Title of your CEP here

CEP Report

By

NAME	RegistrationNumber
Syed Muhammad Hashir	CUI/FA21-BCE-032/ATD
Muhammad Hassan	CUI/FA21-BCE-051/ATD

For the course

Machine Learning

Semester Spring 2024

Supervised by:

Dr. Shoaib Azmat

Department of Electrical & Computer Engineering

COMSATS University Islamabad – Abbottabad Campus

DECLARATION

We Syed Muhammad Hashir (CUI/FA21-BCE-032/ATD), Muhammad Hassan (CUI/ FA21-BCE-032/ATD) hereby declare that we have produced the work presented in this report, during the scheduled period of study. We also declare that we have not taken any material from any source except referred to wherever due. If a violation of rules has occurred in this report, we shall be liable to punishable action.

Date: _____

_____
Syed Muhammad Hashir
(CUI/FA21-BCE-032/ATD)

_____
Muhammad Hassan
(CUI/FA21-BCE-032/ATD)

ABSTRACT

In this project, we developed a system to classify fresh and rotten fruits using Convolutional Neural Networks (CNNs). Our goal was to create an automated model to identify fruit quality, potentially reducing food waste and improving quality control in the food industry. We began by searching for a dataset of images showing various fruits in both fresh and rotten states. These images underwent preprocessing to enhance data quality and consistency, including resizing, and normalization to make the model more robust. A simple CNN architecture was then designed and implemented for the classification task. The model was trained on this dataset to learn the distinct features of fresh and rotten fruits. Once trained, we evaluated the model's performance using a separate test dataset. To further validate its practical application, we tested the model with images we captured ourselves. The results indicated that our CNN model could effectively differentiate between fresh and rotten fruits

This project highlights the potential of machine learning and computer vision in the agricultural and food sectors. By automating fruit quality assessment, our system can reduce the need for manual inspections and decrease the likelihood of rotten fruits reaching consumers. The simplicity of the CNN architecture also suggests that similar methods could be adapted for other food quality assessments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolution Neural Network

1 Introduction

Ensuring the quality of fruits is essential for reducing food waste, maintaining consumer satisfaction, and adhering to safety standards. Traditional manual inspection methods are time-consuming, labor-intensive, and prone to human error. With advancements in machine learning and computer vision, it is possible to automate this process, making it more efficient and reliable.

In this project, we developed a Convolutional Neural Network (CNN) model to classify fresh and rotten fruits. The automation of fruit quality assessment offers several benefits, including increased efficiency, consistency, and accuracy in identifying fruit quality. By reducing the need for manual inspections, the system can minimize human error and expedite the sorting process, ensuring only high-quality produce reaches consumers.

1.1 Objectives

Aims and objectives of our project include:

- Develop a robust image classification model to detect spoiled fruits.
- Automate the quality control process in agriculture to reduce manual labor and human error.
- Enhance the efficiency and accuracy of fruit sorting and packaging systems.
- Contribute to reducing food waste and improving food safety in the supply chain.

1.2 Features

Our model can predict whether fruits are fresh or rotten. It is capable of detecting and classifying images outside of its training dataset, demonstrating its generality and robustness. This ensures that the model is not limited to specific data and can be effectively applied in real-world scenarios.

2 Literature Survey

In developing the Fresh and Rotten Fruit classification model, we undertook a general literature survey to understand existing approaches and methodologies. The primary source of datasets was Kaggle, where we explored various datasets related to fruit classification. Kaggle also provided a wealth of codes and implementations by other people, which offered valuable understandings into different modeling techniques and preprocessing methods.

Beyond Kaggle, we conducted thorough online research using Google to find scholarly articles and open-source projects related to fruit classification using machine learning and computer vision. This research helped identify the best practices in data preprocessing, model architecture, and evaluation metrics.

Additionally, we utilized YouTube as a resource to watch tutorials and project walkthroughs. Many machine learning enthusiasts and professionals share their projects on YouTube, providing step-by-step guidance on data collection, preprocessing, model training, and deployment. These videos were instrumental in gaining a practical understanding of how to implement and fine-tune a CNN for fruit classification.

3 Proposed Methodology

The implementation of our CNN model involved several key steps. First, we collected a diverse dataset of fruit images in both fresh and rotten states. These images underwent preprocessing, including resizing, normalization to enhance their quality and consistency. We then designed and implemented a simple yet effective CNN architecture tailored for this classification task. The model was trained on the preprocessed dataset to learn the distinguishing features of fresh and rotten fruits. Following training, the model's performance was evaluated using a separate test set, and it was further validated with real-world images to ensure practical applicability.

