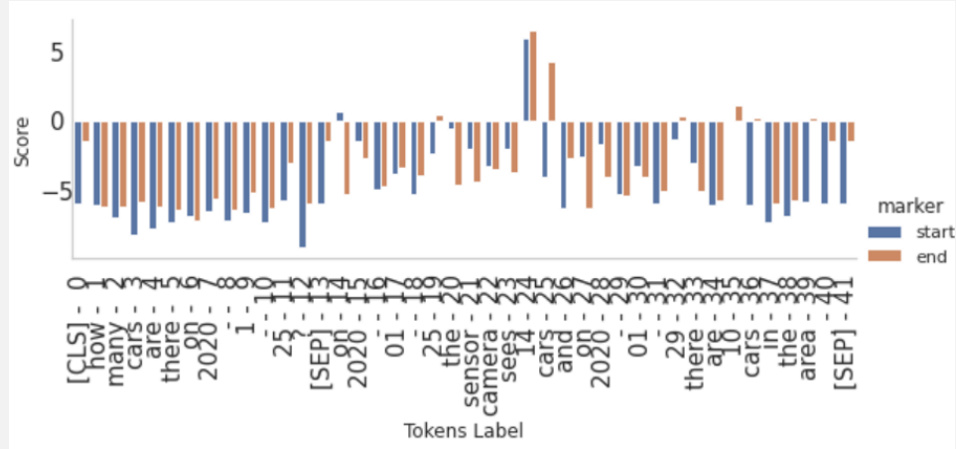


Figure 7.10: Example 2 Q&A score span visualization

## b) Temporal Question Parsing

We envision FloodBot to receive two kinds of temporal questions. The more common one is when people want to know what is happening “right this moment” in the city. The second one is a historical question when people want to know what has already happened in the area. In our context repository, we normalize time expressions and use Stanford Temporal Tagger (SUTime) [37] to tag them. SUTime allows us to find and map human-asked time-phrase to the database stored time-event parameters. As seen in Table 7.2, various time-related attributes are tagged for NLP and NLU understandings.

Table 7.2: Temporal Parser

Text	Value	Timex3 Tag
5/25/2020	5/25/2020	t1 type=DATE value=2020-05-252020-05-25
Current	PRESENT_REF	t2 type=DATE value=PRESENT_REFCurrent
now	PRESENT_REF	t3 type=DATE value=PRESENT_REFnow

### c) Risk Metrics

The vision model allows us to assimilate contextual risk metrics. Essentially, there are two parameters in the risk metrics. The first one is the stream flood condition itself, such as {Flood, Minor Flood or No Flood}. The second one is object identification and their counts around flood vicinity. This data is entered as a Q&A pair to the training model. BERT’s fine-tuning model then allows FoodBot to provide localized and holistic contextual dataset consisting of Ellicott City information. BERT models are asked to answer specific questions, as shown in Figure 7.11.

### d) Unknown Intents

Just like greetings, unknown intent is also an important property of chatbot. To support response to the vague questions and ones outside predefined intent, we infused repeated Q&A pair with a fixed response “Sorry, I am still learning.”.

*Human: How high is Mount Everest?*

*FloodBot: Sorry I am still learning*

*Human: What is Deep Learning?*

*FloodBot: Sorry I am still learning.*

Next, we also tested the BERT based FloodBot on various information created using vision data along with other media contents. This is our FloodBot Dialogue system for comprehension Q&A. The response is expected to identify important concepts and respond to one or more questions from that passage. Given an input as contextual *training set* :

*2020-05-25, there were 16 Car in the parking lot. Current weather condition is Possible Light Rain. The stream is in **Flood** now. On 2020-01-28 there were **22 Car** in the area with Mild Rain showers weather, sunny weather and no flood in water stream.*

Similarly, we posed several questions expecting a specific answer present in those sample paragraphs. The answers from FloodBot are shown in Figure 7.11 with a simulated Chat user interface. We asked the specific date and weather-related questions, and the FloodBot answered them accurately.

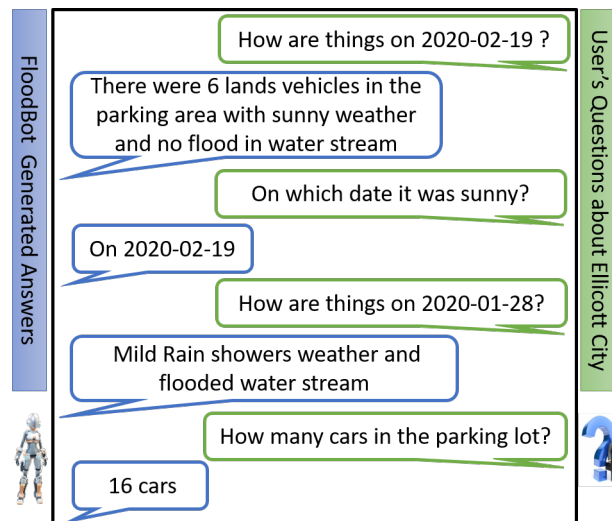


Figure 7.11: Question Answering System Using BERT

## 7.4 Conclusion & Discussion

Social media, particularly Twitter, is being used more and more to improve resilience during extreme weather events/emergency management situations, such as floods, by communicating potential risks and their impacts and informing agencies and responders. We discussed a prototype Twitter data posting and mining pipeline (Social Media Integration) in this chapter to improve stakeholder situational awareness during flooding events. The preliminary data analysis and usage patterns and follower-ship between users and our Tweets show promising results for Twitter as an emergency response tool during a crisis. Our data show that people share, like, and repost more during potential flood situations, which helps raise awareness and illustrate ways people can help each other if needed. The government and regulators can also monitor the spikes to better disaster planning.

Our research presented in this chapter is exploratory in nature, presenting the first steps toward developing a web-based crisis response information system that leverages social media and user-generated content and data. The study's main limitation is the small number of tweets and localized solutions. The followers are Ellicott City residents or people who are interested in the area. Because our data propagation is limited to a smaller town, we have fewer followers than generic national propagation systems. Nonetheless, we believe that deploying such a system could be a valuable tool for local governments and stakeholders looking to conduct emergency response activities or raise social awareness.

