MINOR MINI PROJECT REPORT

FRUIT RECOGNITION SYSTEM

CSE482 MINOR : MINI PROJECT

Submitted By

ANAMIKA M C (PKD21EC015)
ATHIRA P A (PKD21EC022)
HABEEBA K V (PKD21EC030)
K S SYAMKRISHNA (PKD21EC042)

under the guidance of

Dr. SABITHA S

Assistant Professor, Dept.of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GOVERNMENT ENGINEERING COLLEGE PALAKKAD

APRIL 2025

CERTIFICATE

This is to certify that the report entitled **FRUIT RECOGNITION SYSTEM** submitted by **ANAMIKA M C** (PKD21EC015), **ATHIRA P A** (PKD21EC022), **HABEEBA K V** (PKD21EC030), **K S SYAMKRISHNA** (PKD21EC042), to the Department of Computer Science and Engineering, Government Engineering College Palakkad, Sreekrishnapuram, in partial fulfilment of the requirement for the award of B-Tech Minor Degree in Computer Science and Engineering, is a bonafide record of the project carried out by them under our guidance and supervision.

Dr. Sabitha S
(Project Guide)
Assistant Professor
Dept.of CSE
Government Engineering College
Palakkad

Dr. Sabitha S
(Project Coordinator)
Assistant Professor
Dept.of CSE
Government Engineering College
Palakkad

Dr. Sabitha S
Head of Department
Dept.of CSE
Government Engineering College
Palakkad

DECLARATION

We hereby declare that the minor miniproject report "FRUIT RECOGNITION SYSTEM", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under the supervision of Dr. Sabitha S.

This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

ANAMIKA M C
ATHIRA P A
HABEEBA K V
K S SYAMKRISHNA

Palakkad
11/04/2025

ACKNOWLEDGEMENT

First of all we thank the Almighty, for giving us the strength ,courage and knowledge to do this project successfully.

We express our heartfelt gratitude to the Principal **Dr. K R Remesh Babu**, Govt Engineering College, Sreekrishnapuram, for his inspiration throughout the project.

We would also like to thank **Dr. Sabitha S**, Head Of the Department, Computer Science and Engineering, Govt. Engineering College, Palakkad for providing permission and availing all required facilities for undertaking the project in a systematic way.

We are extremely grateful to **Dr. Sabitha S**, Assistant Professor, Govt. Engineering College, Palakkad, who guided us with kind, ordinal, valuable suggestions. We would like to appreciate the guidance given by **Dr. Sabitha S**, Assistant Professor our project Coordinators, Department of Computer Science and Engineering, Govt. Engineering College, Palakkad, for providing sincere guidance.

We would also like to express our sincere gratitude to all teaching and non-teaching staff of Department of Computer Science and Engineering, Govt. Engineering College, Palakkad, for the sincere directions imparted and the cooperation in connection with the project.

We are also thankful to our parents for the support given in connection with the project. Gratitude may be extended to all well-wishers and our friends who supported us for the project.

ABSTRACT

The project utilizes a web application to identify 9 different kinds of fruits from images uploaded to the website. The primary objective of this project is to create a fast and accurate fruit detection system using convolutional neural networks (CNN). Users can upload fruit images through the web application, which then processes the images and outputs the name of the fruit. The system leverages TensorFlow's CNN models to achieve high accuracy in fruit identification. The dataset is trained and saved in hierarchical data format (.h5) to ensure efficient model deployment. The development environment for this project includes Visual Studio Code (VS Code), which provides a robust platform for coding, debugging, and version control. By incorporating additional feature extraction methods, the model's accuracy is further enhanced. This application is particularly useful aiding the fruit industry in sorting and categorizing fruits efficiently. The successful implementation of this system provides a user-friendly interface for accurate fruit identification.

Contents

Acknowledgement **i**

Abstract **ii**

1 INTRODUCTION **1**

1.1 PROBLEM STATEMENT 2

1.2 OBJECTIVE 3

1.3 MOTIVATION 3

2 LITERATURE REVIEW **5**

2.1 Fruits Classification and Detection Application Using Deep Learning[1] 5

2.2 Pure-CNN: A Framework for Fruit Images Classification[2] 6

2.3 Fruit Classification Model Based on Improved Darknet53 Convolutional Neural Network[3] 7

2.4 Automatic Fruit Classification Using Random Forest Algorithm[4] 8

3 DESIGN & METHODOLOGY **9**

3.1 Algorithm 10

3.2 Flowchart 10

4 SOFTWARE USED **13**

4.1 Kaggle 13

4.2 Google colab 13

4.3 VS code 13

5 RESULTS AND DISCUSSION **14**

6 CONCLUSION **18**

Bibliography **18**

A Working Code **20**

A.1 ML Training Code 20

A.2 Flask Code 27

A.3 HTML Code for Landing Page 28

A.4 HTML Code for Result Page 32

List of Figures

3.1 Flow Chart 11

5.1 Website interface 14

5.2 Parameters 15

5.3 Confusion matrix 15

5.4 Accuracy and Loss of Training and validation 16

5.5 Test Result 16

5.6 Webpage Result 17

Chapter 1

INTRODUCTION

In recent years, artificial intelligence and computer vision have significantly advanced, enabling machines to interpret and analyze visual data with remarkable accuracy. One of the key applications of these technologies is automated fruit recognition, which has gained attention in agriculture, food processing, and retail sectors. Accurate fruit classification can help automate food sorting, improve inventory management, and enhance user experiences in consumer applications. Traditional methods of fruit classification often rely on manual sorting, which is time-consuming and prone to errors. By leveraging deep learning and computer vision, this project aims to provide an efficient, automated solution for fruit recognition.

