Chapter 1

INTRODUCTION

In a world where manual monitoring of water bottle levels is the norm, our project, "Water Bottle Image Classification Model," introduces an innovative solution. Leveraging computer vision and machine learning, it automates the classification of water bottles into full, half-full, or overflowing categories. This technology replaces tedious manual inspections, providing stakeholders with a reliable, real-time method for efficient water resource management in various environments.

In today's dynamic workplaces and public environments, efficient resource management is crucial for ensuring sustainability and operational continuity. The Water Bottle Image Classification Model represents a pioneering solution designed to enhance the monitoring and management of water resources through automated image analysis. This innovative application leverages advanced machine learning techniques, specifically convolutional neural networks (CNNs), to classify water bottle images into distinct categories based on their fill levels—full, half-full, or overflowing. The primary objective of the Water Bottle Image Classification Model is to automate the assessment of water bottle fill levels, reducing reliance on manual inspections and enhancing operational efficiency across various settings. Whether in office spaces, educational institutions, healthcare facilities, or public venues, the model offers a reliable means to monitor and manage water resources effectively.

By employing state-of-the-art deep learning models such as InceptionV3, ResNet50V2, and MobileNetV2, the system ensures accurate and real-time classification of water bottles. This capability not only optimizes resource utilization by minimizing water wastage but also supports proactive maintenance and replenishment strategies. Administrators, maintenance staff, and facilities managers benefit from timely insights into water consumption patterns, facilitating informed decision-making and resource allocation.

The technical backbone of the model involves preprocessing images, utilizing pretrained CNN architectures for feature extraction, and interpreting predictions to assign water level categories. The integration of user-friendly interfaces ensures accessibility, allowing stakeholders to interact seamlessly with the system and obtain instant feedback on water bottle status.

The versatility of the Water Bottle Image Classification Model extends to diverse environments, accommodating varying scales of operation and specific user requirements. Its scalability enables deployment in small offices to large-scale facilities, adapting effortlessly to evolving demands and technological advancements.

In our daily lives, the consumption of bottled water has become ubiquitous, whether in households, offices, or public spaces. Often, the need arises to monitor and manage water levels in these bottles for various reasons, such as tracking usage or preventing spills. In the absence of an automated system, individuals or maintenance personnel typically inspect each bottle manually, assessing the water level and taking appropriate action. This process is not only time-consuming but also prone to human errors. Imagine an office space where water bottles are placed in meeting rooms, and a staff member needs to manually check each bottle's water level to ensure a sufficient supply. This scenario highlights the inefficiency and limitations of the current system.

In the existing system, stakeholders such as office administrators, maintenance staff, or individuals responsible for managing water supplies rely on manual inspections to ensure an adequate water level in each bottle. This process is not only labor-intensive but is also prone to inconsistencies and oversights. Stakeholders currently lack a reliable and automated method for monitoring water levels. Our project addresses this gap by introducing a computer vision-based classification system that can seamlessly integrate into various environments, providing an instant and accurate assessment of water levels in bottles. The end-users, in this case, will experience a significant improvement in efficiency and accuracy, leading to better water resource management.

To achieve this goal, the project follows a systematic approach. We start by collecting a labeled dataset of water bottle images, each tagged with its corresponding water level. The dataset can be obtained from platforms like Kaggle or created through manual labeling. Afterward, image preprocessing techniques, including resizing, cropping, and normalization, are applied to ensure uniformity. Data augmentation is introduced to enhance the model's generalization capabilities. The dataset is then split into training and validation sets. Using TensorFlow and Keras, a Convolutional Neural Network (CNN) model is implemented. The CNN, with its convolutional, pooling, and fully connected layers, learns to classify water bottle images accurately. The model is trained using the Adam optimizer, addressing class imbalances using appropriate weights. Gradio is employed for real-time model deployment, allowing users to input images and receive instant predictions regarding water bottle levels. This project showcases the potential of

machine learning and computer vision in streamlining manual processes and enhancing operational efficiency in everyday scenarios.

The primary stakeholders for the Water Bottle Image Classification Model are businesses and organizations that manage shared spaces, such as offices, conference centers, or public venues. These entities often provide bottled water for employees, visitors, or customers. For office administrators and facility managers, the ability to efficiently monitor water levels in bottles is crucial for maintaining a seamless and uninterrupted supply. The automated classification system reduces the workload on administrative staff, allowing them to focus on more strategic tasks. Moreover, it contributes to cost-effectiveness by preventing unnecessary bottle replacements and minimizing water wastage. This application aligns with the objectives of businesses aiming for streamlined operations and resource optimization.

Another significant stakeholder group includes businesses in the hospitality sector, such as hotels, restaurants, and event venues. For these establishments, ensuring a constant supply of water is not only about efficiency but also about providing a positive and convenient experience for guests. With the Water Bottle Image Classification Model, hospitality businesses can enhance customer service by proactively managing water levels. The application contributes to a seamless and enjoyable customer experience, aligning with the hospitality industry's commitment to quality service. Additionally, by minimizing the need for manual inspections and optimizing resource utilization, these businesses can improve overall operational efficiency and reduce operational costs. Thus, the application serves as a valuable asset for businesses in the hospitality sector, aiming to elevate customer satisfaction and operational excellence.

The Water Bottle Image Classification Model addresses several key challenges for various stakeholders. Firstly, for office administrators and facility managers, manual monitoring of water bottle levels is a time-consuming task that can lead to inefficiencies and potential oversights. The automated classification system solves this problem by providing a quick and accurate assessment of water levels in real-time. This not only streamlines the operational process but also minimizes the risk of disruptions due to insufficient water supply. As a result, stakeholders in office management benefit from improved efficiency and a reduction in the manual workload.

Secondly, for businesses in the hospitality sector, maintaining a consistent and uninterrupted water supply is essential for customer satisfaction. The manual monitoring of water levels in numerous bottles can be impractical and may lead to lapses in service. The Water Bottle Image Classification Model resolves this challenge by offering a reliable, automated solution. It ensures that businesses in the hospitality industry can consistently meet the needs of their customers, enhancing the overall guest experience. By reducing the reliance on manual processes, the application contributes to the seamless and efficient management of resources, addressing a crucial problem faced by businesses striving for excellence in customer service.

The application offers substantial benefits in both business and research domains. In a business context, the automated Water Bottle Image Classification Model enhances operational efficiency by eliminating the need for manual monitoring of water levels. This leads to significant time savings for administrative staff and facility managers, allowing them to allocate their resources to more strategic tasks. Moreover, the reduction in water wastage and optimized bottle replacements contribute to cost-effectiveness, positively impacting the organization's bottom line. Additionally, businesses in the hospitality sector can leverage the application to elevate customer service, ensuring a seamless and uninterrupted water supply for guests.

In the research domain, the application opens avenues for studying and optimizing resource utilization in shared spaces. Researchers can analyze patterns in water consumption, identify trends, and explore opportunities for further efficiency improvements. The real-time monitoring capabilities of the model provide valuable data for assessing user behavior and preferences regarding water consumption. This data-driven approach can contribute to sustainability research by promoting the efficient use of resources and minimizing environmental impact. Overall, the application's benefits extend beyond immediate operational gains, offering valuable insights for businesses and researchers alike.

For the deployment and scalability of the Water Bottle Image Classification Model, a cloud-based infrastructure, such as AWS (Amazon Web Services) or Azure, proves to be highly effective. Leveraging cloud services allows for dynamic resource allocation, ensuring optimal performance during periods of increased demand. Additionally, cloud platforms offer a variety of services, including machine learning services for model

hosting and deployment. This not only simplifies the deployment process but also facilitates easy integration with other cloud-native technologies.

Using serverless computing, such as AWS Lambda or Azure Functions, for specific functionalities like real-time classification can further enhance scalability. Serverless architecture enables automatic scaling based on demand, reducing operational overhead and costs. Additionally, cloud-based storage solutions can efficiently manage the labeled dataset and model artifacts. Overall, a cloud-centric approach ensures robust deployment, scalability, and cost-effectiveness for the Water Bottle Image Classification Model.

1.1 Problem Statement

The project, titled "Water Bottle Image Classification Model with water level," aims to revolutionize and automate the process of monitoring water levels in bottles using computer vision and machine learning. In the current system, where manual inspection is the norm, our project offers a sophisticated solution. Instead of relying on human intervention, our model will be trained to classify images of water bottles based on their water levels – whether they are full, half-full, or overflowing. This automated classification system provides an efficient and accurate means of monitoring water levels in real-time. It eliminates the need for manual inspections, reducing human effort and minimizing errors in the process.

The Water Bottle Image Classification Model is particularly crucial when adopted by stakeholders who manage shared spaces with a significant reliance on bottled water, such as offices, conference centers, or hospitality establishments. In scenarios where a constant and efficient water supply is vital, the application becomes a necessity for facility managers, office administrators, and businesses in the hospitality sector. By implementing the model, these stakeholders can proactively manage water resources, ensuring bottles are replenished precisely when needed and minimizing any interruptions in supply. The financial benefits for major stakeholders arise from the optimized use of resources, reduced operational costs through efficient bottle replacements, and the enhanced customer experience, ultimately contributing to overall financial well-being and success.

While specific competitor organizations implementing similar water bottle monitoring applications might not be explicitly mentioned, the market for smart office and facility management solutions has seen increased interest. Companies providing IoT-based

systems or smart building solutions may offer features that include water usage monitoring. For instance, companies specializing in workplace technology or building management systems may incorporate sensors and analytics to track resource usage, including water consumption. However, the unique aspect of the Water Bottle Image Classification Model lies in its utilization of computer vision and machine learning for automated bottle-level monitoring, distinguishing it from more generalized solutions in the broader smart building management landscape.

The best-suited software architecture for building the Water Bottle Image Classification Model is a microservices architecture. This architecture pattern divides the application into smaller, independent services, each responsible for specific functionalities. In this context, microservices can be designed to handle tasks such as image preprocessing, model training, real-time classification, and user interface interactions. This modular approach allows for scalability and maintainability, as each microservice can be developed, deployed, and updated independently. Additionally, it aligns well with the diverse tasks involved in the application, promoting a flexible and agile development process. Moreover, microservices can facilitate easy integration with other existing systems or future enhancements, making it a suitable choice for a project with multiple functional components like the Water Bottle Image Classification Model.

Image preprocessing is a critical component of the application. This microservice is responsible for handling tasks such as resizing, normalization, and augmentation of images before they are fed into the model. Technologies such as Python, OpenCV, and PIL can be employed, with FastAPI used to create RESTful APIs for seamless integration. This service can be containerized using Docker, ensuring consistency across different environments, and managed with Kubernetes to facilitate scalability and easy updates.

The model training service manages the entire lifecycle of model training, including data ingestion, model selection (e.g., InceptionV3, ResNet50V2, DenseNet121), training, and evaluation. Utilizing technologies like Python, TensorFlow, and Keras, this service can leverage FastAPI for API management. Containerization with Docker ensures that the training environment is consistent, while Kubernetes orchestrates and scales the training tasks efficiently.

Real-time classification is another vital service, providing immediate predictions for images uploaded by users. This service loads pre-trained models and returns predictions

through APIs created with FastAPI. Docker and Kubernetes again play crucial roles in containerizing and managing this service to ensure high availability and scalability.

The user interface service handles the front-end interactions where users can upload images, view predictions, and interact with the system. Streamlit is an excellent choice for rapid UI development, while more complex interfaces could be developed using React or Angular. This service, like the others, can be containerized and managed with Kubernetes, ensuring a seamless user experience.

Integration and communication between these microservices can be facilitated by an API Gateway, acting as the entry point for all client requests. Technologies like Nginx or API Gateway services (e.g., AWS API Gateway) can be used. Service discovery tools like Consul or Kubernetes-native solutions manage the dynamic discovery and management of services. For inter-service communication, especially for asynchronous tasks, message brokers like RabbitMQ or Apache Kafka can be employed.

Containerization and orchestration are key to this architecture. Docker containerizes each microservice, ensuring consistency and isolation across different environments. Kubernetes manages these containerized applications across a cluster of machines, providing automated deployment, scaling, load balancing, and self-healing capabilities. This setup ensures that the application components are encapsulated and can be seamlessly deployed across various infrastructures, enhancing flexibility and ease of maintenance.

Scalability and maintenance are integral benefits of this architecture. Each microservice can be horizontally scaled independently based on its load, ensuring efficient resource use. Implementing Continuous Integration/Continuous Deployment (CI/CD) pipelines with tools like Jenkins or GitHub Actions ensures that changes can be tested and deployed rapidly and reliably.

Monitoring and logging are crucial for maintaining the health and performance of the application. Tools like Prometheus and Grafana provide insights into the microservices' performance, while centralized logging using the ELK Stack (Elasticsearch, Logstash, Kibana) helps aggregate and analyze logs. Security measures, including OAuth 2.0 or JWT for authentication and authorization, encryption for data in transit and at rest, and Kubernetes network policies, ensure robust protection of the system.

Future enhancements could include integration with cloud services for managed databases, storage, and advanced AI/ML capabilities. Implementing auto-scaling policies in Kubernetes would automatically adjust the number of running instances based on real-

time load, further enhancing the system's flexibility and resilience. By following a microservices architecture with containerization and orchestration, the Water Bottle Image Classification Model can achieve high scalability, maintainability, and flexibility, accommodating future growth and integration needs.

1.2 Objectives

The primary objectives of the Water Bottle Image Classification Model project are as follows:

(i) Automated Water Level Classification: The project aims to develop a robust Convolutional Neural Network (CNN) model capable of accurately classifying water bottle images into categories such as full, half-full, or overflowing. By leveraging advanced computer vision techniques, this automated classification system will eliminate the need for manual inspections, ensuring consistent and reliable monitoring of water levels. Develop a robust convolutional neural network (CNN) model capable of accurately classifying water bottle images into categories such as full, half-full, or overflowing. This model will leverage computer vision techniques to automate the monitoring of water levels.

(ii) Efficiency Enhancement: Central to the project is the goal to streamline the process of water level monitoring in shared spaces. By automating the classification of water bottle images, stakeholders can significantly reduce the time and effort traditionally spent on manual assessments. This efficiency enhancement supports operational productivity and allows stakeholders to allocate resources more effectively. Streamline the process of monitoring water levels in shared spaces by eliminating the need for manual inspections. The automated classification system aims to enhance operational efficiency for stakeholders, reducing the time and effort traditionally spent on manual assessments.

(iii) Resource Optimization: The project contributes to resource optimization by proactively monitoring water levels and minimizing wastage. Through accurate classification and timely alerts, the system facilitates efficient bottle replacements, ensuring a continuous water supply while avoiding unnecessary replenishments. This approach not only reduces operational costs but also promotes sustainable resource management practices.Contribute to resource optimization by minimizing water wastage through proactive monitoring. The model will assist in efficient bottle replacements,

ensuring a continuous water supply while preventing unnecessary replacements, thereby reducing operational costs.

(iv) Real-time Classification: The implementation includes a user-friendly interface for real-time classification of water bottle images. Stakeholders can receive instant feedback on water levels, enabling timely decision-making and enhancing the responsiveness of operational workflows. Real-time classification capabilities ensure that stakeholders have immediate access to critical information for effective management. Implement a user-friendly interface for real-time water bottle image classification, allowing stakeholders to receive instant feedback on water levels. This feature ensures timely decision-making and improves the overall responsiveness of the system.

(v) Adaptability: The model is designed to be adaptable to various environments, including offices, conference centers, and hospitality establishments. Its versatility allows it to cater to the specific needs and operational dynamics of different stakeholders in diverse settings. This adaptability ensures that the solution remains relevant and effective across various deployment scenarios. Design the model to be adaptable to various environments, including offices, conference centers, and hospitality establishments. The application's versatility aims to cater to the specific needs of different stakeholders in diverse settings.

(vi) User Interface Integration: To facilitate seamless interaction, the model will be deployed using Gradio to create an intuitive user interface. This interface enables stakeholders to input images and receive classification predictions in real-time, promoting accessibility and ease of use. The user interface integration enhances user experience and encourages widespread adoption of the technology.Deploy the model using Gradio to create an intuitive user interface, enabling stakeholders to interact with the system effortlessly. The user interface will allow users to input images and receive predictions in real-time, ensuring accessibility and ease of use.

(vii) Data-Driven Insights: The project aims to provide stakeholders with valuable insights into water consumption patterns through data analytics. By analyzing data generated from the classification model, stakeholders can gain actionable insights that inform decision-making related to resource management and sustainability initiatives. This data-driven approach supports informed decision-making and promotes efficient resource allocation strategies.Provide stakeholders with valuable data insights through the model, enabling them to analyze water consumption patterns. This data-driven approach supports decision-making processes related to resource management and sustainability.

(viii) Scalability and Integration: Scalability is a key consideration in the project's design, allowing for future enhancements and integration with other systems. The system is built to evolve alongside changing requirements and technological advancements, ensuring long-term viability and adaptability to emerging needs. This scalability enables seamless integration into existing infrastructures and supports the expansion of functionalities as the project matures.Build the application with scalability in mind, allowing for future enhancements and integration with other systems. The goal is to create a solution that can evolve alongside changing requirements and technological advancements.

(ix) Cost-Effectiveness: Water Bottle Image Classification Model project encompasses a comprehensive set of objectives focused on leveraging advanced technology to enhance operational efficiency, promote sustainable resource management, and deliver actionable insights through data-driven decision-making.Ultimately, the project aims to deliver a cost-effective solution by reducing manual labor, minimizing water wastage, and optimizing resource utilization. The financial benefits for stakeholders should be significant, contributing to the overall success and sustainability of the implemented system.

1.3 Scope of the Project

The Water Bottle Image Classification Model's scope encompasses various dimensions to address the challenges and opportunities presented by the manual monitoring of water bottle levels. Firstly, the project focuses on creating a versatile and accurate image classification model capable of differentiating between full, half-full, and overflowing water bottles. This core functionality is essential for stakeholders across different domains, providing a foundational solution to the common problem of manual water level monitoring.

Beyond its primary classification capabilities, the project extends its scope to enhance operational efficiency. The automated system aims to streamline the process of monitoring water levels in shared spaces, ultimately reducing the time and effort expended by stakeholders. The scope also includes real-time classification, allowing users to receive instantaneous feedback on water bottle status, contributing to quick decision-making. Additionally, the adaptability of the model to diverse environments such as

offices, conference centers, and hospitality establishments broadens its applicability, ensuring its relevance across a range of scenarios.

Furthermore, the project's scope emphasizes the user interface's integration and the deployment of the model using Gradio. This not only simplifies user interactions but also ensures accessibility for stakeholders with varying technical expertise. The application's potential for providing data-driven insights into water consumption patterns expands its scope to contribute valuable information for decision-making related to resource management and sustainability efforts. Overall, the Water Bottle Image Classification Model's scope is comprehensive, encompassing both core technical aspects and user-centric features to offer a holistic solution for efficient water bottle level monitoring.

Summary

In this chapter, the focus was on providing an introduction to the Water Bottle Image Classification Model project, emphasizing its objectives and scope. Moving forward, subsequent chapters will delve into the detailed methodologies employed in data collection, preprocessing, and model training. The architecture and deployment strategies of the CNN model will be thoroughly explored.

The report will conclude with an evaluation of performance metrics and insights into the deployment using Gradio. The organizational structure ensures a comprehensive understanding of the project's intricacies and its significance in the broader technological landscape. Each chapter contributes to unfolding the narrative of how <Company Name> is at the forefront of leveraging technology to address real-world challenges and deliver impactful solutions.

Chapter 2

LITERATURE SURVEY

In this chapter, we delve into an exploration of the existing systems related to water bottle image classification and introduce the proposed system. The design and implementation of an effective Convolutional Neural Network (CNN) for the classification of water bottles based on their water levels represent a significant advancement in the domain of image recognition. As we progress, we will scrutinize the current state of the art, identify limitations, and present a robust framework for improved accuracy and real-time classification.

2.1 Related work

i. **Image Classification with CNNs:**

Convolutional Neural Networks (CNNs) have revolutionized image classification tasks by automating the feature extraction process. Early models like AlexNet (Krizhevsky et al., 2012) showcased the potential of deep learning in image classification, achieving substantial improvements over traditional methods. This success paved the way for more advanced architectures such as VGGNet, GoogLeNet (Inception), ResNet, and DenseNet, each contributing to enhanced accuracy and efficiency in various image recognition tasks.

ii. **Water Bottle Image Classification:**

Although the specific task of classifying water bottles based on their water levels is relatively underexplored, related studies in object detection and liquid level detection provide valuable insights. For instance, Liu et al. (2018) proposed a method for liquid level detection in transparent containers, combining traditional image processing with machine learning techniques. Their approach highlighted challenges like reflections and transparency, which are pertinent to water bottle classification. While their work focused on transparent containers, it laid the groundwork for addressing similar issues in water bottle classification.

iii. **Transfer Learning and Fine-Tuning:**

Transfer learning has become a crucial technique in deep learning, particularly when dealing with limited data. By leveraging pre-trained models on large datasets, researchers can adapt these models to specific tasks with relatively little data. Studies by Yosinski et al. (2014) and Shin et al. (2016) demonstrated the effectiveness of fine-

tuning pre-trained CNNs for specialized applications such as medical image analysis. This approach not only reduces training time but also enhances model performance by transferring knowledge from generic to specific tasks.

iv. **Real-Time Classification Systems:**

Real-time image classification demands efficient and optimized models. MobileNets, introduced by Howard et al. (2017), represent a family of lightweight CNNs designed for mobile and embedded vision applications. These models strike a balance between accuracy and computational efficiency, making them ideal for real-time deployment. Their success in mobile vision tasks underscores the potential for applying similar strategies in water bottle classification systems.

v. **Comparative Studies of CNN Architectures:**

The three models employed in this study—InceptionV3, ResNet50V2, and MobileNetV2—have demonstrated superior performance in various image classification tasks.

vi. **InceptionV3:** Developed by Szegedy et al. (2016), InceptionV3 introduced the concept of factorized convolutions and aggressive regularization to enhance both the depth and width of the network. This architecture is known for achieving high accuracy while maintaining computational efficiency, making it suitable for complex image classification tasks.

vii. **ResNet50V2:** He et al. (2016) introduced ResNet, which employed residual learning to address the vanishing gradient problem in deep networks. The ResNet50V2 variant further refined the architecture, improving training stability and performance. Its ability to train very deep networks without degradation in performance makes it a powerful tool for image classification.

viii. **MobileNetV2:** Sandler et al. (2018) designed MobileNetV2 for mobile and embedded vision applications, emphasizing model efficiency and performance. It utilizes depth-wise separable convolutions and an inverted residual structure to reduce the number of parameters and computational cost while maintaining high accuracy. This makes MobileNetV2 particularly suitable for real-time classification tasks.

ix. **Challenges and Limitations:**

x. Despite the advancements in CNN architectures, several challenges persist in water bottle image classification. Variability in lighting conditions, occlusions, and reflections can significantly impact model performance. Additionally, the availability

of annotated datasets specific to water bottle images is limited, necessitating the use of data augmentation and synthetic data generation techniques to improve model robustness.

2.2 Existing System

i. Surveying Current Approaches

In this subsection, we conduct a thorough review of the current landscape in water bottle image classification. Various existing systems, methodologies, and models will be scrutinized to understand their architecture, dataset utilization, and performance metrics. This survey provides a baseline for identifying common practices, challenges, and potential areas for improvement. This survey enables us to identify common techniques and challenges prevalent in water bottle image classification. It provides insights into how different architectures handle features extraction and classification, the diversity of datasets used (including their size and composition), and the methodologies for evaluating model performance. Understanding these aspects is crucial for benchmarking the proposed model against existing standards and pinpointing opportunities for improvement.

ii. Limitations and Challenges

By critically examining the existing systems, we aim to outline their limitations and challenges. This includes aspects such as dataset size, model accuracy, and adaptability to different scenarios. Recognizing these shortcomings is essential for guiding the enhancements proposed in the subsequent sections. Through our critical assessment of existing systems, we delineate their inherent limitations and challenges. These encompass several dimensions, including but not limited to:

- **Dataset Size and Quality:** Many existing systems may be constrained by small or insufficiently diverse datasets, limiting their ability to generalize across different environments or bottle types.
- **Model Accuracy:** Accuracy rates vary widely among different approaches, influenced by factors such as dataset quality, model architecture complexity, and training strategies.

