Machine Learning and Vision Using Edge Devices for Multimodal Chatbots and Bio-meteorological Sensing

by

Karthik K. Kulkarni

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2021 by the
Graduate Supervisory Committee:

Suren Jayasuriya, Co-Chair
Ariane Middel, Co-Chair
Hongbin Yu

ARIZONA STATE UNIVERSITY

August 2021

ABSTRACT

Machine learning (ML) and deep learning (DL) has become an intrinsic part of multiple fields. The ability to solve complex problems makes machine learning a panacea. In the last few years, there has been an explosion of data generation, which has greatly improvised machine learning models. But this comes with a cost of high computation, which invariably increases power usage and cost of the hardware. In this thesis we explore applications of ML techniques, applied to two completely different fields - arts, media and theater and urban climate research using low-cost and low-powered edge devices.

The multi-modal chatbot uses different machine learning techniques: natural language processing (NLP) and computer vision (CV) to understand inputs of the user and accordingly perform in the play and interact with the audience. This system is also equipped with other interactive hardware setups like movable LED systems, together they provide an experiential theatrical play tailored to each user. I will discuss how I used edge devices to achieve this AI system which has created a new genre in theatrical play. I will then discuss MaRTiny, which is an AI-based bio-meteorological system that calculates mean radiant temperature (MRT), which is an important parameter for urban climate research. It is equipped with a vision system that performs different machine learning tasks like pedestrian and shade detection. The entire system costs around $200 which can potentially replace the existing setup that costs $20,000. I will further discuss how I overcame the inaccuracies in MRT value caused by the system, using machine learning methods.

These projects although belonging to two very different fields, are implemented using edge devices and use similar ML techniques. In this thesis I will detail out different techniques that are shared between these two projects and how they can be used in several other applications using edge devices.

DEDICATION

To my parents.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 MOTIVATION

In recent years, the availability of large datasets combined with the improvement in algorithms and the exponential growth in computing power led to an unparalleled surge of interest in machine learning. Due to its ubiquitous application and versatility, it can be used in a variety of settings [76, 54]. What started as a computer program to solve simple but generic problems without explicitly programming it, machine learning has grown to new heights in last few years and continues to grow. Time series forecasting, weather prediction, stock market prediction, medical image classification, object detection and tracking etc. [78, 85] are some of the applications of machine learning. With ever-increasing data and continued improvements of the algorithms, these models continues to better their performance.

These models are complex and require high computing power. To support computation, special hardware like Graphical Processing Units (GPU) are leveraged along with Central Processing Unit (CPU), which results in high power consumption. The extra peripheries required to facilitate computation and power, makes the device bulky and expensive, thus making it difficult to use in applications involving smaller devices. This paved way for edge computing and machine learning applied on edge devices.

The Last decade has marked an era for cloud computing, making tremendous progress in infrastructure, security and software development[83, 93]. This progress is primarily due to the ease of applications not having to worry about their own

infrastructure to utilize compute power. Cloud computing works well for applications involving large data, on which processing and analytics can be carried over a period of time. For applications involving real-time inference, cloud computing can have high latency, putting critical processes at risk. Most cloud infrastructure is handled by a vendor, making it difficult for quick changes, adding to the risk of losing data and making the process expensive.

Edge computing on the other hand tackles these problems, making it a robust solution for applications involving Internet of Things (IoT) and real-time data processing [111, 59]. Any device at the edge of the network which produces and/or consumes the data can be referred as an edge device. These programmable devices support many different programming languages and operating systems, making them flexible for variety of applications. Since the computation is carried on-board the device, it has low latency and inference time perfect for real-time data.

There are many edge devices available, specifically meant for running machine learning algorithms [84, 62]. These devices are equipped with hardware and also low-level Application Programming Interface (APIs) to run complex machine learning algorithms. The chapters in this thesis will explore different applications built using these devices along with different machine learning algorithms. They will also discuss difficulties of using these devices and what methods to overcome them. Further, the applications explore two entirely different fields of study which showcases the versatility and flexibility of these devices.

1.1.1 Contributions and Outline

The main contributions of the thesis revolves around development and implementation of the following projects :

- ODO - A multimodal AI chatbot. ODO can be thought as an intelligent interactive entity capable of performing custom plays. It consists of a vision system to understand user's emotions and language system to understand text input and is supported by a movable LED system.

- MaRTiny- A low cost thermal sensing device. MaRTiny is a vision powered bio-meteorological station which calculates Mean Radiant Temperature (MRT). The weather parameters are supported by AI-based vision system which detects pedestrian, regions of shade and counts pedestrian in shade.

Outline: The first chapter provides a background to understand the technical details which will be highlighted in the consequent chapters. In the second chapter we detail out the collaborative implementation of ODO which involves three subparts: language system, vision system and communication system. In particular we discuss the language system in detail as I have worked on it extensively. We will discuss how we implemented different deep learning models on edge devices and started a new genre in the field of arts, media and theater. The third chapter we will discuss design, implementation and development of MaRTiny. We will discuss how the vision system deployed for ODO to understand emotions can also be deployed to perform other tasks like pedestrian counting. Further, we will also discuss how we used machine learning algorithm to estimate MRT value using only a few weather parameters with a higher accuracy than the existing methods. We will also discuss how we came up with a novel approach to detect pedestrian in shade. Finally, we will discuss limitations and future works of ODO and MaRTiny.

Chapter 2

BACKGROUND

2.1 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence [52]. AI is an interdisciplinary science with multiple approaches, but advancements in machine learning and deep learning are creating a paradigm shift in virtually every sector of the technological industry [33, 102].

Figure 2.1 depicts different branches of science and technology combined to make an AI system. Note that each of the these sub-system has its own roots in different domains which does not involve AI.

2.1.1 Machine Learning

Machine learning is a subset of AI, where a computer program is built to solve a problem without having to explicitly code for a given task, but rather the machine learns from the data it is trained on. Just like human intelligence, the machine learns from the data after being repeatedly trained on it. This is the reason why most of the models tend to perform better with more data and this also the reason machine learning has become so popular in last few years. But this is not necessarily true for all the models, in these case we perform "hyper-parameter tuning" to improve the performance [60].

While the concept of ML has been around for a long time (Enigma Machine during WWII), the ability to automate the application of complex mathematical calculations

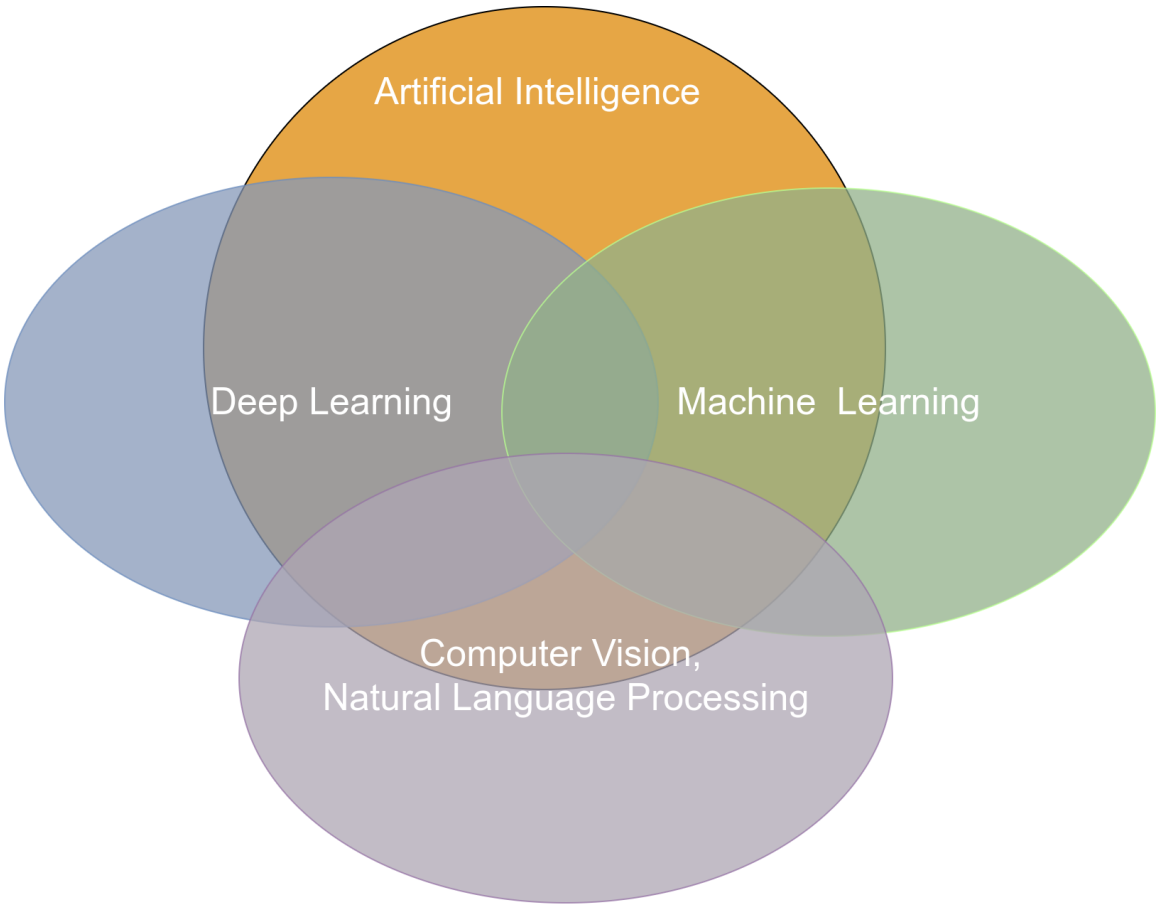


Figure 2.1: Different Branches of Artificial Intelligence

to big data has been gaining momentum over the last few years. In a high level, we can understand ML to be a very complex mathematical function which is performed by the machine based on the input data.

Learning: There are two main types of learning:

- Supervised Learning
- Unsupervised Learning

The process of supervised learning starts with inputting training data into the selected algorithm. The model initially generates random weights and biases (these are basically co-efficient of the equation) and updates these values with every training

data. The output is compared with the ground truth and the difference between them is calculated. This suggests how well a machine is learning a particular task. This is defined as loss function and there are many different types of loss functions. This difference is used to update the weights and the process continues for all the input data, many times [94, 77].

The process of unsupervised learning is similar to that of supervised learning, except there is no ground truth for data comparison. The loss function instead uses other means of understanding the data and their relation with each other. Clustering is one such example of unsupervised learning [120, 49].

In our work we have mainly used supervised learning for prediction, pedestrian detection, shadow detection, MRT calculation which are explained in Chapter 2 and 3. But we have also used clustering algorithm, which is explained in Chapter 2.

2.2 EDGE COMPUTING AND IOT

For a long period of time, centralized cloud computing has been considered as the standardised IT delivery platform. Without having to worry about the physical location of the servers, its scalability and security, the centralised architecture of the cloud makes makes it convenient for the end user [93]. In recent times, there has been rise in the demand for faster computation and lower latency especially for applications involving IoT and real-time data processing. This can be achieved using edge devices, where computing is carried out on the devices which are at the edge of the network. [84, 62]

While there is no definitive definition, edge computing can be thought as a distributed system where data collection and processing is carried out at the edge, where things and people produce or consume this data. "Put another way, edge computing brings the data and the compute closest to the point of interaction." - E.G Nadan,

Chief of Red Hat technology. Below are the few of the edge devices that were leveraged in our projects.

Nvidia Jetson Nano: We chose Jetson Nano because of its compact size and high computational abilities. With the help of the on-board 128 CUDA core Maxwell GPU we can run various DL algorithms. It is also equipped with low-level API - TensorRT using which we can convert dense layered models into low-level graph models executable on the on-board GPUs. It is built using ARM Cortex A-57 MPCore processor along with 4GB RAM which provides enough computational capabilities for our projects. The operating power is 5V and 4A, making it a low-powered device which can be operated using portable power-supplies. It aslo has Camera Serial Interface(CSI) port for camera connection which is used in both our applications to perform various vision task like emotion detection and pedestrian detection.

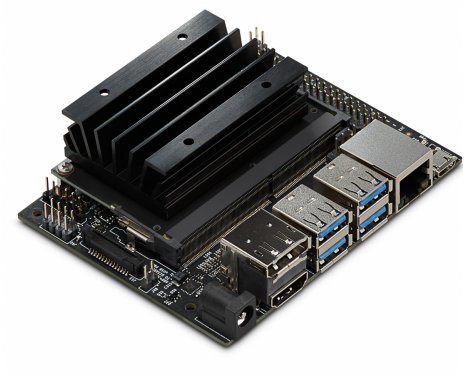


Figure 2.2: Jetson Nano

Due to these hardware and software features, jetson nano is used in several application like object detection and tracking [88, 114], AI traffic control system [103], energy efficient analysis [63] etc. covering various different fields. In Chapters 2 and 3, we will discuss how this device was put to use to perform language understanding and visual recognition tasks using deep learning algorithms.

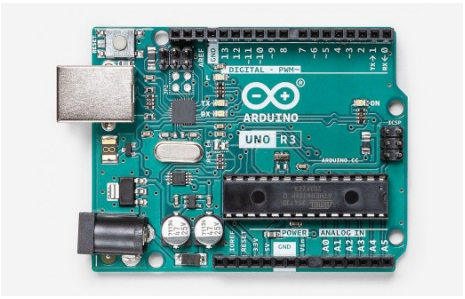


Figure 2.3: Arduino Uno

Arduino Uno and NodeMCU: Arduino is an open-source platform used for constructing and programming of electronics. It can receive and send information to electronic devices, and even through the internet to command the specific electronic device. It uses an Arduino Uno circuit board and software program (Simplified C++) which can be used to program the board. The board does not have a separate piece of hardware in order to load new code onto the board, one can simply use a USB cable to upload. Some of the exmple applications are home automation [1], smart traffic system [48], smart irrigation system [47], car controlling systems [15], heart rate monitoring system [80] etc.

The lack of WiFi module on the Uno board makes it a standalone on-primes IoT device. With only a temporary storage space on uno it becomes necessary to use other micro-controllers to transfer data to a remote database. NodeMCU is one such edge device which is compatible with uno board. Developed on ESP8266 architecture, this single board micro-controller has in-built WiFi, flash memory, and supports the PEM (Privacy Enhanced Mail) file system. These features makes it an ideal edge device connecting the IoT devices to the internet securely. The data from uno board is transferred to NodeMCU using serial communication, which is further transmitted to databases using MQTT protocol. In Chapter 3, we will discuss in detail how we leveraged this communication bridge to transmit various sensor data to AWS databases.

2.2.1 MQTT Protocol

Message Queuing Telemetry Transport (MQTT) is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth.

