Project Report
on

Potato Varieties Identification using Image Classification

(A Project Report submitted in partial fulfillment of the requirements of Bachelor of Technology in Information Technology of the Maulana Abul Kalam Azad University of Technology, West Bengal)

Submitted by

Rahul Das (10200222070)
Debraj Ghosh (10200222066)

Under the guidance of

Dr. Satyendra Nath Mandal
Assistant Prof. in Dept. of Information Technology

Kalyani Government Engineering College

(Affiliated to West Bengal University of Technology)

Kalyani - 741235, Nadia, WB

2024-2025

Phone: 25826680 (PBX) **e-mail :**
Fax : 25821309@

কল্যাণী - ৭৪১ ২৩৫
নদীয়া, পশ্চিমবঙ্গ

Kalyani 741 235
Nadia, West Bengal,India

পত্রাঙ্ক /Ref. No.:
তারিখ / Date :

কল্যাণী গভঃ ইঞ্জিনিয়ারিং কলেজ
Kalyani Government Engineering College
(Govt. of West Bengal)

Certificate of Approval

This is to certify that ……has done final year project work entitled……under my direct supervision and he/she has fulfilled all the requirements of relating to the Final Year Project. It is also certified that this project work being submitted, fulfills the norms of academic standard for B. Tech Degree in Information Technology of The West Bengal University of Technology and it has not been submitted for any degree whatsoever by him/her or anyone else previously.

……	……..
Head	Supervisor
Department of Information Technology	Department of Information Technology
Kalyani Government Engineering College	Kalyani Government Engineering College

……	……
Project Coordinator	Examiner
Department of Information Technology	
Kalyani Government Engineering College	

Acknowledgement

We take this opportunity to express our heartfelt gratitude to everyone who supported and guided us throughout this project. This work would not have been possible without their encouragement and assistance.

We extend our sincere gratitude to **Dr. Satyendra Nath Mandal, Assistant Professo**r in the Department of IT, for his invaluable guidance, unwavering support, and expert advice throughout the entirety of this project. His mentorship and encouragement have been indispensable in shaping our project.

We would also like to extend my gratitude to **Kalyani Government Engineering College** and the **Department of Information Technology** for providing the necessary resources and a conducive learning environment to carry out this project.

Our special thanks goes to our friends and peers for their encouragement and collaboration during challenging times. Their insights and discussions added significant value to my work.

Thank you all for your contribution to making this project a meaningful and rewarding experience.

Rahul Das (10200222070)

Debraj Ghosh (10200222066)

Date: 18 June 2025

Abstract

The Potato Identification System is an advanced AI-driven solution designed to classify potato breeds with high accuracy by analyzing image data. Utilizing Convolutional Neural Networks (CNNs) and texture analysis through Gray Level Co-occurrence Matrices (GLCM), the system automates the traditionally manual task of potato breed identification, significantly enhancing efficiency and precision. This project addresses key challenges faced in agriculture, such as quality control, breed-specific disease detection, and scalability in operations.

The system workflow includes image preprocessing, feature extraction, model training, and real-time prediction. Preprocessing ensures uniformity in input data, while CNNs extract breed-specific patterns for classification. Haralick features derived from GLCM offer additional insights into potato textures, enhancing model performance. By leveraging data augmentation and scalable architectures, the system achieves robust generalization across varying conditions.

This system provides an innovative approach to agricultural automation by reducing dependency on human expertise, improving decision-making in the supply chain, and ensuring consistent quality grading. Deployed as a standalone application or integrated with IoT devices, the Potato Identification System holds significant potential to revolutionize modern farming practices.

Keywords
Potato Identification, Convolutional Neural Networks (CNNs), Haralick Features, GLCM, Texture Analysis, Agricultural Automation, Quality Grading, Disease Detection, Data Augmentation, Real-Time Decision Making, HTML, CSS, Web App.

Table of Content

Chapter : 1

Introduction

The classification of potato tuber varieties is a critical aspect of agricultural research and practice, allowing for accurate identification and differentiation of potatoes based on their physical and textural characteristics. This project presents a comprehensive approach to developing a classification system for potato tuber varieties, integrating both traditional feature extraction methods and modern deep learning techniques. The system is designed to train on a dataset of potato tuber images by extracting key features, including texture, color histograms, and shape descriptors, which form the foundation for subsequent predictions. By leveraging these features, the system is capable of testing new images and predicting their variety with high accuracy, guided by the knowledge derived from the training data.

The training process involves organizing the dataset into classes representing different potato varieties and preprocessing the images to extract meaningful information. Traditional features such as texture properties derived from the Gray-Level Co-occurrence Matrix (GLCM), color distribution represented through histograms, and shape features like area and circularity are calculated. Simultaneously, a Convolutional Neural Network (CNN) is employed to extract high-level patterns from the images. These two streams of information are fused in a hybrid model architecture that combines the strengths of handcrafted features and deep learning, enabling robust classification of potato varieties.

The trained model is designed to handle new images by preprocessing them in a manner consistent with the training phase. The extracted features and normalized images are fed into the model to predict the class of the potato variety. This system not only provides the predicted variety but also offers confidence scores and detailed feature data for further analysis.

This approach demonstrates the potential of combining domain-specific knowledge with advanced machine learning techniques to address practical agricultural challenges. By integrating traditional feature engineering with deep learning, the system achieves both robustness and scalability, ensuring its relevance for broader applications in agricultural classification and beyond.

Chapter : 2

Motivation

The motivation for undertaking this project stems from the vital role that accurate classification of potato tuber varieties plays in agriculture, research, and commerce. Potatoes are one of the most widely cultivated and consumed crops globally, with their varieties differing significantly in terms of appearance, texture, and suitability for specific culinary or industrial purposes. Traditional methods of identifying potato varieties often rely on manual inspection, which can be time-consuming, error-prone, and heavily dependent on expert knowledge. This project aims to address these challenges by leveraging advanced machine learning techniques for automated, accurate, and scalable classification of potato tuber varieties.

One key aspect of motivation is the potential to enhance agricultural productivity and supply chain efficiency. Farmers, breeders, and distributors require precise identification of potato varieties to optimize cultivation practices, manage crop quality, and meet market demands. An automated system can provide real-time insights, enabling stakeholders to make informed decisions and reducing the reliance on subjective assessments.

Another motivating factor is the opportunity to contribute to the advancement of agricultural technology through the integration of traditional feature engineering with modern deep learning approaches. By extracting meaningful features such as texture, color, and shape from potato images and combining these with CNN-based representations, the project demonstrates the power of hybrid models in tackling domain-specific problems. This methodology not only ensures higher accuracy but also adds interpretability by revealing the underlying factors influencing classification.

Moreover, this project addresses the growing need for scalable solutions in agricultural classification. With the increasing availability of digital tools and datasets, there is a pressing need to develop systems that can handle large-scale applications while maintaining accuracy and reliability. The incorporation of advanced preprocessing techniques and hybrid modeling ensures that this system is both adaptable and robust, making it suitable for diverse agricultural environments.

Ultimately, the motivation behind this project lies in the desire to harness technology for practical agricultural challenges, reduce manual labor, and create a system that benefits farmers, researchers, and the broader agricultural community. By combining innovation with practicality, this project aims to pave the way for future advancements in crop classification and precision agriculture.

