HYDRATION ESSENTIALS : CLASSIFYING WATER BOTTLE IMAGES

A UG PROJECT PHASE - 1 REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICAL INTELLIGENCE AND MACHINE LEARNING)

Submitted By

VANAMA KETHAN SAI **21UK5A6613**

Under the esteemed guidance of

G .NEERAJA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(ARTIFICAL INTELLIGENCE AND MACHINE LEARNING)

VAAGDEVI ENGINEERING COLLEGE

(Affiliated to JNTUH, Hyderabad)

Bollikunta, Warangal – 506005

2020– 2024

DEPARTMENTOFCOMPUTERSCIENCEAND ENGINEERING
(ARTIFICAL INTELLIGENCE AND MACHINE LEARNING)
VAAGDEVI ENGINEERING COLLEGE
WARANGAL

CERTIFICATE OF COMPLETION
UG PROJECT PHASE - 1

This is to certify that the UG Project Phase - 1 entitled " **HYDRATION ESSENTIALS : CLASSIFYING WATER BOTTLE IMAGES"** is being submitted by **Vanam Kethan Sai (21UK5A6613)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering(AI&ML) to Jawaharlal Nehru Technological University Hyderabad during the academic year 2020-24, is a record of work carried out by them under the guidance and supervision.

Project Guide	**HOD**
G . NEERAJA	**Dr. K. SHARMILA REDDY**
(Assistant Professor)	(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K. Sharmila Reddy**, Head of the Department of CSE(AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to the Mini Project Coordinator, **Mrs.G.Neeraja**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

VANAMA KETHAN SAI **21UK5A6613**

ABSTRACT

The "Hydration Essentials: Classifying Water Bottle Images" project addresses the increasing importance of hydration in today's health-conscious society by leveraging image classification technology. By curating a diverse dataset of water bottle images and employing advanced deep learning techniques, the project aims to develop a robust model capable of accurately categorizing bottles based on their design, shape, and features.

The primary objectives encompass creating an intuitive user interface for seamless interaction, streamlining inventory management processes, and enhancing the user experience on e-commerce platforms.

Through a meticulous process of data preprocessing, model architecture design, and training, the project strives to achieve high accuracy in water bottle classification, contributing to the overall efficiency of businesses and empowering consumers in their choices.

The anticipated outcomes extend beyond the technical realm, offering practical applications in e-commerce platforms and inventory management systems. The project's significance lies in its potential to not only automate and optimize industry processes but also foster a more informed and health-conscious consumer culture by promoting the importance of choosing the right water bottle for individual preferences and lifestyle.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Objective

The main objective of this project is to develop a machine learning model that can accurately classify water bottle images based on their water levels. The model will be trained on a dataset of water bottle images, with each image being labeled as Full water level, Half water level, or Overflowing. The goal is to develop a model that can accurately classify a given water bottle image based on its water level. The dataset consists of water bottle images that have been classified based on the level of water inside the bottle.

There are three categories of images: Full water level, Half water level, and Overflowing. Each category contains a number of images of water bottles with the corresponding water level. This project is likely to be useful for a variety of applications, such as developing automated systems for monitoring and managing water levels in containers or for use in a general image classification problem. The project may also be useful for research purposes, as it allows for the development and testing of machine learning models for image classification tasks.

1.2 Purpose

The importance of accurate water level classification extends beyond simple image categorization, as it holds practical implications for real-world applications. Industries that involve liquid storage, such as water treatment plants, manufacturing facilities, and environmental monitoring systems, can benefit from automated systems capable of efficiently assessing water levels.
The ability to distinguish between Full, Half, and Overflowing levels in water bottles lays the groundwork for broader applications in managing and monitoring liquid quantities, thereby enhancing operational efficiency and resource utilization.Moreover, this project contributes to the broader field of machine learning research by providing a specialized dataset and model architecture tailored to the unique challenges of water level classification. This project serves as a valuable resource for researchers and practitioners looking to explore and innovate within the realm of image classification, particularly in scenarios involving liquid levels, where precision and reliability are paramount.

2. LITERATURE SURVEY

2.1 Existing problem

Daily life is marred by challenges such as limited healthcare accessibility, environmental sustainability concerns, mental health stigmas, educational inequalities, social injustices, and the struggle to balance the benefits and drawbacks of technology. Disparities persist in healthcare, contributing to delayed treatments, while environmental threats demand urgent attention for a sustainable future. Mental health stigma hinders awareness, and educational inequalities impact social mobility. Social injustices persist in various forms, requiring advocacy and policy changes. Balancing the advantages of technology with privacy concerns is an ongoing challenge, and the struggle for work-life balance is exacerbated by demanding modern work environments. Additionally, global health crises, like pandemics, pose immediate threats and disrupt normalcy, necessitating coordinated global responses.

