SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 13, 2023

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Yen-Jung Lu
Jason Fong
Cheng Zhang
Chan Nam Tieu
Niyibitanga Inosa

ENTITLED

A Study of Processing Beehive Monitoring Data in Edge and Cloud

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

B. Dezfouli
Thesis Advisor

N. Ling
N. Ling (Jun 14, 2023 19:07 PDT)
Department Chair

A Study of Processing Beehive Monitoring Data in Edge and Cloud

by

Yen-Jung Lu
Jason Fong
Cheng Zhang
Chan Nam Tieu
Niyibitanga Inosa

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 14, 2023

A Study of Processing Beehive Monitoring Data in Edge and Cloud

Behnam Dezfouli

Department of Computer Science and Engineering
Santa Clara University
Santa Clara, California
June 14, 2023

ABSTRACT

Bees are vital pollinators for numerous crops and plants, playing an essential role in maintaining the world's ecosystem and food security. Without bees, the global food supply would be threatened, potentially leading to food shortages and increased food prices. Current beehive monitoring solutions in the market are often prohibitively expensive, some commercial beehive monitoring solutions can cost upwards of thousands of dollars. This makes them inaccessible to many small-scale beekeepers. To help protect bee health and ensure the survival of thousands of plant species worldwide, this project aims to create an accessible solution for individuals to monitor the health and activity of bee hives. The solution would use cheap and accessible tools to capture the activity of the bees, analyze it with local weather data, and alert the user when the actual activity is significantly different from the prediction, thus allowing beekeepers to more effectively tend to their many hives around the world.

Table of Contents

1 Introduction & Background **1**
- 1.1 Motivation . . . 1
- 1.2 Current Solutions . . . 2
  - 1.2.1 BuzzBox Mini . . . 2
  - 1.2.2 EyesonHives . . . 3
  - 1.2.3 HM-6 Hive Monitor . . . 4
- 1.3 Project Description . . . 5

2 Requirements **7**

3 Project Risks **9**

4 Design & Implementation **11**
- 4.1 Design Rationale . . . 11
- 4.2 Conceptual Model . . . 12
- 4.3 Technologies Used . . . 14
- 4.4 Use Cases . . . 14
  - 4.4.1 Setting Up . . . 14
  - 4.4.2 Modifications . . . 14
- 4.5 Data transmission . . . 15
  - 4.5.1 Current Research . . . 15
  - 4.5.2 Implementation . . . 15

5 Test Plan & Performance Evaluation **17**
- 5.1 Test Plan . . . 17
- 5.2 Experimentation . . . 17
- 5.3 Results . . . 18
  - 5.3.1 Camera and CV pipeline . . . 18
  - 5.3.2 Machine Learning Accuracy . . . 19
  - 5.3.3 CPU Usage and Power Consumption . . . 19

6 Conclusion & Future Works **24**

A Appendix **26**
- A.1 Sample code for calling python script with parameters hourly . . . 26
- A.2 Sample code for handling tracked objects . . . 26
- A.3 Source Code for activity prediction model . . . 27
- A.4 Source Code for startup script . . . 28
- A.5 Server code . . . 28
- A.6 Client code . . . 29

List of Figures

4.1 Conceptual Model of the System 12
4.2 interconnection of system modules 13
4.3 Observed Rates of Images Sent 16

5.1 tracking system testing 18
5.2 ML Prediction Accuracy 19
5.3 Power and Energy Readings 21
5.4 CPU Usage at Different Frame rates 22

Chapter 1

Introduction & Background

1.1 Motivation

Bees are more than just flying pests that sting when you get too close. They serve a far greater role, responsible for ensuring the survival of thousands of different plant species worldwide while producing honey. Around 80 percents of all of the crops we have are dependent on pollination by bees, bees contribute to a significant portion of global crop yields (1). Without bees, the global food supply would be threatened, potentially leading to food shortages and increased food prices. bees are pollinators for more than 100 U.S.-grown crops, valued as much as 18 billion dollars (2). As a critical element in both the ecosystem and economy, it's important to help bees thrive.

Bee pollination has also played a significant role in the growth of our society. The process of bee pollination helps plants to reproduce, which ultimately leads to the production of fruits, vegetables, and seeds (3). This, in turn, provides humans with food, medicine, and fiber for clothing. In addition, honey produced by bees is a natural sweetener and is used in various food products, cosmetics, and medicines.

The decline of bee populations and beehive colonies has been a growing concern in recent years. According to a study published in the Journal of Economic Entomology, honey bee colonies in the United States declined by 42 percent from April 2014 to April 2015 due to factors such as habitat loss, pesticide exposure, and disease (4). The decline of bee populations and beehive colonies is a major concern as it could have significant economic and ecological consequences. The importance of beehive monitoring cannot be overstated in addressing this issue.

Beehive monitoring is essential to maintain the health and well-being of bees and their colonies. By monitoring the activity of the hive and tracking bee behavior, we can detect anomalies and potential problems, such as diseases, pests, and environmental stressors that can negatively impact the bee population. Without beehive monitoring, diseases could even lead to colony collapse disorder. Colony Collapse Disorder is a phenomenon where worker bees abruptly disappear from the hive, leaving behind the queen and a few immature bees (5). It is a significant concern for beekeepers and can lead to severe economic consequences. By identifying these issues early, we can take actions to prevent further damage and protect the bees.

1.2 Current Solutions

Currently, beehive monitoring is available but not accessible to individuals. One of the main issues with current beehive monitoring products on the market is their high cost and unreliability. Commercial monitoring solutions cost anywhere from 250 (6) to thousands of dollars (7) while DIY solutions often lack the capability to properly analyze and act upon the data collected. This makes them inaccessible to small-scale beekeepers or those with limited resources. Furthermore, some of these products are prone to technical issues or failures, which can further increase their cost and reduce their effectiveness.

1.2.1 BuzzBox Mini

One of the beehive monitoring products in the market that has issues is the BuzzBox Mini(8), a beehive monitoring system that promises to provide real-time information on beehive health and activity. They claimed to use artificial intelligence to inspect a beehive's health and continually report updates to a mobile app throughout the day. There are several features that BuzzBox Mini promises to offer. One of the features is its ability to detect a range of hive conditions such as swarming, missing queen, healthy, sick, or collapsed hives in real-time. This feature allows beekeepers to quickly respond to any issues that arise and take action to prevent any further damage to their hives.