Convolutional neural networks (CNNs) have emerged as a powerful tool for image classification tasks, offering high accuracy in detecting and identifying objects within images. These networks are designed to automatically extract relevant features such as color, shape, and texture, making them ideal for fruit recognition. By training on diverse datasets, CNNs can effectively differentiate between various types of fruits. Additionally, integrating AI-driven recognition systems with web applications allows for seamless user interaction and easy access to results. This project uses TensorFlow's CNN models to ensure fast and accurate identification of fruits, providing a practical and scalable

solution.

With advancements in artificial intelligence, computer vision has become an indispensable tool across various industries, from autonomous vehicles to medical diagnostics. In the agriculture and food industry, automated fruit recognition has the potential to transform traditional practices by increasing efficiency, reducing labor costs, and improving product quality. This project demonstrates the practical application of CNNs in fruit classification while contributing to the broader adoption of AI-powered solutions in real-world scenarios.

1.1 PROBLEM STATEMENT

The process of identifying and classifying fruits manually can be time-consuming and prone to errors, especially in large-scale agricultural and food processing operations. Traditional methods of fruit sorting and identification require significant human intervention, which increases labor costs and can result in inconsistent quality control. There is a growing need for an automated system that can efficiently recognize and classify fruits based on images. This project aims to address this problem by developing a web-based fruit recognition system using Convolutional Neural Networks (CNNs). The system will provide an accurate, fast, and scalable solution for fruit classification, reducing human labor, enhancing productivity, and ensuring consistent quality in fruit sorting and packaging.

1.2 OBJECTIVE

- To design a user-friendly webpage where users can upload images of fruits.
- To preprocess the images, ensuring consistency and improving model performance.
- To implement a CNN model trained on the 9 fruit dataset to classify and recognize fruit types accurately.
- To integrate the CNN model with the web application for real-time fruit recognition upon image upload.
- To evaluate and optimize the model's performance (accuracy, precision, recall) using the 9 fruit dataset.
- To display the name of the recognized fruit on the webpage after image classification.
- To display the confidence score of the prediction, providing users with a measure of certainty in classification.

1.3 MOTIVATION

The need for efficient and accurate fruit classification in agriculture, food processing, and quality control has driven the motivation for this project. Traditional methods of fruit identification and sorting are labor-intensive, time-consuming, and prone to human errors, leading to inefficiencies in production and distribution. With advancements in artificial intelligence and deep learning, particularly Convolutional Neural Networks (CNNs), automated image-based fruit recognition offers

a promising solution. This project aims to leverage CNNs to develop an accurate and efficient fruit recognition system that simplifies classification and enhances productivity. By integrating this technology into a web-based application, we can make fruit identification more accessible and scalable, benefiting industries such as agriculture, food retail, and education. Furthermore, the potential to expand the system for broader applications, such as smart farming and food waste reduction, serves as a strong motivation to explore and implement this innovative solution.

Chapter 2

LITERATURE REVIEW

2.1 Fruits Classification and Detection Application Using Deep Learning[1]

The research article titled "Fruits Classification and Detection Application Using Deep Learning" presents an automated system for fruit classification and detection using deep learning techniques. The study utilizes two datasets: the publicly available FIDS-30 dataset, which contains 971 images across 30 fruit classes, and a custom dataset with 761 images from 8 fruit classes, collected and annotated by the authors. The dataset was preprocessed and augmented using techniques such as projection, rotation, scaling, and brightness/contrast adjustments. The system employs YOLOv3 and YOLOv7 models for detecting multiple fruits within images, while ResNet50 and VGG16 neural networks were used for single-fruit classification. A domain adaptation approach was applied to evaluate the system's ability to generalize across different datasets. The classification models demonstrated high accuracy, with ResNet50 achieving 99% accuracy on the custom dataset and 85% on FIDS-30, while VGG16 achieved 98% and 86%, respectively. The YOLOv7 model, used for fruit detection, achieved 96.1% accuracy, with a precision of 0.93 and recall of 0.89. The domain adaptation tests showed that when trained on the custom dataset and

tested on FIDS-30, ResNet50 and VGG16 achieved 85% and 86% accuracy, respectively, whereas when trained on FIDS-30 and tested on the custom dataset, they achieved 98% and 99% accuracy.The system was deployed as both a Flask-based web application and an Android mobile application called "Fruit Holmes." The mobile application allows real-time fruit detection using a smartphone camera and provides text-based and voice descriptions of the identified fruits. The study highlights the potential applications of this technology in industries, supermarkets, and educational settings. Future improvements include enhancing deep learning models for greater accuracy, expanding the dataset to include more fruit species, and integrating a defect detection system for smart farming applications.

