High Level Design (HLD)

UGV (Unmanned Ground Vehicle) based

Surveillance

Revision Number: 1.0

Last date of revision: 12/04/2024

Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | Author |
| 12/04/2024 | 1 | Initial HLD - VI .0 | Vishal Kumar Mahato |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

Document Version Control...

Abstract.........................................................................

1. Introduction.5

1 .1 Why this High-Level Design Document? ..... . . . . . . . . . .5

1 .2 Scope. . . . . .5

1.3 Definitions ... .5

1. General Description. . . . . .6

2. 1 Product Perspective .. . . . . .6

2.2 Problem statement .. . . . . .6

#### 2.3 PROPOSED SOLUTION .. . . . . .6

2.4 FURTHER IMPROVEMENTS. .6

2.5 Technical.6

2.6 Data Requirements .

2.7 Tools used........................................................................

 2.7. 1 Hardware Requirements........................................................................

2.7.2 ROS(Robotic Operating System) .. .9

2.8 Constraints

2.9 Assumptions................................................................... .. ..

1. Design Details ..... ..... ..... ................................ . . . . . . 1 0
   1. Process Flow....................................................................................................... 1 0
      1. Model Training and Evaluation . . . . . . 1 0
      2. Deployment Process 1 1
   2. Event log 1 1
   3. Error Handling . . . . . . . 1 1
   4. Performance........................................................................................................12
   5. Reusability................................................. . . . . . 1 2
   6. Application Compatibility. . . . . . 1 2
   7. Resource Utilization. . . . . . 1 2
   8. Deployment ..... ..... ........................ .. . . . . . . . 1 2
2. Dashboards. . . . . . 1 3
   1. KPls (Key Performance Indicators).............................................. . . . . . 1 3
3. Conclusion .. . . . 1 4

Abstract

This project explores the use of machine learning to classify mushrooms as edible or poisonous. We leverage data from the Audubon Society Field Guide on 23 gilled mushroom species.

The approach follows classical machine learning steps: data exploration and cleaning, feature engineering, and model building/testing. We evaluate various algorithms including Support Vector Machines, Decision Trees, Random Forests, and K-Nearest Neighbors.

The goal is to develop a model that accurately predicts edibility based on a mushroom's descriptive characteristics. However, the abstract concludes by emphasizing that the model is a tool and should not replace expert identification or reliable field guides for mushroom consumption.

# 1 Introduction

## 1.1 Why this High-Level Design Document?

## This HLD outlines the overall design for our machine learning project aimed at classifying mushrooms as edible or poisonous. A well-defined HLD is crucial for several reasons:

## Clear Model for Coding: This document provides a detailed roadmap for translating the project's goals into functional code. By outlining the chosen algorithms, data processing steps, and system architecture, it ensures a clear understanding for developers during the implementation phase.

## Early Detection of Issues: By defining the system at a high level, inconsistencies or potential problems in the design can be identified and addressed before coding begins. This saves time and effort compared to encountering issues later in the development cycle.

## Communication and Reference: The HLD serves as a reference manual for developers and other stakeholders involved in the project. It fosters clear communication by establishing a shared understanding of how different modules interact within the system.

## 1.2 Scope

This HLD will encompass the following aspects:

* Detailed Design: A breakdown of all design choices, including the chosen machine learning algorithms, data pre-processing techniques, and feature engineering methods.
* User Interface (UI): A description of the user interface for interacting with the model, if applicable (e.g., data input or prediction display).
* System Interfaces: A specification of any external software or hardware dependencies required for the model's operation.
* Performance Requirements: Definition of the expected performance metrics, such as prediction accuracy, processing speed, and resource usage.
* Project Architecture: A high-level overview of the system's overall architecture, including the interaction between different components.
* Non-Functional Requirements: A discussion of the project's non-functional attributes, including security, reliability, maintainability, portability, reusability, application compatibility, resource utilization, and serviceability.

By outlining these components, the HLD provides a comprehensive roadmap for developing a robust and well-designed machine learning system for mushroom classification.