Chatbots have become very popular in recent years, primarily due to advancements in machine learning and other underlying technologies, such as natural language processing. Day by day, their uses are increasing at a pace that may soon replace customer service personnel. We are aware that the object detection from our vision model is probabilistic and somewhat heuristic. We argue that the ultimate use of this kind of application is either to have a sense of emergency or to estimate the possible damage. Thus, object detection and counts measures are mostly for estimation purposes. During disasters and life-saving events, the information provided by our model could be more valuable than trying to get to the correct count.

In this work, we have demonstrated how complex deep learning models can be deployed for practical AI. We believe that we have been able to design and deliver an end to end pipeline of three kinds of sophisticated models and put them behind a user-friendly chatbot-“FloodBot”. The technology is easily transferable and can be a life-saving tool in all three phases (pre, during, and post) of disaster. Recently, the deep learning area has seen substantial user adaptation, and its use in the commercial sector is increasing. Similarly, community support applications such as this can further increase the utility of deep learning.

We only evaluated FloodBot’s conversing power qualitatively and with a limited question to our trained model in the pipeline. In future work, we plan to integrate BLEU (bilingual evaluation understudy) algorithm for evaluating the quality of answers. BLEU is currently the gold standard to assess machine-translation (FloodBot’s response) from one natural language to a human question. Reinforce-

ment Learning is another technique often used in the dialogue system, and we seek to experiment with various kinds of reinforcement learning in our future work.

## Chapter 8: Conclusion & Future Work

Flood disasters and risk reduction can be accomplished in two ways: detection and prediction. In machine learning problems detection and prediction appear to be synonymous with predictive analytics or simply prediction. However, they are two completely different tasks. Detection is the process of extracting insights or information from a data. This can include detecting objects, finding fraud activities, or isolating anomalous instance from dataset. In contrary, prediction is the process of forecasting or estimating future events using historical and current data, most commonly through trend or data pattern analysis. Predictions should be precise and defined by logic. Predictive analytics is the use of data, statistical algorithms, and machine learning techniques to identify the likelihood of future outcomes based on historical data. The goal is to provide the best forecast of what will happen in the future, rather than simply knowing what has happened.

This thesis posed problem of flood detection and not flood prediction. We tried to detect flood event as it happened or by analyzing the past events during flood. The flood forecasting problem is much more challenging and complex because it involves many meteorological phenomenon. Flash floods are a result of favorable

meteorologic and hydrologic condition [75]. Though heavy rainfall is often the cause, heavy rain alone does not cause severe flood events. The hydrologic character of the watershed plays an important role. Some of the hydrologic parameters that play out in flash flooding include: how rain-cycle, how quickly the storm is moving, how intense are the rainfall rates, much rainfall becomes surface runoff, and where it drains to, the soil's conductivity, site permeability, etc. [106]. Therefore, predicting flash flood using a machine learning approach alone might result in a sub-optimal model.

The idea of using multiple sources of information to investigate a problem is not new: we use sight and sound to cross a street, sight, smell, and taste to evaluate a restaurant meal, and use multiple reviews to compare product before deciding to buy one. It is human nature to compare and contrast different source/systems etc and has helped us make more informed decision. Similarly, we want to infuse the redundancy in decision making process and aid the design with multiple data sources. We primarily focused our thesis around flood detection and creating an early warning system to better prepare the community during or after the crisis using multiple source. We called our automated flood detection system prototype as **FloodBot**. We summarize our multimodal data integration approach used in this thesis using Figure 8.1.

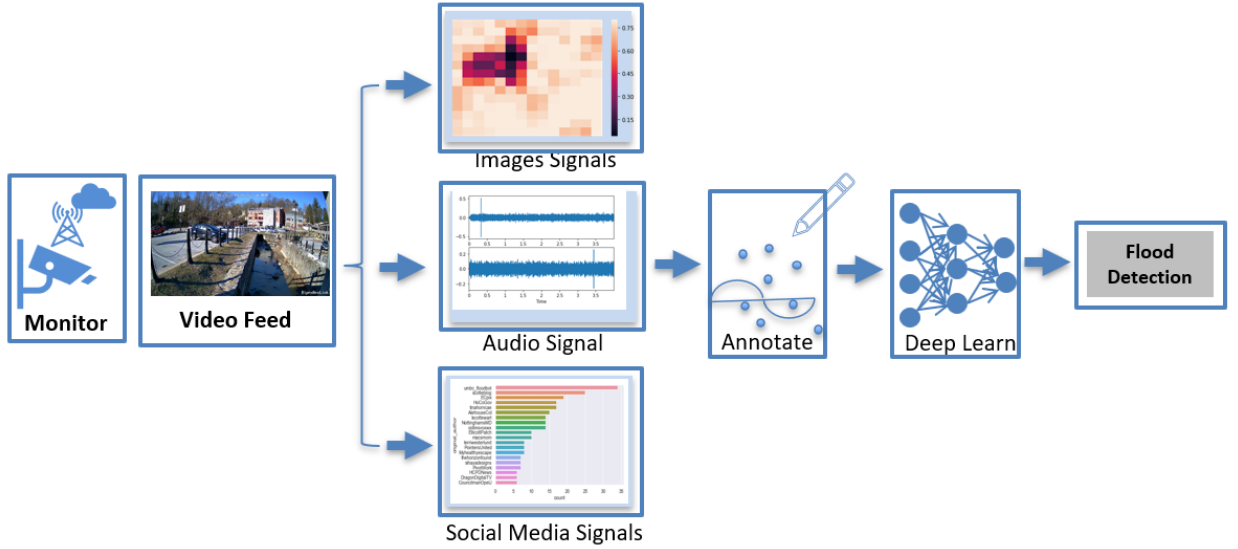


Figure 8.1: Summary: Flood Detection Approach

Our system begins with deploying data capturing (monitoring) sensors on-site. We then transformed them into multimodal data representations (signals). Earlier chapters were focused in designing the stable/scaleable system. The raw data is annotated and prepared for various deep learning models. We detect flood conditions using these deep learning models and discrete data representations.