2.2 Pure-CNN: A Framework for Fruit Images Classification[2]

The project focuses on developing a Pure Convolutional Neural Network (PCNN) specifically designed for fruit image classification, addressing the challenges posed by the diverse varieties of fruits and their similarities in color, shape, size, and texture. The PCNN architecture consists of seven convolutional layers, incorporating Global Average Pooling (GAP) to effectively reduce overfitting and enhance the model's classification performance. The study utilizes the fruit-360 dataset, which comprises 55,244 color images categorized into 81 different types of fruits. The results demonstrate that the PCNN achieves an impressive classification accuracy of 98.88%, surpassing traditional CNN models that utilize Fully Connected (FC) layers and Dropout techniques. This advancement is particularly significant in the context of agricultural

automation, where the labor-intensive nature of fruit sorting and classification can be streamlined through machine learning and computer vision techniques. The architecture employs ReLU activation functions, stride for dimension reduction, and GAP for minimizing the number of parameters, ultimately leading to improved efficiency and accuracy in fruit classification tasks. The findings highlight the potential of PCNNs in automating fruit recognition, thereby contributing to enhanced productivity in the agricultural sector.

2.3 Fruit Classification Model Based on Improved Darknet53 Convolutional Neural Network[3]

The project presents an advanced fruit classification model that leverages a dataset of 28,600 images, encompassing 22 distinct types of fruits, including apples, bananas, and watermelons. By integrating group normalization in place of batch normalization within the DarkNet53 architecture and employing a 22-label SoftMax classifier, the model achieves a remarkable top-1 accuracy of 95.6%, significantly surpassing traditional models such as VGG-16 and Inception V3, which achieved accuracies of 90.6% and 91.4%, respectively. Additionally, the implementation of data augmentation techniques, such as flipping, rotating, and brightness enhancement, further enhances the model's generalization ability, effectively reducing overfitting and improving performance in diverse application scenarios. The results indicate that the improved model not only excels in classification accuracy but also demonstrates reduced memory consumption and increased efficiency, making it well-suited for real-time fruit recognition tasks in complex environments. Overall,

this project contributes to the field of computer vision by providing a robust solution for fruit detection and classification, with potential applications in agriculture and food industry automation.

2.4 Automatic Fruit Classification Using Random Forest Algorithm[4]

The project aims to create an automatic fruit classification system that accurately recognizes and classifies fruit images by name. It consists of three main stages: preprocessing, feature extraction, and classification. In the preprocessing stage, images are resized to 90 x 90 pixels in RGB format. The feature extraction stage employs two methods: one that captures shape and color features using color moments (variance, mean, kurtosis, and skewness) and another that utilizes Scale Invariant Feature Transform (SIFT) for robust keypoint detection and description.

For classification, the Random Forest (RF) algorithm is implemented, and its performance is compared with K-Nearest Neighbor (K-NN) and Support Vector Machine (SVM) algorithms. The system was evaluated using a dataset of 178 fruit images, which included oranges, strawberries, and apples. Results showed that the RF classifier achieved the highest accuracy, with 100% accuracy for both orange and strawberry images when using SIFT features with K-NN. The RF classifier also achieved 96.97% accuracy for apples and 87.50% for oranges when using shape and color features. Overall, the RF algorithm outperformed K-NN and SVM, demonstrating its effectiveness in fruit classification tasks.

Chapter 3

DESIGN & METHODOLOGY

This project employs Convolutional Neural Networks (CNNs), a class of deep learning models known for their exceptional performance in image classification tasks. Specifically, MobileNetV2 is used as the backbone model due to its efficiency and ability to operate in resource-constrained environments. The dataset used in this project contains images of nine different fruit types, which are preprocessed and divided into training (80%), validation (10%), and testing (10%) subsets to ensure a balanced learning process. To improve the model's ability to generalize across different lighting conditions, angles, and backgrounds, data augmentation techniques such as rotation, width and height shifts, shear transformation, zoom, and horizontal flipping are applied.

The MobileNetV2 model is utilized as a feature extractor, with the first 100 layers frozen to retain pre-trained knowledge. A custom classification head is added on top, consisting of fully connected layers and dropout layers to enhance model performance and prevent overfitting. The model is trained using the Adam optimizer and categorical cross-entropy loss function, with training monitored using early stopping and learning rate reduction techniques to ensure optimal performance.

Once training is complete, the model's performance is evaluated using a confusion matrix and a classification report, which provide insights into its accuracy and ability to differentiate between similar fruits. The

final trained model is integrated into a web-based application, where users can upload an image of a fruit, and the system processes the image to classify it and return the corresponding fruit name.

3.1 Algorithm

1. Start
2. Accept an image from the user through the web interface.
3. Preprocess the image by resizing and normalizing it.
4. Extract important features from the image.
5. Send the preprocessed image to the backend server
6. Load the trained classification model.
7. Predict the fruit category and calculate the confidence score.
8. Display the predicted fruit name along with the confidence score.
9. Stop

3.2 Flowchart

The system leverages deep learning techniques to accurately classify fruit images based on key visual features. The process begins with the user uploading an image through a web-based interface, which serves as the input to the system. Once the image is received, it undergoes preprocessing, including resizing and normalization, to standardize its dimensions and improve processing efficiency. Following this, key features such as color, texture, and shape are extracted to aid in classification. The preprocessed image is then sent to the backend server, where