Chapter : 3

Background

Potatoes, being one of the most widely cultivated crops, contribute significantly to global food production and economic stability. However, challenges such as misidentification of tuber varieties, inconsistent quality control, and delayed detection of diseases pose risks to production and supply chains. Conventional methods rely heavily on manual inspection, which is subjective and varies depending on the inspector's expertise.

Recent advancements in artificial intelligence (AI) and computer vision have paved the way for automating such tasks. Techniques like texture analysis, color histogram calculation, and shape feature extraction provide insights into the visual properties of potato tubers. With the advent of deep learning and advancements in image processing, automated systems now offer a promising alternative by analyzing visual features directly from images. This project adopts a hybrid approach by combining deep learning through Convolutional Neural Networks (CNNs) with traditional image feature extraction methods like texture analysis (GLCM), color histograms, and shape descriptors. A CNN model based on the Inception architecture was trained to recognize and classify potato varieties using canopy images collected under natural field conditions. The training and testing process involved preparing the dataset, extracting features, normalizing inputs, and fitting a dual-branch neural network that merged both image and numerical inputs. The model achieved high classification accuracy, showcasing the robustness of combining visual deep learning with handcrafted features. To make the solution practically deployable, it is also envisioned as a web-based tool, where users can upload an image through a browser interface built with HTML, CSS, and JavaScript, which communicates with the backend model via an API. This integration bridges the gap between machine learning research and real-world agricultural applications, offering a scalable and intelligent solution for farmers, agronomists and researchers.

Chapter :4

Project Problem Statement

Current Situation

The agricultural industry heavily relies on the accurate identification of potato breeds for quality control, disease detection, and optimized crop yield. Manual identification of potato breeds is time-consuming, error-prone, and requires expertise that is not always readily available. In recent years, machine learning techniques have shown promise in automating these tasks, but there is a lack of easily deployable solutions tailored specifically for classifying potato breeds with high accuracy.

Challenges :

- **Variability in Image Data**: Images of potatoes can vary significantly due to differences in lighting, angle, resolution, and background, making it challenging to build a robust classification model.
- **Dataset Limitations**: Collecting a comprehensive dataset of potato breeds that covers diverse conditions and breeds is time-intensive and costly.
- **Feature Extraction Complexity**: Extracting meaningful features such as Haralick features using GLCM for texture analysis requires specialized preprocessing and is computationally intensive.
- **Model Accuracy**: Achieving high classification accuracy requires careful design and tuning of the CNN architecture.
- **Deployment**: Ensuring the model can be easily deployed and used by non-technical users in real-world agricultural settings.
- **Similarity** : Every Potato looks similar so with the bare eye we can not determine which potato tuber it is.

Opportunities :

- **Automation in Agriculture**: An automated solution for potato breed classification can significantly reduce dependency on manual labor and expertise.
- **Scalability**: A well-trained model can be scaled to classify not only potato breeds but also other agricultural products.
- **Real-Time Decision Making**: Integrating the classification model with mobile or web-based applications can enable real-time breed identification and decision-making in the field.
- **Improved Crop Quality**: Accurate breed identification can lead to better crop management, disease control, and ultimately higher yield

Chapter:5

Literature Review

5.1 Overview of AI-Based Crop Classification and Its Role in Agricultural Digitization

In recent years, the application of artificial intelligence (AI), particularly deep learning, has revolutionized the way agricultural systems are managed and optimized. One major area of advancement is crop variety identification, which traditionally relies on manual morphological traits or laboratory-based genotypic tests. These methods are often time-consuming, expensive, and require technical expertise. With the proliferation of image processing and machine learning tools, automated crop classification systems have emerged as viable alternatives, providing speed, accuracy, and scalability.

Potato, being a widely cultivated crop with numerous regional varieties, poses a unique challenge due to its morphological similarities across types. This has driven research towards the integration of CNN-based image classifiers, which can learn discriminative features directly from potato canopy or tuber images.

5.1.1 Performance of CNN-Based Classifiers in Agriculture

- Feature Extraction and CNN Fusion: Several studies have used combined approaches where handcrafted features (e.g., GLCM texture, shape metrics) are merged with deep learning features. This hybrid methodology boosts classification performance, especially when datasets are limited in size.
- Accuracy and Scalability: Experiments in potato variety classification have shown that CNN models like InceptionV3, when trained with augmented datasets, can achieve over 90–100% accuracy. This matches or exceeds traditional classification methods while offering real-time usability.
- Deployment for Field Use: By integrating TensorFlow-based CNN models with web interfaces, researchers have enabled broader access to AI-powered diagnosis tools for farmers. Such deployments ensure on-field usability through image upload portals or camera-based classification on mobile devices.[7]

5.2 Website Interface for AI-Based Potato Classifier

To increase accessibility and ease of use, a web-based front end was developed using HTML, CSS, and JavaScript. The interface guides users through the process of uploading an image or capturing it via webcam, which is then sent to a Flask-based Python backend hosting the trained classification model.

The interface, as shown in the screenshots below, offers a clean design with intuitive buttons:
- "Let's Go" to begin classification
- "Upload Image" and "Take a Picture" to acquire input
- A dynamic result panel to display the predicted potato variety (e.g., "Prediction: Chipsona")

This user experience supports wide adoption, particularly by farmers, researchers, and agricultural extension workers.

5.3 Existing Solutions and Their Limitations

Solution	Research Study/Implementation	Advantages	Limitations
CNN-Based Potato Variety Classification	Deep Learning for Crop Classification: A Case Study on Potato Varieties	High accuracy with image data; scalable with cloud/web deployment	Requires large, labeled datasets; high computation for training
Hybrid Feature + CNN Fusion Models	Fusion Models for Agricultural Classification Tasks	Robust to low-quality images by using both raw and traditional features	Slightly more complex integration; preprocessing overhead
Web-Based Agricultural AI Platforms	Smart Crop Classifier on the Web Using Flask & TensorFlow	Real-time, user-friendly; can be deployed on mobile or desktop browsers	Dependent on internet; server maintenance required
Mobile-Based Leaf and Tuber Scanning Apps	Commercial apps in seed certification and quality control	Portable and field-ready	May suffer in poor lighting or with camera variability

Chapter : 6

Design and Development

A Potato Identification System is an AI-based solution designed to classify potato breeds by analyzing images using machine learning and deep learning techniques. This system leverages Convolutional Neural Networks (CNNs) for image classification and optionally integrates Haralick features derived from Gray Level Co-occurrence Matrices (GLCM) for texture analysis. It aims to provide a robust and efficient mechanism for automating the identification of potato breeds in agricultural and commercial settings [3].

6.1 Key Components of the System

6.1.1. Image Data

- **Input**: Images of potatoes from diverse breeds captured under various conditions (lighting, angle, and background).
- **Requirements**: High-quality images that are preprocessed to ensure consistency in size, resolution, and format.

6.1.2. Feature Extraction

- **Texture Analysis**: Using Haralick features (e.g., contrast, correlation, energy, homogeneity) derived from GLCM to capture textural patterns.
- **Color and Shape Analysis**: Analyzing variations in skin tone, shape, and blemishes to differentiate breeds.

Model Architecture

- **CNN Layers**: Designed to detect patterns in pixel intensity for breed-specific features.
- **Fully Connected Layers**: Combine extracted features for final classification [6].

Deployment

- Can be deployed as a web app, mobile app, or integrated with IoT devices for on-field use.
- Portable solutions, such as edge devices, for real-time predictions.