### 8.0.1 Synopsis

This section briefly summarizes the thesis using funnel diagram in Figure 8.2. We started our research by investigating audiovisual sensors, embedded systems, and liquid level sensors (hardware elements). Noting that hardware research is somewhat different than the machine learning problem we were interested on, we focused our research in the system development using machine learning and opted for commercial hardware. During the course of research it was quickly evident that

the performance of traditional machine learning algorithms would be insufficient to address the complexity of our problem (flood detection using multi-modal data).

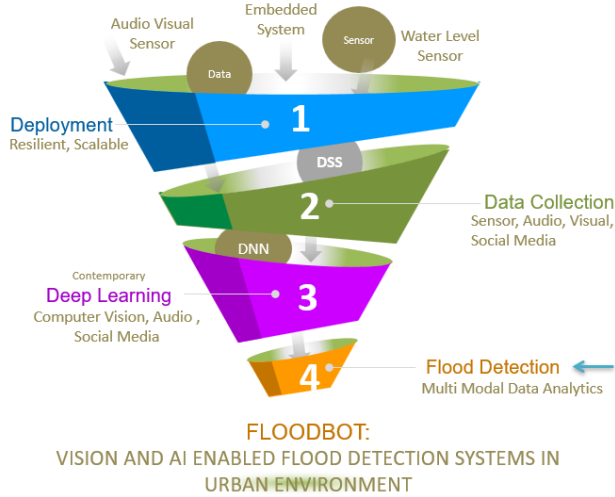


Figure 8.2: Synopsis

We then experimented with a variety of deep learning models, including Computer Vision (CV), Natural Language Processing (NLP), and Acoustic Event Detection (AED). We used multi-modal data such as vision, sound, and social media to transform

one-dimensional water level reading data into a multifaceted representation of the environment. We trained many deep neural networks and used transfer learning to parse images and its contents. We trained the language and vision of the Floodbot to understand the context around the Flooding site. Initially, the scope of this research focused on performing supervised learning. Later, we reposed the question as a similarity search and turned it into semi-supervised learning. All of these expansions allow FloodBot to have better AI-Conversing power. We showed how multimodal data infusion could be achieved using vision model stacking.



(a) Classification Output

(b) Object Detection Output

Figure 8.3: Vision Model Results

## 8.0.2 Image - for Flood Detection

In this section, we created domain-specific Flood Classification and Object detection Models. To classify the current flooding situation at the site, we collected video and posed our flood classification problem as a multi-class classification problem. We categorized the flood images into three classes *No Flood*, *Minor Flood*, *Flood* based on the stream-flow conditions observed in the images.

The classes in our training set are based on turbulence, turbidity and observed high velocity of the flowing water predominantly during no-rain, light rain, and heavy rain conditions. We devised a smart labeling technique that is robust and practical, given the large data set. Figure 8.6(a) shows the result from one of our validation sets. In the given examples, the model correctly classifies images

into  $\{No\ Flood, Minor\ Flood, and\ Flood\}$ , respectively. To evaluate Flood Classification Model’s performance, we experimented with two classification models: a Support Vector Machine (SVM), and a deep Convolutional Neural Network (CNN). We trained these models on our images without using the transfer learning technique. The Cross-Domain Transfer learning substantially improved the accuracy since earlier layers within the base model have already performed and learned feature properties from millions of images.

We processed thousands of image frames and ran them through the Single-Shot Object Detection Model. Our motivation to identify objects in the vicinity is to assess the potential damage/flood that occurs. From the dataset collected over months, the Object Detection model identified more than 90 distinct objects such as  $\{Car, Wheel, House, Tree, Building, Vehicle, Land\ vehicle, Tire, Window, Plant, Bench, Truck\ etc.\}$ .

There were many ways that we could have validated output from our object detection model. We used a pragmatic and timely parameter to validate our model—the Covid19 pandemic impact to the area. The Flood Watch camera is located at a popular tourist destination in Howard County, Maryland. We selected “car/vehicle” as one of our measures and aggregated the counts by week.

It is interesting to note that the vehicles count closely follows Maryland State’s “Shelter at Home” and “Stay at Home” orders. We show the weekly distribution of detected cars from the area shown in Figure 8.4 during the five-month deployment.

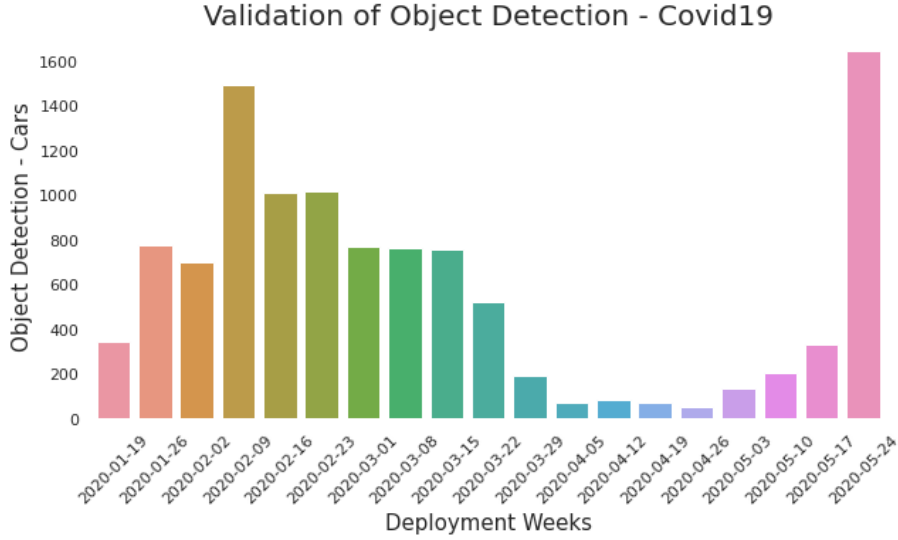


Figure 8.4: Weekly Object(Car) Count During Covid ‘19

The number of detected cars went significantly down around the “Shelter at the home” period, which began in the first week of March 2020, and the numbers are rising as the restriction was lifted. After May 2020, “Stay at Home” order was further loosened, and people are allowed to dine out and enjoy the area. The result directly correlates to the ground truth and validates that such a model can be used for area surveillance and meta-analysis.