Another feature that BuzzBox mini promises to provide is its ability to monitor temperature, local

weather conditions, and humidity (8). This allows beekeepers to understand how external factors affect the behavior of their hives and take necessary steps to ensure their hives are healthy and thriving. The BuzzBox mini also claims to contain anti-theft systems that alert you when your hive is disturbed. This feature is useful for beekeepers who keep their hives in remote locations and want to ensure that their hives are protected from theft.

However, there is a fatal problem with the transmission test of this product, which is a critical step in ensuring that the product is functioning properly. Without proper transmission, the product is essentially useless, as it cannot accurately monitor the beehive or provide useful data to the beekeeper (9).

This issue with the BuzzBox Mini highlights the importance of thoroughly testing and evaluating beehive monitoring products before investing in them. It also underscores the need for more affordable and reliable solutions for small-scale beekeepers and people with limited resources. By improving the accessibility of and reliability of beehive monitoring technology, we can better support the health and productivity of bee populations, which are crucial for the health of our ecosystems and agricultural systems.

1.2.2 EyesonHives

Another beehive monitoring product in the market is Eyesonhives (10). It is a hive monitoring system that provides beekeepers with the ability to remotely monitor and track the activity of their beehives. The system involves positioning a camera approximately two feet away from the beehive entrance and using the camera to capture the number of bees surrounding the entrance of the beehive.

One of the key features of EyesonHives is its real-time hive monitoring capability. The product utilizes advanced sensors and technology to collect and analyze data from the hive, providing beekeepers with up-to-date information about the health and activity of their bees. This real-time monitoring allows beekeepers to stay informed about important hive metrics such as temperature, humidity, weight, and sound levels, enabling them to take timely and proactive measures to ensure

the well-being of their colonies.

While EyesonHiives provides a bee-monitoring solution, the price of this product may pose a challenge for small-scale beekeepers or those operating on a tight budget. The system costs from three hundred and eighty dollars (10), the cost may be prohibitive for beekeepers with limited financial resources or those who are just starting in the field. Affordability is a crucial factor in ensuring accessibility to bee monitoring technology for a wider range of beekeepers, especially those with smaller operations or hobbyist beekeepers.

1.2.3 HM-6 Hive Monitor

HM-6 hive Monitor is another beehive monitoring solution in the market, which is offered by SolutionBee that enables beekeepers to monitor their beehives remotely (11). This device utilizes scales to provide accurate hive weight data, while also delivering information on the outside temperature. By gathering real-time data on these parameters, beekeepers can gain valuable insights into the health of their beehives.

One of the features of the HM-6 Hive Monitor is its ability to measure hive weight. By accurately measuring the weight of the hive, beekeepers can monitor the honey production and overall vitality of the colony. This feature allows beekeepers to make informed decisions regarding hive management, such as determining the optimal time for honey harvesting or identifying potential issues that may require attention.

In addition to hive weight monitoring, the HM-6 Hive Monitor also provides valuable information about the outside temperature surrounding the beehive. By continuously monitoring the outside temperature, beekeepers can assess environmental conditions and make appropriate adjustments to ensure the well-being of their colonies. This feature enables beekeepers to better understand the impact of temperature on hive productivity.

Nevertheless, the price of the HM-6 Hive Monitor may cause a drawback for beekeepers. The system costs from three hundred and fifty-nine dollars, which is a high price that limits accessibility

for hobbyist beekeepers or those who are just starting in the field. Affordability is an important factor to consider, as it ensures that a wider range of beekeepers can benefit from the advantage of hive monitoring.

1.3 Project Description

This project aims at protecting bee health by creating a solution that leverages cheap cameras and sensors as well as the scalability of the cloud, allowing users to assess the health and strength of a bee hive and possibly monitoring bee induced activity such as initial flights, bearding and swarm activity, all at an accessible price.

The project proposes the use of off-the-shelf webcam and Raspberry pi 4 for the hardware end, as they are relatively cheap and accessible. The camera will be positioned at the hive entrance to take videos of bees coming in and out of the hive, this IoT device would perform some local processing of the video information and upload the data to the cloud. An algorithm will be used to detect the number of bees entering and exiting a hive through the video feed, and it will run completely on edge to reduce the amount of data that is uploaded. Simultaneously, deep learning methods will be used to distinguish between worker and drone bees. From this initial process, we will obtain data regarding the number of bees entering and exiting the hive and which type of bee they are. This data will be collected, visualized through a graphic interface, and analyzed by a machine learning algorithm for anomaly detection. This ML algorithm would predict the activity based on weather and past records on a regular basis, and, if a mismatch is found in the prediction and actual activity, the program would alert the beekeeper. A web server will be used to provide an intuitive interface for beekeepers to assess the status of the hive and configure the system to their needs.

Since many beehive locations are off-grid with no access to WiFi or power, there may be constraints on the power & network usage of this system. Thus, one consideration is to run the machine learning model completely on edge, potentially reducing the amount of network used. One problem is doing so consumes more power. Therefore, we want to experiment with using a less powerful Raspberry Pi model with the Coral Accelerator, an Application Specific Integrated Circuit (ASIC)

designed to enable high speed machine learning inferencing. Another potential drawback is that Machine Learning (ML) models running on cloud can continuously learn and improve whereas ML models on edge do not have the capability to do so.

The source code of this project can be accessed at https://github.com/SIOTLAB/BeehiveMonitoring

Chapter 2

Requirements

The project at hand involves the development of a beehive monitoring system that utilizes computer vision and machine learning technologies to track and analyze the activity of bees at the entrance of the hive. To ensure the successful implementation of this project, there are certain requirements that must be met.

The first requirement is the development of an accurate and efficient bee-tracking algorithm. The algorithm must be able to detect and track individual bees as they enter and exit the hive, and record their movement patterns. This algorithm plays an important role in the system, and its accuracy and efficiency are critical to the success of the project.

The second requirement is the implementation of an anomaly detection algorithm that can predict the activity of the hive in the future. The anomaly detection algorithm will use the data collected by the bee tracking algorithm to detect unusual patterns of activity, such as a sudden decrease in the number of bees entering or exiting the hive. This will be achieved through the use of machine learning techniques and is critical for the early detection of potential issues within the hive.

The third requirement is the integration of current and predicted weather data with the bee activity data. The system will combine the two sets of data to gain insight into how weather conditions affect bee activity. This will help to develop a more accurate and comprehensive understanding of hive behavior.

The fourth requirement is the development of a user interface that allows for the visualization of bee activity data. This will enable users to monitor hive behavior in real-time, view historical data,

and receive alerts when unusual patterns of activity are detected. The user interface should be intuitive and easy to use, and accessible from any device with an internet connection.

