"Smart CNN Based Indian Currency Verification System"

Roopa B S
Computer Application
St. Joseph's First Grade College

ABSTRACT

Counterfeit currency poses a serious threat to the economic stability and security of a nation. The circulation of fake Indian currency notes leads to financial losses and undermines public trust in monetary systems. To address this issue, this project proposes an Identification System for Fake Indian Currency using Convolutional Neural Networks (CNN), leveraging the power of deep learning for accurate and automated detection.

The proposed system utilizes a CNN model trained on a dataset consisting of genuine and counterfeit Indian currency note images of various denominations. The model automatically learns discriminative features such as texture patterns, color distribution, watermark regions, security threads, micro-lettering, and latent image areas, which are difficult to replicate accurately in fake notes. Image preprocessing techniques including resizing, grayscale conversion, normalization, and noise removal are applied to enhance feature extraction and classification accuracy.

The CNN architecture comprises multiple convolutional and pooling layers followed by fully connected layers that classify the input currency image as either genuine or fake. The system is robust to variations in lighting, orientation, and partial occlusion of notes. Experimental results indicate that the CNN-based approach achieves high accuracy compared to traditional rule-based and feature-engineering methods.

This system can be effectively deployed in banking institutions, automated teller machines, retail outlets, and mobile applications to assist in real-time counterfeit detection, thereby contributing to enhanced financial security and fraud prevention.

CHAPTER 1

INTRODUCTION

1.1 DOMAIN

This project belongs to the field of artificial intelligence and computer vision, with a focus on financial security and fraud prevention. It uses convolutional neural networks to analyze images of Indian currency notes and identify whether they are genuine or counterfeit. The system automatically learns visual characteristics such as texture, color distribution, patterns, and security features from the input images.

By leveraging deep learning, the proposed approach reduces dependency on manual inspection and traditional rule-based methods. It is capable of handling variations in lighting, orientation, and image quality, making it suitable for real-world use. The system can be applied in banks, retail environments, and automated currency handling systems to support effective counterfeit detection.

1.2 Introduction

The Fake Currency Detection System is an Artificial Intelligence based application developed to identify whether a currency note is genuine or counterfeit using deep learning techniques. Counterfeit currency has become a major issue affecting financial institutions, businesses, and individuals. Manual detection of fake currency is difficult, time-consuming, and requires expert knowledge. Therefore, an automated intelligent system is required to detect fake currency accurately and efficiently.

The proposed system uses deep learning algorithms trained on a dataset containing multiple denominations of currency notes such as ₹10, ₹20, ₹50, ₹100, ₹500, and ₹2000. The system processes the input image of a currency note, extracts important security features, and classifies the note as fake or original. The system is also capable of detecting damaged currency notes and performing prediction based on available features.

The application includes advanced functionalities such as micro text recognition, feature-based detection explanation, multilingual voice alert system, AI help assistant, pie chart visualization for prediction accuracy, and downloadable report generation. The deep learning model is trained using CNN and object detection techniques to analyze currency patterns, security threads, watermark regions, and denomination structures.

The system aims to provide an efficient, accurate, and user-friendly solution for fake currency detection that can be used in banks, shops, ATMs, and public places.

1.3 Existing System

The existing system for detecting fake Indian currency mainly relies on manual inspection and traditional image processing or hardware-based techniques. Manual methods involve trained personnel examining currency notes using visual cues such as watermarks, security threads, and color patterns. In some cases, specialized machines like UV scanners and currency counters with built-in detectors are used. These systems follow predefined rules and limited feature checks to identify counterfeit notes.

Disadvantages

- Manual inspection is time-consuming and depends heavily on human expertise.
- High chances of human error, especially with high-quality counterfeit notes.
- Specialized hardware devices are expensive and not easily accessible everywhere.
- Traditional rule-based systems lack adaptability to new and improved counterfeit techniques.
- Limited accuracy under varying lighting conditions and note orientations.

1.4 Objectives

The main objective of the project titled “Identification of Fake Indian Currency using Convolutional Neural Network” is to develop an automated system that can accurately distinguish between genuine and counterfeit Indian currency notes using deep learning techniques. The system focuses on implementing Convolutional Neural Networks (CNN) to analyze images of currency notes and identify key security features such as watermarks, security threads, and micro-text. It aims to reduce the dependency on manual verification, which is time-consuming and prone to human error, by providing a fast and efficient detection mechanism. Additionally, the project is designed to offer a user-friendly interface for uploading currency images and viewing results, along with features like prediction accuracy visualization and report generation. The system also aims to be scalable, allowing future improvements such as increased dataset size, enhanced model performance, and support for real-time and mobile-based currency detection.

1.5 Proposed System

The proposed system introduces an intelligent and automated solution for counterfeit currency detection using convolutional neural networks. The system supports live scanning of currency notes through a camera, enabling real-time identification of genuine and fake notes. It is designed to handle multiple currencies, making it suitable for use in diverse financial environments. Advanced image analysis techniques are used to recognize fine details such as holograms and texture patterns, which are crucial security features in modern currency notes.

The system processes the scanned image using a trained CNN model that extracts deep visual features and classifies the currency accurately. In addition to detection, the system generates a downloadable report that includes detection results, confidence scores, and pie charts showing accuracy percentages for genuine and counterfeit classifications. These visual analytics help users easily understand system performance and detection outcomes. The proposed system is scalable, user-friendly, and suitable for real-time deployment.

Advantages

1. Enables real-time currency verification through live scanning.
2. Supports detection of multiple currencies and denominations.
3. Accurately recognizes holograms and texture-based security features.
4. Reduces human intervention and minimizes detection errors.
5. Provides downloadable reports with graphical accuracy analysis.
6. Adaptable to new counterfeit patterns through model retraining.
7. Cost-effective and suitable for banks, retail stores, and financial institutions.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

A literature review is a comprehensive summary and analysis of existing research and literature on a particular topic, serving multiple purposes. It surveys the field by providing an overview of what has been studied, identifying key themes, findings, methodologies, and gaps. By contextualizing new research within the existing body of knowledge, it shows how the new study builds on or diverges from previous work. Additionally, it evaluates sources, critically assessing the strengths and weaknesses of previous studies, highlighting methodological rigor, and the reliability of findings. Furthermore, it identifies trends and patterns in the research, offering insights into how the understanding of the topic has evolved over time. A literature review typically includes an introduction explaining the purpose and scope, a body organized by themes, methodologies, or chronological order that summarizes and synthesizes findings from various sources, and a conclusion that summarizes the main findings and often suggests areas for future research. In academic writing, literature reviews are essential components of theses, dissertations, and research articles, providing a foundation for new research by ensuring the researcher is well-informed about the existing knowledge and debates in their field.

2.2 Survey Papers

1. Title: “Fake Indian Currency Recognition Using CNN And Mobile Net Algorithm”

Authors: S. Vinay KumarS. Muni Kumar

Year:2025

This paper presents a system that uses Convolutional Neural Networks (CNN) and MobileNet to detect fake Indian currency. It focuses on accurately distinguishing real and counterfeit notes using deep learning, offering a fast and reliable solution for real-time applications.