### 1.3 Definitions

|  |  |
| --- | --- |
| Term | Description |
| Mushroom Species | Category of mushroom being classified (e.g., Agaricus bisporus, Lepiota cristata) |
| Descriptive Feature | A characteristic used to describe the mushroom (e.g., cap shape, color, gill spacing) |
| Edible | Classification of a mushroom as safe for consumption |
| Poisonous | Classification of a mushroom as harmful or deadly if consumed |
| Machine Learning Model | A computer program trained to identify patterns in data and use them for prediction |
| Training Data | Dataset used to train the machine learning model |
| Testing Data | Dataset used to evaluate the performance of the machine learning model |
| Prediction | The model's output indicating whether a new mushroom is likely edible or poisonous |

# 2 General Description

## 2.1 Product Perspective

### The Mushroom Classification System is an automated tool that utilizes machine learning algorithms to analyse descriptive features of mushrooms and predict their edibility (edible or poisonous). This system is designed to:

### Assist with mushroom identification: By providing predictions based on a mushroom's characteristics, the system can be a helpful tool for mushroom enthusiasts, researchers, or anyone interested in learning more about mushrooms.

### Enhance safety: The system can potentially help identify potentially poisonous mushrooms, reducing the risk of accidental consumption.

### 2.2 Problem statement

### This project aims to develop a machine learning system for classifying mushrooms as edible or poisonous. This system will leverage a dataset containing descriptions of various mushroom species and their edibility labels.

### Benefits of doing mushroom classification:-

### Enhanced Mushroom Identification: This system can be a valuable tool for mushroom enthusiasts, researchers, and anyone interested in learning more about mushrooms.

### Safety Awareness: By accurately classifying mushrooms, the system can potentially help reduce the risk of accidental consumption of poisonous mushrooms.

### 2.3 PROPOSED SOLUTION

### This project tackles mushroom classification with machine learning. By analysing features like cap shape and color from a dataset, the system predicts edibility using trained models. Data cleaning and feature engineering prepare the information for models like Support Vector Machines. The best model, chosen based on accuracy in distinguishing edible from poisonous mushrooms, could be deployed in a user interface (optional). While this tool can aid mushroom identification and safety, it remains a supportive resource. Consulting a reliable field guide or expert is always essential before consuming any mushroom.

### 2.4 FURTHER IMPROVEMENTS

### This project is just the start! Future improvements include image recognition for better accuracy and using environmental sensor data from the mushroom's location. We could explore advanced AI techniques and even develop a mobile app for on-the-spot mushroom identification. These advancements can make the system a powerful tool for mushroom enthusiasts and researchers alike.

### 2.5 Technical Requirements

##### This project focuses on building a machine learning system for classifying mushrooms as edible or poisonous using the following tools:

##### Data Acquisition: Secure a dataset containing descriptions of various mushroom species and their edibility labels (e.g., The Audubon Society Field Guide). Tools for user-generated data collection (descriptions and pictures) can be explored for future enhancements.

##### Computing Power: A computer with sufficient processing power is needed to train and run the machine learning model using scikit-learn. Cloud computing platforms can be considered depending on project needs.

##### Software Tools: Python will be the primary programming language for the project. The core development will utilize scikit-learn, a popular machine learning library in Python, for model building and evaluation.

##### User Interface: Streamlit, a Python framework for building user interfaces (UIs) quickly, will be used to create a user-friendly web app for interacting with the classification model.

##### These requirements provide the foundation for building the machine learning system for mushroom classification with a user-friendly interface using scikit-learn and Streamlit.

\------------------------------------------------------start here--------------------------------------\

##### 2.6 Data Requirements

Data requirement completely depend on our problem statement.

* We need images data that is balanced and must have at least 1000 images.
* We require at least 30- 40 images for each class label with annotation.
* An image is nothing more than a two-dimensional array of numbers(pixels)
* Pixel value ranging between 0 to 255
* It is defined by the mathematical function (x, y), the value off (x, y) at any point is giving the pixel value at that point of an image
* Original image is in the format of (width, height, no of RGB channels).