The fifth requirement is the implementation of an alert system that notifies users when an abnormal pattern of activity is detected. The alert system should be customizable and allow users to set their own threshold for what constitutes abnormal behavior. This will enable users to respond quickly to any issues within the hive and prevent potential losses.

Finally, the system must be designed with scalability in mind. As the number of hives being monitored increases, the system must be able to handle the additional data without sacrificing accuracy or efficiency. This will ensure that the system can grow alongside the needs of its users and remain effective in the long term.

Chapter 3

Project Risks

If not tested and implemented properly, there is the risk of sending false alarms or no alarms when there should be, rendering the product entirely useless. As such, thorough testing must be conducted to minimize the impact of mistakes. In order to remedy this, we have implemented certain tests to ensure that our program detects activity as accurately as possible without slowing down the detection algorithm too severely. Additionally, some checks have been added to the anomaly detection algorithm to ensure that alarms are sent out only when necessary.

Since there are many kinds of bees, they might have different appearances and activity pattern. Thus, while the algorithm still applies, specific training might need to be done on deployment, which add cost. This is inevitably a part of machine learning but the only feature affected would be detection of bee kind since abnormal activity detection, the most important feature, is trained on a hive-to-hive basis in the first place.

As none of the members of this group are expert in expected bee activity, we can potentially misinterpret the data or introduce false data into the system. To prevent this, we need to cooperate closely with the external advisors who are expert bee keepers. Over the past few quarters, we have scheduled regular meetings with our external advisors Gerhard and Lisa Eschelbeck, co-vice presidents of the Santa Clara Valley Beekeepers Guild, who have provided important insights for determining which factors are necessary to consider.

Additionally, performance will be dependent on WiFi connectivity. Due to the constant uploading of data to be processed, stored, and rendered, performance might slow down. This risk will be miti-

gated by testing and determining the proper bandwidth required to handle all the data transmission. While we may not be able to completely isolate our system from the internet, our system is able to perform its most basic functions of tracking and running anomaly detection entirely on the edge. However, the alarm system may need additional work to be done in order to ensure a functional system even in poor network conditions, perhaps utilizing cell plans for basic alert systems and data connection.

Poor weather is another factor that might hinder performance. In the case that it is cloudy or raining, visibility might be reduced, which will impact the the ability to monitor the beehive and its activity. One way to mitigate this risk is to put a structure around the hive to protect it without disturbing the usual activity or simply sending users an alert that visibility is low and the beehive can't be adequately monitored. However, more effective solutions were developed after in-person visits to the hive and more thorough analysis of the environment. This issue is difficult to fix on the software end considering the unpredictability of weather conditions we may face. To supplement the technical limitations of our software, this issue is being mitigated on the hardware end through effective weatherproofing of the hive entrance. Installation of a spindle that allows the camera to move horizontally could be another possible solution.

Chapter 4

Design & Implementation

4.1 Design Rationale

In order to monitor beehive activity, there are a few criteria that must be fulfilled. The first consideration is how to collect the data. After some discussion with the hardware team, we quickly settled on modifying the entrance of the beehive and setting up a camera to count the bees' movement. The camera needs to be calibrated since we want to determine the number of bees passing by, where the size of bees and location of the entrance are some of the most important factors for ensuring this is done accurately. The image passes through an OpenCV pipeline for object detection and another algorithm to perform anomaly detection. Because this requires a good amount of training data, we would need to collect them and interpret them correctly, which as for some expertise in beekeeping, our external advisors have been invaluable in supporting our understanding of these events.

This measured bee activity would be referred to as raw activity. It would be processed by some conventional statistical method to form a normalized activity to account for the change in bee hive population. This activity would then be fed into the anomaly detection algorithm that is a ML model which takes local weather, beehive sensor, and past activity to predict future activities. It is a prediction model but used in the assumption that, if a hive's behavior is significantly different to the prediction, it indicates an anomaly.

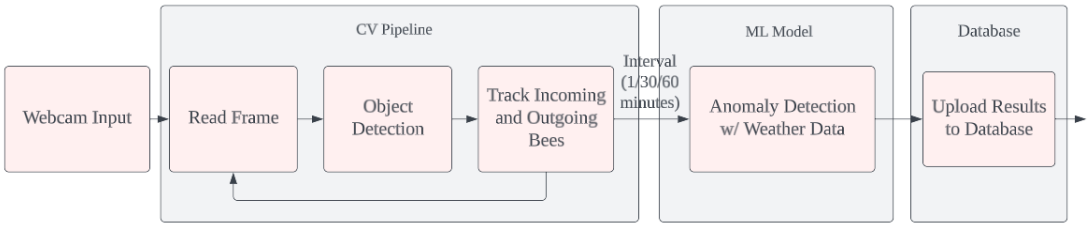


Figure 4.1: Conceptual Model of the System

4.2 Conceptual Model

Figure 4.1 above represents the conceptual model of the project focused on how data is collected and analyzed. As shown, the data is collected using cameras and sensors to be preprocessed within the edge, reducing the amount of data that needs to be sent to the database. This preprocessed data will be sent to a database, where it will be stored for some period of time before getting deleted. Using the Amazon Elastic Compute Cloud (EC2), this data will be further analyzed to obtain more information on beehive activity, such as the type of bee and whether it is entering or exiting the hive. This processed data will be stored in the database which will then be accessed by user though a web frontend. It will also be used in conjunction with the weather conditions to detect abnormal activity. If an anomaly is found, an alert will be logged in the database and sent to the user, alerting the beekeeper to unusual patterns in behavior and possibly take action to prevent a potential problem.

As this project is done in conjunction with other teams, this report will not include details about hardware and web front-end. As an overview, Figure 4.2 below outlines the interconnection between various components of the system. One team researched and developed aspects of the hardware and webcam streaming. They focused on ensuring that the system is resilient to a variety of weather conditions and that cameras placed at the hive entrance are capable of streaming video feed for user to view.

Another team developed the web front-end components, creating an intuitive user interface capable of allowing users to securely input their credentials and access the status of their hives. We have been working extensively with this group to ensure that data is sent correctly from the Pi to the server. As a part of this work, a set of APIs are negotiated to exchange data between the edge and the cloud.

