**Mangonet: A Vgg16-Based Neural Network For Mango Classification**

**Final Project Report**

|  |  |
| --- | --- |
| Date | 5 July 2025 |
| Team ID | SWTID1750180871 |
| Project Title | Mangonet: A Vgg16-Based Neural Network For Mango Classification |
| Maximum Marks | 10 Marks |

**Team members:**

Samruddhi Ramdhave

Anirudh CV

Narendra V B

Pragya Arya

**INDEX**

1. Introduction
   1. Project overviews
   2. Objectives
2. Project Initialization and Planning Phase
   1. Define Problem Statement
   2. Project Proposal (Proposed Solution)
   3. Initial Project Planning
3. Data Collection and Preprocessing Phase
   1. Data Collection Plan and Raw Data Sources Identified
   2. Data Quality Report
   3. Data Preprocessing
4. Model Development Phase
   1. Model Selection Report
   2. Initial Model Training Code, Model Validation and Evaluation Report
5. Model Optimization and Tuning Phase
   1. Tuning Documentation
   2. Final Model Selection Justification
6. Results
   1. Output Screenshots
7. Advantages & Disadvantages
8. Conclusion
9. Future Scope
10. Appendix
    1. Source Code
    2. GitHub & Project Demo Link

**MangoNet: A VGG16-Based Neural Network for Mango Classification**

**1. Introduction**

**1.1 Project Overview**

MangoNet is a deep learning-based image classification system designed to automate the

identification and grading of mango varieties using visual data. Built on the VGG16

architecture, MangoNet leverages transfer learning to classify mangoes by type and quality.

The project is intended for use in agricultural sorting, fruit market grading, and quality

control in processing facilities, aiming to streamline sorting, ensure quality, and boost

productivity.

**1.2 Objectives**

- Understand and apply fundamental concepts of Convolutional Neural Networks (CNNs).

- Gain practical experience in image data handling and preprocessing.

- Build, train, and evaluate a deep learning model for mango classification.

- Develop a user-friendly web application for image-based mango classification using Flask.

- Demonstrate the application of transfer learning for efficient and accurate fruit

classification.

**2. Project Initialization and Planning Phase**

**2.1 Define Problem Statement**

Manual mango sorting is labor-intensive, subjective, and prone to errors, especially at scale.

There is a need for an automated, reliable, and scalable system to classify and grade mangoes

based on visual characteristics such as variety, ripeness, and defects.

**2.2 Project Proposal (Proposed Solution)**

The proposed solution is a VGG16-based CNN model fine-tuned via transfer learning,

capable of classifying mango images into predefined categories. The model is integrated into

a Flask web application, allowing users to upload images and receive instant classification

results.

**2.3 Initial Project Planning**

- Data acquisition: Identify and download a comprehensive mango image dataset.

- Preprocessing: Standardize and augment images to improve model robustness.

- Model development: Implement transfer learning with VGG16.

- Evaluation: Use accuracy and loss metrics, along with confusion matrices, for validation.

- Deployment: Build a web interface for end-user interaction.

**3. Data Collection and Preprocessing Phase**

**3.1 Data Collection Plan and Raw Data Sources Identified**

-Primary dataset: Mango Varieties Classification and Grading Dataset from Kaggle,

containing 3,200+ images across 8 mango classes.

- Directory structure: Images organized by class folders for compatibility with Keras’

`ImageDataGenerator`.

**3.2 Data Quality Report**

- Class balance: 1,280 training, 320 validation, and 1,600 test images, evenly distributed

across 8 classes.

- Image quality: Images checked for clarity, correct labeling, and absence of duplicates or

corrupt files.

**3.3 Data Preprocessing**

- Resizing: All images resized to 224x224 pixels to match VGG16 input requirements.

- Rescaling:Pixel values normalized to [0,1] using `rescale=1./255`.

- Augmentation: Real-time augmentation (rotation, zoom, shift, flip) applied to training data

to reduce overfitting.

- Splitting: Data split into training, validation, and test sets using Keras’

`ImageDataGenerator` with `validation\_split`.

