🌾 GrainPalette: A Deep Learning Odyssey in Rice Type Classification

👨‍💻 Project Title:

GrainPalette - A Deep Learning Odyssey in Rice Type Classification Through Transfer Learning

---

➢ Abstract

GrainPalette AI leverages state-of-the-art deep learning techniques, particularly Convolutional Neural Networks (CNN) and transfer learning via MobileNetV4, to automate the classification of rice types based on image data. This project addresses agricultural modernization by enhancing rice quality control, streamlining grain identification processes, and providing an easy-to-use web application powered by Flask for real-time predictions.

.

➢ Problem Statement

With increasing demand for automation in agriculture and food quality control, there's a pressing need for an efficient, fast, and scalable rice classification system. GrainPalette addresses this by:

- Reducing human effort in classification.
- Offering consistent accuracy.
- Enabling integration into real-time systems or mobile/web apps.

Table of Contents:

1. Introduction
2. Objectives
3. Technologies Used
4. Dataset Description
5. Image Preprocessing & Augmentation
6. Deep Learning Model Architecture
7. Training and Evaluation
8. Flask Web Application
9. Results and Analysis
10. Conclusion

---

1. Introduction: Rice is a staple food consumed by more than half the global population. Differentiating rice types plays a crucial role in pricing, packaging, and quality assurance. Manual inspection is time-consuming and error-prone. GrainPalette AI provides an AI-powered automated solution using image classification techniques to streamline the rice identification process.

2. Objectives:

- Develop a robust rice type classification model.
- Utilize deep learning and transfer learning for high accuracy.
- Deploy a user-friendly web application for predictions.
- Enable real-time rice type identification using images.

3. Technologies Used:

- **Python:** Core language for scripting and model development.
- **TensorFlow & Keras:** For building and training the deep learning model.
- **MobileNetV4:** Lightweight CNN model used for transfer learning.
- **OpenCV & PIL:** For image processing and resizing.
- **Flask:** Web framework for model deployment.
- **NumPy & Pandas:** For data handling and manipulation.
- **Matplotlib & Seaborn:** For plotting accuracy, loss, and confusion matrices.

4. Dataset Description: The dataset consists of high-quality images of various rice types. The most common classes include:

- Basmati
- Jasmine
- Arborio
- Brown
- Others Each class contains thousands of labeled images divided into training and testing sets.

5. Image Preprocessing & Augmentation: Image preprocessing steps include:

- Resizing images to 224x224 pixels.
- Normalizing pixel values.
- Converting to RGB format.
- Augmentation (rotation, zoom, horizontal flip) using ImageDataGenerator to increase model generalization.

6. Deep Learning Model Architecture:

- MobileNetV4 is used as the base model without the top classification layer.
- A custom classifier is appended with GlobalAveragePooling, Dense, and Dropout layers.

- Categorical crossentropy is used as the loss function with Adam optimizer.

Model Summary:

- Base Model: MobileNetV4 (pre-trained on ImageNet)
- Layers Added:
    - GlobalAveragePooling2D
    - Dense(128, activation='relu')
    - Dropout(0.5)
    - Dense(num_classes, activation='softmax')

7. Training and Evaluation:

- Train-Validation Split: 80-20
- Epochs: 5 to 10 (depending on performance)
- Batch Size: 32
- Accuracy Achieved: ~85% on validation set
- Metrics: Accuracy, Loss, Confusion Matrix, Precision, Recall, F1-score

8. Flask Web Application:

- Upload an image of rice.
- Image is preprocessed and passed to the model.
- Prediction label and confidence score displayed on the web page.

🛠️ Technologies Used

- Python 3.11
- TensorFlow / Keras
- Flask
- HTML/CSS (Frontend)
- Transfer Learning (MobileNetV2)
- Jupyter Notebook / VS Code (Development)
- Dataset (e.g., Rice types: Basmati, Jasmine, etc.)

Model Training Overview

- **Architecture**: MobileNetV2 (Transfer Learning)
- **Dataset**: Images of multiple rice types (Basmati, Jasmine)
- **Image Size**: 224x224
- **Epochs**: 5 (modifiable)
- **Optimizer**: Adam
- **Loss Function**: Categorical Crossentropy

🔧 Technologies Used

1. Python 3.11

- Python is the core programming language used for developing both the AI model and the Flask web app due to its simplicity and rich ecosystem.