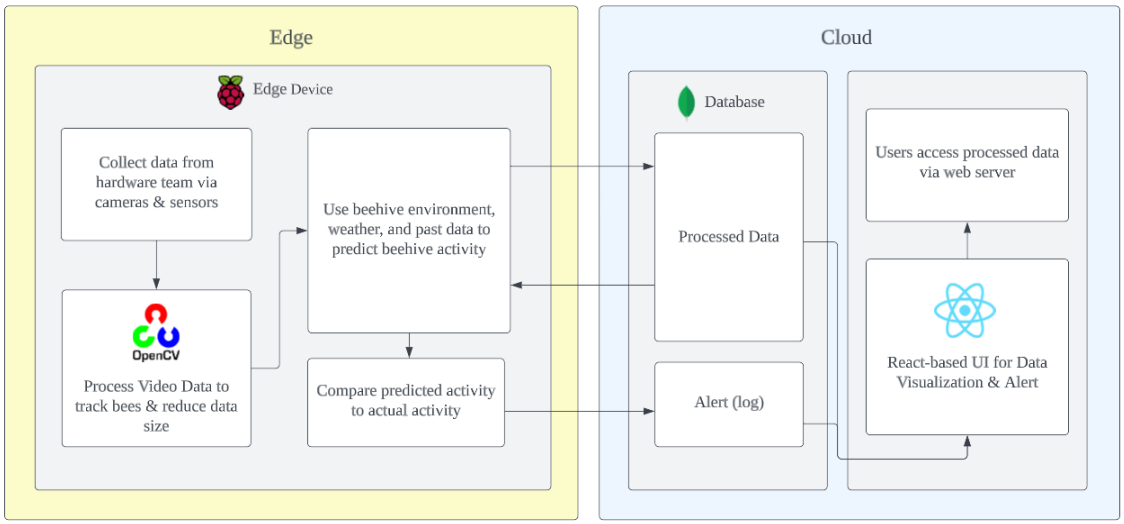


Figure 4.2: interconnection of system modules

Our machine learning model makes use of the Keras API, the high-level API of TensorFlow. It provides an approachable, highly-productive interface for solving machine learning (ML) problems, with a focus on modern deep learning. Keras covers every step of the machine learning workflow, from data processing to hyperparameter tuning to deployment(12). For this project, a ML model is used to predict bee activities and standard statistical method, such as standard distribution and deviation, is used for evaluating the prediction and future data points. Since the ML model outputs the beehive activity (a pair of integers), a linear layer is added before the output layer. At this end, a size of 16 is chosen to reflect the reasonable range of bee activities. On the other end, the input consists of 7 numbers including past activities and weather data. Thus, a layer size of 128 is chosen. and Fully connected dense layers are added in between to emulate any interactions. A number of past data would be used to train this model specifically to the hive before a prediction

are performed, eliminating hive-to-hive differences.

4.3 Technologies Used

The technologies used in this project include the Raspberry Pi 4B, a baseline single-board computer used for acquiring and preprocessing data, the Nvidia Jetson Xavier NX, an alternative and more powerful option to the Raspberry Pi specialized in performing AI inferencing, the Google Coral USB Accelerator, a hardware accelerator that adds an edge Tensorflow coprocessor to more efficiently run tensorflow-based ML methods, and a USB-C Power meter, a standard voltage meter used for measuring the power usage of devices while running programs.

4.4 Use Cases

Our system will provide users with access to live footage of their beehive, the ability to analyze the population of bees entering and departing the beehive, an analysis of the beehive environment such as temperature, humidity, light, and activity, and detection of any unusual behavior with an alert being sent to the beekeepers. From the user's point of view, there is not much direct interaction with our program, which focuses on the back-end implementation of the system.

4.4.1 Setting Up

With our current implementation, one of the few use cases that beekeepers should be aware of is the need to set up key components manually. This entails downloading the source code and the necessary libraries, then assigning entrance and threshold coordinates along with size threshold for tracked objects by editing values in the code directly. More detailed setup instructions are available on GitHub. It should run automatically at startup but it can be restarted manually should any unexpected errors occur, after which it would run indefinitely.

4.4.2 Modifications

Given that our project is open source, users are free to modify or add features as well, which can be done by editing source files in the Bee-Tracking-main directory. This can be done at the user's

discretion and doesn't require much further explanation.

4.5 Data transmission

4.5.1 Current Research

We researched the factors affecting the bandwidth and speed of data transmission on a Raspberry Pi. It's important to keep in mind the differences between bandwidth and speed, although at times they can affect one another. Bandwidth refers to how much data can be transmitted in a given period of time. One factor that can determine the bandwidth is the medium on which data is transmitted. Speed on the other hand is an indicator of the rate at which data moves from point a to point b. Some factors that can affect speed are the quantity and type of devices connected, the amount of traffic, uploading vs downloading, etc.

4.5.2 Implementation

For the implementation, we started by experimenting with different settings such as the frequency of performing anomaly detection and size thresholds of bees and the hive entrance after moving the raspberry pi on site. This was critical for identifying the best way to increase the bandwidth and speed of its data transmission, as well as improving the performance of the program as a whole. Although physical connections may have better performance, we focused on Wifi as our medium due to its ease of use in most scenarios.

In order to obtain images for local use, we implemented a Python program to stream video from the Raspberry Pi camera to the local computer. The program consists of two parts: the server and the client. The client code runs on the Raspberry Pi, while the server code runs on the local computer. The server starts a socket on the local host and listens for connections. The client will connect to the socket and sends a continual stream of images to the server. The client was able to send around 1000 images of size 640x480 in 60 seconds, corresponding to a rate of 16fps. However, when a movement took place, there was a 5-second delay before it could be observed

on the server's side (the local computer). Below is the chart measuring the rate in frames per second that the client was able to send for different image resolutions, as well as the outputs for the server.py and client.py code using an image resolution of 640x480.

Figure 4.3: Observed Rates of Images Sent

server.py code outoput:

```
C:\Users\admin\Documents\Code>python server.py
Image is 640 x 480
Image is verified
Image is 640 x 480
Image is verified
Image is 640 x 480
Image is verified
 ...
```

client.py code output:

```
admin@raspberrypi:~/VideoStreaming $ python client.py
Sent 1007 images in 60 seconds at 16.58 fps
admin@raspberrypi:~/VideoStreaming $ python client.py
Sent 978 images in 60 seconds at 16.28 fps
```

```
admin@raspberrypi:~/VideoStreaming $ python client.py
Sent 1009 images in 60 seconds at 16.74 fps
admin@raspberrypi:~/VideoStreaming $ python client.py
Sent 1006 images in 60 seconds at 16.75 fps
...
```

Chapter 5

Test Plan & Performance Evaluation

5.1 Test Plan

This system has 3 main components in which data is collected and analyzed, which includes camera, computer vision (CV) pipeline, and machine learning (ML) model. To fully evaluate the system it either need to be tested end-to-end or per component. As end-to-end testing requires beehive in various conditions, which is not easily accessible to us, component tests are used.