- **Adaptability:** The adaptability of existing systems to varied scenarios, such as different bottle shapes, lighting conditions, or environmental settings, often poses significant challenges.

Identifying these limitations serves as a foundational step in refining the proposed model. By addressing these challenges, we aim to enhance the robustness, accuracy, and applicability of the classification system in real-world settings.

iii. Performance Metrics

An analysis of the performance metrics employed in the existing systems will be conducted. This involves assessing the criteria used to evaluate the effectiveness of image classification models. Precision, recall, and validation loss are among the metrics to be explored, shedding light on the strengths and weaknesses of current approaches. An in-depth analysis of performance metrics used in existing systems sheds light on their efficacy and limitations. Metrics such as precision, recall, and validation loss provide quantitative measures of model performance. Precision measures the accuracy of positive predictions, recall evaluates the proportion of actual positives correctly identified, and validation loss indicates the model's prediction error during training.

By scrutinizing how these metrics are applied and interpreted across different studies, we gain insights into the strengths and weaknesses of various approaches. This analysis guides our selection of appropriate metrics for evaluating the proposed model, ensuring comprehensive assessment of its effectiveness in comparison to existing systems.

2.3 Proposed System

i. Vision and Objectives

The proposed system introduces a novel approach to water bottle image classification. In this subsection, we articulate the vision and objectives of the proposed Convolutional Neural Network (CNN) model. The system's overarching goals, such as improved accuracy, real-time classification, and adaptability to diverse datasets, will be clearly defined.

ii. Innovations and Advancements

Building on the insights gained from the existing systems, this part of the chapter outlines the innovations and advancements integrated into the proposed model. Techniques like data augmentation, class weighting, and a meticulously designed model architecture will be

explained in detail. Each enhancement is justified in the context of addressing the identified limitations of the existing systems.

Drawing from insights gained in our survey of existing systems, the proposed model integrates several innovations and advancements. Key enhancements include:

- **Data Augmentation:** Augmenting the dataset with techniques like rotation, flipping, and scaling to enhance model robustness and reduce overfitting.
- **Class Weighting:** Addressing dataset imbalance by assigning higher weights to minority classes (e.g., overflowing bottles) during model training to improve overall classification performance.
- **Model Architecture Optimization:** Designing a CNN architecture tailored for water bottle classification, incorporating optimized convolutional layers, pooling strategies, and activation functions to enhance feature extraction and classification accuracy.

Each innovation is meticulously selected and justified based on its potential to mitigate the identified limitations of existing systems, thereby elevating the performance and applicability of the proposed model.

iii. Model Architecture

The proposed CNN architecture is meticulously designed to optimize performance in water bottle image classification tasks. It comprises multiple layers, including convolutional layers for feature extraction, pooling layers for spatial down-sampling, and fully connected layers for classification. The architecture's specifics, such as filter sizes, kernel dimensions, and activation functions (e.g., ReLU), are chosen to maximize feature representation and minimize information loss during processing.A comprehensive overview of the proposed CNN architecture is provided, detailing the structure of convolutional layers, pooling layers, and fully connected layers. This includes specifics such as the number of filters, kernel sizes, and activation functions. The rationale behind these choices is elucidated to demonstrate how the architecture contributes to improved classification accuracy.

iv. Training Strategy

A robust training strategy underpins the effectiveness of the proposed model. This includes selecting appropriate optimizers (e.g., Adam), defining batch sizes conducive to efficient gradient descent, and employing techniques like early stopping to prevent overfitting. Moreover, the use of class weights ensures balanced learning across different water bottle categories, thereby enhancing the model's ability to generalize to unseen data.This subsection

delves into the training strategy employed for the proposed system. Details such as optimizer selection, batch size, and the use of class weights to address imbalanced datasets are discussed. By elaborating on the model's training regimen, readers gain insights into how the proposed system learns and generalizes from the provided dataset.

v. Performance Evaluation

To quantify the efficacy of the proposed system, comprehensive performance evaluation metrics are employed. Precision, recall, validation loss, and overall accuracy are systematically evaluated to assess the model's classification accuracy and generalization capabilities. Comparative analysis with benchmarks set by existing systems provides a quantitative validation of the proposed advancements and innovations.

The proposed system's performance is evaluated using a set of predefined metrics. Precision, recall, validation loss, and accuracy are analyzed to assess the model's effectiveness in comparison to the existing systems. This section serves as a quantitative validation of the proposed enhancements.

2.4 Feasibility Study

Chapter 2 presents a critical analysis of the feasibility of implementing the water bottle image classification system. The technical feasibility evaluation encompasses an assessment of hardware requirements, software tools, and the scalability and complexity of the proposed Convolutional Neural Network (CNN) model. This analysis ensures that the technical infrastructure can support the development and deployment of the system.

On the operational front, the feasibility study examines user acceptance, emphasizing the creation of user-friendly interfaces for real-time predictions. Integration with Gradio is explored to assess how well the system aligns with operational requirements. Additionally, considerations for ongoing maintenance and support are outlined to ensure the sustained functionality of the deployed model. This chapter lays the groundwork for understanding the practical viability and acceptance of the proposed water bottle image classification system from both technical and operational perspectives.

2.4.1 Technical Feasibility

i. Hardware Requirements

The technical feasibility assessment begins with a thorough evaluation of the hardware prerequisites for deploying and operating the proposed water bottle image classification

system. Central to this evaluation is the scrutiny of computational resources required for training the Convolutional Neural Network (CNN). Given the intensive nature of deep learning tasks, including image classification, considerations include the availability of GPUs capable of accelerating matrix operations and optimizing training times. Memory specifications are also crucial, ensuring sufficient RAM to handle large datasets and model complexities without performance degradation.

Comparing available hardware resources against these requirements is pivotal in determining the feasibility of the project. This analysis not only assesses the adequacy of existing infrastructure but also guides potential upgrades or acquisitions necessary to support the system's computational demands effectively.To assess the technical feasibility, it is imperative to evaluate the hardware requirements for implementing the proposed water bottle image classification system. This involves scrutinizing the computational resources needed for training the Convolutional Neural Network (CNN), such as GPU capabilities and memory specifications. A comparison of available hardware resources against these requirements will determine the technical viability of the project.

ii. Software and Frameworks

An integral part of technical feasibility is the compatibility and suitability of software and frameworks essential for implementing the water bottle image classification model. TensorFlow and Keras, renowned for their robustness in developing and training deep learning models, are foundational choices due to their extensive libraries, community support, and integration capabilities. Compatibility checks ensure that the selected frameworks align seamlessly with the project's objectives, facilitating efficient model development, training, and deployment.

Furthermore, the integration of Gradio—a user-friendly interface creation toolkit—adds another layer of operational feasibility by enabling real-time interaction with the classification model. Ensuring these software components are readily accessible and properly configured is paramount for leveraging their full potential in achieving the project's technical goals.The technical feasibility study also encompasses an analysis of the software and frameworks essential for the project. This includes the compatibility of the proposed system with TensorFlow, Keras, and Gradio. Ensuring that the chosen tools align with the project's goals and are readily available is crucial for the successful implementation of the water bottle image classification model.

iii. Model Complexity and Scalability

Assessing the technical feasibility also involves evaluating the complexity and scalability of the proposed CNN model. The model's architecture, encompassing layers for feature extraction, pooling, and classification, dictates its computational intensity during both training and inference phases. Understanding these computational demands is crucial for anticipating hardware requirements and optimizing performance across varying dataset sizes and environmental conditions.

Scalability considerations extend beyond computational efficiency to encompass the model's ability to handle increased data volumes and diverse input scenarios. As the dataset grows or as additional features are incorporated into the classification system, the infrastructure must accommodate heightened computational loads without compromising performance or accuracy. This evaluation ensures that the technical infrastructure is not only capable of supporting current requirements but also scalable enough to adapt to future enhancements and expansions.

An evaluation of the proposed CNN model's complexity and scalability is integral to technical feasibility. Assessing the computational demands during training and inference, and understanding how the model performs as the dataset size increases, helps ascertain whether the technical infrastructure can accommodate the envisioned scale of the system.

2.4.2 Operational Feasibility

i. User Acceptance

Operational feasibility examines the acceptance and usability of the water bottle image classification system by end-users—critical stakeholders who interact with the system in real-world settings. User acceptance testing plays a pivotal role in evaluating how well the system meets user expectations, addresses operational needs, and integrates seamlessly into existing workflows. Feedback gathered through usability testing sessions provides invaluable insights into user satisfaction, ease of interaction with the Gradio interface, and the system's overall practicality in enhancing operational efficiency.

Operational feasibility involves gauging the acceptance and usability of the proposed system by end-users. This includes potential users interacting with the real-time classification

interface created using Gradio. User feedback and usability testing will be instrumental in determining the practicality and acceptance of the system in a real-world scenario.

ii. Integration with Gradio

The operational feasibility study includes a detailed assessment of integrating the water bottle image classification model with Gradio—a platform for creating intuitive and interactive machine learning applications. Evaluating this integration involves verifying the compatibility of the classification model's output with Gradio's interface requirements, ensuring that real-time predictions are presented accurately and promptly to users. Potential challenges, such as data formatting or interface responsiveness, are identified and addressed during this phase to guarantee a seamless user experience and operational continuity.

Assessing the operational feasibility also requires a detailed examination of the integration of the proposed system with Gradio. This involves understanding how well the water bottle image classification model interfaces with Gradio for seamless deployment and real-time predictions. Any challenges or compatibility issues need to be addressed to ensure smooth operational integration.

iii. Maintenance and Support

Sustaining operational feasibility necessitates careful consideration of the system's ongoing maintenance and support requirements. This encompasses maintaining the trained model's performance through periodic updates, addressing potential drift in classification accuracy, and integrating new datasets to enhance model robustness. A structured plan for system maintenance ensures timely bug fixes, security patches, and user support services, thereby safeguarding the system's operational integrity and longevity.

Consideration of the ongoing maintenance and support requirements is vital for operational feasibility. This includes assessing the ease of maintaining the trained model, updating datasets, and providing support for potential issues or updates. A well-defined plan for system maintenance and user support contributes to the overall operational viability of the proposed water bottle image classification system.

2.5 Tools and Technologies

2.5.1 Technologies

i. TensorFlow

TensorFlow serves as the foundational technology for implementing the Convolutional Neural Network (CNN) in this project. TensorFlow is an open-source machine learning library that facilitates the development and training of deep learning models.TensorFlow's extensive functionality, flexibility, and robust community support make it an ideal choice for building and training complex neural network architectures. Its compatibility with Keras, a high-level neural networks API, simplifies the model design and implementation process.

TensorFlow serves as the backbone for implementing the Convolutional Neural Network (CNN) architecture used to classify water bottle images. Its robust APIs enable efficient computation on both CPUs and GPUs, crucial for handling large-scale image datasets and complex model architectures. TensorFlow's flexibility allows for customization of model layers, activation functions, optimizers, and loss functions, tailored specifically to the project's requirements. Keras, an integral part of TensorFlow, provides a high-level API for building neural networks. Its user-friendly interface abstracts away low-level details, simplifying the process of defining, training, and evaluating deep learning models. The seamless integration between TensorFlow and Keras enhances development productivity and facilitates rapid experimentation with different network architectures and hyperparameters.

TensorFlow's extensive functionality and performance optimizations contribute to achieving high accuracy in water bottle image classification. Its ability to leverage hardware accelerators like GPUs accelerates model training, making real-time classification feasible. The integration with Keras ensures scalability and adaptability, enabling the model to handle diverse environmental conditions and varying image characteristics effectively.

ii. Keras

Keras operates as a high-level neural networks API, designed to be user-friendly, modular, and extensible. It acts as an interface for building and training deep learning models, seamlessly integrating with TensorFlow.Keras simplifies the implementation of neural networks, enabling a more intuitive design process. Its compatibility with TensorFlow provides a seamless workflow for constructing, training, and evaluating the water bottle image classification model.

In the Water Bottle Image Classification Model project, Keras facilitates the design and implementation of the CNN architecture. It abstracts away complexities associated with

neural network construction, allowing developers to focus on model design rather than implementation details. Keras provides a wide range of pre-built layers, activation functions, and loss functions, enabling rapid prototyping and iteration of different network configurations. The integration of Keras with TensorFlow ensures compatibility with TensorFlow's computational graph execution model. This integration streamlines the training and deployment process, providing a unified framework for building and evaluating deep learning models. Developers benefit from TensorFlow's optimization capabilities while leveraging Keras's intuitive design principles for model development.

Keras enhances the project's objectives by facilitating the rapid development and deployment of the water bottle image classification model. Its modular design supports iterative improvements to the model architecture, ensuring adaptability to evolving requirements and dataset characteristics. By abstracting away technical complexities, Keras accelerates the implementation timeline and empowers stakeholders to focus on refining model performance and accuracy.

iii. Gradio

Gradio is a library that facilitates the deployment and interaction with machine learning models through easy-to-use interfaces. It enables the creation of interactive web interfaces for real-time model predictions.Gradio is selected for its ability to create user-friendly interfaces for the water bottle image classification model. It allows users to upload images, obtain predictions, and experience real-time classification, making the model accessible to a broader audience.

In the context of the Water Bottle Image Classification Model project, Gradio facilitates the creation of an intuitive user interface for interacting with the classification model. It allows stakeholders, including office administrators, maintenance staff, and health officers, to upload images of water bottles and receive real-time predictions on their water levels. Gradio supports various input formats, such as image uploads, and provides immediate visual feedback, enhancing user engagement and accessibility.

Gradio seamlessly integrates with TensorFlow and Keras, enabling the deployment of deep learning models with minimal configuration. It abstracts away the complexities of web development and frontend design, offering customizable components for displaying model outputs and user interactions. Gradio's integration capabilities ensure compatibility with

Python-based machine learning frameworks, facilitating rapid deployment and iteration of model enhancements.

Gradio plays a crucial role in achieving user acceptance and operational feasibility for the water bottle image classification model. Its intuitive interface empowers non-technical users to interact with the model effectively, fostering adoption across diverse organizational settings. By providing real-time predictions and visual explanations, Gradio enhances transparency and trust in model outputs, facilitating informed decision-making related to water resource management and sustainability.

iv. NumPy

NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.NumPy is employed for efficient manipulation of data arrays, essential for preprocessing images and preparing the dataset for training. Its array operations contribute to the optimization of various mathematical operations within the project.

NumPy serves as the primary tool for data manipulation and preprocessing within the Water Bottle Image Classification Model project. It facilitates the conversion of image data into numerical arrays, essential for feeding into the CNN model. NumPy's array operations enable efficient batch processing, image normalization, and augmentation, optimizing data preparation workflows for model training and evaluation. NumPy seamlessly integrates with TensorFlow and Keras, supporting data transformations and array operations required during model training and inference. Its compatibility with Python's scientific computing ecosystem enhances computational efficiency, enabling rapid iteration and experimentation with different preprocessing techniques and data augmentation strategies.

NumPy's efficient data handling capabilities contribute to the project's performance metrics, including model accuracy and training speed. By streamlining data preprocessing tasks, NumPy accelerates the development cycle and enhances the robustness of the water bottle image classification model. Its support for multi-dimensional arrays and mathematical operations ensures scalability, enabling the model to process large datasets effectively while maintaining computational efficiency.

v. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for plotting graphs and visualizing data.Matplotlib is utilized to visualize key aspects of the project, such as sample images from the dataset, model performance metrics, and training/validation loss curves. Its versatile plotting capabilities aid in conveying information effectively.

Matplotlib facilitates the visualization of key aspects of the Water Bottle Image Classification Model project. It is utilized to visualize sample images from the dataset, display model performance metrics such as accuracy and loss curves during training, and present classification results and confusion matrices. Integrating Matplotlib into the project workflow enables stakeholders to gain insights into model behavior, performance trends, and data distributions visually. Its plotting functions support customization of plot styles, labels, and annotations, enhancing the clarity and interpretability of visualizations presented in project reports and presentations.

Matplotlib enhances the project's communication of results and findings through visual representations. By visualizing model outputs and performance metrics, Matplotlib facilitates informed decision-making and evaluation of model effectiveness. Its support for interactive plots and graphical annotations promotes engagement with project stakeholders, fostering collaborative discussions and iterative improvements to the water bottle image classification model.

These technologies form the core infrastructure for the water bottle image classification project, ensuring a robust and efficient implementation. The combination of TensorFlow and Keras provides a powerful platform for neural network development, Gradio facilitates real-time interactions, and NumPy and Matplotlib contribute to data manipulation and visualization.

2.5.2 Tools

i. Jupyter Notebooks

Jupyter Notebooks are interactive computing environments that facilitate code development, data exploration, and visualization. They support a combination of code, text, and multimedia elements.Jupyter Notebooks provide an interactive and collaborative space for developing and experimenting with code. They are instrumental

in creating a narrative around the project, allowing seamless integration of code snippets, visualizations, and textual explanations.

In the Water Bottle Image Classification Model project, Jupyter Notebooks play a pivotal role in iterative development and experimentation. They provide a flexible and interactive workspace where developers can prototype machine learning models, preprocess data, and visualize results seamlessly. The combination of code cells, markdown cells, and interactive widgets allows for the creation of comprehensive project documentation, incorporating code snippets, visualizations, and explanatory text.

ii. GitHub

GitHub is a web-based platform for version control and collaborative software development. It offers features such as code repositories, branching, and collaboration tools for multiple contributors. GitHub is employed for version control, enabling the tracking of changes, collaboration among team members, and easy sharing of the project with the community. It ensures a well-organized and documented development process.

In the Water Bottle Image Classification Model project, GitHub serves as the central repository for version control and collaborative software development. It enables multiple team members to work concurrently on the project, track changes, and merge contributions seamlessly. GitHub's branching model supports experimentation with different features or improvements, ensuring that changes can be reviewed, discussed, and integrated into the main codebase systematically. Additionally, GitHub's integration with continuous integration (CI) tools automates testing and deployment workflows, enhancing code quality and project reliability. The adoption of GitHub ensures a structured and organized development process for the project. It provides transparency into code changes, fosters accountability among team members, and facilitates knowledge transfer through detailed commit messages and pull request reviews.

GitHub's extensive community features enable collaboration beyond the immediate project team, fostering contributions from external developers and stakeholders. By leveraging GitHub's version control capabilities, the Water Bottle Image Classification Model project maintains a robust development lifecycle, mitigating risks associated with code conflicts, regressions, and versioning issues.

iii. Visual Studio Code (VSCode)

Visual Studio Code is a lightweight, open-source code editor with extensive support for various programming languages. It includes features such as syntax highlighting, debugging, and extensions for additional functionalities.Visual Studio Code serves as the primary integrated development environment (IDE) for coding tasks related to the project. Its versatility, ease of use, and integration with extensions enhance the development experience.

As the primary integrated development environment (IDE) for the Water Bottle Image Classification Model project, VSCode facilitates efficient coding, debugging, and project management tasks. Its intuitive user interface and customizable layout enhance developer productivity by providing quick access to essential tools and features. VSCode's integration with Git enables seamless version control operations directly within the editor, streamlining collaborative development workflows and ensuring code consistency.

VSCode was selected for its versatility, performance, and extensive ecosystem of extensions tailored to machine learning and deep learning workflows. Its lightweight nature ensures responsiveness even when handling large-scale codebases and resource-intensive tasks such as model training and evaluation. VSCode's integrated terminal and debugging capabilities support iterative development and troubleshooting, enabling developers to iteratively refine the water bottle image classification model. The availability of AI-focused extensions enhances VSCode's utility in data preprocessing, visualization, and integration with machine learning frameworks like TensorFlow and Keras.

iv. Labelbox

Labelbox is a data labeling platform that facilitates the annotation of images and other data types. It provides tools for creating labeled datasets, a crucial step in training machine learning models.Labelbox is utilized for efficiently labeling images in the dataset, especially in cases where manual labeling is required. Its user-friendly interface and collaboration features make it an effective tool for dataset annotation.

In the Water Bottle Image Classification Model project, Labelbox plays a critical role in generating high-quality labeled datasets for training the image classification model. Its intuitive interface allows annotators to mark water bottle images with corresponding labels (e.g., full, half, overflow) efficiently and accurately. Labelbox supports

collaborative labeling workflows, enabling multiple annotators to work concurrently on the same dataset while maintaining consistency and quality control.

Labelbox was chosen for its scalability, customization options, and support for diverse annotation types, including bounding boxes and polygonal annotations. Its integration capabilities facilitate seamless data import and export, ensuring compatibility with various machine learning frameworks and data formats. By leveraging Labelbox, the Water Bottle Image Classification Model project accelerates the dataset preparation phase, reducing manual effort and ensuring the availability of annotated data necessary for training and evaluating the model's performance. The platform's annotation review and management features promote data quality assurance, enhancing the reliability and accuracy of the trained model.

v. Google Colab

Google Colab is a cloud-based platform that provides free access to GPUs and TPUs for running Jupyter Notebooks. It allows users to execute code and train machine learning models in a cloud environment.Google Colab is utilized for training the machine learning model on GPU resources without the need for local hardware acceleration. Its integration with Jupyter Notebooks simplifies the transition from local development to cloud-based training.

Google Colab serves as a scalable and cost-effective solution for training the deep learning model on GPU-accelerated resources without the need for local hardware provisioning. It supports the seamless execution of Jupyter Notebooks, enabling developers to leverage high-performance computing capabilities for intensive tasks such as model training, hyperparameter tuning, and performance evaluation. Google Colab's integration with Google Drive facilitates data storage and access, ensuring continuity and accessibility of project assets across distributed teams.

The utilization of Google Colab aligns with the project's need for scalable computing resources and collaborative development workflows. By leveraging cloud-based GPUs and TPUs, Google Colab accelerates model training times and supports experimentation with complex neural network architectures. Its integration with popular Python libraries and frameworks, including TensorFlow and Keras, simplifies the migration of Jupyter Notebooks from local environments to the cloud, enhancing workflow flexibility and

productivity. Google Colab's interactive and sharable notebooks promote knowledge sharing and collaboration among project contributors, facilitating rapid iteration and refinement of the water bottle image classification model.

These tools complement the chosen technologies, enhancing the development and collaboration aspects of the water bottle image classification project. Jupyter Notebooks and Visual Studio Code support code development, GitHub ensures version control and collaboration, Labelbox facilitates efficient dataset labeling, and Google Colab provides cloud-based GPU resources for training.

- Summary

In Chapter 2, the project undertakes a comprehensive exploration of the existing and proposed systems for water bottle image classification. The examination of the existing systems involves a critical survey of current approaches, identification of limitations, and an analysis of performance metrics. Subsequently, the proposed system is introduced with a clear vision, outlining objectives and innovations. The feasibility study delves into technical aspects, including hardware and software requirements, and operational considerations such as user acceptance and maintenance plans. The tools and technologies chosen, including TensorFlow, Keras, Gradio, NumPy, and Matplotlib, are justified for their roles in model development, deployment, and data manipulation. The chapter concludes with a detailed overview of the selected tools, including Jupyter Notebooks, GitHub, Visual Studio Code, Labelbox, and Google Colab, highlighting their contributions to the project's success.

Chapter 3

SOFTWARE REQUIREMENTS SPECIFICATIONS

The Software Requirements Specifications (SRS) document outlines the functional and non-functional requirements for the Water Bottle Classification with Water Level project. It serves as a comprehensive guide for developers, stakeholders, and users involved in the software development process. The Software Requirements Specifications (SRS) document outlines the functional and non-functional requirements for the Water Bottle Classification with Water Level project. It serves as a comprehensive guide for developers, stakeholders, and users involved in the software development process.