**4. Model Development Phase**

**4.1 Model Selection Report**

- Model: VGG16 pre-trained on ImageNet, chosen for its proven feature extraction

capabilities in image classification tasks.

- Transfer learning: Base layers frozen; custom dense, dropout, and softmax layers added for

mango classification.

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Output Shape | Parameters | Notes |
| VGG16 | ((None, 7, 7, 512) | 14,714,688 | Pre-trained, frozen |
| Flatten | (None, 25088) | 0 |  |
| Dense (256) | (None, 256) | 6,422,784 | ReLU activation |
| Dropout (0.5) | (None, 256) | 0 | Regularization |
| Dense (8) | (None, 8) | 2,056 | Softmax activation |

**4.2 Initial Model Training Code, Model Validation and Evaluation Report**

-Optimizer: Adam

-Loss function: Categorical cross-entropy

-Metrics:Accuracy

-Epochs: 10

- Training results:

- Final training accuracy: ~82.7%

- Final validation accuracy: ~80.6%

- Final training loss: 0.46

- Final validation loss: 0.63

- Evaluation:

- Validation set: Loss ≈ 0.548, Accuracy ≈ 0.84

- Test set: Loss ≈ 0.159, Accuracy ≈ 0.967

- Interpretation:High test accuracy and low loss indicate strong generalization and minimal

overfitting.

**5. Model Optimization and Tuning Phase**

**5.1 Tuning Documentation**

- Hyperparameters tuned: Learning rate, batch size, number of epochs.

- Regularization: Dropout layer (rate 0.5) added to reduce overfitting.

- Augmentation:Extensive real-time image augmentation applied to increase data diversity.

**5.2 Final Model Selection Justification**

The final model was selected based on its superior validation and test accuracy, low loss, and

stability across epochs. The use of transfer learning with VGG16 enabled high accuracy with

limited training data and reduced training time.

**6. Results**

**6.1 Output Screenshots**

- Training/Validation Curves:

Plots show steadily increasing accuracy and decreasing loss for both training and

validation sets.

A graph with blue and orange lines

AI-generated content may be incorrect.

A graph with blue and orange lines

AI-generated content may be incorrect.

- Prediction Example:

Uploaded mango image displayed with predicted class label.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A close up of a fruit

AI-generated content may be incorrect.

- Web App UI:

Home, upload, and results pages allow user interaction and display predictions.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**7. Advantages & Disadvantages**

**Advantages:**

- High classification accuracy (>96% on test set).

- Reduces manual labor and subjectivity in mango sorting.

- Scalable and adaptable to other fruit classification tasks.

- Real-time image augmentation and transfer learning improve robustness.

**Disadvantages:**

- Model performance depends on quality and diversity of training data.

- Requires GPU resources for efficient training.

- May misclassify images with severe occlusion or poor lighting.

**8. Conclusion**

MangoNet demonstrates that transfer learning with VGG16 is highly effective for automated

mango classification and grading. The project achieves high accuracy, robust performance,

and practical utility for agricultural and commercial applications. Integrating the model into a

web application further enhances accessibility and usability for end-users.

**9. Future Scope**

- Dataset expansion: Incorporate more mango varieties and images from diverse conditions to

further improve generalization.

- Advanced imaging: Explore multispectral/hyperspectral data for enhanced defect and

ripeness detection.

- Edge deployment: Optimize for real-time classification on embedded systems for on-site

agricultural use.

- Model enhancements: Investigate newer architectures (e.g., EfficientNet, attention

mechanisms) for further accuracy gains.

**10. Appendix**

**10.1 Source Code**

- Model training and evaluation:See `mangonet.ipynb` for full code, including data

preprocessing, model architecture, training, and evaluation steps[1].

- Flask application:`app.py` handles image uploads, preprocessing, prediction, and result

rendering.

- HTML templates: `index.html`, `results.html`, and `home.html` provide the web interface.

**10.2 GitHub & Project Demo Link**

- GitHub Repository:

https://github.com/samruddhisr4/MangoNet/tree/main

-Project Demo Link:

https://youtu.be/WRZFV1wRVAk