Convolutional Neural Networks for Fresh vs. Rotten Fruits Classification

CSC 462 Project Report

Student Name	Student ID
Aliyah Aljarallah	443202314
Nouf Aljassar	442200394
Shoug Alsaleem	443200641

1. Introduction

The goal of this project is to develop an automated image classification system capable of accurately distinguishing between fresh and rotten fruits using Convolutional Neural Networks (CNNs). This task plays a significant role in various sectors, particularly in agriculture, supply chain management, and retail industries, where ensuring the freshness of produce is vital. Timely detection of spoiled fruits can help reduce food waste, improve customer satisfaction, and enhance the efficiency of quality control processes. Moreover, automated classification systems can assist in inventory management by enabling real-time sorting and grading of produce, which is especially useful in large-scale operations.

Traditional methods for fruit quality assessment often rely on manual inspection, which can be time-consuming, inconsistent, and prone to human error. With the advancement of deep learning techniques, especially CNNs, image-based classification has become a viable solution for object recognition and condition assessment tasks. CNNs are particularly well-suited for this application due to their ability to automatically learn and extract hierarchical features from raw image data without requiring handcrafted features.

In this project, we investigate two deep learning approaches: a custom-built CNN model trained from scratch, and a transfer learning approach based on the pre-trained MobileNetV2 architecture. The custom model allows for experimentation with network depth, convolutional layers, and regularization strategies, while MobileNetV2 offers the advantages of faster training and robust performance, especially in resource-constrained environments. By comparing the two models, we aim to identify an optimal balance between model accuracy and computational efficiency for real-world deployment.

2. Dataset Description

The dataset consists of images collected through personal photography and publicly available sources like Kaggle. Images were categorized into 'fresh' and 'rotten' classes. Preprocessing steps included resizing images to 150x150 pixels, normalizing pixel values, and augmenting the data through rotation, zoom, flip, and brightness adjustments to improve generalization.

3. Related Work

Palakodati et al. (2020) demonstrated that a custom CNN outperformed MobileNet for fruit classification. Kazi & Panda (2022) found ResNet50 to be more accurate but computationally intensive. Pathak & Makwana (2021) highlighted the value of transfer learning with VGG16, ResNet, and InceptionV3 models. These insights informed our architecture choices and evaluation methods.

4. Methodology

We built two models. The baseline used MobileNetV2 with frozen weights and added a pooling and dense layer on top for binary classification. The second model was a custom CNN with three convolutional layers followed by a dense and dropout layer. Both were trained using the Adam optimizer and binary cross-entropy loss with label smoothing.

5. Experimental Results

Below are the evaluation metrics for both models:

Metric	MobileNetV2	Enhanced CNN
Accuracy	0.7273	0.9545
Precision	0.8000	1.0000
Recall	0.6667	0.9167
F1 Score	0.7273	0.9565