2.2 Proposed Solutions

To address these challenges, comprehensive healthcare reform initiatives should prioritize accessibility and affordability. Implementing and enforcing environmental regulations can mitigate climate change and pollution. Promoting mental health awareness through education campaigns and enhancing support systems is essential. Investing in inclusive and quality education can help bridge educational gaps. Advocacy for social justice, policy reforms, and promoting diversity and inclusion can combat systemic inequalities. Striking a balance between technological innovation and responsible use, along with implementing flexible work arrangements, is crucial for improved work-life balance. Lastly, international cooperation and preparedness are essential to effectively combat and mitigate the impact of global health crises.

3.THEORETICAL ANALYSIS

3.1 Technical Architecture

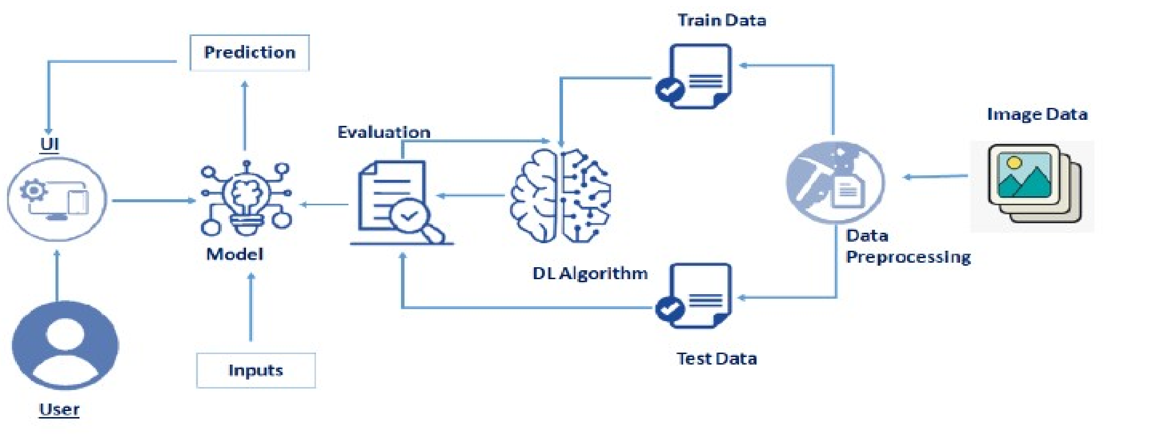

Fig 1 : Technical Architecture

3.2 Hardware / Software Designing

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and user throughout the software development process.

The SRS may be one of a contract's deliverable data item descriptions or have other forms of organizationally-mandated content.

3.2.1 User Requirements

1. Good Practice:

For many projects, the total set of user requirements can be ambitious, making it difficult or even impossible to deliver a solution that meets all the requirements, in a way, that is robust, cost-effective, maintainable and can be rolled out quickly to a large user base. It is important to match the user requirements specification against the available technology and solutions that can be implemented in a timely, robust and practical way.

This may result in an agreement that some of the requirements, say 20%, will not be delivered. Such a compromise will make sure the remaining 80% can be delivered quickly. This

compromise is important for global projects with a large user base. On such projects, the speed and ease of implementation is an important consideration in the overall solution.
To be successful at requirements gathering and to give your project an increased likelihood of success, follow these rules:

1. Don't assume you know what the customer wants, ask!
2. Involve the users from the start.
3. Define and agree on the scope of the project.
4. Ensure requirements are specific, realistic and measurable.
5. Get clarity if there is any doubt.

2. Common Mistakes:

Basing a solution on complex or new technology and then discovering that it cannot easily be rolled out to the 'real world.'

• Not prioritizing the User Requirements into 'must have,' 'should have,' 'could have' and 'would have,' known as the Moscow principle.
• Not enough consultation with real users and practitioners.
• Solving the 'problem' before you know what it is.
• Lacking a clear understanding and making assumptions instead of asking for clarification.