5.2 Experimentation

In practice, we acquired segmented recordings at the entrance of the hive and ran them through the bee tracking and anomaly detection algorithms. For bee tracking, we are using the OpenCV library, which is a computer vision library that can be used to implement object-tracking algorithms. By using OpenCV, we can track bees'movements and collect data on their behavior, such as the number of bees entering and leaving the hive. Once a bee is detected, OpenCV can draw a box around it to highlight its location and track its path to determine whether it is entering or exiting the hive. From this method, we were able to verify that the bee tracking was functional and output data in a format usable by the anomaly detection algorithm.

Analyzing bee activity data is a crucial step in understanding the behavior of a beehive. By using OpenCV, we can track and detect movement-based bee activity at the entrance of the hive. This data can then be accumulated over a specific time frame, such as 1, 30, or 60 minutes, and combined with current and predicted weather data. By custom-fitting the model to the historical data, we can

make inferences about the activity of the hive.

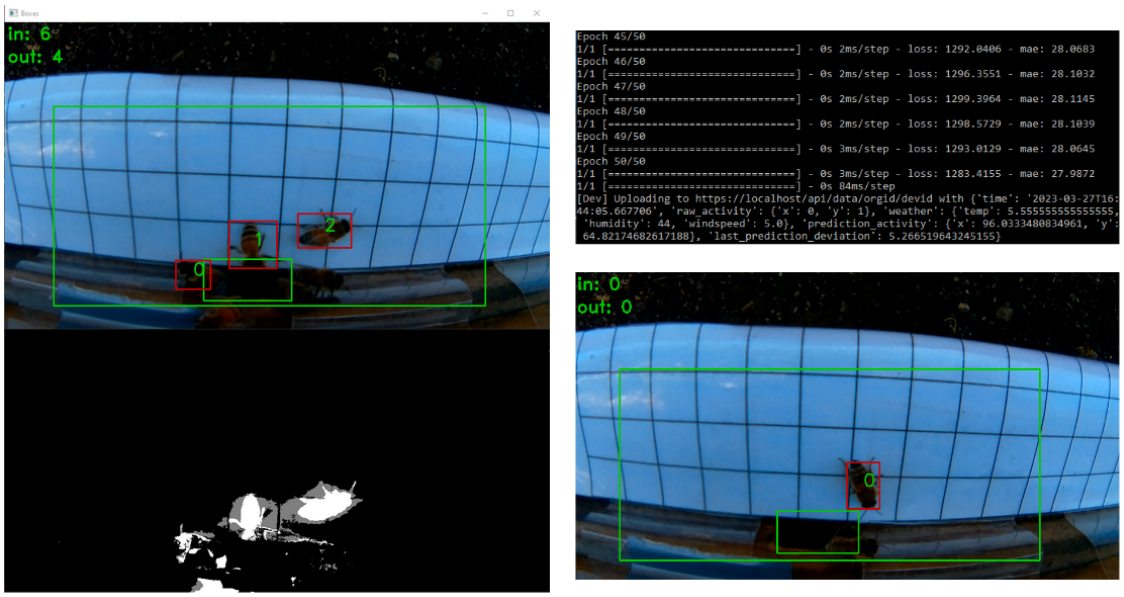


Figure 5.1: tracking system testing

Figure 5.1 above includes a few screenshots of our program used on one of many pre-recorded videos, where bees are detected and tracked as they move around the frame. Two rectangles are drawn to indicate the entrance of the hive and the threshold which bees must cross to be counted. For testing purposes, we included a counter for the number of bees entering or exiting the hive. At one minute intervals, we ran the script that performs anomaly detection and uploads relevant data to the web server, as shown in the top right image. Afterwards, the counters are cleared and the object detection program gathers information for the next iteration.

5.3 Results

5.3.1 Camera and CV pipeline

The camera used in this system gives a 1920x1080 stream at 30fps. After manual review of the footage as well as comparing it to the tracking result, it is deemed that the camera and CV pipeline are able to track bees reliably. Unfortunately, not enough footage have been manually reviewed to give a precise conclusion as to how accurate this part is.

5.3.2 Machine Learning Accuracy

Since the ML program outputs a z-score as a description of how significant the data point collected is, thus how abnormal it is. It can also be used to evaluate the accuracy of itself. Using synthesized date generated from a random linear correlation between weather and bee activity, as shown in Fig 5.2, the model would mark a data point abnormal 80% of time if a abnormal boundary of $z > 2$ is used. While this isn't impressive, it is based on non-sequential data where as actual bee data would have more sequential characteristics which the model is optimized for. Non-sequential test data is intentionally used to avoid testing the model with what is essentially a generative model of the same structure.