2. Title: “ Original vs Counterfeit Indian Currency Detection”

Authors: Anushka Kulkarni, Prachi Kedar, Aishwarya Pupala, Priyanka Shingane.

Year: 2020

This paper uses image processing algorithms to detect fake Indian currency. ORB algorithm is found to be the most efficient, giving high accuracy and fast real-time detection**.**

3.Title: "Automated Indian Currency Verification Using Convolutional Neural Networks and Image Processing"

Authors: Y. Giri Babu, K. Jaya Krishna

Year:2025

This paper uses CNN and image processing techniques to classify currency as real or fake. It analyzes security features and suggests improved deep learning models for better accuracy in future.

4. Title: "Fake Banknote Recognition Using Deep Learning"

Authors: César G. Pachón, Dora M. Ballesteros, Diego Renza**.**

Year: 2021

This paper proposes a deep learning-based system using CNNs to detect fake banknotes. It compares transfer learning models with a custom CNN and finds that while ResNet18 gives the highest accuracy, the custom model is faster and more efficient, making it suitable for real-time counterfeit detection**.**

5.Title: "Identification of Indian Currency Note Authenticity Using Ensemble Learning and Image Processing"

Authors: Vivek Sharan, Amandeep Kaur, Parvinder Singh

Year: 2025

This paper proposes a method to detect fake Indian currency using image processing and a voting ensemble model (SVM, KNN, DT). It uses processed currency images with different conditions and achieves about 81.6% accuracy, performing better than individual models.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 Introduction

The Fake Currency Detection System is an Artificial Intelligence based application developed to identify whether a currency note is genuine or counterfeit using deep learning techniques. Counterfeit currency has become a major issue affecting financial institutions, businesses, and individuals. Manual detection of fake currency is difficult, time-consuming, and requires expert knowledge. Therefore, an automated intelligent system is required to detect fake currency accurately and efficiently.

The proposed system uses deep learning algorithms trained on a dataset containing multiple denominations of currency notes such as ₹10, ₹20, ₹50, ₹100, ₹500, and ₹2000. The system processes the input image of a currency note, extracts important security features, and classifies the note as fake or original. The system is also capable of detecting damaged currency notes and performing prediction based on available features.

The application includes advanced functionalities such as micro text recognition, feature-based detection explanation, multilingual voice alert system, AI help assistant, pie chart visualization for prediction accuracy, and downloadable report generation. The deep learning model is trained using CNN and object detection techniques to analyze currency patterns, security threads, watermark regions, and denomination structures.

The system aims to provide an efficient, accurate, and user-friendly solution for fake currency detection that can be used in banks, shops, ATMs, and public places.

3.2 Functional Requirements Specifications

1.Currency Image Input
The system should allow users to provide currency input either through live camera scanning or by uploading an image from local storage. The captured image should be sent to the backend for processing and classification.

2.Multiple Currency Detection
The system should detect multiple currency denominations including ₹10, ₹20, ₹50, ₹100, ₹500, and ₹2000. The deep learning model should first identify the denomination and then perform fake vs original classification.

3.Fake vs Original Classification
The system should classify the currency note as fake or original using a trained deep learning model. The classification should be based on detected currency features.

4.Micro Text Recognition
The system should detect micro text present in the currency note using OCR techniques. The recognized text should be compared with original micro text patterns to assist in classification.

5.Security Feature Detection
The system should analyze security features present in currency notes such as watermark, security thread, denomination number, RBI emblem, and portrait alignment. These features should be used as input to the classification model.

6.Damaged Currency Detection
The system should detect whether the currency note is damaged, torn, folded, or partially visible. The model should still attempt classification using available visible features.

7.Feature-Based Prediction Explanation
The system should display the specific features used for prediction such as micro text detected, watermark visible, denomination recognized, and damaged region detected.

8.Accuracy Prediction
The system should calculate the prediction confidence score and display the accuracy percentage.

9.Pie Chart Visualization
The system should display prediction confidence using a pie chart showing fake vs original probability.

10.Voice Alert System
The system should provide voice alerts indicating whether the currency note is fake or original.

11.Multi-Language Support
The system should support multiple languages for voice alerts such as English, Hindi, and regional languages.

12.AI Agent Help Box
The system should provide an AI-based help assistant to guide users and explain detection results.

13.Downloadable Report
The system should generate a downloadable report containing:

- Currency image
- Prediction result
- Accuracy percentage
- Detected features
- Date and time

3.3 Non-Functional Requirements

1.Performance
The system should provide fast and efficient processing of currency images. The time taken from image input to prediction output should be minimal to support real-time detection. The deep learning model should be optimized to reduce latency during inference. Image preprocessing, feature extraction, and classification should be executed efficiently to avoid delays. The system should also handle multiple requests without performance degradation. When using live camera scanning, the system should process frames continuously and provide prediction within a few seconds.

2.Accuracy
The system should provide high accuracy in detecting fake and original currency notes. The deep learning model should be trained using a diverse dataset including multiple denominations, fake notes, and damaged notes. The system should correctly identify currency features such as watermark, micro text, and security thread. The prediction confidence score should reflect the reliability of the model. The system should maintain consistent accuracy even under variations such as lighting conditions, partial damage, and rotated images.

3.Reliability
The system should provide consistent results when the same currency image is processed multiple times. The detection should work even when the currency note is slightly damaged, folded, or partially visible. The system should not crash during image processing or model prediction. In case of low confidence prediction, the system should notify the user that the note cannot be verified. The application should continue functioning even if one module fails, such as voice alert or report generation.

4.Usability
The system interface should be user-friendly and easy to operate. Users should be able to scan or upload currency images without technical knowledge. The UI should clearly display prediction results, accuracy score, and detected features. The voice alert system should guide users by announcing whether the note is fake or original. Multi-language support should help users understand the output in their preferred language. The system should provide simple buttons for scanning, uploading, and downloading reports.

5.Scalability
The system should be designed in a scalable manner so that additional currency denominations can be added in the future. The dataset can be expanded with new training samples to improve accuracy. The model should allow retraining with new currency features. The system should also support adding more languages to the voice alert module. Additional modules such as UV detection and hologram detection can be integrated in future upgrades.

6.Security

The system should securely handle uploaded currency images. The images should not be stored permanently unless required for training. Unauthorized users should not access system data or model files. The backend API should validate input requests before processing. The system should prevent malicious file uploads. The downloadable reports should contain only detection-related information and should not expose system internals.

7.Maintainability

The system should be easy to maintain and update. The deep learning model should be modular so that it can be retrained when new dataset is available. The preprocessing and feature extraction modules should be separated for easy modification. The code structure should be organized and documented. The system should allow adding new features such as improved OCR detection or additional security checks without affecting existing modules.

8.Portability

The system should run on different platforms such as Windows, Linux, and macOS. The application should also be deployable as a web-based system accessible from browsers. The system should support running on both local machines and cloud environments. The deep learning model should be exportable and reusable across platforms. The system should also support integration with mobile applications.

9.Availability

The system should be available whenever required for currency detection. The application should start quickly and allow users to perform detection without delay. The backend API should remain active during operation. The system should recover automatically from minor failures. The detection modules should remain accessible during runtime.