### 8.0.3 Audio - for Flood Detection

Our objective in proposing multi-modal flood detection is to bring in the robustness to the system and provide alternate means to detect flood when one of them fails. One of the challenge in all of vision related work is model performance during low light condition (night, fog, heavy rain etc). We believe that audio and natural sound parsing might help FloodBot to get a better performance. Therefore,

we explored audio signal to classify flood severity.

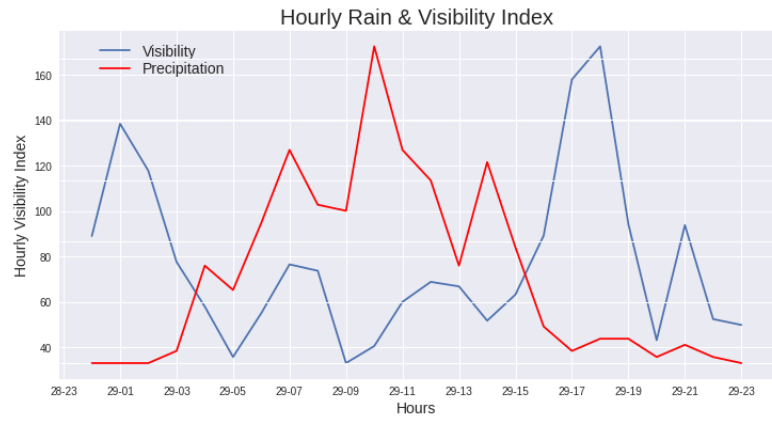
We present an example where one modalities (image) fails and alternate modalities (sound) can still be used to detect flood. We show qualitative analysis of one stormy day (10/29/2020) in Figure 8.5. The first Figure 8.5(a) shows our site capture. Next we show the inverse relationship between rainfall and visibility. As seen in Figure 8.5(b) the visibility is at minimal during the peak rainfall, another inherent problem for vision only solution. As the rainfall increases camera starts to loose its visibility and reaches to minimal around 11 AM when the picture in Figure 8.5(a) was captured.

Due to heavy rain and fog the vision is blurred and failed all three of our vision based models. Since there is no ‘rock’ or island mass to track, we have no way to run our segmentation model. The wind and rain storm seems to have moved the camera from its original angle and views to detect object seems to be hindered too. Lastly, since the camera is completely blurred the classification model was unable to detect the flood situation. These are far more common issues with vision only based solutions.

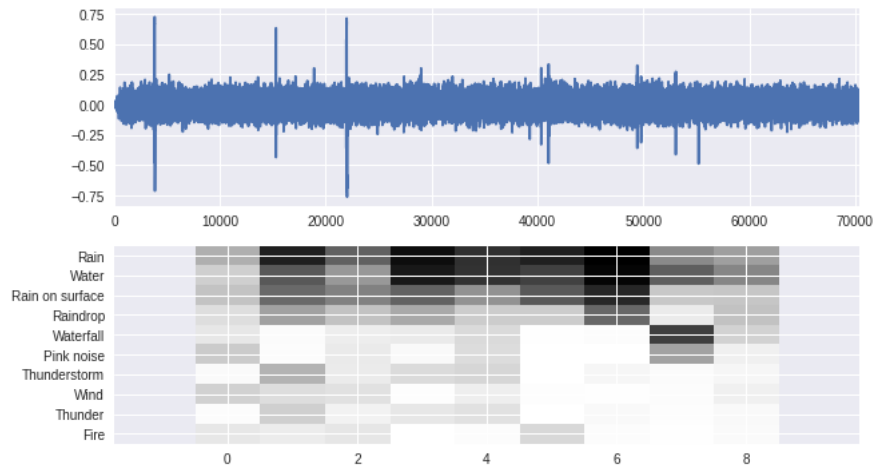
In this situation, vision only based flood detection would have been failed and we would have no option to detect flood using machine learning techniques. To avert such problem, we decided to use alternate data modalities in this case and relied on sound signal. We show the result from sound based Flood Detection on figure 8.5(c). We used is a pretrained deep net called that YAMNet [64] that is



(a) Site Image Capture



(b) Visibility & Rainfall



(c) Audio Based Flood Detection

Figure 8.5: Qualitative Analysis: Multimodal Flood Detection

trained to predicts 521 audio event classes based on the AudioSet-YouTube corpus, and employs the Mobilenet\_v1 depthwise-separable convolution architecture.

As shown on Figure 8.5(c), YAMNet is able to detect {Rain, Water, Rain on Surface, Raindrop, Waterfall ...} etc as the highest detected classes. These classes can be used to infer that the site is experiencing a rain. Continued monitoring of such sound detection (long term rain often yields to flood, no rain sound detection means less chances of flood etc.)

#### 8.0.4 Social Media - for Flood Detection

For our social media integration, we created a framework that uses various flood disaster data from sensor network sources and social media sources to build a sensor-social data-fused flood detection system. We presented a novel data fusion framework and data analysis for accurately classifying localized context rich flood-related tweets along with full system integration and design for social media integration platform. We deployed our sensor system and integrated a connection to a social media platform to obtain direct local contextual feeds from social users.

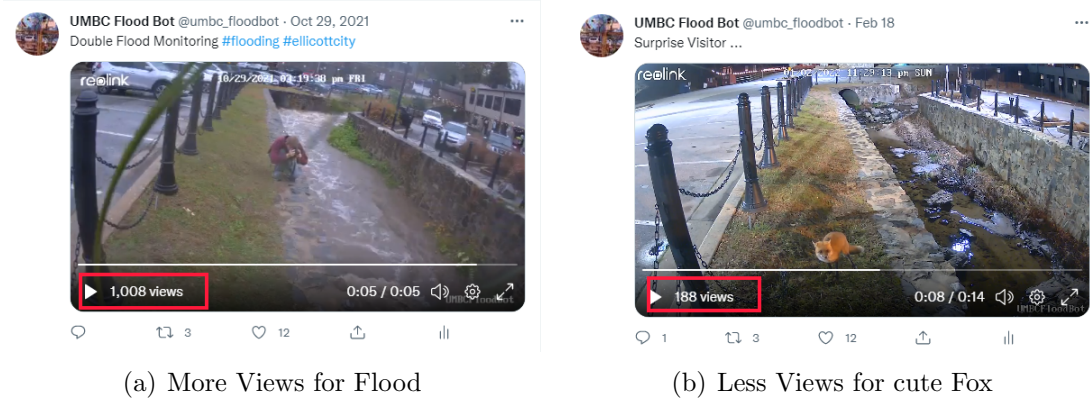


Figure 8.6: Elevated View During Flood

During the course of this research, we conducted interviews and site reconnaissance, we met and talked to various community members including local government and found that the recurring flash flood problems have kept people of Ellicott City on their toes, and every passing storm reminds them of previous flood devastation. Therefore, as part of community integration, we plan to find out the current state of research already happening in Ellicott City and make an earnest effort to bring all stakeholders as active participants into our community integration process. As seen by the view counts there are more than 1000 views when Elliott City is experiencing a high flood versus when the scene is normal from flooding but still interesting for social media (fox in the picture).