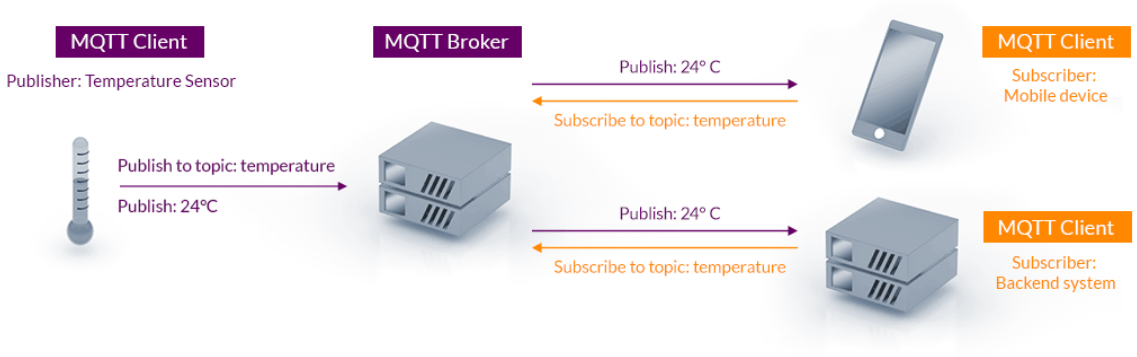


Figure 2.4: MQTT Protocol Architecture

Figure 2.4 explains the general architecture of the protocol. The broker holds the responsibility to initialize communication, create a bridge and transfer data between different clients. Although the broker can be developed using another edge device, we have leveraged an API from AWS to establish this communication. In chapter 3 we will detail out how the sensor data from uno board, which acts as a client is transferred to online databases through AWS APIs.

Chapter 3

ODO

In this chapter, I detail our efforts in making a multi-modal AI chatbot using Nvidia Jetson Nano for use in an interactive theater performance. This chatbot has an architecture consisting of vision and natural language processing capabilities, as well as embodiment in a non-anthropomorphic movable LED array set in a stage. Designed for interaction with up to five users at a time, the system can perform tasks including face detection and emotion classification, tracking of crowd movement through mobile phones, and real-time conversation to guide users through a nonlinear story and interactive games. The final prototype, named ODO, is a tangible embodiment of a distributed multimedia system that solves several technical challenges to provide users with a unique experience through novel interaction.

3.1 BACKGROUND

3.1.1 Motivation

With the advances in artificial intelligence, especially natural language processing, machine agents serving as natural language user interfaces (i;e chatbots) have become highly sophisticated. These chatbots have seen widespread use in client communication and customer services . While some of the traditional criticism of these chatbots include their limitations of understanding the language and expressions used by human beings, there has also been work to try and improve the effective computing underlying chatbots [10, 121]. This has allowed them to be deployed in diverse forward-facing settings including psychological counseling and therapy services [117].

However, there is a lack of research on enabling chatbots for artistic installations, particularly chatbots combined with existing multi-media setups including distributed audio, video and illumination systems. This is primarily due to two main challenges: (1) existing chatbots can rarely integrate multimodal sensory inputs such as combined audio and video to perform their functions, and (2) existing experiential media systems typically feature primitive or non-intelligent signal processing or algorithmic controllers, which rarely can adapt to users in semantically meaningful ways.

Further, there is interest in making these systems energy-efficient to reduce the carbon footprint and enable new use cases for the technology. Edge computing devices have recently gained sophistication in their capability to handle complex workloads including that of image and audio processing necessary for embedded machine learning. This is an emerging paradigm that can be leveraged by experiential multimedia systems for maximum impact.

Inspired by these challenges, we have endeavored to create a novel, multimodal chatbot that can successfully integrate with an experiential media system. The goal of this system is to provide a new, interactive experience for multiple users, and designed for digital and interactive theater. Our chatbot is able to perform crucial vision and natural language processing to communicate with end users, while still communicating and actuating a distributed media system to change the environment and ambience settings.

3.1.2 Related Work

Our work draws upon a vast literature in the areas of intelligent personal assistants, experiential media systems, and the deployment of computer characters in theater productions. We situate our work with ODO at the intersection of these three areas, comparing similarities and differences.

Intelligent Personal Assistants. There has been several recent developments in the area of intelligent personal assistants (IPAs) [91] include the creation of Amazon Alexa and Google Assistant. In this project, we will refer these IPAs as chatbots for ease of use. There exist many currently available frameworks for chatbots including OpenDialog [55], Google Dialogflow [89], IBM Watson [90], and Microsoft Bot Framework (MBF) [65]. While all of these options feature advanced functionality especially in their NLP (Natural Language Processing) capabilities, we decided to implement our own chatbot with its own custom dialogue management system in the Python programming language. Our reason for doing so is maintaining an easy-to-use, simple code interface that could interact with our multimodal stage and to create a non-linear model of storing-telling. Our chatbot was heavily modeled after the Amazon Lex framework [113], but we needed to bypass the Amazon Web Services (AWS) database in order to do custom TCP/IP communications for our system and linear setup of interaction. Recently, the use of deep learning and neural networks within these chatbots has led to improvements in state-of-the-art performance [32, 92, 95, 105]. We also augment our chatbot with deep learning networks, particularly in computing word similarity for improved NLP performance, but also adding vision capabilities including facial detection/recognition and emotion classification and crowd clustering and segmentation.

The advances in chatbot technology has in turn spurred a renaissance of research on the topic [24], including human-computer interaction [7, 30, 82]. Many studies have focused on the effective components of interacting with chatbots [20, 121] and how humans language use changes when interacting with them [38]. In this chapter, we are focused on the implementation details of our chatbot. A full HCI study analyzing the efficacy of our system for user engagement is the subject of future work.

Experiential media systems. We follow the framework introduced by [97] of experiential media systems. Such systems build on the foundation of experiential computing [43, 25], and typically feature sensing, computation, and multimodal feedback to the users in the system. Experiential media systems have been built for purposes such as biofeedback for stroke rehabilitation [16] and studying coordinated group activity [45]. One potentially similar experiential media system is the "Sound and Light" festival held on the Island of Rhodes, where multimodal interaction uses stereophonic light and sound to tell the medieval history of Rhodes without actors. Our system differs from this particular example as we involve human audience participants to interact with an AI chatbot as part of the performance. ODO is an experiential media system which utilizes a physically-instantiated matrix of 26 LED lights that can move up/down on the stage coupled with displays and speakers to give users an interactive theater experience. In particular, the feedback given to the user is driven by the chatbot, thus yielding a level of personalized feedback that is rare among typical experiential media systems.

Autonomous computer characters in theater. There has been an active thread of research using autonomous computer characters in theater productions. In [81], the first computer character was introduced to interact with a human actor on stage. In this work, the computer uses gesture recognition to analyze a human actor's movements, and then digitally projected a virtual character onto a screen to advance the story. Following this line of work, there have been several more uses of robots in theater [51, 39, 66] including exploring improvisation and non-scripted behavior [67]. All these examples were designed to give the automated character presence on the stage, helping to draw the audience's attention and engage them.

In contrast to these previous works, in our production, ODO is an invisible character that is not represented by any one device, but which manifests itself only through voice and ephemeral events. 26 lights moving in space form a cloud which triggers visual associations utilizing a LED stage. At every stage, users are encouraged to use their imagination to have a complete experience of the plays. Rather than a limitation of the system, this is an intentional benefit to trigger associations for the audience and also to avoid representational issues with giving the chatbot a type of distributed embodiment rather than a single body/terminal which limits user interaction. One of the main aims of post-dramatic theater is to question the central role of the text or script in a performance. Rather other elements of theater include movement, image and sound, and the new post dramatic theater dramaturgy puts all media elements of the stage on the same equitable level with no hierarchy. With this, ODO can be described as a post-dramatic digital character, using all media elements on stage including the underlying chatbot to realize its presence.

3.2 SYSTEM OVERVIEW

Our system consists of several key components including the chatbot itself, its computer vision sub-system for face detection/recognition and emotion classification, and the crowd tracking through mobile phones. All of this is wrapped together through a communication protocol designed for real-time responsivity and feedback with the users. A full diagram of the system architecture can be seen in Figure 3.4b. In the following subsections, we describe the design choices for each of the subsystems that make up the entire architecture.

3.2.1 Chatbot

Designs of chatbots can range from the conventional, knowledge-based architectures to more free-ranging, open models. In our case, the design requirements of the chatbot include: (1) the ability to interpret user behavior and inputs to provide appropriate responses, and (2) a "character" or dramatic identity to aid the theater performance. Thus we built our own chatbot from scratch, inspired by the design of the Amazon Lex chatbot.

Intent Modeling for Storytelling:: Stories are stored as a collection of intents and each intent is a pair of utterances and responses. Utterances are set of sentences which probably a user can utter to the chatbot. These utterances will trigger its associated responses which are nothing but the story lines. Depending on the utterance, different story lines are presented to the user. For instance when the user utters words/sentences like "plane", "go for plane", "fix the plane", triggers the response, which is one of the dialogue "let's fix the plane and then search water". These both are wrapped in one collection called intent (in this case intent name is "plane"), which basically checks the intention of the user behind uttering their words. In this way, a conventional chatbot is converted into a theatrical story teller.

While simple keyword spotting could give some base functionality for our chatbot, we implemented a deep learning network for natural language Understanding to help ensure that the system could robustly identify synonyms and other utterances that are similar to the desired intents. To process the words/sentences uttered by the user and map them to the utterances that we have stored, the chatbot uses a pre-trained Deep Learning model - USE (Universal Sentence Encoder) [11]. This takes the user input, performs word embedding (namely word2vec [75, 74]), combines the words using composition function and then finds similarity between the two sentences and

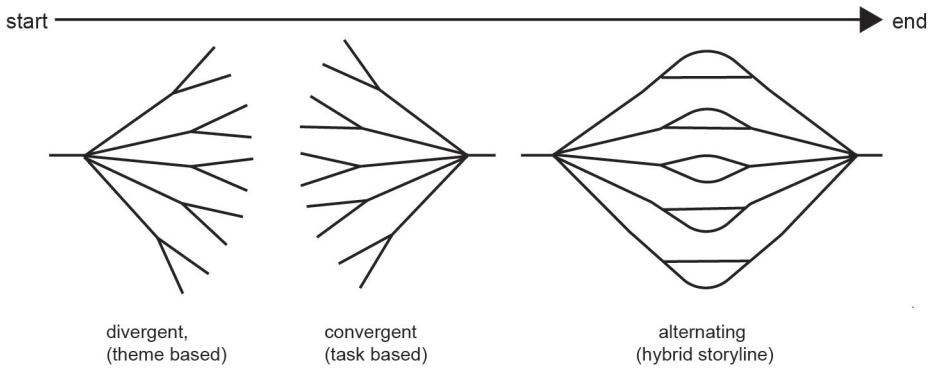


Figure 3.2: Conversational Strategies for Chatbots

the one which has highest similarity is chosen. For instance, sentence like "repair the plane" and "fix the plane" basically have the same meaning. This similarity is understood by using USE and this way the user doesn't have to utter the exact words/sentences, giving them the flexibility to converse with the chatbot. For out of domain utterances, if the similarity score was below a certain threshold, a standard response was selected which reset the conversation or returned back to the previous intent in the conversation.

Conversational Strategies: Conventional chatbots are either convergent or task-based (e.g. Amazon Lex, query systems) or divergent/theme-based (e.g. Pandorabot's Mitusku [79]). Both these conversational strategies do not fit to a conversation in theater, where the main goal is to develop a conversation based on a given story or dramaturgy.

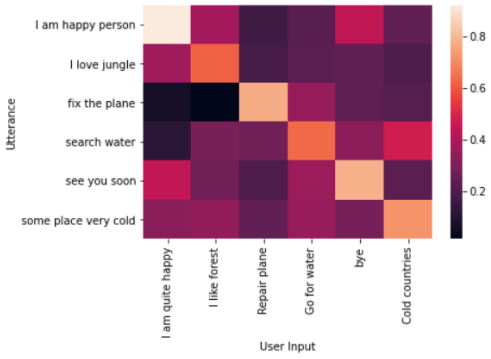


Figure 3.1: Sentence Matching Using the Universal Sentence Encoder

We tested branching hierarchies and nonlinear storytelling strategies till we found a combined system of convergent and

divergent conversational strategies which managed to both involve an audience in a conversation and also drive the story. A chatbot uses utterances, intents and entities or slots to structure a conversation. The utterance is the human to machine conversation to find out an intent of a conversation, which leads to the entities or slots to set cornerstones of a conversational path. In a theatrical conversation theses paths have to be both divergent and convergent, but they are set in specific times which develops a specific chatbot dramaturgy. It resembles a road map connecting cities, where a main road branches out into meandering smaller streets entering a city. Inside the city the map is two dimensional and nonlinear. Leaving the city the streets need to converge back into a linear system of one main road to connect the next city or hub. We visualize this strategy in Figure 3.2

For the nonlinear conversation part, the chatbot was given the ability to interact with the audience through physical games and interaction with the system, or recite haikus to the users. This gave a whimsical nature to the chatbot that is refreshing and charming, leading the user through the story without feeling forced or rushed towards the end goal, at least in design.

3.2.2 Emotion Detection

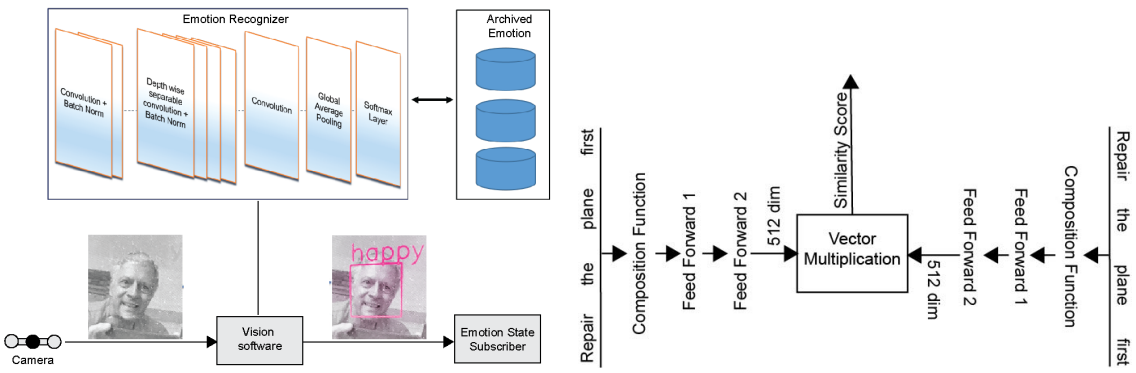
Along with language understanding, we empower ODO with vision capabilities to understand the emotion of the users. We primarily use two video feeds from 160 degree cameras and design the system to accommodate 5 users at a time. We chose 5 users because this was the maximum number of users that we could accommodate for our interaction games which is a main focus of ODO's performance. Our vision subsystem consists of the main task of (1) detecting the faces of these five users, and (2) performing emotion classification to help aid the chatbot in performing effective decision making in its responses to user utterances.

