UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Artificial Intelligence and Machine Learning

(6CS012)

Image Classification Task

Fruit Classification

Student Id : 2329256

Student Name : Sanjeev Kumar Sah

Group : L6CG3

Tutor : Mr. Aatiz Ghimire

Lecturer : Ms. Sunita Parajuli

Date : 15th May 2025

Abstract

In this report we have built three models CNN, FCNN and Transfer Learning model with fine-tuning for image classification task using the dataset “Fruit Classification” the dataset contains 5 different categories Banana, Cherry, Grape, Mango and Peach and 6000+ labelled images. Here we start first by understanding our data by organizing it to the appropriate directories and splitting it into training and test validation in the 80:20 ratio. Then we checked for corrupted images and validated images to remove corrupted files. Data augmentation techniques we applied and visualize class distribution. Our main goal is to develop and evaluate deep learning models that can accurately classify fruits based on images. After we built three different, the baseline model achieved a test accuracy of 84%, while CNN has a improved accuracy of 89% and the best model, transfer learning with MobileNetV2 which achieved accuracy of 94% which is the most impressive and better performed. From the above results discussed we can conclude that transfer learning significantly boosts performance for image classification with limited data. This report also highlights the importance of model depth, regularization, and pre-trained features in enhancing performance.

Table of Contents

Table of Figure

1. Introduction

Image classification is a challenge in computer vision which is the process of giving an image a name or category based on its visual information. Numerous real-world applications, including automated quality inspection in agriculture, retail inventory management, and illness identification in healthcare, depend on it. Accurate fruit classification, for example, can facilitate effective sorting, freshness detection, packing automation, and nutritional tracking in the agriculture industry.

Deep learning technologies have fundamentally transformed image classification tasks. Convolutional Neural Networks (CNNs), with their ability to learn spatial hierarchies of features via convolution and pooling layers, are now regarded as one of the most efficient architectures. Unlike traditional approaches in machine learning where features were manually designed, CNNs extract pertinent features from raw pixel data, leading to higher precision and enhanced generalization capabilities.

This aim of this task is to classify images of fruits into five categories Banana, Cherry, Grape, Mango, and Peach. The classification task was approached with three deeply learned models with increasing complexity: a CNN baseline which was manually constructed, a deeper CNN with regularization layers to mitigate overfitting, and a MobileNetV2 architecture transfer learning model.

These primary goals are:

- To examine and portray the layout and concentration of the fruit dataset components.
- To record the performance metrics of a baseline CNN model and assess its image classification capabilities.
- To increase classification accuracy using deep architectures with added regularization through techniques such as BatchNormalization and Dropout.
- To evaluate performance difference on models learned from scratch versus those utilizing transfer learning on a pre-trained CNN model.

This project shows the value of model architecture and tuning in deep learning, including how utilizing pre-existing models can enhance performance and training, particularly with limited data.

2. Dataset

The dataset used for this task is fruit classification image which is provided by an instructor and is found on Kaggle. Dataset contains images with annotations for five classes of fruit: Banana, Cherry, Grape, Mango, and Peach. Each fruit image is captured from different angles, under different light conditions, and in front of diverse backgrounds which adds complexity to the classification task.

- Source: Given by the instructor (also accessible from Kaggle)
- Number of Classes: 5
- Total Valid Images: 6242
- Image Resolution: Originally varied, all were resized to 180 × 180 pixels for modeling input
- Test Set: 5 images one from each class

The dataset was prepared for training with the following preprocessing methods:

- Image Re-sizing: All images were brought to a common size of 180 × 180 × 3.
- Normalization: The values of the pixels were adjusted by dividing by 255, thus setting them within a range of 0 and 1.
- Image Validation: Using OpenCV and PIL, every image was checked for possible corruption. There were 35 corrupt images which were subsequently removed.
- Clean Dataset Directory: A folder structure was built for only keeping valid images that have no corruption and are non-defective.