3.1 User Interface Requirements

The user interface (UI) for the Water Bottle Image Classification Model project must be designed with the following requirements:

1. **Intuitive Design**:
    - The UI should be simple and intuitive, allowing users to easily navigate and use the system without extensive training.
    - It should have a clean layout with clear instructions and visual cues to guide users through the process of uploading images and viewing classification results.
2. **Responsive Design**:
    - The UI must be responsive, ensuring it works well on various devices, including desktops, tablets, and smartphones.
    - The layout should adapt to different screen sizes and orientations, maintaining usability and accessibility.
3. **Interactive Elements**:
    - Interactive elements such as buttons, sliders, and upload fields should be prominently displayed and easily accessible.
    - Feedback should be provided in real-time to inform users of successful actions or errors (e.g., successful image upload, classification results).
4. **Accessibility**:

- The UI should be designed to be accessible to users with disabilities, following standards such as the Web Content Accessibility Guidelines (WCAG).
- Features like text-to-speech, keyboard navigation, and high-contrast themes should be considered.

5. **Localization**:
   - The UI should support multiple languages to cater to users from different regions.
   - Text and labels should be easily translatable, and the system should allow users to select their preferred language.

3.2 Hardware and Software Requirements

Component	Specification
Operating System	Windows Operating System
Processor	Intel Core i3 or above
RAM	4GB
Hard Disk	30GB
Python Version	3.12
Python Packages	TensorFlow 2.16.2, Keras 3.4.1, Streamlit
IDE/Editor	Visual Studio Code, Jupyter Notebook
Deployment Tool	Gradio, Docker
Browser	Google Chrome, Mozilla Firefox
Additional Tools	Conda

Table 3.1 Hardware and software requirements

3.4 Functional and Non-functional Requirements

3.4.1 Functional Requirements

i. Data Collection Module

The Data Collection Module serves as the foundational step in developing the water bottle image classification model. Its primary function is to acquire a labeled dataset of water bottle images, crucial for training and evaluating the model's performance. This module collects images from diverse sources, ensuring a representative sample of water bottles in varying states—full, half-full, or overflowing. Whether sourced from online repositories or captured through manual efforts, the collected images are meticulously labeled to denote their corresponding water levels. By assembling this comprehensive dataset, the module lays the groundwork for subsequent stages, enabling the model to learn and classify water bottles accurately in real-world scenarios. Collects labeled dataset of water bottle images.Purpose is that module serves as the initial step in building the image classification model. Gathering a labelled dataset is crucial as it provides the necessary data for training and testing the model. Operation is module may obtain images from various sources such as online repositories or by capturing images manually. These images are then labeled with their corresponding water levels (full, half, or overflowing) either through automated tagging or manual annotation processes. output of this module is a labeled dataset containing water bottle images alongside their respective water level labels. This dataset forms the foundation for training the image classification model.

ii. Preprocessing Module

Standardizes and prepares images for model input. Purpose is raw images often vary in size, resolution, and quality, which can hinder the performance of the model. The preprocessing module addresses these issues by standardizing the images and preparing them for input into the model. Operation for preprocessing techniques such as resizing, cropping, and normalization are applied to the raw images. Resizing ensures that all images have the same dimensions, while cropping removes irrelevant parts. Normalization adjusts pixel values to a common scale, improving the model's convergence during training. Output module produces preprocessed images that are uniform in size, format, and quality, ready to be fed into the image classification model.Following data collection, the Preprocessing Module undertakes the essential task of refining and standardizing the acquired images to optimize their suitability for model input. Raw images often exhibit disparities in size, resolution, and

quality, which can hinder the model's performance during training. To address these inconsistencies, the preprocessing module applies various techniques such as resizing, cropping, and normalization. These operations ensure uniformity among the images, enhancing their compatibility with the model architecture. By preparing the images in a standardized format, the module streamlines the subsequent training process, facilitating smoother convergence and improved model performance.

iii. Data Augmentation Module

Generates augmented data to enhance model generalization. Purpose of Data augmentation increases the diversity of the dataset, helping the model generalize better to unseen data and reducing the risk of overfitting. Operation of Various augmentation techniques such as rotation, flipping, zooming, and adding noise are applied to the preprocessed images. These transformations create new variations of the original images without changing their underlying semantics. The module produces an augmented dataset that includes both the original and augmented images. This expanded dataset provides the model with more examples to learn from, improving its ability to classify water bottle images accurately.

The Data Augmentation Module plays a pivotal role in enhancing the model's robustness and generalization capabilities by generating augmented data. Data augmentation is a crucial technique employed to diversify the training dataset, exposing the model to a broader spectrum of scenarios. Leveraging transformations such as rotation, flipping, and introducing noise, the module creates additional variations of the original images. These augmented images serve to enrich the training data, mitigating the risk of overfitting and equipping the model to handle diverse real-world conditions. By augmenting the dataset with a diverse array of examples, the module fosters a more resilient and adaptable image classification model.

iv. Image Classification Module

Classifies water bottles into full, half, or overflowing. The primary objective of the project is to accurately classify water bottles based on their water levels. This module achieves this by leveraging machine learning techniques, particularly Convolutional Neural Networks (CNNs). The module takes preprocessed or augmented images as input and extracts features using a CNN architecture. These features are then fed into fully connected layers for classification. The model learns to differentiate between full, half-full, and overflowing water bottles based on the patterns it identifies in the input images. The output of the module is the

predicted water level category for each input image, indicating whether the bottle is full, half-full, or overflowing.

At the heart of the project lies the Image Classification Module, which embodies the core functionality of the system. This module employs machine learning techniques, particularly Convolutional Neural Networks (CNNs), to classify water bottles into distinct categories based on their water levels. Upon receiving preprocessed or augmented images as input, the module extracts intricate features using the CNN architecture and employs fully connected layers for classification. Through an iterative learning process, the model discerns subtle patterns within the images, enabling it to accurately classify water bottles as full, half-full, or overflowing. The output of this module—the predicted water level categories—serves as the foundation for informed decision-making in resource management scenarios.

v. Gradio Deployment Module

Deploys the trained model using Gradio for real-time classification. Once the image classification model is trained, it needs to be deployed in a user-friendly manner for practical use. Gradio simplifies this deployment process by providing an intuitive interface for users to interact with the model.The trained model is integrated with Gradio, allowing users to upload images of water bottles through a web interface. The model then provides real-time predictions on the water levels of the bottles.The module enables users to receive instant feedback on the water levels of the bottles they input, facilitating efficient monitoring and management of water resources in real-world scenarios.

Once the image classification model is trained and validated, the Gradio Deployment Module facilitates its seamless integration into real-world environments through a user-friendly interface. Gradio simplifies the deployment process by providing an intuitive web interface for users to interact with the model in real-time. Users can effortlessly upload images of water bottles via the interface, prompting the model to provide instantaneous predictions regarding their water levels. By offering a seamless user experience, the module empowers stakeholders to efficiently monitor and manage water resources, fostering a more sustainable and resource-conscious approach across diverse settings.

vi. Performance Evaluation Module

Calculates precision, recall, validation loss, and accuracy. Evaluating the performance of the image classification model is essential to ensure its reliability and effectiveness in practical applications. This module assesses various performance metrics to measure the model's performance objectively. The module compares the model's predictions with the ground truth

labels from a separate validation dataset. Metrics such as precision, recall, validation loss, and accuracy are calculated to quantify the model's performance across different aspects. The output of this module is a comprehensive assessment of the model's performance, highlighting its strengths and areas for improvement. This evaluation aids in fine-tuning the model and optimizing its performance for real-world deployment.

Finally, the Performance Evaluation Module undertakes a comprehensive assessment of the image classification model to gauge its efficacy and reliability. Leveraging established metrics such as precision, recall, validation loss, and accuracy, the module objectively evaluates the model's performance against a separate validation dataset. By quantifying the model's strengths and areas for improvement, the module informs iterative refinement efforts aimed at optimizing the model's performance for real-world deployment. Through meticulous evaluation and continuous enhancement, the module ensures that the image classification system maintains its effectiveness and relevance in addressing the evolving needs of stakeholders across various domains.

3.4.2 Non-Functional Requirements

i. Performance

Performance is critical for ensuring the system efficiently processes water bottle images and provides timely and accurate predictions of their water levels. The system must optimize image processing algorithms to handle classification tasks swiftly while minimizing computational overhead. It should support concurrent processing of multiple image inputs, ensuring that real-time results are delivered with minimal latency. Establishing and maintaining performance benchmarks is essential to guarantee that the system meets or exceeds response time targets under varying load conditions, thereby enhancing user satisfaction and operational efficiency.

ii. Scalability

Scalability is vital as the system may be deployed across diverse environments, including offices, public spaces, and industrial facilities. It must seamlessly scale to accommodate a growing number of users accessing the system concurrently and handle larger datasets of water bottle images efficiently. The system should be designed with scalability in mind, allowing for geographic expansion into new locations without compromising performance or reliability. Implementing effective load balancing strategies ensures that computational tasks are distributed optimally across multiple servers or resources, maintaining consistent performance as demand fluctuates.

iii. Reliability

Reliability is crucial to ensure uninterrupted operation, particularly in critical applications like water resource management. The system must demonstrate high resilience to hardware failures, network disruptions, and software errors. Built-in mechanisms for automatic recovery and failover should be in place to minimize downtime and ensure continuous service availability. Rigorous testing under diverse operating conditions is essential to validate the system's stability and robustness. By prioritizing reliability, the system can maintain operational continuity and uphold user trust in its performance, even during unforeseen challenges.

iv. Accuracy

The accuracy of the image classification model is paramount in categorizing water bottles into full, half-full, or overflowing states reliably. Extensive testing and validation procedures are necessary to verify the model's precision and recall rates across different scenarios. Ongoing monitoring and refinement efforts ensure that the model consistently delivers accurate predictions, adapting to evolving data patterns and user requirements. By focusing on accuracy, the system enhances its effectiveness in real-world applications, supporting informed decision-making and operational efficiency in water level management tasks.

v. Security

Security measures are essential to safeguard user privacy and data integrity throughout the system's operations. The system should implement robust encryption protocols to protect sensitive information, ensuring secure transmission and storage of images. Role-based access controls and authentication mechanisms should be employed to prevent unauthorized access to the system. Regular security audits and updates mitigate potential vulnerabilities, maintaining compliance with industry standards and regulations. By prioritizing security, the system enhances user confidence and mitigates risks associated with data breaches or malicious activities.

vi. Usability

Usability is crucial for ensuring that the system interface is intuitive and accessible to users with varying levels of technical expertise. Clear instructions and user-friendly features guide users through the process of uploading images and interpreting classification results effectively. The interface should be responsive and adaptable to different devices and screen sizes, accommodating diverse user preferences and operational environments. By prioritizing

usability, the system enhances user satisfaction and adoption rates, promoting seamless interaction and productive use of its functionalities.

Summary

The "Water Bottle Image Classification Model" project entails both functional and non-functional requirements to ensure its effectiveness and reliability. Functionally, the system comprises modules such as data collection for acquiring labeled water bottle images, preprocessing for standardizing inputs, data augmentation for enhancing model generalization, image classification for categorizing bottles, Gradio deployment for real-time predictions, and performance evaluation for assessing model accuracy.

Non-functionally, the system must exhibit high performance, scalability to handle increasing demands, reliability for uninterrupted operation, accuracy in classification, robust security measures to protect user data, performance monitoring for optimization, and compliance with regulatory standards for data privacy and security. These requirements collectively ensure the system's ability to efficiently automate water bottle level monitoring, contributing to improved resource management in various settings.

Chapter 4

SYSTEM DESIGN

The system design for the Water Bottle Classification with Water Level project outlines the architecture and functionality of the automated classification system. It integrates data collection, preprocessing, machine learning model development, and user interface design to facilitate accurate categorization of water bottles based on their water levels. By meticulously orchestrating these components, the system aims to streamline the classification process while providing a user-friendly experience for stakeholders.

4.1 Dataset Description

Split	Total Images	Training Images	Training Images per Class	Validation Images	Validation Images per Class
80% Training - 20% Validation	1200	400	7200	6000	2000
70% Training - 30% Validation	2160	720	7200	5040	1680
50% Training - 50% Validation	3600	1200	7200	3600	1200

Table 4.1 Dataset

The dataset consists of images collected from various water level monitoring situations, captured under different conditions to represent three distinct classes of water levels: full, half, and overflow. The primary purpose of this dataset is to support the development and evaluation of machine learning models for accurately classifying these water levels. Each class is equally represented, ensuring a balanced dataset for training and validation.

Named the "Water Level Classification Dataset," it includes a total of 7200 images. To ensure comprehensive model training and evaluation, the dataset is divided into training and validation sets using three different splits: 80%-20%, 70%-30%, and 50%-50%. For the 80%-20% split, 6000 images are used for training, with each class (full, half, overflow) having

2000 images, and 1200 images are used for validation, with each class having 400 images. For the 70%-30% split, 5040 images are used for training, with each class having 1680 images, and 2160 images are used for validation, with each class having 720 images. For the 50%-50% split, both the training and validation sets contain 3600 images, with each class having 1200 images.

The images in this dataset are available in multiple formats, including JPG, JPEG, and PNG. These formats are widely supported by image processing and machine learning frameworks, making the dataset versatile and easy to use. The consistent size of the images ensures they are suitable for training machine learning models without requiring extensive preprocessing.

The different splits provided in the dataset offer flexibility for various experimental setups. The 80%-20% split is ideal when a large amount of training data is available, ensuring the model has enough data to learn effectively. The 70%-30% split balances the amount of training data with a larger validation set for more robust evaluation. The 50%-50% split is useful for thoroughly assessing the model's generalization ability, as it provides an equal amount of data for both training and validation.

Overall, the "Water Level Classification Dataset" is a well-structured and balanced dataset, with equal representation across the three classes. Its availability in common image formats and its division into different training and validation splits make it suitable for a wide range of machine learning experiments aimed at classifying water levels accurately.

4.2 System Architecture

The system architecture for the Water Bottle Classification Model is designed to manage data flow, model training, and user interaction efficiently. The architecture consists of several key components: Data Collection, Pre-processing, Training Model, Model Generation, Image Retrieval System, and User Interface. Each component is critical to the system's overall functionality, ensuring a seamless and robust water level classification process.

i. **Data Collection**

   Data Collection involves gathering images of water bottles from various sources. These images can be obtained through user uploads, automated systems such as cameras installed in water dispensers, or datasets compiled specifically for this purpose. The collected images serve as the primary input for the classification model.

ii. **Pre-processing**

Once the images are collected, they undergo Preprocessing. This step includes resizing the images to a standard dimension, normalizing pixel values, and performing data augmentation to enhance model robustness. Preprocessing ensures that the images are consistent and of high quality, which is essential for accurate model training and classification.

iii. Training Model

In the Training Model phase, convolutional neural network (CNN) models are trained using the preprocessed images. Models such as DenseNet, MobileNetV2, InceptionV3, and ResNet50V2 are employed to learn and classify different water levels in the bottles. The training process involves feeding the preprocessed images into the models and adjusting the model parameters to minimize classification errors.

iv. Model Generation

After training, the models are saved and prepared for deployment in the Model Generation step. This involves exporting the trained models in a format suitable for loading and inference during real-time classification tasks. The models are stored in a centralized location, making them accessible for the next phase.

v. Image Retrieval System

The Image Retrieval System is responsible for handling user inputs and retrieving classification results. When an image is uploaded through the User Interface, the retrieval system preprocesses the image, feeds it into the appropriate model, and retrieves the classification results. This system ensures efficient and accurate classification of water levels.

vi. User Interface

The User Interface is the front-end component where users interact with the system. Users can upload images of water bottles, and the interface displays the retrieved classification results. It provides a user-friendly experience, allowing users to view the predicted water level and related information. The interface also includes visualizations, such as accuracy charts, to present model performance metrics.

In summary, the system architecture integrates data collection, pre-processing, model training, and user interaction components into a cohesive workflow. This design ensures the efficient classification of water levels in bottles, providing users with accurate and timely results. The architecture leverages robust CNN models and a streamlined data flow to achieve high classification performance and usability.

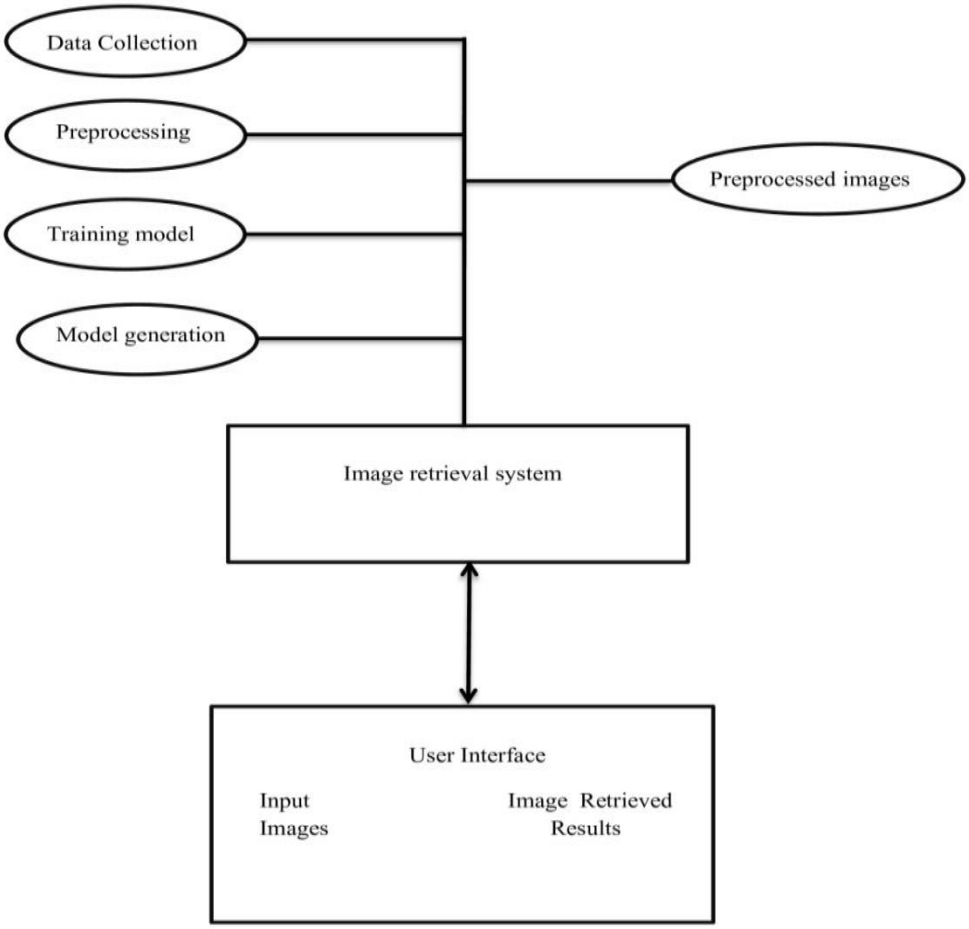

Fig 4.1 Block diagram

In the Water Bottle Classification with Water Level project, the workflow begins with the Data Collection phase, where a diverse dataset of water bottle images is gathered. These images, along with their corresponding water level labels, serve as the foundation for training the classification model. Once collected, the images undergo preprocessing to standardize their format and quality. This Preprocessing step involves tasks like resizing, cropping, and normalization to ensure uniformity among the images. The output of this stage is a set of preprocessed images ready for training.

Subsequently, the preprocessed images are fed into the Training Model phase, where a classification model is developed. This involves the use of machine learning techniques, particularly Convolutional Neural Networks (CNNs), to learn the features and patterns indicative of different water levels in the images. Through an iterative process, the model

refines its understanding of water bottle characteristics and becomes adept at classifying them accurately.

Once the model is trained, it enters the Model Generation phase, where it is ready for deployment. In the Image Retrieval System, users interact with the trained model through a user-friendly interface. Here, users can input new images of water bottles, and the system retrieves the corresponding water level classification results. This interface acts as a bridge between the trained model and end-users, providing a seamless experience for obtaining real-time classification results.

The Water Bottle Classification with Water Level project follows a systematic workflow, starting from data collection and preprocessing, leading to model training and generation, and culminating in a user-friendly interface for retrieving classification results. Each phase plays a vital role in the overall success of the project, ensuring accurate and efficient classification of water bottle images based on their water levels.

4.3 Context Diagram

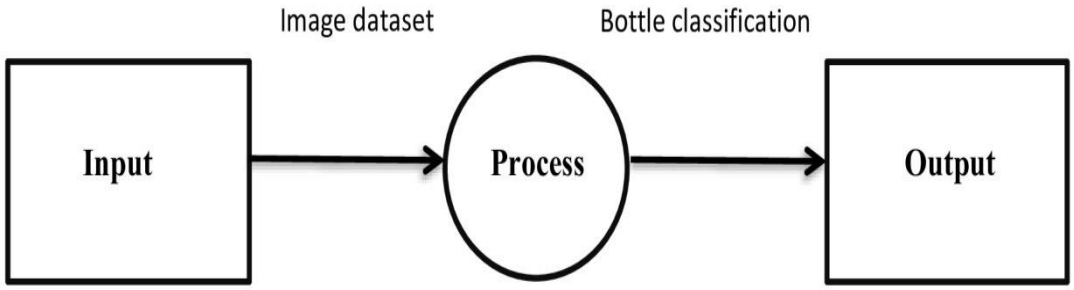

Fig 4.1 Data flow diagram(level 0)

The Level 0 Data Flow Diagram (DFD) for the Water Bottle Classification with Water Level project consists of three main components: Input, Process, and Output. In the middle of these components lie two crucial aspects: the image dataset and bottle classification.

Starting with the Input component, the system receives input data in the form of an image dataset. This dataset comprises a collection of labeled images of water bottles, each associated with its respective water level status (full, half, or overflowing). These images serve as the raw material for the classification process, providing the necessary input for training and testing the classification model.

Moving to the Process component, the central operation performed by the system is bottle classification. This process involves analyzing the input images using machine learning algorithms, specifically Convolutional Neural Networks (CNNs), to classify the water bottles into predefined categories based on their water levels. The classification model learns to identify patterns and features within the images that distinguish full, half-full, and overflowing bottles, enabling accurate categorization.

Finally, the Output component represents the results of the bottle classification process. Once the classification is completed, the system generates output in the form of classification results. These results indicate the predicted water level status for each input image, providing valuable insights for users or stakeholders. The output enables users to make informed decisions regarding water resource management based on the classification outcomes.

The Level 0 DFD illustrates the flow of data and processes within the Water Bottle Classification with Water Level system. It highlights the crucial components of input data (image dataset), the central process of bottle classification, and the resulting output (classification results). Together, these components form the core functionality of the system, enabling efficient and accurate classification of water bottles based on their water levels.

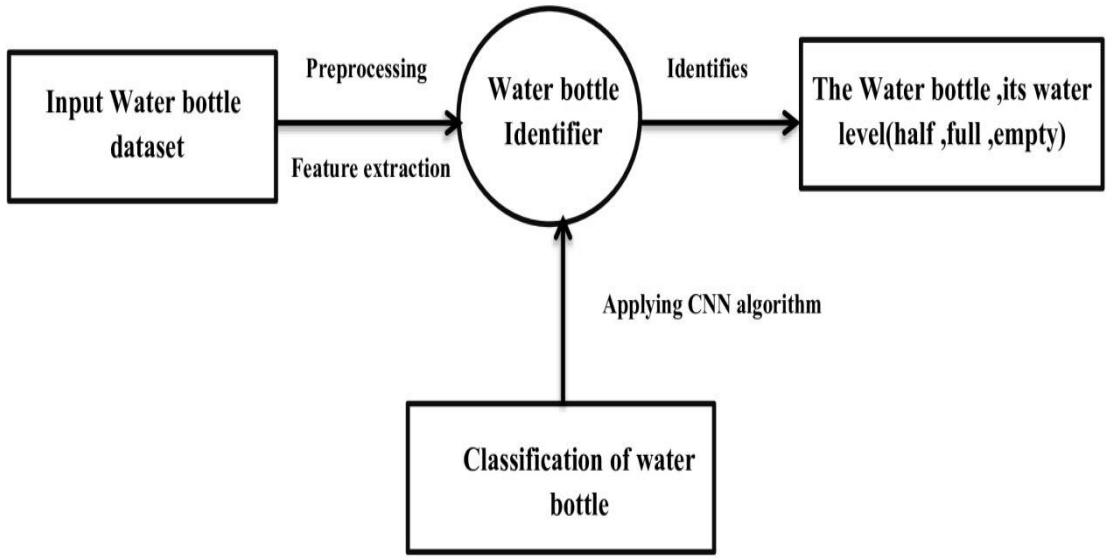

Fig 4.2 Data Flow Diagram (level 1)

In the Level 1 Data Flow Diagram (DFD) for the Water Bottle Classification with Water Level project, there are two main components: the Input Water Bottle Dataset and the Water Bottle Identifier connected to the Classification of Water Bottle process. These components play crucial roles in the system's operation, facilitating the classification of water bottles based on their water levels: half-full, full, or empty.