Reading images coming off a live camera stream, the system first performs face detection using Haar feature-based cascade classifier [106] built into the OpenCV library. After face detection, emotion estimation is performed using a trained deep learning emotion recognition model based on a convolutional neural network [5]. Every detected emotion is classified in one of angry, disgust, scared, happy, sad, surprised, and neutral. While we acknowledge such a coarse, discrete categorization is not indicative of the range of human emotions and subtleties that can occur in a theater performance, we found this enabled a tractable computational platform to perform decision making for the chatbot.

The software archives the detected emotion of the audiences and shares this information using a client/server architecture and Transmission Control Protocol/Internet Protocol (TCP/IP) protocol. Each of the vision sub-systems is identified by their name and the chatbot can request the emotion of a particular vision system having a specified field of view or the consolidated emotion estimate of all the vision subsystems. Moreover, the chatbot can request the current emotion index or the average emotion for a specified period. In our current deployment we have two vision subsystems to capture and estimate the emotion of the audience.

3.2.3 Crowd Clustering

In addition to visual data collected of the users, the chatbot also collects gestural and movement data via users' mobile devices. This yields a lot of opportunities for the chatbot, including the possibility to play movement games and other embodied practices within the context of the physical system that the chatbot is embedded in. To do so, we develop the notion of a crowd index system.



(a) Vision (b) NLP

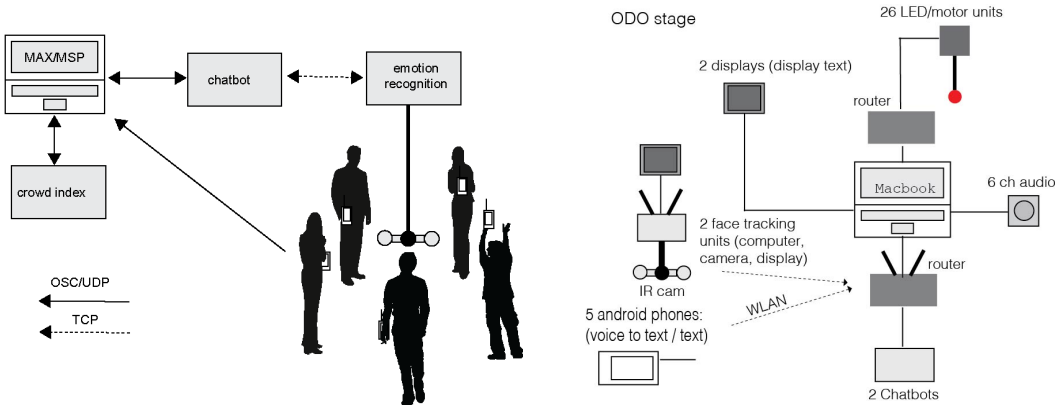
Figure 3.3: Vision and NLP Architectures for the Chatbot

The crowd index system communicates with the camera using the OSC protocol. The crowd index continuously receives the location data captured by the user device, via an intermediate software known as Max/MSP [22]. The crowd index system uses this data to make an estimate of the crowd behavior and shares it with the max system whenever it is requested. The system leverages the Density-Based clustering (DBSAN) algorithm to estimate the number of clusters formed by the audience and to identify the presence of isolated audience members [29]. Moreover, the software computes the velocity and acceleration on a per audience basis, based on their real-time location data to get an estimate of the energy and agility among the audience.

3.2.4 System Communication via Max/MSP

The entire system is necessarily complex and thus needs to be coordinated with a global protocol and communication scheme to be effective. A key design consideration for us is that we need the system to be interactive rates (only a few seconds of latency), so that users do not get frustrated when interacting with the chatbot.

The Max/MSP system is the backbone for communication in ODO. It receives user chat, location and other sensory information and forwards it to the other destination



(a) System Architecture (b) Hardware Connectivity

Figure 3.4: System Architecture for ODO Including the High Level Block Diagram

systems for processing. The chatbot, along with vision sub-system and crowd index sub-system, processes the Max input and communicates back to Max with the result metric.

Max/MSP system uses OSC (Open Sound Control) for communication [31]. OSC adds a layer on top of a UDP (User Datagram Protocol) to standardize information transfer between media systems. Being a connection-less protocol, it decouples the chatbot from the Max/MSP, making the system modular. The vision sub-system communicates with the chatbot over TCP/IP (Transmission Control Protocol/Internet Protocol). This has been done to closely integrate the system with chatbot, as the vision sub-system will only start after the chatbot. The crowd index system is loosely coupled with the chatbot via the OSC interface.

Figure 3.5 depicts the overall system call flow diagram. Upon start the chatbot triggers a request to the vision system to start the camera and receives an acknowledgement. Next the chatbot system sends a trigger to the Max system to start the conversation and receives the acknowledgement. The max system receives the user input over the OSC interface and forwards it to the chatbot to get a response. The

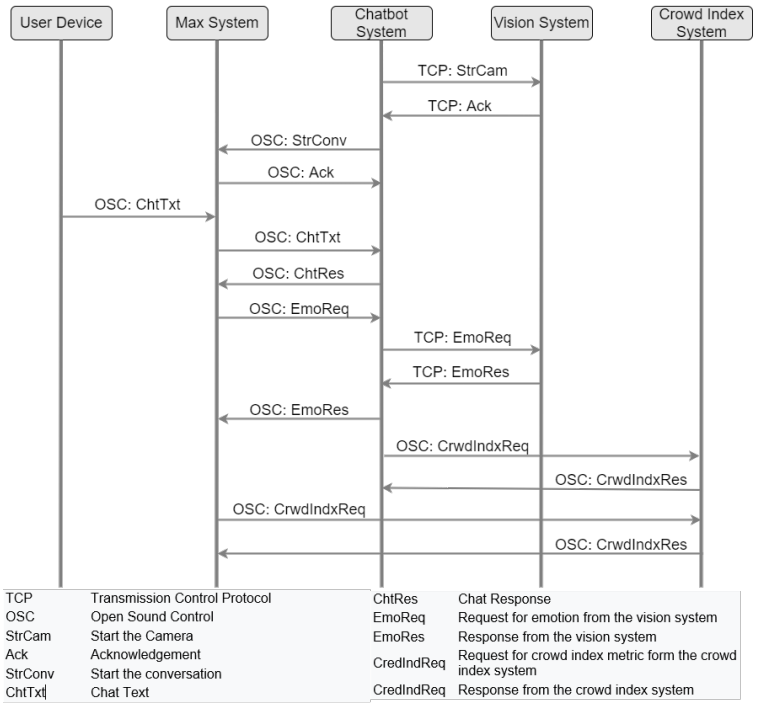
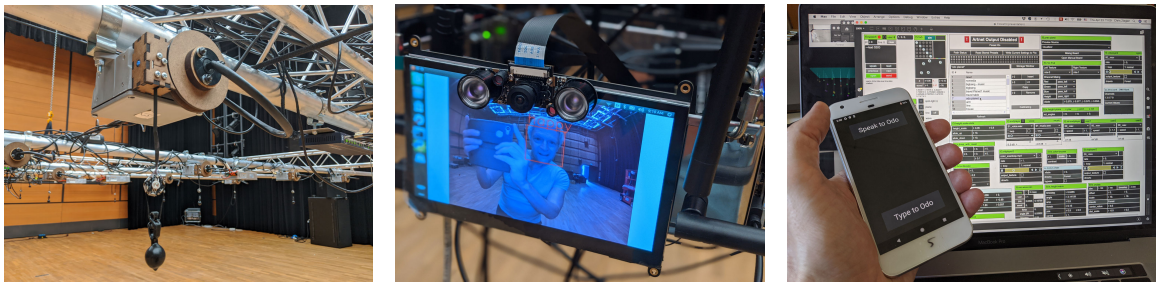


Figure 3.5: Call Flow among the Components of the ODO

response is then shared with the user. Additionally, the max system and the chatbot can request the vision system to get the emotion estimate of the audience and receives the response. Finally, whenever required the max system requests for the crowd index system for the mobility related metric and receives a response.

3.3 IMPLEMENTATION

In this section, we discuss implementation of the system and how we achieve the desired design for our chatbot. The overall system performs all of its computing locally on either the computers or the Nvidia Jetson Nano rather than user's phones. This avoids the latency due to round trip back and forth from a cloud server due to streaming video, which we experienced in a previous iteration of the system using the Amazon DeepLens for our vision processing. Finally, we utilize LAN based communication to setup our own internal network in which to run the entire system.



(a) Motorized LED (b) Camera Setup (c) Max Software Setup

Figure 3.6: ODO Setup on Stage

3.3.1 Stage Hardware

For the chatbot, we are using 4 NVIDIA Jetson Nanos (NVIDIA Jetson Nano Developer Kit Item Model Nr 945-13450-0000-000, Arm Cortex A57) along with two 160 degree IR cameras (Waveshare IMX219-160 IR Camera). Each Nano is used to perform certain task and is distributed to avoid overloading of computation on devices. The chatbot runs on the master Jetson Nano which encapsulates the code for the chatbot, communication code for the 2 cameras, and communication code with the MAX system. Each of the cameras is connected to a separate Nano, which captures and processes emotions of the crowd. We are also monitoring movement of the crowd though a mobile application which communicates with the 4th Nano. Clustering of the crowd is calculated using their relative positions on the stage. Both crowd emotions and crowd clustering are then sent to the chatbot, where the chatbot engages with the crowd depending on the crowd behaviour.

Multi-modal Stage: The chatbot is embodied physically in a system on a black-box theatrical stage. The stage size is 8meters $\times$ 6meters with a height of 5 meters. Motorized lights are hung from a scaffolding and truss system as shown in Figure 3.6a. The entire stage is covered with these double sided LED lights, connected to motors to move the lights vertically which is shown in Figure 3.7, which immerses the audience

in light and movement. This stage is similar to the forest3 stage by Ziegler et al. used in the dance production COSMOS [18, 19]. The lights are controlled with ARTNET, a network protocol for theatrical control over UDP/IP. The Max/MSP light and motor control software uses GPU accelerated rendering techniques to compute the output animation of the physical LED units with minimal latency. All of the animations can be mixed before final ARTNET output to the lighting network, affording artists a degree of variability in visual design without the need to implement or add new components to the software. This dedication to efficiency and flexibility enables rapid prototyping for designers of exhibitions and modalities of interaction.

Computers and cameras are hung from the center of the truss construction. The audiences position and movement data is sent to a central host, which relays the data to the chatbot network. A network of 4 NVIDIA Jetson Nanos with one Mac computer communicate with Open Sound Control (OSC) [31] via LAN and WLAN. 5 Android phones with a custom app sends text and motion data from each user to a host computer, which is coordinated through the Max/MSP system, as shown in Figure 3.6c. Two camera-based linux systems feed face tracking data to the chatbot which sends text and metadata of the conversation to the central host. An additional external crowd index computer permanently analyzes motion and position data from the audience and sends movement vectors and velocity data of each audience member to the host computer. The central computer feeds two main displays on stage with cluster index which signifies cluster and separation of audience on the stage.

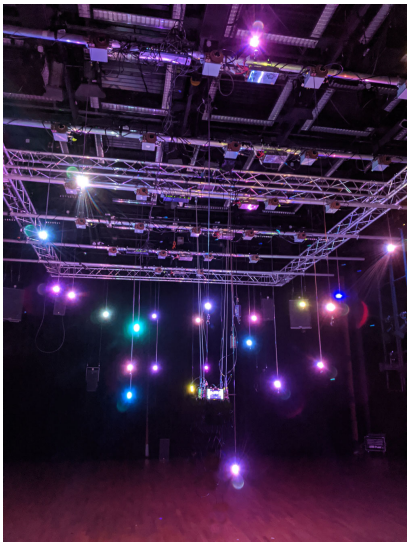


Figure 3.7: Moving LED Stage

3.3.2 Algorithms

NLP Architecture: Stories are divided into a set of different intents which consists of a pair of utterances and responses. Utterances are the the possible words/sentences that a user can use to converse with the chatbot, which triggers responses from the chatbot which are story lines from different section of the story. We utilize Google's Universal Sentence Encoder-USE [11] to understand users inputs. Universal Sentence Encoder gives the similarity index between the input and utterances. If a similarity index above the threshold (we have used 70%) is calculated, then the corresponding response of the matched utterance is triggered, otherwise the chatbot will request for a new input relevant to the story. This way conversation is focused on a given topic of story and doesn't meander out. Figure 3.1 shows the correlation between user input text and saved utterances using the network. The utterance with highest correlation coefficient is selected and the corresponding story intent is executed.

Since there can be same/similar utterances for different intents corresponding to the different story-lines, it is difficult to identify the right intent as this scenario can trigger multiple intents. To avoid this, we utilize weighted intents to uniquely identify the correct intent to trigger in the story. As the story progresses, weights are updated and is tracked, this way the story always proceeds forward and does not loop back into previous story lines even utterance matches.

Vision architecture: To detect the emotion we use a trained fully convolutional neural network-based mini-Xception model [6, 5]. The mini-Xception architecture as proposed in [6] comprises of a fully convolutional neural network that comprises of 4 residual depth-wise separable convolutions [17] with batch normalization [42], ReLU activations, and a last layer of global average pooling and a soft-max activation

function to get the prediction of the emotion. Owing to elimination of the fully connected layers and use of depth-wise separable convolutions, the mini-Xception model has low latency in detecting the emotion, which is paramount in detecting emotion in a live video feed, and hence we use it for our vision sub-system. The architecture has approximately 60,000 parameters which is much less compared to other architectures and henceforth makes it lightweight and fast in the prediction. The model was trained on the FER 2013 data set [34] and the accuracy was reported to be 66 percent. We found this accuracy to be suitable for our purposes as emotion data would only be periodically be queried by the chatbot in the performance, and thus any mistakes in classifications were sparse in the actual performance. Future work could try to improve the emotion recognition by finetuning the network on captured data from the system if absolute accuracy was desired from the performance.

Once the software detects the emotions for a frame it is archived along with the current timestamp. This archived information is used to make a response to the request for an emotion metric. To facilitate this, the vision sub-system software hosts a TCP/IP server and makes a response to the client by using the request-response messaging pattern. When the server receives a request from the chatbot, it shares the emotion metric with the chatbot.

Crowd Index System: As depicted in Figure 4 the crowd index system has software running on an Nvidia Jetson Nano. It receives the real-time sensor information captured by the User Device. The received data comprises of the audience's real-time location. The Crowd index receives this captured location of the audiences in real time. To compute the Crowd index, we use the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [29] clustering algorithm which is characterized by two parameters: ϵ and MinPoints. ϵ defines what point should be considered as within a proximity and MinPoints defines the minimum numbers of points required

to form a cluster. We choose the distance metric as Euclidian distance and the value of the ϵ is based on the dimension of the stage and a reasonable definition of proximity distance. The MinPoints is chosen as 2 and thus cluster can be formed by two or more people. The noise points is used to identify an isolated audience member. In addition, we also calculate the velocity and acceleration of each user from their phone, which is also forwarded to the main system via OSC.