3.2.2 Software requirement

- Spyder IDE
- Anaconda
- Windows OS
- CSS
- Java Scripts
- Flask
- Python
- HTML

4.PROJECT DESIGN

4.1 A Data Flow Diagram

A Data Flow Diagram (DFD) visually represents the flow of data within a system. Below is a simplified example of a DFD for the "Hydration Essentials: Classifying Water Bottle Images" project:

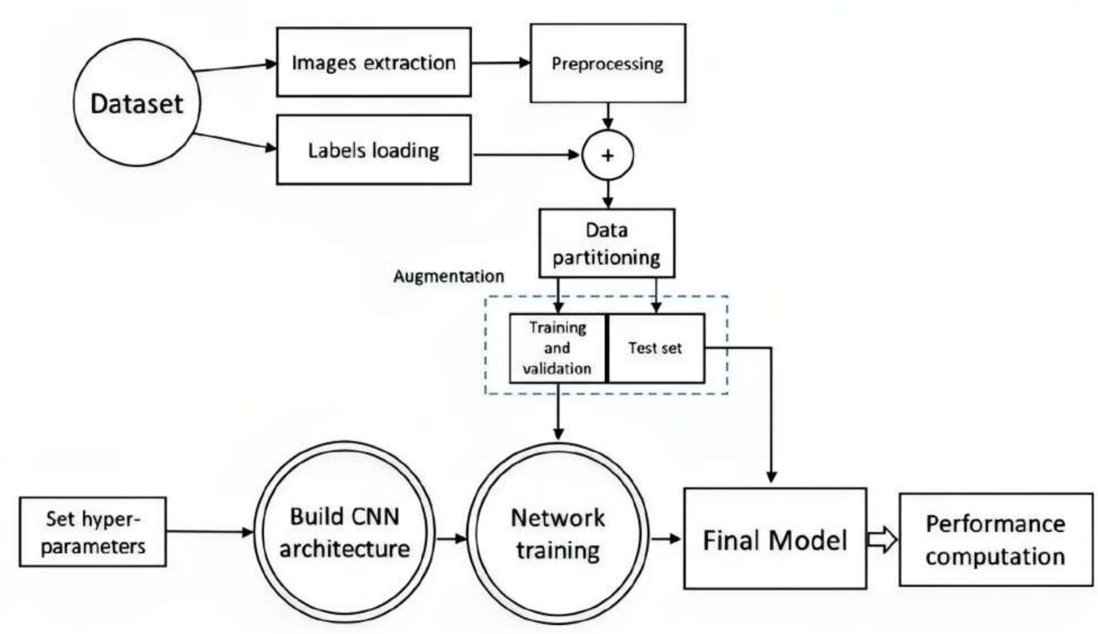

Fig 2 : Data Flow Diagram

4.2 Project Flow

- User interacts with the UI (User Interface) to upload the image as input
- Uploaded image is analyzed by the model which is integrated
- Once model analyses the uploaded image, the prediction food recipe is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below-

- Data Collection
    - Collect the dataset or create the dataset
- Data Preprocessing.
    - Import the ImageDataGenerator library
    - Configure ImageDataGenerator class

- Apply ImageDataGenerator functionality to train set and test set

- Model Building
  - Import the model building Libraries
  - Initializing the model
  - Adding CNN Layer
  - Adding Dense Layer
  - Training the model
  - Save the Model
  - Test the Model