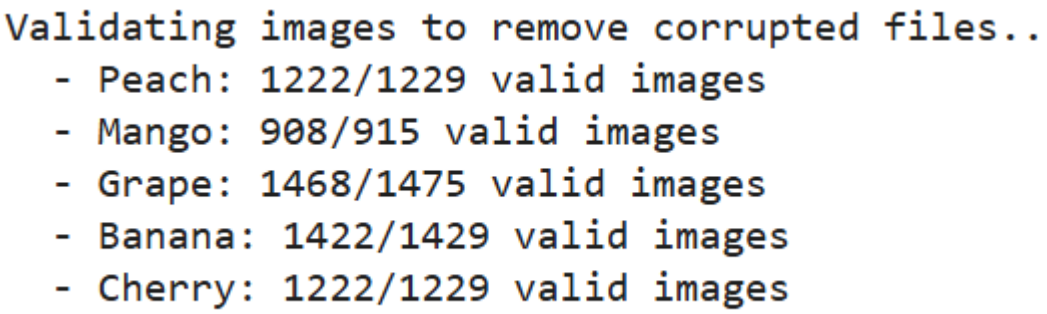

Figure 1 Dataset Statistics

The dataset is split into 80% for training and 20% for validation set within the training pipeline using Kera's ImageDataGenerator with validation_split.

Data augmentation is applied to improve generalization and reduce overfitting by the following transformation:

- Random rotation
- Horizontal and vertical flips
- Shift of width and height of the image
- Zooming

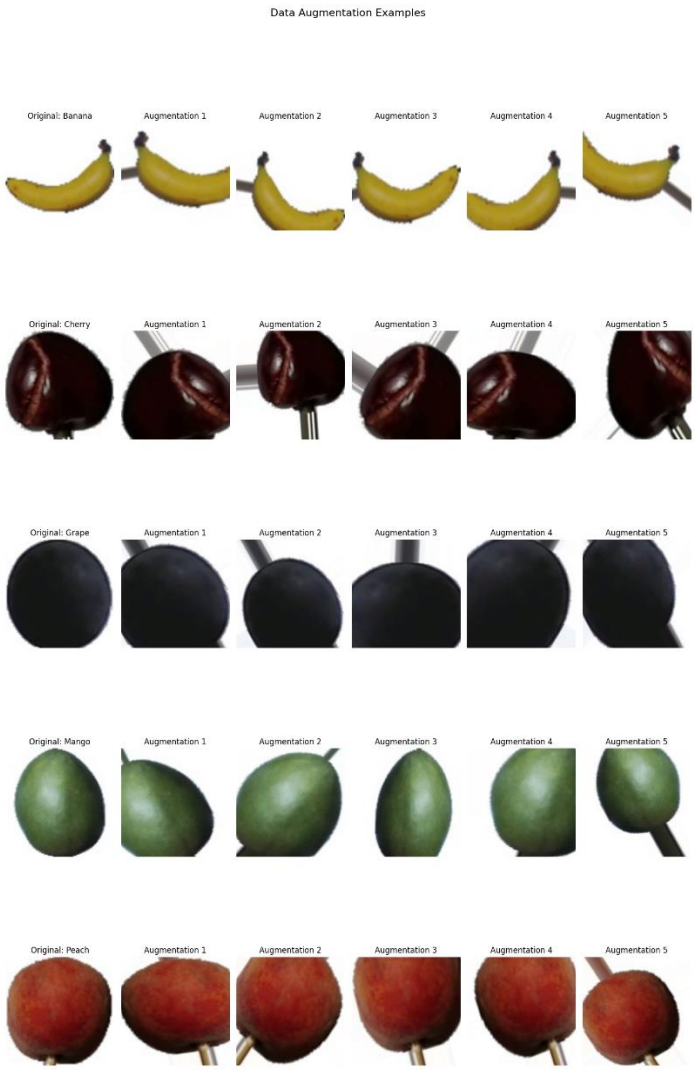

Figure 2 Data Augmentation

Challenges faced during data collection are:

Although the dataset is provided by the instructor. So rather than collecting data there are few challenges like corrupted images, class imbalance and so on.