3.5 System Requirements

Hardware Requirements

- **Processor**: Intel i5 2.53GHz
- **Hard Disk** : 30GB
- **Ram** : 8 GB or above
- **Display** : 1366x768 is required for UI display.

Software Requirements

- **Operating system**: Windows, Linux, or macOS
- **Coding Language**: Python
- **Framework** : Flask or FastAPI for backend API.
- **Version**:3.6 & above
- **IDE**: Visual Studio Code
- **Frontend**: HTML, CSS, JS or React
- **Database**: MySQL
- **Libraries** : OpenCV for image preprocessing
  TensorFlow / PyTorch for deep learning
  NumPy for numerical computation
  Matplotlib for pie chart visualization
  Tesseract OCR for micro text detection

Python

Python is used as the core programming language in this fake currency detection system due to its simplicity and strong support for machine learning and image processing. In this project, various Python libraries are used to perform specific tasks. OpenCV is used for image preprocessing techniques such as noise reduction, resizing, and feature

enhancement of currency images. TensorFlow or PyTorch is used to build and train the Convolutional Neural Network (CNN) model for classifying currency as genuine or counterfeit. NumPy is used for efficient numerical computations and handling image data in array format. Matplotlib is used to visualize results, such as generating pie charts for analysis. Additionally, Tesseract OCR is integrated to detect and extract micro text from currency notes, which helps in verifying authenticity. Overall, Python enables smooth integration of all these libraries, making the system efficient and easy to develop.

IDE: Visual Studio Code

Visual Studio Code (VS Code) is the chosen Integrated Development Environment (IDE) for developing the *Fake Currency Detection System*. It is a lightweight, customizable, and feature-rich code editor that supports multiple programming languages, including Python. VS Code provides an efficient environment for both frontend and backend development.

It offers key features such as intelligent code completion, debugging tools, built-in Git support, and a wide range of extensions. Extensions like the Python Extension and Pylance improve syntax highlighting, IntelliSense, and error detection, which help in writing accurate and efficient code. These features are useful for implementing machine learning models and image processing tasks in the project.

VS Code also includes an integrated terminal that allows running scripts, installing libraries, and testing models within the editor. Its cross-platform support and user-friendly interface make development smooth and organized. Overall, VS Code enhances productivity and simplifies the development of the system.

Frontend (HTML, CSS, JavaScript/React)

The frontend of the Fake Currency Detection System is developed using web technologies such as HTML, CSS, and JavaScript, or modern frameworks like React.

HTML (HyperText Markup Language) is used to structure the web pages and create the user interface components.

CSS (Cascading Style Sheets) is used for designing and styling the application, ensuring a clean and user-friendly interface.

JavaScript is used to add interactivity, handle user inputs, and communicate with the backend API.

React (optional) is a modern JavaScript library used to build dynamic and responsive user interfaces with reusable components.

The frontend allows users to upload images of currency notes, view detection results, receive alerts, visualize prediction accuracy through charts, and download reports. It ensures a smooth and interactive user experience.

Database:MySQL

MySQL is used as the database management system for storing and managing application data. It helps in maintaining records such as user details, uploaded currency images, prediction results, and generated reports.

The database ensures efficient data storage, retrieval, and security. It also supports structured query operations, enabling the system to handle large amounts of data reliably and maintain consistency across the application.

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

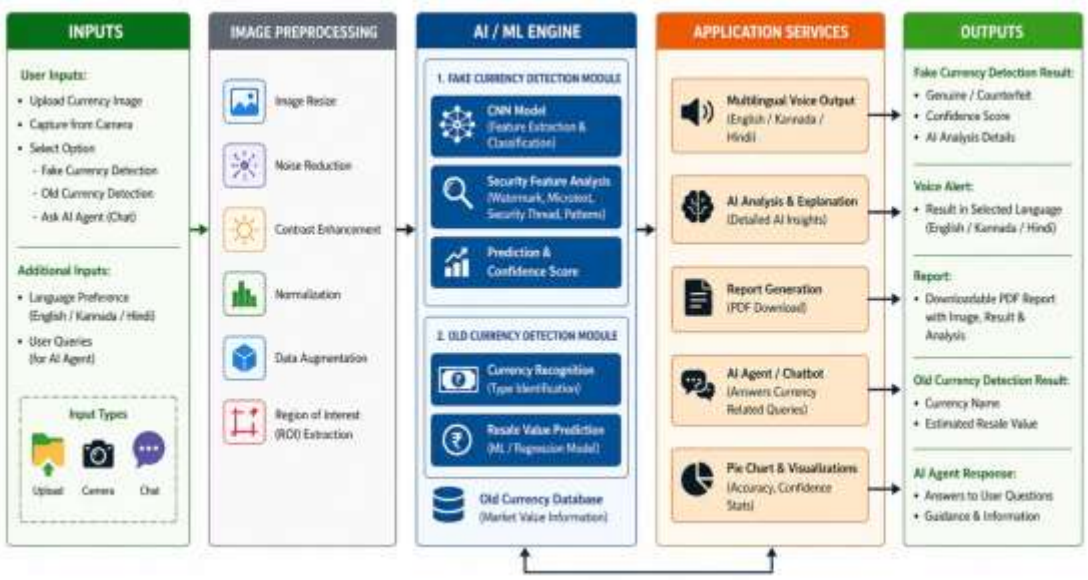
System design describes the overall architecture and working flow of the Fake Currency Detection System. It explains how different components interact with each other to process the currency image and generate prediction results. The system is designed using a modular architecture where each module performs a specific task such as image acquisition, preprocessing, feature extraction, classification, and result display.

The proposed system uses deep learning techniques to analyze currency images and detect whether the note is fake or original. The input currency image is captured using a camera or uploaded by the user. The image is then processed

using image preprocessing techniques to enhance quality and remove noise. After preprocessing, important features such as watermark, micro text, denomination, and security thread are extracted. These features are passed to a trained deep learning model which performs classification. The output is then displayed to the user along with prediction accuracy and feature explanation.

The system also includes additional modules such as voice alert system, pie chart visualization, downloadable report generation, and AI help assistant. These modules improve usability and provide detailed insights into the detection process. The architecture is designed to support real-time detection and future scalability.

4.2 SYSTEM ARCHITECTURE



4.2 System Architecture

System Architecture Explanation

1. Fake Currency Dataset

The system starts with collecting a Fake Currency Dataset containing:

- Fake Notes
- Damaged Notes
- Original Notes
- Multiple denominations (₹10, ₹20, ₹50, ₹100, ₹200, ₹500)

This dataset is used to train the model to learn differences between fake and real currency notes.

The dataset includes:

- Different lighting conditions
- Different angles
- Folded and damaged notes
- Real-world captured images

This improves model accuracy and robustness.

2. Data Preprocessing

Before training the model, the dataset undergoes preprocessing. This stage prepares images for better learning. Preprocessing steps include:

- Image Resize (OpenCV)

All images are resized to a fixed dimension (example: 224×224) so the model receives uniform input.

- Noise Removal

Unwanted background noise is removed using filtering techniques such as:

Gaussian Blur
Median Filter
Bilateral Filter

This improves feature clarity.