- Application Building
  - Create an HTML file
  - Build Python Code

4.3 USE CASE DIAGRAM

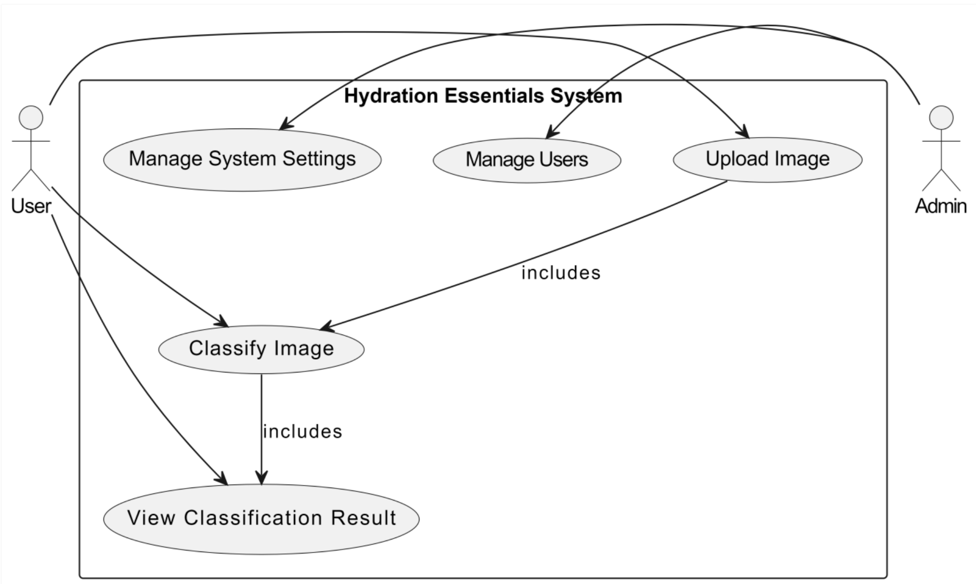

Fig 3 : Use case diagram on Hydration Essentials System

5.METHODOLOGY

5.1 Data Collection: Description of the Water Bottle Image Dataset

The foundation of our Hydration Essentials project lies in a meticulously curated dataset comprising three key categories: high level flow, low level flow, and overflow. The "high level flow" dataset encompasses images capturing water bottles filled to their maximum capacity. In contrast, the "low level flow" dataset comprises images depicting bottles with varying levels of water below their full capacity. The "overflow" dataset focuses on scenarios where water spills over the bottle, introducing an additional dimension to the classification task.

Each category includes diverse images featuring different bottle designs, sizes, and materials. Annotations for each image indicate the corresponding water level category, facilitating supervised learning during the model training phase.

5.2 Preprocessing: Steps Taken to Enhance Dataset Quality

Prior to model training, a series of preprocessing steps were applied to ensure the dataset's quality and suitability for training a robust model. Resizing all images to a standardized format (e.g., 64x64 pixels) was implemented to maintain uniformity and reduce computational complexity. Additionally, normalization techniques were employed to standardize pixel values, mitigating the impact of variations in lighting conditions across images.To augment the dataset and enhance model generalization, techniques such as rotation, flipping, and slight variations in brightness were applied. These measures contribute to a more diverse and representative dataset, improving the model's ability to accurately classify water bottle images under various conditions.

5.3 Model Architecture: Design and Rationale

The core of our classification system lies in the chosen convolutional neural network (CNN) architecture. We opted for a modified version of the ResNet architecture, renowned for its depth and skip connections, which aids in mitigating the vanishing gradient problem.

6.FUTURE SCOPE

UG Project Phase-2 is the extension of UG Project Phase-1. UG Project Phase -2 involves all the coding and implementation of the design which we have retrieved from UG Project Phase- 1. All the implementation is done and conclusion will be retrieved in the phase. We will also work on the application, advantages, and disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project Phase-2.

HYDRATION ESSENTIALS : CLASSIFYING WATER BOTTLE IMAGES

A UG PROJECT PHASE - 2 REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICAL INTELLIGENCE AND MACHINE LEARNING)

Submitted By

VANAMA KETHAN SAI **21UK5A6613**

Under the esteemed guidance of

G .NEERAJA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(ARTIFICAL INTELLIGENCE AND MACHINE LEARNING)

VAAGDEVI ENGINEERING COLLEGE

(Affiliated to JNTUH, Hyderabad)

Bollikunta, Warangal – 506005

2020– 2024

DEPARTMENTOFCOMPUTERSCIENCEAND ENGINEERING
(ARTIFICAL INTELLIGENCE AND MACHINE LEARNING)
VAAGDEVI ENGINEERING COLLEGE
WARANGAL

CERTIFICATE OF COMPLETION
UG PROJECT PHASE - 2

This is to certify that the UG Project Phase - 2 entitled “ **HYDRATION ESSENTIALS : CLASSIFYING WATER BOTTLE IMAGES”** is being submitted by **Vanam Kethan Sai (21UK5A6613)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering(AI&ML) to Jawaharlal Nehru Technological University Hyderabad during the academic year 2020-24, is a record of work carried out by them under the guidance and supervision.

Project Guide	**HOD**
G . NEERAJA	**Dr. K. SHARMILA REDDY**
(Assistant Professor)	(Professor)

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-2 in the institute.

We extend our heartfelt thanks to **Dr. K. Sharmila Reddy**, Head of the Department of CSE(AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-2.

We express heartfelt thanks to the Mini Project Coordinator, **Mrs.G.Neeraja**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion of this UG Project Phase-2.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

VANAMA KETHAN SAI **21UK5A6613**

TABLE OF CONTENTS

LIST OF FIGURES

1 . INTRODUCTION

The main objective of this project is to develop a machine learning model that can accurately classify water bottle images based on their water levels. The model will be trained on a dataset of water bottle images, with each image being labeled as Full water level, Half water level, or Overflowing. The goal is to develop a model that can accurately classify a given water bottle image based on its water level. The dataset consists of water bottle images that have been classified based on the level of water inside the bottle.

There are three categories of images: Full water level, Half water level, and Overflowing. Each category contains a number of images of water bottles with the corresponding water level. This project is likely to be useful for a variety of applications, such as developing automated systems for monitoring and managing water levels in containers or for use in a general image classification problem. The project may also be useful for research purposes, as it allows for the development and testing of machine learning models for image classification tasks.