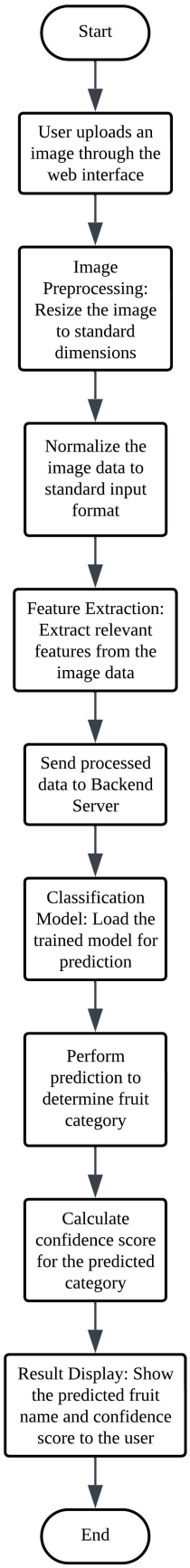

Figure 3.1: Flow Chart

a trained Convolutional Neural Network (CNN) model is loaded. The model processes the image, predicts the fruit category, and generates a confidence score to indicate the reliability of the classification. The predicted fruit name, along with the confidence score, is then displayed on the web interface, providing real-time feedback to the user. Finally, the process concludes after successfully displaying the result, ensuring an efficient and automated approach to fruit classification.

Chapter 4

SOFTWARE USED

4.1 Kaggle

Kaggle is used for dataset exploration and initial data preprocessing. The platform provides a large collection of datasets, including fruit image datasets, which were used for training the CNN model. Kaggle also offers built-in Jupyter Notebook support, which was helpful in testing initial models.

4.2 Google colab

Google Colab is used for model training and evaluation. Since Colab provides free GPU access, it significantly reduced training time for the CNN model. It also allowed seamless integration with Google Drive for storing datasets and model checkpoints.

4.3 VS code

Visual Studio Code (VS Code) is used for writing and debugging Python scripts. It provided better code organization, version control (via Git), and easy integration with libraries such as TensorFlow and OpenCV. The final implementation of the fruit recognition system, including the web interface, was developed and tested using VS Code.

Chapter 5

RESULTS AND DISCUSSION

The webpage serves as the user interface for the fruit recognition system, enabling users to upload fruit images for identification. Designed with a visually appealing background of various fruits, it features interactive elements like a file upload option and a prediction button to simplify the recognition process.

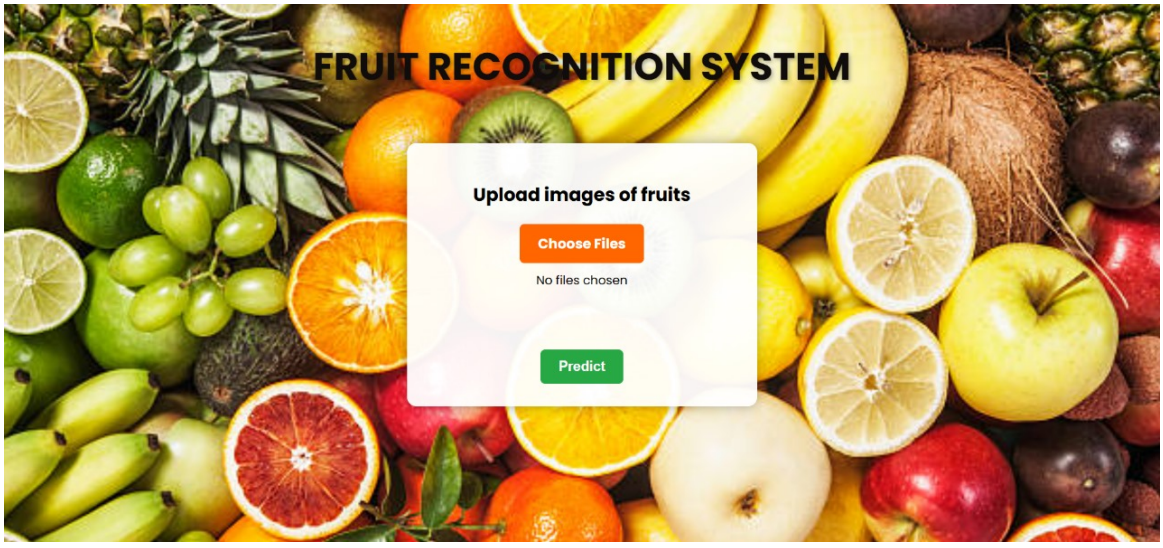

Figure 5.1: Website interface

The fruit recognition system achieves exceptional performance with 98% accuracy, showcasing strong precision, recall, and F1-scores in classifying fruits like apples, bananas, mangoes, watermelons etc. Its reliability and balanced metrics make it a promising tool for applications in automated sorting, inventory management, and smart kitchen assistants.

	precision	recall	f1-score	support
Apple	0.97	0.94	0.96	35
banana	0.99	1.00	1.00	102
guava	1.00	1.00	1.00	103
mango	1.00	0.99	1.00	104
orange	0.97	1.00	0.99	104
papaya	1.00	0.99	0.99	100
pineapple	1.00	1.00	1.00	100
waterapple	0.90	0.99	0.94	91
watermelon	1.00	0.90	0.95	100
accuracy			0.98	839
macro avg	0.98	0.98	0.98	839
weighted avg	0.98	0.98	0.98	839

Figure 5.2: Parameters

The confusion matrix highlights the fruit recognition system's strong performance, with accurate classifications for most fruit categories, reflecting its reliability and precision. The minimal instances of misclassification indicate areas for fine-tuning while demonstrating overall effectiveness.