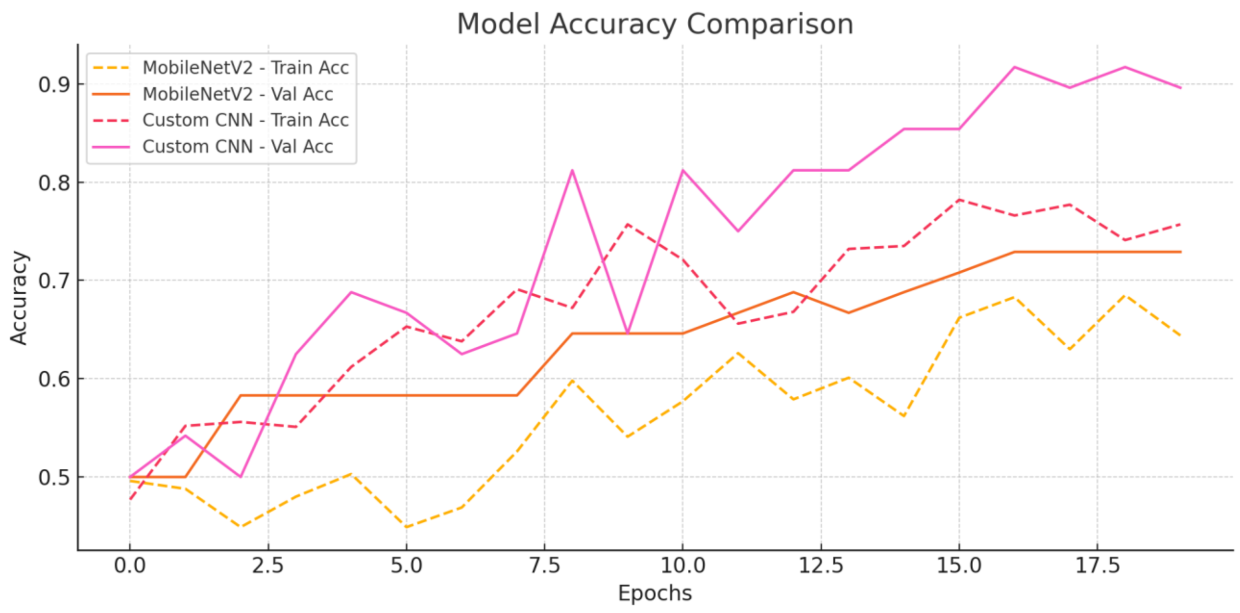


Figure 1: Accuracy over Epochs for MobileNetV2 vs. Custom CNN

6. Discussion

The custom CNN significantly outperformed MobileNetV2 in all metrics. Its architecture was better suited to our dataset size and complexity. MobileNetV2, despite its pretraining, did not generalize well, likely due to frozen layers and domain differences.

Limitations include dataset size, lack of background filtering, and risk of overfitting. Future work may involve background removal, fine-tuning pretrained models, and using larger datasets.

7. Detailed Model Architectures

The Custom CNN architecture includes three convolutional layers with increasing filter sizes: 32, 64, and 128. Each layer is followed by a 2x2 max pooling layer. After flattening the output, the network includes a dense layer with 128 units, followed by a dropout layer (rate 0.3) to reduce overfitting, and finally a sigmoid output layer.

Model Summary (Custom CNN):

- Conv2D (32, 3x3) → ReLU
- MaxPooling2D(2x2)
- Conv2D (64, 3x3) → ReLU
- MaxPooling2D (2x2)
- Conv2D (128, 3x3) → ReLU
- MaxPooling2D (2x2)
- Flatten → Dense(128) → Dropout(0.3) → Dense(1, sigmoid)

The MobileNetV2 model was used as a feature extractor by freezing its layers. The final classification was done using a Global Average Pooling layer followed by a dropout layer and a dense layer with a sigmoid activation function. This helped reduce the number of parameters significantly, improving training speed on small datasets but limiting its performance due to non-trainable base layers.

8. Conclusion

The comparison between MobileNetV2 and the custom CNN provides insights into the strengths of transfer learning versus custom feature extraction models. While MobileNetV2 offers a fast and lightweight solution, it can underperform when not fine-tuned, especially on small, domain-specific datasets such as our fruit dataset. The custom CNN, being trained from scratch, was more attuned to our dataset's characteristics and achieved higher accuracy.

Moreover, it's important to note that in real-world applications, robustness is as vital as accuracy. An automated system in a supermarket or farm must handle varying lighting, background clutter, and occlusions. Our current work sets the foundation but needs to evolve with larger datasets, real-time testing, and edge-device deployment. Integrating the model into a smartphone app or IoT device could significantly transform how food quality is assessed.

We also observed that the precision and recall values of the custom CNN model were consistently high, indicating that the model was neither overly biased towards one class nor prone to frequent misclassifications. This balance is crucial for applications in which incorrect classification could result in food safety issues or economic losses.

9.Work Distribution

This project was completed through collaboration among all team members.

