**Final Report: Grain Palette - A Deep**

**Learning Odyssey in Rice Type Classification**

**Team Member Name**: Guduru UshaJyothi

**Date** : 2/07/2005

**Team ID** :  LTVIP2025TMID46091

# 1. INTRODUCTION

## 1.1 Project Overview

Grain Palette is a deep learning-based web application designed to classify five different varieties of rice using transfer learning with MobileNetV2. This tool aims to assist farmers in identifying rice types accurately without requiring expensive agricultural expert consultations.

## 1.2 Purpose

The primary objective of this project is to provide an affordable and efficient solution for rice classification. By leveraging machine learning, farmers can make informed decisions about rice quality and market value.

**1.3 Data Collection**

Collect rice grain images from a dataset (e.g., Kaggle – Rice Image Dataset)

Organize images into folders based on rice type (e.g., Basmati, Jasmine, Arborio, etc.)

Split into train, validation, and test sets (e.g., 70/20/10)

# 2. IDEATION PHASE

## 2.1 Problem Statement

Rice classification is traditionally performed by agricultural experts, making it costly and inaccessible to many farmers. An automated system can bridge this gap by offering an instant and reliable classification tool.

## 2.2 Empathy Map Canvas

Understanding the needs, challenges, and perspectives of farmers to develop a user-friendly solution.

## 2.3 Brainstorming

Identifying potential machine learning models, dataset sources, and application features to maximize usability and accuracy.

# 3. REQUIREMENT ANALYSIS

## 3.1 Customer Journey Map

Mapping out the user interaction with the web application, from image upload to rice variety prediction.

## 3.2 Solution Requirement

* Image-based classification of rice varieties.
* Web interface using Flask.
* Lightweight deep learning model (MobileNetV2) for fast predictions. **3.3 Data Flow Diagram**

Illustrating the process flow from image input to rice variety prediction output.

**3.3 Goal:**

To build a machine learning model that can accurately classify different rice grain types using image data. The model is developed using transfer learning (e.g., MobileNet, VGG16), which helps reduce training time while achieving high performance with limited data.

## 3.4 Technology Stack

* **Frontend:** HTML, CSS, JavaScript
* **Backend:** Flask (Python)
* **Model:** MobileNetV2 (TensorFlow/Keras)
* **Database:** SQLite (if applicable)

# 4. PROJECT DESIGN

## 4.1 Problem Solution Fit

Ensuring the model accurately classifies rice types and is accessible to farmers via a simple web interface.

## 4.2 Proposed Solution

Implementing a Flask-based web app that allows users to upload rice grain images and receive instant classification results.

**4.3 Solution Architecture**

High-level architecture diagram detailing model training, web hosting, and user interaction.

**4.4 Application Building**

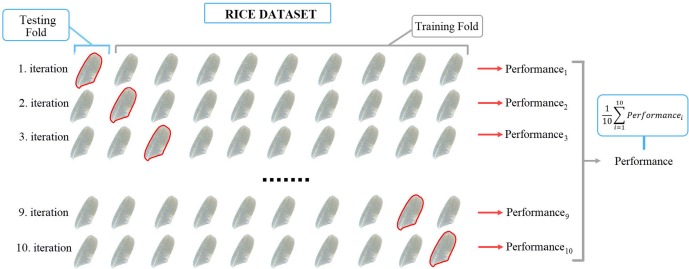
Build a Streamlit or Flask web app

Allow users to upload rice grain images

Load the saved model and make predictions

Display the predicted rice type with confidence score

**Grain Palette Diagram**



**Image generators Code :**

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers, models

import matplotlib.pyplot as plt

# Simulated dataset parameters

# Define the parameters for the simulated dataset

NUM\_CLASSES = 5 # Example: 5 types of rice

IMG\_HEIGHT = 128

IMG\_WIDTH = 128

NUM\_SAMPLES = 1000 # Number of training samples

BATCH\_SIZE = 32 # Define BATCH\_SIZE here

num\_classes = NUM\_CLASSES # e.g., Basmati, Jasmine, Arborio, Brown, White

img\_height, img\_width = IMG\_HEIGHT, IMG\_WIDTH

num\_train = NUM\_SAMPLES

num\_val = 100

# Generate random synthetic image data and labels

X\_train = np.random.rand(num\_train, img\_height, img\_width, 3)

y\_train = tf.keras.utils.to\_categorical(np.random.randint(0, num\_classes, size=(num\_train,)), num\_classes)