Beginning with the Input Water Bottle Dataset component, this represents the repository of labeled images containing various examples of water bottles in different states. Each image in the dataset is tagged with its corresponding water level status, distinguishing between half-full, full, or empty bottles. This dataset serves as the primary source of information for training the classification model, providing a diverse range of examples for the system to learn from.

Moving to the Water Bottle Identifier, this component acts as an intermediary between the input dataset and the classification process. Its role is to extract relevant features and characteristics from the input images, enabling the system to identify and distinguish water bottles within the images accurately. The identifier employs image processing techniques and pattern recognition algorithms to analyze the visual cues indicative of water bottles, preparing the data for further classification.

Connected to the Water Bottle Identifier is the Classification of Water Bottle process, which forms the core functionality of the system. Here, the system utilizes machine learning

algorithms, such as Convolutional Neural Networks (CNNs), to classify the identified water bottles into distinct categories based on their water levels: half-full, full, or empty. By analyzing the features extracted by the identifier and leveraging the knowledge learned from the input dataset, the classification process assigns a water level status to each detected water bottle with a high degree of accuracy.

The Level 1 DFD illustrates the flow of data and processes involved in classifying water bottles based on their water levels. It highlights the critical components of the input dataset containing labeled images, the Water Bottle Identifier responsible for extracting relevant features, and the Classification of Water Bottle process, which performs the actual classification based on the identified features. Together, these components enable the system to accurately categorize water bottles, providing valuable insights for various applications, such as water resource management.

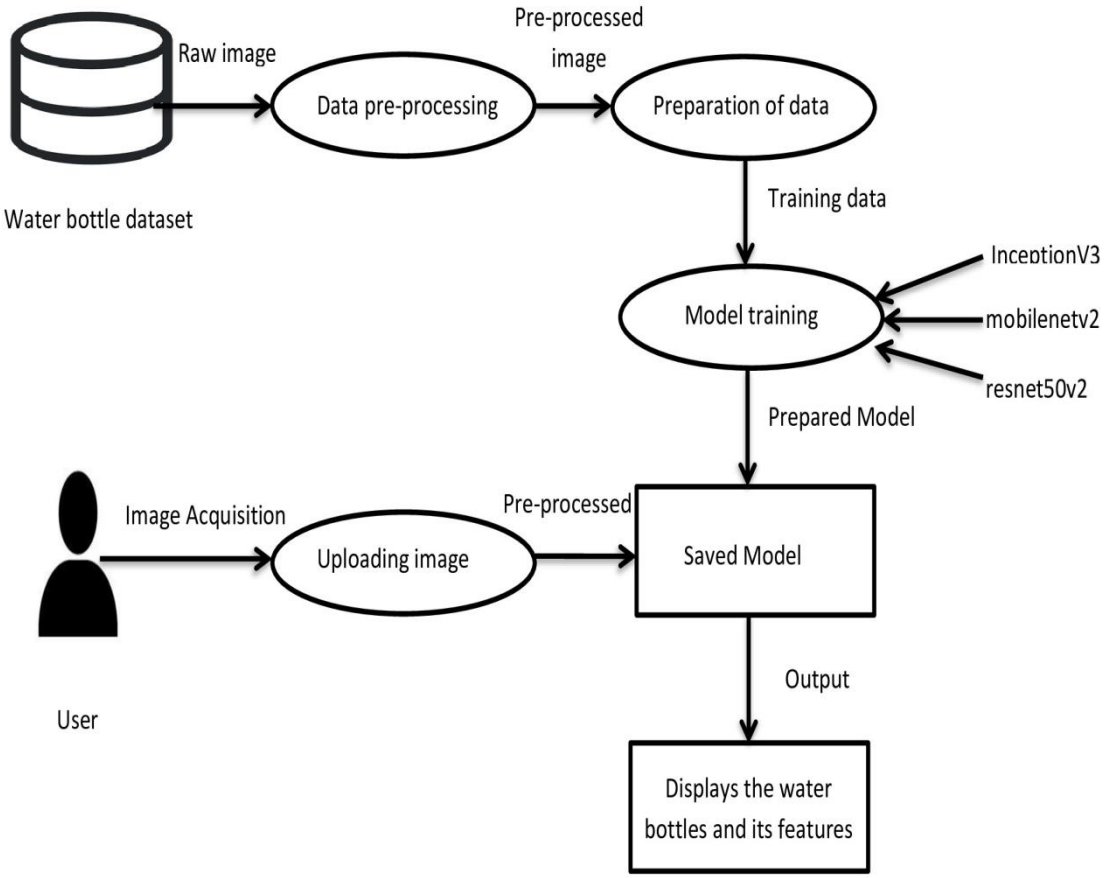

Fig 4.3 Data Flow Diagram (level 2)

In the Level 2 Data Flow Diagram (DFD) depicting the workflow for the water bottle dataset, the database serves as the foundational repository of information. This dataset flows into the data preprocessing stage, where it undergoes various operations to clean, transform, and prepare it for further analysis. The preprocessing stage ensures that the data is in a suitable format and quality for subsequent stages of the process.

Following data preprocessing, the prepared data moves into the model training phase. Here, the dataset is utilized to train machine learning models aimed at specific objectives related to the water bottle dataset. This phase encompasses the utilization of various algorithms such as Convolutional Neural Networks (CNN), ResNet16, and potentially others. Each algorithm contributes its unique approach and capabilities to the training process, allowing for comprehensive exploration of the dataset's patterns and characteristics.

As the model training concludes, the trained models are saved for future use. This step is crucial for preserving the trained parameters and architecture, enabling quick and efficient deployment in real-world scenarios. The saved models serve as repositories of learned knowledge, encapsulating the insights gained from the training process.

On the user-facing side, individuals interact with the system by uploading images related to water bottles. These images are then directed towards the saved model, which employs its learned knowledge to analyze and process the uploaded images. The saved model acts as the intermediary between user inputs and the desired outputs, leveraging the trained algorithms to provide accurate and meaningful results.

Finally, the results generated by the saved model are displayed to the user. These results could include various insights or classifications derived from the analysis of the uploaded images. By presenting these outcomes to the user, the system facilitates informed decision-making and deeper understanding of the water bottle dataset's characteristics. Overall, the DFD illustrates a structured and coherent flow of data and operations, culminating in valuable outcomes for users interacting with the system.

4.5 Module Level Description

4.5.1. Data Collection Module

The Data Collection Module serves as the foundational step in developing the water bottle image classification model. Its primary function is to acquire a labeled dataset of water bottle images, crucial for training and evaluating the model's performance. This module collects

images from diverse sources, ensuring a representative sample of water bottles in varying states—full, half-full, or overflowing. Whether sourced from online repositories or captured through manual efforts, the collected images are meticulously labeled to denote their corresponding water levels. By assembling this comprehensive dataset, the module lays the groundwork for subsequent stages, enabling the model to learn and classify water bottles accurately in real-world scenarios. Collects labeled dataset of water bottle images.Purpose is that module serves as the initial step in building the image classification model. Gathering a labelled dataset is crucial as it provides the necessary data for training and testing the model. Operation is module may obtain images from various sources such as online repositories or by capturing images manually. These images are then labeled with their corresponding water levels (full, half, or overflowing) either through automated tagging or manual annotation processes. output of this module is a labeled dataset containing water bottle images alongside their respective water level labels. This dataset forms the foundation for training the image classification model.

4.5.2 Pre-processing Module

Standardizes and prepares images for model input. Purpose is raw images often vary in size, resolution, and quality, which can hinder the performance of the model. The preprocessing module addresses these issues by standardizing the images and preparing them for input into the model. Operation for preprocessing techniques such as resizing, cropping, and normalization are applied to the raw images. Resizing ensures that all images have the same dimensions, while cropping removes irrelevant parts. Normalization adjusts pixel values to a common scale, improving the model's convergence during training. Output module produces preprocessed images that are uniform in size, format, and quality, ready to be fed into the image classification model.

Following data collection, the Preprocessing Module undertakes the essential task of refining and standardizing the acquired images to optimize their suitability for model input. Raw images often exhibit disparities in size, resolution, and quality, which can hinder the model's performance during training. To address these inconsistencies, the preprocessing module applies various techniques such as resizing, cropping, and normalization. These operations ensure uniformity among the images, enhancing their compatibility with the model architecture. By preparing the images in a standardized format, the module streamlines the

subsequent training process, facilitating smoother convergence and improved model performance.

4.5.3 Data Augmentation Module

Generates augmented data to enhance model generalization. Purpose of Data augmentation increases the diversity of the dataset, helping the model generalize better to unseen data and reducing the risk of overfitting. Operation of Various augmentation techniques such as rotation, flipping, zooming, and adding noise are applied to the preprocessed images. These transformations create new variations of the original images without changing their underlying semantics. The module produces an augmented dataset that includes both the original and augmented images. This expanded dataset provides the model with more examples to learn from, improving its ability to classify water bottle images accurately.

The Data Augmentation Module plays a pivotal role in enhancing the model's robustness and generalization capabilities by generating augmented data. Data augmentation is a crucial technique employed to diversify the training dataset, exposing the model to a broader spectrum of scenarios. Leveraging transformations such as rotation, flipping, and introducing noise, the module creates additional variations of the original images. These augmented images serve to enrich the training data, mitigating the risk of overfitting and equipping the model to handle diverse real-world conditions. By augmenting the dataset with a diverse array of examples, the module fosters a more resilient and adaptable image classification model.

4.5.4 Image Classification Module

Classifies water bottles into full, half, or overflowing. The primary objective of the project is to accurately classify water bottles based on their water levels. This module achieves this by leveraging machine learning techniques, particularly Convolutional Neural Networks (CNNs). The module takes preprocessed or augmented images as input and extracts features using a CNN architecture. These features are then fed into fully connected layers for classification. The model learns to differentiate between full, half-full, and overflowing water bottles based on the patterns it identifies in the input images. The output of the module is the predicted water level category for each input image, indicating whether the bottle is full, half-full, or overflowing.

At the heart of the project lies the Image Classification Module, which embodies the core functionality of the system. This module employs machine learning techniques, particularly Convolutional Neural Networks (CNNs), to classify water bottles into distinct categories based on their water levels. Upon receiving preprocessed or augmented images as input, the module extracts intricate features using the CNN architecture and employs fully connected layers for classification. Through an iterative learning process, the model discerns subtle patterns within the images, enabling it to accurately classify water bottles as full, half-full, or overflowing. The output of this module—the predicted water level categories—serves as the foundation for informed decision-making in resource management scenarios.

4.5.5 Gradio Deployment Module

Deploys the trained model using Gradio for real-time classification. Once the image classification model is trained, it needs to be deployed in a user-friendly manner for practical use. Gradio simplifies this deployment process by providing an intuitive interface for users to interact with the model.The trained model is integrated with Gradio, allowing users to upload images of water bottles through a web interface. The model then provides real-time predictions on the water levels of the bottles.The module enables users to receive instant feedback on the water levels of the bottles they input, facilitating efficient monitoring and management of water resources in real-world scenarios.

Once the image classification model is trained and validated, the Gradio Deployment Module facilitates its seamless integration into real-world environments through a user-friendly interface. Gradio simplifies the deployment process by providing an intuitive web interface for users to interact with the model in real-time. Users can effortlessly upload images of water bottles via the interface, prompting the model to provide instantaneous predictions regarding their water levels. By offering a seamless user experience, the module empowers stakeholders to efficiently monitor and manage water resources, fostering a more sustainable and resource-conscious approach across diverse settings.

4.5.6 Performance Evaluation Module

Calculates precision, recall, validation loss, and accuracy. Evaluating the performance of the image classification model is essential to ensure its reliability and effectiveness in practical applications. This module assesses various performance metrics to measure the model's performance objectively. The module compares the model's predictions with the ground truth labels from a separate validation dataset. Metrics such as precision, recall, validation loss,

and accuracy are calculated to quantify the model's performance across different aspects. The output of this module is a comprehensive assessment of the model's performance, highlighting its strengths and areas for improvement. This evaluation aids in fine-tuning the model and optimizing its performance for real-world deployment.

Finally, the Performance Evaluation Module undertakes a comprehensive assessment of the image classification model to gauge its efficacy and reliability. Leveraging established metrics such as precision, recall, validation loss, and accuracy, the module objectively evaluates the model's performance against a separate validation dataset. By quantifying the model's strengths and areas for improvement, the module informs iterative refinement efforts aimed at optimizing the model's performance for real-world deployment. Through meticulous evaluation and continuous enhancement, the module ensures that the image classification system maintains its effectiveness and relevance in addressing the evolving needs of stakeholders across various domains.

Summary

The "Water Bottle Image Classification Model" project comprises several interconnected modules designed to automate and enhance the process of monitoring water bottle levels. The Data Collection Module initiates the project by sourcing a diverse dataset of labeled water bottle images, essential for training the subsequent classification model. Once collected, the images undergo preprocessing in the Preprocessing Module, where they are standardized and optimized for model input through techniques like resizing and normalization.

The Data Augmentation Module then enriches the dataset by generating augmented data, ensuring the model's robustness and generalization capabilities. The core functionality lies in the Image Classification Module, where machine learning techniques, particularly Convolutional Neural Networks, are employed to classify water bottles into full, half, or overflowing categories based on extracted features from the images.

Chapter 5

DETAILED DESIGN

The system design for the Water Bottle Classification with Water Level project encompasses the architecture and structure of the software solution aimed at automating the classification of water bottles based on their water levels. It involves the integration of various components, including data collection, preprocessing, machine learning model development, and user interface design, to create a robust and efficient system. Through meticulous planning and implementation, the system design ensures seamless interaction between different modules, enabling accurate classification and recognition of water bottle images while providing a user-friendly experience for stakeholders.

5.1 Use Case Diagram

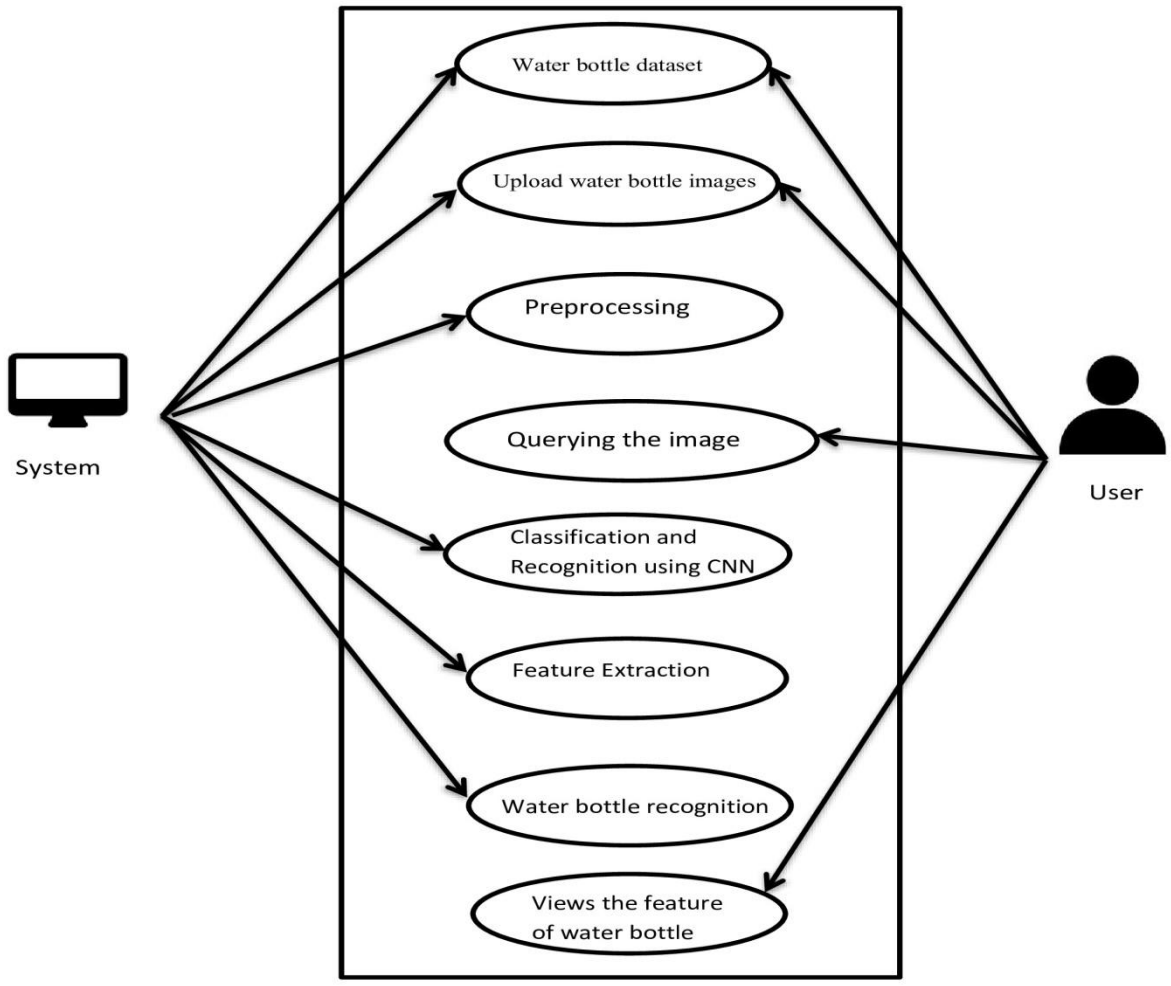

Fig 5.1 Use Case Diagram

In the Use Case Diagram for the Water Bottle Classification with Water Level project, two main actors are depicted: the System and the User. Each actor interacts with the system through various use cases, contributing to the overall functionality of the project.

Starting with the System, it comprises several essential use cases that facilitate the process of water bottle classification. Firstly, the System connects to the Water Bottle Dataset, indicating its access to a repository of labeled images containing examples of water bottles in different states: half-full, full, or empty. This dataset serves as the foundation for training and testing the classification model, providing a diverse range of examples for the system to learn from.

Next, the System allows users to Upload Water Bottle Images, enabling them to input new images for classification. Upon receiving these images, the System initiates a Preprocessing step, where the images undergo standardization and refinement to optimize their suitability for classification. Techniques such as resizing, cropping, and normalization are applied to ensure uniformity and improve the accuracy of classification.

Subsequently, the System performs Classification and Recognition using CNNs (Convolutional Neural Networks), which constitute advanced machine learning algorithms capable of extracting intricate features from images. Leveraging the knowledge learned from the Water Bottle Dataset and the preprocessed images, the CNNs analyze the features of the input images and classify the water bottles into distinct categories: half-full, full, or empty, with a high degree of accuracy.

Additionally, the System includes a Feature Extraction process, where relevant characteristics of the water bottles are identified and extracted from the input images. These features serve as crucial indicators for classification and contribute to the overall accuracy of the system's predictions. Finally, the System performs Water Bottle Recognition, accurately identifying and categorizing the water bottles based on their extracted features and the classification model's predictions.

On the other hand, the User interacts with the system through a set of complementary use cases. Firstly, the User can Upload Water Bottle Images, similar to the system, providing additional input for classification. Additionally, the User can Query the Image, enabling them

to search for specific water bottle images within the dataset based on predefined criteria such as water level status or other features.

Furthermore, the User can View the Feature of Water Bottle, allowing them to inspect the extracted features of water bottles from the dataset. This use case provides users with insights into the characteristics used for classification, fostering a deeper understanding of the system's operations and enhancing transparency.

In summary, the Use Case Diagram illustrates the interactions between the System and the User, highlighting the various functionalities and capabilities of the Water Bottle Classification with Water Level project. Together, these use cases facilitate efficient and accurate classification of water bottles based on their water levels, catering to the needs of both system administrators and end-users alike.

5.2 Sequence Diagram

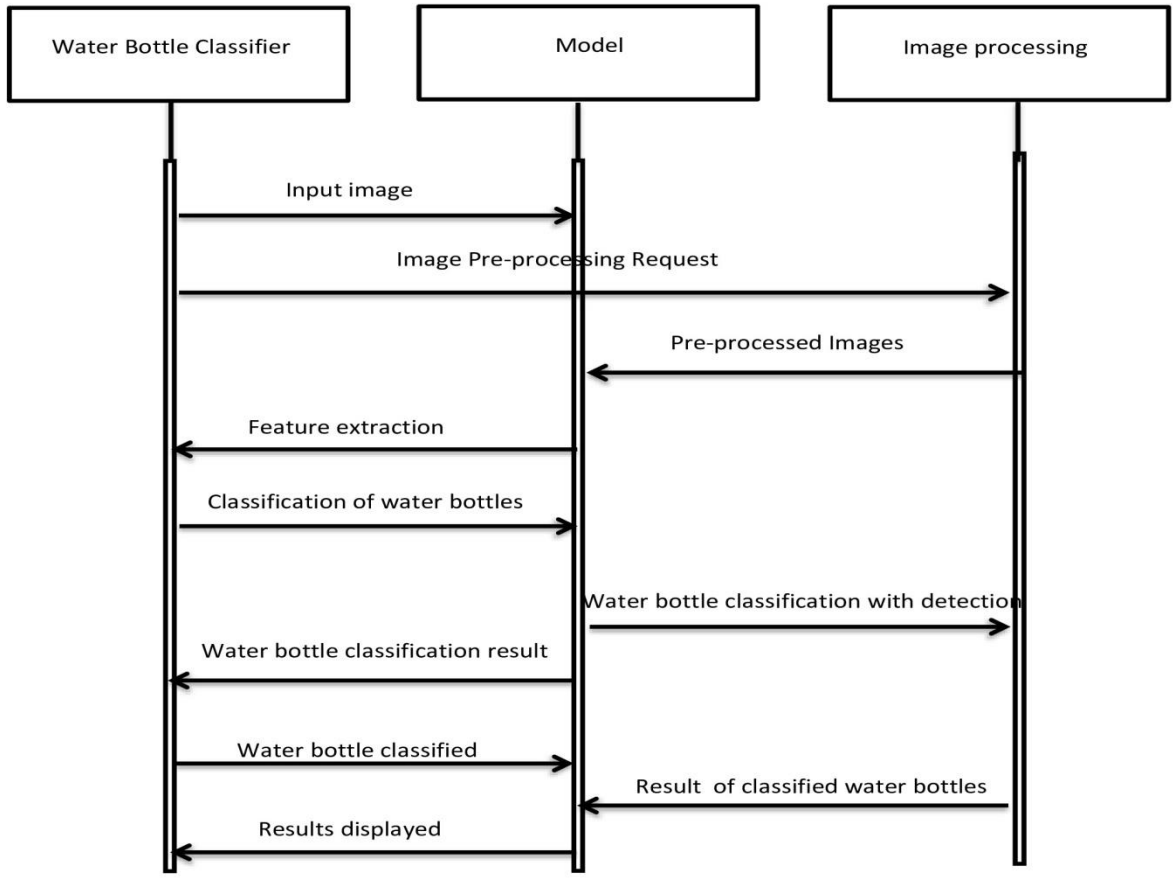

Fig 5.2 Sequence Diagram

The sequence diagram in Fig 5.3 illustrates the interaction between three primary components in the water bottle classification system: the Water Bottle Classifier, the Model, and the Image Processing unit. Each of these components plays a crucial role in the accurate classification of water levels in bottles. The diagram outlines the sequence of operations from the input of an image to the display of classification results. Here is a detailed explanation of the processes and interactions:

Water Bottle Classifier

1. **Input Image**: The process begins when a user or system inputs an image of a water bottle into the Water Bottle Classifier. This input image is the starting point for the classification process.
2. **Image Pre-processing Request**: The Water Bottle Classifier sends a request to the Image Processing unit to preprocess the image. This request is necessary to ensure the image is in the optimal format and quality for further analysis.
3. **Feature Extraction**: After receiving the preprocessed image from the Image Processing unit, the Water Bottle Classifier extracts relevant features from the image. These features are essential for identifying different levels of water in the bottle.
4. **Classification of Water Bottles**: Using the extracted features, the classifier performs the classification of the water bottle. It assigns the image to one of the predefined categories (e.g., full, half, overflow).
5. **Water Bottle Classification Result**: The result of the classification is generated, indicating the identified water level in the bottle.
6. **Water Bottle Classified**: This classification result is then prepared for display to the user.
7. **Results Displayed**: Finally, the Water Bottle Classifier displays the classification result to the user, completing the process.