There may be some error introduced into the location estimation algorithms based on noise in the smartphone sensing. However, since our crowd index measure uses density-based clustering, the overall algorithm is resilient to individual errors in the localization, and the chatbot further only uses the crowd index to help guide the audience towards completing the interaction games at a coarse location granularity (meter scale in our implementation). Thus we did not find that noise significantly affects the system performance.

3.4 RESULTS

3.4.1 ODO Conversations

ODO uses its multiple sensing modalities synergistically to interact with up to five given audience members at a time. A text-based conversation is limited like every other natural conversation to one partner, but in games and physical interactions the chatbot can interact with up to 5 users at the same time. At the start of the conversation, ODO asks each audience member for their names. ODO leads the conversation dialog. The audience members use either a text-based or voice-to-text application on their mobile phones to send system input. These are stored in the order in which they are sent, and are responded to in turn by the system. Since ODO has multiple responses for any particular intent, there is a very low probability of

repeated responses from ODO due to multiple similar inputs. However, having ODO respond to each participant in turn does limit the ability of the system to have a long, extended conversations with each user. At the same time as the conversation, the apps also send movement and position sensor information continuously. Most of the script for ODO tends to encourage somatic and physical games for the audience leading through a story of a journey, while the chatbot cannot leave the stage physically. The play is based on the premise that ODO is aware of being stuck on stage. The chatbot needs to communicate and to play with the audience to grow as a character. At the end of the performance ODO thanks the audience for sharing time and space to "create a world together".

The vision cameras track up to 5 faces and perform emotion recognition on these faces. Typically the five emotion values are combined together via majority vote so that a single emotion value is held for a particular time period, which the system uses to inform its responses as a particular intent. The audience is perceived as one audience body. At the same time, the crowd index system delivers an estimate of the crowd activity, which is periodically sent every minute to the chatbot that will trigger unique responses in the output during the performance. Thus the chatbot alternates between responding to user input, emotions received from each vision system, and the crowd index with a combination of asynchronous (as they arrive) as well as synchronous (periodic emotion queries, etc) behavior in the order in which these intents are triggered. In Figure 9, we show an example of the output of the system for a sample conversation of ODO. We can see ODO exhibiting several of the conversational strategies described earlier, including working with the emotion data to inform its responses at the end.

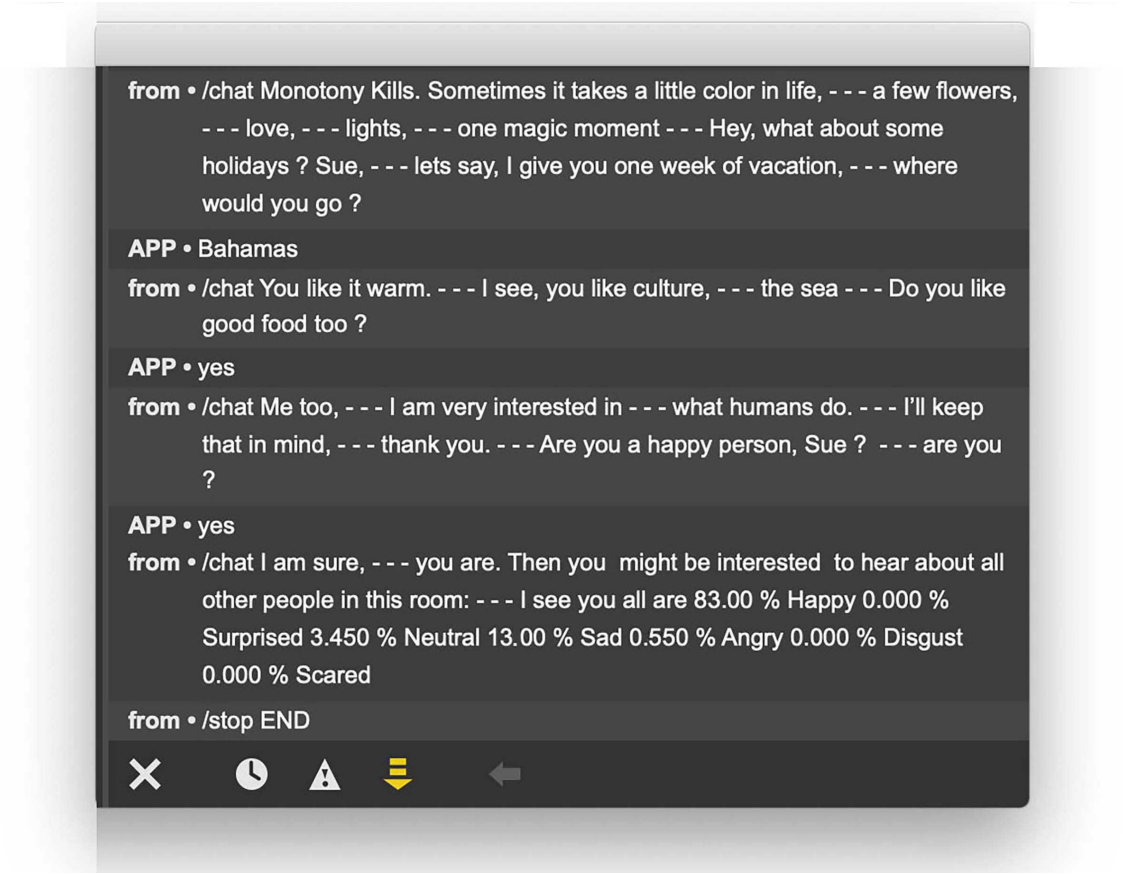


Figure 3.8: Example of ODO Conversation

3.4.2 Interaction and Games

One of the main user interactions that ODO performs is playing games with the audience members. 3D Games usually are programmed for user interfaces, performing in virtual space. Using the LED lights described in the implementation section, the following games are implemented in the system using position and motion data from 5 users. Each of these games are triggered periodically in the story based on the playwright's script, as ODO is led out of the cave with the help of the audience. Users mainly use their phone to communicate and perform tasks at the game.

- POINT Game: One users controls the z / height of one light by using position and z / height information via magnetic / pressure sensor information.
- LINE Game: Two users generate a line of lights between them controls the z / height of both ends of the line.
- WAVES game: The acceleration (x,y,z) information of all users is averaged and mapped to the speed of a perlin noise generator controling motion movement of all 26 lights.
- PLANE game: Three users control the 3D orientation of a flat plane of lights. The host receives the x and y position of each user and z / height information via magnetic / pressure sensor information.
- LANDSCAPE game: up to 5 users can move into the center field of stage between the 26 lights and build valleys and mountains using a terrain builder algorithm which uses the user's x and y position and z / height information via magnetic / pressure sensor information in the APP to shape a 3D landscape terrain model.

3.4.3 ODO Conversation

ODO uses its multiple sensing modalities synergistically to interact with up to five given audience members at a time. A text-based conversation is limited like every other natural conversation to one partner, but in games and physical interactions the chatbot can interact with up to 5 users at the same time. At the start of the conversation, ODO asks each audience member for their names. ODO leads the conversation dialog. The audience members use either a text-based or voice-to-text application on their mobile phones to send system input. These are stored in the

order in which they are sent, and are responded to in turn by the system. Since ODO has multiple responses for any particular intent, there is a very low probability of repeated responses from ODO due to multiple similar inputs. However, having ODO respond to each participant in turn does limit the ability of the system to have a long, extended conversations with each user. At the same time as the conversation, the apps also send movement and position sensor information continuously. Most of the script for ODO tends to encourage somatic and physical games for the audience leading through a story of a journey, while the chatbot cannot leave the stage physically. The play is based on the premise that ODO is aware of being stuck on stage. The chatbot needs to communicate and to play with the audience to grow as a character. At the end of the performance ODO thanks the audience for sharing time and space to "create a world together".

The vision cameras track up to 5 faces and perform emotion recognition on these faces. Typically the five emotion values are combined together via majority vote so that a single emotion value is held for a particular time period, which the system uses to inform its responses as a particular intent. The audience is perceived as one audience body. At the same time, the crowd index system delivers an estimate of the crowd activity, which is periodically sent every minute to the chatbot that will trigger unique responses in the output during the performance. Thus the chatbot alternates between responding to user input, emotions received from each vision system, and the crowd index with a combination of asynchronous (as they arrive) as well as synchronous (periodic emotion queries, etc) behavior in the order in which these intents are triggered. In Figure 9, we show an example of the output of the system for a sample conversation of ODO. We can see ODO exhibiting several of the conversational strategies described earlier, including working with the emotion data to inform its responses at the end.

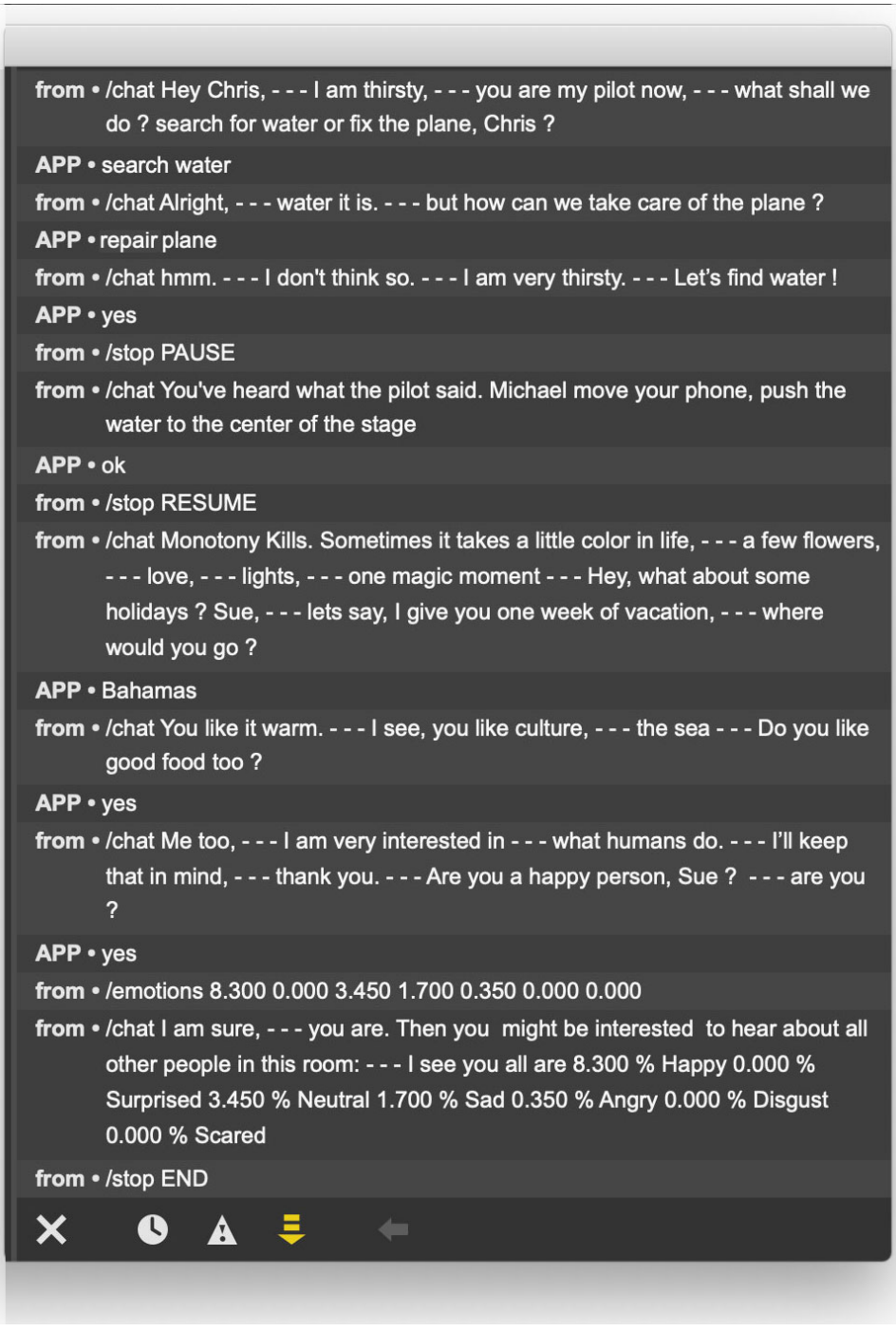


Figure 3.9: Example ODO Conversation

Average Latency and bandwidth of the System	
Max-Chatbot	10 ms
Max-Sentence Matching	02 ms
Max-Haiku Poem	60 ms
Max-Emotions	42 ms
Max-Crowd Index	1470 ms*

Table 3.1: Latency of Different Sub-systems in the Chatbot

3.4.4 Latency

We did experiments to quantify the latency of the system. We have used iPerf [100] to calculate the available bandwidth among the components of the overall system. We observed the available bandwidth to be 930 +- 5 Mbps among all the system components which need to communicate in the ODO's intranet. Since all the computing happens on the edge, we do not need internet communication. The latency including the request and response time is computed and is depicted in the Table 3.1. As we show, the latency of most of the sub-systems are at interactive frame rates, and thus are not significantly noticeable for the human users. While it is difficult to compare these numbers to any benchmarks or baseline numbers, we note that milliseconds are well within the expectation for an interactive system, and the crowd-index, the only operation which is longer than a second, is only periodically queried every minute in the background by the system.

3.5 DISCUSSION

ODO is a theater production with a multi modal stage, which challenges the audience to play an active role as listeners and participants. In this production the audience plays a similar role compared to the chorus of Greek ancient theater. The chorus is positioned in the "orchestra", which is the stage area between the audience and the center of stage. The chorus of the ancient theater is the "translator" from the imagined world on stage to the world of the audience.

Our multimedia system has a lot of design implications for future work. We note that the primary novelty of this project is not the individual components as we use off-the-shelf vision, NLP, and mobile applications to build the system. However, the culmination of the components, implemented in the system, realizes a full interactive experience for users, with sensory input and feedback in terms of motion and lighting. The design of the chatbot and its conversational strategies forms an intermediate between convergent/task-based and divergent/open-ended designs, which can help inform the next generation of chatbots. The resulting system design has been achieved on low-cost edge computing hardware (NVIDIA Jetson Nanos) with minimal latency, and the actual physical mechanisms of the LED stage with trusses is portable and mobile, able to be installed in multiple venues with little effort. We believe this system can serve as an inspiration for artistic installations that leverage state-of-the-art AI and physical interaction into the system.

Future work is needed to fully evaluate the efficacy of ODO as an experiential media system, and how it engages users who participate in the production. Putting the audience into an active role is a difficult task, because the audience expects an experience, watching a performance from distant. To offer the reader some sense of the feedback given by the audience, we did informally ask their opinions after

interacting with the system. The audience gave a mixed feedback on their experience, mentioning that doing active tasks during the performance also makes it difficult to enjoy the "linear" parts of the story. The experience, playing an active role was new to them. Some of them responded to have enjoyed a new role in this performance. They saw it as a new and different experience compared to a traditional theater watching experience. Both groups feel better prepared experiencing other interactive productions. This is good feedback as we go through the iterative design process for this theatrical performance in the future, and work to evaluate the system from a HCI perspective. The theater installation "No Body lives here (ODO)" premiered July 2020 in Munich is also presented in "Artificial Empathy" exhibition at Center for Art and Media ZKM Karlsruhe, Germany in Fall 2020.

