AgriVision: Automated Multi-Class Fruit Classification

---

1. Introduction & Objective

Introduction: The agricultural industry relies heavily on manual fruit sorting and quality inspection, which is labor-intensive and prone to errors. The goal of this project is to develop a computer vision system capable of classifying 10 types of fruits automatically, improving efficiency and reducing waste in food supply chains.

Objective: - Automate fruit classification using deep learning. - Evaluate model performance using accuracy, precision, recall, and visualizations. - Provide a deployable solution for real-time inference.

---

2. Methodology

Dataset: - Source: Roboflow Fruit Dataset - Classes: Apple, Avocado, Banana, Cherry, Kiwi, Mango, Orange, Pineapple, Strawberries, Watermelon. - Data split: Train / Validation / Test

Preprocessing & Data Augmentation:

- Resize images to 224x224 pixels - Normalize pixel values to [0,1] - Data augmentation on training set: - Rotation: ±20° - Zoom: 0.2 - Width/Height shift: 0.1 - Horizontal flip

Model Architecture:

MobileNetV2 was chosen because it is a lightweight and efficient model that trains quickly, uses low memory, and works well on limited resources such as Google Colab, making it ideal for small agricultural datasets.

Despite its compact size, it provides strong accuracy through transfer learning by capturing both low-level textures and high-level features like fruit shape and color, while reducing overfitting on small datasets. Additionally, its fast inference speed makes it suitable for real-time deployment in an automated AgriVision fruit classification system.

- Base model: MobileNetV2 (pre-trained on ImageNet) - Frozen base layers + custom classification head: - GlobalAveragePooling2D - Dense(128) + ReLU - Dropout(0.4) - Dense(10) + Softmax - Optimizer: Adam - Loss: Categorical Crossentropy - Metrics: Accuracy

Training Strategy: -

Trained for 10 epochs with EarlyStopping to prevent overfitting - Optional fine-tuning on top 20 layers using a very low learning rate (1e-5)

---

3. Results

3.1 Test Accuracy: **86.27%**

3.2 Classification Report (Precision, Recall, F1-Score)

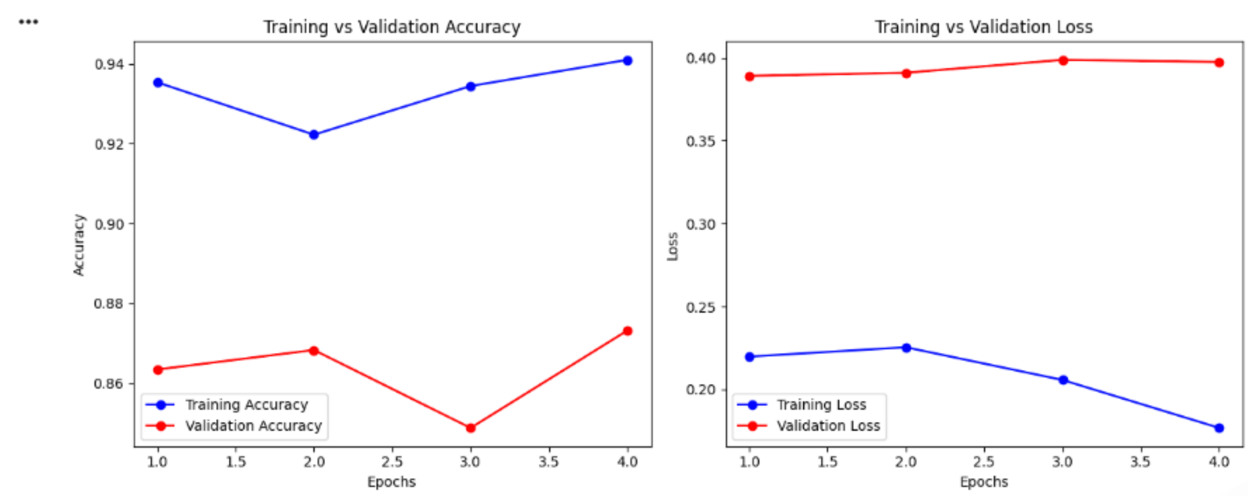
```
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 124ms/step
✅Test Accuracy: 86.27%
✅Classification Report:
              precision    recall  f1-score   support

       apple       0.91      0.91      0.91        11
     avocado       0.86      1.00      0.92        12
      banana       1.00      0.75      0.86        12
      cherry       1.00      0.83      0.91         6
        kiwi       0.78      0.88      0.82         8
       mango       0.73      0.89      0.80         9
      orange       0.89      0.73      0.80        11
  pinenapple       0.77      1.00      0.87        10
strawberries       0.88      0.88      0.88         8
  watermelon       0.92      0.80      0.86        15

    accuracy                           0.86       102
   macro avg       0.87      0.87      0.86       102
weighted avg       0.88      0.86      0.86       102
```