- Data Augmentation

To increase dataset size and improve accuracy, augmentation techniques are applied:

Rotation
Flipping
Zooming
Brightness adjustment
Cropping

This helps the model generalize better.

- Train-Test Split

The dataset is divided into:

Training Data (80%)
Testing Data (20%)

Training data is used to train the model, while testing data is used to validate performance.

3. CNN / YOLO Classification

After preprocessing, the images are passed into a deep learning model such as:

CNN (Convolutional Neural Network)
YOLO (You Only Look Once)

The model performs:

- Currency detection

- Feature learning
- Fake vs Original classification

The model learns patterns such as:

- Color differences
- Texture variations
- Security features
- Serial number style

4. Model Validation

The trained model is validated using test data. This stage checks:

- Model accuracy
- Precision
- Recall
- F1-score

Validation ensures the model is working correctly before deployment.

5. Build Model

After validation, the final trained model is built. This model is saved and used for real-time prediction.

The trained model contains:

- Learned weights
- Feature patterns
- Classification rules

This model is then used to detect fake currency.

6. Feature Extraction

When a new input image is given, feature extraction is performed. The system extracts:

- Currency Detection
- Identifies the denomination of the currency.
- Micro Text Recognition
- Detects tiny printed texts present in original notes.
- Security Feature Detection

Checks for:

- Watermark
- Security thread
- Latent image
- RBI logo
- Damaged Note Detection

Identifies:

- Torn notes
- Folded notes
- Scratched notes

These features help in classification.

7. Class Label Prediction

After extracting features, the trained model predicts the class label:

- Fake
- Original

The prediction is based on learned patterns from training data.

8. Accuracy Calculation & Confidence Score

The system calculates:

- Prediction Accuracy
- Confidence Score

Example output:
Fake Note – 96.5% Confidence
Original Note – 98.2% Confidence

This helps users understand prediction reliability.

9. Output Result

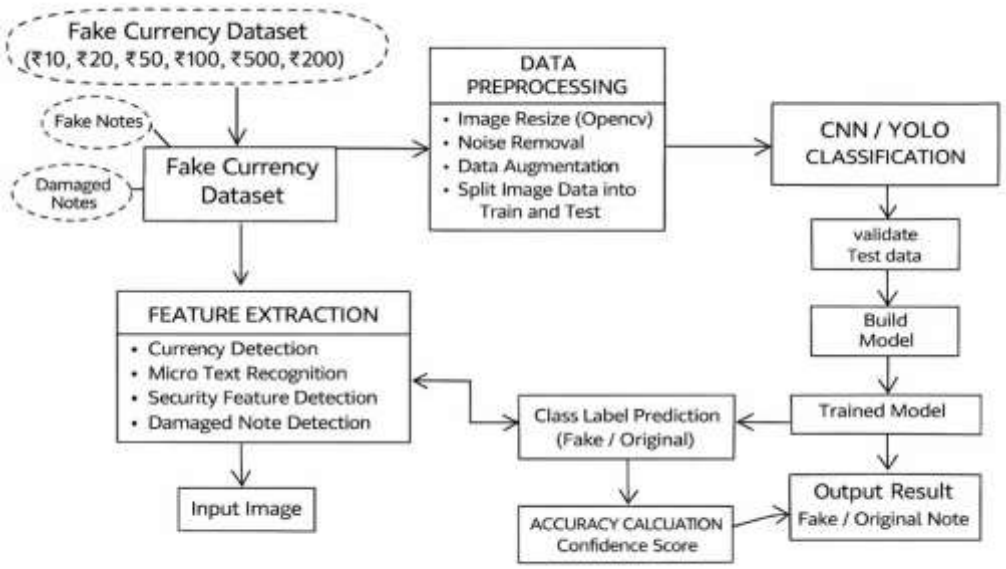
Finally, the system displays the output:

Fake / Original Note

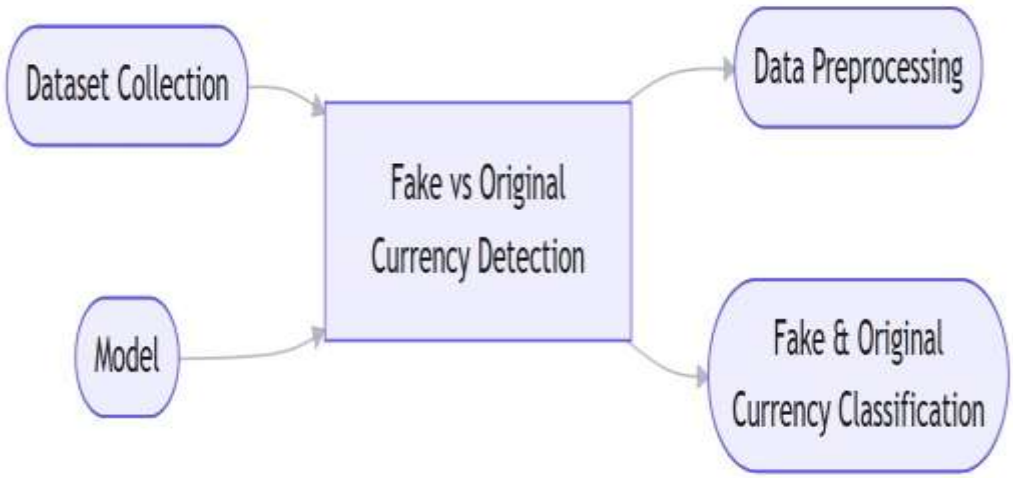
Along with:

- Confidence Score
- Detected denomination
- Feature analysis (optional)

4.3 Block Diagram



4.3 Block Diagram
4.4 Data Flow Diagram



4.4 Level 0 DFD

Level 0 DFD Explanation

- Dataset Collection

Currency images of different denominations are collected. The dataset contains fake notes, original notes, and damaged notes. These images are used for training the detection system.

- Data Preprocessing

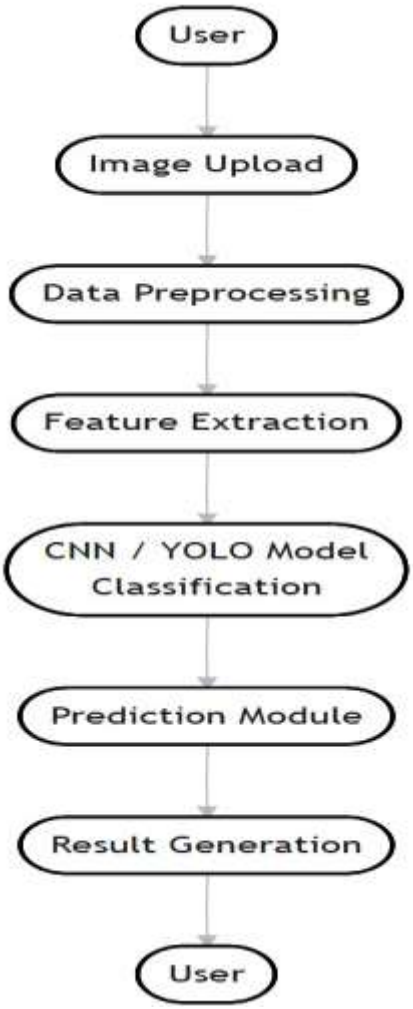
The collected images are preprocessed using resizing, noise removal, normalization, and augmentation. This step improves image quality and prepares the dataset for model training.