## 8.1 Future Work

We envision to equip the communities, with greater preparedness and resilience to natural disaster-*Floods*. Ultimately our thesis and the design will foster think-

ing among communities to evaluate best design for their community (ground or remote sensing, deployment density and locations, key resilience indicators, alert and recovery etc.) as we extend, deploy and further evaluate our preliminary cyber-physical-social sensing prototype, FloodBot [19, 22].

In future, we plan to expand our prototype, FloodBot and enhance its resiliency by exploiting ground, remote & social sensing capabilities such as moisture sensors, on-street camera networks, lidar, radars, drones, satellites, twitter with more descriptive hydrological models and micro-weather predictive analytics. We plan to enhance applied machine learning techniques in time series, computer vision, remote sensing, NLP, and citizen science inspired community feedback collection and synthesis mechanisms to envision, , as a holistic scalable and adaptable flood monitoring and alert networks in urban environment.

We are expanding the scope of our thesis beyond the Ellicott City area. For example, we are collaborating with colleagues at the University of Pisa’s Department of Information Engineering in Italy to investigate the possibility of extending the FloodBot’s current centralized cloud computing system to a hybrid cloud/edge computing system. The decentralization of functionalities in the flood monitoring system implementation is expected to have a significant impact on scalability and cost efficacy.

We are working to improve the system by reducing data transfer and easing the load on the cloud, which is critical for the system’s wide adoption and commercial

feasibility. By establishing direct communication, we are experimenting with edge computing to reduce latency between monitoring stations and the data analysis service. Finally, and perhaps most notably, a decentralized implementation improves system resilience by reducing the need for continuous data offloading and transfer, thereby minimizing network interruption-related issues.

Ultimately, our contribution to this thesis and applied research is a functional prototype that has been deployed and operational for over two years and our research findings [16, 17, 22, 19, 18, 21, 19, 136, 133, 134] and released our data in public domain [28]. We recognize that the complexity is most evident in the early phase, when we are setting up the system and starting to train the model. Following the preliminary work, we believe that a system such as FloodBot should be simple to scale and replicate beyond flood detection.

We sincerely hope that The AI-Ready data released in public domain [28] will create many new research direction and help future researchers to look at our data beyond the context of Flood detection.

\*\*\*\* The End \*\*\*\*

## Bibliography

- [1] Detection and classification of acoustic scenes and events.
- [2] Flood susceptibility mapping in dingnan county (china) using adaptive neuro-fuzzy inference system with biogeography based optimization and imperialistic competitive algorithm. *Journal of Environmental Management*, 247:712–729, 2019.
- [3] Detection and classification of acoustic scenes and events challenge, 2020.
- [4] Index. In Siddharth Misra, Yifu Han, Yuteng Jin, and Pratiksha Tathed, editors, *Multifrequency Electromagnetic Data Interpretation for Subsurface Characterization*, pages 351–358. Elsevier, 2021.
- [5] Jantan-Aman Omolara Abiodun Esther Dada Kemi Victoria Abiodun, Olu-dare Isaac. State-of-the-art in artificial neural network applications: A survey. 2018.
- [6] Eleni Adamopoulou and Lefteris Moussiades. Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2:100006, 2020.
- [7] Chris Hallacy Aditya Ramesh Gabriel Goh Sandhini Agarwal Girish Sastry Amanda Askell Pamela Mishkin Jack Clark Gretchen Krueger Ilya Sutskever Alec Radford, Jong Wook Kim. Learning transferable visual models from natural language supervision. 2021.
- [8] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases-an introduction. *arXiv preprint cmp-lg/9503016*, 1995.
- [9] Margarita Angelidou. Smart cities: A conjuncture of four forces. *Cities*, 47, 05 2015.

- [10] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.
- [11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.
- [12] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D. Plumbley. Acoustic scene classification. *CoRR*, abs/1411.3715, 2014.
- [13] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D. Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, May 2015.
- [14] Jon Barker, Emmanuel Vincent, Ning Ma, Heidi Christensen, and Phil Green. The pascal chime speech separation and recognition challenge. *Computer Speech Language*, 27:621–633, 05 2013.
- [15] Bipendra Basnyat. Feasibility study of design development of cyber-physical device for real-time detection of flash floods in the smart city]. *UMBC Master Thesis*, 2017.
- [16] Bipendra Basnyat, Amrita Anam, Neha Singh, Aryya Gangopadhyay, and Nirmalya Roy. Analyzing social media texts and images to assess the impact of flash floods in cities. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6. IEEE, 2017.
- [17] Bipendra Basnyat, Nirmalya Roy, and Aryya Gangopadhyay. A flash flood categorization system using scene-text recognition. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 147–154. IEEE, 2018.
- [18] Bipendra Basnyat, Nirmalya Roy, and Aryya Gangopadhyay. Towards ai conversing: Floodbot using deep learning model stacks, 2020.
- [19] Bipendra Basnyat, Nirmalya Roy, and Aryya Gangopadhyay. Flood detection using semantic segmentation and multimodal data fusion, 2021.
- [20] Bipendra Basnyat, Neha Singh, Aryya Gangopadhyay, and Nirmalya Roy. Vision powered conversational ai for easy human dialogue systems. In *IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2020.
- [21] Bipendra Basnyat, Neha Singh, Aryya Gangopadhyay, and Nirmalya Roy. Vision powered conversational ai for easy human dialogue systems. In *IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2020.

