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

**Team ID : LTVIP2025TMID60400**

* Team Size: 4 members
* **Team Leader :** Menda Tejasri
* **Team member :** Maruprolu Nithin Srihari Reddy
* **Team member :** Pilli Syam Prakash
* **Team member :** Shaik Inthyaz

**Table of Contents**

1. Project Overview
2. System Architecture
3. Technology Stack
4. Project Structure
5. Implementation Details
6. Development Workflow
7. Setup and Installation
8. Features and Functionality
9. API Documentation
10. Screenshots and Results
11. Challenges and Solutions
12. Future Enhancements
13. Conclusion

**1. Project Overview**

**GrainPalette is an AI-powered web application designed to classify rice grains based on their visual characteristics. Users can upload an image of a rice grain, and the app will return the most probable type of rice using a deep learning model. This tool is particularly beneficial for farmers, agricultural researchers, exporters, and traders who need to identify rice varieties efficiently.**

**The core objective is to make rice type identification quick, accurate, and accessible using computer vision and transfer learning techniques.**

**2. System Architecture**

**The system follows a modular architecture with three main layers:**

**a. Frontend Layer**

* **HTML/CSS templates to enable user interaction.**
* **File input to upload rice grain images.**

**b. Backend Layer (Flask)**

* **Handles routing, image uploads, and rendering templates.**
* **Passes image data to the model for prediction.**

**c. Model Layer**

* **A Convolutional Neural Network (CNN) trained using transfer learning (e.g., MobileNet).**
* **Outputs the predicted rice variety based on image analysis.**

**Workflow:**

1. **User uploads an image via the frontend.**
2. **Flask receives the request and saves the image.**
3. **Image is preprocessed and fed into the model.**
4. **Model returns prediction.**
5. **Flask renders the result on the web page.**

**3. Technology Stack**

* **Language: Python**
* **Framework: Flask**
* **Deep Learning: TensorFlow, Keras**
* **Frontend: HTML5, CSS3**
* **Libraries:**
  + **numpy (numerical operations)**
  + **Pillow (image processing)**
  + **tensorflow.keras (model training and inference)**

**4. Project Structure**

**rice-classifier-app/**

**├── app.py # Main Flask app**

**├── requirements.txt # Dependencies**

**├── rice\_model.h5 # Trained CNN model**

**├── static/**

**│ └── style.css # Webpage styles**

**├── templates/**

**│ ├── index.html # Upload form**

**│ └── result.html # Prediction display**

**5. Implementation Details**

* **The model is trained on a dataset of labeled rice grain images.**
* **Images are resized to 224x224 and normalized.**
* **A MobileNet-based CNN is used for better generalization and fast predictions.**
* **The model is saved as rice\_model.h5 after training.**
* **Flask loads the model at runtime and performs inference on uploaded images.**

**6. Development Workflow**

1. **Data collection and preprocessing.**
2. **Model design and training using Keras in a Jupyter notebook.**
3. **Export trained model.**
4. **Build Flask app (app.py) to serve the model.**
5. **Create HTML templates for the frontend.**
6. **Link form submission to prediction logic.**
7. **Test locally and refine the user experience.**

**7. Setup and Installation**

**Requirements:**

* **Python 3.8+**
* **pip**

**Installation Steps:**

**# Step 1: Navigate to project directory**

**cd rice-classifier-app**

**# Step 2: Install dependencies**

**pip install -r requirements.txt**

**# Step 3: Run the application**

**python app.py**

**Access:**

**Open your browser and go to: http://127.0.0.1:5000/**

**8. Features and Functionality**

* **Clean and intuitive user interface.**
* **Upload rice grain images in .jpg or .png format.**
* **Real-time predictions using a trained deep learning model.**
* **Display of predicted rice type along with uploaded image.**
* **Error handling for invalid uploads or unsupported files.**

**9. API Documentation**

**GET /**

* **Loads the homepage.**
* **Returns the upload form.**

**POST /predict**

* **Accepts an image file.**
* **Preprocesses the image and predicts rice type.**
* **Returns the result page with the prediction and image.**

**10. Screenshots and Results**

**Home Page:**

* **File input and upload button.**

**Result Page:**

* **Displays uploaded image.**
* **Shows predicted rice type (e.g., "Jasmine").**

**Model Accuracy:**

* **Validation accuracy: ~95% (can be updated based on final model).**

**11. Challenges and Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| **Similar appearance across classes** | **Used transfer learning for robust feature learning** |
| **Large model size** | **Optimized with MobileNet to reduce size** |
| **UI responsiveness** | **Applied lightweight custom CSS** |
| **Handling unsupported file formats** | **Added file validation in Flask** |

**12. Future Enhancements**

* **Deploy on cloud (Render, AWS, or Heroku).**
* **Add mobile responsiveness to UI.**
* **Use cloud storage for uploaded images.**
* **Implement top-3 prediction output.**
* **Add multilingual support.**
* **Train on larger and more diverse rice datasets.**

**13. Conclusion**

**GrainPalette successfully demonstrates how deep learning and web development can solve real-world problems in agriculture. By allowing users to identify rice types with a simple image upload, the system reduces manual effort and improves decision-making. This project sets the stage for further innovations in AI-based crop classification and can easily be extended to support more grains, seeds, or plant-based datasets in the future.**

**Github link:**

[**https://github.com/238x1a1216-debug/-GrainPalette---A-Deep-Learning-Odyssey-In-Rice-Type-Classification-Through-Transfer-Learning**](https://github.com/238x1a1216-debug/-GrainPalette---A-Deep-Learning-Odyssey-In-Rice-Type-Classification-Through-Transfer-Learning)

**Project code link:https://colab.research.google.com/drive/1qBKVc7BJQ0SpZCgKNR4u4q9syof-GPfp?usp=drive\_link**