Chapter 4

MARTINY

In this chapter, we propose an in-house built, low-cost and low-powered smart weather station called MaRTiny. This $200 system is built with an intention to replace the existing $20,000 MaRTy system along with cloud and vision capabilities by leveraging different edge devices. Different machine learning and deep learning models are implemented on these devices using low-level API -TensorRT to enhance performance.

4.1 BACKGROUND

4.1.1 Motivation

The year 2020 marks the Earth's warmest 10-year period with an average increase in global temperature of 1.3°C above pre-industrial levels. Extreme heat puts tremendous stress on individuals' health and well-being and limits their ability to work, travel, and socialize in outdoor settings. Future trends of urban warming indicate the need for adaption measures to promote resilience in the population.

The outdoor urban environment is a complex arrangement of urban forms and materials that impact how heat is experienced by pedestrians at the microscale. In hot, dry cities pedestrian comfort is strongly dictated by the availability of shade [69, 72]. Pedestrian may respond to microscale outdoor conditions by changing their walking path from sun to shade or vice versa based on their thermal comfort.

The most common way to report urban heat is air temperature, which has been shown to be insufficient to quantify personal heat exposure [36, 53]. A more human-centric metric that emphasizes the heat load on the human body is the Mean Radiant

Temperature (MRT). MRT objectively quantifies the total short- and longwave radiation the human body is exposed to at a given location and time [46]. This includes longwave radiation emitted from hot surfaces, such as asphalt parking lots, and shortwave radiation from the sun. MRT roughly equals air temperature in the shade but can be 30°C higher in the sun, making a person feel much less comfortable when it is hot [68]. In desert cities such as Phoenix, Arizona, USA, MRT is the type of temperature that best describes how people experience heat.

MRT has been successfully used in urban climate and human biometeorology research to predict heat-related mortality and outperformed air temperature as predictor [99]. Using computer simulations, MRT was estimated to assess the impact of tree planting strategies on human thermal exposure under climate change in Vancouver, Canada [3] and to perform thermal comfort routing in Tempe, Arizona, USA [71]. Observational studies have quantified the benefit of shade for thermal comfort of different shade types including trees, engineered structures, and urban form [68]. Accurate, high resolution MRT measurements require expensive equipment, such as the bio-meteorological instrument platform MaRTy [70], but lower-cost alternatives such as the grey 38 mm globe thermometers have been developed [98].

Active shade management in cities is important, especially in the Southwestern US, to provide shade where people work, travel, and socialize outdoors, because cooling benefits are hyperlocal. Yet, little information exists on how people use public spaces and when and where they are exposed to outdoor heat. We close this gap by developing a novel low-cost, portable, smart IOT weather station (MaRTiny) that can count people in the shade and sun. Connecting hyperlocal meteorological conditions with space use data captured by a camera reveals behavioral patterns of shade and sun preferences that vary by time of day, location, and ambient conditions.

4.2 RELATED WORK

4.2.1 MRT Sensing and Modeling

MRT Sensing: There are two ways to calculating MRT value, by building hardware setup for measurements or by using mathematical models for estimation. MRT can most accurately be sensed through integral radiation measurements using the so-called 6-directional method [41]. This setup aligns three net radiometers orthogonally to each other to measure the longwave and shortwave radiation in 6 directions which is summarized into a temperature value using the Stefan-Boltzmann Law:

$$MRT = \sqrt[4]{\frac{\sum_{i=1}^{6} W_i(a_k K_i + a_l L_i)}{a_l \sigma}} - 273.15 \tag{4.1}$$

where K_i and L_i are the directional shortwave and longwave radiation fluxes, respectively; a_k and a_l are absorption coefficients for short- and long wave radiation fluxes, respectively; σ is the Stefan-Boltzmann constant; and W_i are factors that weigh the directional fluxes to match the cylindrical shape of the human standing body (0.06 is used for sensors pointing up and down, 0.22 for lateral sensors). The method is expensive with three net radiometers that cost $5k each.

A more affordable but less accurate method to estimate MRT is using a black globe thermometer. Globe thermometers such as the Kestrel Heat Stress meter ($500) have been widely used in outdoor thermal comfort studies to quantify the heat load of pedestrians [44]. Thorsson et al. [98] developed a low-cost globe thermometer using a thermocouple in a grey ping pong ball (<$100). The grey color of the globe almost matches the albedo of the human skin and is an easy and reliable method to estimate MRT. To convert grey globe temperature T_g to MRT, Vanos et al. [104] developed an empirical correction equation based on air temperature T_a, wind speed V_a, globe thermometer diameter $D = 38mm$, and emissivity $\epsilon = 0.97$ of the globe.

MRT Modeling: Due to limited sensing resources, MRT measurements across space and time are usually sparse. To address this gap, microclimate and radiation models have been developed that can calculate MRT using information on the built form and meteorological data. For example, RayMan [64] is a point-based, single location model that requires hemispherical fisheye photos as input and calculates MRT based on the horizon limitation and standard weather information. ENVI-met [9] is a 3D gridded computational fluid dynamics (CFD) model that has been widely used in urban climates studies to assess heat at the neighborhood level. ENVI-met and RayMan calculate MRT based on the position of the sun to calculate the direct solar radiation and other radiative fluxes. However, Crank et al. [21] found that both models do not perform well in extreme heat cases and struggle with complex urban forms. Currently, no model can accurately estimate MRT in the absence of detailed urban form parameters.

4.2.2 Pedestrian Counting

Lot of works have been purposed for pedestrian counting and crowd estimation using different techniques. Sensor-based techniques [56, 86, 109, 118] use passive infrared (PIR) and proximity sensors to monitor pedestrians in motion. Although these setups are compact and low-cost, they have a low accuracy and are prone to miss-classification. These setups are not versatile and work best only under certain environmental conditions. There also network-based techniques [112, 26, 50] that use Bluetooth and WiFi networks for crowd sensing.

More recently, there have been works involving machine learning techniques where low-level image features are extracted using different techniques [13, 14], such as Haar cascade [107] and HOG (Histogram of Oriented Gradient) [116, 23] combined with

a regression model like SVM (Support Vector Machine) [116] or with a detector like AdaBoost [108]. With the introduction of different variants of Convolution Neural Network, feature extraction became simpler and thus several deep learning models were built for crowd estimation using individual detection [110, 115, 8, 96, 61] and using perspective maps [119, 12, 58].

Further, there are works revolving around analysis and study on crowd behaviour in urban areas [57, 37, 40] and their relation with thermal comfort [28, 4, 33, 27]. Our main motto is not to challenge any of the existing pedestrian counting technique, but to combine it with a weather station as a single setup.

In summary, thermal exposure measurements in tandem with public space use assessments are crucial for active shade management in cities, but accurate MRT measurement setups are expensive and bulky. Low-cost systems such as grey globe thermometers have been developed but are not connected to the cloud for easy data storage and analysis. None of the existing MRT sensing platforms have vision capabilities, and space use is often assessed through time-consuming manual observations. Finally, physics-based MRT models require detailed 3D data of the urban environment to model radiation flux densities and sun-exposure. Our MaRTiny system and machine learning approach address these gaps.

4.3 SYSTEM OVERVIEW

4.3.1 MaRTiny Weather Station

MaRTiny is configured with four types of sensors to collect meteorological data every minute - a temperature probe/thermometer, UV sensor, humidity sensor, and anemometer (wind speed). Two temperature probes are utilized for globe and air temperature respectively. Globe temperature is measured using a gray ping-pong ball

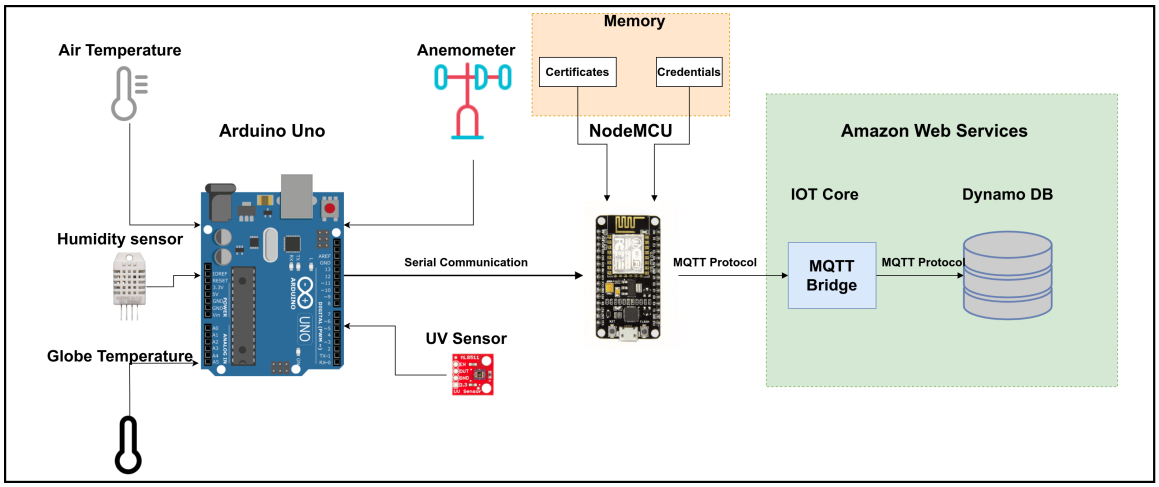


Figure 4.1: Block Diagram of Martiny Setup with Communication Protocols

attached on top of its probe. The grey color of the globe almost matches the albedo of the human skin. This emulates omnidirectional thermal exposure for a human body as a function of radiation, air temperature, and velocity [98]. Air temperature is measured using a downward hanging white cup that shades the attached temperature probe. The white cup reflects most of the solar radiation instead of absorbing it to provide an air temperature "free" from the influence of solar radiations. For a full list of meteorological parameters description and units, please see Table 4.2 and for the full list of sensors used, please refer to Table 4.1. MaRTiny is powered by a DC adapter of 5V/4A, which is shared by both systems (weather station and vision system). The anemometer is supplied with 9V power by stepping up the primary voltage source. This setup can be easily scaled with more sensors without compromising on space and power.

In practice, low-cost sensors are subject to noise and variation, which can yield errors in MRT estimation as we show later in Section 4.5. To solve this problem, we introduce a machine learning model to robustly estimate MRT despite these inaccuracies.

Meteorological Parameters		
Parameter	Description	Unit
Air Temperature	Temperature of the surrounding air	°C
Globe Temperature	Temperature experienced in the gray globe	°C
UV Intensity	Medium and long wave UV radiation	mW/cm^2
Humidity	Amount of humidity in the air	g/kg
Wind-speed	Speed of the wind	m/s

Table 4.1: List of Meteorological Parameters Measured by MaRTiny

Part List		
Sensor	Part no.	Cost ($)
Temperature Probe	DS18B20	9
Humidity Sensor	DHT22	5
UV Sensor	ML8511	5
Anemometer	Adafruit	35
Arduino	Uno	20
Node MCU	ESP8266	7
CSI Camera	MIPI	20
Nvidia Jetson	Jetson Nano	108

Table 4.2: List of Electrical Parts and Its Cost Used in MaRTiny

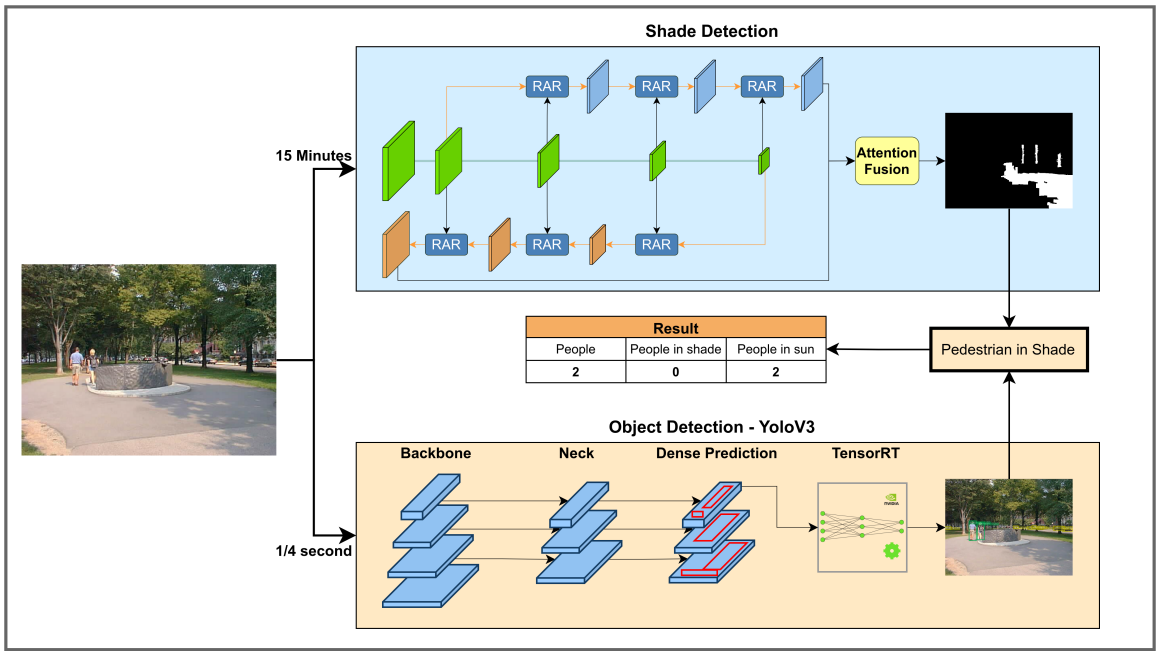


Figure 4.2: System Overview of the MaRTiny Vision Using Different Deep Learning Models

4.3.2 MaRTiny Vision System

Along with meteorological parameters, MaRTiny requires vision capabilities including object detection and identification as well as shade detection in outdoor areas. We leverage the NVIDIA Jetson Nano, which is a low-cost and low-powered edge device capable of running state-of-the-art deep learning models. The Jetson Nano features an ARM-based micro-processor built with a Nvidia V100 GPU that is programmed through Nvidia's low level API TensorRT engine. To capture video, we utilize a compact MIPI (Mobile Industry Processor Interface) camera and stream the data to the Jetson Nano using a gstreamer pipeline. Vision data is sent to AWS via an external USB WiFi on-board. In the next section, we describe in detail our deep learning networks to detect people as well as shade regions in a scene in a privacy-preserving manner.

