PROJECT

Project Title :

GrainPalette: A Deep Learning Odyssey in Rice Type Classification through Transfer learning

Team Name :

Team Tech Harvest

Team Members :

- Kocherla Hemanth Kumar
- Karlapati Raja Avinash
- Karella P V S P Ganesh
- Kodali Sandeep

Phase-1 : Brainstorming & Ideation

Objective:

To develop an efficient deep learning-based model using transfer learning techniques for accurately classifying different rice types, thereby enhancing automation and precision in agricultural product identification.

Key Points:

1. **Problem Statement:**
   - Rice, a staple food for billions ,comes in hundreds of varieties that differ in size, shape and texture. Accurate classification is critical for quality control and trade. Manual methods are labor-intensive and inconsistent.
   - Grain Palette introduces a transfer learning -based model that classifies rice grains with precision. The solution automates grain sorting, enhances packaging accuracy and supports agricultural commerce with reliable identification tools.
2. **Proposed Solution:**

   - To address the challenge of rice type classification, we propose a deep learning-based approach leveraging transfer learning. Pre-trained convolutional neural networks (CNNs) such as VGG16, ResNet50, or EfficientNet will be fine-tuned on a curated dataset of rice grain images. This approach significantly reduces training time and improves accuracy, especially when working with limited labeled data.

3. **Target Users:**

   - Farmers: To help them identify and sort rice varieties accurately post-harvest, enabling better crop value estimation and market readiness.
   - Agricultural Researchers: For analyzing grain characteristics and automating classification in crop research and development
   - Food Grain Traders & Exporters: For ensuring proper labeling and quality assurance of rice varieties in trade and export documentation.

4. **Expected Outcome:**

   - A reliable and efficient deep learning model capable of accurately classifying multiple rice types based on image data.
   - A user-friendly system or interface (if implemented) that can be integrated into existing agri-tech workflows.

- A significant reduction in time and cost for rice type identification in agricultural and commercial settings.

Phase-2:

Objective:

Requirement Analysis

Define the technical and functional requirements for the Tech Harvest application.

Key Points:

1. **Technical Requirements:**

    - **Programming Language: Python**
    - **Python Packages:** NumPy, Pandas, Scikit-learn, Matplotlib, TensorFlow, Flask
    - **Frameworks:** Flask for web integration, TensorFlow for deep learning
    - **Pre-trained Model:** VGG16 (used for transfer learning)
    - **Development Tools:** Command Line (pip install)

2. **Functional Requirements:**

    - The system should be capable of accepting rice grain images as input from users and preprocessing them through normalization, and augmentation to ensure compatibility with the deep learning model.
    - A pre-trained transfer learning model will be utilized to classify the rice type based on the image features.
    - Upon prediction, the system should display the identified rice type along with a confidence score.
    - During development and testing, the system must support the use of labeled datasets and enable evaluation through metrics such as accuracy, precision, recall to assess model performance.

3. **Constraints & Challenges:**

    - Limited Dataset Availability: Difficulty in accessing a large and diverse labeled dataset of rice grain images.

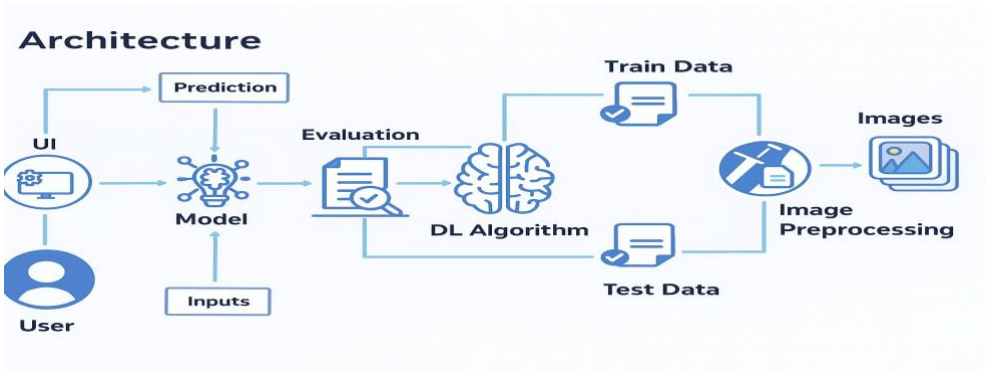
Phase-3:

Objective:

- High Visual Similarity: Some rice types look very similar, making classification challenging even for deep learning models.
- Optimizing model size and performance for faster processing and easy deployment.
- Ensuring the web interface is responsive and user-friendly on all devices.