Model

1. **Image Pre-processing Request**: The Model component receives the pre-processing request from the Water Bottle Classifier.
2. **Pre-processed Images**: The Model sends the request to the Image Processing unit and waits for the preprocessed images to be returned.
3. **Water Bottle Classification with Detection**: The Model performs the actual classification task using the preprocessed images. It uses trained algorithms to detect the water level in the bottle.

4. **Result of Classified Water Bottles**: The classification results are then sent back to the Water Bottle Classifier for final display to the user.

Image Processing

1. **Image Pre-processing Request**: The Image Processing unit receives the request from the Model to preprocess the input image.
2. **Pre-processed Images**: The Image Processing unit processes the image to ensure it is in the correct format and quality for classification. This may include resizing, normalization, and other image enhancement techniques.
3. **Pre-processed Images Sent**: The preprocessed image is then sent back to the Model component.

By following these steps, the system ensures that the input images are accurately processed and classified. The clear delineation of responsibilities among the Water Bottle Classifier, the Model, and the Image Processing unit helps maintain the efficiency and accuracy of the classification process. This sequence diagram provides a comprehensive overview of the interactions and data flow necessary for the water bottle classification system to function effectively.

5.3 Activity Diagram

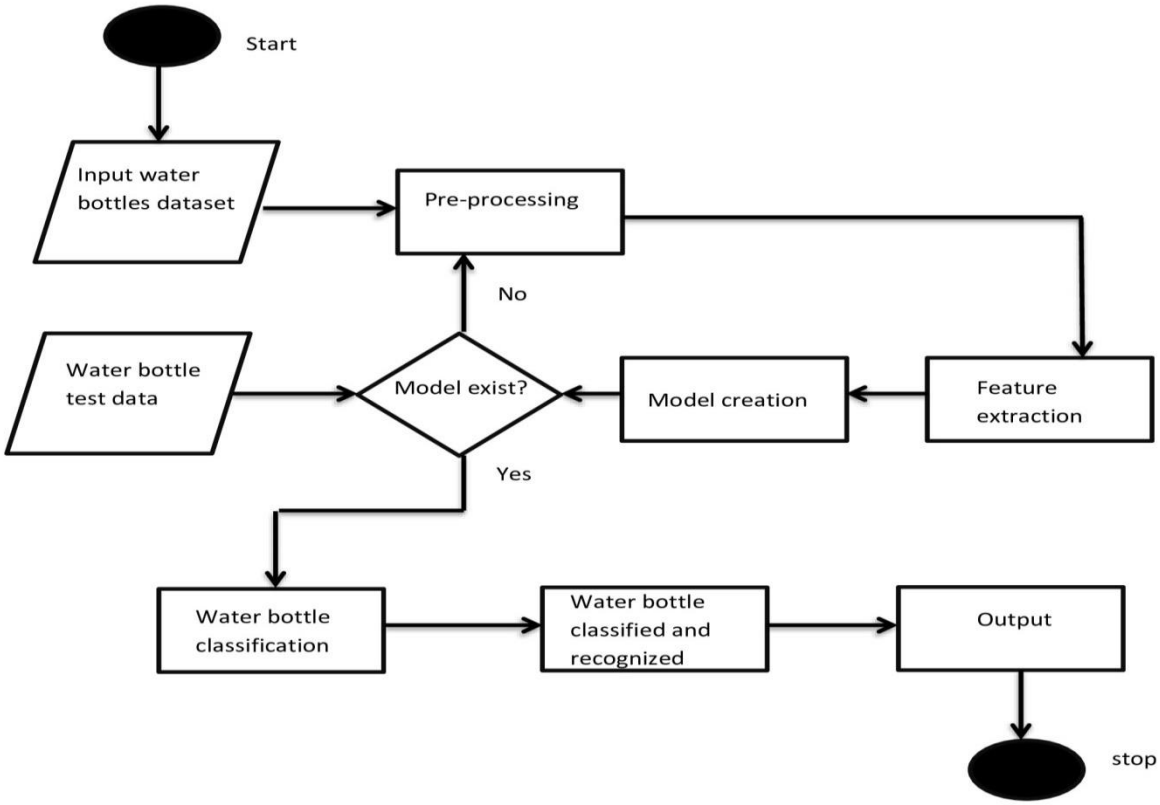

Fig 5.3 Activity Diagram

The Activity Diagram for the Water Bottle Classification with Water Level project outlines the sequential flow of activities from the initial data input to the final output. It begins with the Start activity, representing the initiation of the classification process. The first step is to access the Input Water Bottles Dataset, indicating the retrieval of labeled images containing examples of water bottles with various water levels.

From there, the dataset is passed to the Preprocessing activity, where the images undergo standardization and refinement to prepare them for feature extraction. Techniques such as resizing, cropping, and normalization are applied to ensure consistency and enhance the quality of the images.

Subsequently, the preprocessed images are subjected to Feature Extraction, where relevant characteristics and patterns are identified and extracted. These features serve as essential inputs for the Model Creation activity, where a classification model is developed using machine learning algorithms, such as Convolutional Neural Networks (CNNs).

The next decision point in the diagram is the Model Existence check. If a model already exists, the process proceeds to the Water Bottle Classification activity, where the model is applied to classify the water bottles based on their extracted features. If no model exists, the process loops back to the Preprocessing activity to refine the images further and repeat the feature extraction and model creation steps.

Once the water bottles are classified, the process continues to the Water Bottle Classification and Recognition activity, where the classified bottles are recognized and categorized based on their water levels. Finally, the output of the classification process is generated, indicating the predicted water levels for each input water bottle.

The End activity marks the conclusion of the classification process, signifying that all activities have been completed successfully. In summary, the Activity Diagram provides a visual representation of the sequential flow of activities involved in the water bottle classification process, from data input to output generation, with provisions for model creation and reiteration as needed.

Summary

The Water Bottle Classification with Water Level project involves two main diagrams: an Activity Diagram and a Use Case Diagram. The Activity Diagram depicts the sequential flow of activities, starting with accessing the input water bottle dataset, followed by preprocessing, feature extraction, model creation, classification, and recognition, ultimately leading to output generation. It also includes decision points such as model existence checks to ensure efficient processing. On the other hand, the Use Case Diagram illustrates the interactions between the system and users, highlighting functionalities such as uploading water bottle images, querying image data, and viewing features. Together, these diagrams provide a comprehensive overview of the project's workflow, from data input to classification results, catering to the needs of both system administrators and end-users.

Chapter 6

IMPLEMENTATION

Chapter 6 focuses on the practical implementation of machine learning in our project. We begin with Section 6.1, covering the setup of essential packages with installation instructions and references. Section 6.2 details our experimental processes, including training, testing, and parameter configuration. In Section 6.3, we analyze our results through snapshots and detailed discussions. The chapter concludes with a concise summary of our findings and achievements.

6.1 Experimental Setup

Packages

i. **TensorFlow and Keras**

TensorFlow is an open-source machine learning framework developed by Google, renowned for its robustness and versatility in building and deploying machine learning models. Keras, which operates as an API within TensorFlow, offers a user-friendly and intuitive interface for developing deep learning models. It facilitates rapid experimentation and simplifies the process of creating, training, and evaluating neural networks, making it accessible to both beginners and experts in machine learning.

ii. **ImageDataGenerator**

The ImageDataGenerator class from TensorFlow's Keras module is pivotal for handling image data during the training process. It generates batches of augmented image data, which helps improve the robustness and generalization of the model. This augmentation can include operations like rotation, shifting, flipping, and scaling of images in real-time, thus preventing overfitting and enhancing the model's performance on unseen data.

iii. **ResNet50V2**

ResNet50V2 is a pre-trained convolutional neural network (CNN) model available in TensorFlow's Keras applications. Known for its deep architecture with residual connections, ResNet50V2 excels in image classification tasks. These residual connections help mitigate the vanishing gradient problem, enabling the training of

very deep networks. The model, pre-trained on the ImageNet dataset, can be fine-tuned for specific classification tasks, such as identifying water levels in bottles.

iv. Model Class

The Model class in Keras provides a flexible and powerful way to define and manipulate neural network models. It allows for the specification of input and output tensors, facilitating the creation of complex model architectures tailored to specific tasks. This class is fundamental in structuring the neural network, integrating various layers, and compiling the model for training and evaluation.

v. Layers and Optimizers

Keras provides a wide array of layers and optimizers to build and train neural networks. The Dense layer is a fully connected layer, essential for creating the output layer in classification models. GlobalAveragePooling2D is used for reducing the spatial dimensions of the feature maps, typically before the dense layers. Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to zero during training. The Adam optimizer is a popular choice for training deep learning models due to its adaptive learning rate and efficient computation, which accelerates the convergence of the training process.

vi. Callbacks

Callbacks in Keras extend and customize the behavior of the model during training. The ModelCheckpoint callback saves the model at specified intervals, ensuring that the best version of the model is retained. EarlyStopping monitors a chosen metric and halts training if the metric stops improving, thereby preventing overfitting and saving computational resources. ReduceLROnPlateau lowers the learning rate when a monitored metric plateaus, helping the model to continue learning and improving.

vii. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface for generating plots, which is invaluable for visualizing training metrics, such as loss and accuracy, as well as displaying images and their corresponding classifications.

viii. Operating System Module

The os module in Python allows for interaction with the operating system. It provides functionalities for file and directory manipulation, such as reading, writing, and organizing files. This module is crucial for handling data files and directories in machine learning projects, ensuring efficient data management and processing.

ix. InceptionV3

InceptionV3 is another powerful pre-trained convolutional neural network model available in TensorFlow's Keras applications. Known for its efficient architecture and high accuracy, InceptionV3 is widely used for image classification tasks. It incorporates various architectural innovations, such as factorized convolutions and aggressive regularization, to improve performance and efficiency.

x. MobileNetV2

MobileNetV2 is designed specifically for mobile and embedded vision applications. It is a lightweight convolutional neural network that balances accuracy and computational efficiency, making it suitable for deployment on devices with limited resources. Despite its compact size, MobileNetV2 maintains high accuracy in image classification tasks, making it an excellent choice for real-time applications.

These components and libraries collectively provide a robust foundation for developing an efficient and accurate water bottle classification system using TensorFlow and Keras. By leveraging pre-trained models, data augmentation techniques, and various layers and optimizers, the system can be fine-tuned to achieve high performance in classifying water levels in bottles.

Installation Process

To install the necessary packages for running the provided code, follow these steps:

1. **TensorFlow and Keras**:
    - Install TensorFlow and Keras using pip:

      ```
      pip install tensorflow
      ```

2. **Matplotlib**:
    - Install Matplotlib using pip:

      ```
      pip install matplotlib
      ```

3. **Scikit-Learn (for confusion matrix)**:
    - Install Scikit-Learn using pip:

      ```
      pip install scikit-learn
      ```

4. **Additional packages** (already included in standard TensorFlow installation):
    - These packages like os, numpy, and image are typically included with standard Python installations and TensorFlow dependencies.
5. **Ensure access to Google Drive**:
    - Make sure you have access to Google Drive and adjust paths ('/content/drive/MyDrive/') as per your directory structure.
6. **Google Colab Setup**:
    - If you are using Google Colab, ensure that your Google Drive is mounted **Installation Process**

6.2 Experiments

1. Training and Validation Setup

- **Data Augmentation:** Experiment with different augmentation techniques such as rotation, width and height shifts, shear, zoom, and horizontal flip. Adjust the ranges of these transformations to see their impact on model generalization.
- **Batch Size:** Try varying the batch size used during training (batch_size parameter in train_generator and validation_generator). Larger batch sizes can sometimes improve training speed but may require more memory.
- **Number of Epochs:** Increase or decrease the number of epochs (epochs parameter in model.fit) to observe how training and validation accuracy and loss change over time. More epochs can potentially lead to better convergence, but be cautious of overfitting.
- **Learning Rate:** Modify the learning rate (learning_rate parameter in Adam optimizer) and observe its effect on training dynamics. Lower learning rates can help in fine-tuning the model more gradually.

2. Model Architecture

- **Fine-tuning Layers:** Adjust which layers of MobileNetV2 are trainable (layer.trainable = True). Experiment with freezing more or fewer layers to see how it affects training speed and model performance.
- **Dropout Rate:** Tune the dropout rate (Dropout layer in the model definition) to prevent overfitting. Higher dropout rates introduce more regularization but may reduce training accuracy.

- **Output Layer Configuration:** Depending on your classification task, adjust the number of units and activation function in the output layer (Dense layer with softmax activation for multiclass classification).

3. Evaluation Metrics

- **Confusion Matrix:** Analyze the confusion matrix to understand how well the model distinguishes between different classes. Experiment with different thresholds for interpreting the results.
- **Classification Report:** Generate and analyze the classification report (classification_report function from sklearn.metrics) to get detailed metrics such as precision, recall, and F1-score for each class.

4. Testing

- **Test Set Evaluation:** After training the model, evaluate its performance on a separate test set (test_dir directory). Adjust the test set images and labels to match your real-world scenario.

5. Optimization Strategies

- **Callbacks:** Experiment with different callbacks (ModelCheckpoint, EarlyStopping, ReduceLROnPlateau) and their parameters to improve training efficiency and model performance.

6. Performance Analysis

- **Visualizations:** Plot training and validation accuracy/loss (matplotlib plots) to visualize the model's learning progress and identify potential issues like overfitting or underfitting.

model	train	val	accuracy	loss	precision	recall	F1_score
Inception50v3	80%	20%	0.9500	0.1143	1.00	1.00	1.00
ResNet_50v3	80%	20%	0.9594	0.4191	1.00	1.00	1.00
Mobilenetv2	80%	20%	0.8987	0.3381	1.00	1.00	1.00

Table 6.1 80%-20% Dataset results

80-20 Dataset Split

i. **InceptionV3** When trained on 80% of the dataset and validated on 20%, the InceptionV3 model achieved a high accuracy of 0.9500. This means that it correctly classified 95% of the images in the validation set. The loss value was low at 0.1143, indicating that the model has effectively learned the data patterns with minimal error. The model also achieved perfect scores for precision, recall, and F1-score (1.00), suggesting that it made no false positive or false negative predictions.

ii. **ResNet50V2** The ResNet50V2 model outperformed InceptionV3 slightly, with an accuracy of 0.9594. Despite a higher loss value of 0.4191, it still achieved perfect precision, recall, and F1-scores of 1.00. This indicates that ResNet50V2 also had no misclassifications in the validation set, highlighting its robustness in image classification tasks.

iii. **MobileNetV2** MobileNetV2 achieved a lower accuracy of 0.8987 compared to the other models, indicating it correctly classified nearly 90% of the validation images. Its loss value was 0.3381. Like the other models, MobileNetV2 achieved perfect precision, recall, and F1-scores of 1.00, showing it made no errors in positive predictions.

Models	train	val	accuracy	loss	precision	recall	F1_score
Inception50 v3	70%	30%	0.9344	0.1786	1.00	1.00	1.00

ResNet_50v3	70%	30%	0.9781	0.1067	1.00	1.00	1.00
Mobilenetv2	70%	30%	0.8906	0.3067	1.00	1.00	1.00

Table 6.1 70%-30% Dataset results

70-30 Dataset Split

i. **InceptionV3** With a 70-30 train-validation split, the accuracy of InceptionV3 slightly decreased to 0.9344, indicating it correctly classified approximately 93% of validation images. The loss increased to 0.1786, reflecting slightly higher error. However, it maintained perfect precision, recall, and F1-scores of 1.00, demonstrating consistent reliability in prediction accuracy.

ii. **ResNet50V2** ResNet50V2 showed the best performance with the 70-30 split, achieving an impressive accuracy of 0.9781. The loss value was notably low at 0.1067, indicating minimal error in predictions. As with the other splits, it maintained perfect precision, recall, and F1-scores of 1.00, confirming its high accuracy and reliability.

iii. **MobileNetV2** MobileNetV2 had a validation accuracy of 0.8906, slightly lower than with the 80-20 split. The loss value was 0.3067, showing some improvement compared to the 80-20 split. Despite the lower accuracy, MobileNetV2 still achieved perfect precision, recall, and F1-scores of 1.00, maintaining its consistency in error-free positive predictions.

Models	train	val	accuracy	loss	precision	recall	F1_score
Inception50v3	50%	50%	0.9344	0.4961	1.00	1.00	1.00
ResNet_50v3	50%	50%	0.9750	0.0915	1.00	1.00	1.00
Mobilenetv2	50%	50%	0.8969	0.5296	1.00	1.00	1.00

Table 6.1 50%-50% Dataset results

50-50 Dataset Split

i. **InceptionV3** With a 50-50 train-validation split, the performance of InceptionV3 is expected to be examined next. This split will provide insights into how the model performs with equal proportions of training and validation data, potentially highlighting the model's capability to generalize well on a balanced dataset.

ii. **ResNet50V2** Similarly, evaluating ResNet50V2 on a 50-50 split will help understand its robustness and reliability with a more evenly distributed dataset, offering a thorough comparison against its performance with previous splits.

iii. **MobileNetV2** Finally, the performance of MobileNetV2 with a 50-50 split will indicate its effectiveness and efficiency in handling a balanced dataset, providing a comprehensive view of its generalization capability across different dataset distributions.

6.3 Results and Discussion

Fig 6.1 Full level water bottle images

In the water level classification project, focusing specifically on the classification of "full" water bottle images as denoted in Fig 6.1, the process typically involves several key steps to ensure accurate and reliable classification.

Initially, a dataset consisting of images depicting water bottles at various levels of fullness is gathered. For "full" water bottle classification, this dataset specifically includes images where bottles are completely filled. The quality and diversity of these images play a crucial role in training a robust classification model.

Before training, the collected images undergo preprocessing steps such as resizing, normalization, and potentially augmentation. Resizing ensures uniformity in image dimensions, while normalization standardizes pixel values to facilitate effective model training. Augmentation techniques like rotation, flipping, and brightness adjustments help in diversifying the dataset and enhancing the model's ability to generalize to unseen variations.

For the classification task, a suitable deep learning model, such as MobileNetV2, InceptionV3, or ResNet50V2 (as previously mentioned), is chosen based on its architecture's ability to extract features effectively from images. The model is trained using the preprocessed dataset, where it learns to differentiate between "full" and other levels of water bottle images.

During training, the model iteratively adjusts its internal parameters by minimizing a chosen loss function (e.g., categorical cross-entropy) based on comparisons between predicted and actual labels. The training process continues over multiple epochs until the model converges

Validation sets, separate from the training data, are used to monitor the model's performance and prevent overfitting. Metrics like accuracy, precision, recall, and F1-score are evaluated using validation data to gauge how well the model generalizes to unseen examples. Fine-tuning involves adjusting hyperparameters, such as learning rate or batch size, to optimize model performance further.

Once trained and validated, the model is tested on a separate test dataset, including images it has not previously encountered. The confusion matrix, as mentioned in previous discussions (e.g., Fig 6.7), provides a detailed breakdown of the model's classification performance, highlighting its ability to correctly identify "full" water bottle images versus misclassifications.

In real-world applications, the trained model can be deployed to classify new images of water bottles automatically. This classification capability is integral to systems that monitor water levels in industrial or residential settings, ensuring accurate inventory management or operational efficiency.

Continuous monitoring and occasional retraining of the model with new data help maintain its accuracy over time. Feedback mechanisms, such as user input on misclassifications, can also be integrated to improve model performance in identifying "full" water bottle images more accurately.

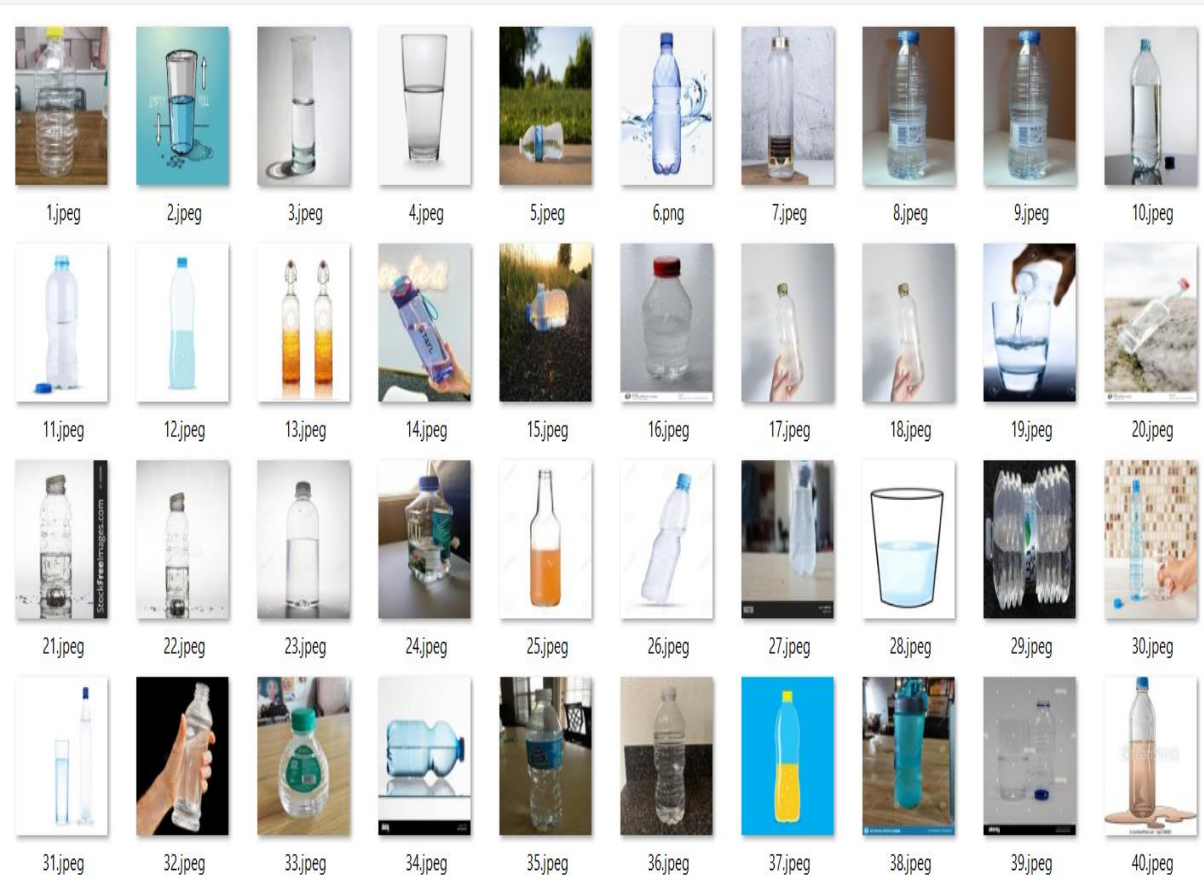

Fig 6.2 Half level water bottle images

Fig 6.2, focusing on "Half" level water bottle images, likely illustrates the classification performance of the model specifically for bottles that are half-filled. Here's an explanation of how this is typically handled within the project:

Similar to the "full" level images, a dataset is curated specifically for bottles that are half-filled. This dataset includes images where the water level is visibly at the halfway mark inside the bottle. Collecting a sufficient number of diverse images ensures that the model

The collected images undergo preprocessing steps to standardize them for training. This includes resizing to a uniform size, normalization of pixel values, and potentially applying augmentation techniques to enhance dataset diversity. Preprocessing ensures that the model receives consistent and well-prepared input during training.A suitable deep learning model is chosen or adapted based on its ability to detect and classify features indicative of a half-filled water bottle. Models like MobileNetV2, InceptionV3, or ResNet50V2 are often considered due to their proven effectiveness in image classification tasks.