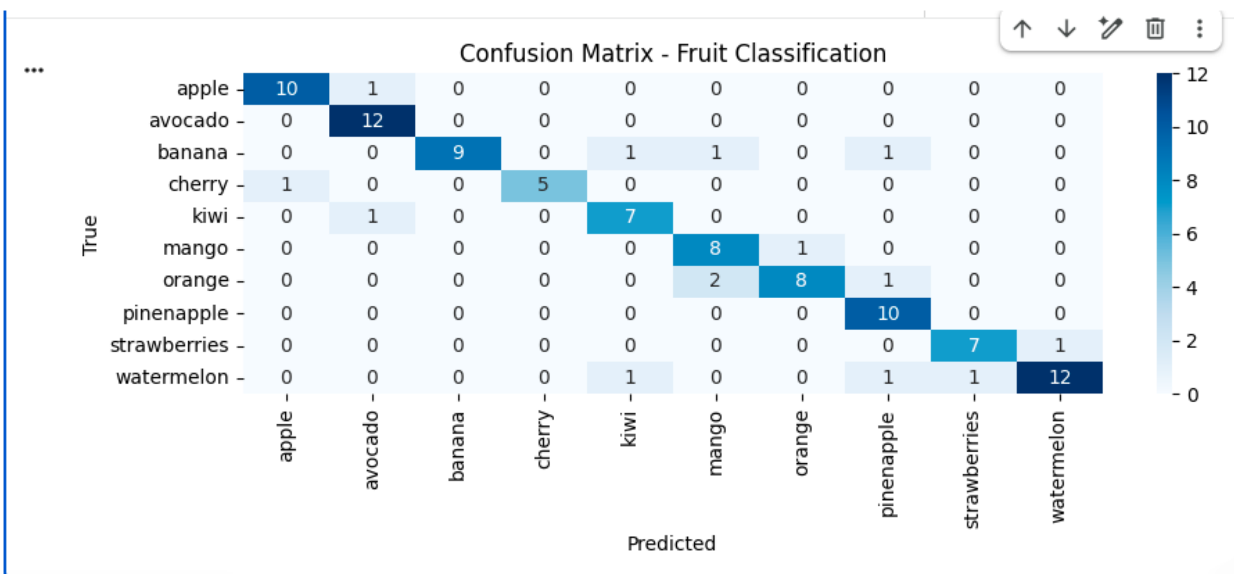
3.3 Training Curves:

screenshots of training vs validation accuracy and loss over epochs here.



3.4 Confusion Matrix:

heatmap screenshot of the confusion matrix here.



3.5 Error Analysis

5 misclassified images - Observations: Confusions due to similar shapes/colors (e.g., Apple vs Cherry)

```
Example comments for the 5 images:
```

1. green Apple → Predicted as avocado

```
Confusion due to similar color and small size

2. banana → Predicted as pipeapple
Confused due to hanging style

3. Banana → Predicted as Mango
Confused due to pattern on the fruit surface

4. Banana → Predicted as kiwi
Confused due to inner shape

5. Apple → Predicted as Cherry
Confusion because of similar round shape and color tone
```

---

4. Conclusion

What Worked: - MobileNetV2 provided good accuracy with fast training. - Data augmentation improved generalization. - Transfer learning allowed the model to perform well even on a small dataset.

What Didn't Work / Limitations:- Confusion between fruits with similar shapes/colors. - Model can be further improved using fine-tuning + Keras Tuner hyperparameter optimization.