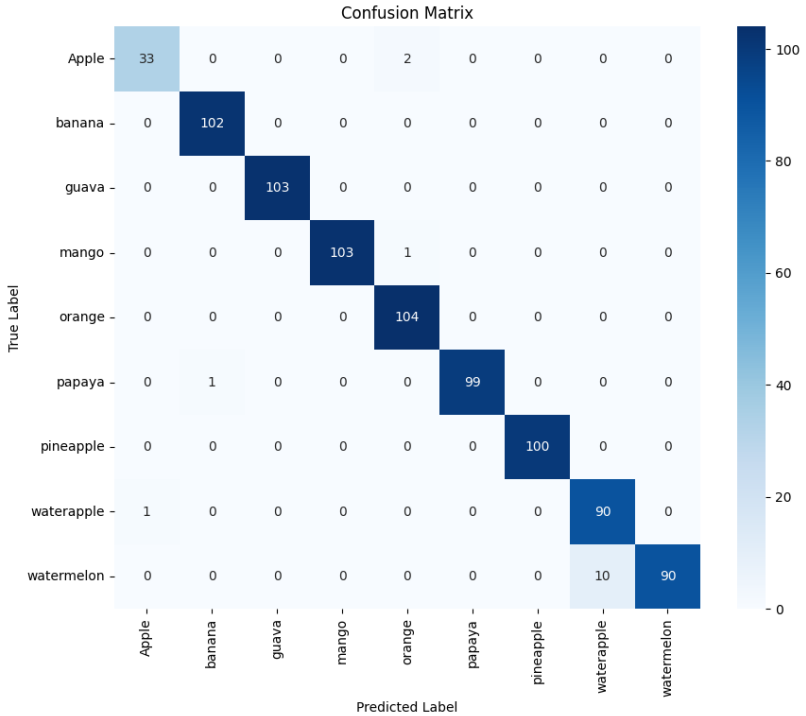

Figure 5.3: Confusion matrix

The graphs illustrate the performance of the fruit recognition system across 20 training epochs, showcasing its progressive improvement. The accuracy graph highlights a consistent rise in both training and validation accuracy, approaching 1.0, indicating effective learning. The loss graph demonstrates a steady reduction in training loss and fluctuating yet declining validation loss, signifying improved model generalization and minimized errors. These results underline the system's reliability and robust performance in fruit classification tasks.

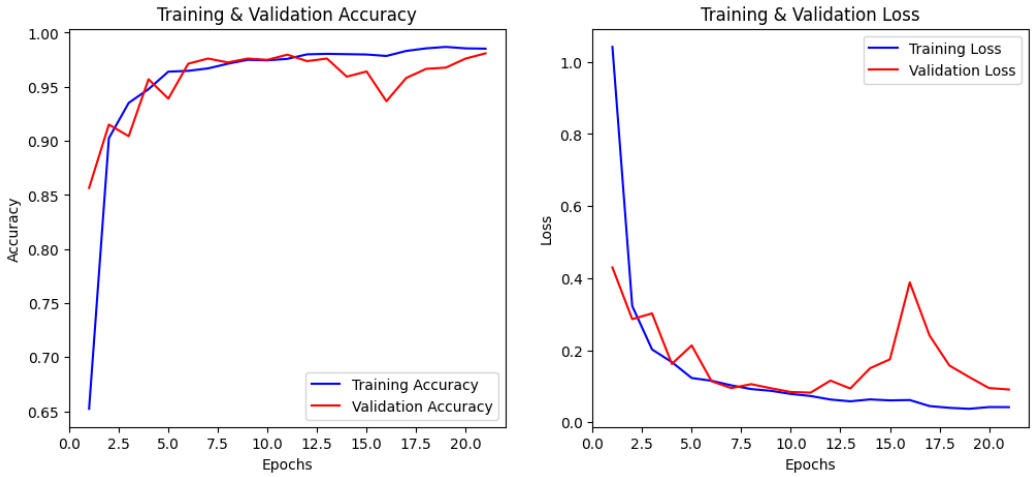

Figure 5.4: Accuracy and Loss of Training and validation

The model was successfully tested in Google Colab using a dataset of fruit images. The predictions made by the model aligned with the true labels, as demonstrated in the test images. All images were accurately classified, showcasing the model's efficiency and precision in fruit type identification.

Figure 5.5: Test Result

The web page displayed the output when multiple images were uploaded. Each image was accurately identified, with the corresponding labels and confidence levels displayed alongside, showcasing effective and reliable image categorization.