3.1 Model Architecture

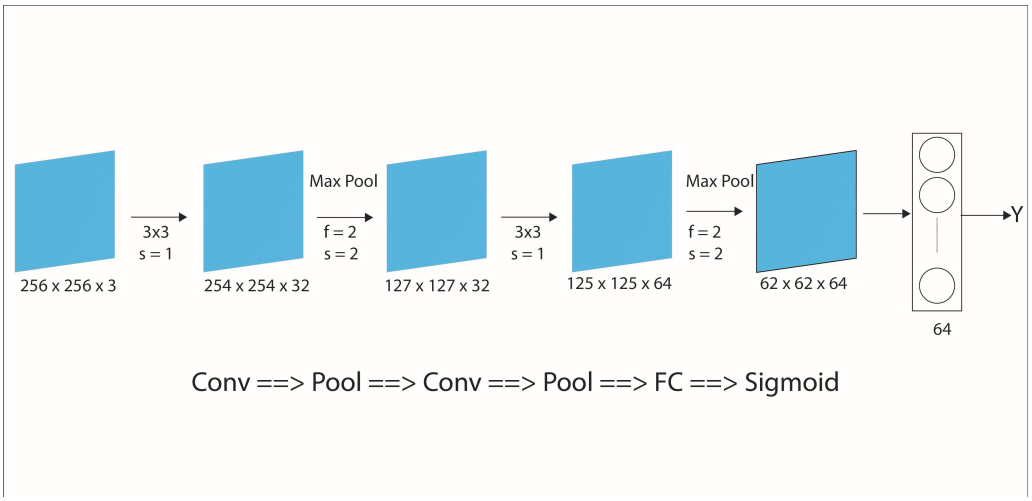

Fig: 3.1 Design Diagram

3.2 Flow chart

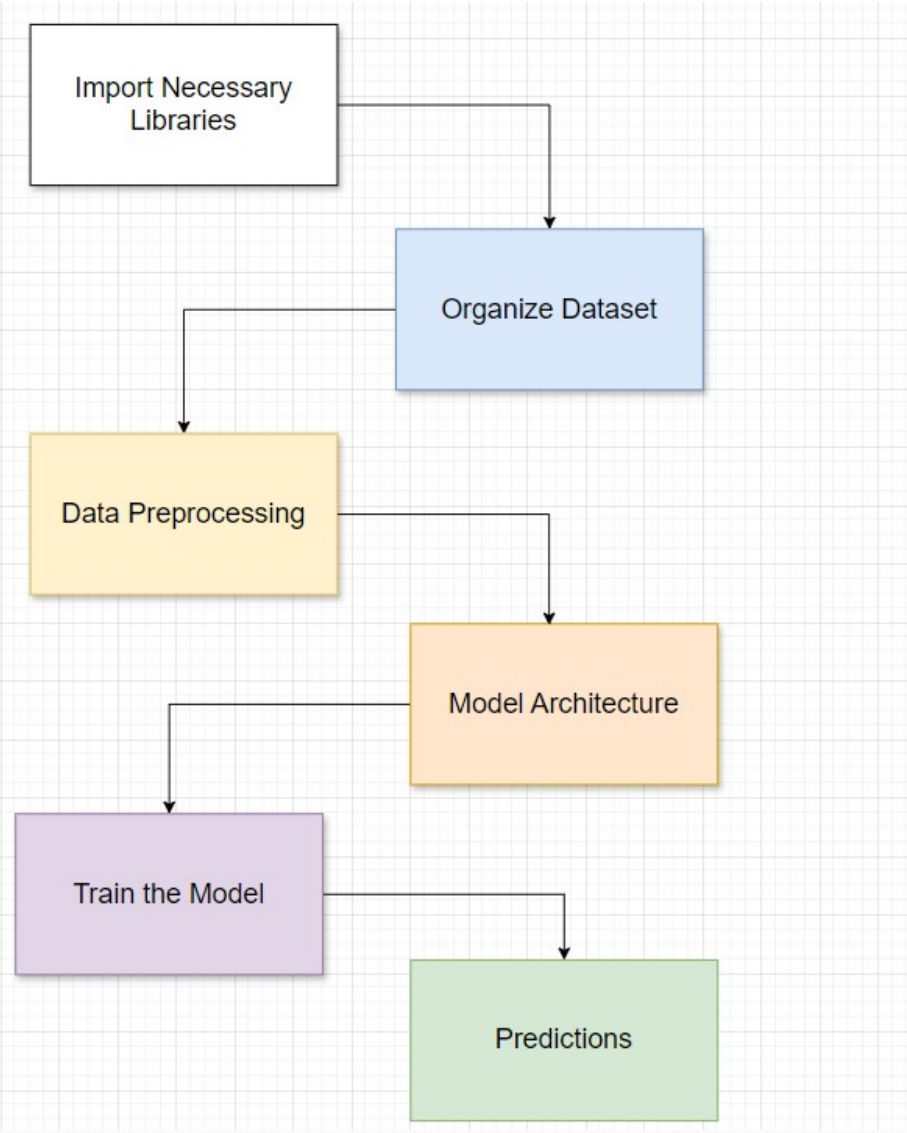

Fig: 3.2 Flow chart

4 Simulation

Test 1

We have run various tests on our model. First we do prediction on the test data which was already present in the dataset. Below are the result of that prediction. As the model has about 98% accuracy, so in the below 16 predicted images, all are correct. Sometime a prediction might get incorrect as it is not 100% accurate but most of the times, the prediction is correct.