- [22] Bipendra Basnyat, Neha Singh, Nirmalya Roy, and Aryya Gangopadhyay. Design and deployment of a flash flood monitoring iot: Challenges and opportunities, 2020.
- [23] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What’s the point: Semantic segmentation with point supervision. pages 549–565, 2016.
- [24] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, jan 2009.
- [25] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014.
- [26] Mathias Berglund, Tapani Raiko, Mikko Honkala, Leo Kärkkäinen, Akos Vetek, and Juha Karhunen. Bidirectional recurrent neural networks as generative models - reconstructing gaps in time series. *CoRR*, abs/1504.01575, 2015.
- [27] Bidhan Bhattacharya, R. Price, and Dimitri Solomatine. A machine learning approach to modeling sediment transport. *Journal of Hydraulic Engineering-asce - J HYDRAUL ENG-ASCE*, 133, 04 2007.
- [28] Ahmadur Rahman Bipendra Basnyat, Arooj Zia. Floodbot:vision and ai enabled flood detection systems in urban environment - data repository.
- [29] Chris Bishop. Pattern recognition and machine learning. In *Pattern Recognition and Machine Learning*, volume 1 of *1613-9011*, pages 123–140. Springer-Verlag New York, 2006.
- [30] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.
- [31] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog, 2016.
- [32] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [33] DeLiang Wang; Guy J. Brown. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. 2006.
- [34] Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *CoRR*, abs/1810.00278, 2018.

- [35] Jack Cahn. Chatbot: Architecture, design, & development. *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*, 2017.
- [36] Zhaowei Cai, Quanfu Fan, Rogerio S. Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 354–370, Cham, 2016. Springer International Publishing.
- [37] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740, 2012.
- [38] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *International workshop on artificial intelligence and statistics*, pages 57–64. PMLR, 2005.
- [39] Eric Chaumillon, Xavier Bertin, André B. Fortunato, Marco Bajo, Jean-Luc Schneider, Laurent Dezileau, John Patrick Walsh, Agnès Michelot, Etienne Chauveau, Axel Créach, Alain Hénaff, Thierry Sauzeau, Benoit Waeles, Bruno Gervais, Gwenaële Jan, Juliette Baumann, Jean-François Breilh, and Rodrigo Pedreros. Storm-induced marine flooding: Lessons from a multidisciplinary approach. *Earth-Science Reviews*, 165:151–184, 2017.
- [40] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017.
- [41] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.
- [42] Guang Li Yanan Li Junpei Zhong Chenguang Yang, Jing Na. "neural network for complex systems: Theory and applications complexity". (2), 2018.
- [43] David Wollman Edward Griffor Christopher Greer, Martin Burns. Cyber-physical systems and internet of things. *National Institute of Standards and Technology*, (3), 2019.
- [44] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 379–387. Curran Associates, Inc., 2016.
- [45] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

- [46] Judith E. Dayhoff. *Neural network architectures : an introduction*. Van Nostrand Reinhold New York, N.Y, 1990.
- [47] Boris Defréville, Pierre Roy, Christophe Rosin, and Francois Pachet. Automatic recognition of urban sound sources. volume 2, 05 2006.
- [48] DHS. News Release: DHS S&T Partners with Local Communities to Improve Flood Resiliency. <https://www.dhs.gov/science-and-technology/news/2018/05/21/news-release-dhs-st-partners-local-communities-improve-flood/>, 2020.
- [49] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.
- [50] Evigia Systems LLC. Evigia systems llc). [://evigia.com/](http://evigia.com/).
- [51] Mahdi Fahmideh and Didar Zowghi. Iot smart city architectures: an analytical evaluation. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 709–715. IEEE, 2018.
- [52] Graham D. Finlayson. Colour and illumination in computer vision, 2018.
- [53] National Science Foundation. Floodbot: Sensors social media alert residents to dangerous flash floods. Youtube, 2020.
- [54] Warren Gay. *Raspberry Pi Hardware Reference*. Apress, USA, 1st edition, 2014.
- [55] M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8367–8371, 2013.
- [56] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [58] Saptarsi Goswami, Sanjay Chakraborty, Sanhita Ghosh, Amlan Chakrabarti, and Basabi Chakraborty. A review on application of data mining techniques to combat natural disasters. *Ain Shams Engineering Journal*, 9(3):365–378, 2018.
- [59] Ruslan Salakhutdinov Gregory Koch, Richard Zemel. Siamese neural networks for one-shot image recognition. *International Conference on Machine Learning*, Jun 2015.

- [60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [62] Roopa B. Hegde, Keerthana Prasad, Harishchandra Hebbar, and Brij Mohan Kumar Singh. Comparison of traditional image processing and deep learning approaches for classification of white blood cells in peripheral blood smear images. *Biocybernetics and Biomedical Engineering*, 39(2):382–392, 2019.
- [63] Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. pages 292–299, 01 2014.
- [64] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.
- [65] Nazari Foad Smith Virginia Nataraj C Hosseiny, Hossein. A framework for modeling flood depth using a hybrid of hydraulics and machine learning. pages 2045–2322, May 2020.
- [66] Yuanbo Hou, Qiuqiang Kong, Shengchen Li, and Mark D. Plumbley. Sound event detection with sequentially labelled data based on connectionist temporal classification and unsupervised clustering. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.
- [67] Howard County Department of Public Works. Howard County Department of Public Works. <https://www.howardcountymd.gov/public-works>, 2022.
- [68] HowardCounty. Flood sensor monitoring dashboard). <https://https://howardcounty.onerain.com/home.php/>, 2020.
- [69] Guoning Hu and DeLiang Wang. Auditory segmentation based on onset and offset analysis. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15:396 – 405, 03 2007.
- [70] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)*, 47(4):67, 2015.
- [71] Kazi Aminul Islam, Mohammad Shahab Uddin, Chiman Kwan, and Jiang Li. Flood detection using multi-modal and multi-temporal images: A comparative study. *Remote Sensing*, 12(15), 2020.