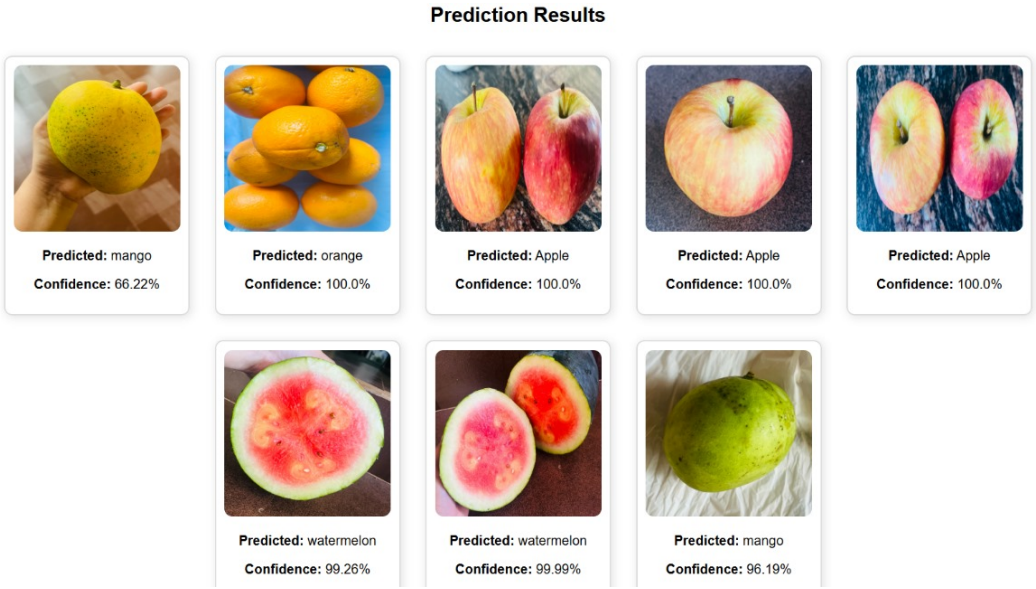

Figure 5.6: Webpage Result

Chapter 6

CONCLUSION

The fruit recognition system using Convolutional Neural Networks (CNN) achieved an impressive accuracy of 98%, demonstrating its reliability in identifying 9 different fruit categories. The project involved dataset acquisition from Kaggle, preprocessing (resizing, normalization, and data augmentation), and model training using Google Colab with GPU support. The final implementation in VS Code allowed users to upload an image via a web interface, where the system not only predicted the correct fruit name but also displayed the confidence level of each prediction.

By leveraging deep learning, the model effectively extracted distinguishing features from fruit images, ensuring precise classification. The ability to display confidence scores adds transparency to the predictions, helping users assess the model's reliability in uncertain cases. This approach is well-suited for applications in smart agriculture, automated grocery sorting, and food quality control.

Future improvements could include expanding the dataset to cover more fruit varieties, optimizing the model for real-time classification, and integrating it with IoT-based systems for enhanced efficiency and automation.

BIBLIOGRAPHY

[1] Nur-E-Aznin Mimma, Sumon Ahmed, Tahsin Rahman, and Riasat Khan. Fruits classification and detection application using deep learning. *Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh*, November 2022.

[2] Asia Kausar, Mohsin Sharif, JinHyuck Park, and Dong Ryeol Shin. Pure-cnn: A framework for fruit images classification. *Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, South Korea*, 2022.

[3] Hui Wang, Fan Zhang, and Li Wang. Fruit classification model based on improved darknet53 convolutional neural network. *College of Information Engineering, Minzu University of China, Beijing, China*, 2022.

[4] Hossam M. Zawbaa, Maryam Hazman, Mona Abbass, and Aboul Ella Hassanien. Automatic fruit classification using random forest algorithm. 2014.