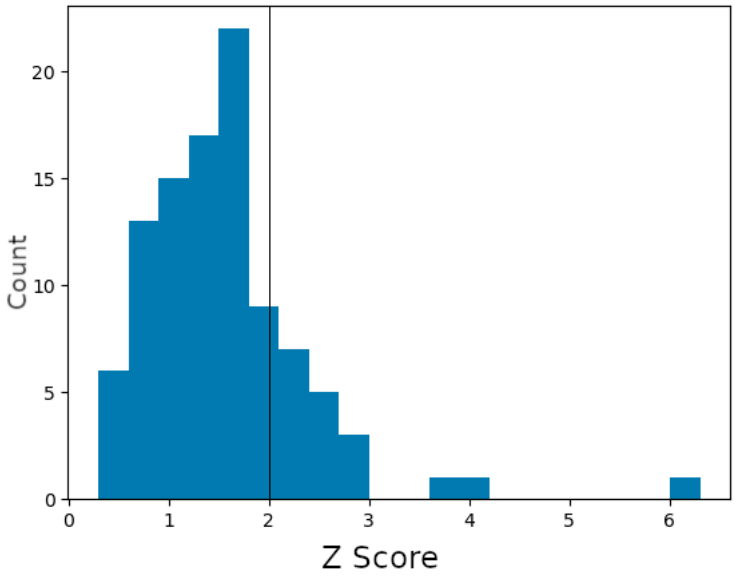


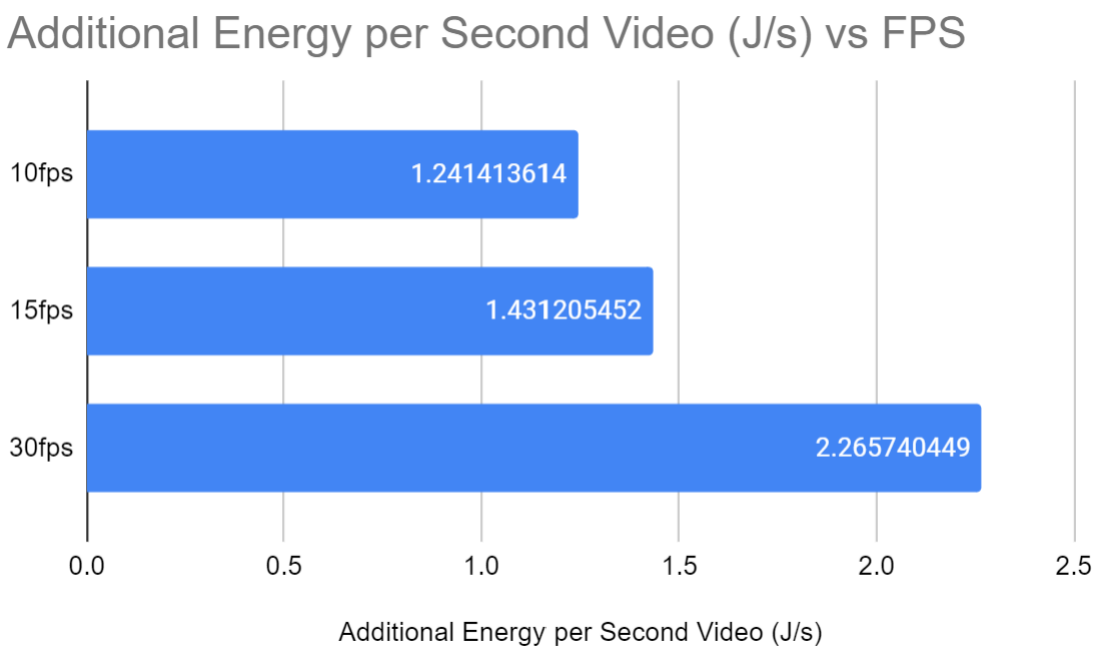
Figure 5.2: ML Prediction Accuracy

5.3.3 CPU Usage and Power Consumption

For testing, we ran the program with prerecorded camera footage at varying frame rates, specifically 10, 15, and 30 frames per second while evaluating a few key statistics. To measure the costs of our computer vision and anomaly detection pipeline, we measured the duration that it took to completely parse through every frame, the average CPU usage of the program, and the overall

power consumption as it was running. CPU usage was measured using the top command, which displays the processing power used by each program, while power consumption was measured using a USB-C power meter. For our control test, we measured the power consumption of the Pi while it was idle. From these factors, we were able to compare the average power and CPU usage of our program against the idle state and determine the general resource consumption of our computer vision and anomaly detection module over an extended period. Our estimates for energy were formed by subtracting idle power from power consumption while running the program to isolate the power consumption of our program, then multiplying it by the time it would take to process one second of video to find an estimate of energy per second video processed in units of J/s.

During these tests, we also noted a few factors that affect the overall performance of the program in different tests. For one, the number of pixels being read by the camera directly affects the performance, where down-scaling the size of the video input produced noticeable improvements. Additionally, individual hardware or software differences between Pi's may affect the overall behavior of the program.