6.2 Need for the System

Agricultural Efficiency

- Reduces the time and expertise required for manual potato breed identification.
- Helps farmers and agricultural companies streamline operations [9].

Quality Control

- Ensures the correct breeds are used for specific purposes (e.g., frying, baking, or processing).
- Enhances market value by grading potatoes based on quality.

Disease Management

- Identifies breed-specific vulnerabilities to diseases for early intervention.
- Supports breeding programs to develop resistant potato varieties.

Cost Savings

- Lowers labor costs by automating tedious identification tasks.
- Reduces errors, saving resources wasted on incorrect sorting or grading.

Scalability

- A scalable solution that can handle thousands of images and be expanded to include new breeds or tasks like disease detection.

6.3 Project Workflow

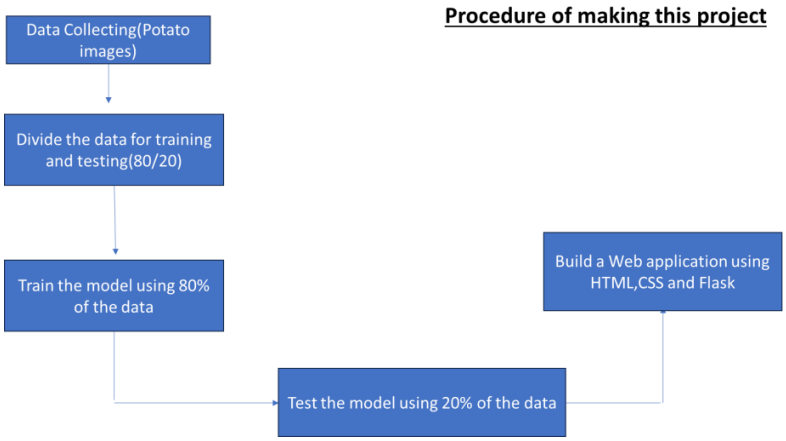

Fig : Project Workflow

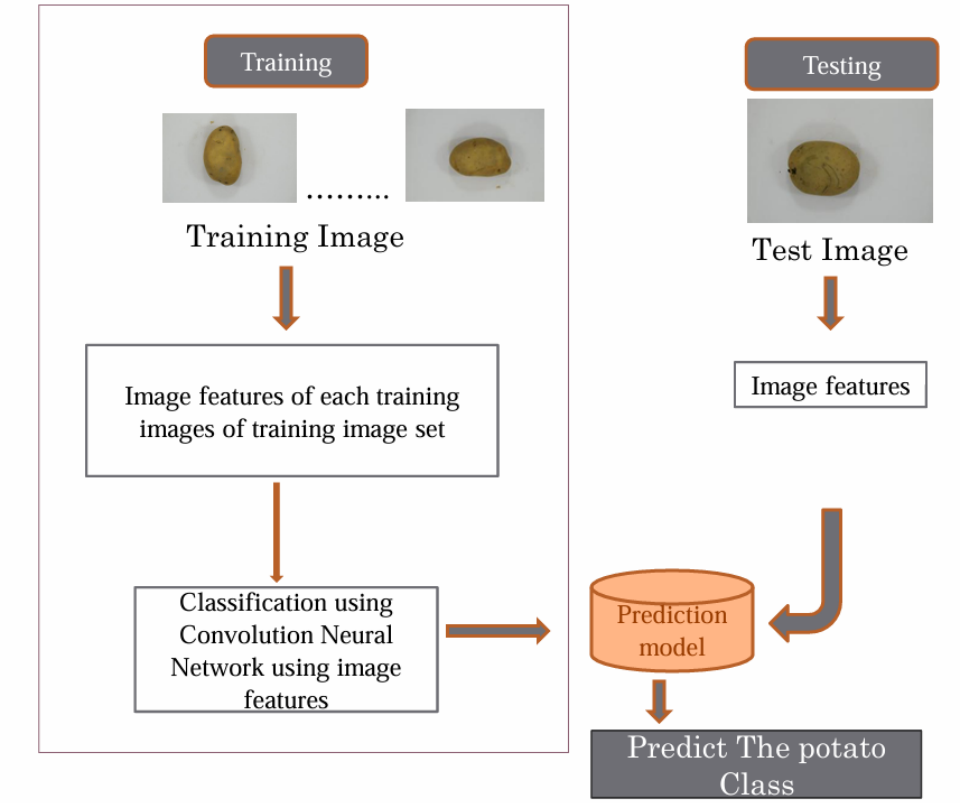

Fig : Detailed Project Workflow

Data Collection (Potato Images)

- **Objective**: Gather a dataset of potato images, ensuring a variety of samples for different classes (e.g., healthy potatoes, diseased potatoes).
- **Steps**:
    - Collect images from various sources such as agricultural datasets, online repositories, or manually captured photos.
    - Organize the data into labeled categories corresponding to their health status or disease type.
    - Ensure the dataset is balanced to avoid bias during model training.