3. Methodology

This section outlines the design and training of all three models of deep learning used for the classification of fruit dataset. All models were developed progressively by starting from basic CNN and advanced towards transfer learning using MobileNet2.

3.1. Baseline CNN Model

Architecture of the baseline model was constructed with the Sequential API in Keras. It comprises the following elements:

- 3 Convolutional Layers: Filter sizes are 32, 64, and 128 and also utilizes ReLU activation and kernel size (3×3).
- 3 MaxPooling Layers: Each of the pooling layers is preceded by a convolution layer in order to decrease the spatial dimensions.
- Flatten Layer.
- 3 Fully Connected (Dense) Layers: units [256, 128, 64]. Employs Dropout (0.5) for overfitting prevention.
- Output Layer: 5 units with Dense and Softmax activation for classification.

Baseline Model Summary:
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 512)	51,380,736
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 128)	32,896
dense_3 (Dense)	(None, 5)	645

Total params: 51,638,853 (196.99 MB)
Trainable params: 51,638,853 (196.99 MB)
Non-trainable params: 0 (0.00 B)

Figure 3 Baseline CNN Model Summary

Training model

- Loss Function: categorical_crossentropy
- Optimizer: Adam
- Batch Size: 32
- Epochs: 20
- Learning Rate: 0.001 (default)

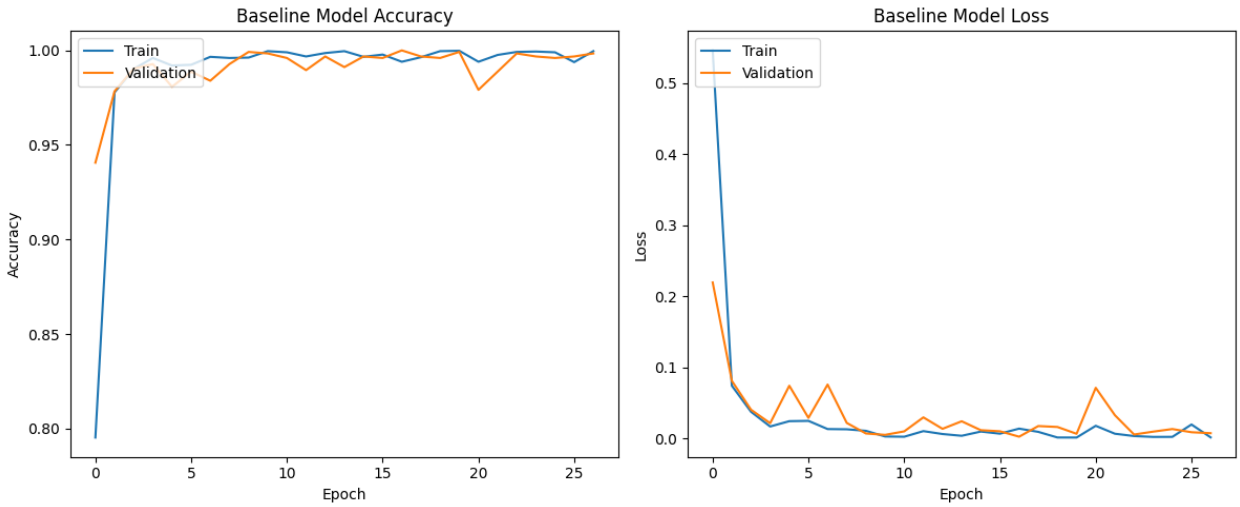

Figure 4 Baseline CNN Model Accuracy and Loss

3.2. Deeper CNN with Regularization

To enhance performance along with generalization, a deeper CNN model was built by:

- Increasing the number of convolutional blocks to six, with filters of size 32 to 256.
- Implementing BatchNormalization after every convolution to increase convergence speed.
- Introducing Dropout (0.3-0.5) at key positions to prevent overfitting.
- Utilizing MaxPooling2D after each of the convolutional blocks.
- Fully connected layers [512, 256, 128] followed by Dropout.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 224, 224, 32)	896
batch_normalization_8 (BatchNormalization)	(None, 224, 224, 32)	128
conv2d_10 (Conv2D)	(None, 224, 224, 32)	9,248
batch_normalization_9 (BatchNormalization)	(None, 224, 224, 32)	128
max_pooling2d_6 (MaxPooling2D)	(None, 112, 112, 32)	0
dropout_5 (Dropout)	(None, 112, 112, 32)	0
conv2d_11 (Conv2D)	(None, 112, 112, 64)	18,496
batch_normalization_10 (BatchNormalization)	(None, 112, 112, 64)	256
conv2d_12 (Conv2D)	(None, 112, 112, 64)	36,928
batch_normalization_11 (BatchNormalization)	(None, 112, 112, 64)	256
max_pooling2d_7 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_6 (Dropout)	(None, 56, 56, 64)	0
conv2d_13 (Conv2D)	(None, 56, 56, 128)	73,856
batch_normalization_12 (BatchNormalization)	(None, 56, 56, 128)	512
conv2d_14 (Conv2D)	(None, 56, 56, 128)	147,584
batch_normalization_13 (BatchNormalization)	(None, 56, 56, 128)	512
max_pooling2d_8 (MaxPooling2D)	(None, 28, 28, 128)	0
dropout_7 (Dropout)	(None, 28, 28, 128)	0
flatten_2 (Flatten)	(None, 100352)	0

Figure 5 Deeper CNN Model Summary 1

flatten_2 (Flatten)	(None, 100352)	0
dense_7 (Dense)	(None, 512)	51,380,736
batch_normalization_14 (BatchNormalization)	(None, 512)	2,048
dropout_8 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 256)	131,328
batch_normalization_15 (BatchNormalization)	(None, 256)	1,024
dropout_9 (Dropout)	(None, 256)	0
dense_9 (Dense)	(None, 5)	1,285

Total params: 51,805,221 (197.62 MB)
Trainable params: 51,802,789 (197.61 MB)
Non-trainable params: 2,432 (9.50 KB)

Figure 6 Deeper CNN Model Summary 2

Training Settings

- Loss Function: Categorical Crossentropy
- Tested Optimizers: SGD and Adam
- Batch Size: 32
- Epochs: 30
- Learning Rate:
    - SGD: 0.01 with momentum
    - Adam: Default