Appendix A

Working Code

A.1 ML Training Code

```
from google.colab import drive
drive.mount('/content/drive')

import tensorflow as tf
from tensorflow.keras.preprocessing.image import
    ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout,
    GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau,
    EarlyStopping
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix,
    classification_report
import numpy as np
import os
import shutil
import random

# Define dataset paths
original_dataset_dir = '/content/drive/MyDrive/full'
base_dir = '/content/fruit_data_split'
```

```
22
23 # Remove previous dataset splits (if any) to ensure random
       selection each time
24 if os.path.exists(base_dir):
25     shutil.rmtree(base_dir)
26 os.makedirs(base_dir)
27
28 # Create subdirectories for train, validation, and test
29 train_dir = os.path.join(base_dir, 'train')
30 val_dir = os.path.join(base_dir, 'val')
31 test_dir = os.path.join(base_dir, 'test')
32 for directory in [train_dir, val_dir, test_dir]:
33     os.makedirs(directory)
34
35 # Randomly split dataset each run
36 for fruit_class in os.listdir(original_dataset_dir):
37     class_path = os.path.join(original_dataset_dir, fruit_class)
38     if not os.path.isdir(class_path):
39         continue
40     images = os.listdir(class_path)
41     random.shuffle(images)  # Shuffle images randomly
42     train_split = int(0.8 * len(images))
43     val_split = int(0.9 * len(images))  # 10% validation,
       remaining 10% test
44     train_images = images[:train_split]
45     val_images = images[train_split:val_split]
46     test_images = images[val_split:]
47     for split, image_list in zip([train_dir, val_dir, test_dir],
       [train_images, val_images, test_images]):
48         class_split_path = os.path.join(split, fruit_class)
49         os.makedirs(class_split_path, exist_ok=True)
50         for image in image_list:
51             src = os.path.join(class_path, image)
52             dst = os.path.join(class_split_path, image)
53             shutil.copyfile(src, dst)
```

```
54 print("Dataset successfully split into train, val, and test sets
       !")
55
56 IMG_SIZE = 224
57 BATCH_SIZE = 32
58
59 # Data augmentation for training set
60 train_datagen = ImageDataGenerator(
61     rescale=1./255,
62     rotation_range=30,
63     width_shift_range=0.2,
64     height_shift_range=0.2,
65     shear_range=0.2,
66     zoom_range=0.2,
67     horizontal_flip=True,
68     fill_mode='nearest'
69 )
70
71 # Only rescale for validation & test sets
72 val_test_datagen = ImageDataGenerator(rescale=1./255)
73
74 # Load dataset
75 train_data = train_datagen.flow_from_directory(train_dir,
       target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,
       class_mode='categorical')
76 val_data = val_test_datagen.flow_from_directory(val_dir,
       target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,
       class_mode='categorical')
77 test_data = val_test_datagen.flow_from_directory(test_dir,
       target_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,
       class_mode='categorical', shuffle=False)
78 num_classes = len(train_data.class_indices)
79 print(f"Number of fruit classes: {num_classes}")
80
81 # Load MobileNetV2 without top layers
```

```
82 base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3),
       include_top=False, weights='imagenet')
83
84 # Freeze the first 100 layers for better feature extraction
85 for layer in base_model.layers[:100]:
86     layer.trainable = False
87
88 # Custom classification head
89 x = GlobalAveragePooling2D()(base_model.output)
90 x = Dense(512, activation='relu')(x)
91 x = Dropout(0.5)(x)
92 x = Dense(256, activation='relu')(x)
93 x = Dropout(0.3)(x)
94 output = Dense(num_classes, activation='softmax')(x)
95
96 # Define final model
97 model = Model(inputs=base_model.input, outputs=output)
98
99 # Compile the model
100 model.compile(optimizer=Adam(learning_rate=0.0001), loss='
       categorical_crossentropy', metrics=['accuracy'])
101
102 # Callbacks to improve performance
103 reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
       patience=5, min_lr=1e-6, verbose=1)
104 early_stop = EarlyStopping(monitor='val_loss', patience=10,
       restore_best_weights=True, verbose=1)
105
106 # Train model
107 history = model.fit(train_data, validation_data=val_data, epochs
       =50, callbacks=[reduce_lr, early_stop])
108 model.save('/content/drive/MyDrive/fruit_classifier_model.h5')
109 print("Model saved successfully!")
110
111 # Evaluate on test set
```

```
112 y_true = test_data.classes
113 y_pred = np.argmax(model.predict(test_data), axis=1)
114
115 # Confusion matrix
116 cm = confusion_matrix(y_true, y_pred)
117 # Plot confusion matrix
118 plt.figure(figsize=(10, 8))
119 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=
    train_data.class_indices.keys(), yticklabels=train_data.
    class_indices.keys())
120 plt.xlabel("Predicted Label")
121 plt.ylabel("True Label")
122 plt.title("Confusion Matrix")
123 plt.show()
124
125 # Classification report
126 print(classification_report(y_true, y_pred, target_names=
    train_data.class_indices.keys()))
127 import matplotlib.pyplot as plt
128
129 # Assuming 'history' is the output from model.fit()
130 def plot_accuracy_loss(history):
131     acc = history.history['accuracy']
132     val_acc = history.history['val_accuracy']
133     loss = history.history['loss']
134     val_loss = history.history['val_loss']
135     epochs = range(1, len(acc) + 1)
136
137     # Accuracy Plot
138     plt.figure(figsize=(12, 5))
139     plt.subplot(1, 2, 1)
140     plt.plot(epochs, acc, 'b-', label='Training Accuracy')
141     plt.plot(epochs, val_acc, 'r-', label='Validation Accuracy')
142     plt.xlabel('Epochs')
143     plt.ylabel('Accuracy')
```

```
144     plt.legend()
145     plt.title('Training & Validation Accuracy')
146 
147     # Loss Plot
148     plt.subplot(1, 2, 2)
149     plt.plot(epochs, loss, 'b-', label='Training Loss')
150     plt.plot(epochs, val_loss, 'r-', label='Validation Loss')
151     plt.xlabel('Epochs')
152     plt.ylabel('Loss')
153     plt.legend()
154     plt.title('Training & Validation Loss')
155     plt.show()
156 
157 # Call this function after training
158 plot_accuracy_loss(history)
159 from tensorflow.keras.preprocessing.image import
        ImageDataGenerator
160 
161 # Define test dataset directory
162 test_dir = "/content/fruit_data_split/test"
163 
164 # Create ImageDataGenerator for testing (No Augmentation)
165 test_datagen = ImageDataGenerator(rescale=1./255)
166 test_generator = test_datagen.flow_from_directory(
167     test_dir,
168     target_size=(224, 224),  # Match model input size
169     batch_size=32,
170     class_mode='categorical',
171     shuffle=True  # Shuffle images for variety
172 )
173 
174 # Get class labels
175 class_labels = list(test_generator.class_indices.keys())
176 import numpy as np
177 import matplotlib.pyplot as plt
```

```
178
179 def plot_sample_predictions(model, test_generator, class_labels,
    num_images=5):
180     # Get a batch of test images and labels
181     test_images, test_labels = next(test_generator)  # Get batch
182     plt.figure(figsize=(15, 10))
183     for i in range(num_images):
184         img = test_images[i]
185         true_label = class_labels[np.argmax(test_labels[i])]
186
187         # Predict
188         img_array = np.expand_dims(img, axis=0)
189         predictions = model.predict(img_array)
190         predicted_label = class_labels[np.argmax(predictions)]
191
192         # Plot Image with Predictions
193         plt.subplot(1, num_images, i + 1)
194         plt.imshow(img)
195         plt.axis('off')
196         plt.title(f'True: {true_label}\nPred: {predicted_label}'
    , color='green' if true_label == predicted_label else 'red')
197         plt.show()
198
199 # Call this function after defining test_generator
200 plot_sample_predictions(model, test_generator, class_labels)
201 import json
202
203 # Save class labels to a file
204 class_labels = list(test_generator.class_indices.keys())
205 with open('/content/drive/MyDrive/class_labels.json', 'w') as f:
206     json.dump(class_labels, f)
```