There are numerous image file formats out there so it can be hard to know which file type best suits your image needs (on your requirement).

* TIFF — Tagged image file format O BMP — Bitmap image file form o JPEG - Joint photographic experts' groups O GIF - graphics interchange format

O PNG — portable network graphics O EPS — encapsulated post script o RAW image files

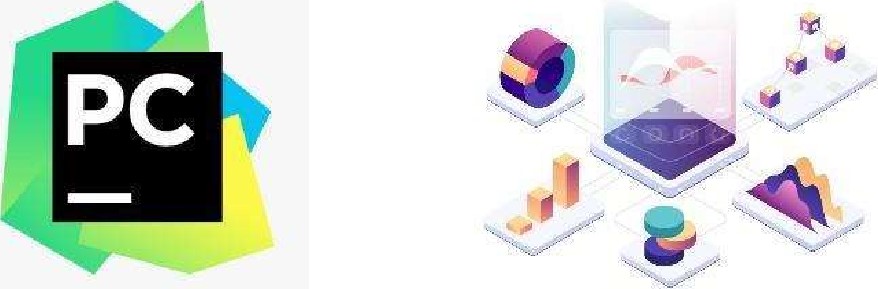
* Tiffs are great for printing. These are lossless image files meaning they don't need to compress or lose any image quality or information. These format images are high quality images.
* bmp format developed by Microsoft for windows. There is no compression or information loss; this format is generally recommended for high quality scans.  JPEG is a lossy format meaning that the image is compressed to make a smaller file but this loss is not noticeable.
* JPEG is a very popular format for digital cameras.
* GIFS are widely used for web graphics because they are limited to only 256 colours, can allow for transparency and can be animated. These types of files are typically small in size and very portable.
* PNG are a lossless image format; these files are able to handle up to 16 million colours unlike the 256 colours supported by GIF.
* EPS is a common vector type file.
* RAW images that are unprocessed that have been created by a camera or scanner. Digital cameras can shoot in raw, mostly used in photography.

If the data is in video format like (MP4) convert into images based on FPS (no. of frames displayed per second) in real time processing. There are number of tools to convert videos into images. Using cv we can convert video into images

#### 2.7 Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, TensorFlow, Keras and Roboflow are used to build the whole model.

TensorFlow 

 django

 z: ROS e

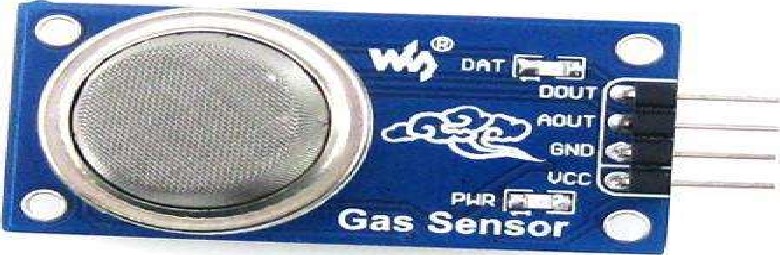
GAZEBO



* PyCharm is used as IDE.
* For visualization of the plots, Matplotlib, Seaborn and Plotly are used.
* AWS is used for deployment of the model.
* Tableau/Power Bl is used for dashboard creation.
* MySQL/MongoDB is used to retrieve, insert, delete, and update the database.  Front end development is done using HTML/CSS  Python Django is used for backend development.  GitHub is used as version control system.

2.7.1 Hardware Requirements

* USB Camera for object Detection
* LM35 temperature sensor
* MQ Smoke Detector Sensor
* PC (check you are system supports: https://7dfps.com/ros-systemrequirements/)
* HC-SR04 Ultrasonic sensor
* Ground vehicle
* Raspberry Pi

 Vcc (SV) %round 



2.7.2 ROS (Robotic Operating System)

Robot Operating System is an open-source robotics middleware suite. Although ROS is not an operating system but a collection of software frameworks for robot software development, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management.

##### 2.8 Constraints

The UGV based Surveillance solution system must be user friendly, as automated as possible and users should not be required to know any of the workings.