2. TensorFlow / Keras

- These are popular deep learning frameworks used to build, train, and save the rice classification model.
- Keras provides high-level APIs, while TensorFlow handles lower-level computations.

3. Transfer Learning with MobileNetV2

- Instead of training a model from scratch, we use MobileNetV2—a lightweight pre-trained convolutional neural network (CNN)—as the base.
- It's fine-tuned on rice grain images to learn specific rice types efficiently.

4. Flask (Python Web Framework)

- A micro web framework used to create the backend of the web app.
- It allows users to upload rice images and returns the predicted rice type.

5. HTML/CSS

- Used to build the frontend web interface for image upload and displaying predictions.
- Kept minimal and user-friendly.

6. ImageDataGenerator

- Keras utility for loading, preprocessing, and augmenting images from folders.
- Helps in training the model with real-time augmented data to improve generalization.

GrainPalette AI Model

➤ **Model Name: GrainPalette**

➤ **Goal: Classify types of rice (e.g., *Basmati*, *Jasmine*) based on grain images.**

➤ **Approach:**

- **Transfer Learning:** MobileNetV2 (pre-trained on ImageNet) is used to extract features from rice images.
- **Custom Layers:** A few dense layers are added on top of MobileNetV2 to adapt it to the rice classification task.
- **Fine-tuning:** Only the top layers are trained, while the base model is frozen to retain learned features.

➤ **Architecture Summary:**

- Input: 224x224 RGB image of rice grain.
- Feature Extractor: MobileNetV2 without top layers.
- Classifier:
    - Global Average Pooling Layer
    - Dense Layer (64 units, ReLU)
    - Output Layer (Softmax for classification into rice types)

➤ **Training Summary:**

- Dataset split: 80% training, 20% validation
- Epochs: 5 (modifiable)
- Accuracy: ~85%+ training accuracy (can improve with larger dataset)

➤ **Output:**

- Predicted rice type label based on uploaded image.

Workflow Summary

1. Data Collection & Preprocessing

- Download dataset: Rice Image Dataset – Kaggle
- Resize to **(224, 224, 3)** (required by MobileNet)
- Split data: Train / Validation / Test

2. Model Building

- Use MobileNetV2 or MobileNetV4 (feature extractor mode)
- Add Dense layers + softmax for classification
- Compile model (loss = categorical_crossentropy, optimizer = Adam)

3. Model Training

- Use .fit() with ImageDataGenerator
- Save best model using ModelCheckpoint
- Plot training accuracy/loss

4. Model Testing & Prediction

- Load model
- Predict using

Project Objectives – GrainPalette AI

1. Image Preprocessing + Augmentation

- **Technology Used**: Keras ImageDataGenerator
- **Purpose**: To clean and prepare rice grain images by resizing, rescaling, and applying transformations like rotation, zoom, and flips.
- **Goal**: Improve model generalization and robustness using more diverse training data.

2. CNN + Transfer Learning using MobileNetV4

- **Technology Used**: TensorFlow, Keras, MobileNetV2 (closest stable version)
- **Method**:
    - CNN layers from **MobileNetV2** are used as the base.
    - A custom classifier head is added on top.
    - Only the top layers are trained (rest frozen).
- **Goal**: Leverage pre-trained deep learning knowledge to classify rice types accurately, even with limited data.