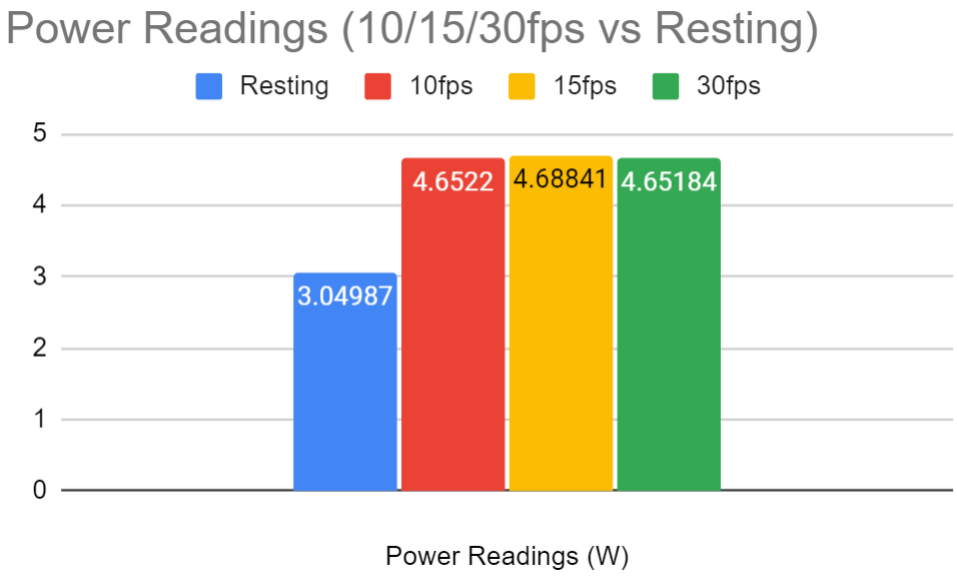


Figure 5.3: Power and Energy Readings

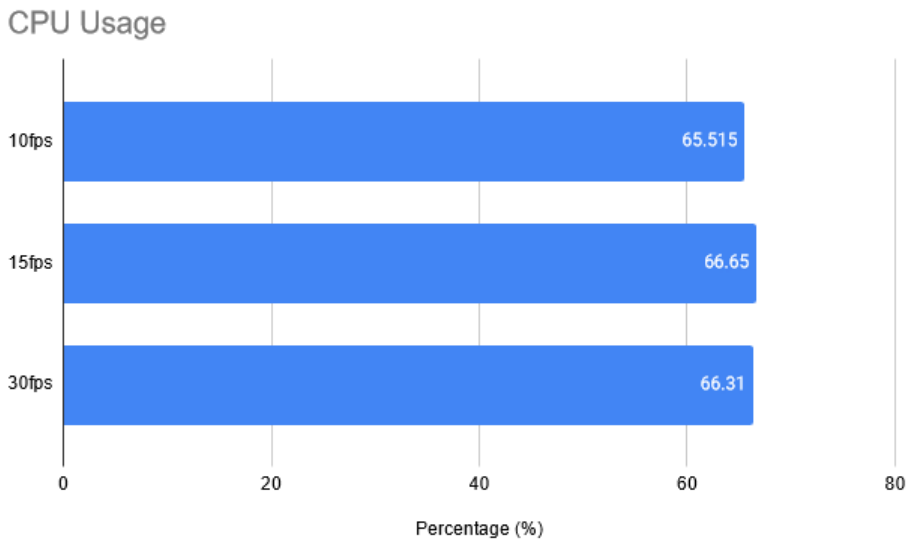


Figure 5.4: CPU Usage at Different Frame rates

From these tests, we observed that the CPU usage and power consumption of the raspberry pi were consistent regardless of the frame rate of the video itself. Rather, because we were using pre-recorded videos, the program would always be reading through each frame at the same speed. From this, we noticed that the time it took for the process to parse through each frame was different from the length of the video, so while the CPU usage and power consumption were consistent throughout, the energy usage for the different videos was vastly different. Unexpectedly, the energy consumption was found to not be directly proportional to the frame rate of the video. This may have to do with a variety of reasons, such as the number of frames that don't have information being read or modified. Nonetheless, it is useful to know that parsing videos at higher frame rates causes a less-than-proportional increase in energy consumption. Additionally, using higher frame rates provides markedly improved program behavior, allowing the computer vision pipeline to more effectively track bees moving at faster speeds.

While energy usage is an important takeaway from these measurements, it may not prove as useful as our findings of the power consumption and CPU usage depending on how we choose to run it. Unlike going through a video frame-by-frame until the end, the scale of time becomes less

relevant as we continuously read from the camera until the process is terminated. Either way, an improvement in power consumption would lead to a direct decrease in energy consumption, so our focus lies in improving the program's efficiency regardless.

Chapter 6

Conclusion & Future Works

Through the collective efforts of multiple teams, we were able to create a baseline for an open source system capable of alleviating the burden of beekeepers around the world through the assistance of computer vision and machine learning. Starting from a Raspberry Pi and a small camera positioned in front of the hive, we ended with a framework that allows beekeepers greater access to important hive data at a much more affordable price than solutions available on the market. With many of our main objectives achieved, we hope that future groups can focus on providing more depth to the project than we were able to in the span of our senior year.

While we have established a solid foundation for this project, there remain a multitude of unexplored avenues to pursue. The possibilities for refinement of existing code and the integration of new features are quite promising. In the future, changes and optimizations can be made to improve the tracking system's resiliency. For instance, the tracking system can be further optimized to work in more cases and avoid getting confused by shadows or blur. Another potential change would be to allow the program to determine the appropriate size threshold for bee tracking and fit the entrance size, which may be more complicated but could save beekeepers the time needed to adjust the settings for each hive. The power draw should also be validated, and the system can potentially be migrated to platforms with lower power draw and cost.

We were also considering using bee size or other distinguishing factors to determine the type of bee, whether drone or worker. However, this provided a few constraints that we were unable to work out amid the higher priority objectives. In terms of the algorithm, further experimentation is

needed to determine whether we will use OpenCV image classifier or do we need to build a new model in, for example, Tensorflow. Furthermore, a significant amount of training data would be required for it to classify bees of all shapes and sizes at different angles, so this objective has yet to be pursued.

When parsing bee activity data, size should also be considered to avoid inconsistencies caused by camera position and shape, particularly when a grid is not added to the entrance. While we have a basic framework for distinguishing bees based on size, more data is required to validate and train the system to account for these inconsistencies. Future work will also involve testing the model with live data and refining it further to account for variables such as temperature, humidity, and weather patterns. As machine learning continues to gain prominence in modern software, we can expect significant advancements to our system in the foreseeable future.

Appendix A

Appendix

This is a selection of key part of the source code, full source can be found at

https://github.com/SIOTLAB/BeehiveMonitoring.

A.1 Sample code for calling python script with parameters hourly

```
time_t currentTime = time(0);
tm current;
localtime_s(&current, &currentTime);

if (current.tm_hour != hour) {
string cmd = "python process_data_dev.py " + to_string(in) + " " + to_string(out);
//  number of bees going entering as argv[1], exiting as argv[2]
 system(cmd.c_str()); // call python script
 // Update variables
    hour = current.tm_hour;
     in = 0;
     out = 0;
    }
```

A.2 Sample code for handling tracked objects

```
for (int i = 0; i < contours.size(); i++) {
// for objects on screen, add to currentObjects if above size threshold
    if (((boundingRect(contours[i]).height) < 50 || (boundingRect(contours[i]).width)
    < 50)) continue;
    Object newObject(contours[i]);
    currentObjects.push_back(newObject);
  }

 if (firstFrame == true) {
    for (auto& currentObject : currentObjects) {
    objects.push_back(currentObject);
    }
```

```
  }
   else {
    matchCurrentFrameToExisting(objects, currentObjects, frame);
    }
}
```

A.3 Source Code for activity prediction model

```
# Weather: temp, humidity, windspeed
# Activity: numIn, numOut
# model input: 2 previous activity pair & next weather
# model output: predicted activity pair
# model trained on last 360 data
def create_model():
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Dense(128, input_shape=(7,), activation='relu'))
    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dense(32, activation='relu'))
    model.add(tf.keras.layers.Dense(16, activation='relu'))
    model.add(tf.keras.layers.Dense(2, activation='linear'))
    model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
    return model

# Generate testing data

temperature = np.random.normal(20,10,size=(100,1))
humidity = np.random.normal(60,10,size=(100,1))
windspeed = np.abs(np.random.normal(10,10,size=(100,1)))
weathers = np.concatenate((temperature,humidity,windspeed),axis=1)
some_correlation = np.random.normal(2,1,size=(3,2))
activities = weathers @ some_correlation

X_train = np.concatenate((activities[-39:-3],activities[-38:-2],weathers[-37:-1]),
axis=1)
Y_train = activities[-37:-1]
X_test = np.concatenate((activities[-4:-2],activities[-3:-1],weathers[-2:]),axis=1)
Y_test = activities[-2:]

model.fit(X_train, Y_train, epochs=50, batch_size=64, validation_data=(X_test, Y_test))
prediction = model.predict(X_test)

np.max(np.abs((prediction-Y_test)/np.std(Y_train, axis=0)))

%%capture
test_results = np.zeros(100)
for i in range(100):
    model = create_model()
    temperature = np.random.normal(20,10,size=(50,1))
    humidity = np.random.normal(60,10,size=(50,1))
    windspeed = np.abs(np.random.normal(10,10,size=(50,1)))
    weathers = np.concatenate((temperature,humidity,windspeed),axis=1)
    some_correlation = np.random.normal(2,1,size=(3,2))
```

```
    activities = weathers @ some_correlation

    X_train = np.concatenate((activities[-39:-3],activities[-38:-2],weathers[-37:-1]),axis=1)
    Y_train = activities[-37:-1]
    X_test = np.concatenate((activities[-4:-2],activities[-3:-1],weathers[-2:]),axis=1)
    Y_test = activities[-2:]

    model.fit(X_train, Y_train, epochs=50, batch_size=64, validation_data=(X_test, Y_test))
    prediction = model.predict(X_test)
    test_results[i]=np.max(np.abs((prediction-Y_test)/np.std(Y_train, axis=0)))
```

