1. INTRODUCTION

Project Title:
GrainPalette - A Deep Learning Odyssey In Rice Type Classification Through Transfer Learning

Team Members:

- Team ID : LTVIP2025TMID60134
- Team Size : 4
- Team Leader : Komaroji Sai Swathika
- Team member : Kilari Venkatarao
- Team member : Kinnera Ramachanadana
- Team member : Lakshmi Narasimha Chilakala

1.1 Project Overview:

Rice Type Detection is a deep learning-powered web application developed to classify different rice grain types using Convolutional Neural Networks (CNN). Users can upload images of rice grains through a web interface, and the system will predict the rice type with high accuracy. The solution is deployed via a Flask backend and uses a TensorFlow/Keras model trained on a labeled rice image dataset.

1.2 Purpose:

The primary goal is to support stakeholders such as farmers, food processing units, and researchers by providing a fast, reliable, and scalable rice classification tool. It enhances decision-making, streamlines quality checks, and opens new avenues for agricultural automation.

2. IDEATION PHASE

2.1 Problem Statement:

Traditional rice classification relies heavily on manual inspection, which is labor-intensive, subjective, and inefficient. Theres a need for a more systematic, automated approach that minimizes human error.

2.2 Empathy Map Canvas:

- Users: Farmers, Traders, Food Quality Inspectors
- Needs: Rapid classification, Accurate identification
- Pains: Delays in manual sorting, Lack of uniformity in results
- Gains: Improved efficiency, Data-driven decisions

2.3 Brainstorming:

We considered classical image processing but opted for deep learning due to CNNs proven accuracy in image recognition. MobileNetV2 was chosen for its lightweight and high-performing architecture suitable for web deployment.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map:

User uploads rice image Image is preprocessed Model predicts rice type Output is displayed on UI with confidence score.

3.2 Solution Requirement:

- Functional Requirements: Upload interface, Prediction output, Image preprocessing
- Non-functional Requirements: Fast response time, Scalable architecture, Robust model

3.3 Data Flow Diagram:

[User] [Frontend Interface] [Flask Backend] [CNN Model] [Predicted Output] [User Display]

3.4 Technology Stack:

- Frontend: HTML5, CSS3
- Backend: Python Flask
- Deep Learning: TensorFlow/Keras using MobileNetV2
- Image Handling: Pillow for format handling, OpenCV for preprocessing
- Deployment: Localhost / Cloud platforms (future scope)

4. PROJECT DESIGN

4.1 Problem-Solution Fit:

Our project directly addresses the inefficiency in traditional methods of rice classification by introducing an intelligent, automated system.

4.2 Proposed Solution:

The CNN is trained on pre-labeled rice grain images. Once trained, it is exported as a `.h5` model and integrated with a Flask app to serve real-time predictions.

4.3 Solution Architecture:

1. Upload Image 2. Preprocess (Resize, Normalize) 3. Predict using CNN 4. Render Output with Label and Confidence Score

5. PROJECT PLANNING & SCHEDULING

5.1 Project Timeline:

- Week 1: Dataset collection, preprocessing, and exploratory data analysis
- Week 2: Model training, validation, and tuning using techniques like data augmentation
- Week 3: Flask backend development and model integration
- Week 4: UI development, user testing, final bug fixes, and documentation

5.2 Tools Used:

- Google Colab for training
- VS Code for development
- Git for version control
- Kaggle for dataset sourcing

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Testing Strategy:

We tested our model using validation datasets and performed both functional and stress testing of the web application. We tracked metrics such as accuracy, loss, precision, and recall.