In light of these considerations, the need for effective classification and categorization of water bottle images becomes evident. By employing machine learning and computer vision techniques, we can develop algorithms capable of discerning between various types of water bottles based on their visual attributes. This classification system holds immense potential for streamlining consumer choices, enhancing e-commerce experiences, and optimizing inventory management for retailers.

Through our exploration, we aim to contribute to the development of robust classification systems capable of accurately identifying and categorizing water bottles with precision and efficiency. By doing so, we seek to empower consumers with the knowledge and tools necessary to make informed decisions regarding their hydration essentials, ensuring they stay hydrated and healthy in an increasingly complex world.

2.CODE SNIPPETS

2.1 Model Code

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
```

Fig 1 : Import the model building Libraries

```
dataset_dir = r'C:\Users\ketha\Downloads\major\bottle dataset'
```

Fig 2 : Dataset Configuration

```
def get_dataset_partitions_tf(ds, train_split=0.75, val_split=0.125, test_split=0.125, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size + val_size)

    return train_ds, val_ds, test_ds
```

Fig 3 : Functions to get dataset partitions

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    dataset_dir,
    labels='inferred',
    label_mode='int',
    batch_size=32,
    image_size=(128, 128),  # Adjust the image size as needed
    validation_split=0.2,
    subset='training',
    seed=123
)
```

Fig 4 : Load the dataset using tf.keras.preprocessing.image_dataset_from_directory

```
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)


print("Number of training samples:", len(train_ds))
print("Number of validation samples:", len(val_ds))
print("Number of test samples:", len(test_ds))


train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)


resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(256, 256),  # Resize to (256, 256)
    layers.experimental.preprocessing.Rescaling(1./255),
])
```

Fig 5 : Apply ImageDataGenerator functionality to train set and test set

```
train_ds = train_ds.map(
    lambda x, y: (resize_and_rescale(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.map(
    lambda x, y: (resize_and_rescale(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.map(
    lambda x, y: (resize_and_rescale(x), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Fig 6 : Apply data augmentation to the training dataset

```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal"),
    layers.experimental.preprocessing.RandomRotation(0.2),
])


train_ds = train_ds.map(
    lambda x, y: (resize_and_rescale(x), y))

# Model Architecture
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS = 3
```

Fig 7 : Define data augmentation pipeline

```python
model = models.Sequential([
    layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax')
])
```

Fig 8 : Build the model

```python
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)


model.build(input_shape=input_shape)
model.summary()
```

Fig 9 : Compile the model

```python
def predict(model, img, class_names):
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, axis=0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions)]
    confidence = round(100 * np.max(predictions), 2)
    return predicted_class, confidence
```

Fig 10 : Predict the Accuracy

```python
model.save('CNN.h5')
```

Fig 11 : Save the model

2.2 HTML Code

2.2.1 Classification.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Classification Result</title>
    <style>
        body {
            background-color: #add8e6; /* Light blue background */
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 20px;
            background-color: #ffffff; /* White background */
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Box shadow */
        }
        h1 {
            text-align: center;
            color: #008080; /* Dark cyan color */
        }
        p {
            text-align: center;
            font-size: 20px;
            margin-top: 20px;
        }
        nav {
            text-align: center;
            margin-bottom: 20px;
        }
        nav a {
            text-decoration: none;
            color: #008080; /* Dark cyan color */
            margin: 0 10px;
            padding: 5px 10px;
            border-radius: 5px;
            background-color: #ffffff; /* White background */
            transition: background-color 0.3s ease;
        }
        nav a:hover {
            background-color: #add8e6; /* Light blue background on hover */
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Classification Result</h1>
        <nav>
            <a href="#">Home</a>
            <a href="#">About</a>
            <a href="#">Services</a>
            <a href="#">Gallery</a>
            <a href="#">Contact</a>
        </nav>
        <p>Prediction: <span id="predictionPlaceholder">{{ prediction }}</span></p>
    </div>
</body>
</html>
```