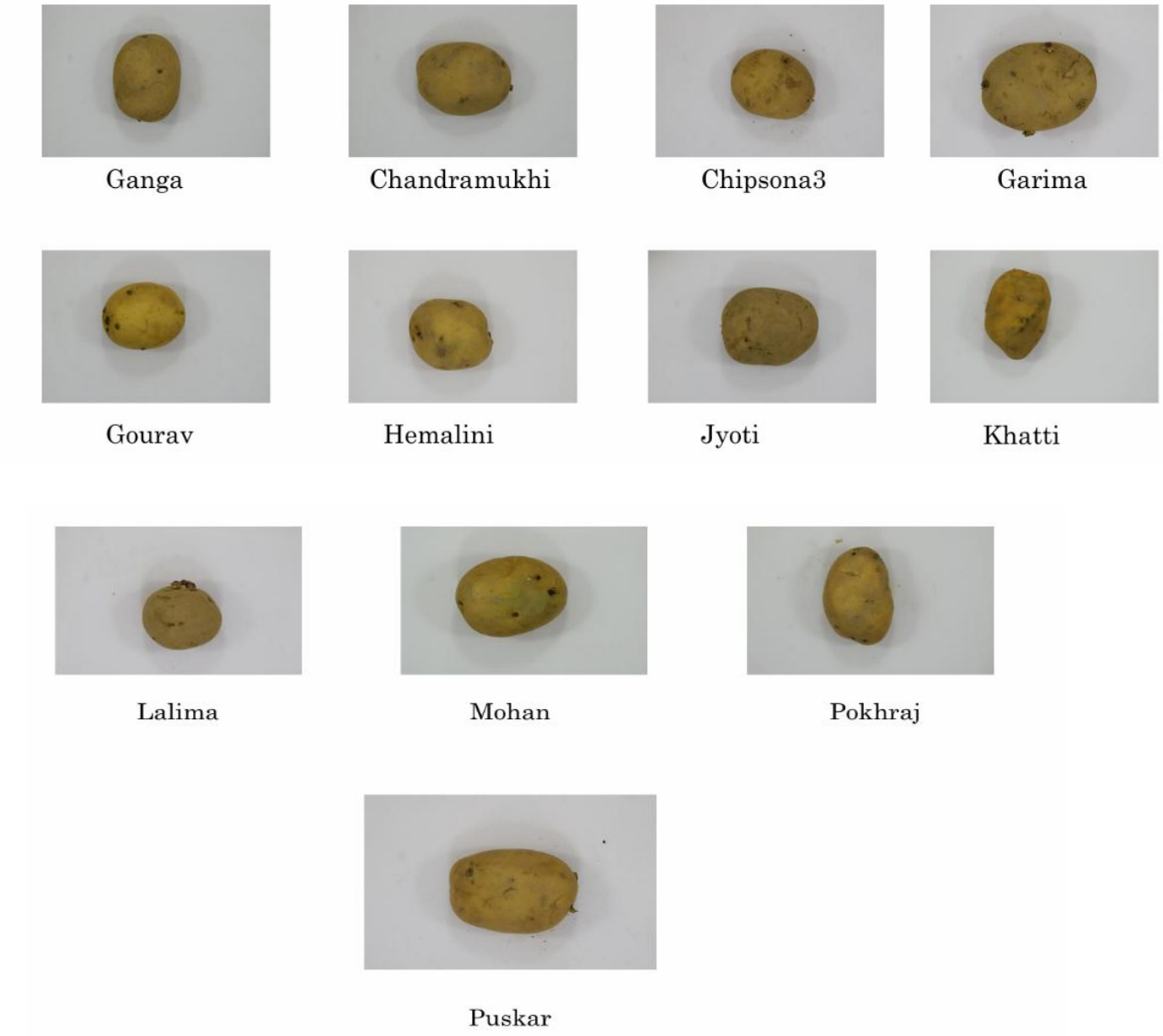

Fig : Varieties of Potato we use for making dataset

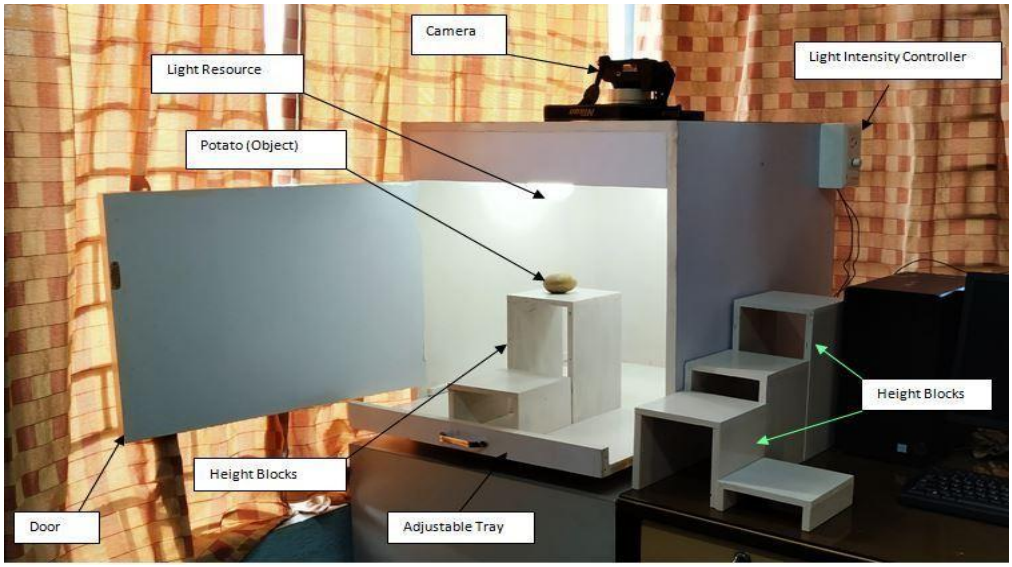

Fig : Setup used for taking the pictures

Potato Varity	Mode	Number of Training Image	Number of Testing Image
Ganga	DSLR/Mobile	268	30
Chandramukhi	DSLR/Mobile	267	30
Chipsona3	DSLR/Mobile	233	30
Garima	DSLR/Mobile	368	30
Gourav	DSLR/Mobile	285	30
Hamlini	DSLR/Mobile	284	30
Jyoti	DSLR/Mobile	346	30
Khatti	DSLR/Mobile	270	30
Lalima	DSLR/Mobile	359	30
Puskar	DSLR/Mobile	328	30
Mohan	DSLR/Mobile	328	30
Pokhraj	DSLR/Mobile	320	30

Fig : Potato Image Database

6.4. Divide the Data for Training and Testing (80/20 Split)

- **Objective**: Split the dataset into training and testing subsets to evaluate the model effectively.
- **Steps**:
    - Use 80% of the data for training the model.
    - Allocate 20% of the data for testing to evaluate model performance.
    - Perform stratified sampling to maintain class balance in both subsets.

6.5. Train the Model Using 80% of the Data

- **Objective**: Train the hybrid model (CNN + traditional features) on the training data to recognize patterns in potato images.
- **Steps**:
    - Use the hybrid model architecture that combines CNN for image features and traditional feature engineering for color, texture, and shape analysis.
    - Implement data augmentation techniques (e.g., rotation, flipping, brightness adjustment) to improve model generalization.
    - Set up training parameters such as learning rate, batch size, and the number of epochs.
    - Use optimization algorithms (e.g., Adam optimizer) and evaluate the model on the validation set during training to monitor overfitting.[10]

6.6. Test the Model Using 20% of the Data

- **Objective**: Evaluate the trained model on the testing dataset to measure its performance.
- **Steps**:
    - Load the saved model and run predictions on the testing dataset.
    - Calculate performance metrics such as accuracy, precision, recall, and F1-score.
    - Identify misclassified samples and analyze their characteristics for potential improvements.

6.7.Build a Web Application Using HTML, CSS, and Flask

- **Objective**: Deploy the trained model in a user-friendly web interface for practical use.
- **Steps**:
    - **Frontend Development**:
        - Design the user interface using HTML and CSS.
        - Create a simple form to allow users to upload potato images for classification.
    - **Backend Development**:
        - Use Flask as the backend framework to handle user requests and connect to the trained model.
        - Write Python code to preprocess uploaded images, pass them through the model, and return classification results.
    - **Integration**:
        - Integrate the model's predictions into the web interface to display the result (e.g., "Healthy Potato" or "Diseased Potato").
        - Host the application locally or on a cloud platform for wider accessibility.

Chapter : 7
Technologies Used

Frontend (User Interface)

- **HTML :** Defines the structure of the web pages (e.g., headings, forms, buttons).
- **CSS :** Styles the HTML elements (e.g., background image, font, layout, animations).
- **JavaScript :** Adds interactivity like image previews or button effects.

Backend (Server-Side Logic)

- **Flask :** A lightweight web framework used to handle web requests, process form data (like image uploads), and return results to the user.

Machine Learning / AI Stack

- **TensorFlow / Keras** : Used to build, train, and load the CNN model for classifying potato images.
- **NumPy** : For numerical operations like reshaping arrays and preprocessing images.
- **Pandas** : For loading structured data (e.g., features, results) and exporting predictions to Excel.
- **scikit-learn** : Used to apply the same feature scaling (standardization or normalization) to input images as during training (via `scaler.pkl`).
- **OpenCV / PIL** : For reading, resizing, and converting images to a format suitable for prediction.

Model and Data Handling

- **.keras file :** Contains the saved CNN model trained on potato images, which is loaded during prediction.

- **scaler.pkl :** Stores the fitted scaler (e.g., StandardScaler) used during training, to ensure the new input data is scaled the same way.

