Full Stack Development with MERN Project Documentation format

# Introduction

* + **Project Title:** GrainPalette – A Deep Learning Odyssey in Rice Type Classification Through Transfer Learning
  + **Team Members:**
  + **Manavarti Peethambari** – Data Evaluation
  + **Mullapudi Valli Gayathri** – Model Creation and Data Preprocessing

# Project Overview

* + **Purpose:**

GrainPalette is a deep learning-based image classification model that identifies different rice varieties using transfer learning. It aims to support farmers, researchers, and gardening enthusiasts by providing a reliable system to recognize rice types through image uploads. This facilitates informed decision-making in cultivation, resource allocation, and agricultural research.

* + **Features:** Highlight key features and functionalities.
  + Upload an image of rice grains and get instant classification
  + Predicts among five different rice varieties
  + Powered by a Convolutional Neural Network with MobileNetV2
  + High accuracy using pre-trained weights and fine-tuning
  + Useful for farmers, scientists, home growers, and educators

# Architecture

**Frontend:** Developed using **Streamlit**, a lightweight Python library for building interactive web apps.

* Users upload rice grain images through a browser UI.
* The app displays the image and predicts the rice type upon clicking "Predict".
* Prediction confidence is shown with the output.

**Technologies Used:**

* Streamlit
* PIL (for image processing)
* TensorFlow (for loading the model and prediction)

**Backend:** The backend is powered by a CNN-based deep learning model utilizing **MobileNetV2** architecture for transfer learning. The model is trained using TensorFlow and Keras to classify five rice types: Arborio, Basmati, Ipsala, Jasmine, and Karacadag.

**Database:**

 Dataset used: **Rice\_Image\_Dataset**

 The dataset contains labeled image data divided into training and validation sets using ImageDataGenerator with preprocessing for image resizing, scaling, and augmentation.

# Setup Instructions

 **Prerequisites:**

* Python 3.7+
* TensorFlow 2.x
* Keras, NumPy, Matplotlib, Scikit-learn
* Hardware with GPU recommended for faster training
* OS: Windows 8 or later
* Internet Bandwidth: Minimum 30 Mbps

 **Installation:**

1. Clone the repository
2. Install dependencies:

pip install tensorflow matplotlib numpy scikit-learn

1. Place the Rice\_Image\_Dataset folder in the project root
2. Run the training script:

python rice\_model\_training.py

1. The trained model will be saved as rice\_type\_model.h5

# Folder Structure

SMARTINTERNZ\_PROJECT/

│

├── Rice\_Image\_Dataset/

│ ├── Arborio/

│ ├── Basmati/

│ ├── Ipsala/

│ ├── Jasmine/

│ └── Karacadag/

│ └── Rice\_Citation\_Request.txt

│

├── app.py ← Streamlit frontend

├── predict.py ← Image preprocessing & prediction logic

├── testmodel.py ← Training/validation testing scripts

├── rice\_type\_model.py ← Model creation and training

├── rice\_type\_model.h5 ← Final trained model (MobileNetV2)

# Running the Application

 Use the following command to run the **Streamlit frontend**:

streamlit run app.py

 No separate backend server is required since Streamlit handles both UI and model inference internally.

# API Documentation

**Note:** Your project does not use a REST API. All functionality (image upload, processing, and prediction) is handled within the Streamlit app.

If APIs were to be added, endpoints might look like:

* POST /predict – Accepts image and returns predicted rice type
* Parameters: Image file (form-data)
* Response: JSON with predicted label and confidence

# Authentication

No authentication is implemented in this project as it is a standalone local application meant for testing and demo purposes.

For future improvements:

* Add user login system if hosted online
* Use libraries like streamlit-authenticator or Firebase Auth for security

# User Interface

* Built using **Streamlit**, which provides an interactive web UI.
* Key components:
  + Image uploader to accept rice grain images
  + Display area to show uploaded image
  + Button to trigger prediction
  + Text output to display the predicted rice type and confidence

Example UI snapshot:

------------------------------------------

| Rice Type Classifier |

|----------------------------------------|

| [ Upload Image ] |

| |

| [ Predict Button ] |

| |

| Predicted: Jasmine (92.4%) |

------------------------------------------

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\C9FA5A72863B86FE0C7DC5902E21FB17\WhatsApp Image 2025-07-18 at 20.14.23_8ecec8ca.jpg

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\047120FA7417B3D37A0573E8A24C9E43\WhatsApp Image 2025-07-18 at 20.14.23_ec4d794b.jpg

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\2572BD363583B4C28B31FCBB15DF5837\WhatsApp Image 2025-07-18 at 20.14.23_ee0ed1c1.jpg

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\3C8179DE1979ED5F080F0871C7BD9BDE\WhatsApp Image 2025-07-18 at 20.14.22_3831cd09.jpg

# Testing

**Strategy Used**:

* Manually tested with rice grain images from each category (Arborio, Basmati, Ipsala, Jasmine, Karacadag).
* Verified predictions using the trained model (rice\_type\_model.h5) through the Streamlit UI.

**Tools**:

* Streamlit for interactive UI testing.
* TensorFlow/Keras for evaluating model accuracy.
* NumPy and matplotlib (if plots were generated) for debugging model predictions.

# Screenshots or Demo

# 

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\C9FA5A72863B86FE0C7DC5902E21FB17\WhatsApp Image 2025-07-18 at 20.14.23_8ecec8ca.jpg

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\047120FA7417B3D37A0573E8A24C9E43\WhatsApp Image 2025-07-18 at 20.14.23_ec4d794b.jpg

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\2572BD363583B4C28B31FCBB15DF5837\WhatsApp Image 2025-07-18 at 20.14.23_ee0ed1c1.jpg

# C:\Users\mulla\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\3C8179DE1979ED5F080F0871C7BD9BDE\WhatsApp Image 2025-07-18 at 20.14.22_3831cd09.jpg

# Known Issues

 The model might occasionally confuse similar-looking rice types due to close feature patterns.

 Image preprocessing assumes images are clear and focused; blurry images may reduce accuracy.

 Currently supports only .jpg, .png, and .jpeg formats.

# Future Enhancements

 Deploy the model using services like **Streamlit Cloud**, **Heroku**, or **Render**.

 Add **user authentication** for secure access and feedback logging.

 Build a **REST API backend** for model predictions (e.g., using FastAPI or Flask).

 Implement a **data augmentation** pipeline to improve model robustness.

 Add **visualization tools** like Grad-CAM for model explainability.