- Model (CNN / YOLO)

The preprocessed data is passed to the deep learning model. The CNN/YOLO model learns features such as watermark, texture, serial number, and security thread.

- Fake and Original Currency Classification

The trained model classifies the currency note as Fake or Original. The system then outputs the classification result with confidence score.



4.4.1 Level 1 DFD

Level 1 DFD Explanation

1. Image Upload
User uploads the currency note image into the system.

2. Data Preprocessing
- Image resize
- Noise removal
- Normalization
- Enhancement

3. Feature Extraction
System extracts:

- Security thread

- Watermark
- Micro text
- Serial number
- Currency denomination

4. CNN / YOLO Model Classification
Deep learning model analyzes extracted features and identifies fake patterns.

5. Prediction Module
Model predicts:

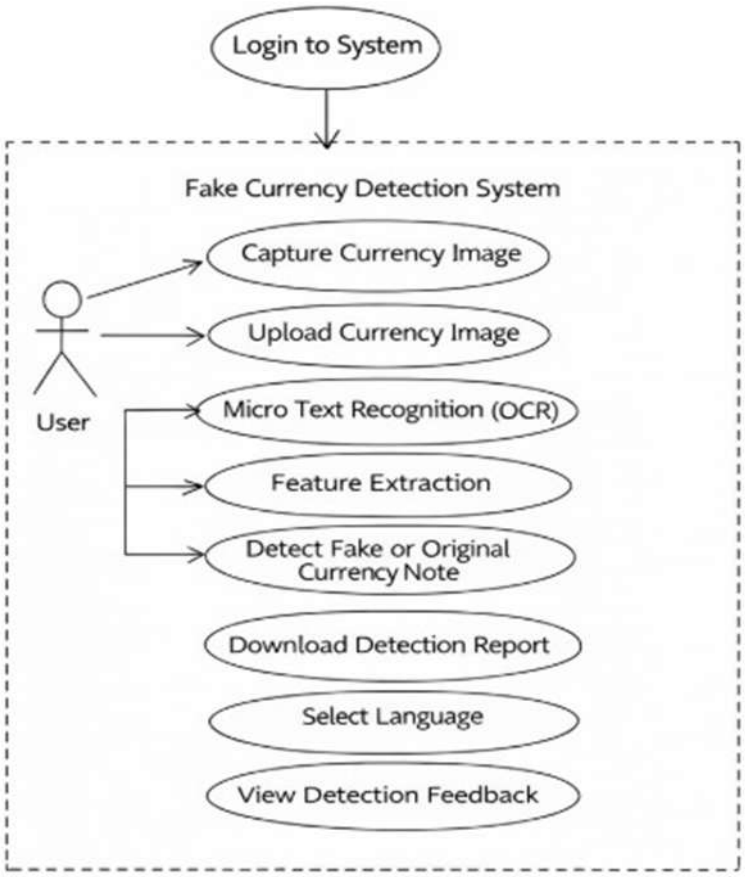
- Fake Currency
- Original Currency

6. Result Generation
System generates:

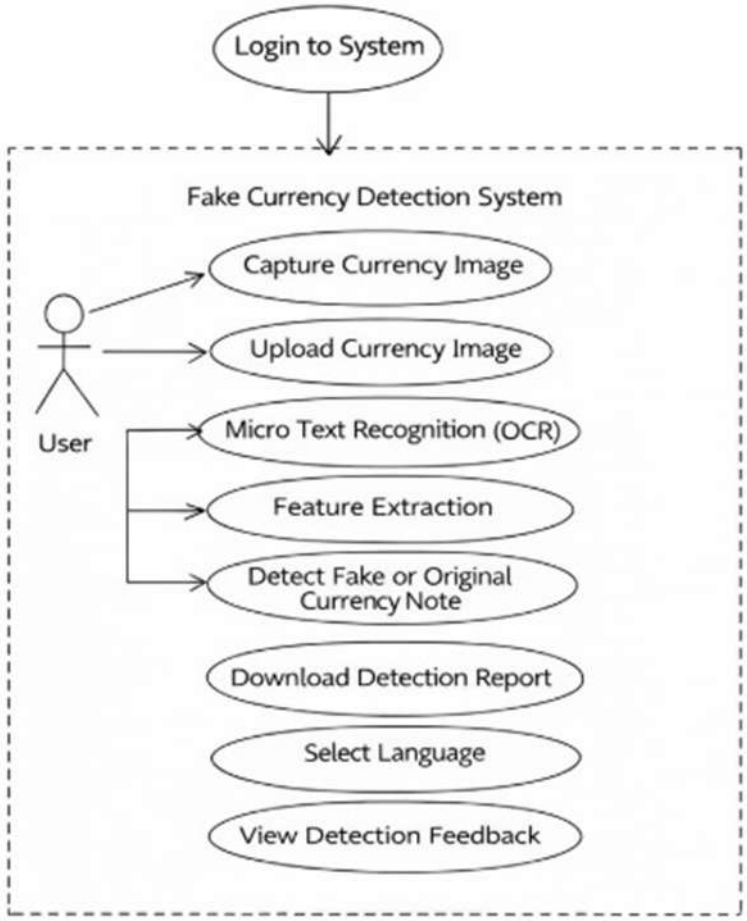
- Fake / Original result
- Confidence score
- Denomination

7. Output to User
Final result displayed to the user.

4.5 Use case diagram



4.5 Use case diagram



The Use Case Diagram represents the interaction between the user and the Fake Currency Detection System. It shows the functionalities provided by the system and how the user interacts with them. The main actor in the system is the user, who performs various operations such as uploading currency images, detecting fake notes, and downloading reports.

The system begins with the user accessing the application. The user first logs into the system to start using the fake currency detection features. After login, the user can capture the currency image using a camera or upload an existing currency image from local storage. These two use cases represent the input stage of the system.

Once the image is provided, the system performs Micro Text Recognition using OCR techniques. This process extracts tiny text printed on currency notes, which is an important security feature. After OCR, the system performs feature extraction. In this step, the system identifies currency note features such as watermark, denomination number, RBI emblem, Mahatma Gandhi portrait, and security thread.

After extracting the features, the system performs fake or original currency detection. The deep learning model analyzes the extracted features and classifies the currency note as fake or genuine. The detection result is displayed to the user along with prediction confidence.