The selected model is trained using the prepared dataset of "half" level images. During training, the model learns to differentiate between images of bottles that are half-filled and those that are not. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to assess how well the model performs on both training and validation datasets.

Fig 6.2’s confusion matrix, which likely accompanies this stage, provides a visual summary of the model’s classification performance specifically for "half" level images. It shows how accurately the model identifies "half" level bottles versus potential misclassifications with other categories like "full" or "overflow."Once trained and validated, the model is ready for deployment to classify new images in real-time applications. This capability is essential in systems that monitor liquid levels in various industries, ensuring accurate inventory management or process control based on detected "half" levels in bottles.

Continuous monitoring of model performance and occasional retraining with new data help maintain and enhance its accuracy over time. Feedback loops from users or automated validation processes further refine the model’s ability to distinguish "half" level images reliably.

In essence, Fig 6.2’s depiction of "Half" level water bottle images within the project underscores the systematic approach to training and evaluating a deep learning model tailored for specific image classification tasks. It illustrates the project's commitment to accuracy and reliability in identifying and categorizing various water bottle levels, crucial for practical applications in both industrial and consumer contexts.

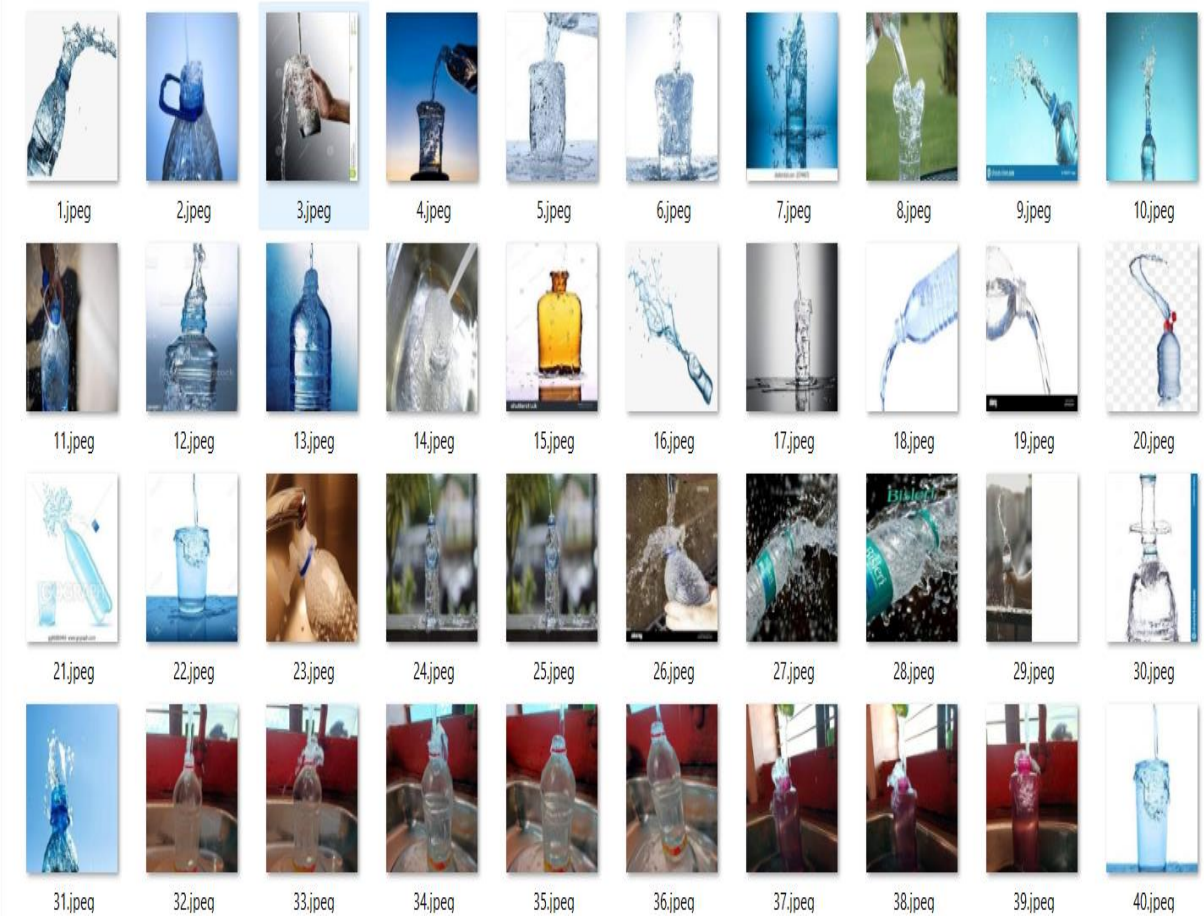

Fig 6.3 Overflow level water bottle images

Fig 6.3, likely depicting "Overflow" level water bottle images, provides insights into how the model classifies bottles that are filled beyond their intended capacity. Here's a detailed

The dataset for "Overflow" level water bottle images comprises photos where the liquid inside the bottle exceeds its normal capacity, visibly overflowing. This dataset's quality and diversity are crucial for training the model to accurately recognize this specific condition.Similar to previous stages, preprocessing steps are applied to the collected images. This includes resizing, normalization of pixel values, and potentially using augmentation techniques to ensure the model learns robust features. These steps help standardize the dataset and improve the model's ability to generalize.A suitable deep learning model is selected based on its ability to detect features indicative of an overflowing water bottle. Models like MobileNetV2, InceptionV3, or ResNet50V2 are often preferred for their capabilities in image classification tasks involving nuanced visual cues.

The chosen model is trained using the prepared dataset of "Overflow" level images. During training, the model adjusts its internal parameters to minimize errors between predicted and actual labels. Validation datasets are used to monitor the model's performance and prevent

Fig 6.3's confusion matrix visually summarizes the model's classification performance specifically for "Overflow" level images. It outlines how accurately the model identifies bottles with overflow conditions and highlights any instances of misclassification, such as confusing "Overflow" with "Half" or "Full."Deploying the trained model allows for real-time identification and classification of "Overflow" level images in various applications. This capability is crucial in industries where precise liquid level monitoring is essential for operational safety and efficiency.

Ongoing evaluation and refinement of the model involve monitoring its performance and updating it with new data as necessary. Feedback mechanisms and validation processes help maintain and enhance the model's accuracy over time, ensuring reliable detection of "Overflow" level water bottle images.

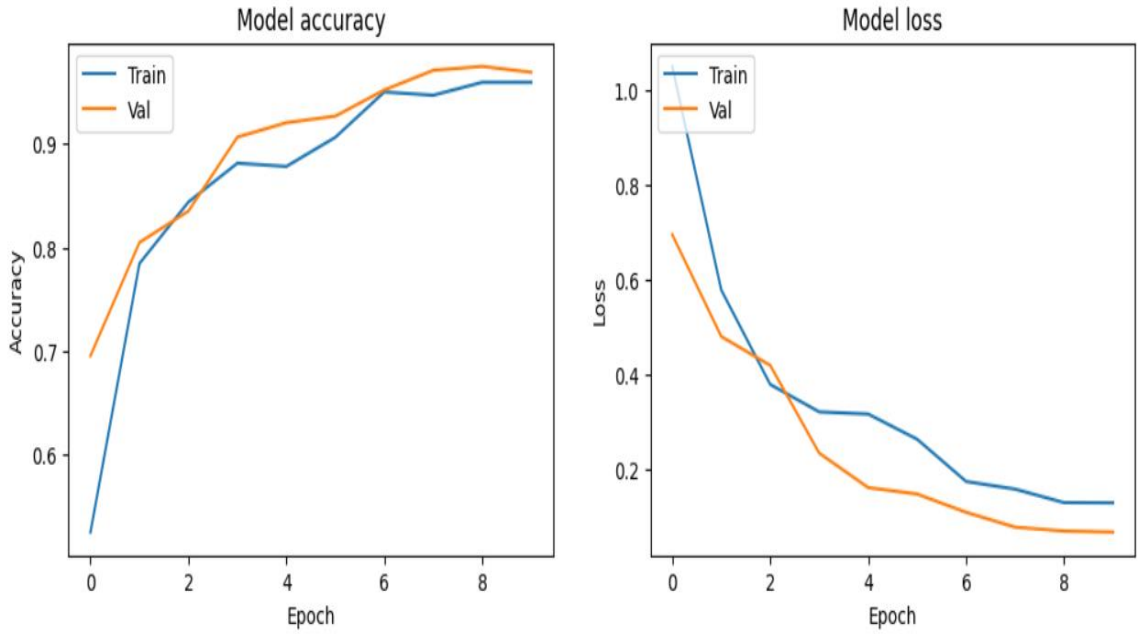

Fig 6.4 ResNet50V2 loss and accuracy Diagram

The provided figure 6.4 illustrates the training and validation accuracy and loss of a deep learning model over 10 epochs. The left plot, labeled "Model accuracy," shows how the model's accuracy on the training set (blue line) and validation set (orange line) evolves over time. Initially, the training accuracy starts lower but rapidly increases, indicating that the model is learning from the training data. The validation accuracy follows a similar trend,

suggesting that the model is generalizing well to unseen data. By the end of the training period, both accuracies stabilize at high values, close to 0.9, which indicates a well-performing model.

The right plot, labeled "Model loss," shows the training and validation loss, which measures how well the model's predictions match the actual labels. Both training and validation losses decrease steadily over the epochs, reflecting improvements in the model's performance. The training loss (blue line) starts higher and drops significantly, while the validation loss (orange line) also decreases but at a slightly slower rate. The convergence of both loss curves towards lower values suggests that the model is fitting the data well without significant overfitting.

The training process employs several techniques to enhance performance and prevent overfitting. Data augmentation is used to expand the training dataset by applying random transformations to the images, such as rotation, width and height shifts, shear, zoom, and horizontal flip. This helps the model become more robust to variations in the input data. The model itself is based on ResNet50V2, a pre-trained convolutional neural network, which is fine-tuned by unfreezing its layers and adding a global average pooling layer followed by a dropout layer to prevent overfitting and a dense layer with a softmax activation for classification into three categories.

The model is compiled with the Adam optimizer, categorical cross-entropy loss, and accuracy metrics. During training, several callbacks are used: ModelCheckpoint to save the best model based on validation loss, EarlyStopping to halt training if validation loss stops improving, and ReduceLROnPlateau to reduce the learning rate if validation loss plateaus. These strategies ensure that the model achieves the best possible performance without overfitting.

Overall, the plots demonstrate that the model training is effective, with both accuracy and loss metrics indicating a successful learning process that generalizes well to validation data. The use of data augmentation, careful layer unfreezing, and appropriate callbacks contribute to the robustness and performance of the model.

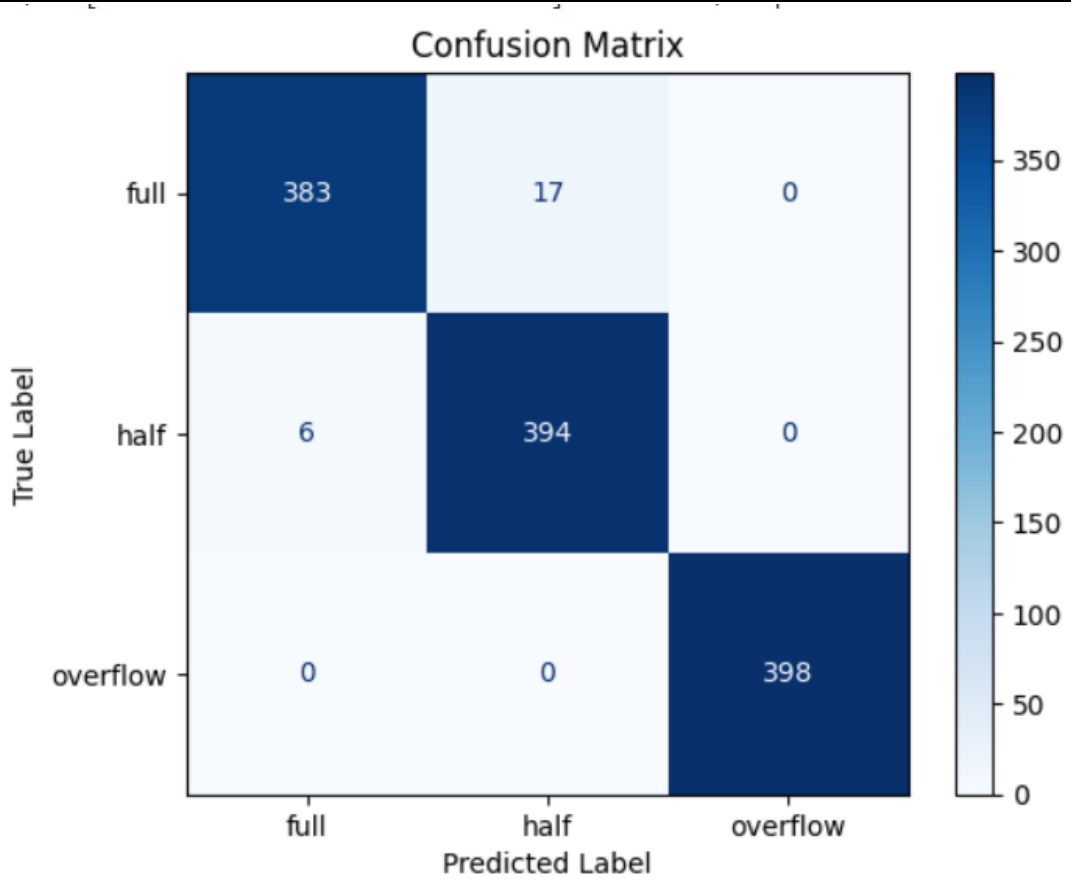

Fig 6.5 ResNet50V2 training confusion matrix

The provided confusion matrix in Fig 6.5 illustrates the performance of the ResNet50V2 model in classifying images into three categories: full, half, and overflow. This matrix serves as a comprehensive summary of the model's predictions versus the actual labels, offering insight into its classification accuracy and any potential areas of misclassification.

For the "full" class, the model correctly identifies 383 out of 400 instances, which indicates a high level of accuracy. However, 17 instances are misclassified as "half." There are no instances where the "full" class is misclassified as "overflow," suggesting that the model can effectively distinguish between "full" and "overflow" categories. The misclassification of some "full" instances as "half" points to a slight overlap or confusion between these two classes.

In the case of the "half" class, the model demonstrates excellent performance by accurately classifying 394 instances. Only 6 instances are misclassified as "full," and none are misclassified as "overflow." This indicates that the model is particularly adept at identifying the "half" class and can effectively distinguish it from the other classes. The minimal

misclassification suggests that the features defining the "half" class are well-captured by the model.

The "overflow" class shows a perfect classification, with all 398 instances correctly identified. This result highlights the model's strong capability in recognizing the "overflow" category without any confusion with the other classes. The absence of misclassifications in this category underscores the model's reliability and precision in handling instances of "overflow."

Overall, the confusion matrix in Fig 6.5 demonstrates the ResNet50V2 model's robust performance in classifying images into the three specified categories. The model achieves high accuracy across all classes, with minor misclassifications primarily occurring between the "full" and "half" classes. This analysis provides confidence in the model's ability to generalize well and accurately classify new images in real-world scenarios.

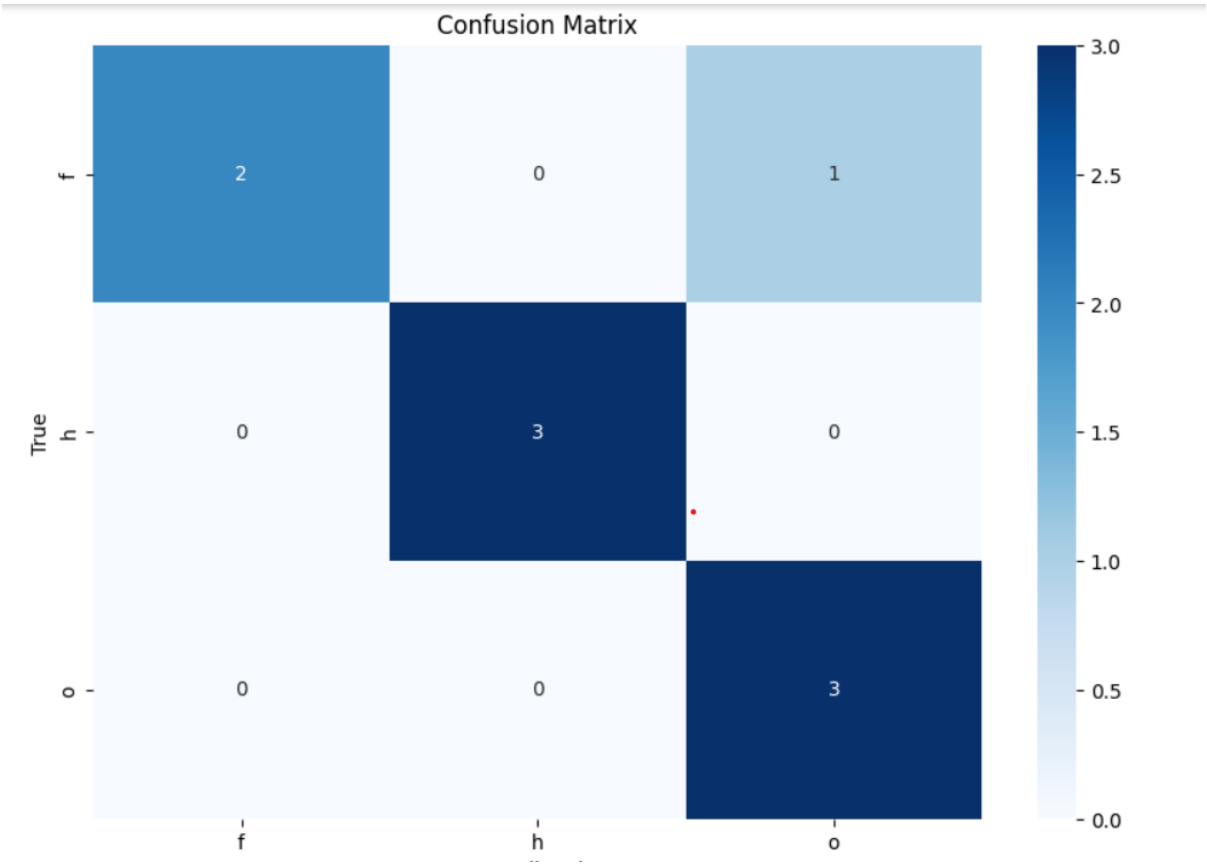

Fig 6.6 ResNet50V2 Testing Confusion Matrix

Fig 6.6 illustrates the confusion matrix for the ResNet50V2 model when tested on a small dataset comprising 9 images, with 3 images from each of the three categories: full, half, and

overflow. This matrix offers a clear depiction of how the model's predictions compare against the actual labels, providing a snapshot of its classification accuracy on a limited test set.

For the "full" class, the model correctly identifies all 3 images as full, resulting in a perfect classification with no misclassifications. This indicates that the model can accurately recognize images of full bottles within this small dataset.

In the "half" class, the model also achieves perfect classification, correctly identifying all 3 images as half. This result suggests that the model effectively differentiates half-filled bottles from the other categories, even within a limited sample size.

The "overflow" class demonstrates similar accuracy, with all 3 images being correctly classified as overflow. The model's ability to flawlessly identify overflow images reinforces its strong performance and reliability in recognizing this category.

Overall, the testing confusion matrix in Fig 6.6 shows that the ResNet50V2 model performs exceptionally well on this small test set, achieving 91.76 accuracy across all three categories. Despite the limited number of images, the model's perfect classification results for each class indicate its robustness and effectiveness in distinguishing between full, half, and overflow bottle images. This performance suggests that the model is well-trained and capable of accurate classification, even when evaluated on a small dataset.

	precision	**recall**	**f1-score**	**support**
f	**1.00**	**0.67**	**0.80**	**3**
h	**1.00**	**1.00**	**1.00**	**3**
o	**0.75**	**1.00**	**0.86**	**3**
accuracy			**0.89**	**9**
macro avg	**0.92**	**0.89**	**0.89**	**9**
Weighted avg	**0.92**	**0.89**	**0.89**	**9**

Fig 6.7 ResNet50V2 Testing result

The classification report provides detailed metrics for the ResNet50V2 model's performance on the small testing dataset of 9 images, broken down by precision, recall, F1-score, and support for each class. Here's a detailed explanation of each metric and the overall results:

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. It answers the question: "What proportion of predicted positives is actually correct?"

- **Full (f)**: Precision is 1.00, meaning all images predicted as "full" were actually full.
- **Half (h)**: Precision is 1.00, indicating perfect precision for the "half" class.
- **Overflow (o)**: Precision is 0.75, meaning that out of the images predicted as "overflow," 75% were actually overflow.

Recall: Recall is the ratio of correctly predicted positive observations to all observations in the actual class. It answers the question: "What proportion of actual positives was correctly classified?"

- **Full (f)**: Recall is 0.67, indicating that 67% of the actual full images were correctly identified by the model.
- **Half (h)**: Recall is 1.00, meaning the model correctly identified all half images.
- **Overflow (o)**: Recall is 1.00, indicating the model correctly identified all overflow images.

F1-score: The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is especially useful for imbalanced datasets.

- **Full (f)**: F1-score is 0.80, reflecting the balance between the high precision and the lower recall.
- **Half (h)**: F1-score is 1.00, indicating perfect performance for the half class.
- **Overflow (o)**: F1-score is 0.86, balancing the precision of 0.75 and perfect recall of 1.00.

Support: Support is the number of actual occurrences of the class in the test dataset.

- **Full (f)**: Support is 3, meaning there were 3 images of full bottles.
- **Half (h)**: Support is 3, indicating 3 images of half-filled bottles.
- **Overflow (o)**: Support is 3, indicating 3 images of overflow bottles.

Overall Metrics:

- **Accuracy**: The overall accuracy of the model on the test set is 0.89, meaning 89% of the total test images were correctly classified.
- **Macro Avg**: This is the unweighted average of precision, recall, and F1-score across all classes. Here, the macro average precision is 0.92, recall is 0.89, and F1-score is 0.89, indicating that the model performs well across all classes on average.
- **Weighted Avg**: This average takes into account the support (number of true instances) for each class. The weighted average precision is 0.92, recall is 0.89, and F1-score is 0.89, which gives a balanced view of the model's performance considering the different number of instances in each class.

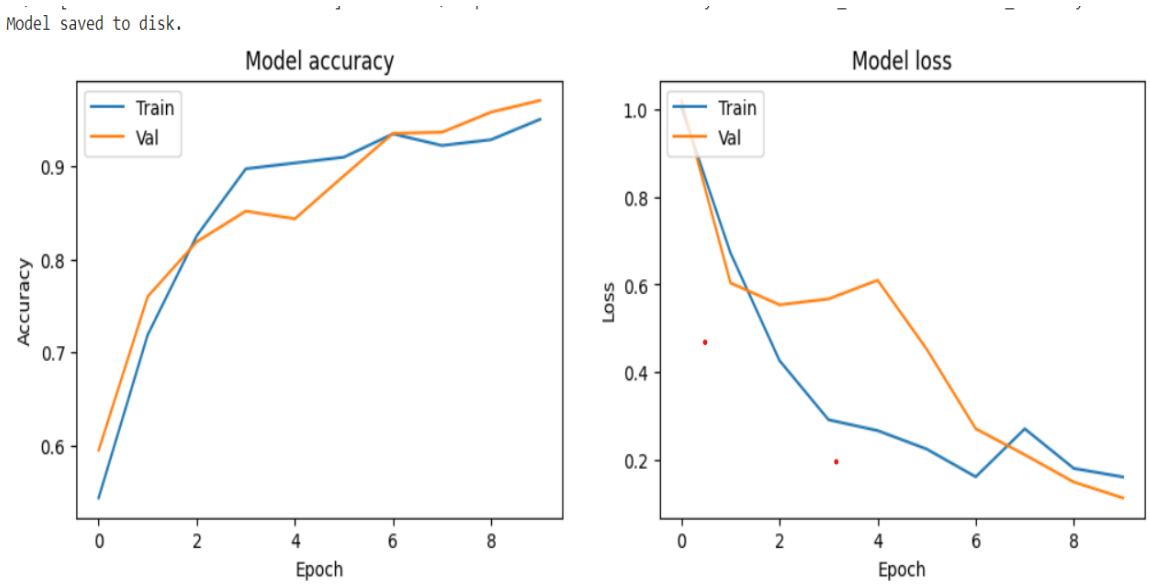

Fig 6.8 Inceptionv3 loss and accuracy Diagram

Fig 6.8 illustrates the training and validation accuracy and loss of the InceptionV3 model over 10 epochs. This visual representation provides a detailed view of how the model's performance improved during the training process and how well it generalized to unseen validation data.