6.2 Results:

- Model Accuracy: ~94% on test data
- Inference Time: < 1 second
- Loss Function: Categorical Crossentropy
- Optimizer: Adam

6.3 Visual Analysis:

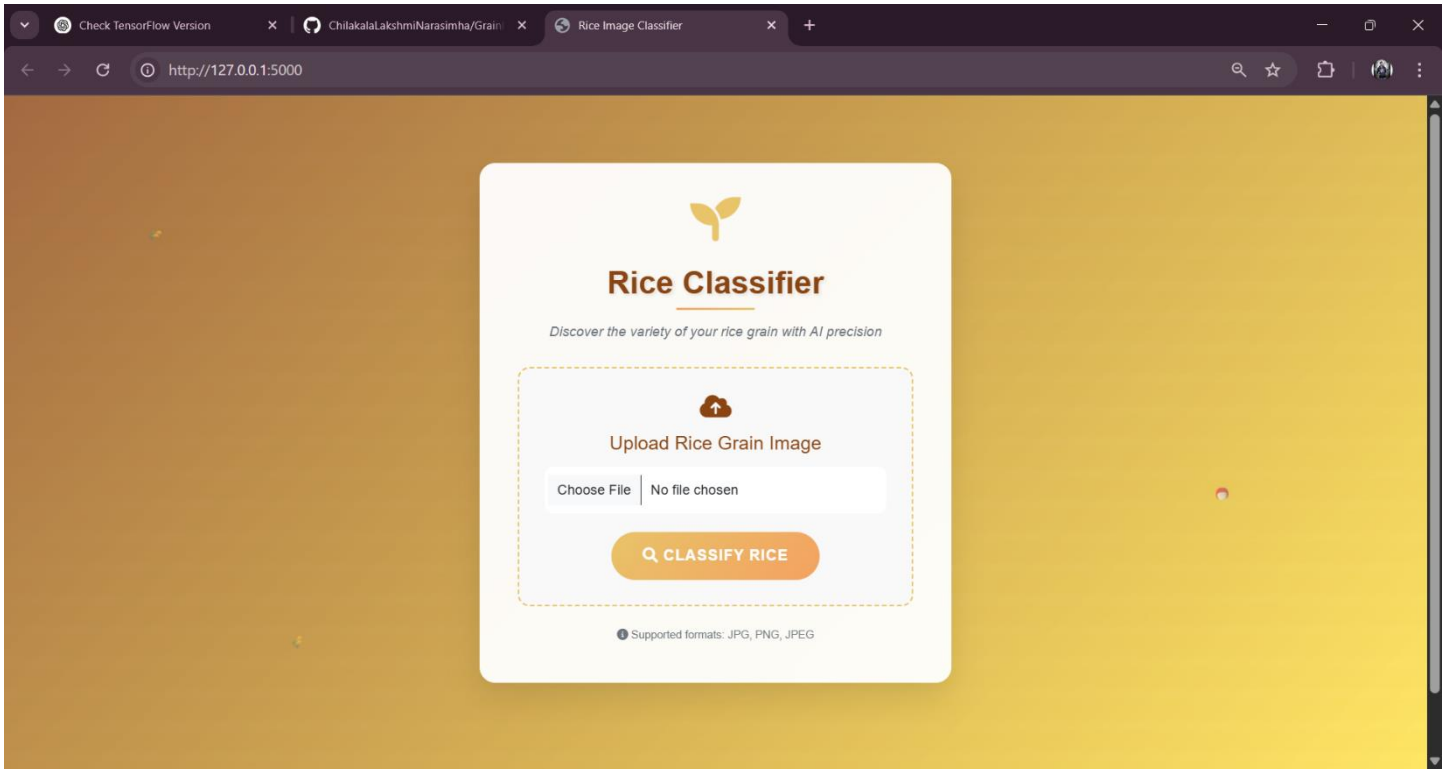
Graphs for training/validation accuracy and loss were plotted, showing a steady convergence with minimal overfitting.