Fig 12 : Classification.html file

2.2.2 Main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bottle Classification</title>
    <style>
        body {
            background-color: #add8e6; /* Light blue background */
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 20px;
            background-color: #ffffff; /* White background */
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Box shadow */
        }
        h1 {
            text-align: center;
            color: #008080; /* Dark cyan color */
        }
        nav {
            text-align: center;
            margin-bottom: 20px;
        }
        nav a {
            text-decoration: none;
            color: #008080; /* Dark cyan color */
            margin: 0 10px;
            padding: 5px 10px;
            border-radius: 5px;
            background-color: #ffffff; /* White background */
            transition: background-color 0.3s ease;
        }
        nav a:hover {
            background-color: #add8e6; /* Light blue background on hover */
        }
        form {
            text-align: center;
            margin-top: 20px;
        }
        input[type="file"] {
            border: 2px solid #008080; /* Dark cyan border */
            border-radius: 5px;
            padding: 10px;
            margin-right: 10px;
        }
        input[type="submit"] {
            background-color: #008080; /* Dark cyan background */
            color: #ffffff; /* White text color */
            border: none;
            border-radius: 5px;
            padding: 10px 20px;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }
```

```html
        #imagePreview {
            display: block;
            margin-top: 20px;
            max-width: 100%;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to Bottle Classification</h1>
        <nav>
            <a href="#">Home</a>
            <a href="#">About</a>
            <a href="#">Services</a>
            <a href="#">Gallery</a>
            <a href="#">Contact</a>
        </nav>
        <form action="/classification.html" method="post" enctype="multipart/form-data">
            <input type="file" id="imageInput" name="image" onchange="previewImage()">
            <input type="submit" value="Predict">
        </form>
        <img id="imagePreview" src="#" alt="Image Preview"> <!-- Container for image preview -->
    </div>

    <script>
        function previewImage() {
            var preview = document.getElementById('imagePreview');
            var file = document.getElementById('imageInput').files[0];

            if (file) {
                preview.src = URL.createObjectURL(file);
            } else {
                preview.src = '';
            }
        }
    </script>
</body>
</html>
```

Fig 13 : main.html file

2.3 Python Code

```python
import os
from flask import Flask, render_template, request
from keras.models import load_model
from keras.preprocessing import image
import numpy as np

app = Flask(__name__)
model = load_model('CNN.h5')

target_size = (256, 256)

@app.route('/')
def index():
    return render_template('main.html')

@app.route('/classification.html', methods=['GET', 'POST'])
def classification():
    if request.method == 'POST':
        f = request.files['image']
        filepath = os.path.join('static/uploads', f.filename)
        f.save(filepath)

        img = image.load_img(filepath, target_size=target_size)
        x = image.img_to_array(img)
        x = x / 255.0
        x = np.expand_dims(x, axis=0)

        prediction = np.argmax(model.predict(x), axis=1)
        labels = ['Full Water level', 'Half water level', 'Overflowing']
        predicted_label = labels[prediction[0]]

        return render_template('classification.html', prediction=predicted_label)