- [72] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning, 2021.
- [73] Chaitanya K. Joshi, Fei Mi, and Boi Faltings. Personalization in goal-oriented dialog. *CoRR*, abs/1706.07503, 2017.
- [74] Khabat Khosravi, Himan Shahabi, Binh Thai Pham, Jan Adamowski, Ataollah Shirzadi, Biswajeet Pradhan, Jie Dou, Hai-Bang Ly, Gyula Gróf, Huu Loc Ho, Haoyuan Hong, Kamran Chapi, and Indra Prakash. A comparative assessment of flood susceptibility modeling using multi-criteria decision-making analysis and machine learning methods. *Journal of Hydrology*, 573:311–323, 2019.
- [75] Pirasteh Saied Pradhan Biswajeet Mahmud Ahmad Rodzi Sulaiman Wan Nor Azmin Moradi Abbas Kia, Masoud Bakhtyari. An artificial neural network model for flood simulation using gis: Johor river basin, malaysia. *Environmental Earth Sciences*, 67(9):251–264, 2012. 8th IFAC/IFOORS Symposium on Identification and System Parameter Estimation 1988, Beijing, PRC, 27-31 August.
- [76] Jooho Kim and Makarand Hastak. Social network analysis: Characteristics of online social networks after a disaster. *International Journal of Information Management*, 38(1):86–96, 2018.
- [77] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.
- [78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [79] Dana Lahat, Tülay Adalı, and Christian Jutten. Multimodal data fusion: An overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015.
- [80] Shervan Fekri-Ershad Laleh Armi. Texture image analysis and texture classification methods - a review. *International Online Journal of Image Processing and Pattern Recognition*, 2000.
- [81] Xuan-Hien Le, Hung Viet Ho, Giha Lee, and Sungho Jung. Application of long short-term memory (lstm) neural network for flood forecasting. *Water*, 11(7), 2019.

- [82] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
- [83] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [84] Roy R Gu Li Yitian. Modeling flow and sediment transport in a river system using an artificial neural network. environmental management. *Environmental management*, 31:122–134, 2003.
- [85] Shiyu Liang and R. Srikant. Why deep neural networks for function approximation?, 2017.
- [86] Libelium. Waspnote development wiki). [://www.libelium.com/iot-products/waspnote/](http://www.libelium.com/iot-products/waspnote/).
- [87] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.
- [88] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.
- [89] Yu Liu, Xi Liu, Song Gao, Li Gong, Chaogui Kang, Ye Zhi, Guanghua Chi, and Li Shi. Social sensing: A new approach to understanding our socioeconomic environments. *Annals of the Association of American Geographers*, 105(3):512–530, 2015.
- [90] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.
- [91] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *CoRR*, abs/1506.08909, 2015.
- [92] Dang Mai and Florimond Smedt. A combined hydrological and hydraulic model for flood prediction in vietnam applied to the huong river basin as a test case study. *Water*, 9:879, 11 2017.
- [93] Stephen McAdams and Emmanuel Bigand. *Thinking in Sound: The Cognitive Psychology of Human Audition*. 1993.
- [94] Annamaria Mesaros, Toni Heittola, Emmanouil Benetos, Peter Foster, Mathieu Lagrange, Tuomas Virtanen, and Mark D. Plumbley. Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2):379–393, 2018.

- [95] Annamaria Mesaros, Toni Heittola, Antti Eronen, and Tuomas Virtanen. Acoustic event detection in real life recordings. pages 1267–1271, 2010.
- [96] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38(5):67–83, Sep 2021.
- [97] Tom Mitchell. Machine learning. 1997.
- [98] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 2016.
- [99] Amir Mosavi, Pinar Ozturk, and Kwok-wing Chau. Flood prediction using machine learning models: Literature review. *Water*, 10(11), 2018.
- [100] Maria E Niessen, Leendert Van Maanen, and Tjeerd C Andringa. Disambiguating sound through context. *International Journal of Semantic Computing*, 2(03):327–341, 2008.
- [101] Jim Nilsson and Tomas Akenine-Möller. Understanding ssim, 2020.
- [102] Jeerana Noymanee and Thanaruk Theeramunkong. Flood forecasting with machine learning technique on hydrological modeling. *Procedia Computer Science*, 156:377–386, 2019. 8th International Young Scientists Conference on Computational Science, YSC2019, 24-28 June 2019, Heraklion, Greece.
- [103] NSF. Floodbot: Using sensors and social media to alert residents of dangerous flash floods. "NSF Science Now", 2020.
- [104] NSF. Nsf award search: award#1640625-distributed data analytics for real-time monitoring and detection of flash floods in smart city 2020). <https://www.nsf.gov/awardsearch/showAward?AWDID=1640625,journal=Nsf.gov/>, 2020.
- [105] NSSL. Color indexing. *International Journal of Computer Vision*, Jun 191.
- [106] NSSL. Severe weather 101. *The National Severe Storms Laboratory*, Jun 2020.
- [107] US Army Corps of Engineer. Hec-ras river analysis system. <https://www.hec.usace.army.mil/software/hecras/documentation/>, 2019.
- [108] Abiodun Esther Omolara Kemi Victoria Dada Nachaat AbdElatif Mohamed Humaira Arshad Oludare Isaac Abiodun, Aman Jantan. State-of-the-art in artificial neural network applications: A survey. volume 4 of *Developments in Environmental Modelling*. Elsevier, 2018.
- [109] Leysia Palen and Amanda L Hughes. Social media in disaster communication. In *Handbook of disaster research*, pages 497–518. Springer, 2018.