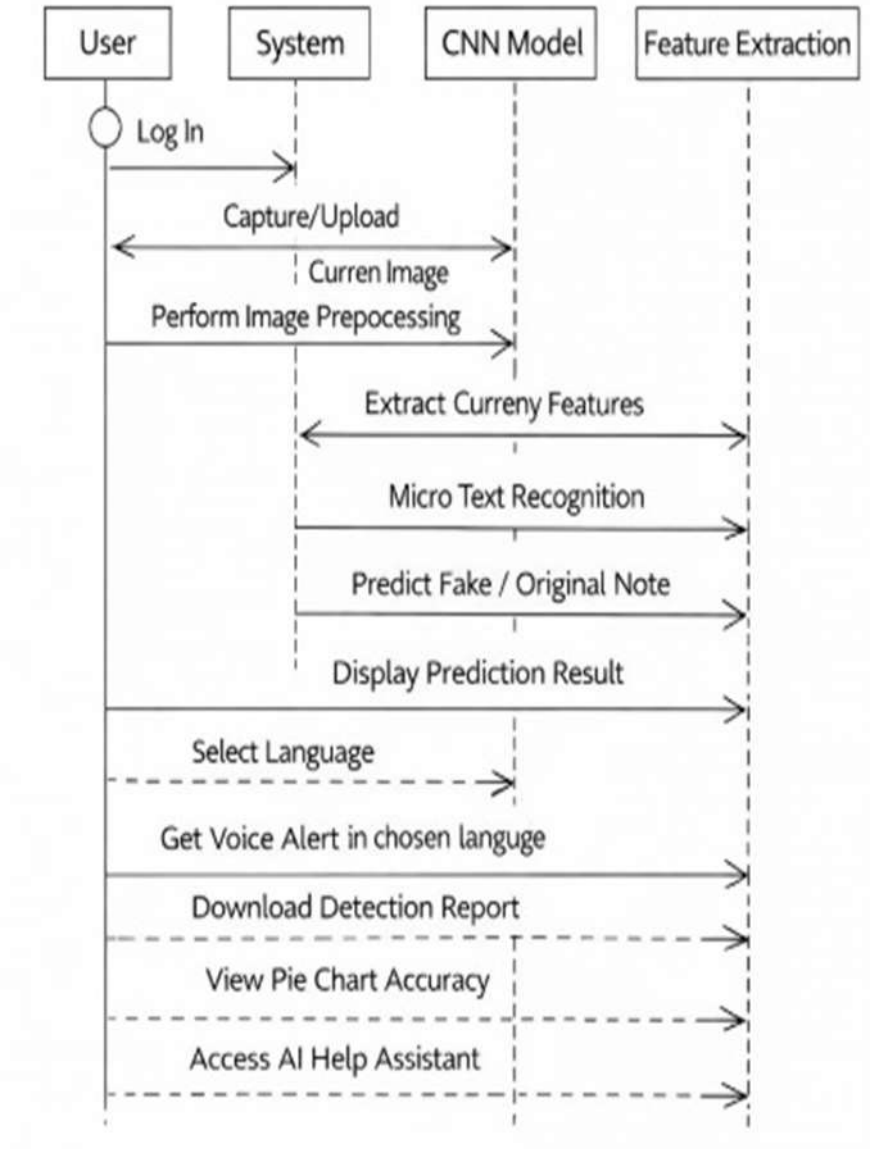
The system also provides an option to download the detection report. The report contains currency image, prediction result, accuracy score, and detected features. This allows users to store or share the result.

The user can also select the language for voice alert. The system supports multiple languages to announce the detection result. This improves accessibility for different users.

Finally, the system provides detection feedback to the user. The user can view the prediction accuracy and feature-based explanation. This helps users understand how the system determined whether the note is fake or original.

Thus, the Use Case Diagram shows the complete interaction between the user and the fake currency detection system.

4.6 Sequence Diagram



4.6 Sequence Diagram

The Sequence Diagram represents the step-by-step interaction between system components during fake currency detection. It shows how the user, system, CNN model, and feature extraction module communicate in sequence.

The process begins when the user logs into the system. The login request is sent to the system, and access is granted. After login, the user captures or uploads the currency image. The image is sent to the system for processing.

Once the image is received, the system performs image preprocessing. This includes resizing, noise removal, contrast enhancement, and normalization. Preprocessing improves image quality for accurate detection.

After preprocessing, the system sends the processed image to the feature extraction module. The feature extraction module identifies important security features such as watermark, security thread, denomination number, and portrait alignment. The extracted features are returned to the system.

The system then performs micro text recognition using OCR. This step extracts tiny printed text from the currency note. The recognized text is compared with known currency patterns.

Next, the processed image and extracted features are passed to the CNN model. The CNN model analyzes the image and predicts whether the currency note is fake or original. The model also calculates prediction confidence.

The prediction result is returned to the system. The system then displays the detection result to the user. The output includes fake or original classification, confidence score, and detected features.

After viewing the result, the user can select a preferred language. The system then provides a voice alert in the chosen language announcing the detection result.

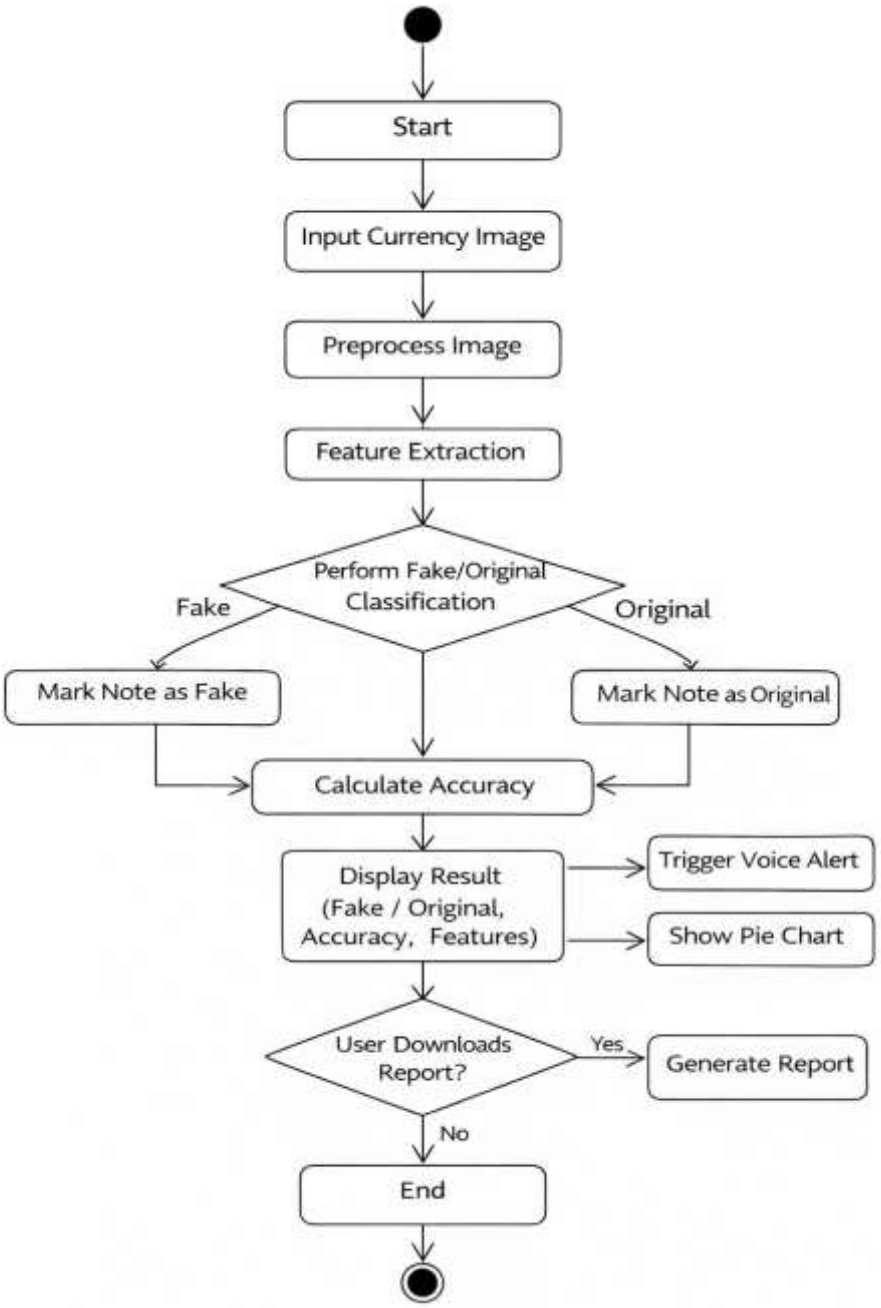
The user also has the option to download the detection report. The system generates a report containing the currency image, prediction result, accuracy, and timestamp.