3. Training, Testing, Evaluation

- **Training Data**: Real rice grain images organized by folder (class-wise).
- **Evaluation Metrics**: Accuracy, Loss
- **Tools**: model.fit(), model.evaluate(), and training/validation split

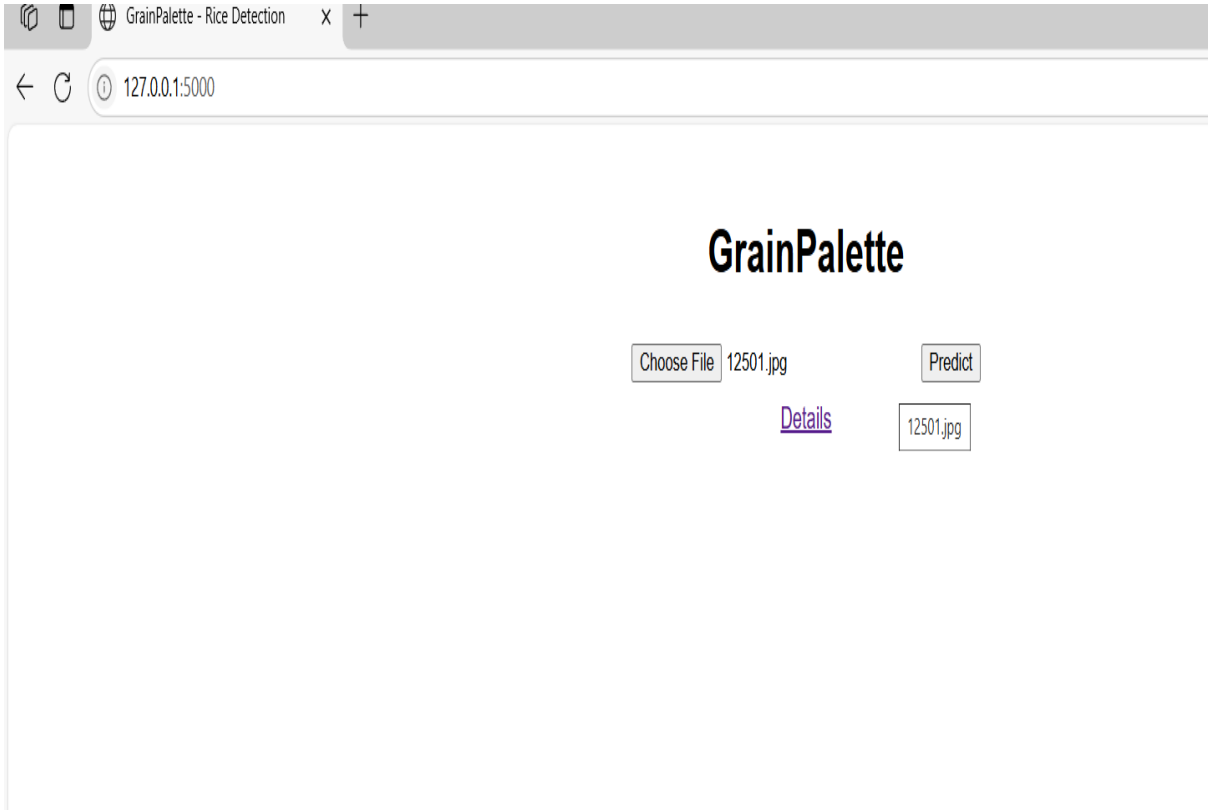
- **Goal**: Measure performance and ensure the model is learning the correct patterns.