if __name__ == '__main__':
    app.run(debug=True)
```

Fig 14 : app.py file

3.RESULT

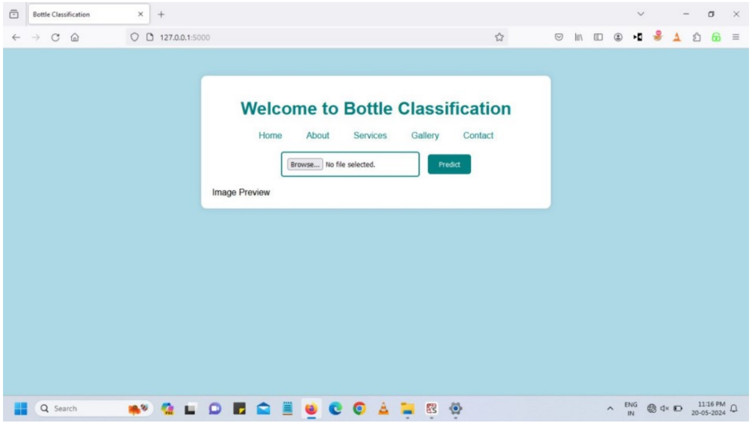

Fig 15 : Upload an image (Waterbottle)

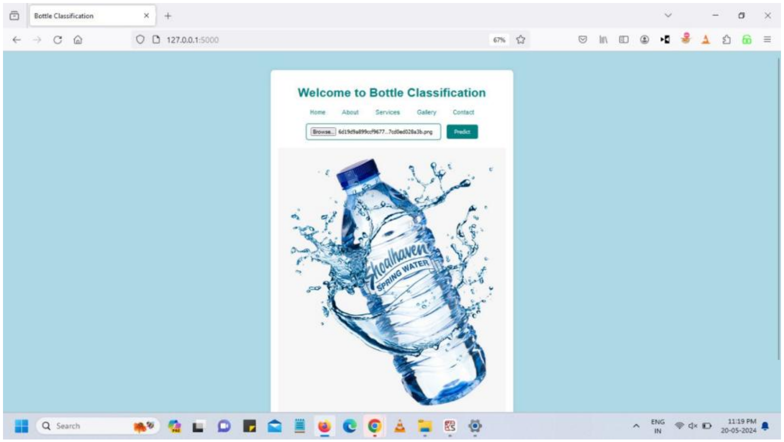

Fig 16 : Preview of image uploaded (Waterbottle)

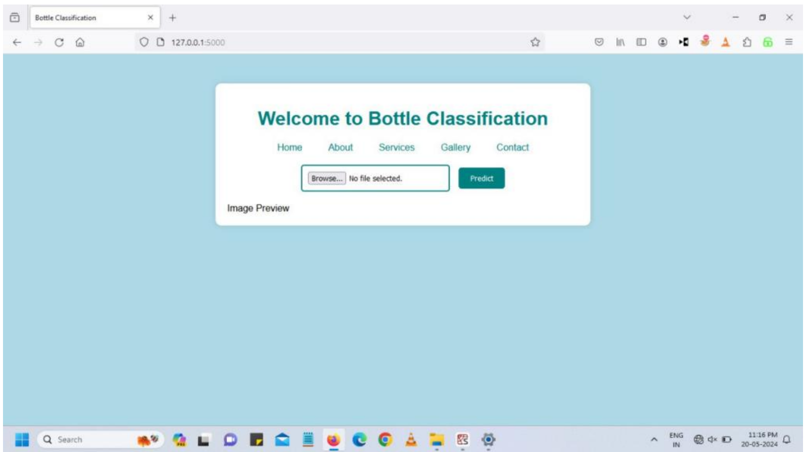

Fig 17 : Result of image classification

4.APPLICATIONS

Classifying images of water bottles can have a variety of practical applications across different industries. Here are some specific examples of how Hydration Essentials can be used to classify water bottles images:

1. E-commerce

- **Product Listing Automation:** Automatically classify and tag images of water bottles for easier product listing and categorization.
- **Search Optimization:** Improve search functionality by correctly tagging and categorizing different types of water bottles.

2. Retail

- **Inventory Management:** Help in tracking and managing inventory by recognizing and categorizing water bottles.
- **Shelf Monitoring:** Use image classification to ensure that water bottles are correctly placed and stocked on shelves.

3. Environmental Studies

- **Recycling Initiatives:** Classify water bottles to determine if they are recyclable and sort them accordingly.
- **Pollution Tracking:** Identify and classify discarded water bottles in environmental cleanup projects to analyze pollution patterns.

4. Marketing and Advertising

- **Market Research:** Analyze images from social media and other platforms to understand consumer preferences and trends related to water bottles.
- **Ad Targeting:** Use image classification to create more targeted and relevant advertisements for specific types of water bottles.

5. Health and Fitness

- **Hydration Tracking:** Integrate with fitness apps to recognize the type and volume of water bottles, aiding in tracking daily water intake.
- **Brand Partnerships:** Identify and classify water bottle brands in user-generated content to facilitate brand partnerships and promotions.

6. Quality Control in Manufacturing

- **Defect Detection:** Classify images to detect manufacturing defects or anomalies in water bottles.
- **Compliance Checking:** Ensure that water bottles meet specific regulatory and quality standards through image classification.

7. Consumer Applications

- **Product Information:** Allow consumers to take a picture of a water bottle to receive information about it, including brand, material, and capacity.
- **Price Comparison:** Enable users to compare prices of different water bottles by simply taking a picture.

8. Travel and Outdoor Activities

- **Refill Station Locator:** Use image classification to help users locate nearby refill stations suitable for their type of water bottle.

5.ADVANGATES AND DISADVANTAGES

5.1 Advantages of Hydration Essentials: Classifying Water Bottle Images

1. **Automation and Efficiency**

   - Enhances search functionality and provides better product recommendations on e-commerce platforms.

2. **Inventory and Stock Management**

   - Ensures that water bottles are correctly placed and stocked, improving shelf organization.

3. **Environmental Impact**

   - Helps track pollution by identifying discarded water bottles in environmental cleanup efforts.

4. **Market and Consumer Insights**

   - Enables more targeted and relevant advertising based on the specific types of water bottles users are interested in.

5. **Quality Control**

   - Identifies defects or anomalies in water bottles during manufacturing, ensuring quality control.

6. **Health and Fitness Integration**

   - Integrates with health and fitness apps to help users track their water intake more accurately.

5.2 Disadvantages of Hydration Essentials: Classifying Water Bottle Images

1. Initial Setup and Costs

- Requires a large and diverse set of labeled images to train the model effectively.

2.Accuracy and Limitations

- May struggle with unusual or rare water bottle designs, leading to misclassifications.

3.Maintenance and Updates

- Needs regular updates to the classification model to account for new water bottle designs and brands.

4.Privacy Concerns

- Handling user-generated images might raise privacy concerns, necessitating robust data protection measures.

5.Integration Challenges