The system also provides pie chart accuracy visualization. This shows the probability distribution of fake vs original classification.

Finally, the user can access the AI help assistant. The assistant explains detection results and provides additional guidance.

The sequence diagram therefore illustrates the complete workflow from image input to result output in the fake currency detection system.

4.7 Activity Diagrams



4.7 Activity Diagrams

The Activity Diagram represents the workflow of the Fake Currency Detection System from input image capture to final result generation. It illustrates the sequence of activities and decision-making steps involved in detecting whether a currency note is fake or original.

The process begins with the Start node, which indicates the beginning of the currency detection workflow. Once the system is initiated, the user provides the currency image. The input can be given either by capturing the image using a camera or by uploading an image from local storage. This step is represented as Input Currency Image.

After receiving the currency image, the system performs Image Preprocessing. In this stage, the input image is enhanced to improve detection accuracy. The preprocessing operations include resizing the image, converting to grayscale, removing noise, adjusting brightness and contrast, and aligning the currency note. These operations ensure that the image is suitable for further analysis.

Following preprocessing, the system performs Feature Extraction. In this step, important security features of the currency note are identified. These features include watermark detection, security thread, denomination number, RBI emblem, Mahatma Gandhi portrait, micro text, and serial number alignment. These extracted features are used by the classification model to determine authenticity.

After feature extraction, the system moves to the decision stage labeled Perform Fake/Original Classification. This is the core decision point of the activity diagram. The trained deep learning model analyzes the processed image and extracted features to determine whether the currency note is fake or original.

If the classification result indicates that the note is fake, the system executes the activity Mark Note as Fake. If the classification indicates that the note is genuine, the system executes Mark Note as Original. Both outcomes then proceed to the next stage.

Once the classification is completed, the system performs Calculate Accuracy. In this step, the model calculates the confidence score of the prediction. This accuracy percentage indicates how confident the system is about the classification result.
After calculating accuracy, the system proceeds to Display Result. The result displayed to the user includes:

- Fake or Original status
- Prediction accuracy percentage
- Detected currency denomination
- Extracted features
- Currency image preview

At the same time, additional functionalities are triggered. The system activates Voice Alert, which announces the detection result in the selected language. This improves accessibility for users. The system also Shows Pie Chart, which visually represents the probability of fake versus original classification.

After displaying the result, the system checks whether the user wants to download the report. This is represented by the decision node User Downloads Report?. If the user selects yes, the system performs Generate Report. The generated report contains the currency image, detection result, accuracy score, detected features, and timestamp.

If the user chooses not to download the report, the system directly proceeds to the final stage. The process ends with the End node, which indicates completion of currency detection.

Thus, the activity diagram clearly illustrates the complete workflow of the Fake Currency Detection System, starting from image input, preprocessing, feature extraction, classification, result display, optional report generation, and ending the process.

CHAPTER 5

IMPLEMENTATION

Introduction

The implementation phase of the Fake Currency Detection System focuses on transforming the designed architecture into a functional application using deep learning and image processing techniques. The system is developed to automatically detect whether a given currency note is genuine or counterfeit by analyzing its visual features. The application allows users to upload or capture currency images, which are then processed through a backend system where a trained deep learning model performs classification.

The system is implemented using Python along with libraries such as OpenCV for image processing and TensorFlow or PyTorch for deep learning model development. The implementation also includes additional user-centric features such as multilingual voice alerts in English, Kannada, and Hindi, downloadable report generation, and an AI-based help assistant. Furthermore, the system incorporates an advanced module for detecting old currency notes and estimating their resale value, making the application more versatile and practical. The overall implementation is modular in nature, ensuring scalability, maintainability, and efficient performance.

5.1 Background Study and Methodology

Background Study

The detection of counterfeit currency has traditionally relied on manual inspection techniques such as checking watermarks, security threads, and micro text using ultraviolet light or magnifying tools. However, these methods require expertise and are not always reliable, especially when dealing with high-quality counterfeit notes. With the rapid advancement of Artificial Intelligence and Deep Learning, automated systems have become capable of analyzing images and detecting complex patterns with high accuracy.

Deep learning models, particularly Convolutional Neural Networks (CNNs), have proven to be highly effective for image classification tasks. These models can automatically learn hierarchical features from images, such as edges, textures, and shapes, which are critical for distinguishing between genuine and fake currency notes. In this project, the system leverages CNN-based deep learning models along with image processing techniques and Optical Character Recognition (OCR) to analyze currency features such as micro text, watermark, denomination patterns, and security threads.

In addition to fake currency detection, the system includes features such as old currency recognition and resale value estimation. This is achieved by training the model on images of old currency notes and associating them with their respective market values. The inclusion of an AI-based assistant further enhances the usability of the system by providing real-time guidance and answering user queries related to currency detection.

Methodology

The methodology adopted for this project follows a systematic approach consisting of several stages, starting from data collection to final result generation. Initially, a dataset is collected containing images of genuine currency notes, counterfeit notes, damaged notes, and old currency notes. The dataset is carefully labeled to ensure accurate training of the model.

Once the dataset is collected, the images undergo preprocessing. This step includes resizing the images to a uniform dimension, removing noise, adjusting brightness and contrast, and normalizing pixel values. Data augmentation techniques such as rotation, flipping, and blurring are also applied to increase dataset diversity and improve model robustness.

After preprocessing, the images are used to train a Convolutional Neural Network. The CNN model learns to extract features from the currency images and classify them into categories such as fake or original. During training, the model adjusts its internal parameters to minimize classification error and improve accuracy.

Following model training, feature extraction is performed on the input images during runtime. Important features such as micro text, watermark patterns, security threads, and denomination numbers are identified using computer vision techniques and OCR. These features are then passed to the trained model for prediction.

The model generates a prediction indicating whether the currency note is fake or original, along with a confidence score. The results are then displayed to the user through the interface, accompanied by additional features such as pie

chart visualization, voice alerts, and report generation. The methodology also includes modules for old currency detection and resale value estimation, as well as an AI assistant that interacts with users and provides explanations.

5.2 About CNN

The core component of the system is the Convolutional Neural Network (CNN), which is used for image classification. The CNN model processes the input currency image through multiple layers to extract meaningful features and perform classification. The first layer of the CNN is the convolutional layer, which applies filters to the input image to detect basic features such as edges and textures. These features are then passed through an activation function, typically ReLU (Rectified Linear Unit), which introduces non-linearity into the model.