- [110] Y.-S. Park and S. Lek. Chapter 7 - artificial neural networks: Multilayer perceptron for ecological modeling. In Sven Erik Jørgensen, editor, *Ecological Model Types*, volume 28 of *Developments in Environmental Modelling*, pages 123–140. Elsevier, 2016.
- [111] Karol J. Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM ’15, page 1015–1018, New York, NY, USA, 2015. Association for Computing Machinery.
- [112] Progeny Inc. Progeny systems). <https://www.progeny.net/about/>, 2020.
- [113] Rifaqat Zaheer Shahid Khan Rafiullah Khan, Sarmad Ullah Khan. Future internet: The internet of things architecture, possible applications and key challenges. 2012.
- [114] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.
- [115] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.
- [116] Alain Rakotomamonjy and Gilles Gasso. Histogram of gradients of time–frequency representations for audio scene classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):142–153, 2015.
- [117] Ajay Divakaran Regunathan Radhakrishnan and Paris Smaragdis. Audio analysis for surveillance applications. *Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005.
- [118] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.
- [119] Catherine Rentz, Colin Campbell. How elicott city flooded: A timeline.
- [120] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [121] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [122] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

- [123] Apaydin H. Band S. S. Mosavi A. Sattari, M. T. and R Prasad. Comparative analysis of kernel-based versus ann and deep learning methods in monthly reference evapotranspiration estimation.
- [124] Nitin Sawhney and Pattie Maes. Situational awareness from environmental sounds. 12 1998.
- [125] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.
- [126] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808, 2015.
- [127] Himan Shahabi, Ataollah Shirzadi, Kayvan Ghaderi, Ebrahim Omidvar, Nadhir Al-Ansari, John J. Clague, Marten Geertsema, Khabat Khosravi, Ata Amini, Sepideh Bahrami, Omid Rahmati, Kyoumars Habibi, Ayub Mohammadi, Hoang Nguyen, Assefa M. Melesse, Baharin Bin Ahmad, and Anuar Ahmad. Flood detection and susceptibility mapping using sentinel-1 remote sensing data and a machine learning approach: Hybrid intelligence of bagging ensemble based on k-nearest neighbor classifier. *Remote Sensing*, 12(2), 2020.
- [128] Nauman Shahid, Ijaz Naqvi, and Saad Qaisar. Characteristics and classification of outlier detection techniques for wireless sensor networks in harsh environments: A survey. *Artificial Intelligence Review*, 43, 02 2012.
- [129] Asaad Y. Shamseldin. Artificial neural network model for river flow forecasting in a developing country. *Journal of Hydroinformatics*, 12(1):22–35, 09 2009.
- [130] Jivitesh Sharma, Ole-Christoffer Granmo, and Morten Goodwin. Environment sound classification using multiple feature channels and attention based deep convolutional neural network, 2020.
- [131] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015.
- [132] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [133] Neha Singh, Bipendra Basnyat, Nirmalya Roy, and Aryya Gangopadhyay. Flood detection framework fusing the physical sensing & social sensing. In *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 374–379. IEEE, 2020.
- [134] Neha Singh, Bipendra Basnyat, Nirmalya Roy, and Aryya Gangopadhyay. Flood detection framework fusing the physical sensing and social sensing, 2020.

- [135] Neha Singh, Nirmalya Roy, and Aryya Gangopadhyay. Analyzing the sentiment of crowd for improving the emergency response services. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2018.
- [136] Neha Singh, Nirmalya Roy, and Aryya Gangopadhyay. Analyzing the emotions of crowd for improving the emergency response services. *Pervasive and mobile computing*, 58:101018, 2019.
- [137] Neha Singh, Nirmalya Roy, and Aryya Gangopadhyay. Localized flood detection with minimal labeled social media data using transfer learning. *arXiv preprint arXiv:2003.04973*, 2020.
- [138] Noah Smith, Michael Heilman, and Rebecca Hwa. Question generation as a competitive undergraduate course project. 01 2008.
- [139] CBS Baltimore Staff. Flood bot program uses social media to alert ellicott city residents to potential flooding. <https://baltimore.cbslocal.com/2020/09/21/flood-bot-ellicott-city-flooding-program/>, 2020.
- [140] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.
- [141] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning, 2018.
- [142] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.
- [143] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, USA, 2004.
- [144] Carlo Tomasi. Histograms of oriented gradients. 2015.
- [145] Twitter. [https://twitter.com/umbc\\_floodbot?lang=en](https://twitter.com/umbc_floodbot?lang=en).
- [146] UMBC. Flood bot: Umc researchers expand flood warning work in ellicott city. UMBC News, 2020.
- [147] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [148] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

- [149] Sun-Chong Wang. *Artificial Neural Network*, pages 81–100. Springer US, Boston, MA, 2003.
- [150] Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. Neural machine translation advised by statistical machine translation. *CoRR*, abs/1610.05150, 2016.
- [151] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [152] Kilian Q Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18, 2005.
- [153] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.
- [154] M. Weiser. The computer for the 21 st century. *Scientific American*, 265(3):94–105, 1991.
- [155] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks, 2015.
- [156] Wikipedia. 2016 maryland flood. Wikipedia, 2016.
- [157] Wikipedia. 2018 maryland flood. Wikipedia, 2018.
- [158] Piper Wolters, Chris Daw, Brian Hutchinson, and Lauren Phillips. Proposal-based few-shot sound event detection for speech and environmental sounds with perceivers, 2021.
- [159] www.weather.gov. Ellicott City historic rain and flash flood july 30 2016. <https://www.weather.gov/lwx/EllicottCityFlood2018>, 2016.
- [160] www.weather.gov. 2018 flooding Ellicott City & Catonsville. <https://www.weather.gov/lwx/EllicottCityFlood2018>, 2018.
- [161] Yahoo. Flood bot program uses social media to alert ellicott city residents to potential flooding. <https://news.yahoo.com/flood-bot-program-uses-social-215100993.html?guccounter=1>, 2020.
- [162] Maoling Yan, Pingzeng Liu, Rui Zhao, Lining Liu, Weijie Chen, Xueru Yu, and Jianyong Zhang. Field microclimate monitoring system based on wireless sensor network. *Journal of Intelligent & Fuzzy Systems*, 35(2):1325–1337, 2018.

- [163] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics.
- [164] S. Yun, S. Kim, J.Cho S.Moon, and T. Kim. “discriminative training of gmm parameters for audio scene classification.
- [165] Daniel Dajun Zeng, Hsinchun Chen, Robert F. Lusch, and Shu-Hsing Li. Social media analytics and intelligence. *IEEE Intell. Syst.*, 25:13–16, 2010.
- [166] Zhong-Qiu Zhao, Peng Zheng, Shou tao Xu, and Xindong Wu. Object detection with deep learning: A review, 2019.
- [167] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns, 2015.
- [168] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.
- [169] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2019.
- [170] Emre Çakır and Tuomas Virtanen. End-to-end polyphonic sound event detection using convolutional recurrent neural networks with learned time-frequency representation input, 2018.

