Classification of Fruit Types and Conditions in Agriculture

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-051/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-051/ATD)

ABSTRACT

In this project, we developed a system to classify various fruits and their conditions (fresh or rotten) using Convolutional Neural Networks (CNNs). Our goal was to create an automated model to identify both fruit types and their quality, potentially reducing food waste and improving quality control in the food industry. We began by searching for a dataset of images showing different 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 both the fruit types and their conditions. Once trained, we evaluated the model's performance using a separate test dataset.

This project highlights the potential of machine learning and computer vision in the agricultural and food sectors. By automating fruit quality and type 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 various fruits and their conditions (fresh or rotten). The automation of fruit quality and type assessment offers several benefits, including increased efficiency, consistency, and accuracy in identifying both the fruit type and its 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, their types, 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 can also predict the type of fruit. This will help in packaging different fruits effectively.

2 Literature Survey

In developing of the model Classification of Fruit types and their conditions in Agriculture, 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 along with their types. Following training, the model's performance was evaluated using a separate test set.

Model Architecture

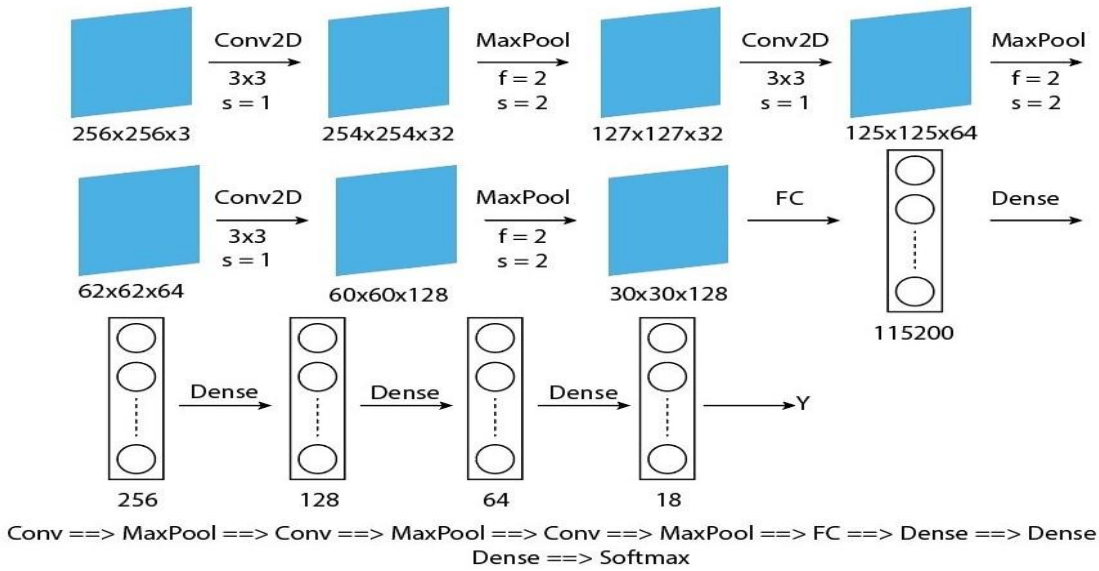

Fig: 3.1 Design Diagram

3.2 Model Summary

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_7 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_7 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_8 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_8 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten_2 (Flatten)	(None, 115200)	0
dense_8 (Dense)	(None, 256)	29,491,456
dense_9 (Dense)	(None, 128)	32,896
dense_10 (Dense)	(None, 64)	8,256
dense_11 (Dense)	(None, 18)	1,170

Total params: 29,627,026 (113.02 MB)

Trainable params: 29,627,026 (113.02 MB)

Non-trainable params: 0 (0.00 B)

Fig: 3.2 Model summary

3.3 Epochs and the corresponding Testing and Validation accuracy

Note:

This result is of 11th-13th epoch, before this the model has been run for 10 epochs.

```
Epoch 1/5
591/591 ━━━━━━━━━━━━━━━━━━━━ 167s 279ms/step - accuracy: 0.9609 - loss:
0.7556 - val_accuracy: 0.7698 - val_loss: 2.6526
Epoch 2/5
591/591 ━━━━━━━━━━━━━━━━━━━━ 151s 254ms/step - accuracy: 0.9751 - loss:
0.4044 - val_accuracy: 0.7643 - val_loss: 4.1180
Epoch 3/5
342/591 ━━━━━━━━━━━━━━━━━━━━ 50s 203ms/step - accuracy: 0.9799 - loss:
0.0607
```

3.4 Flow chart

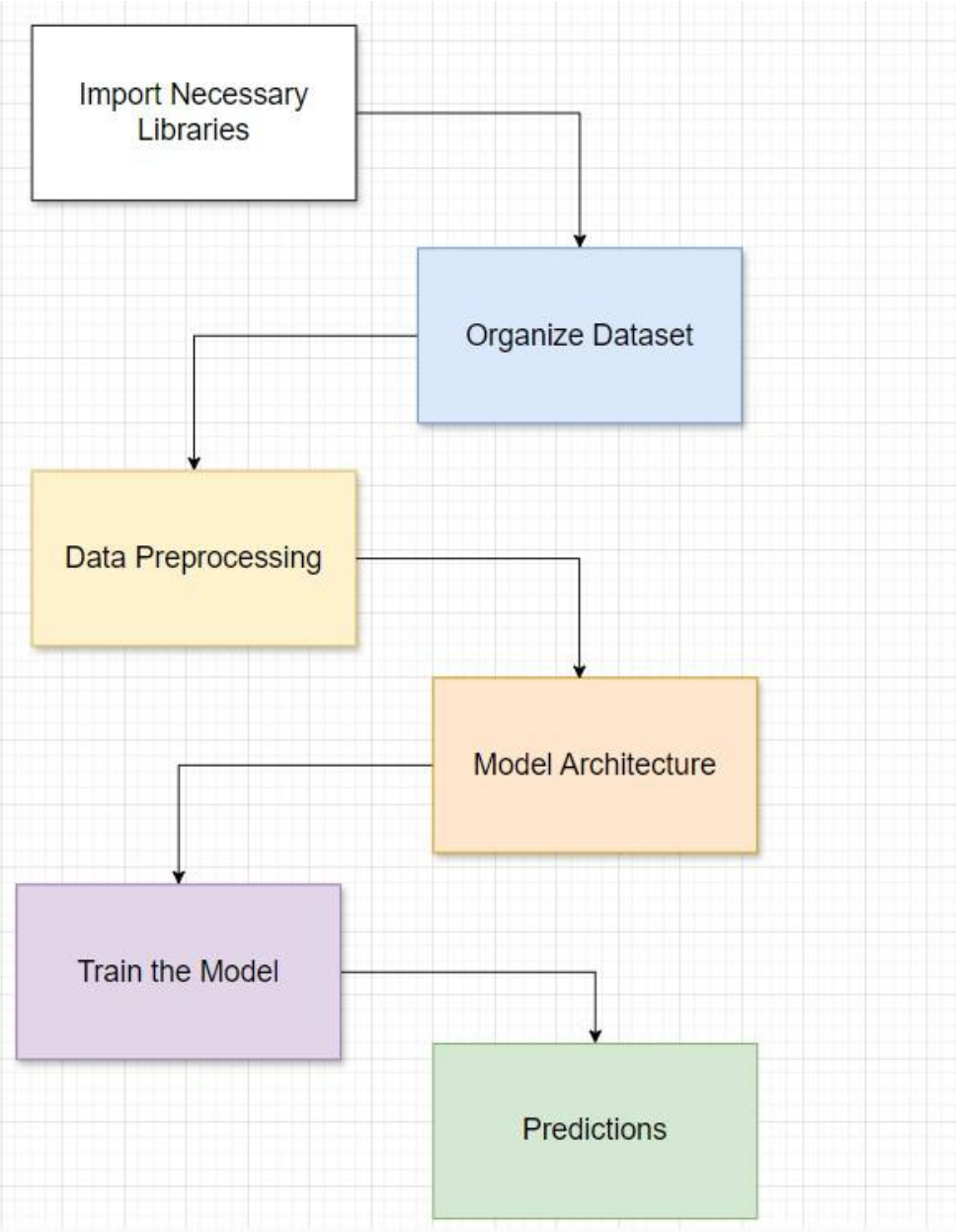

Fig: 3.3 Flow chart

4 Simulation

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 76% test accuracy, so in the below 16 predicted images, not all images are predicted correct.