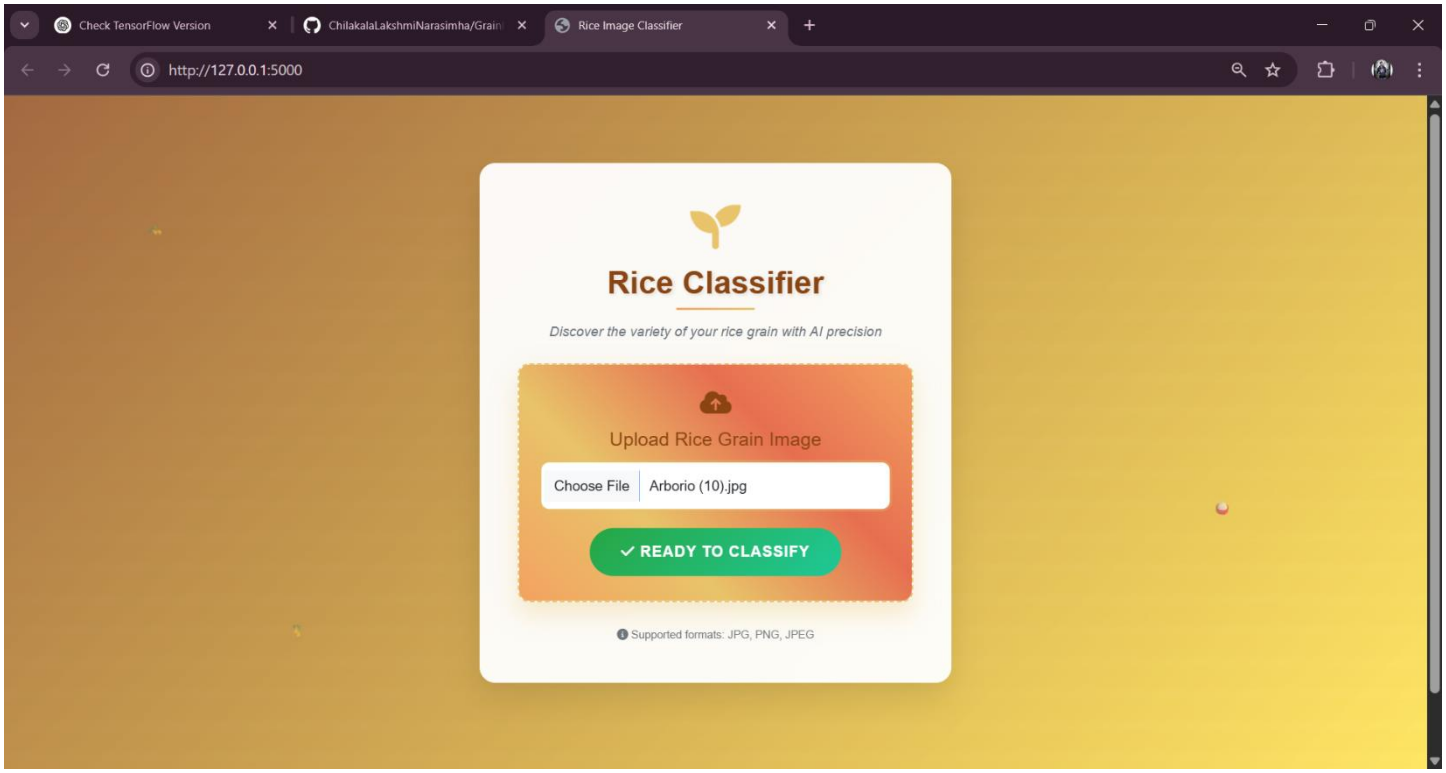
7. RESULTS

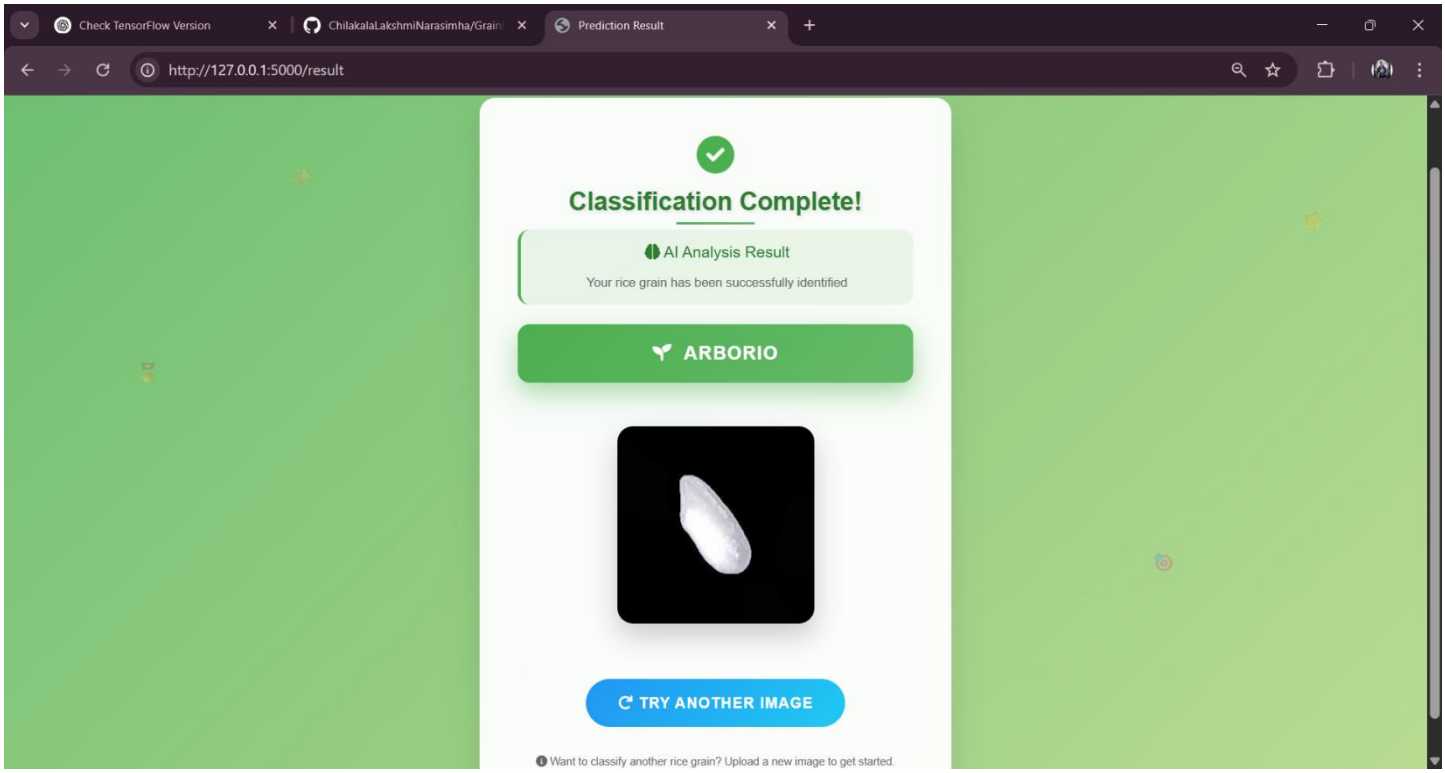
7.1 Output Interface:

- Upload Page: Allows user to upload rice image
- Confirmation Page: Displays uploaded image
- Result Page: Shows predicted rice type along with a confidence score