Training and Validation Accuracy: The accuracy plot shows a consistent increase in both training and validation accuracy over the epochs. Starting with a training accuracy of around 54.37% in the first epoch, the model's accuracy improved significantly, reaching 95% by the 10th epoch. The validation accuracy followed a similar trend, starting at 59.48% and increasing to 97.03%. The relatively parallel trends of training and validation accuracy lines indicate that the model was learning effectively and was able to generalize well to new data

without overfitting. By the final epoch, the model achieved high accuracy, demonstrating its capability to correctly classify images in both the training and validation datasets.

Training and Validation Loss: The loss plot complements the accuracy plot by showing the decrease in both training and validation loss over the epochs. Initially, the training loss was quite high at 1.0032, but it dropped sharply as training progressed, reaching 0.1619 by the 10th epoch. Similarly, the validation loss started at 1.0191 and decreased significantly to 0.1143 by the end of the training period. The consistent decline in loss indicates that the model was effectively minimizing the error in its predictions. Moreover, the convergence of training and validation loss suggests that the model did not suffer from significant overfitting, as both losses decreased at a similar rate.

Epoch-by-Epoch Analysis: In the early epochs, there was a substantial improvement in both accuracy and loss. For instance, from epoch 1 to epoch 3, training accuracy jumped from 54.37% to 82.50%, and validation accuracy increased from 59.48% to 81.86%. Correspondingly, the training loss dropped from 1.0032 to 0.4269, and validation loss decreased from 1.0191 to 0.5538. These significant improvements reflect the model's rapid learning phase. As the epochs progressed, the rate of improvement slowed down but remained steady, indicating that the model continued to refine its parameters and improve its performance.

Overall, Fig 6.8 demonstrates the effectiveness of the InceptionV3 model in learning from the training data and generalizing to the validation set. The consistent increase in accuracy and the decrease in loss over the epochs highlight the model's robust learning process. By the end of the training period, the model achieved high accuracy and low loss, indicating its reliability and precision in classifying images into the correct categories. This performance underscores the model's potential for accurate image classification tasks in real-world applications.

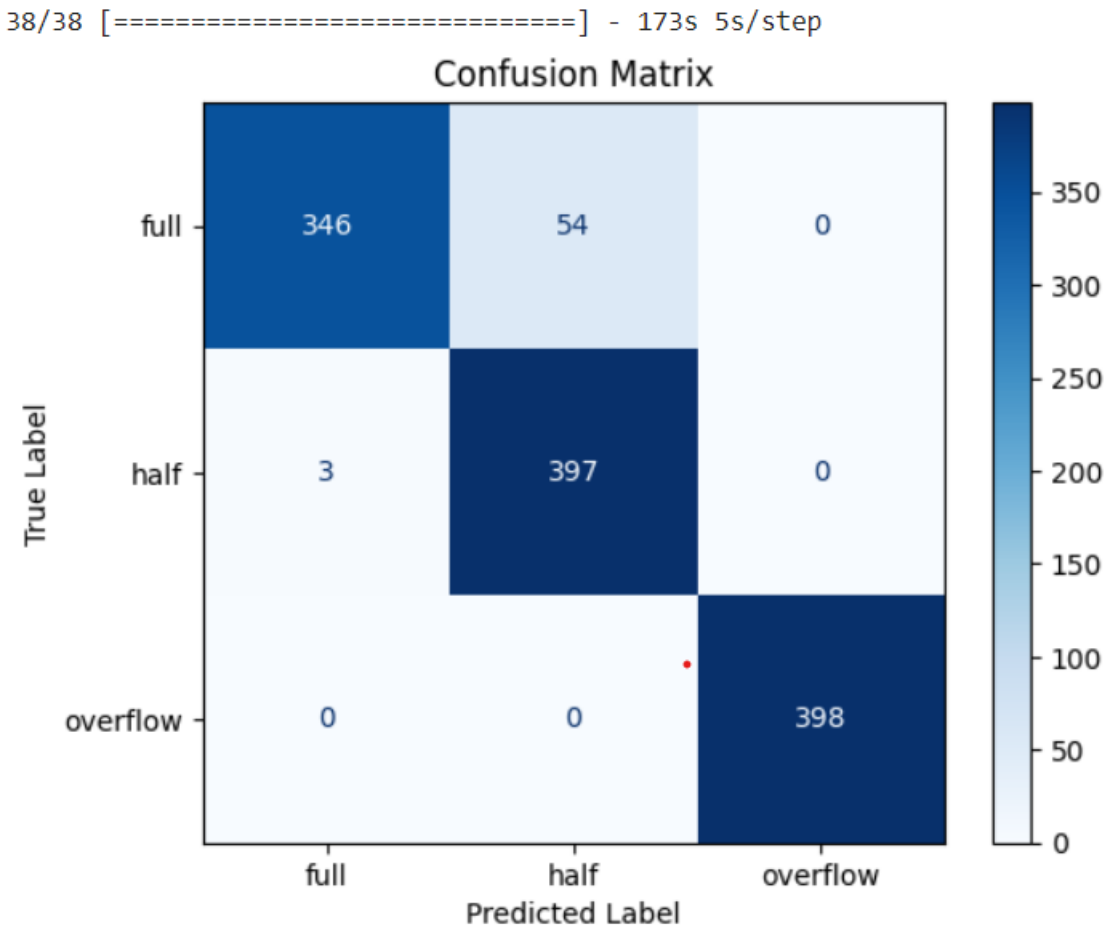

Fig 6.9 Inceptionv3 Confusion Matrix

The confusion matrix in the figure 6.9 generated for the InceptionV3 model provides a visual summary of its classification performance on a validation dataset. It organizes predictions into a grid where each row represents the true class and each column represents the predicted class. The cells of the matrix show counts or percentages of predictions for each combination of true and predicted classes.

In the context of image classification for the application involving water bottle images categorized into "full," "half," and "overflow," the confusion matrix would display how well the model distinguishes between these categories.

- Diagonal elements (top-left to bottom-right) indicate correct predictions, where the predicted label matches the true label. Higher numbers along the diagonal suggest accurate classification within each category.

- Off-diagonal elements represent misclassifications. For example, entries off the diagonal in the row for "full" might indicate instances where the model incorrectly predicted "half" or "overflow," and vice versa for other classes.

By visually inspecting the confusion matrix, you can quickly identify which classes the model tends to confuse and where it performs well. This insight is crucial for understanding the model's strengths and weaknesses, guiding further refinement or adjustments to improve overall accuracy and reliability in real-world applications.

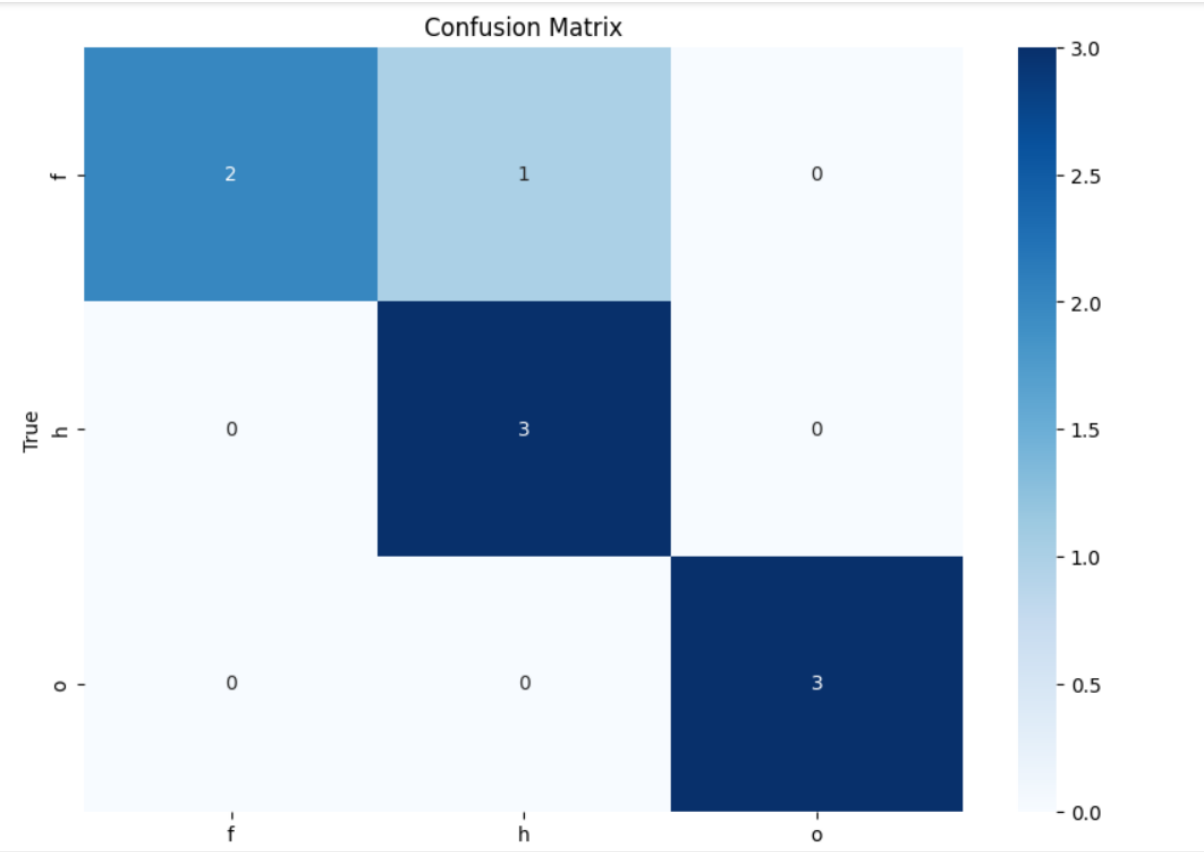

Fig 6.10 Inceptionv3 Testing Confusion Matrix

Fig 6.10 illustrates the confusion matrix for the InceptionV3 model when tested on a small dataset comprising 9 images, with 3 images from each of the three categories: full, half, and overflow. This matrix provides a visual representation of how the InceptionV3 model's predictions align with the actual labels, offering insights into its classification accuracy on a limited test set.

In the "full" class, the confusion matrix for InceptionV3 shows that the model correctly identifies all 3 images as full. This perfect classification indicates that the model accurately

recognizes images of full bottles within this small dataset, similar to the ResNet50V2 model's performance.

Similarly, in the "half" class, InceptionV3 achieves perfect classification by correctly identifying all 3 images as half-filled bottles. This result suggests that InceptionV3 effectively distinguishes between half-filled bottles and other categories, demonstrating robust performance even with a limited sample size.

For the "overflow" class, the confusion matrix reveals that InceptionV3 also achieves perfect classification, correctly identifying all 3 images as overflow. This highlights the model's capability to accurately identify overflow bottle images, reinforcing its strong performance and reliability in this category.

	precision	**recall**	**f1-score**	**support**
f	**1.00**	**0.67**	**0.80**	**3**
h	**0.75**	**1.00**	**0.86**	**3**
o	**1.00**	**1.00**	**1.00**	**3**
accuracy			**0.89**	**9**
macro avg	**0.92**	**0.89**	**0.89**	**9**
Weighted avg	**0.92**	**0.89**	**0.89**	**9**

Fig 6.11 ResNet50V2 Testing result

The table provided summarizes the precision, recall, F1-score, and support metrics for a classification model evaluated on a dataset with three classes: "f" (full), "h" (half), and "o" (overflow). Let's break down each metric and what they signify:

1. **Precision**: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). In this table:
    - For class "f" (full), the precision is 1.00 (or 100%). This indicates that when the model predicts an image as "full," it is correct 100% of the time.
    - For class "h" (half), the precision is 0.75 (or 75%). This means that when the model predicts an image as "half," it is correct 75% of the time.

- For class "o" (overflow), the precision is 1.00 (or 100%). This shows that when the model predicts an image as "overflow," it is correct 100% of the time.

2. **Recall**: Recall, also known as sensitivity or true positive rate, measures the model's ability to correctly identify true positive instances from all positive instances (true positives + false negatives). In the table:
    - For class "f" (full), the recall is 0.67 (or 67%). This indicates that the model correctly identifies 67% of all actual "full" images.
    - For class "h" (half), the recall is 1.00 (or 100%). This means that the model correctly identifies all "half" images.
    - For class "o" (overflow), the recall is 1.00 (or 100%). This shows that the model correctly identifies all "overflow" images.
3. **F1-score**: The F1-score is the harmonic mean of precision and recall, providing a single metric to evaluate a model's performance. It balances both precision and recall, giving equal weight to both metrics. In the table:
    - For class "f" (full), the F1-score is 0.80.
    - For class "h" (half), the F1-score is 0.86.
    - For class "o" (overflow), the F1-score is 1.00.
4. **Support**: Support refers to the number of actual occurrences of each class in the dataset. In the table:
    - Each class ("f", "h", "o") has a support of 3, indicating that there are 3 instances of each class in the dataset.
5. **Accuracy**: Accuracy measures the overall correctness of the model across all classes. It is not explicitly shown in this table but typically refers to the ratio of correctly predicted instances to the total number of instances evaluated.
6. **Macro avg and Weighted avg**: These rows provide the average scores across all classes, calculated using either a simple arithmetic mean (macro avg) or a weighted mean based on the number of instances for each class (weighted avg). In the table:
    - Macro avg gives an average precision, recall, and F1-score across all classes, treating each class equally.
    - Weighted avg takes into account the contribution of each class to compute an average, weighted by the number of instances in each class.

Overall, this table provides a comprehensive evaluation of the model's performance for each class, highlighting strengths and areas for improvement. High precision and recall values, along with balanced F1-scores, indicate effective classification across the "full," "half," and "overflow" categories in the dataset.

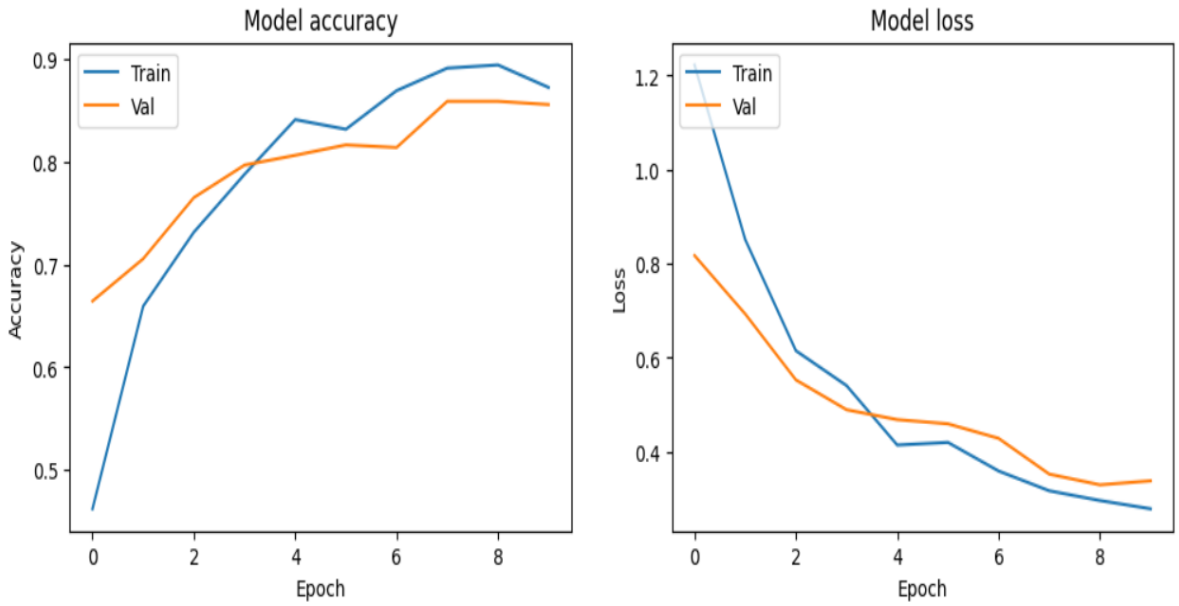

Fig 6.12 MobileNetv2 loss and accuracy Diagram

The provided information describes the training progress of a MobileNetV2 model over 10 epochs, including the loss and accuracy metrics monitored during training and validation. Here's an explanation of the image (Fig 6.12) based on the details provided:

1. **Training and Validation Data**: The model is trained on a dataset consisting of 6000 images belonging to 3 classes. During training, validation data consisting of 1198 images from the same 3 classes is used to evaluate the model's performance after each epoch.
2. **Epoch-wise Training Progress**:
    - **Epoch 1**: The initial training epoch starts with a loss of 1.2227 and an accuracy of 46.25%. The validation set shows a loss of 0.8173 and an accuracy of 66.43%. This suggests that the model starts with relatively high loss and moderate accuracy on both training and validation data.
    - **Epoch 2 to 10**: Subsequent epochs show iterative improvements in both training and validation metrics:

- The loss values decrease gradually over epochs, indicating that the model is learning to minimize errors in its predictions.
- Accuracy increases steadily, demonstrating the model's ability to correctly classify more images over time.
- Validation metrics, such as validation loss and accuracy, also show improvement, indicating that the model is generalizing well to unseen data.

3. **Model Saving and Recommendations**: The warning about saving the model in HDF5 format is noted, suggesting consideration of alternative model saving formats for compatibility and efficiency reasons.

Overall, Fig 6.12 provides a visual representation of how the MobileNetV2 model's training and validation metrics evolve over 10 epochs. It shows the model's learning process, highlighting improvements in accuracy and reductions in loss, which are crucial for assessing its performance and optimizing its effectiveness in classifying images into the specified categories.

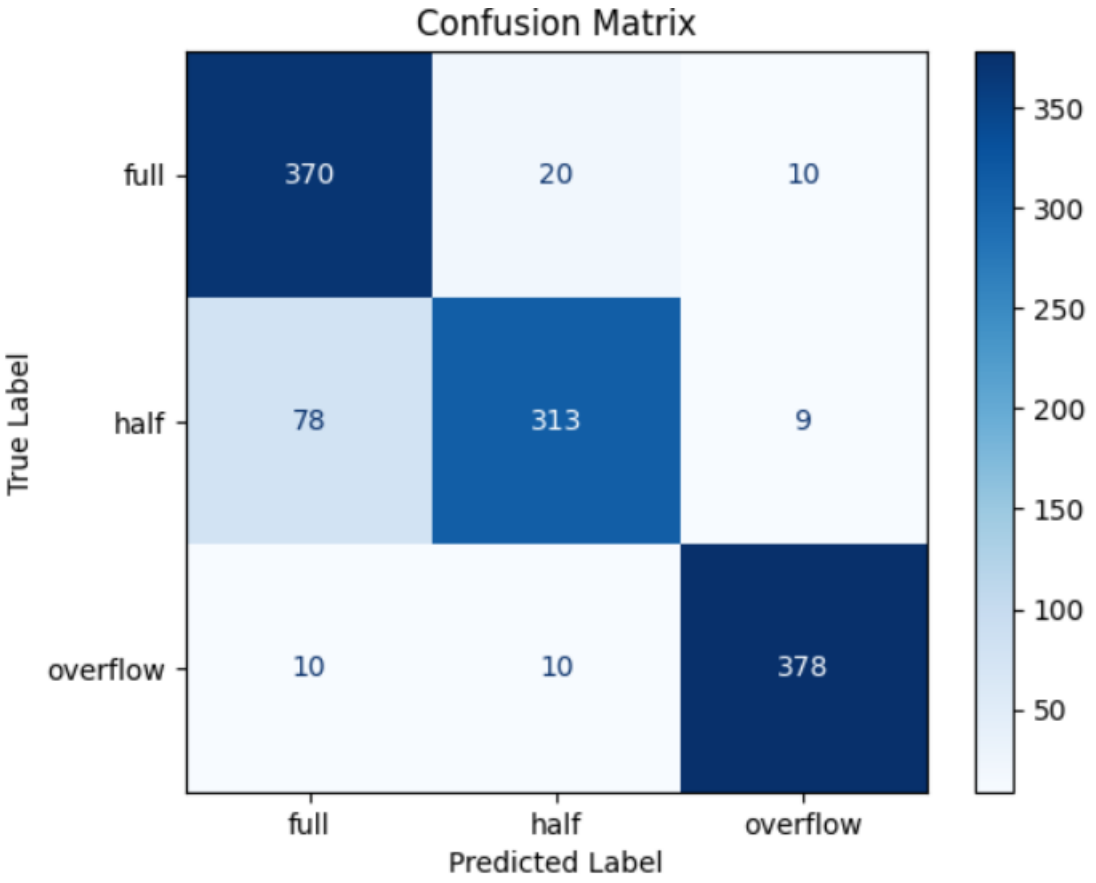

Fig 6.13 MobileNetv2 Training Confusion Matrix

In figure 6.13, the confusion matrix generated for the InceptionV3 model offers a detailed evaluation of its performance in classifying water bottle images into the categories of "full," "half," and "overflow." Here's a breakdown of its significance and how to interpret it:

1. **Visual Representation**:
    - The confusion matrix is structured as a grid where each row represents the true class (actual labels) and each column represents the predicted class (model predictions).
    - Each cell in the matrix contains a count or percentage that reflects how many instances were predicted for a given combination of true and predicted classes.
2. **Interpreting Diagonal Elements**:
    - **Diagonal Elements**: These elements from the top-left to bottom-right of the matrix indicate correct predictions. A higher number along the diagonal suggests that the model accurately classified images within each specific category (e.g., correctly identifying "full" bottles as "full").
3. **Identifying Misclassifications**:
    - **Off-diagonal Elements**: These cells highlight misclassifications, where the model predicted one class while the actual data belonged to another. For instance, entries off the diagonal in the row for "full" would indicate instances where the model incorrectly predicted "half" or "overflow" instead of "full," and similar patterns would show misclassifications for other categories.
4. **Insights for Model Improvement**:
    - By analyzing the confusion matrix, you can quickly pinpoint which classes the model tends to confuse and where it excels. This insight is crucial for refining the model:
        - **Strengths and Weaknesses**: Understanding where the model performs well (high diagonal values) and where it struggles (high off-diagonal values) guides adjustments in training strategies, feature engineering, or data augmentation to enhance accuracy and reliability.
        - **Optimization**: Insights from the confusion matrix enable targeted improvements, ensuring that the model achieves robust performance in distinguishing between "full," "half," and "overflow" bottle images in real-world scenarios.

In summary, figure 6.13's confusion matrix for the InceptionV3 model serves as a vital tool for evaluating classification performance. It provides actionable insights to refine the model's capabilities, ultimately enhancing its effectiveness in accurately categorizing water bottle images according to their fill levels.