4.3.3 Data Logging and Communication

To read the meteorological sensor data, we use Arduino Uno, a cost effective and versatile micro-controller. The Uno board communicates with a NodeMCU micro-controller featuring an ESP8266 architecture that has an inbuilt WiFi module, flash memory, and supports the PEM (Privacy Enhanced Mail) file system. Sensor data are continuously read in a loop by the Uno board with a small delay of 1ms to avoid overheating. Records are collected in a buffer, and an average is calculated for every minute, which is then transmitted to the NodeMCU board. On average, the Uno acquires around 80 readings per minute. Both boards make use of the serial communication protocol UART (Universal Asynchronous Receiver/Transmitter) to communicate with each other.

The NodeMCU communicates securely with the online database. For our application we utilize AWS DynamoDB, which is a NoSQL database. Unlike traditional relational database systems, NoSQL can handle unstructured data, making it very flexible. All the necessary security PEM files are stored in the flash memory of NodeMCU, which is required during authentication of MaRTiny. Using these files, NodeMCU establishes a communication path with AWS through the MQTT protocol. MQTT is an extremely lightweight publish/subscribe messaging protocol designed for IoT communication. Once the communication is established, Node MCU waits for bytes of data to be received from the Uno board. Sensor data collected by Uno is sent to NodeMCU via serial communication every minute, which is then transmitted to DynamoDB using the MQTT protocol.

4.4 IMPLEMENTATION

To fully leverage the MaRTiny system, we require advanced machine learning capabilities on-board the device to fully interpret and analyze the meteorological sensor data and images captured. In this section, we first describe our approach for robustly estimating accurate MRT from the sensor data via supervised learning with a SVM (Support Vector Machine) and ANN (Artificial Neural Network) models. Next, we describe deep learning models for people detection and shade identification in images, which allows to count people in shade. To preserve privacy, images taken by our vision are purely used for detection purposes and are not stored on the device memory or in the cloud.

4.4.1 Machine Learning Model for Accurate MRT Estimation

As MaRTiny is a low-cost, compact alternative to the MaRTy sensing platform [70, 73, 68] the replacement of highly accurate sensors has drawbacks including less sensitivity and sensor lag [35]. We noticed these inaccuracies caused serious errors in the calculated MRT values (Figure 4.7). In particular, MRT was sensitive to the positioning and orientation of the MaRTiny relative to MaRTy (e.g the MaRTiny was shaded, which resulted in lower MRT values, while MaRTy's net radiometers were partially sun-exposed).

To overcome this limitation, we formulate MRT estimation as a supervised learning problem. This requires labeled ground-truth MRT values to be provided in correspondence with our less robust meteorological sensor data. In Section 4.5, we discuss dataset collection consisting of paired MaRTy and MaRTiny measurements to create this labeled data. This allows us to train a machine learning model to estimate MRT accurately from MaRTiny sensor data.

Figure 4.3: MaRTy and MaRTiny experimental setup

We explored both, a traditional machine learning method using support vector machine (SVM) model, with three different kernels as well as deep learning method by using an artificial neural network (ANN) model to perform this supervised learning task. We outline the results of this estimation in Section 4.5. In particular, we observed an SVM with RBF (Radial Basis Function) kernel achieved the highest accuracy on our evaluation dataset. This method has the added advantage of being computationally lightweight, so it can be easily deployed on the Jetson Nano with minimal resource requirements for training and inference purposes.

4.4.2 People and Shade Detection

We leverage state-of-the-art deep learning models to detect people in the shade and sun. Important model selection criteria: the models need to fit onto the NVIDIA Jetson Nano and they must be able to process frames coming from the MIPI camera with a frame rate of at least 1fps.

Shadow Detection: To perform shadow detection in an image, we use the deep learning model Bi-directional Feature Pyramid with Recurrent Attention Residual Modules (BDRAR) [122], visualized in the upper branch of Figure 4.2. This BDRAR network takes a single image as input and outputs a binary shadow map as output in an end-to-end manner. First, it leverages a convolutional neural network (CNN) to extract feature maps at different spatial resolutions. It then employs two series of recurrent attention residual modules to fully exploit global and local context for these feature maps. The features captured by shallow layers exploit shadow details in the local regions and the features captured by deep layers understands the overall shadow regions of the image. To implement this network in practice on the Jetson Nano, we updated the code base to the Jetpack-v4.4 Linux system running on our embedded device including rewriting the code in Python 3.6.

Figure 4.4: Shadow Masks Captured by the Martiny Vision System

Object Detection: For object detection, we utilise the state-of-the-art Yolov3 network [2]. The model is trained on 80 different classes of the Microsoft COCO dataset with high object detection performance. The Yolov3 algorithm can be built using two different frameworks - DarkNet and MobileNet [87]. The MobileNet framework is computationally light but has low accuracy, hence we decided to use the Darknet framework.

Since the YOLOv3-darknet model is large and computationally expensive to run on the NVIDIA Jetson Nano, we converted it into a simple neural graph using Nvidia's TensorRT. This allowed the model to run successfully on the Nano with a frame-rate of 4fps, which is sufficient for our application.

Figure 4.5: Pedestrian Counter for Sun Exposed and in Shade Regions

Pedestrian in Shade Detection

Images represent the 3D environment in 2D, limiting our ability to determine the exact location of a person on the ground and distance from the camera. We introduce a simple approach to identify pedestrians in shade without determining the position in 3D space. First, a binary shadow map from BDRAR indicating the presence of shade per pixel is computed periodically (in our case, every 15 minutes as shade does not vary significantly in such a short time). For every frame from the 4fps MIPI camera, yolov3 outputs objects with their bounding boxes consisting of pixel coordinates for the corners. Our algorithm calculates the IOU (Intersection over Union) of the bounding box with the shade map. We consider a person to be in shade if 40% of the bounding box region is inside the shade map (i.e. $IOU = 0.4$).

Calculating IOU without considering the position of the person with respect to shade can lead to errors. For example, in the Figure 4.6, one person is sun-exposed and the other person is in the shade. IOU of the bounding box with the shadow map equals 60% in the first case and 40% in the second case. IOU for the first case is high due to shade in the background and shadow cast by the person's body. This is

the most common type of error that occurs at different orientations and positions of a person; therefore, it is necessary to distinguish between shade from a person and shade from the surroundings. The algorithm first checks if the edge of the bounding box is in the shade. Aa person does not have to be completely in shade to feel the cooling effect, hence we consider only the bottom half i.e. 50% of the bounding box as ROI (Region of Interest). We then calculate IOU between this region and the shadow map. For our application an IOU of 80% (which implies an IOU of 40% in the overall picture) is considered the optimum value for a person to experience the cooling effects of shade. The ROI and IOU is subjected to change based on the environment and application, but the core logic will remain same.

People count data for sun and shade along with other space use relevant counts (umbrellas, pets, and bicycles) are reported to the online database, and the frame with identifying features is discarded. This allows our device to preserve the privacy of the individuals being observed, which is necessary for public deployment of the sensing platform.

4.4.3 Training

As MaRTiny is a low-cost, compact alternative to the MaRTy sensing platform [70, 73, 68] the replacement of highly accurate sensors has drawbacks including less sensitivity and sensor lag [35]. We noticed these inaccuracies caused serious errors in the calculated MRT values (Figure 4.7). In particular, MRT was sensitive to the positioning and orientation of the MaRTiny relative to MaRTy (e.g the MaRTiny was shaded, which resulted in lower MRT values, while MaRTy's net radiometers were partially sun-exposed).

To overcome this limitation, we formulate MRT estimation as a supervised learning problem. This requires labeled ground-truth MRT values to be provided in corre-

spondence with our less robust meteorological sensor data. In Section 4.5, we discuss dataset collection consisting of paired MaRTy and MaRTiny measurements to create this labeled data. This allows us to train a machine learning model to estimate MRT accurately from MaRTiny sensor data.

We explored both, a traditional machine learning method using SVM model with three different kernels as well as Deep learning method by using an ANN model to perform this supervised learning task. We outline the results of this estimation in Section 4.5. In particular, we observed an SVM with RBF (Radial Basis Function) kernel achieved the highest accuracy on our evaluation dataset. This method has the added advantage of being computationally lightweight, so it can be easily deployed on the Jetson Nano with minimal resource requirements for training and inference purposes.

4.5 RESULTS

4.5.1 Data Collection

To evaluate the MaRTiny system, we collected a custom dataset of ground truth MRT values for two sun-exposed outdoor locations for three days in Tempe, Arizona, USA. For validation purposes, the MaRTy human-biometeorological platform [70] was paired with the MaRTiny system for simultaneous data logging. Figure 4.3 illustrates the paired setup and the difference in scale between MaRTy and MaRTiny. In addition, an image dataset was collected to evaluate the performance of the object and shade detection. Using a pipeline, the images from the MIPI camera were stored at random intervals along with the bounding boxes of present objects. Ground truth bounding boxes were drawn manually using tools such as [2, 101] for 30 images consisting of around 50 different objects. We calculate Precision and Recall for each

object, which is further used to calculate MAP. The same images were used to evaluate shade detection using IOU (Intersection Over Union) metrics. Small snippets of videos were stored at random intervals which helped to cross verify number of people in a given time frame. All the images and videos were stored in AWS S3 bucket and later deleted to preserve privacy.

4.5.2 MRT Estimation

We first evaluated the performance of MaRTiny in estimating MRT values. We utilize Equation (4.1) with the sensor data on-board the system to calculate MRT. MaRTy logs data every 2 seconds, and MaRTiny stores data every minute, hence we calculated 1-min averages for comparison. Ground truth MRT was calculated using Equation (4.1). Figure 4.7 shows MaRTiny MRT results in green and MaRTY's ground truth calculation in red. A significant error in MaRTiny's estimation of MRT was found in the morning with an MSE of 10°. The error is due to the offset of the two devices, which caused the MaRTiny sensor to be partially shaded by a nearby palm tree in the morning while MaRTy's net radiometers were sun-exposed.

To improve MRT estimations, we utilize our supervised learning approach using both support vector machines and artificial neural networks. Machine learning models were trained on selected meteorological parameters - air temperature, globe temperature, humidity and UV intensity, which were comparatively more accurate and less prone to noise. We used around 12,000 data points for training and 3000 for testing from a range of dates, times, and locations in the sensing period. Cross-validation was used to tune hyperparameters of the models. A separate dataset for evaluation consisted of around 700 data points from a single location collected in a day as is the usual application for this algorithm.

Since there is a non-linear relation of globe temperature and air temperature with MRT given in (4.1), we anticipated the need of machine learning models which could understand complex non-linear relation between these parameters. SVM with RBF kernel and a neural network with Relu activation function are example of such models. In Table 4.3, we present a comparison of SVMs with three different kernels and a traditional artificial neural network. We report root mean square error (RMSE) for both the testing and evaluation datasets. Note that the results of linear and polynomial SVM kernels justify our earlier assumption and the results of SVM with RBF kernel as well as the ANN achieved the best performance in quantitative metrics. We visualize SVM-RBF's MRT curve in Figure 4.7.

We trained our ANN on a i7 CPU. We set our learning rate α to 0.001, which took around 5 minutes and 300 epochs to converge. Although this model performs slightly better on the test dataset thant the SVM-RBF model, performance is identical on the evaluation dataset. The SVM model is lighter and can be easily trained and deployed on edge devices such as Jetson Nano.

4.5.3 Shade and Object Detection

For the object and shade detection algorithms, we used pre-trained models that were evaluated by their respective studies [2, 122]. However, for our example application of detecting people in shade, we leverage our small custom image dataset that we curated to estimate this information.

The standard evaluation metric used for any object detection is mAP (Mean Average Precision). Bounding boxes were manually drawn using the tool for the dataset consisting of 30 images and IOU was calculated with the bounding boxes predicted by our model. Precision and recall is calculated for a series of different IOU thresholds ranging from 0.5 to 0.95. A precision-recall graph is constructed and the

area under this graph provides us the mAP value of around 55%. All These steps were carried out programmatically in python language using frameworks like PyTorch and numpy. For our application IOU threshold of 0.5 gives us the optimal results. We also achieved an Average Precision of more than 85% for each of the following classes - Pedestrian, Bike, Pets and Umbrella.

The evaluation of the shadow detection is done on a per pixel basis, which is a binary evaluation method. A dataset consisting of 30 shade images was collected from different locations and time. We manually annotated these images using the tool [101] and calculated IOU of the shadow map with the ground truth and found a precision of 90%. This is not the most effective method to calculate the accuracy of the model due to the irregular shapes, human error in annotation and small dataset and hence we back it up with the evaluation metrics of the original paper [122].

Machine Learning Algorithms for MRT		
Algorithm	RMS-Test	RMS-Eval
SVM-Linear	16.8	20.2
SVM-Poly	12.01	10.02
SVM-RBF	4.61	3.95
ANN	3.28	3.82

Table 4.3: Performance of Machine Learning

4.6 DISCUSSION

In our work, we combine meteorological sensing with computer vision techniques to understand the relationship between weather and public space usage. MaRTiny leverages edge devices which are low-cost, low-powered, and yet computationally ca-

pable of running state-of-the-art Machine Learning algorithms. We compared our results with different setups, metrics, and methodologies 4.5 and conclude that the MaRTiny system can replace $20k worth MaRTy system in addition to the IOT and vision capabilities, but considering few of its limitations.

The MaRTiny Weather Station is built using off-the-shelf sensors, making them highly sensitive to environmental and circuit noise, effecting the accuracy of MRT calculation. During the experiments we observed that the position and orientation of MaRTiny makes a significance difference for MRT calculation. The shading effect caused by surrounding trees and buildings is one such example. We also observed errors in MRT calculation during cloudy and hazy days, as globe temperature sensor and UV sensor were affected by the clouds. This limitation is true even for our trained machine learning model as it is not trained on different climatic conditions. The conversion of the codebase of BDRAR network to python3.6 (explained in the Section 4.4), causes small inaccuracies in the shadow map estimation. Further, there are few scenarios where the pedestrian is partially under shade but our algorithm fails to recognise these.

In the future, we will build several MaRTinies and deploy them in public parks and playgrounds in collaboration with the City of Tempe. Data will provide important insight into how people use public spaces under varying meteorological conditions, especially in the summer when temperatures can reach 115°F in the Phoenix metropolitan area. We also plan to train our ML model to include a variety of weather conditions, such as cloudy and windy days, to make it more accurate, robust, and scalable to other locations and geographies.