A.2 Flask Code

```
from flask import Flask, render_template, request, url_for
import tensorflow as tf
import numpy as np
import os
from PIL import Image

app = Flask(__name__)

# Ensure 'static/uploads' directory exists
UPLOAD_FOLDER = "static/uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Load the trained model
model = tf.keras.models.load_model("multiple.h5")

# Define class labels (update according to your dataset)
class_labels = ["Apple", "banana", "guava", "mango","orange", "
    papaya", "pineapple", "waterapple", "watermelon"]

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

@app.route('/predict', methods=['POST'])
def predict():
    uploaded_files = request.files.getlist("images")  # Get
    multiple files
    predictions = []

    for file in uploaded_files:
        if file:
            # Save image to 'static/uploads' directory
```

```
            filepath = os.path.join(UPLOAD_FOLDER, file.filename)
            file.save(filepath)

            # Process image
            img = Image.open(file).resize((224, 224))  # Adjust size based on your model
            img = np.array(img) / 255.0  # Normalize
            img = np.expand_dims(img, axis=0)  # Add batch dimension

            # Predict
            pred = model.predict(img)
            predicted_class = class_labels[np.argmax(pred)]
            confidence = round(100 * np.max(pred), 2)

            predictions.append((file.filename, predicted_class, confidence))

    return render_template("result.html", predictions=predictions)

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

A.3 HTML Code for Landing Page

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Fruit Recognition System</title>
    <link href="https://fonts.googleapis.com/css2?family=Poppins
:wght@300;400;700&display=swap" rel="stylesheet">
    <style>
        body {
            font-family: 'Poppins', sans-serif;
            text-align: center;
            background: url("{{ url_for('static', filename='
fruits-bg.jpg') }}") no-repeat center center fixed;
            background-size: cover;
            color: white;
            margin: 0;
            padding: 0;
        }

        h1 {
            font-size: 50px;
            font-weight: 700;
            text-transform: uppercase;
            margin-top: 50px;
            text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);
            color: #f5f1f1;
        }

        .upload-box {
            background: rgba(255, 255, 255, 0.95);
            padding: 30px;
            border-radius: 12px;
            box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.3);
            display: inline-block;
            margin-top: 30px;
            width: 400px;
        }

        h2 {
```

```
            font-size: 24px;
            font-weight: 600;
            color: black;
            margin-bottom: 20px;
        }

        input[type="file"] {
            display: none;
        }

        label.upload-btn {
            display: inline-block;
            background: #ff6600;
            color: white;
            padding: 12px 24px;
            font-size: 18px;
            font-weight: 600;
            border-radius: 6px;
            cursor: pointer;
            transition: 0.3s;
        }

        label.upload-btn:hover {
            background: #ff4500;
        }

        button {
            background: #28a745;
            color: white;
            padding: 12px 24px;
            font-size: 18px;
            font-weight: 600;
            border: none;
            border-radius: 6px;
            cursor: pointer;
```

```
            transition: 0.3s;
            margin-top: 15px;
        }

        button:hover {
            background: #218838;
        }

        #file-count {
            margin-top: 10px;
            font-size: 16px;
            font-weight: 500;
            color: black;
        }
    </style>
</head>

<body>
    <h1>Fruit Recognition System</h1>
    <div class="upload-box">
        <h2>Upload images of fruits</h2>
        <form action="/predict" method="post" enctype="multipart/form-data">
            <input type="file" name="images" id="file-upload" multiple required>
            <label for="file-upload" class="upload-btn">Choose Files</label>
            <p id="file-count">No files chosen</p>
            <br><br>
            <button type="submit">Predict</button>
        </form>
    </div>

    <script>
```

```
106         const fileInput = document.getElementById('file-upload')
    ;
107         const fileCount = document.getElementById('file-count');
108 
109         fileInput.addEventListener('change', () => {
110             const count = fileInput.files.length;
111             fileCount.textContent = count > 0 ? `${count} file(s
    ) selected` : 'No files chosen';
112         });
113     </script>
114 </body>
115 
116 </html>
```

A.4 HTML Code for Result Page

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Prediction Results</title>
5      <style>
6          body {
7              font-family: Arial, sans-serif;
8              text-align: center;
9          }
10         .container {
11             display: flex;
12             flex-wrap: wrap;
13             justify-content: center;
14         }
15         .image-box {
16             margin: 15px;
17             padding: 10px;
18             border: 2px solid #ddd;
```

```
            border-radius: 10px;
            box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
            text-align: center;
        }
        img {
            width: 200px;
            height: 200px;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <h2>Prediction Results</h2>
    <div class="container">
        {% for image, fruit, confidence in predictions %}
        <div class="image-box">
            <img src="{{ url_for('static', filename='uploads/' + image) }}" alt="Fruit Image">
            <p><strong>Predicted:</strong> {{ fruit }}</p>
            <p><strong>Confidence:</strong> {{ confidence }}%</p>
        </div>
        {% endfor %}
    </div>
    <br>
    <a href="/">Upload More Images</a>
</body>
</html>
```