Results

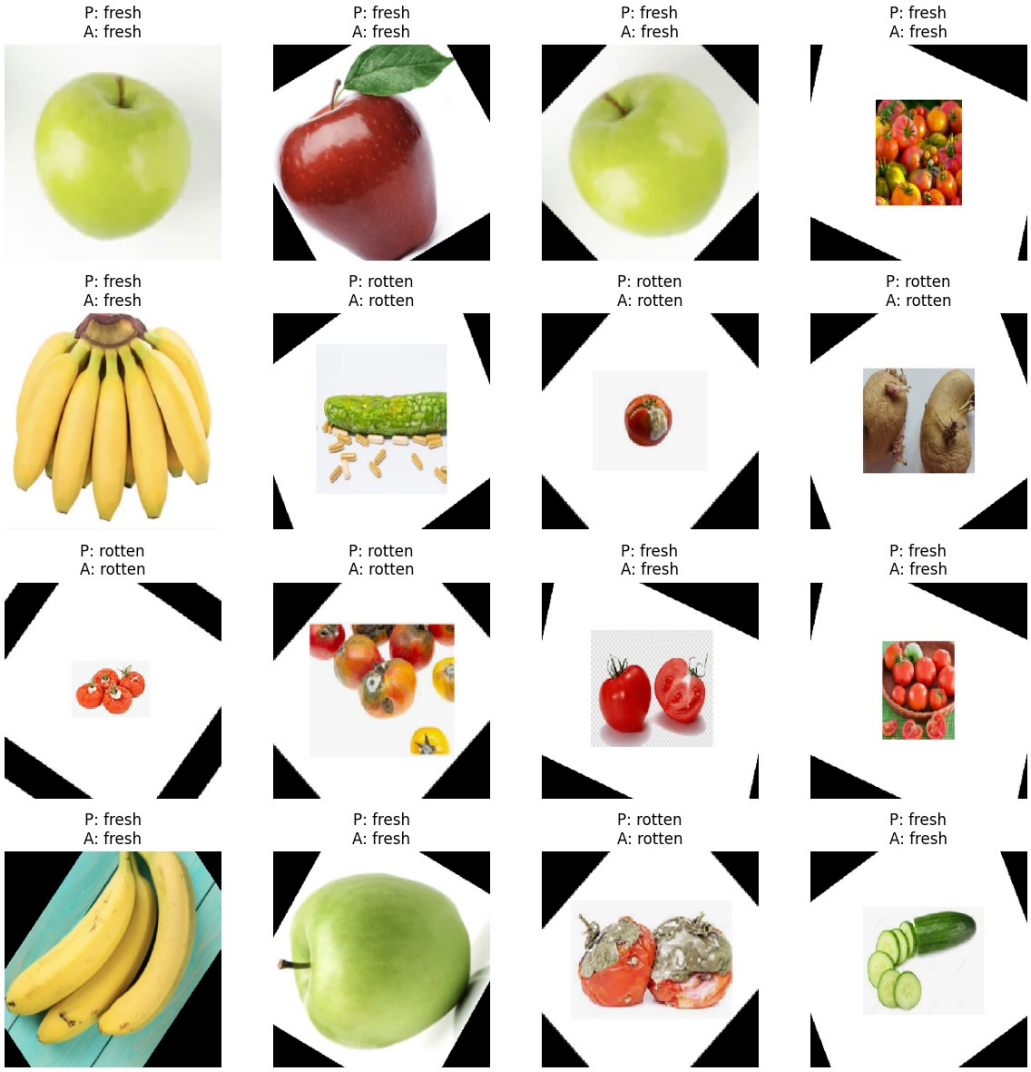

Fig: 4.1 simulation results 1

Test 2

In the following test, we have predicted our own collected images and tested it on the model. It can be seen that the model predicts correctly. Also we have tested the model on the images which were not part of the training dataset, it classifies those images as well into rotten and fresh category.

Results

Fig: 4.2 simulation results 2

The result below is of images not included in our original dataset

Fig: 4.3 simulation results 2

4.1Design parameters

Data-Related Parameters

Image Size: We have resized all our images to 256 x 256 x 3

Color Channels: We have used RGB images which have 3 channels

Data Augmentation Techniques: We have used rescaling method, and as for the rest of techniques, our dataset was already such that the images were from different angles and zoomed in or zoomed out so other data augmentation techniques were not necessary.

Model Architecture Parameters

Number of Layers: We have used 2 convolutional layers, 2 Max pool layers one fully connected layer and one dense layer.

Filter Size: Our filter size is 3x3

Number of Filters: The number of filters in 1st Convolution layer are 32, in 2nd 64

Stride: The step size of the convolution operation is 1

Padding: We have not set the padding parameter

Activation Functions: Convolutional as well as fully connected layers use ReLu activiation function except for the last layer which sigmoid as activation function.

Training Parameters

Batch Size: Batch size is 32

Learning Rate**:** Learning rate is 0.001

Number of Epochs: Number of epochs are 15

Optimizer: The optimizer we used is Adam

Loss Function: Our loss function is binary_crossentropy

Discussion