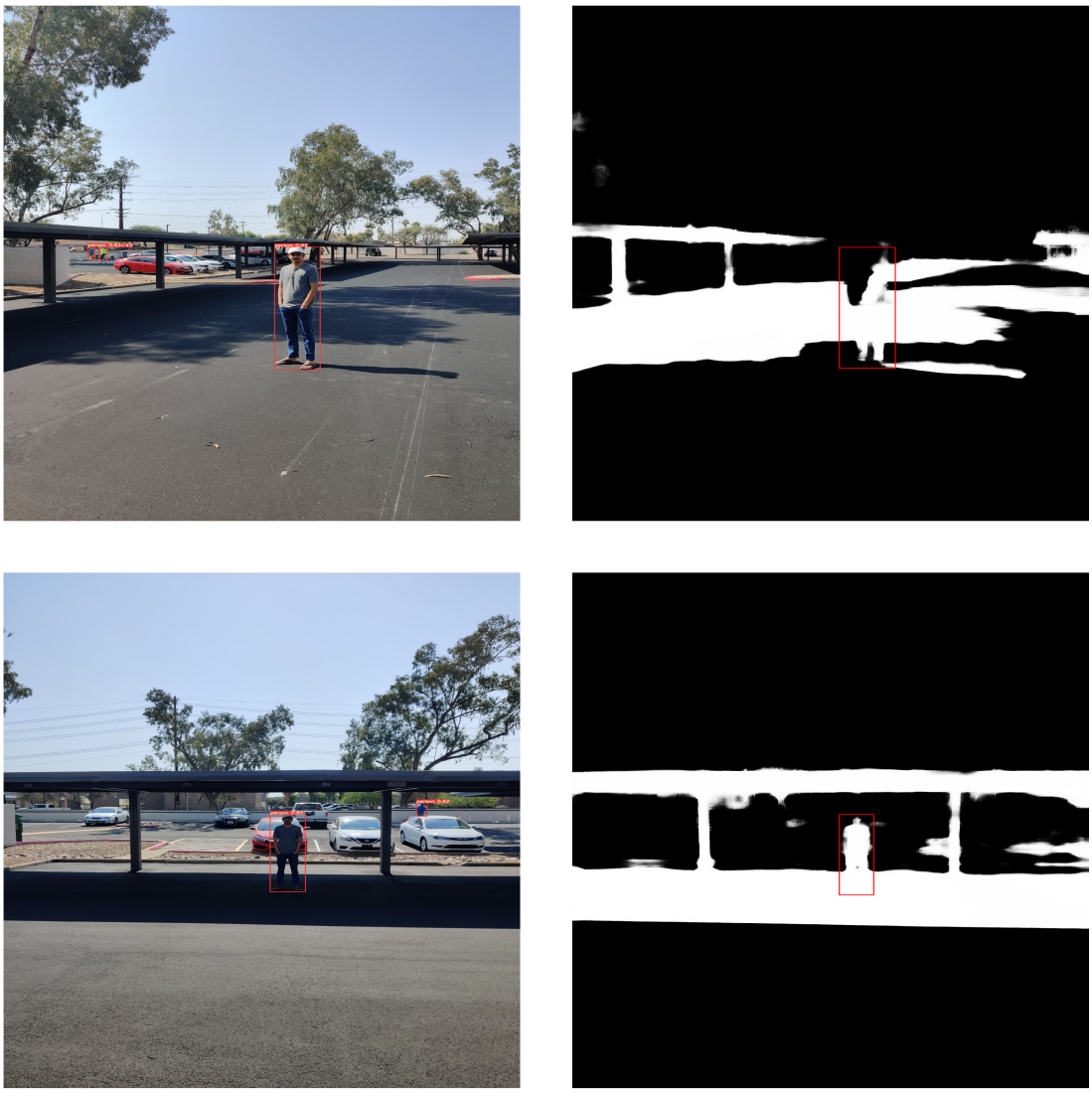


Figure 4.6: Images of a Person with Bounding Box and Shadow Map of the Surrounding

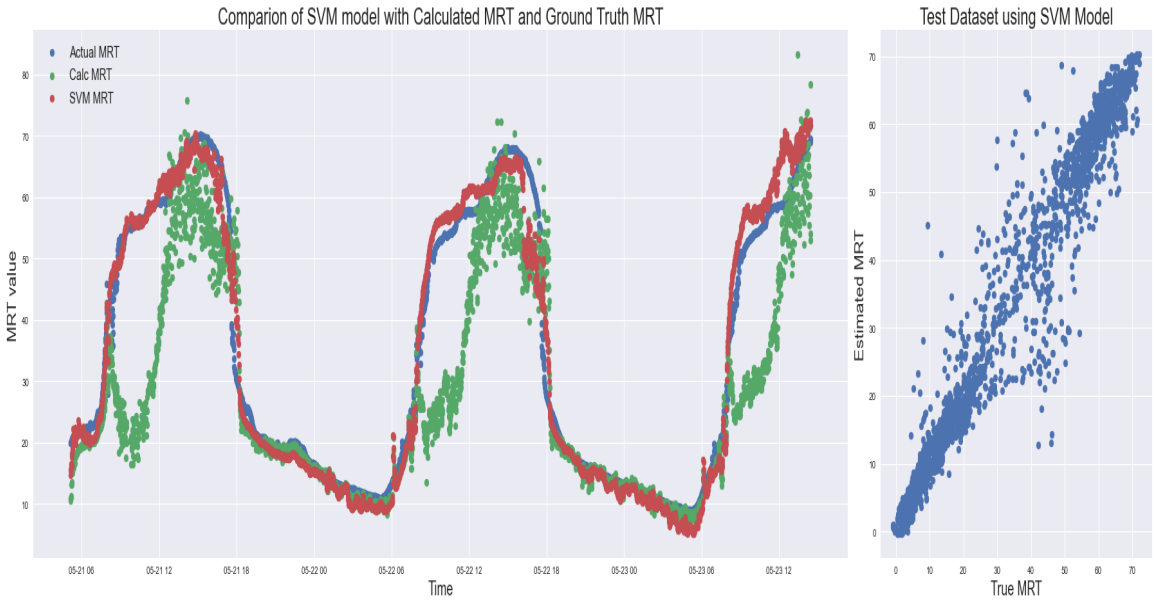


Figure 4.7: Graph Comparison of ML model with ground truth

Figure 4.8: Sample Images of Shade Detection and Pedestrian Counting

Chapter 5

CONCLUSION

In this thesis I presented machine learning and computer vision techniques applied on edge devices in two different fields. I first discussed how different edge devices are used to achieve better understanding of people in Chapter 2 and better understanding of an environment in Chapter 3. Techniques like edge detection and key-feature extraction are crucial for every image analysis. Applications of these techniques are discussed in Chapter 2 for understanding and classifying emotions and in Chapter 3 to detect pedestrians and regions of shade.

I also discussed how different domains are combined with deep learning methods to achieve interdisciplinary goals. In our ODO project we combined sequence translation methods, used to interpret language with vision techniques and clustering algorithms to build a sophisticated AI system capable of interacting with users. In our MaRTiny project, we combined an IoT weather station, which calculates MRT value with pedestrian detection algorithm to provide a full-fledged understating of the public space usage.

ODO project paved a new path in the field of arts, media and theater. ODO challenges the conventional method of theatrical plays and encourages audience to be a part of the play and also gives freedom to construct their own story. This portable system which is packaged with flawless story narration along with user interactions and supported by movable LED system was made possible using edge devices and their abilities to run machine learning algorithms. Although the story is customized according to the user, they are still bound within the boundaries set by the narrator. As a part of future work, ODO is to be developed with general purpose intelligence,

capable of striking general conversations with the users without any boundaries.

While ODO is built to create better experience of the theater, MaRTiny on the other hand is built with an intent to solve urban city meteorological problems. MaRTiny has the potential to replace $20,000 system, saving around 99% of the expense. The IoT system along with vision capabilities makes it one of its-kind device for urban city research. These systems when installed across a city, can provide great insight into the weather conditions at different parts of the city, people's outdoor behaviour at different climatic conditions and public space usage. These parameters play very crucial role for town planning, architecture and civil engineering.

We have reached a stage in technological growth, where every field can make use of machine learning algorithms to improvise their processes and solve complex problems. The two projects discussed through out the thesis are examples of this. From arts, media and theater to urban climate research, machine learning is used to either solve an existing problem or to build something new. The low-cost, low-powered and portable edge devices makes this possible by facilitating the platform to run ML algorithms efficiently.

REFERENCES

[1] Abu Sulayman, I. I., S. H. Almalki, M. S. Soliman and M. O. Dwairi, "Designing and implementation of home automation system based on remote sensing technique with arduino uno microcontroller", in "2017 9th IEEE-GCC Conference and Exhibition (GCCCE)", pp. 1–9 (2017).

[2] AlexyAB, "YOLO_mark", `https://github.com/AlexeyAB/Yolo_mark` (2016).

[3] Aminipouri, M., D. Rayner, F. Lindberg, S. Thorsson, A. J. Knudby, K. Zickfeld, A. Middel and E. S. Krayenhoff, "Urban tree planting to maintain outdoor thermal comfort under climate change: The case of Vancouver's local climate zones", Building and Environment **158**, 226–236 (2019).

[4] Arens, E. and P. Bosselmann, "Wind, sun and temperature—predicting the thermal comforf of people in outdoor spaces", Building and Environment **24**, 4, 315–320, URL `https://www.sciencedirect.com/science/article/pii/0360132389900255` (1989).

[5] Arriaga, O., M. Valdenegro-Toro and P. Plöger, "Real-time convolutional neural networks for emotion and gender classification", arXiv preprint arXiv:1710.07557 (2017).

[6] Ayman, O., "Emotion-recognition", `https://github.com/omar178/Emotion-recognition` (2019).

[7] Brandtzaeg, P. B. and A. Følstad, "Why people use chatbots", in "International Conference on Internet Science", pp. 377–392 (Springer, 2017).

[8] Brostow, G. J. and R. Cipolla, "Unsupervised bayesian detection of independent motion in crowds", CVPR '06 (IEEE Computer Society, USA, 2006), URL `https://doi.org/10.1109/CVPR.2006.320`.

[9] Bruse, M. and H. Fleer, "Simulating surface-plant-air interactions inside urban environments with a three dimensional numerical model", Environmental Modelling and Software **13**, 3-4, 373–384 (1998).

[10] Catania, F., M. Spitale, D. Fisicaro and F. Garzotto, "Cork: A conversational agent framework exploiting both rational and emotional intelligence.", (2019).

[11] Cer, D., Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder", arXiv preprint arXiv:1803.11175 (2018).

[12] Chan, A., J. Liang and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking", (2008).

[13] Chen, K., S. Gong, T. Xiang and C. C. Loy, "Cumulative attribute space for age and crowd density estimation", (2013).

[14] Chen, K., C. C. Loy, S. Gong and T. Xiang, "Feature mining for localised crowd counting", in "Proceedings of the British Machine Vision Conference", pp. 21.1–21.11 (BMVA Press, 2012).

[15] Chen, L., J. Zhang and Y. Wang, "Wireless car control system based on arduino uno r3", in "2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)", pp. 1783–1787 (2018).

[16] Chen, Y., H. Sundaram, T. Rikakis, T. Ingalls, L. Olson and J. He, "Experiential media systems–the biofeedback project", in "Multimedia Content Analysis", pp. 1–34 (Springer, 2009).

[17] Chollet, F., "Xception: Deep learning with depthwise separable convolutions", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1251–1258 (2017).

[18] Christian Ziegler, T. S., Sayaka Kaiwa and H. Paquete, *COSMOS*, URL `http://www.chrisziegler.de/?type=performing&txt_id=145&lng=eng` (????).

[19] Christian Ziegler, T. S., Sayaka Kaiwa and H. Paquete, *COSMOS*, URL `https://vimeo.com/271233798` (????).

[20] Ciechanowski, L., A. Przegalinska, M. Magnuski and P. Gloor, "In the shades of the uncanny valley: An experimental study of human–chatbot interaction", Future Generation Computer Systems **92**, 539–548 (2019).

[21] Crank, P. J., A. Middel, M. Wagner, D. Hoots, M. Smith and A. Brazel, "Validation of seasonal mean radiant temperature simulations in hot arid urban climates", Science of the Total Environment **749** (2020).

[22] Cycling74, *Max/MSP*, URL `https://cycling74.com/` (1997 (accessed May 22, 2020)).

[23] Dalal, N. and B. Triggs, "Histograms of oriented gradients for human detection", in "2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)", vol. 1, pp. 886–893 vol. 1 (2005).

[24] Dale, R., "The return of the chatbots", Natural Language Engineering **22**, 5, 811–817 (2016).

[25] Davis, M., "Theoretical foundations for experiential systems design", in "Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence", pp. 45–52 (2003).

[26] Depatla, S., A. Muralidharan and Y. Mostofi, "Occupancy estimation using only wifi power measurements", IEEE Journal on Selected Areas in Communications **33**, 1–1 (2015).

[27] Eliasson, I., I. Knez, U. Westerberg, S. Thorsson and F. Lindberg, "Climate and behaviour in a nordic city", Landscape and Urban Planning **82**, 1, 72–84, URL https://www.sciencedirect.com/science/article/pii/S0169204607000473 (2007).

[28] Eom, S. and Y. Nishihori, "How weather and special events affect pedestrian activities: volume, space, and time", International Journal of Sustainable Transportation **0**, 0, 1–31, URL https://doi.org/10.1080/15568318.2021.1897907 (2021).

[29] Ester, M., H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.", in "SIGKDD Conference on Knowledge Discovery and Data Mining", vol. 96, pp. 226–231 (1996).

[30] Følstad, A. and P. B. Brandtzæg, "Chatbots and the new world of hci", interactions **24**, 4, 38–42 (2017).

[31] Freed, A., "Open sound control: A new protocol for communicating with sound synthesizers", in "International Computer Music Conference (ICMC)", (1997).

[32] Gao, J., M. Galley and L. Li, "Neural approaches to conversational ai", in "The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval", pp. 1371–1374 (2018).

[33] Givoni, B., M. Noguchi, H. Saaroni, O. Potchter, Y. Yaakov, N. Feller and S. Becker, "Outdoor comfort research issues", Energy and Buildings **35**, 77–86 (2003).

[34] Goodfellow, I. J., D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee *et al.*, "Challenges in representation learning: A report on three machine learning contests", in "International Conference on Neural Information Processing", pp. 117–124 (Springer, 2013).

[35] Häb, K., B. L. Ruddell and A. Middel, "Sensor lag correction for mobile urban microclimate measurements", Urban Climate **14**, 622–635, URL https://linkinghub.elsevier.com/retrieve/pii/S2212095515300274 (2015).

[36] Harlan, S., A. J. Brazel, L. Prashad, W. L. Stefanov and L. Larsen, "Neighborhood microclimates and vulnerability to heat stress", Social Science & Medicine **63**, 11, 2847–2863 (2006).

[37] Hashimoto, Y., Y. Gu, L.-T. Hsu, M. Iryo-Asano and S. Kamijo, "A probabilistic model of pedestrian crossing behavior at signalized intersections for connected vehicles", Transportation Research Part C: Emerging Technologies **71**, 164–181, URL https://www.sciencedirect.com/science/article/pii/S0968090X1630119X (2016).

[38] Hill, J., W. R. Ford and I. G. Farreras, "Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations", Computers in human behavior **49**, 245–250 (2015).

[39] Hoffman, G., R. Kubat and C. Breazeal, "A hybrid control system for puppeteering a live robotic stage actor", in "RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication", pp. 354–359 (IEEE, 2008).

[40] Hoogendoorn, S. and P. Bovy, "Pedestrian route-choice and activity scheduling theory and models", Transportation Research Part B: Methodological **38**, 2, 169–190, URL `https://www.sciencedirect.com/science/article/pii/S0191261503000079` (2004).