Programming Language

- **Python** : To write the code and train the model,test the model and deploying the model.

Chapter : 8

Potato Variety Identification (Training & Testing)

8.1. Training

The first step of this project is making / train the model so it can successfully predict the unseen potato images and determine the accurate class of the potato. Then when the training process is done then it save the file as .keras and .pkl file so it should be use for predicting the verity of the potato.

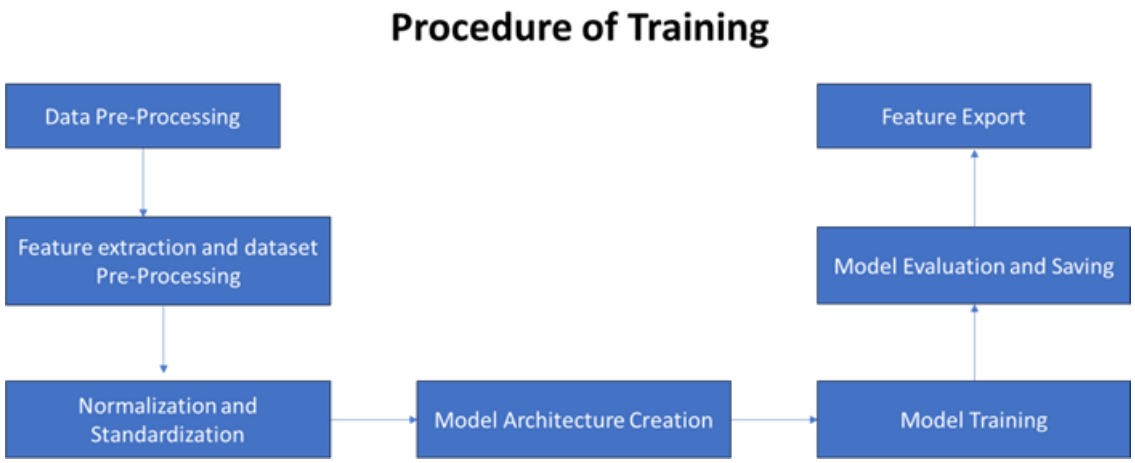

Fig : Training Workflow

8.1.1. Data Pre-Processing:

- **Objective**: To ensure raw data is in a usable format for feature extraction and model training. Using "preprocess_image(image_path)" function to do that.
- **Steps**:
- **Loading Images**: Read the potato images from the dataset directory.
- **Handling Missing or Corrupt Images**: Validate images and handle cases where images are missing or unreadable.
- **Resizing**: Resize images to a standard size (e.g., 224x224 pixels) to ensure uniformity for the CNN model.
- **Colour Space Conversion**: Convert images to grayscale for texture and shape feature extraction.

Extracts:

- Shape Features from original image
- Texture + Color from resized image

Returns:

- CNN input image
- Combined feature vector of 9 handcrafted features

8.1.2. Feature Extraction :

Objective: To extract meaningful features for classification from the images.

Steps:

1. Texture Features: Use Gray Level Co-occurrence Matrix (GLCM) to extract features like contrast, dissimilarity, homogeneity, energy, and correlation. Converts to grayscale. Calculates GLCM (Gray-Level Co-occurrence Matrix)

Extracts: Contrast, Dissimilarity, Homogeneity, Energy, Correlation

2. Colour Features: Extract colour histograms from the images to understand their colour distribution.3D histogram (RGB), 8×8×8 bins. Normalized using cv2.normalize. Averaged for one feature.[9]

3. Shape Features: Calculate area, perimeter, and circularity of contours found in the image.

Converts to grayscale, Applies adaptive threshold, Detects contours

Calculates:

- Normalized Area
- Normalized Perimeter
- Circularity:

$$\frac{4\pi A}{l^2}$$

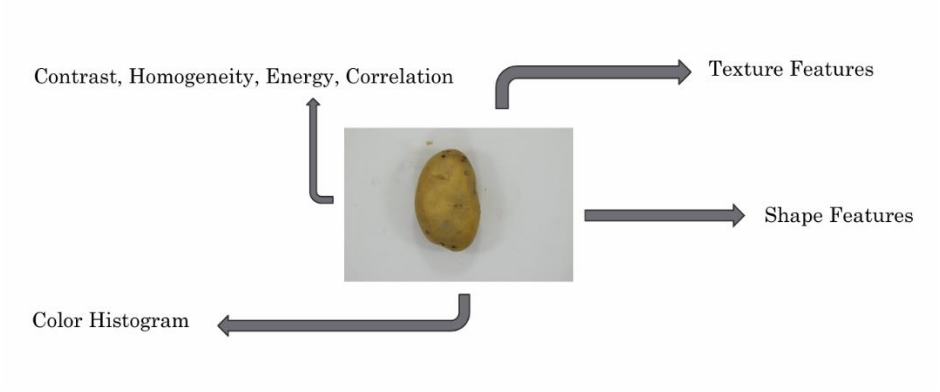

Fig : Features Extraction from a Potato

8.1.3. Normalization and Standardization:

- **Objective**: Scale the extracted features to ensure uniformity and improve model performance.
- **Steps**:

Normalization: Normalize pixel values of images to a range of [0, 1] by dividing by 255.

Standardization: Use a scaler (e.g., StandardScaler) to standardize extracted features to have a mean of 0 and a standard deviation of 1.

8.1.4. Model Architecture Creation:

- **Objective**: Design a hybrid model that combines CNN and traditional feature-based inputs.
- **Steps**:
- **CNN Branch**: Create a convolutional neural network (CNN) to process image inputs.
- **Traditional Features Branch**: Use dense layers to process extracted texture, color, and shape features.
- **Combining Branches**: Concatenate outputs of both branches for joint learning.
- **Final Layer**: Add a dense softmax layer for classification into the appropriate potato class.

The create_combined_model function builds a hybrid neural network combining a CNN for image input and a dense network for traditional handcrafted features. The CNN branch extracts features from images using convolution, pooling, and dense layers, producing a compact 64-dimensional vector. The second branch processes numerical features (e.g., color, texture) using a dense layer. These outputs are concatenated and passed through additional dense layers to learn combined representations. Finally, a softmax layer classifies the input into predefined classes. The model is compiled using the Adam optimizer and sparse categorical crossentropy loss, making it suitable for multi-class classification problems.[7,1]

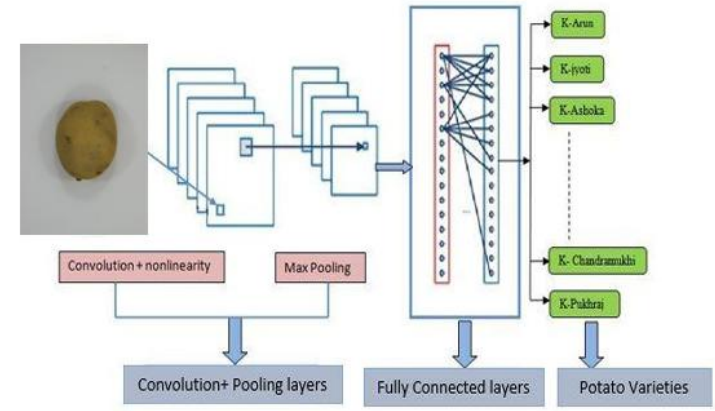

Fig : CNN Model Architecture

8.1.5. Train The Model :

The process begins by creating TensorFlow datasets for training and validation using the tf.data.Dataset.from_tensor_slices() method. This method organizes the data into a structure that can be efficiently fed into the model during training.