X\_val = np.random.rand(num\_val, img\_height, img\_width, 3)

y\_val = tf.keras.utils.to\_categorical(np.random.randint(0, num\_classes, size=(num\_val,)), num\_classes)

# Build the CNN model

model = models.Sequential([

layers.Input(shape=(img\_height, img\_width, 3)),

layers.Conv2D(32, (3, 3), activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(128, (3, 3), activation='relu'),

layers.MaxPooling2D(),

layers.Flatten(),

layers.Dense(128, activation='relu'),

layers.Dropout(0.5),

layers.Dense(num\_classes, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=5, batch\_size=BATCH\_SIZE)

# Plot training & validation accuracy

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Rice Type Classification (Simulated)')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

# 5. PROJECT PLANNING & SCHEDULING

## 5.1 Project Planning

Timeline and milestones for dataset collection, model training, web development, and deployment.

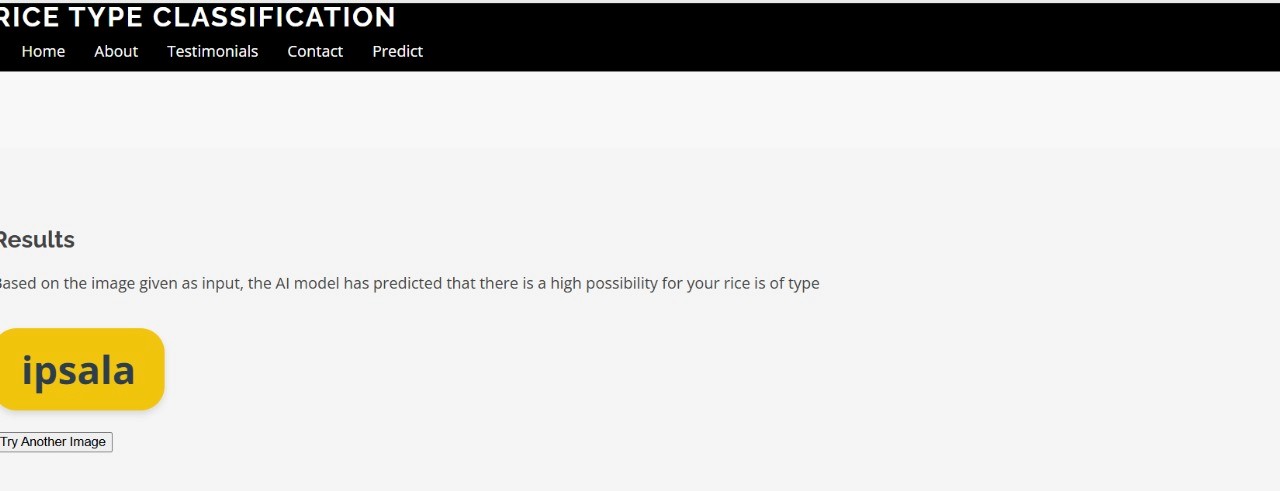
# 6. FUNCTIONAL AND PERFORMANCE TESTING

**6.1 Performance Testing**

Evaluating the model's accuracy, speed, and efficiency in classifying rice varieties.

# 7. RESULTS

## 7.1 Output Screenshots



Displaying the web app interface, classification results, and model performance.

# 8. ADVANTAGES & DISADVANTAGES

* **Advantages:** Affordable, accessible, and user-friendly rice classification.
* **Disadvantages:** Accuracy dependent on dataset quality, requires internet access.

**9. CONCLUSION**

Summarizing project achievements, challenges, and potential improvements.

# 10. FUTURE SCOPE

Enhancements such as integrating more rice varieties, improving model accuracy, and developing a mobile app version.

**Expected Output:**

A trained model capable of classifying rice grain types with good accuracy.

A deployable app for rice grain prediction.

Visualizations of training curves and model architecture.

GitHub repo with all project code, images, and documentation.

# 11. APPENDIX

# Dataset Link: [Rice Image Dataset](https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset)

* **Project Demo Link:**

https://drive.google.com/file/d/1-MzKiaU0SYV47o3sPe1\_Zph7R8CUvvf\_/view?usp=drivesdk

**Thank You!**  
*Project guided and supported by SmartBridge.*