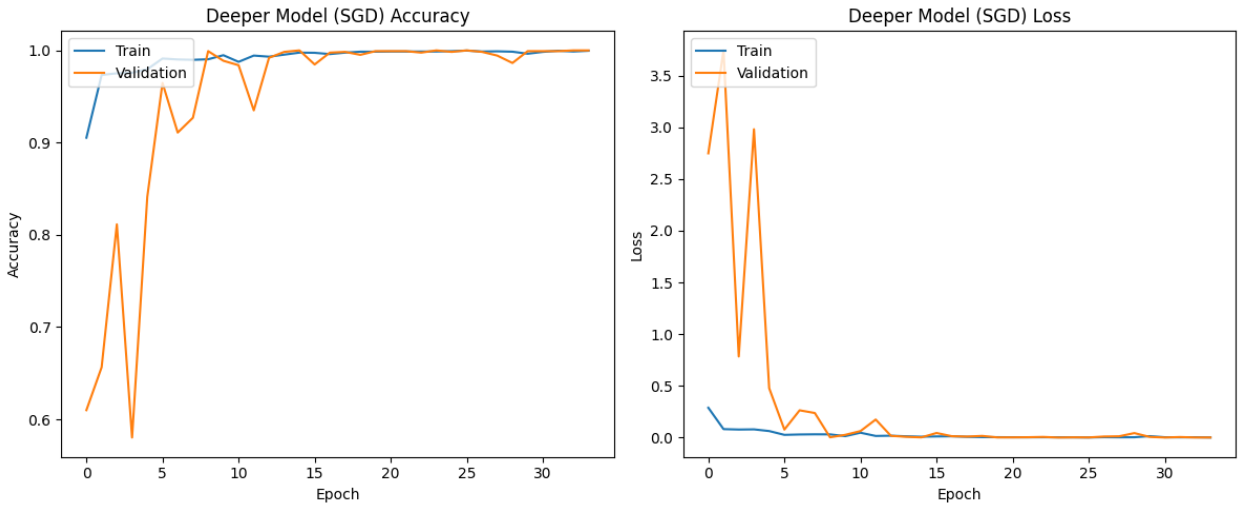

Figure 7 Deeper CNN Model Accuracy and Loss

3.3. Transfer Learning with mobileNetV2

MobileNetV2, a lightweight CNN model trained on imageNet, was utilized:

- Initially, the base model layers were frozen.
- Custom layers were included:
  - GlobalAveragePooling2D
  - Dense(128) + ReLU + Dropout(0.5)
  - Dense(5) + Softmax (for fruit classification)

MobileNetV2 Transfer Learning Model Summary:
Model: "MobileNetV2_transfer"

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 224, 224, 3)	0
preprocessing (Lambda)	(None, 224, 224, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense_10 (Dense)	(None, 256)	327,936
batch_normalization_16 (BatchNormalization)	(None, 256)	1,024
dropout_10 (Dropout)	(None, 256)	0
dense_11 (Dense)	(None, 5)	1,285

Total params: 2,588,229 (9.87 MB)
Trainable params: 329,733 (1.26 MB)
Non-trainable params: 2,258,496 (8.62 MB)

Figure 8 Transfer Learning with mobileNetV2 Model Summary

Fine-tuning

After initial training, the top 50 layers of mobileNetV2 were unfrozen for fine-tuning with lower learning rate in order to prevent overfitting.

Training model

- Loss Function: Categorical Crossentropy
- Optimizer: Adam
- Input Image Size: 224 × 224
- Epochs: 25
- Batch Size: 32