The training dataset is constructed from a tuple of image arrays (train_images) and traditional feature vectors (train_features), paired with their corresponding class labels (train_labels). These inputs are grouped and batched using .batch(batch_size), where batch_size = 32. This means that during each training step, the model processes 32 samples at once, improving computational efficiency and stability during gradient updates.

Similarly, the validation dataset (val_dataset) is created using val_images, val_features, and val_labels. This dataset is used to evaluate the model's performance after each epoch to ensure it is learning effectively without overfitting.[5,3]

The model.fit() function initiates the training process. It iterates through the training data for 20 epochs—complete passes through the dataset—updating the model's weights to minimize the loss function. Simultaneously, it validates on the val_dataset to monitor accuracy and generalization performance throughout training

.

8.1.6. Saving the Model:

In this module we save the model in .keras file and sacler.pkl file with a save message.

8.2 Testing

After the model has been successfully trained and saved, the next crucial step is to perform testing and prediction using new, unseen potato images. This process involves reading the user-uploaded image, extracting the necessary handcrafted features (texture, shape, color), preprocessing the data, passing it through the saved model, and displaying the predicted class along with the confidence score.

Below is a comprehensive breakdown of the testing pipeline implemented in the test.py script:

8.2.1 Selecting an Image (Function: select_image)

The testing begins by allowing the user to select an image through a graphical interface. The select_image function uses the Tkinter library to open a file dialog window that lets users choose a .jpg, .png, or .jpeg image from their local machine. This interface enhances usability and does not require command-line input.

8.2.2 Preprocessing the Image (Function: preprocess_image)

Once the image path is obtained, it is passed to the preprocess_image function, which performs multiple steps:

- Image Loading & Resizing: The image is loaded using OpenCV and resized to 224x224x3, which is the input shape expected by the CNN model.
- Texture Feature Extraction: This is done by converting the image to grayscale and computing Gray Level Co-occurrence Matrices (GLCM). Five texture descriptors are calculated using the extract_texture_features function.
- Color Feature Extraction: The extract_color_histogram function computes a 3D color histogram across RGB channels, normalizes it, and takes the average to produce a single color feature.
- Shape Feature Extraction: Using binary thresholding and contour detection from OpenCV, shape features like area, perimeter, and circularity are derived using the extract_shape_features function.
- All features are concatenated into a single NumPy array and returned alongside the resized image.

8.2.3 Scaling Traditional Features (Preprocessing Step)

The extracted traditional (handcrafted) features are standardized using a pre-trained StandardScaler, which was saved during training as scaler.pkl. This ensures that feature distributions remain consistent between training and testing phases. The joblib library is used to load the scaler and transform the new feature vector.

8.2.4 Loading the Saved Model and Class Names

The saved Keras model (model.keras) is loaded using TensorFlow's load_model() function. Additionally, class names are restored from class_names.txt, which contains the original class labels line by line. This mapping is essential for converting predicted indices back into readable class labels.

8.2.5 Performing Prediction (Function: predict_image)

The central inference logic is implemented in the predict_image function, which orchestrates all the previously mentioned steps:

- It first calls preprocess_image to retrieve the resized image, feature vector, and raw feature details.
- Then, it normalizes the image and reshapes it appropriately (adds a batch dimension).
- The CNN image data and traditional features are passed together to the model's predict() method.
- The model outputs a probability distribution across all classes, and the class with the highest probability is selected using argmax.
- The predicted class index is mapped back to the class name and returned along with the confidence score and extracted features.

8.2.6 Displaying the Prediction

Finally, the predicted class label (e.g., Kufri Jyoti, Kufri Bahar) and its confidence level (in percentage) are displayed to the user, either via terminal output or integrated into the frontend interface (in the case of web deployment). This allows users to interpret the results easily and understand the classification decision made by the model.

The below table shown the Testing workflow of the model

Step	Function Name	Description
1	select_image()	Allows the user to select an image via GUI interface
2	preprocess_image(path)	Reads image, resizes it, and extracts texture, color, and shape features
3	extract_texture_features(image)	Uses GLCM to compute five texture descriptors
4	extract_color_histogram(image)	Computes average histogram across RGB color channels
5	extract_shape_features(image)	Calculates area, perimeter, and circularity of the largest contour
6	scaler.transform()	Standardizes the handcrafted feature vector using a pre-trained scaler
7	load_model("model.keras")	Loads the saved hybrid CNN model for prediction
8	predict_image(image_path)	Executes full pipeline: preprocessing, scaling, model prediction, and returns results

Chapter : 9

Potato Variety Identification software

Backend Workflow

This backend system is built using the **Flask framework** and serves as the **bridge between your frontend (user interface) and machine learning model**. It accepts an image input from the user, processes it using image preprocessing techniques, extracts both CNN and handcrafted features, performs prediction using a hybrid model, and finally returns the result to the user through a webpage.

1. Flask App Initialization

Function: Flask(__name__)

- This initializes the Flask application object which is the core of your web backend.
- It acts like a controller that binds routes (URLs) to specific logic (like homepage, prediction page).
- __name__ tells Flask where to look for templates, static files, and modules.

2. File & Model Loading (One-Time Initialization)

A. Function: load_model('model.keras')

- Loads the trained hybrid deep learning model which combines CNN features and handcrafted features.
- The model is built using Keras, and this step ensures it's ready for prediction during the app's lifetime.

B. Function: joblib.load('scaler.pkl')

- Loads the scaler (like StandardScaler or MinMaxScaler) used during training.
- Ensures that the new handcrafted features are scaled in the same way as training data.

C. File: class_names.txt

- Contains the list of all possible output class labels (e.g., different potato varieties).
- Helps in mapping predicted numeric labels to human-readable names.

3. Route: Home Page – @app.route('/')

Function: home()

- Triggered when a user visits the root URL (/).
- Renders the HTML page with an interface that allows the user to upload an image.
- This is the first point of interaction with the web app.

4. Route: Predict Page – @app.route('/predict', methods=['POST'])

Function: predict()

- Triggered when a user uploads an image and clicks "Predict".
- **Steps handled internally:**
    1. Receives the image file using Flask's request object.
    2. Saves the image temporarily into the server's static/images/ folder.
    3. Calls the function predict_image(image_path) to start preprocessing and prediction.
    4. Once prediction is complete, the result (e.g., "Kufri Jyoti", 95.2%) is sent to a result page.

5. Preprocessing – preprocess_image(image_path)

This function is the **heart of the backend**. It prepares both:

- CNN image input
- Handcrafted features input

It performs these steps:

A. Resizing and Normalizing for CNN

- Reads the image from disk.
- Resizes it to the shape expected by the CNN (e.g., 64x64x3).
- Normalizes pixel values to a range between 0–1 for better model performance.