A.4 Source Code for startup script

```
#!/bin/bash

# output logging
echo "\n\n"
echo "[INFO] Service Started at $(date)"

# setup pwd
cd /home/pi/Downloads/Bee-Tracking-main

# setup env variables
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python
if [[ ! -e env.sh ]]; then
    # default env file
    cp env.sh.template env.sh
    serial=$(uuid)
    printf "export SDP_DEVICE_SERIAL=$serial\n" >> env.sh
    printf "[INFO] Configuration file not found, new config created with serial number
    $serial\n\n"
fi
source env.sh

# start CV program
/home/pi/Downloads/Bee-Tracking-main/Webcam
```

A.5 Server code

```
# server.py

import io
import socket
import struct
from PIL import Image
import matplotlib.pyplot as pl

server_socket = socket.socket()
port = 8000
```

```
server_socket.bind(('169.254.241.227', port))
server_socket.listen(0)

connection = server_socket.accept()[0].makefile('rb')
try:
    img = None
    while True:
        # Read the length of the image as a 32-bit unsigned int. If the length
        # is zero, exit the loop
        image_len = struct.unpack('<L', connection.read(struct.calcsize('<L')))[0]
        if not image_len:
            break
        # Construct a stream to hold the image data and read the image data
        # from the connection
        image_stream = io.BytesIO()
        image_stream.write(connection.read(image_len))
        # Rewind the stream, open it as an image with PIL and do some
        # processing on it
        image_stream.seek(0)
        image = Image.open(image_stream)

        if img is None:
            img = pl.imshow(image)
        else:
            img.set_data(image)

        pl.pause(0.01)
        pl.draw()

        print('Image is %dx%d' % image.size)
        image.verify()
        print('Image is verified')

finally:
    connection.close()
    server_socket.close()
```

A.6 Client code

```
# client.py

import io
import socket
import struct
import time
import picamera

class SplitFrames(object):
    def __init__(self, connection):
        self.connection = connection
        self.stream = io.BytesIO()
        self.count = 0
```

```
    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # Start of new frame; send the old one's length then the data
            size = self.stream.tell()
            if size > 0:
                self.connection.write(struct.pack('<L', size))
                self.connection.flush()
                self.stream.seek(0)
                self.connection.write(self.stream.read(size))
                self.count += 1
                self.stream.seek(0)
        self.stream.write(buf)

# Connect a client socket to my_server:8000
client_socket = socket.socket()
port = 8000
client_socket.connect(('169.254.241.227', port))

# Make a file-like object out of the connection
connection = client_socket.makefile('wb')
try:
    output = SplitFrames(connection)
    with picamera.PiCamera(resolution='VGA', framerate=30) as camera:
        # Let the camera warm up for 2 seconds
        time.sleep(2)

        # Note the start time and construct a stream to hold image data temporarily
        start = time.time()
        # Recording for 60 seconds
        camera.start_recording(output, format='mjpeg')
        camera.wait_recording(60)
        camera.stop_recording()
        # Write the terminating 0-length to the connection to let the
        # server know we're done
        connection.write(struct.pack('<L', 0))
finally:
    connection.close()
    client_socket.close()
    finish = time.time()
print('Sent %d images in %d seconds at %.2ffps' % (
output.count, finish-start, output.count / (finish-start)))
```

Bibliography

[1] B. Randall, "The value of birds and bees," Jun 2022. Available at https://www.farmers.gov/blog/value-birds-and-bees#:~:text=Honey, accessed on June 04, 2023.

[2] Growers House, "Smartbee environmental base system (the hive + lth sensor + smart strip 4)." Available at https://growershouse.com/smartbee-environmental-base-system-the-hive-lth-sensor-smart-strip-4, accessed June 04, 2023.

[3] U. E. Programme, "Why bees are essential to people and planet." Available at https://www.unep.org/news-and-stories/story/why-bees-are-essential-people-and-planet, accessed on June 04, 2023.

[4] S. G. Potts, V. Imperatriz-Fonseca, H. T. Ngo, M. A. Aizen, J. C. Biesmeijer, T. D. Breeze, L. V. Dicks, L. A. Garibaldi, R. Hill, J. Settele, and et al., "Safeguarding pollinators and their values to human well-being," *Nature*, vol. 540, no. 7632, p. 220–229, 2016. accessed on May 28, 2023.

[5] J. D. Evans and Y. J. Chen, "Colony collapse disorder and honey bee health," *Honey Bee Medicine for the Veterinary Practitioner*, p. 229–234, 2021. Accessed on June 01, 2023.

[6] P. b. R. M. Nowierski, "Pollinators at a crossroads," Jun 2020. Available at https://www.usda.gov/media/blog/2020/06/24/pollinators-crossroads, accessed on June 04, 2023.

[7] Beehivemonitoring, "Bee hive weight monitoring." Available at https://www.beehivemonitoring.com/en/, accessed on June 04, 2023.

[8] osbeehives, "Buzzbox mini." Available at https://www.osbeehives.com/products/buzzbox-mini#, accessed on June 08, 2023.

[9] pdanhieux, "Anyone successfully using os buzzbox mini??," Apr 2022. Available at https://www.beesource.com/threads/anyone-successfully-using-os-buzzbox-mini.370843/, accessed on June 08, 2023.

[10] Eyesonhives, "Eyesonhives scout b - beehive monitor," Apr 2023. Available at https://www.eyesonhives.com/product/eyesonhives-scout/#reviews, accessed on June 11, 2023.

[11] Solutionbee, "Hm-6 wifi hive monitor (weight amp; outside temperature)," Jan 2023. Available at https://solutionbee.com/store/HM-6-WiFi-Hive-Monitor-Weight-&-Outside-Temperature-p180473748, accessed on June 12, 2023.

[12] "Keras: The high-level api for tensorflow." Available at https://www.tensorflow.org/guide/keras, accessed on June 13, 2023.