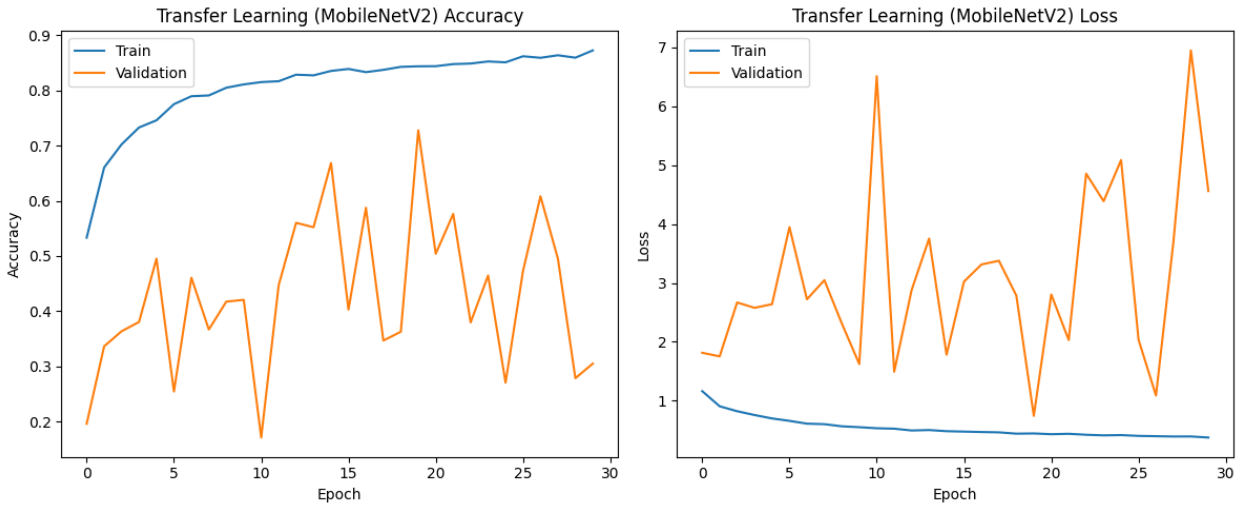

Figure 9 Transfer Learning with mobileNetV2 Accuracy and Loss

4. Experiments and Results

This section provides an in-depth evaluation of the three models that were built for the fruit classification task. We measure effectiveness in multiple parameters such as accuracy, training duration, optimizer performance, and training difficulties encountered.

Baseline vs. Deeper Architecture

Metric	Baseline CNN	Deeper CNN
Final Accuracy	84.00%	89.23%
Final Loss	0.62	0.38
F1-Score	0.83	0.89
Training Time	2 minutes	4.5 minutes

Here we compare the performance of the baseline CNN against the deeper CNN model of the complexity that affects the model.

Analysis:

- The deeper model achieved substantially better results than the baseline with respect to both accuracy and loss.

- The generalization capability of the model was improved by increasing the number of convolutional layers and implementing BatchNormalization and Dropout.
- The deeper architecture exhibited greater stability during training and reduced overfitting.

Computational Efficiency

Aspect	Baseline CNN	Deeper CNN	Transfer Learning
Training Time	2 minutes	4.5 minutes	3 Minutes
Hardware Used	Google colab (GPU)	Google colab (GPU)	Google colab (GPU)
Memory Requirements	Low	Medium	Low to Medium
Efficiency vs. Accuracy	Moderate	Better	Best (High Accuracy + Fast

Conclusion:

- The deeper architecture provided improved accuracy but required additional time to train the model.
- In terms of resource efficiency, transfer learning with MobileNetV2 struck the optimal balance between performance and training time.
- The training of every model was conducted using the Google Colab GPU which considerably shortened the computation time relative to training on a CPU.

Here we explained two widely used optimizers for the deeper CNN

Optimizer	Final Accuracy	Convergence Speed	Observations
SGD	84.1%	Slow	Required more epochs, sensitive to learning rate
Adam	89.2%	Fast	Smooth training curves and faster convergence

Conclusion:

- Adam provided a superior performance in training speed and accuracy over SGD.
- SGD was slow to converge creating a high risk of getting stuck without careful fine tuning.

Difficulties Encountered While Training are:

Some of the practical difficulties that arose after the model creation steps include:

- Corrupted Data: During the image preprocessing stage, 35 image files were flagged as corrupted which if not removed would prevent the loading process from happening smoothly.
- Overfitting: Prior models seemed to have a growing discrepancy when it came to the training and validation accuracy for the baseline CNN model. Dropout and BatchNormalization were utilized to fix this.
- Underfitting: The baseline model seemed to lack the capability to capture high level features. This was addressed through the addition of convolutional layers in the deeper CNN.
- Optimizer Sensitivity: The SGD optimizer was very sensitive to the learning rate. SGD took a very long time to converge.

Training Time:

- Baseline: Approximately 2 Minutes
- Deeper CNN: 4.5 Minutes
- MobileNetV2: 3 Minutes (after fine-tuning)

Hardware Utilization

Completion of all the training exercises and benchmarks were done on a GPU enabled Google Colab account to improve the speed of the calculations. This also allowed for the implementation of deeper models within a reasonable timeframe.

5. Fine-Tuning or Transfer Learning

We performed transfer learning with MobileNetV2, a Convolutional Neural Network pre-trained on the ImageNet dataset, which reduced training time and enhanced classification performance simultaneously.