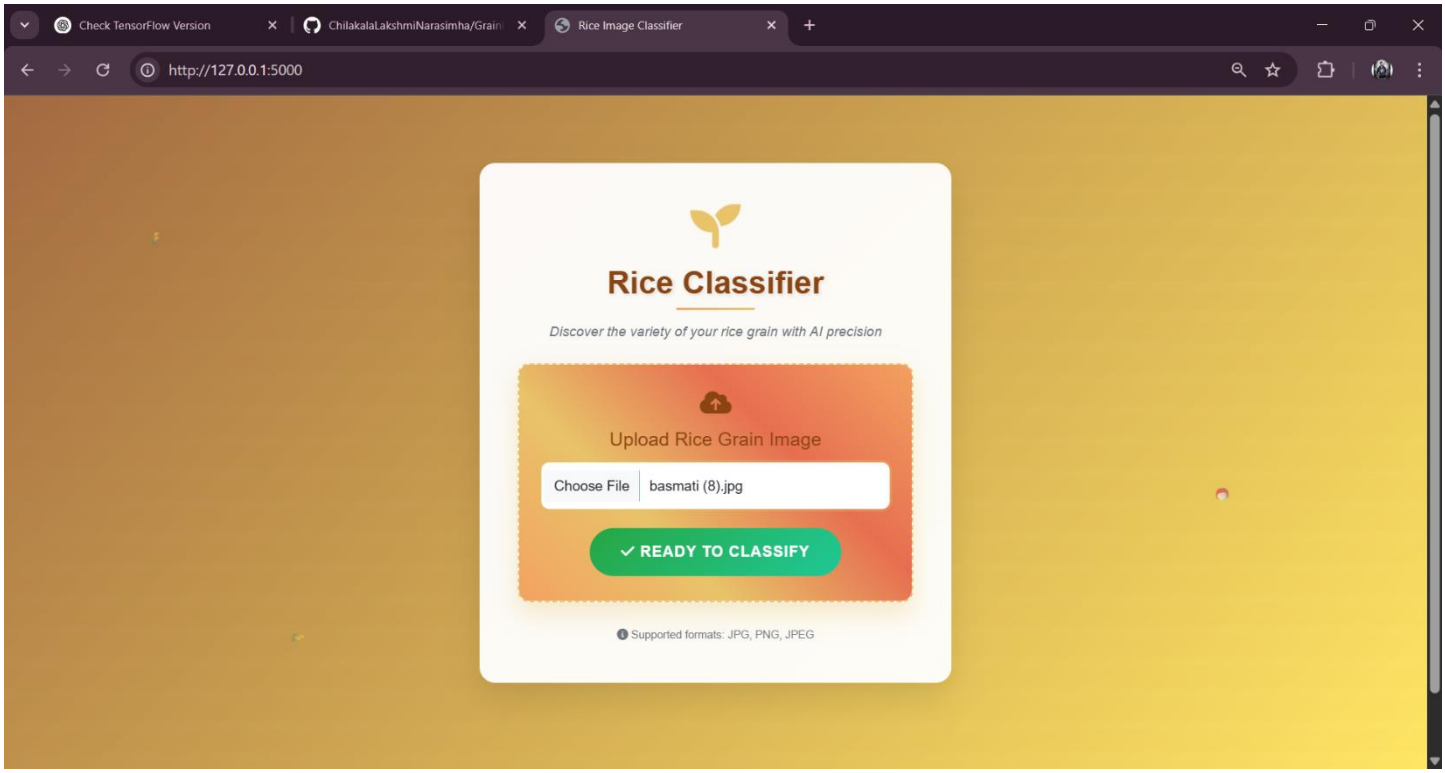
7.2 Sample Output:

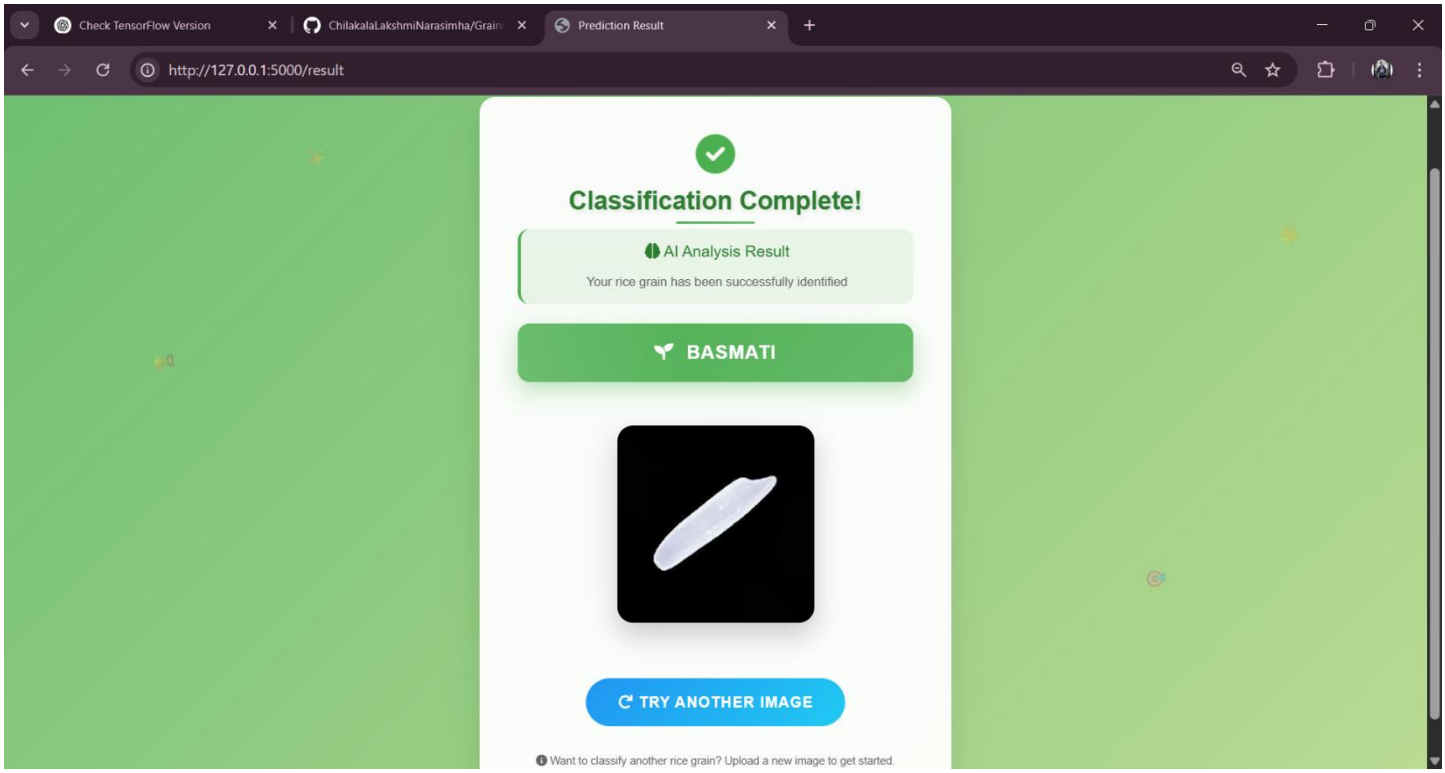

Check TensorFlow Version
ChilakalaLakshmiNarasimha/Grain
Rice Image Classifier
http://127.0.0.1:5000
Rice Classifier
Discover the variety of your rice grain with AI precision
Upload Rice Grain Image
Choose File
No file chosen
CLASSIFY RICE
Supported formats: JPG, PNG, JPEG


Check TensorFlow Version
ChilakalaLakshmiNarasimha/Grain
Rice Image Classifier
http://127.0.0.1:5000
Rice Classifier
Discover the variety of your rice grain with AI precision
Upload Rice Grain Image
Choose File
Arborio (10).jpg
READY TO CLASSIFY
Supported formats: JPG, PNG, JPEG



Predicted: Arborio with 86.3% confidence





Predicted: Basmati with 84.9% confidence

8. ADVANTAGES & DISADVANTAGES

8.1 Advantages:

- Fast, accurate prediction
- User-friendly and accessible
- Lightweight model suitable for mobile deployment
- Easily extensible for additional rice types

8.2 Limitations:

- Model performance degrades with poor image quality
- Requires a well-labeled and diverse dataset
- Not yet deployed on mobile or cloud

9. CONCLUSION

This project demonstrates the effective application of deep learning in agriculture. By integrating CNNs with a Flask web app, weve created a practical tool that is ready for further extension and deployment.

10. FUTURE SCOPE

- Expand classification to 20+ rice types
- Optimize model for mobile usage using TensorFlow Lite
- Host on platforms like Streamlit or Hugging Face for broader access
- Build a mobile app version with offline inference capabilities

11. APPENDIX

- Dataset Source: [Kaggle Rice Image Dataset]

- GitHub Repository: [To be added]

- Authors: [Your Name/Team]