Our model successfully classifies fresh and rotten fruits with high accuracy. The use of a Convolutional Neural Network (CNN) allows the system to learn complex patterns and features from the fruit images, improving classification performance. Through testing

with both existing and new datasets, we confirmed the model's robustness and generalizability. The initial tests on the pre-existing test set yielded a high accuracy rate of 98%, demonstrating the effectiveness of our approach. Further testing on real-world images, including those outside the training dataset, showed that the model can accurately predict fruit quality in diverse scenarios. The model's ability to generalize well to new images is crucial for practical applications in the agriculture and food industries. This project highlights the potential for machine learning to streamline quality control processes, reduce human error, and improve overall efficiency.

Conclusion

In this project, we developed a CNN model to classify fresh and rotten fruits, achieving a high accuracy of 98%. The model's ability to accurately predict fruit quality, even with images outside its training dataset, demonstrates its robustness and practicality. Automating fruit quality assessment can significantly reduce manual labor, minimize human error, and improve efficiency in the food supply chain. Our approach shows promise in contributing to food waste reduction and enhancing food safety. This project underscores the potential of machine learning and computer vision technologies in revolutionizing agricultural practices and ensuring the delivery of high-quality produce to consumers.

5. References

Should be in standard IEEE format…

[1]. Nayak, S. S. (2023). Fresh and Stale Classification. Kaggle [online] Available at: https://www.kaggle.com/datasets/swoyam2609/fresh-and-stale-classification Accessed on 07-06-2024.