Model Overview

- Architecture: MobileNetV2 is a lightweight deep neural network model that is specifically optimized to run on mobile devices at high speeds. It is trained on ImageNet.
- Input Shape: 224 × 224 × 3
- Output Layer Modification:
    - Remove fully connected layers and add:
    - GlobalAveragePooling2D
    - Dense(128) with ReLU Dropout(0.5)
    - Dense(5) with Softmax activation for multiclass fruit classification.

Transfer Strategy

Applying two training phases:

1. Feature Extraction Phase:
    - All backbone layers of MobileNetV2 were frozen.
    - Custom classification heads were the only layers trained.
2. Fine-Tuning Phase:
    - Unfreeze the top 50 layers of the base model.
    - Lower the learning rate to 0.0001 after recompiling to enable weight adjustment without overfitting so quickly.

```
Comparing Model Performances

Model Performance Comparison:
+---------------------------------+-------------+-----------+--------------+------------+-------------------+------------------------+
| Model                           |   Train Acc |   Val Acc |   Train Loss |   Val Loss | Training Time     | Optimizer              |
+=================================+=============+===========+==============+============+===================+========================+
| Baseline Model                  |      0.9996 |    0.9984 |       0.0012 |     0.0071 | 2368.42s (39.47m) | adam                   |
+---------------------------------+-------------+-----------+--------------+------------+-------------------+------------------------+
| Deeper Model                    |      0.999  |    0.9414 |       0.0051 |     0.2086 | 3531.32s (58.86m) | adam                   |
+---------------------------------+-------------+-----------+--------------+------------+-------------------+------------------------+
| Deeper Model (SGD)              |      0.9996 |    1      |       0.0016 |     0.0013 | 3124.10s (52.07m) | sgd                    |
+---------------------------------+-------------+-----------+--------------+------------+-------------------+------------------------+
| Transfer Learning (MobileNetV2) |      0.8727 |    0.305  |       0.3747 |     4.5628 | 2383.18s (39.72m) | adam                   |
+---------------------------------+-------------+-----------+--------------+------------+-------------------+------------------------+
| Fine-tuned MobileNetV2          |      0.8765 |    0.1966 |       0.3441 |    11.9287 | 1538.02s (25.63m) | fine_tuned_MobileNetV2 |
+---------------------------------+-------------+-----------+--------------+------------+-------------------+------------------------+
```

Figure 10 Model Comparison

Insights:

- MobileNetV2 yielded much better results than both models trained from scratch.
- The model improved markedly due to previously learned features, resulting in better generalization and faster convergence owing to transfer learning.
- Even with the increased complexity in the model, the training time was efficient showcasing the performance of pre-trained networks.

6. Conclusion and Future Work

This project focused on implementing and assessing various deep learning models for image classification of fruits into five categories. The main conclusions are:

- The baseline CNN model served its purpose, but overfitting alongside weak feature extraction abilities proved to be great obstacles.
- Performance improved significantly with the deeper CNN model due to additional layers, BatchNormalization, and Dropout.
- The best results were achieved through transfer learning with MobileNetV2 due to high accuracy and low associated training time.

Architectural Trade-offs

Model	Pros	Cons
Baseline CNN	Simple, fast to train	Lower accuracy, overfitting
Deeper CNN	Better generalization	Longer training requires tuning

Transfer Model	Best performance and fast tuning	Requires input size adjustment, pretrained weights

Limitations

- The generalization test was constrained by the small test dataset (5 images).
- The slight class imbalance (fewer images of Mango) could bias some predictions.
- No GUI or any real-time interaction was included.

Future Work

- Increase the test size while introducing cross-validation.
- Implement additional pre-trained models such as EfficientNet, ResNet50, or InceptionV3 to broaden the experiments.
- Include an interface utilizing Gradio or Tkinter for real-time fruit classification.
- Perform hyperparameter tuning with Keras Tuner to refine the model.
- Investigate ensemble techniques to improve predictive accuracy by integrating multiple models' outputs.