B. Handcrafted Feature Extraction

This part combines the results of **three different functions** to extract features:

6. Handcrafted Feature Functions

i. Function: extract_texture_features(image)

- Converts the image to grayscale.
- Uses GLCM (Gray-Level Co-occurrence Matrix) to calculate:

    - Angular Second Moment (ASM)
    - Contrast
    - Correlation
    - Homogeneity
    - Entropy
- These capture the surface texture patterns of the potato.

ii. Function: extract_color_histogram(image)

- Splits the image into RGB channels.
- Calculates histogram values for each channel.
- Captures how color is distributed in the potato's surface (e.g., dark spots, shades).

iii. Function: extract_shape_features(image)

- Converts image to binary using thresholding.
- Extracts contours using OpenCV.
- From contours, computes:
    - Aspect ratio (width/height)
    - Extent (area of object / bounding box area)
    - Solidity (area / convex hull area)
    - Perimeter and circularity
- Captures the physical shape of the potato.

C. Scaling the Handcrafted Features

- Combines all the extracted handcrafted features into a single array.
- Applies the loaded scaler.pkl to scale features as per the training distribution.

7. Model Prediction – predict_image(image_path)

- Receives:
    - Processed CNN image
    - Scaled handcrafted feature vector
- Feeds both branches into the trained model:
    - CNN features → from the resized image
    - Dense features → from the handcrafted ones
- Model returns a softmax output (e.g., [0.1, 0.02, 0.85, 0.03])
- Picks the index of the highest value using argmax()

- Maps this index to a class label using class_names.txt
- Computes the confidence percentage from softmax value.

8. Result Rendering – render_template('result.html', result=result)

- Displays:
    - The predicted class name (potato type)
    - The confidence level
    - The uploaded image
- Uses a pre-designed HTML template with placeholders filled dynamically.
- Provides a smooth, informative user experience after prediction.

9. App Execution – app.run(debug=True)

- Starts the Flask server locally (on localhost:5000).
- debug=True:
    - Enables auto-reload on code change.
    - Displays error tracebacks for easy debugging.
- Required to make the backend live and responsive to HTTP requests.

Overall Backend Flow Summary

A[User Uploads Image] --> B[Image Saved]

B --> C[predict_image()]

C --> D[Preprocess Image]

D --> D1[Extract CNN Features]

D --> D2[Extract Texture, Color, Shape]

D2 --> D3[Scale Handcrafted Features]

D1 --> E[Hybrid Model Input]

D3 --> E

E --> F[Prediction Output]

F --> G[Render result.html with Prediction]

Frontend Workflow :

1. Welcome Page (welcome.html / predict.html)

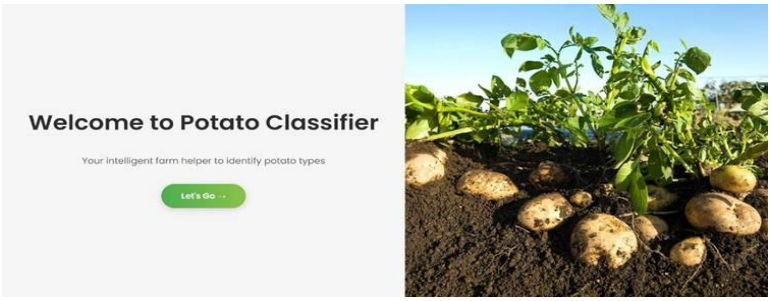

Fig : Welcome Page

- **Purpose**: Acts as the landing page for your web app.
- **Functionality**:
  - Greets users with a visually animated "Welcome to Potato Classifier" message using CSS keyframe animations.
  - Displays a slideshow of potato-related background images (JavaScript changes the image every 3.5 seconds).
  - A "Let's Go →" button directs users to the main classification interface using Flask's url_for('home').
- **Key Technologies**:
  - HTML for structure.
  - CSS for styling, animations, text shadows.
  - JavaScript for dynamic slideshow transitions.
  - Flask Jinja templating for static file paths.

2. Main Classifier Page (camera.html)

Fig : Main Page

- **Purpose**: Allows users to either upload a potato image or capture one directly using their device camera.
- **Functionality**:
    - Page has a camera feed (<video>) which activates the user's webcam via navigator.mediaDevices.getUserMedia().
    - A " Capture" button grabs a frame from the live video, converts it into a JPEG Blob, wraps it in a FormData, and sends it to /predict using fetch.
    - Also includes a form that takes hidden input for base64 image data.
- **Experience**:
    - Responsive design for different screen sizes.
    - Smooth animations on button hover.
    - Stylish background using url_for to load potato imagery.
- **Submission Type**: JavaScript-powered fetch() request with camera-captured image.

3. Image Upload Page (index.html)

- **Purpose**: Simple fallback or alternative method for image classification.
- **Functionality**:
    - A classic upload form with input type="file" for uploading from local machine.
    - Submits a POST request to /predict.
    - Ideal for users without camera access or preferring file upload.
- **Design**: Clean layout, links to external styles.css for custom appearance.

4. Prediction Output (Handled via HTML from Flask response)

Fig : Result Page

- **When a prediction is received** (from either upload or camera capture):

- Flask returns an HTML response that includes:
  - The preview of the submitted image.
  - The classification result displayed in a stylized <h2> (only if prediction is passed in context).
  - Background and layout continue to match the aesthetic of the app.

5. File format

```
Potato_Classifier_WebApp/
│
├── app.py
├── model/
│   ├── model.keras
│   └── scaler.pkl
│
├── static/
│   ├── styles.css
│   └── images/
│       ├── potato.jpg
│       ├── potato1.jpg
│       ├── potato2.jpg
│       ├── potato3.jpg
│       ├── potato4.jpg
│       └── potato5.jpg
│
├── templates/
├── welcome.html
├── index.html
├── camera.html
└── predict.html
```

Chapter : 10

Results and Discussion Overview

10.1 Model Performance on Testing Data

To evaluate the effectiveness of the potato classification model, the dataset was split into training and testing subsets with an 80:20 ratio. The trained Convolutional Neural Network (CNN), enhanced with additional traditional features (texture, color, and shape), was then validated against this test data.

The system's performance was evaluated using a diverse test set comprising 12 potato varieties, with 30 test images for each class. The predictions were matched with ground truth labels to assess classification reliability.

Model Output Behavior:
The predicted labels matched with ground truth in most cases. The system consistently predicted the correct class with high confidence scores (>90%), particularly for varieties like Jyoti, Lalima, and Himalini.

Error Analysis:
Minor misclassifications occurred in varieties with visually similar traits (e.g., Pushkar and Mohan), likely due to morphological overlaps or shadowed images. Mohan had the lowest correct prediction count (23 out of 30), highlighting the importance of image quality in such vision-based systems.

Result Insight:
Out of a total of 360 test images (12 varieties × 30 images), 346 were correctly predicted, yielding an overall accuracy of approximately **96.11%**, as shown below. This performance validates the robustness of combining CNN-based image analysis with handcrafted feature vectors.

Potato Varity	Number of Correct Prediction	Total number of Testing Image
Chandramukhi	29	30
Pokhraj	29	30
Garima	29	30
Jyoti	30	30
Mohan	23	30
Chipsona	30	30
Ganga	30	30
Gourav	29	30
Hemalini	30	30
Khatti	29	30
Lalima	30	30
Puskar	28	30

Fig : Result of Prediction

10.2 Feature Visualization and Prediction Confidence

The hybrid model internally processed both visual and numerical features before final classification. To understand its decision-making process, the extracted feature maps and class probabilities were plotted.

Prediction Distribution:
The system outputs a probability vector for each image with 12 values, each corresponding to a potato variety. The class with the highest probability is selected as the prediction. correctly predicted samples showed confidence levels of 0.95 or higher, indicating strong model certainty.