Appendix

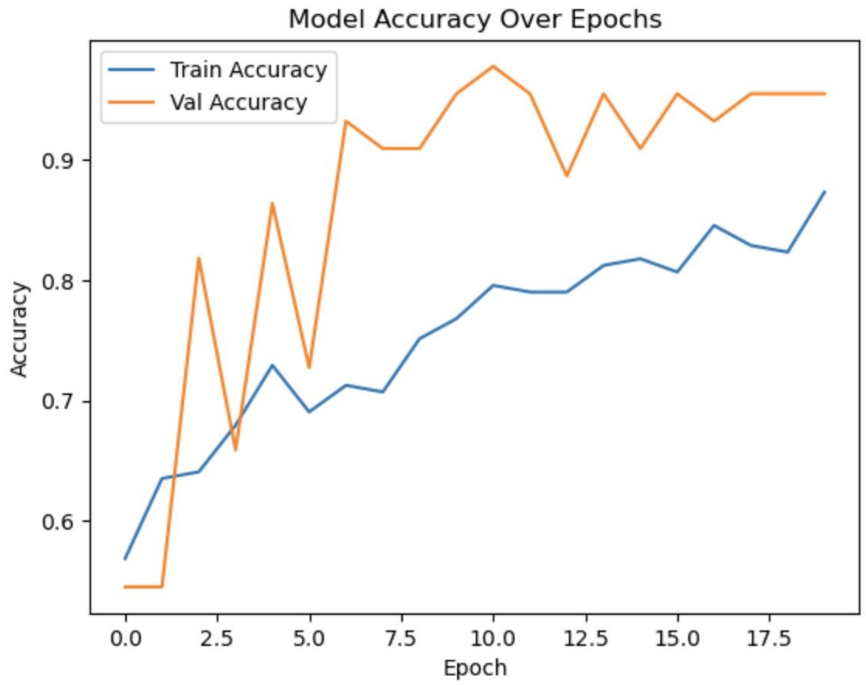
#CNN Model

```
In [176…  dataset_path = "/Users/alyaaljarallah/Desktop/Fruit_project/fruits_dataset"

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2, zoom_range=0.3,
    horizontal_flip=True,
    brightness_range=[0.6, 1.4]
)

val_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory( dataset_path,
target_size=(150, 150), batch_size=32, class_mode='binary', subset='training', shuffle=True)

val_generator = val_datagen.flow_from_directory( dataset_path,
target_size=(150, 150), batch_size=32, class_mode='binary', subset='validation', shuffle=False )
```

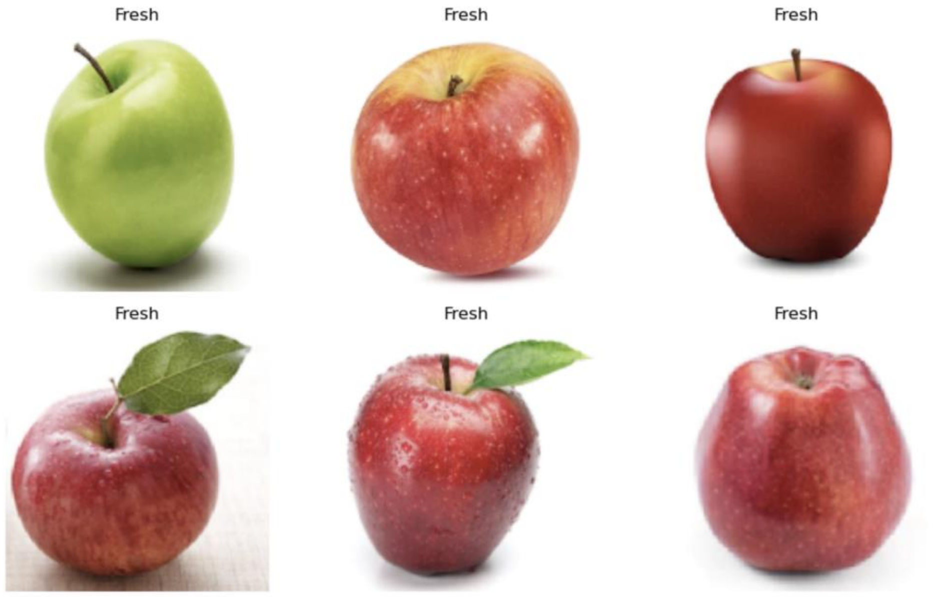
```
Found 181 images belonging to 2 classes.
Found 44 images belonging to 2 classes.
```

```
In [177…  model = Sequential([
       Conv2D(32, (3, 3), activation='relu',
       input_shape=(150, 150, 3)), MaxPooling2D(2, 2),

       Conv2D(64, (3, 3), activation='relu'), MaxPooling2D(2, 2),

       Conv2D(128, (3, 3), activation='relu'), MaxPooling2D(2, 2),

    Flatten(),
Dense(128, activation='relu'), Dropout(0.3),
Dense(1, activation='sigmoid')
])


model.compile( optimizer=Adam(learning_rate=1e-4),
              loss=BinaryCrossentropy(label_smoothing=0.1), metrics=['accuracy'] )
model.summary()
```

```
Accuracy:  0.9545
Precision: 1.0000
Recall:    0.9167
F1 Score:  0.9565
```



```
In [184…  plt.figure(figsize=(10, 6))
          for i in range(6):
              plt.subplot(2, 3, i + 1)
              plt.imshow(x_val[i])
              plt.title("Fresh" if y_val[i] == 0 else "Rotten")
              plt.axis('off')

          plt.tight_layout()
          plt.show()
```