[41] Höppe, P., "A new procedure to determine the mean radiant temperature outdoors", Wetter und Leben (1992).

[42] Ioffe, S. and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", arXiv preprint arXiv:1502.03167 (2015).

[43] Jain, R., "Experiential computing", Communications of the ACM **46**, 7, 48–55 (2003).

[44] Johansson, E., S. Thorsson, R. Emmanuel and E. Krüger, "Instruments and methods in outdoor thermal comfort studies – The need for standardization", Urban Climate (2014).

[45] Johnson, G. L., B. J. Peterson, T. Ingalls and S. X. Wei, "Lanterns: An enacted and material approach to ensemble group activity with responsive media", in "Proceedings of the 5th International Conference on Movement and Computing", MOCO '18 (Association for Computing Machinery, New York, NY, USA, 2018), URL `https://doi.org/10.1145/3212721.3212848`.

[46] Kántor, N. and J. Unger, "The most problematic variable in the course of human-biometeorological comfort assessment - The mean radiant temperature", Central European Journal of Geosciences **3**, 1, 90–100 (2011).

[47] Karmokar, C., J. Hasan, S. Arefin Khan and M. I. Ibne Alam, "Arduino uno based smart irrigation system using gsm module, soil moisture sensor, sun tracking system and inverter", in "2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)", pp. 98–101 (2018).

[48] Kaur, A., S. S. Saini, L. Singh, A. Sharma and E. Sidhu, "Efficient arduino uno driven smart highway/bridge/tunnel lighting system employing rochelle piezoelectric sensor", in "2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)", pp. 1–4 (2016).

[49] Khanam, M., T. Mahboob, W. Imtiaz, H. Ghafoor and R. Sehar, "A survey on unsupervised machine learning algorithms for automation, classification and maintenance", International Journal of Computer Applications **119**, 34–39 (2015).

[50] Kjærgaard, M., M. Wirz, D. Roggen and G. Troster, "Mobile sensing of pedestrian flocks in indoor environments using wifi signals", 2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012 (2012).

[51] Knight, H., "Eight lessons learned about non-verbal interactions through robot theater", in "International Conference on Social Robotics", pp. 42–51 (Springer, 2011).

[52] Knuth, D. E., *Seminumerical Algorithms* (Addison-Wesley, 1981).

[53] Kuras, E. R., M. B. Richardson, M. M. Calkins, K. L. Ebi, J. J. Hess, K. W. Kintziger, M. A. Jagger, A. Middel, A. A. Scott, J. T. Spector, C. K. Uejio, J. K. Vanos, B. F. Zaitchik, J. M. Gohlke and D. M. Hondula, "Opportunities and challenges for personal heat exposure research", Environmental Health Perspectives **125**, 8 (2017).

[54] Kuutti, S., R. Bowden, Y. Jin, P. Barber and S. Fallah, "A survey of deep learning applications to autonomous vehicle control", (2019).

[55] Labs, G., URL `https://www.opendialog.ai/` (????).

[56] Lau, B. P. L., N. Wijerathne, B. K. K. Ng and C. Yuen, "Sensor fusion for public space utilization monitoring in a smart city", IEEE Internet of Things Journal **5**, 2, 473–481 (2018).

[57] Lee, J. M., "Exploring walking behavior in the streets of new york city using hourly pedestrian count data", Sustainability **12**, 19, URL `https://www.mdpi.com/2071-1050/12/19/7863` (2020).

[58] Lempitsky, V. and A. Zisserman, "Learning to count objects in images", in "Advances in Neural Information Processing Systems", edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel and A. Culotta, vol. 23 (Curran Associates, Inc., 2010), URL `https://proceedings.neurips.cc/paper/2010/file/fe73f687e5bc5280214e0486b273a5f9-Paper.pdf`.

[59] Lin, L., X. Liao, H. Jin and P. Li, "Computation offloading toward edge computing", Proceedings of the IEEE **107**, 8, 1584–1607 (2019).

[60] Liu, J.-J., T.-H. Yang, S.-A. Chen and C.-J. Lin, "Parameter selection: Why we should pay more attention to it", (2021).

[61] Liu, W., M. Salzmann and P. Fua, "Context-aware crowd counting", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)", (2019).

[62] Liu, Y., M. Peng, G. Shou, Y. Chen and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5g and internet of things", IEEE Internet of Things Journal **7**, 8, 6722–6747 (2020).

[63] Matthews, S. J. and A. S. Leger, "Energy-efficient analysis of synchrophasor data using the nvidia jetson nano", in "2020 IEEE High Performance Extreme Computing Conference (HPEC)", pp. 1–7 (2020).

[64] Matzarakis, A., F. Rutz and H. Mayer, "Modelling radiation fluxes in simple and complex environments: Basics of the RayMan model", International Journal of Biometeorology **54**, 2, 131–139 (2010).

[65] Mayo, J., *Programming the Microsoft Bot Framework: A Multiplatform Approach to Building Chatbots* (Microsoft Press, 2017).

[66] Mazalek, A., M. Nitsche, C. Rébola, A. Wu, P. Clifton, F. Peer and M. Drake, "Pictures at an exhibition: a physical/digital puppetry performance piece", in "Proceedings of the 8th ACM conference on Creativity and cognition", pp. 441–442 (2011).

[67] Meyer, T. and C. Messom, "Improvisation in theatre rehearsals for synthetic actors", in "International Conference on Entertainment Computing", pp. 172–175 (Springer, 2004).

[68] Middel, A., S. AlKhaled, F. A. Schneider, B. Hagen and P. Coseo, "50 Grades of Shade", Bulletin of the American Meteorological Society pp. 1–35, URL `https://journals.ametsoc.org/view/journals/bams/aop/BAMS-D-20-0193.1/BAMS-D-20-0193.1.xml` (2021).

[69] Middel, A., K. Häb, A. J. Brazel, C. A. Martin and S. Guhathakurta, "Impact of urban form and design on mid-afternoon microclimate in Phoenix Local Climate Zones", Landscape and Urban Planning **122**, 16–28 (2014).

[70] Middel, A. and E. Krayenhoff, "Micrometeorological determinants of pedestrian thermal exposure during record-breaking heat in tempe, arizona: Introducing the marty observational platform", The Science of the total environment **687**, 137–151 (2019).

[71] Middel, A., J. Lukasczyk, R. Maciejewski *et al.*, "Sky view factors from synthetic fisheye photos for thermal comfort routing—a case study in phoenix, arizona", Urban Planning **2**, 1, 19–30 (2017).

[72] Middel, A., N. Selover, B. Hagen and N. Chhetri, "Impact of shade on outdoor thermal comfort—a seasonal field study in Tempe, Arizona", International Journal of Biometeorology **60**, 12, 1849–1861, URL `http://link.springer.com/10.1007/s00484-016-1172-5` (2016).

[73] Middel, A., V. K. Turner, F. A. Schneider, Y. Zhang and M. Stiller, "Solar reflective pavements-A policy panacea to heat mitigation?", Environmental Research Letters **15**, 6 (2020).

[74] Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space", arXiv preprint arXiv:1301.3781 (2013).

[75] Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality", in "Advances in neural information processing systems", pp. 3111–3119 (2013).

[76] Nandanwar, H., A. Chauhan, D. Pahl and H. Meena, "A survey of application of ml and data mining techniques for smart irrigation system", in "2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)", pp. 205–212 (2020).

[77] Ouali, Y., C. Hudelot and M. Tami, "An overview of deep semi-supervised learning", (2020).

[78] Ozbayoglu, A. M., M. U. Gudelek and O. B. Sezer, "Deep learning for financial applications : A survey", (2020).

[79] Pandorabot, *Mitusku*, URL `https://www.pandorabots.com/mitsuku/` (????).

[80] Pawar, P. A., "Heart rate monitoring system using ir base sensor amp; arduino uno", in "2014 Conference on IT in Business, Industry and Government (CSIBIG)", pp. 1–3 (2014).

[81] Pinhanez, C. S. and A. F. Bobick, ""it/i": a theater play featuring an autonomous computer character", Presence: Teleoperators & Virtual Environments **11**, 5, 536–548 (2002).

[82] Purington, A., J. G. Taft, S. Sannon, N. N. Bazarova and S. H. Taylor, "" alexa is my new bff" social roles, user satisfaction, and personification of the amazon echo", in "Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems", pp. 2853–2859 (2017).

[83] Qi, Q. and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing", IEEE Access **7**, 86769–86777 (2019).

[84] Qi, X., C. Liu and S. Schuckers, "Iot edge device based key frame extraction for face in video recognition", in "2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)", pp. 641–644 (2018).

[85] Raghu, M. and E. Schmidt, "A survey of deep learning for scientific discovery", (2020).

[86] Raykov, Y., E. Özer, G. Dasika, A. Boukouvalas and M. Little, "Predicting room occupancy with a single passive infrared (pir) sensor through behavior extraction", pp. 1016–1027 (2016).

[87] Redmon, J., S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection", (2016).

[88] Rehman, S. U., M. R. Razzaq and M. H. Hussian, "Training of ssd(single shot detector) for facial detection using nvidia jetson nano", (2021).

[89] Sabharwal, N. and A. Agrawal, "Introduction to google dialogflow", in "Cognitive Virtual Assistants Using Google Dialogflow", pp. 13–54 (Springer, 2020).

[90] Sabharwal, N., S. Barua, N. Anand and P. Aggarwal, "Bot frameworks", in "Developing Cognitive Bots Using the IBM Watson Engine", pp. 39–46 (Springer, 2020).

[91] Sarikaya, R., "The technology behind personal digital assistants: An overview of the system architecture and key components", IEEE Signal Processing Magazine **34**, 1, 67–81 (2017).

[92] Serban, I. V., A. Sordoni, Y. Bengio, A. Courville and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models", in "Thirtieth AAAI Conference on Artificial Intelligence", (2016).

[93] Shiraz, M., A. Gani, R. H. Khokhar and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing", IEEE Communications Surveys Tutorials **15**, 3, 1294–1313 (2013).

[94] Singh, A., N. Thakur and A. Sharma, "A review of supervised machine learning algorithms", in "2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)", pp. 1310–1315 (2016).

[95] Sordoni, A., M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao and B. Dolan, "A neural network approach to context-sensitive generation of conversational responses", arXiv preprint arXiv:1506.06714 (2015).

[96] Stewart, R., M. Andriluka and A. Y. Ng, "End-to-end people detection in crowded scenes", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2016).

[97] Sundaram, H., "Experiential media systems", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **9**, 1s, 1–4 (2013).

[98] Thorsson, S., T. Honjo, F. Lindberg, I. Eliasson and E.-M. Lim, "Thermal comfort and outdoor activity in japanese urban public places", Environment and Behavior **39**, 660–684 (2006).

[99] Thorsson, S., J. Rocklöv, J. Konarska, F. Lindberg, B. Holmer, B. Dousset and D. Rayner, "Mean radiant temperature–a predictor of heat related mortality", Urban Climate **10**, 332–345 (2014).

[100] Tirumala, A., F. Qin, J. M. Dugan, J. A. Ferguson and K. A. Gibbs, "iperf: Tcp/udp bandwidth measurement tool", (2005).

[101] TOsmonav, "Cvat", `https://github.com/openvinotoolkit/cvat` (2020).

[102] TUG, "Institutional members of the TEX users group", URL `http://wwtug.org/instmem.html` (2017).

[103] Uddin, M. I., M. S. Alamgir, M. M. Rahman, M. S. Bhuiyan and M. A. Moral, "Ai traffic control system based on deepstream and iot using nvidia jetson nano", in "2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)", pp. 115–119 (2021).

[104] Vanos, J., K. Rykaczewski, A. Middel, D. Vecellio, R. Brown and T. Gillespie, "Improved methods for estimating mean radiant temperature in hot and sunny outdoor settings", International Journal of Biometeorology **65** (2021).

[105] Vinyals, O. and Q. Le, "A neural conversational model", arXiv preprint arXiv:1506.05869 (2015).

[106] Viola, P. and M. Jones, "Rapid object detection using a boosted cascade of simple features", in "Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001", vol. 1, pp. I–I (2001).

[107] Viola, P. and M. Jones, "Rapid object detection using a boosted cascade of simple features", vol. 1, pp. I–511 (2001).

[108] Viola, P., M. Jones and D. Snow, "Detecting pedestrians using patterns of motion and appearance.", vol. 63, pp. 734–741 (2003).

[109] Wahl, F., M. Milenkovic and O. Amft, "A distributed pir-based approach for estimating people count in office environments", in "2012 IEEE 15th International Conference on Computational Science and Engineering", pp. 640–647 (2012).

[110] Wang, M. and X. Wang, "Automatic adaptation of a generic pedestrian detector to a specific traffic scene", in "CVPR 2011", pp. 3401–3408 (2011).

[111] Wang, X., Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey", IEEE Communications Surveys Tutorials **22**, 2, 869–904 (2020).

[112] Weppner, J. and P. Lukowicz, "Bluetooth based collaborative crowd density estimation with mobile phones", pp. 193–200 (2013).

[113] Williams, S., *Hands-On Chatbot Development with Alexa Skills and Amazon Lex: Create custom conversational and voice interfaces for your Amazon Echo devices and web platforms* (Packt Publishing Ltd, 2018).

[114] Wong, A., M. Famuori, M. J. Shafiee, F. Li, B. Chwyl and J. Chung, "Yolo nano: a highly compact you only look once convolutional neural network for object detection", in "2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)", pp. 22–25 (2019).

[115] Wu, B. and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors", Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 **1**, 90–97 Vol. 1 (2005).

[116] Yao, Y., X. Zhang, Y. Liang, X. Zhang, F. Shen and J. Zhao, "A real-time pedestrian counting system based on rgb-d", in "2020 12th International Conference on Advanced Computational Intelligence (ICACI)", pp. 110–117 (2020).

[117] Yin, J., Z. Chen, K. Zhou and C. Yu, "A deep learning based chatbot for campus psychological therapy", arXiv preprint arXiv:1910.06707 (2019).

[118] Zappi, P., E. Farella and L. Benini, "Tracking motion direction and distance with pyroelectric ir sensors", IEEE Sensors Journal **10**, 9, 1486–1494 (2010).

[119] Zhang, C., H. Li, X. Wang and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2015).

[120] Zhang, Q., Y. Yang and Y. N. Wu, "Unsupervised learning of neural networks to explain neural networks (extended abstract)", (2019).

[121] Zhou, L., J. Gao, D. Li and H.-Y. Shum, "The design and implementation of xiaoice, an empathetic social chatbot", Computational Linguistics **46**, 1, 53–93 (2020).

[122] Zhu, L., Z. Deng, X. Hu, C.-W. Fu, X. Xu, J. Qin and P.-A. Heng, "Bidirectional feature pyramid network with recurrent attention residual modules for shadow detection", in "ECCV", (2018).