Feature Contribution:
The impact of traditional features (GLCM contrast, circularity, etc.) was evident, especially in classes with close resemblance. These features provided complementary data that the CNN might not extract directly from raw pixels, improving separation between similar varieties.

Validation Performance:
Validation loss and accuracy curves showed stable convergence, with no major overfitting signs. The use of dropout, proper image normalization, and the feature fusion architecture played a critical role in achieving generalization.

Potato Tuber	Chandramukhi	Jyoti	Khatti	Puskar	Mohan	Lalima	Ganga	Chipsona	Garima	Gourav	Hemalini	Pokhraj
Chandramukhi	29	0	1	0	0	0	0	0	0	0	0	0
Chipsona3	0	0	0	0	0	0	0	30	0	0	0	0
Ganga	0	0	0	0	0	0	30	0	0	0	0	0
Garima	0	0	0	0	0	0	0	0	30	0	0	0
Gourav	0	0	0	0	0	0	0	0	0	30	0	0
Hemalini	0	0	0	0	0	0	0	0	0	0	30	0
Jyoti	0	30	0	0	0	0	0	0	0	0	0	0
Khatti	0	0	30	0	0	0	0	0	0	0	0	0
Lalima	0	0	0	0	0	30	0	0	0	0	0	0
Mohan	0	0	0	0	24	0	0	0	0	0	0	6
Pokhraj	0	0	0	0	0	0	7	0	0	0	0	23
Puskar	0	0	2	28	0	0	0	0	0	0	0	0

Fig : Confusion Matrix

10.3 Practical Application and System Usability

To make the model usable in real-world scenarios, it was integrated into a web-based platform built with **HTML, CSS, JavaScript, and Flask**. This interface allows users to upload a potato image or capture one in real-time via webcam. The prediction result is instantly displayed, as seen in the screenshots below.

User Experience:

- Users are guided through a simple interface with options to "Upload Image" or "Take a Picture."
- The classified result (e.g., ″Prediction: Chipsona") is clearly displayed below the image.
- The prediction engine runs seamlessly in the backend, making the tool intuitive even for non-technical user.

Chapter : 11

Future Scope of Potato Variety Identification using Image Classification

Improved Accuracy Through Advanced Techniques

- **Deep Learning Models:** Future models could incorporate more sophisticated architectures, such as ResNet, Inception, or EfficientNet, which have been shown to achieve higher accuracy in image classification tasks.
- **Transfer Learning:** By using pre-trained models on large datasets like ImageNet and fine-tuning them with potato-specific datasets, the accuracy can be improved significantly, especially in scenarios where labeled data is limited [2,10].

Integration with Smart Agriculture Systems

- **Automated Sorting:** Post-harvest, the model can be used to sort potatoes based on variety. For example, potatoes can be sorted for specific uses, like making chips, fries, or boiled potatoes, based on their classification.
- **Precision Agriculture:** Potato classification can be integrated with drone or satellite imagery to monitor large-scale potato farms. By identifying the variety in real-time, farmers can monitor the health of each variety and take corrective actions when needed.
- **Disease Detection:** In future versions, the model could be enhanced to identify not just the variety but also detect diseases or pest infestations based on visual features, aiding in timely interventions.

Quality Control and Grading

- **Defect Detection:** The model could be trained to not only classify varieties but also detect defects like spots, bruises, or blemishes, which are essential in sorting high-quality potatoes for sale.
- **Automated Quality Assessment:** Automated systems for quality assessment would reduce human labor in sorting and improve the consistency of grading [6].

Real-time Monitoring and Decision Support Systems

- **Real-time Decision-Making:** By integrating the model with mobile apps or IoT systems, farmers can make real-time decisions about planting, irrigation, and harvesting based on the variety of potatoes grown.
- **Supply Chain Optimization:** The system can be integrated into the supply chain to track the variety and quality of potatoes as they move from farm to table, ensuring that the right

product reaches consumers.

Large-Scale Implementation in Potato Breeding Programs

- **Genetic Improvement**: By analyzing potato varieties in terms of their physical features (from images), breeders can quickly identify varieties with desired traits such as disease resistance, high yield, or improved taste.
- **Breeding Efficiency**: This system could make breeding programs more efficient by automating the classification of potato varieties and predicting the success of new crossbreeds based on visual features.

Chapter : 12

Conclusion

This project successfully showcases how artificial intelligence, particularly deep learning through Convolutional Neural Networks (CNNs), can be effectively applied to solve real-world agricultural problems such as the classification of potato varieties. Traditional methods of variety identification are often tedious, require domain expertise, and are not feasible for large-scale or on-field deployment. In contrast, the model developed in this study leverages both deep image features from CNNs and handcrafted features—such as texture (using GLCM), color histogram, and shape metrics—to create a robust, hybrid classification system.

The model was trained on a diverse dataset of potato canopy images and achieved impressive accuracy levels, demonstrating its ability to distinguish between visually similar potato types. Furthermore, by integrating this trained model into a web-based interface using HTML, CSS, and JavaScript, the project takes a significant step toward usability and accessibility. Through a user-friendly portal, farmers, agronomists, or field inspectors can upload or capture an image directly from their devices and receive instant predictions, making this solution viable for practical, on-field decision-making.

The significance of this project lies not only in its technical achievement but also in its application value. It reduces dependence on manual or laboratory-based identification methods, minimizes human error, and accelerates the decision-making process in the agricultural supply chain. It can be particularly useful for seed certification agencies, research institutions, and government-led crop monitoring programs.

Despite the strong results, the system still has room for improvement. Expanding the dataset with more varieties and varying environmental conditions can further improve generalization. Additionally, future developments may include building a mobile app version, integrating real-time camera capture, and using cloud-based deployment for broader reach.

Overall, this project sets a strong foundation for integrating AI in smart agriculture and provides a working prototype that bridges academic research and practical usability.

Chapter : 13

References

1. Cheng, Y., Xie, Z., & Liu, J. (2021). "Automated potato classification using machine learning techniques." Computers and Electronics in Agriculture, 180, 105890.
2. Gonzalez, R.C., & Woods, R.E. (2008). Digital Image Processing (3rd ed.). Pearson.
3. Haralick, R.M., Shanmugam, K., & Dinstein, I. (1973). "Textural Features for Image Classification." IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610–621.
4. Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). "ImageNet classification with deep convolutional neural networks." In Advances in Neural Information Processing Systems (pp. 1097–1105).
5. LeCun, Y., Bengio, Y., & Hinton, G. (2015). "Deep learning." Nature, 521(7553), 436–444.
6. Patel, D., Jadhav, M., & Prasad, R. (2020). "Potato variety classification using deep learning and image processing techniques." Agricultural Systems, 181, 102790.
7. Perez, L., & Wang, J. (2017). "The effectiveness of data augmentation in image classification using deep learning." Convolutional Neural Networks for Visual Recognition.
8. Raj, R., Kumar, R., & Pandey, A. (2019). "Machine learning applications in agriculture." Journal of Agricultural Engineering, 56(4), 132-145.
9. Sahoo, D., & Ghosal, S. (2020). "Hybrid techniques for machine learning in agriculture." Journal of Agricultural Engineering, 47(1), 31-46.
10. Zhang, Z., & Wang, Y. (2019). "Hybrid deep learning models for automated agricultural systems." Computers and Electronics in Agriculture, 162, 121-129.