##### 2.9 Assumptions

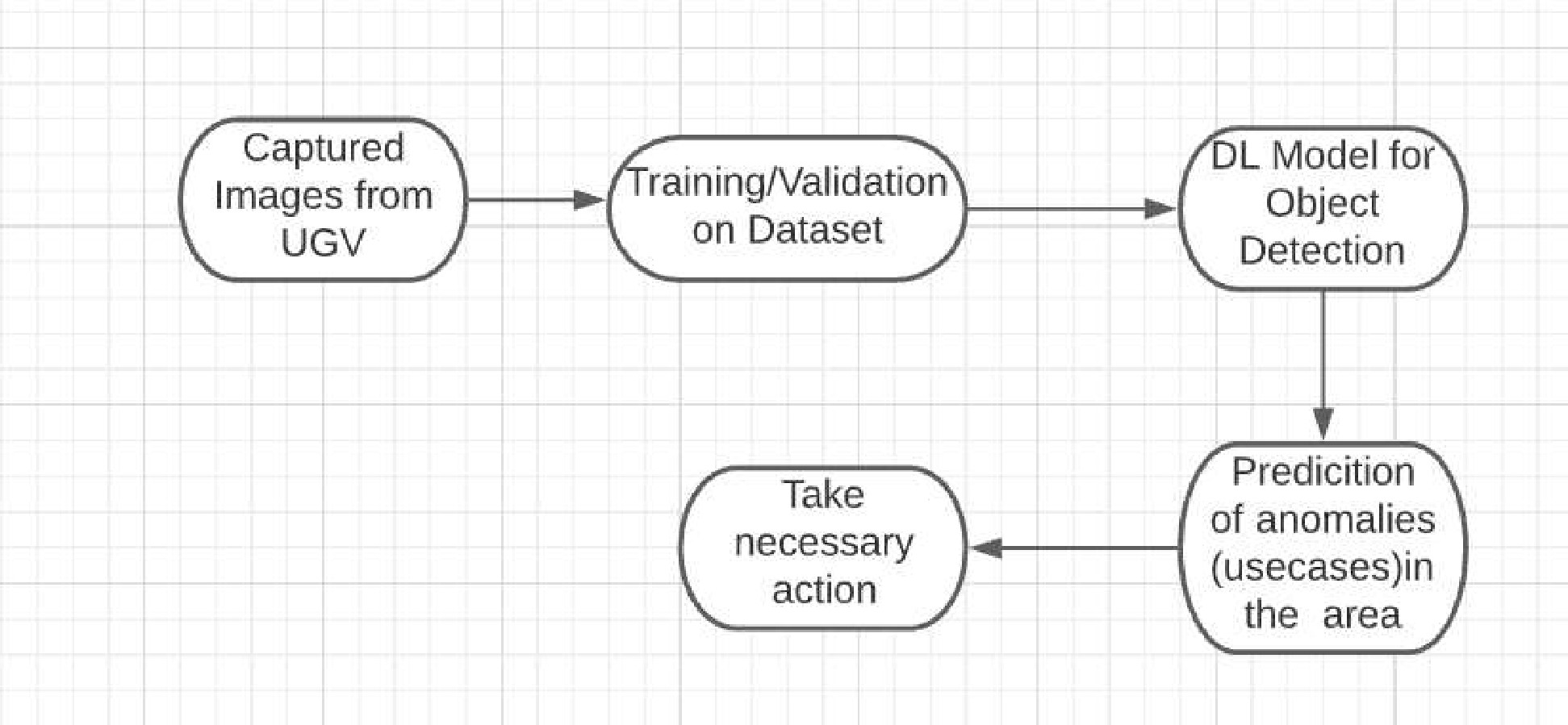
The main objective of the project is to implement the use cases as previously mentioned (2.2 Problem Statement) for new dataset that comes through UGV vehicle which has camera installed for capturing the live videos. Deep Learning based object detection model is used for detecting the above-mentioned use cases based on the input data. It is also assumed that all aspects of this project have the ability to work together in thevwthe designer is expecting.