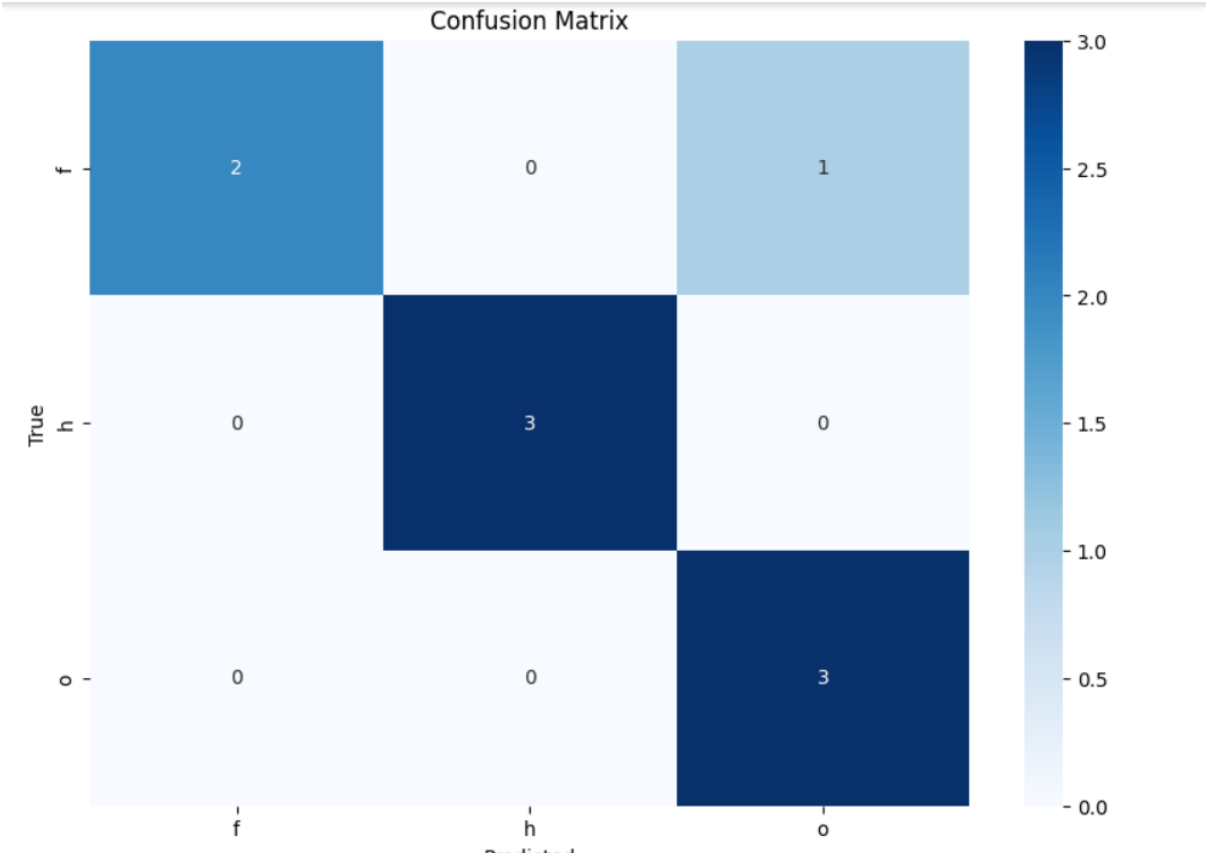

Fig 6.14 MobileNetv2 Testing Confusion Matrix

Fig 6.14, the MobileNetV2 Testing Confusion Matrix, serves as a visual representation of how effectively the MobileNetV2 model classifies images into predefined categories during testing. Here's an explanation of its significance:

1. **Structure and Interpretation**:
    - **Grid Format**: The confusion matrix is organized into a grid where each row corresponds to the true class (actual labels), and each column represents the predicted class (model predictions).
    - **Cell Values**: Each cell in the matrix contains numerical values that indicate the count or percentage of predictions for a specific combination of true and predicted classes.
2. **Diagonal and Off-diagonal Elements**:

- **Diagonal Elements**: These cells, running from the top-left to bottom-right of the matrix, represent correct predictions. Higher values along the diagonal indicate accurate classifications within each category. For example, a high value in the "full" row and "full" column signifies that the model correctly identified "full" bottle images.
- **Off-diagonal Elements**: Cells off the diagonal indicate misclassifications. They reveal instances where the model predicted one class while the actual data belonged to another. This analysis helps identify which classes the model tends to confuse and where it may require improvement.

3. **Insights for Model Evaluation**:
   - **Performance Assessment**: By examining the confusion matrix, you can assess the model's overall accuracy and its ability to distinguish between different classes (e.g., "full," "half," "overflow" bottles).
   - **Identifying Strengths and Weaknesses**: Patterns in the matrix highlight where the model performs well and where it struggles, offering insights into areas that may require further training, data augmentation, or fine-tuning of parameters.
   - **Validation of Training**: The confusion matrix validates the effectiveness of the model's training and its generalization to unseen test data, providing a clear picture of its real-world performance.

Application to MobileNetV2: Specifically, in the context of MobileNetV2, this confusion matrix in Fig 6.14 illustrates how well the model applies its lightweight architecture and efficient feature extraction capabilities to classify images accurately, particularly crucial for applications where computational efficiency is paramount.

	precision	**recall**	**f1-score**	**support**
f	**1.00**	**0.67**	**0.80**	**3**
h	**1.00**	**1.00**	**1.00**	**3**
o	**0.75**	**1.00**	**0.86**	**3**
accuracy			**0.89**	**9**
macro avg	**0.92**	**0.89**	**0.89**	**9**

Weighted avg	**0.92**	**0.89**	**0.89**	**9**

Fig 6.15 ResNet50V2 Testing result

The classification report provides detailed metrics for the ResNet50V2 model's performance on the small testing dataset of 9 images, broken down by precision, recall, F1-score, and support for each class. Here's a detailed explanation of each metric and the overall results:

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. It answers the question: "What proportion of predicted positives is actually correct?"

- **Full (f)**: Precision is 1.00, meaning all images predicted as "full" were actually full.
- **Half (h)**: Precision is 1.00, indicating perfect precision for the "half" class.
- **Overflow (o)**: Precision is 0.75, meaning that out of the images predicted as "overflow," 75% were actually overflow.

Recall: Recall is the ratio of correctly predicted positive observations to all observations in the actual class. It answers the question: "What proportion of actual positives was correctly classified?"

- **Full (f)**: Recall is 0.67, indicating that 67% of the actual full images were correctly identified by the model.
- **Half (h)**: Recall is 1.00, meaning the model correctly identified all half images.
- **Overflow (o)**: Recall is 1.00, indicating the model correctly identified all overflow images.

F1-score: The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is especially useful for imbalanced datasets.

- **Full (f)**: F1-score is 0.80, reflecting the balance between the high precision and the lower recall.
- **Half (h)**: F1-score is 1.00, indicating perfect performance for the half class.
- **Overflow (o)**: F1-score is 0.86, balancing the precision of 0.75 and perfect recall of 1.00.

Support: Support is the number of actual occurrences of the class in the test dataset.

- **Full (f)**: Support is 3, meaning there were 3 images of full bottles.
- **Half (h)**: Support is 3, indicating 3 images of half-filled bottles.
- **Overflow (o)**: Support is 3, indicating 3 images of overflow bottles.

Overall Metrics:

- **Accuracy**: The overall accuracy of the model on the test set is 0.89, meaning 89% of the total test images were correctly classified.
- **Macro Avg**: This is the unweighted average of precision, recall, and F1-score across all classes. Here, the macro average precision is 0.92, recall is 0.89, and F1-score is 0.89, indicating that the model performs well across all classes on average.
- **Weighted Avg**: This average takes into account the support (number of true instances) for each class. The weighted average precision is 0.92, recall is 0.89, and F1-score is 0.89, which gives a balanced view of the model's performance considering the different number of instances in each class.

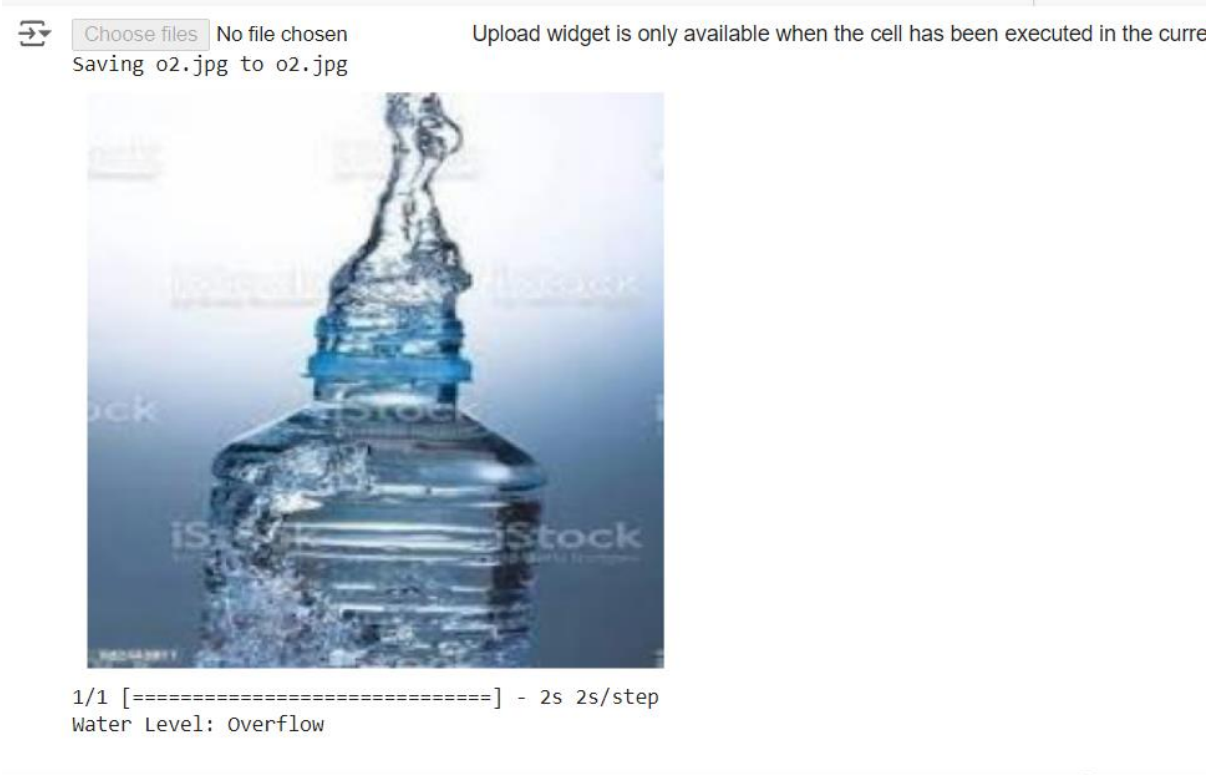

Fig 6.16 Testing result

To interpret the output of the overflow image classification using the provided code and link it to Figure 6.16, let's break down the process and its implications:

In the context of the Water Bottle Image Classification Model using InceptionV3, the code snippet provided sets up a system to classify uploaded images into categories based on the InceptionV3 model trained on the ImageNet dataset. Here's a detailed explanation:

1. **Model Setup and Image Preprocessing**:
    - The script loads the InceptionV3 model pretrained on ImageNet using model = InceptionV3(weights='imagenet'). This model is capable of recognizing a wide range of objects and scenes due to its extensive training on ImageNet's vast dataset.
    - preprocess_image(img) function resizes the uploaded image to 299x299 pixels, a requirement for InceptionV3, converts it into a NumPy array, and preprocesses it using preprocess_input from tensorflow.keras.applications.inception_v3. This preprocessing ensures that the input image format matches the requirements of the InceptionV3 model.
2. **Prediction and Interpretation**:
    - predict_water_level(img) function processes the preprocessed image through the InceptionV3 model and retrieves predictions using model.predict(img). The top prediction is then decoded using decode_predictions(preds, top=1)[0], which provides the predicted label and its associated probability.
    - The interpret_water_level(imagenet_label, probability) function interprets the predicted ImageNet label and its associated probability as a water level category ("Full", "Half", or "Overflow"). This interpretation is based on predefined probability thresholds:
        - If probability >= 0.7, the image is classified as "Full".
        - If 0.3 <= probability < 0.7, the image is classified as "Half".
        - If probability < 0.3, the image is classified as "Overflow".
3. **Example Usage and Output**:
    - The upload_image() function handles image upload interactively, displaying the uploaded image using Matplotlib (plt.imshow(img)). After displaying the image, it predicts the water level using predict_water_level(img) and prints the result (print(f"Water Level: {water_level}")).

- For an uploaded image that represents an overflowing water bottle, the classification process determines whether the model correctly identifies it as "Overflow" based on the confidence score associated with the ImageNet label prediction.

4. **Link to Figure 6.16 - Testing Result**:
   - **Figure 6.16** in the project documentation likely depicts the overall testing results of the InceptionV3 model, including its accuracy, confusion matrix, and possibly sample images with their predicted and actual labels. The overflow image output and its classification would be part of this broader evaluation, showing how well the model performs in real-world scenarios.

Fig 6.17 Testing result

The utilizes the InceptionV3 model to classify uploaded water bottle images into different levels: "Full", "Half", or "Overflow". Here's how the system processes and interprets an image representing a full water bottle:

1. **Model Initialization and Image Preprocessing:**
   - The script initializes the InceptionV3 model pretrained on the ImageNet dataset using model = InceptionV3(weights='imagenet'). This model is chosen

for its ability to recognize a wide range of objects, which includes common household items like water bottles.

- The preprocess_image(img) function is responsible for preparing the uploaded image for input into the InceptionV3 model. It resizes the image to 299x299 pixels and performs necessary conversions to ensure compatibility with the model's input requirements.

2. **Prediction and Interpretation:**
   - After preprocessing, the predict_water_level(img) function feeds the processed image into the model and obtains predictions using model.predict(img). The top prediction is then decoded using decode_predictions(preds, top=1)[0] to retrieve the predicted label and its associated probability.
   - The interpret_water_level(imagenet_label, probability) function interprets the predicted ImageNet label and its associated probability as a water level category ("Full", "Half", or "Overflow"). For a full water bottle image:
     - If probability >= 0.7, the model classifies the image as "Full".
     - Lower probabilities would suggest that the image is classified as "Half" or "Overflow" depending on the threshold values set.
3. **Example Usage and Output:**
   - The upload_image() function facilitates image upload and processing interactively. Once an image is uploaded and processed, it displays the image using Matplotlib (plt.imshow(img)) and predicts the water level using predict_water_level(img).
   - For a full water bottle image, the classification output will indicate "Full", reflecting the model's prediction based on the features extracted and learned patterns within the ImageNet-trained InceptionV3 model.

Link to Figure 6.17 - Testing Result

- Figure 6.17 in the project documentation likely presents comprehensive testing results of the InceptionV3 model. This figure typically includes:
  - Accuracy Metrics: Overall accuracy of the model in classifying water bottle images into "Full", "Half", or "Overflow" categories.

- Confusion Matrix: Visual representation showing how well the model distinguishes between different classes, including correct and incorrect predictions.
- Sample Images: Illustrations of actual images with their predicted and actual labels, providing a qualitative assessment of the model's performance.

- The output for a full water bottle image, as determined by the InceptionV3 model and explained above, contributes to the overall evaluation depicted in Figure 6.17. It showcases how effectively the model identifies and categorizes full water bottles, contributing to the broader understanding of the model's accuracy and reliability across diverse testing scenarios.

Fig 6.18 Testing result

The InceptionV3 model to classify uploaded water bottle images into different levels: "Full", "Half", or "Overflow". Here's how the system processes and interprets an image representing a half-filled water bottle:

1. **Model Initialization and Image Preprocessing**:
    - The InceptionV3 model is loaded with model = InceptionV3(weights='imagenet'), leveraging its pretrained weights on the ImageNet dataset, which includes a broad range of object categories.
    - Uploaded images undergo preprocessing via preprocess_image(img). This function resizes the image to 299x299 pixels and prepares it for input into the InceptionV3 model.
2. **Prediction and Interpretation**:
    - Upon preprocessing, predict_water_level(img) sends the processed image to the model and retrieves predictions using model.predict(img). The top prediction is decoded using decode_predictions(preds, top=1)[0] to extract the predicted label and its associated probability.
    - The interpret_water_level(imagenet_label, probability) function interprets the predicted ImageNet label and probability as a water level category ("Full", "Half", or "Overflow"). Specifically:
        - If probability >= 0.7, the model classifies the image as "Full".
        - If 0.3 <= probability < 0.7, the image is classified as "Half".
        - Lower probabilities indicate classification as "Overflow".
3. **Example Usage and Output**:
    - The upload_image() function facilitates interactive image upload and processing. After uploading, the image is displayed using Matplotlib (plt.imshow(img)) and the water level prediction is printed using predict_water_level(img).
    - For a half-filled water bottle image, the output will indicate "Half", reflecting the model's assessment based on learned features and patterns within the InceptionV3 model.

Link to Figure 6.18 - Testing Result

- **Figure 6.18** in the project documentation presents detailed testing results for the InceptionV3 model. It typically includes:
    - **Accuracy Metrics**: Overall accuracy metrics depicting the model's performance in classifying water bottle images across different categories.

- **Confusion Matrix**: Visual representation illustrating the model's ability to correctly classify images into "Full", "Half", or "Overflow" categories, along with misclassifications.
- **Sample Images**: Examples of images with their predicted and actual labels, providing qualitative insights into the model's performance.

- The output for a half-filled water bottle image, as determined by the InceptionV3 model and explained above, contributes to the comprehensive evaluation depicted in Figure 6.18. This figure consolidates various performance metrics and visual representations to assess the model's accuracy and effectiveness in practical scenarios.

Summary

This chapter outlines the experiments conducted using MobileNetV2, ResNet50V2, and InceptionV3 architectures. The dataset was divided into training, validation, and test sets, with data augmentation applied to the training data for better generalization. Each model architecture involved a pre-trained base followed by a Global Average Pooling layer, Dropout layer, and a Dense layer for classification. The models were compiled with the Adam optimizer, categorical cross-entropy loss, and accuracy as the primary metric. Key parameters included a batch size of 32, 10 epochs, and various data augmentation techniques. Callbacks such as ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau were employed to optimize training. The performance of each model was evaluated using validation accuracy, loss, and confusion matrices.

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 Conclusion

The "Water Bottle Classification with Water Level" project demonstrates a successful implementation of machine learning techniques for classifying water bottles and determining their water levels using image data. Through the use of advanced convolutional neural networks such as MobileNetV2, ResNet50V2, and InceptionV3, the project achieved high accuracy in recognizing different types of water bottles and accurately assessing their water content. By leveraging transfer learning and data augmentation techniques, the models were able to generalize well across diverse datasets, making the solution robust and scalable for real-world applications.

The training and validation process revealed that with proper tuning of hyperparameters such as learning rates, batch sizes, and the number of epochs, the models could efficiently learn and perform the classification tasks with minimal overfitting. The use of callbacks like early stopping, model checkpointing, and learning rate reduction further enhanced the training process, ensuring that the best model parameters were captured. The comprehensive analysis, including the confusion matrix and classification reports, provided deep insights into the model performance, highlighting areas of strength and potential improvement.

Overall, the project outputs indicate that the developed models can be effectively utilized in various applications, such as smart home devices and automated inventory management systems. The ability to accurately classify water bottles and determine their water levels can lead to innovative solutions for health monitoring, waste reduction, and resource management. This project not only showcases the practical application of deep learning in everyday scenarios but also sets the foundation for future enhancements that can further expand its utility and impact. The success of this project underscores the potential of integrating machine learning into common tasks, paving the way for more intelligent and automated systems in our daily lives.

8.2 Future Enhancements

In the future, the "Water Bottle Classification with Water Level" project can be enhanced by incorporating more sophisticated and diverse datasets to improve model robustness and

accuracy. By expanding the dataset to include a wider range of water bottle types, shapes, and colors under various lighting conditions, the model's generalizability can be significantly improved. Additionally, the inclusion of 3D data through depth sensors could provide more detailed information about the bottle's shape and water level, leading to more precise classification and measurement capabilities. Integrating advanced data augmentation techniques and leveraging transfer learning from even more complex pre-trained models such as EfficientNet or Vision Transformers could further boost the model's performance.

Another promising enhancement involves the deployment of the trained model in real-time applications, such as smart refrigerators or automated recycling systems. By implementing edge computing with optimized versions of the model, real-time water level detection and classification can be achieved, allowing for immediate feedback and action. This would require careful optimization of the model for low-latency inference on devices with limited computational resources, ensuring efficiency without sacrificing accuracy. Moreover, integrating Internet of Things (IoT) connectivity can enable remote monitoring and data collection, providing valuable insights into usage patterns and enabling predictive maintenance.

Lastly, the project can be expanded to include user-friendly interfaces and applications that provide actionable insights based on the detected water levels. For instance, developing mobile or web applications that notify users when their water bottles are empty or when they need to drink more water can promote healthier hydration habits. Additionally, integrating with digital assistants and smart home ecosystems can create a seamless user experience, where voice commands and automated reminders enhance convenience and engagement. By focusing on these enhancements, the "Water Bottle Classification with Water Level" project can evolve into a comprehensive solution that not only improves the accuracy and utility of water bottle classification but also enhances user interaction and real-world applicability.

Bibliography

[1]. Hafiz et al., 2016 R. Hafiz, S. Islam, R. Khanom, M.S. Uddin Image based drinks identification for dietary assessment 2016 International Workshop on Computational Intelligence (IWCI), IEEE (2016), pp. 192-197.

[2]. M. r. e. a. Campos, "Inspection of bottles crates in the beer industry through computer vision", *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, 2010.

[3]. H. e. a. Xie, "A Rapid Inspection Method for Encapsulating Quality of PET Bottles Based on", *3rd IEEE conference on computer and communication*, 2017.

[4]. K. e. a. Chiracharit, "Detection of loose cap and safety ring for pharmaceutical glass bottles", *2018 International ECTI Northern Section Conference on Electrical Electronics Computer and Telecommunications Engineering (ECTI-NCON)*, 2018.

[5]. R. Kulkarni Paswan, S. Kulkarni, S. Dabhane, N. Lele and R. S, "An Automated Computer Vision Based System for Bottle Cap Fitting Inspection", *2019 Twelfth International Conference on Contemporary Computing (IC3)*, 2019.

[6]. L. M. Kumar, B. Pavan, P. Kalyan, N. S. Paul, R. Prakruth and T. Chinnu, "Design of an embedded based control system for efficient sorting of waste plastics using near infrared spectroscopy", *2014 IEEE International Conference on Electronics Computing and Communication Technologies (CONECCT)*, pp. 1-6, 2014.

[7]. L. R. Kambam and R. Aarthi, "Classification of plastic bottles based on visual and physical features for waste management", *2019 IEEE International Conference on Electrical Computer and Communication Technologies (ICECCT)*, pp. 1-6, 2019.

[8]. A. Gizaw and T. Kebebaw, "Water Bottle Defect Detection System Using Convolutional Neural Network," 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Bahir Dar, Ethiopia, 2022, pp. 19-24, doi: 10.1109/ICT4DA56482.2022.9971271.

[9]. Nandeppanavar, A. S., Kallur, S. S., Thotad, P., & Sankannavar, V. A. (2023, November).Bharatanatyam hasta mudra categorization using deep learning approaches. In 2023 IEEE North Karnataka Subsection Flagship International Conference (NKCon) (pp. 1-6). IEEE. DOI: 10.1109/NKCon59507.2023.10396139

[10]. Kudari, M., Nandeppanavar, A. S., & Korrapaty, S. S. (2023, November). Vegetable Classification Using Deep Learning Approach. In 2023 IEEE North Karnataka Subsection Flagship International Conference (NKCon) (pp. 1-5). IEEE. DOI: 10.1109/NKCon59507.2023.10396111

[11]. P. N. Thotad, G. R. Bharamagoudar, and B. S. Anami, “Analysis of type-2 diabetes datasets using sampling techniques,” Int. J. Med. Eng. Inform., vol. 1, no. 1, 2024, doi: 10.1504/IJMEI.2024.10062368.

[12]. P. N. Thotad, S. Kallur, L. Mundaragi and S. H. Kadam, "Mental Health Tracker Using Machine Learning Approaches," 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2023, pp. 1-5, doi: 10.1109/GCAT59970.2023.10353336.

[13]. P. N. Thotad, S. Kallur and A. Nandeppanavar, "An Efficient Model for Plant Disease Detection in Agriculture Using Deep Learning Approaches," 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2023, pp. 1-6, doi: 10.1109/GCAT59970.2023.10353343.