6. Appendix

```
import os
import shutil
import numpy as np
import random
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf


# Organizing the dataset
# These paths will be used ahead
root_path = 'D:/Cui atd/Sem 6/Machine Learning/CEP_ML/Dataset/dataset'
output_path = 'D:/Cui atd/Sem 6/Machine Learning/Output_CEP'

for x in ['Train', 'Test']:
    path = os.path.join(root_path, x)
    content = os.listdir(path)
    os.makedirs(os.path.join(output_path, x, '2Fresh'), exist_ok=True)
    os.makedirs(os.path.join(output_path, x, '1Rotten'), exist_ok=True)
    for sub_dir in tqdm(content):
        sub_dir_path = os.path.join(path, sub_dir)
        if os.path.isdir(sub_dir_path):
            image_paths = os.listdir(sub_dir_path)
            for image in image_paths:
                source = os.path.join(sub_dir_path, image)
                if sub_dir[0] == 'f':
                    destination = os.path.join(output_path, x,
'2Fresh', image)
                elif sub_dir[0] == 'r':
                    destination = os.path.join(output_path, x,
'1Rotten', image)
                shutil.copy(source, destination)

# Here we are creating ImageDataGenerator instances
generator = ImageDataGenerator(rescale=1/255.)

# Creating train and test generators
# Train generator is given the path till Train
train_generator = generator.flow_from_directory(
    os.path.join(output_path, 'Train'),
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary'
```

```
)
# Test generator is given the path till Test
test_generator = generator.flow_from_directory(
    os.path.join(output_path, 'Test'),
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary'
)

test_generator.class_indices

import tensorflow as tf
from tensorflow.keras.models import load_model
new_model =
tf.keras.models.load_model('C:/Users/m_has/Downloads/ML_BC.keras',
compile = False)
# Compiling the model
new_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00
1),
              loss='binary_crossentropy',
              metrics=['accuracy'])

new_model.summary()



import matplotlib.pyplot as plt
import numpy as np
import random

num_images = 16

# Here we retreive a batch of data for actual vs predicted comparison
batch = next(test_generator)
X_test_batch, Y_test_batch = batch

# If the batch size is smaller than num_images, then adjust it
accordingly
num_images = min(num_images, len(X_test_batch))

fig, axes = plt.subplots(4, 4, figsize=(12, 12))

for i in range(num_images):
    # For displaying image
    ax = axes[i // 4, i % 4]
```

```
    ax.imshow(X_test_batch[i])
    ax.axis('off')

    # Prediction on the image
    y_pred = new_model.predict(X_test_batch[i].reshape(1, 256, 256, 3))
    predicted_class = np.where((y_pred) >= 0.5, 1, 0)

    # Map predicted class to label
    if predicted_class == 1:
        pred = 'fresh'
    else:
        pred = 'rotten'
    actual_class = int(Y_test_batch[i])  # Assuming Y_test_batch is
one-hot encoded
    if actual_class == 1:
        actual = 'fresh'
    else:
        actual = 'rotten'
    ax.set_title(f"P: {pred}\nA: {actual}")

plt.tight_layout()
plt.show()

# Printing the predictions and actual labels for each displayed image
for i in range(num_images):
    y_pred = new_model.predict(X_test_batch[i].reshape(1, 256, 256, 3))
    predicted_class = np.where((y_pred) >= 0.5, 1, 0)

    if predicted_class == 1:
        pred = 'fresh'
    else:
        pred = 'rotten'

    actual_class = int(Y_test_batch[i])

    if actual_class == 1:
        actual = 'fresh'
    else:
        actual = 'rotten'

    print(f"Image {i + 1}: Predicted as {pred}, Actual: {actual}")


# Predicting on our own dataset
import matplotlib.pyplot as plt
```

```
import numpy as np
from tensorflow.keras.preprocessing import image
import tensorflow as tf

# Load and preprocess the image
def load_and_preprocess_image(img_path, target_size=(256, 256)):
    img = image.load_img(img_path, target_size=target_size)
    img_array = image.img_to_array(img)
    # Expand dimensions to match the expected input shape
    img_array = np.expand_dims(img_array, axis=0)
    # Normalize the image array
    img_array = img_array / 255.0
    return img_array

# Display the image
def display_image(img_path):
    img = image.load_img(img_path)
    plt.imshow(img)
    plt.axis('off')
    plt.show()

# Prediction function
def predict_image(img_path, model):
    img_array = load_and_preprocess_image(img_path)
    display_image(img_path)
    y_pred = new_model.predict(img_array)
    predicted_class = np.where(y_pred >= 0.5, 1, 0)

    if predicted_class == 1:
        pred = 'fresh'
    else:
        pred = 'rotten'

    print("Our model predicts it is:", pred)
img_path = 'D:/Cui atd/Sem 6/Machine
Learning/Dataset_Ourselves/RottenApple.jpg'
predict_image(img_path, new_model)
```

Teachers should assess CLO2, CLO3 and CLO4 based on the given rubrics (overall weightage 20%)

Recommended Percentage Breakdown

CLO	**Percentage**
CLO2 (Investigation)	10%
CLO3 (Referencing/Citations) *(Turnitin report should be generated.)*	5%
CLO4 (Communication)	5%