1. Design Details

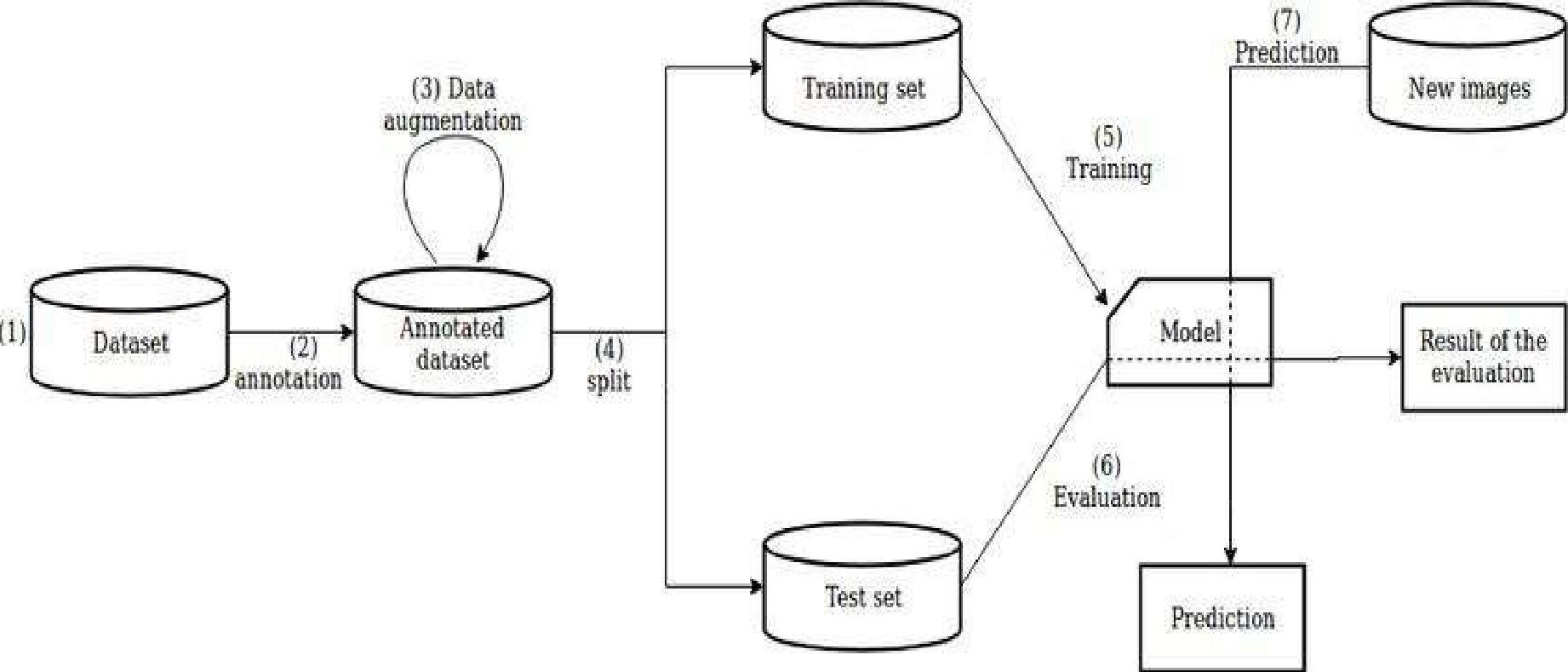
3.1 Process Flow

For identifying the different types of anomalies, we will use a deep learning base model. Below is the process flow diagram is as shown below.