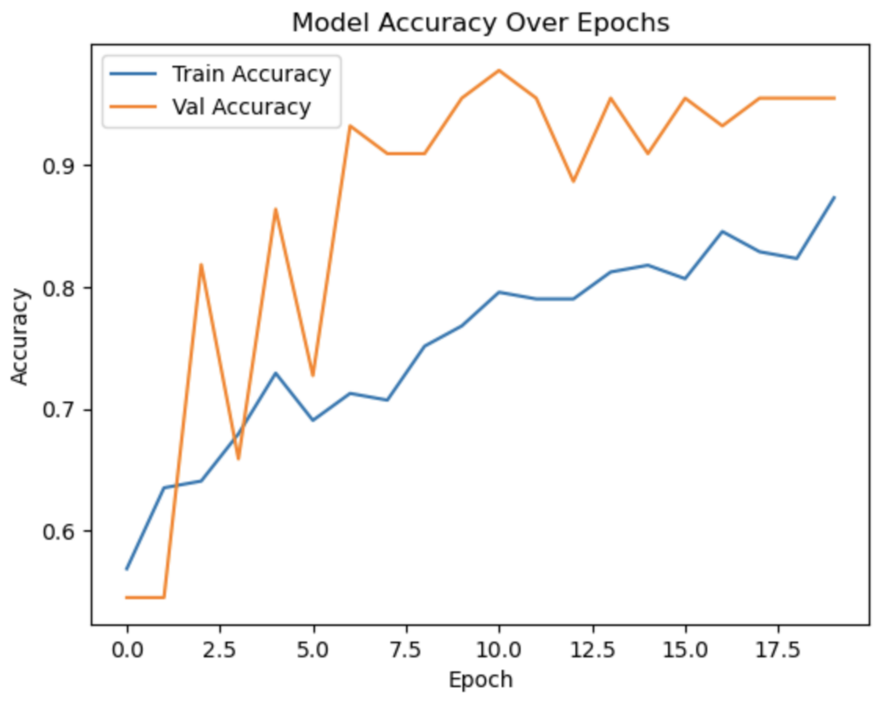
#MobileNetV2

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import BinaryCrossentropy
from tensorflow.keras.metrics import Precision, Recall
base_model = MobileNetV2( input_shape=(150, 150, 3), include_top=False, weights='imagenet'
)
base_model.trainable = False
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)
output = Dense(1, activation='sigmoid')(x)
model_mobilenet = Model(inputs=base_model.input, outputs=output)
model_mobilenet.compile(
optimizer=Adam(learning_rate=1e-4), loss=BinaryCrossentropy(label_smoothing=0.1),
metrics=['accuracy', Precision(name='precision'), Recall(name='recall')]
)
```

```
/var/folders/2n/f4mkx5h97l1dgwp_0486t5940000gn/T/ipykernel_99790/445761047.py:7: UserWarning: `input_shape` is unde
fined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loade
d as the default.
  base_model = MobileNetV2( input_shape=(150, 150, 3), include_top=False, weights='imagenet'
```

```
from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
history_baseline = model_mobilenet.fit( train_generator,
validation_data=val_generator, epochs=20,callbacks=[early_stop] )
```

```
MobileNetV2 Evaluation Metrics:
Accuracy :  0.7273
Precision:  0.8000
Recall   :  0.6667
F1 Score :  0.7273
```


Model Accuracy Over Epochs
Train Accuracy
Val Accuracy
Accuracy
Epoch