Results

Fig: 4.1 simulation results

4.1 Design 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 3 convolutional layers, 3 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 and in 3rd we have used 128.

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 13

Optimizer: The optimizer we used is Adam

Loss Function: Our loss function is categorical_crossentropy

Discussion

Our model successfully classifies fresh and rotten fruits along with fruit types with medium 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. The initial tests on the pre-existing test set yielded a medium accuracy rate of 76%, demonstrating the effectiveness of our approach. We tried different model architectures to improve testing and validation accuracy and to overcome te overfitting issue in our training. These different approaches included applying data augmentaion and regularization techniques and changing mode hyperperameters. All of those models were overfitting and there seemed no real improvement.

Conclusion

In this project, we developed a CNN model to classify fresh and rotten fruits along with the fruit type, achieving a medium accuracy of 76%. The model's ability to predict fruit quality and fruit type, demonstrates its 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

[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 numpy as np
import random
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

# Organize the dataset
root_path = '/kaggle/input/fresh-and-stale-classification/dataset'
output_path = '/kaggle/working'

generator = ImageDataGenerator(
rescale=1./255,
#     width_shift_range=0.2,
#     height_shift_range=0.2,
#     zoom_range=0.2,
#     horizontal_flip=True,
#     fill_mode='nearest',
validation_split=0.2
)


# Here we are creating train and test generators
train_generator = generator.flow_from_directory(
os.path.join(root_path, 'Train'),
target_size=(256, 256),
batch_size=32,
class_mode='categorical',
subset='training',
interpolation = 'bilinear'
)

test_generator = generator.flow_from_directory(
os.path.join(root_path, 'Train'),
target_size=(256, 256),
batch_size=32,
class_mode='categorical',
subset='validation',
interpolation = 'bilinear'
)
test_generator.class_indices
```

```
import tensorflow as tf
# Defining our model
model = tf.keras.models.Sequential([
tf.keras.layers.Conv2D(32, (3, 3), activation="relu", input_shape=(256,
256, 3)),
tf.keras.layers.MaxPool2D((2, 2)),

tf.keras.layers.Conv2D(64, (3, 3), activation="relu"),
tf.keras.layers.MaxPool2D((2, 2)),

tf.keras.layers.Conv2D(128, (3, 3), activation="relu"),
tf.keras.layers.MaxPool2D((2, 2)),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(256, activation="relu"),
tf.keras.layers.Dense(128, activation="relu"),
tf.keras.layers.Dense(64, activation="relu"),

tf.keras.layers.Dense(18, activation="softmax")
])

model.summary()

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])


# Train the model
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3)

history = model.fit(train_generator,
                    epochs=5,
validation_data=test_generator,callbacks=[early_stopping])

# Evaluate the model
loss, accuracy = model.evaluate(test_generator)
print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')


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

```
# Fetch a batch of data
batch = next(test_generator)
X_test_batch, Y_test_batch = batch

# Predicting on the batch
predictions = model.predict(X_test_batch)

# Defining number of images to display
num_images = min(16, len(X_test_batch))

# Here we are getting class labels from the generator
class_indices = test_generator.class_indices
labels = {v: k for k, v in class_indices.items()}

plt.figure(figsize=(15, 15))

for i in range(num_images):

image = X_test_batch[i]


predicted_class_idx = np.argmax(predictions[i])
predicted_class = labels[predicted_class_idx]


actual_class_idx = np.argmax(Y_test_batch[i])
actual_class = labels[actual_class_idx]

plt.subplot(4, 4, i + 1)
plt.imshow(image)
plt.title(f"Prediction: {predicted_class}\nActual: {actual_class}")
plt.axis('off')

plt.tight_layout()
plt.savefig('sv.png')
plt.show()
```

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%