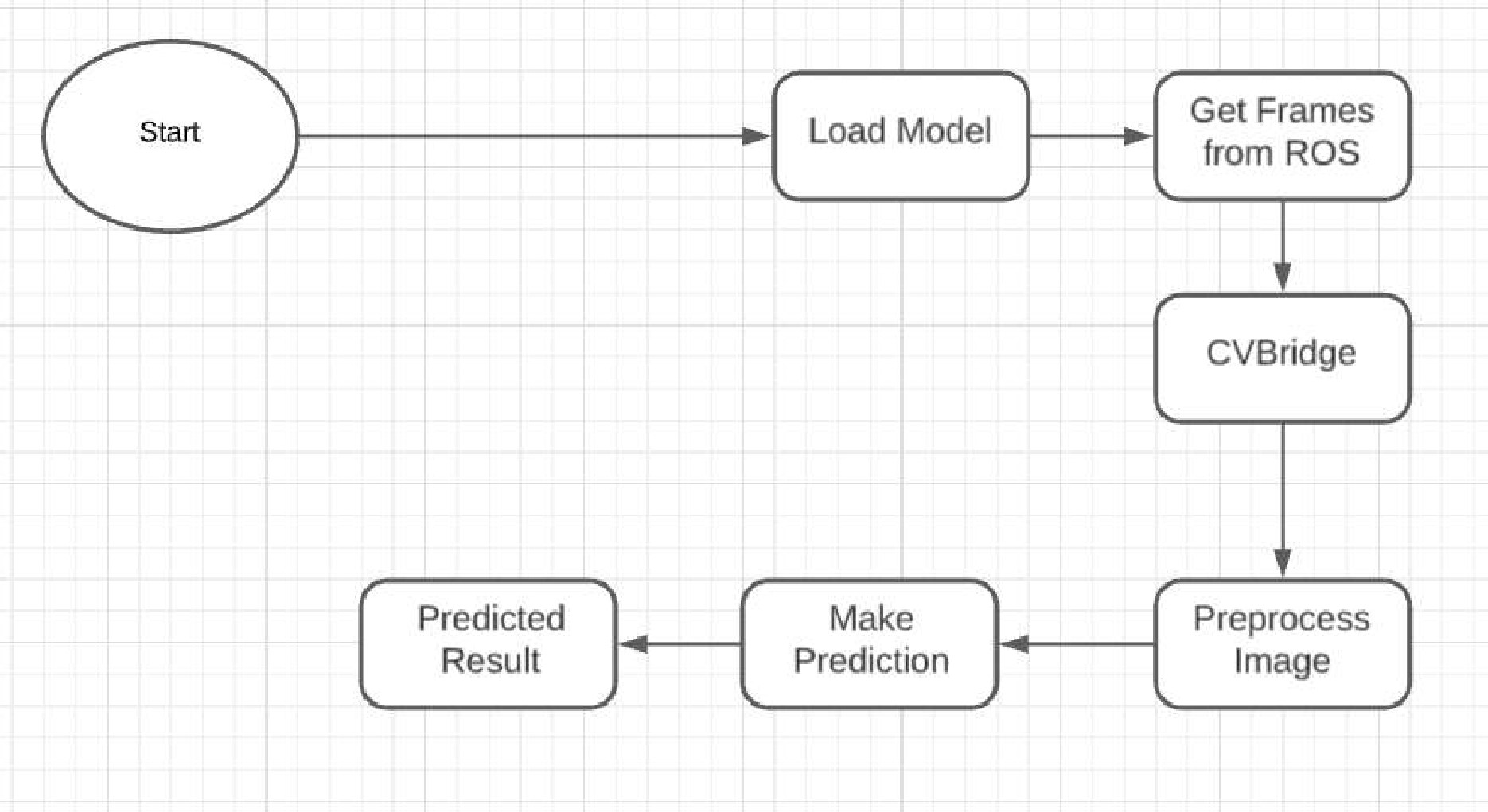
#### Proposed methodology



##### 3.1.1 Model Training and Evaluation



##### 3.1.2 Deployment Process



#### 3.2 Event log

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

#### 3.3 Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

12

Performance

The UGV based surveillance solution is used for detection of anomalies in the society whenever UGV detects any anomalies (mob, medical emergency, fire, smoke, etc...) it will inform concern authorities and takes necessary action, so it should be as accurate as possible. So that it will not mislead the concern authorities (like hospitals, cops, etc..). Also, model retraining is very important to improve the performance.