Project Design

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**

Phase-4:

Objective:

- Input image of rice grain is captured or uploaded by the user.
- Preprocessed image is fed into a pre-trained CNN model (e.g., ResNet50, EfficientNet) using transfer learning.
- Predicted rice type along with confidence score is generated as output.
- The prediction is evaluated and displayed back on the UI with confidence levels.
- During development, the system logs performance metrics like accuracy, precision, recall, and F1-score.
- The training data is used to fine-tune the DL algorithm using transfer learning, while test data is used for evaluation.

2. User Flow:

- Step 1: User opens the web application.
- Step 2: User uploads or captures an image of a rice grain sample through the interface.
- Step 3: The deep learning model processes the image and classifies the rice type using transfer learning.
- Step 4: Predicted rice type and confidence score are displayed to the user on the screen.
- Step 5: User can upload another image or exit the system after viewing the result.

Phase-5:

Objective:

Project Planning

Breakdown development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned to	Dependencies	Expected outcome
Sprint 1	Environment Setup & Package Installation	High	3 hours	Day 1	Member 1	Anaconda , Python	Project environment ready
Sprint 1	Dataset Collection & Preprocessing	High	4 hours	Day 1	Member 2	Dataset access	Clean,prepared image dataset
Sprint 2	Model Building using Transfer Learning	High	5 hours	Day 2	Member 3	Preprocessed data, TensorFlow	Trained classification model

Phase-6:

Objective:

Sprint 2	Flask Web App Integration	Medium	3 hours	Day 2	Member 1 & 4	Trained Model,Flask installed	Working web interface
Sprint 3	Testing & Debugging	Medium	2 hours	Day 2	Member 2 & 3	Complete System	Bug-free and responsive system
Sprint 3	Final Presentation & Deployment	Low	1 hour	End of Day 2	Entire Team	Working application	Project deployed and demo-ready

Sprint Planning with Priorities

Sprint 1 - Setup & Preparation (Day 1)

High Priority

- Collect and curate a labeled rice grain image dataset
- Perform data cleaning and augmentation

- Set up the development environment (Python, TensorFlow/PyTorch, OpenCV).

Sprint 2 – Model Development & Integration (Day 2)

🔴 High Priority

- Implement transfer learning with the selected model.
- Integrate the trained model with the Flask web application.

Sprint 3 – Testing, Deployment & Submission (Day 3)

🟡 Medium Priority

- Test the app functionality, fix bugs, and improve UI responsiveness.

🟢 Low Priority

- Finalize deployment and prepare presentation/demo materials.

Phase-5: Project Development

Objective:

Implement the core features of the HematoVision application using transfer learning for blood cell classification.

Key Points:

1. **Technology Stack Used:**

   - **Frontend:** HTML (via Flask templates)
   - **Backend:** Flask Framework
   - **Deep Learning:** TensorFlow with pre-trained VGG16 model
   - **Programming Language:** Python

2. **Development Process:**

   - Identify the problem and gather requirements. Plan the tasks, timeline, and tools to be used
   - Design system architecture and model workflow.
   - Integrate the trained model into a flask application.
   - Developed a user interface to upload images and display classification **results with prediction confidence**.

3. **Challenges & Fixes:**

   - **Challenge:** Limited or imbalanced dataset.

   Fix: Applied data augmentation to increase and balance the dataset.

   - **Challenge:** Overfitting during training.
   - **Fix:** Used dropout layers and early stopping to prevent overfitting.
   - **Challenge:** Poor image quality or inconsistent resolution.
     Fix: Preprocessed images using OpenCV (resizing, normalization).

Phase-6: Functional & Performance

Objective:

Ensure that the Tech Harvest application performs accurately, reliably, and consistently across various test cases and environments.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Upload image of rice.	Correct cell type identified with confidence score	✅ Passed	Tester 1
TC-002	Functional Testing	Upload mixed grains image	Multi-class classification displayed correctly	✅ Passed	Tester 2
TC-003	Performance Testing	Check model response time	Results displayed under 2 secods	⚠Needs Optimization	Tester 3
TC-004	Bug Fix Validation	Image with poor lighting	System still makes reasonable prediction	✅ Fixed	Developer
TC-005	UI Responsiveness	Test on mobile browser	Layout adjusts properly on mobile	❌ Failed	Tester 2
TC-006	Deployment Testing	Hosted on local server and accessed remotely	App loads and predicts successfully online	🚀 Deployed	DevOps