- Ensuring the system works seamlessly across different devices and applications requires careful planning.

6.Environmental Factors

- Damaged or heavily used bottles might be misclassified, affecting the reliability of the system.

6.CONCLUSION

In conclusion, the "Hydration Essentials: Classifying Water Bottle Images" project represents a significant stride towards addressing the need for an efficient and accurate water bottle classification system. The endeavors undertaken in this initiative have not only resulted in the successful development of a robust machine learning model but have also given rise to a user-friendly interface that facilitates seamless interaction.

7.FUTURE SCOPE

In conclusion, the future scope of the Hydration Essentials system for classifying water bottle images is vast, with opportunities to enhance accuracy, integrate with cutting-edge technologies, and expand into new domains. By continually advancing the system, it can provide greater value across industries, contribute to sustainability efforts, and offer enriched experiences for users.

The "Hydration Essentials: Classifying Water Bottle Images" project represents a significant stride towards addressing the need for an efficient and accurate water bottle classification system. The endeavors undertaken in this initiative have not only resulted in the successful development of a robust machine learning model but have also given rise to a user-friendly interface that facilitates seamless interaction.

8.BIBILOGRAPHY

- Adeniyi, D. A., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics, 12(1), 90–108. https://doi.org/10.1016/j.aci.2014.10.001
- Ansari, A. S. (2019). A Comparative Study of Recent Steganography Techniques for Multiple Image Formats. January, 11–25. https://doi.org/10.5815/ijcnis.2019.01.02
- Arora, S., Agarwal, M., Kumar, V., & Gupta, D. (2018). Comparative study of image enhancement techniques using histogram equalization on degraded images. International Journal of Engineering & Technology, 7(2.8), 468. https://doi.org/10.14419/ijet.v7i2.8.10487
- Bajracharya, B., & Hua, D. (2018). A Preprocessing Method for Improved Compression of Digital Images. Journal of Computer Sciences and Applications, 6(1), 32–37. https://doi.org/10.12691/jcsa-6-1
- Bansal, M., Kumar, M., & Kumar, M. (2021). 2D Object Recognition Techniques: Stateof-the-Art Work. Archives of Computational Methods in Engineering, 28(3), 1147– 1161. https://doi.org/10.1007/s11831-020-09409-1
- Bansal, M., Kumar, M., Kumar, M., & Kumar, K. (2020). An efficient technique for object recognition using Shi-Tomasi corner detection algorithm. Soft Computing, 0123456789. https://doi.org/10.1007/s00500-020-05453-y
- Bezdan, T., & Bačanin Džakula, N. (2019). Convolutional Neural Network Layers and Architectures. January, 445–451. https://doi.org/10.15308/sinteza-2019-445-451

9.HELP FILE

STEP-1: Go to **Start,** search and launch **ANACONDA NAVIGATOR.**

STEP-2: After launching of **ANACONDA NAVIGATOR,** launch **JUPYTER NOTEBOOK.**

STEP-3: Open **"Major project code" IPYNB file.**

STEP-4: Then run all the cells.

STEP-5: All the **data preprocessing**, **training and testing**, **model building**, **accuracy** of the model
can be showcased.

STEP-6: And a .h5 file will be generated.

STEP-7: Create a Folder named **FLASK** on the **DESKTOP.** Extract the .h5 file into this Flask Folder.

STEP-8: Extract all the html files (index.html, result.html) and python file(app.py) into the **FLASK**
Folder.

STEP-9: Then go back to **ANACONDA NAVIGATOR** and the launch the **SPYDER.**

STEP-10: After launching Spyder, give the path of **FLASK FOLDER** which you have created on the
DESKTOP.

STEP-11: Open all the app.py and html files present in the Flask Folder.

STEP-12: After running of the app.py, open **ANACONDA PROMPT** and follow the below steps:

cd File →Path click enter python app.py→click enter (we could see running of files)

STEP-13: Then open **BROWSER,** at the URL area type **—localhost:5000".**

STEP-14: Home page of the project will be displayed.

STEP-15: Click on **—Go to Proceed".** Directly it will be navigated to index page.

STEP-16: A index page will be displayed where the user needs to give the image from folders and then
click on **—Proceed"**. Output will be generated.