#### 4.1 Reusability

The code written and the components used should have the ability to be reused with no problems.

#### 4.2 Application Compatibility

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

#### 4.3 Resource Utilization

When any task is performed, it will likely use all the processing power available until that

function is finished.

4.4 Deployment

Microsoft

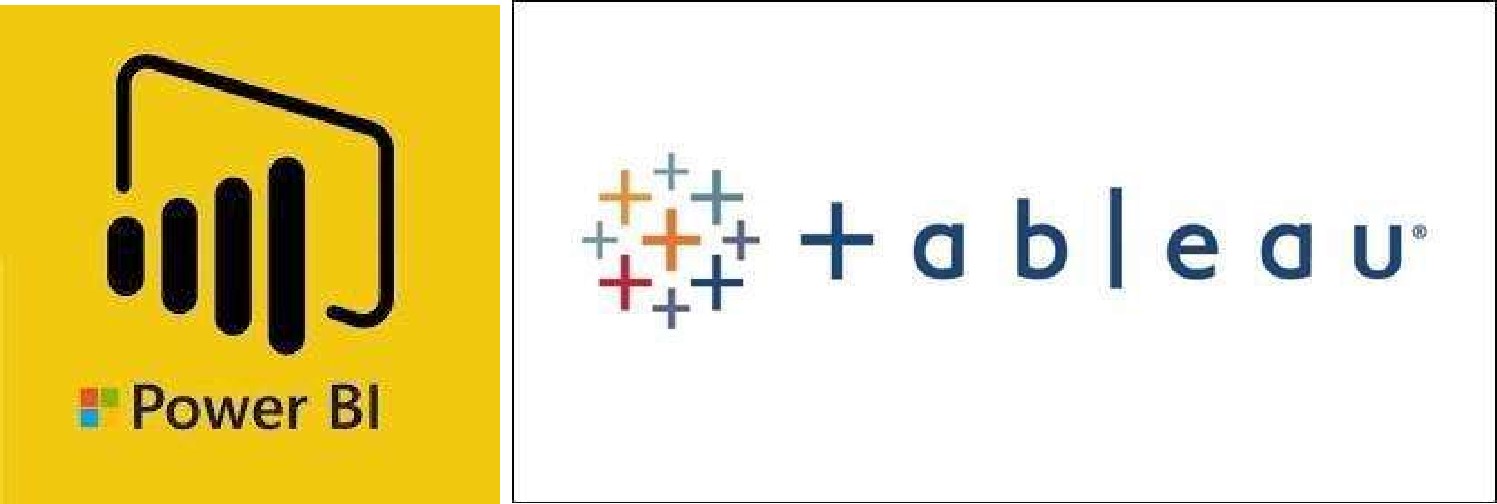
Azurewebservicesruamazon

Google Cloud

13

Dashboards

Dashboards will be implemented to display and indicate certain KPls and relevant indicators for the unveiled problems that if not addressed in time could cause catastrophes of unimaginable impact.



As and when, the system starts to capture the historical/periodic data for a user, the dashboards will be included to display charts over time with progress on various indicators or factors.

5.1 KPIs (Key Performance Indicators)

1 . Key indicators displaying a summary of the anomaly detection in the society/area.

1. Time and workload reduction using the UGV based surveillance.
2. To detect mob (illegal) activities and inform police.
3. On time alert to nearest hospital on medical emergency (accident).
4. Taking adequate evidence of mob.
5. Send disaster details to concerned authorities.
6. Display of battery life and percentage of UGV.
7. Distance travelled by UGV.
8. Get the exact location of UGV.

Conclusion

The Designed UGV (Unmanned Ground Vehicle) will detect an anomaly in the locality based on various anomalies data used to train our algorithm, so we can identify the imbalance in the society in early stages and can take necessary action to stop them immediately, so we can have a pleasant environment in that area or location.

15

References

1 . https://en.wikipedia.orq/wiki/Unmanned qround vehicle

1. Google.com for images of UGV hardware.
2. https://www.ros.org/

16