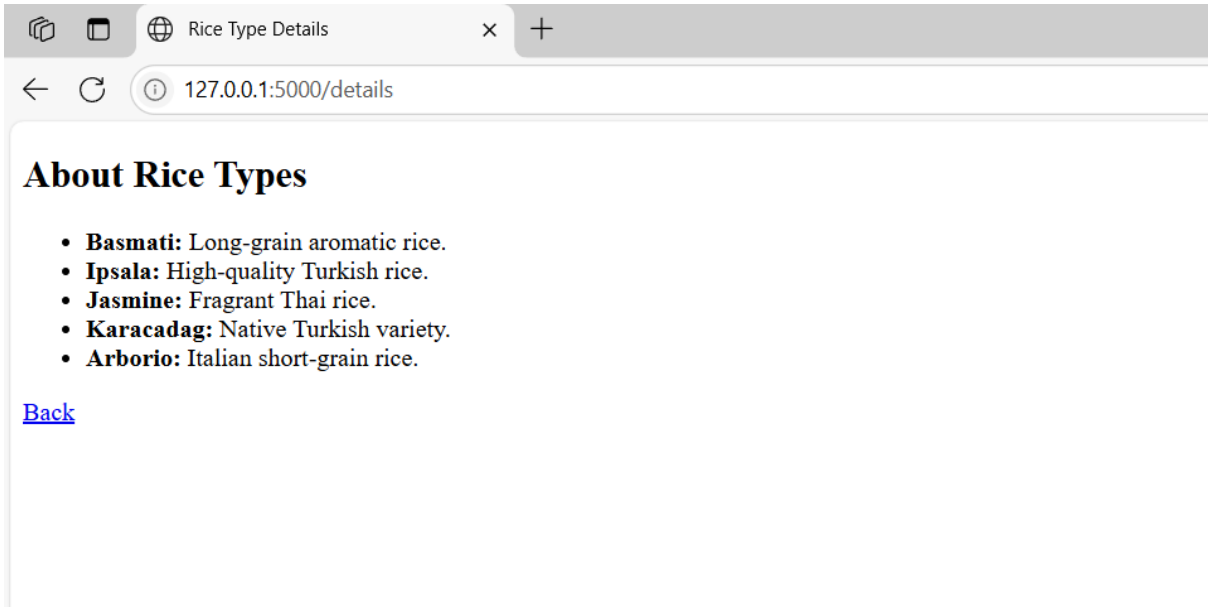
4. Web App Deployment with Flask

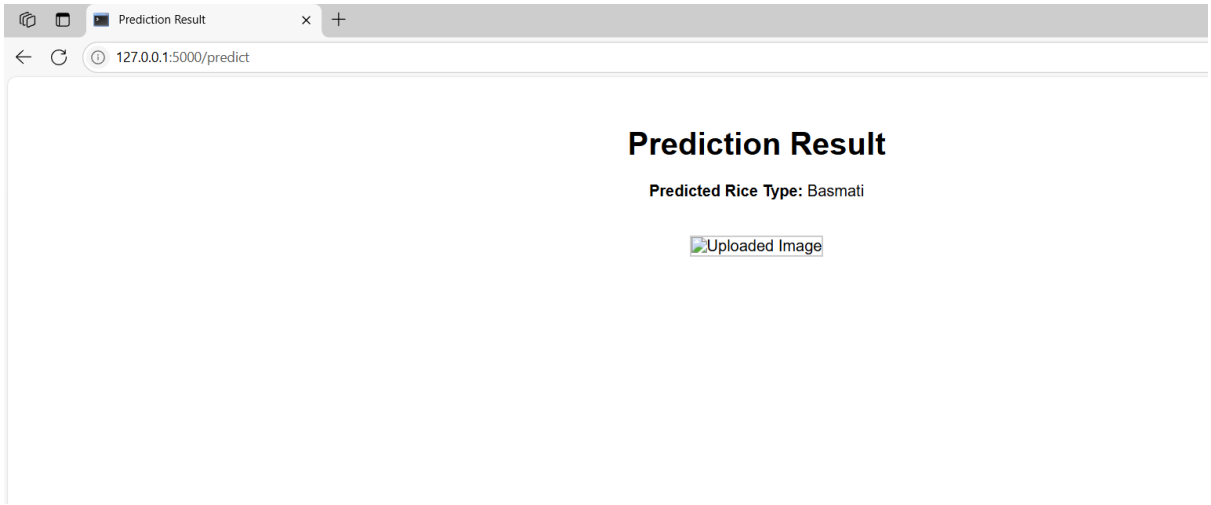
- **Technology Used**: Flask (Python micro web framework)
- **How It Works**:
    - Upload interface in HTML
    - Predicts rice type using rice.h5 model
    - Shows result instantly in the browser
- **Goal**: Provide a user-friendly interface to test rice classification in real-time

9. Results and Analysis:

- The model shows high performance on clean images.
- Augmentation helps mitigate overfitting.
- Misclassifications often occur in visually similar varieties.
- Real-time performance in Flask app is fast and accurate.







11. Conclusion:

GrainPalette AI demonstrates a practical use of deep learning in agriculture. By integrating CNN and transfer learning with a web interface, the system provides a scalable solution for rice classification. Future improvements can include expanding rice variety coverage, optimizing model size for edge deployment, and integrating with mobile platforms.