After activation, the image passes through a pooling layer, which reduces the spatial dimensions of the feature maps while retaining important information. This helps in reducing computational complexity and preventing overfitting. The process of convolution, activation, and pooling is repeated multiple times to extract higher-level features from the image.

Once the feature extraction process is complete, the output is flattened into a one-dimensional vector. This vector is then passed through fully connected layers, which learn the relationships between extracted features and the final output classes. The final layer of the CNN is the output layer, which uses a softmax or sigmoid function to produce the probability of the image belonging to each class. Based on these probabilities, the system classifies the currency note as fake or original.

The CNN model is trained using a labeled dataset and optimized using techniques such as backpropagation and gradient descent. The model achieves high accuracy by learning complex patterns in currency images, making it suitable for real-world applications.

1. Convolutional Neural Network (CNN)
CNN is used to classify currency images as genuine or fake. It processes images through multiple layers to automatically detect features like edges, textures, and shapes. The model then predicts whether the note is original or counterfeit.

2. YOLO Algorithm
YOLO is used for detecting important regions in the currency note. It identifies areas like watermark, serial number, and security features in a single step. This helps improve speed and accuracy.

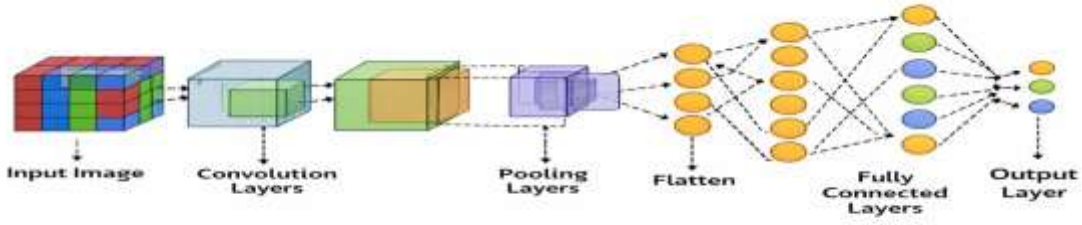
3. Optical Character Recognition (OCR)
OCR is used to extract text from currency notes. It reads details like serial numbers and small printed text. The extracted text is compared to verify authenticity.

4. Image Processing Algorithms
Image processing techniques improve image quality before analysis. Methods like noise reduction, edge detection, and normalization help the model detect features more clearly.

5. Text-to-Speech (TTS)
TTS converts the result into voice output. It announces whether the currency is fake or real. It supports multiple languages for better usability.

6. Regression Algorithm

Regression is used to estimate the value of old currency notes. It analyzes features like condition and rarity to predict resale value.



5.2. CNN Architecture

CNN Architecture

The Convolutional Neural Network (CNN) is designed to automatically learn and extract important features from an input image and classify it into categories. The architecture shown in the diagram consists of multiple stages, each performing a specific task.

1. Input Image

The process begins with the input image, which is the currency note image uploaded by the user.
This image is represented as a matrix of pixel values (RGB channels).

2. Convolution Layers

The convolution layer is the core building block of CNN.
It applies multiple filters (kernels) to the image.
Each filter detects specific features such as:

- Edges
- Textures
- Patterns
- Shapes

3. Activation Function (ReLU)

After convolution, an activation function is applied.
Most commonly used: ReLU (Rectified Linear Unit)

4. Pooling Layers

Pooling reduces the size of feature maps.
Common type: Max Pooling
It selects the maximum value from a region

5. Flatten Layer

After multiple convolution + pooling operations:
The 2D feature maps are converted into a 1D vector

6. Fully Connected Layers (Dense Layers)

These layers act like a traditional neural network.
Every neuron is connected to every other neuron
Learns relationships between features

7. Output Layer
The final layer produces the prediction.

5.4 Coding

```
import os, sys, uuid, json, logging, csv, asyncio, base64, io
import numpy as np
from flask import Flask, render_template, request, jsonify, send_file
from werkzeug.utils import secure_filename
import httpx
import re
import os
# TEMPORARY - delete after fix
Withopen(os.path.join(os.path.dirname(os.path.abspath(__file__)), "old_currency_value.csv"), "rb") as f:
    raw = f.read(500)
print("RAW CSV BYTES:", raw)
print("CWD:", os.getcwd())
print("__file__:", __file__)
print("CSVpath:",os.path.join(os.path.dirname(os.path.abspath(__file__)), "old_currency_value.csv"))
print("CSVexists:",os.path.exists(os.path.join(os.path.dirname(os.path.abspa_file__)), "old_currency_value.csv")))

def normalize_name(name: str) -> str:
    name = name.lower()
    name = re.sub(r'[^a-z0-9\s]', '', name)
    name = name.replace("rupees", "rupee")
    name = name.strip()
    return name
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)
app = Flask(__name__)
app.config["MAX_CONTENT_LENGTH"] = 32 * 1024 * 1024
app.config["UPLOAD_FOLDER"] = os.path.join("static", "uploads")
app.config["SECRET_KEY"] = "currency-detection-secret-2024"

ALLOWED_EXTENSIONS = {"jpg", "jpeg", "png", "webp", "bmp"}

# ── LLM API config (from provided app.py) ─
WEBUI_BASE_URL=os.getenv("WEBUI_BASE_URL", "https://chat.ivislabs.in/api")
API_KEY=os.getenv("API_KEY", "sk-44371278273c49fc97847b5e5166bca1")
DEFAULT_MODEL=os.getenv("DEFAULT_MODEL","llama3.2vision:latest")

# ── Old Currency CSV loader ──
OLD_CURRENCY_VALUES = {}

def load_old_currency_csv():
    csv_path=os.path.join(os.path.dirname(os.path.abspath(__file__)), "old_currency_value.csv")
    if not os.path.exists(csv_path):
        print(">>> CSV NOT FOUND!")
        return
```

```
    with open(csv_path, encoding='utf-8') as f:
        for line in f:
            line = line.strip().strip("")
            if not line or line.lower().startswith("currency_name"):
                continue
            last_comma = line.rfind(",")
            if last_comma == -1:
                continue
            name  = line[:last_comma].strip()
            value = line[last_comma + 1:].strip()
            if name and value:
                OLD_CURRENCY_VALUES[name.lower()] = value
    print(f">>> TOTAL LOADED: {len(OLD_CURRENCY_VALUES)}")

load_old_currency_csv()
# ── Model manager ──
from utils.model_manager import ModelManager
model_manager = ModelManager()

def allowed_file(filename):
    return "." in filename and filename.rsplit(".", 1)[1].lower() in ALLOWED_EXTENSIONS

# ── LLM helpers ─
async def call_llm_async(prompt: str, image_b64: str = None) -> str:
    content = []
    if image_b64:
content.append({"type":"image_url","image_url":{"url":f"{image_b64}"}})
    content.append({"type": "text", "text": prompt})

    messages = [
        {"role": "system", "content": (
            "You are an expert Indian currency authentication specialist with deep knowledge of "
            "RBI security features, micro-printing, watermarks, and old collectible banknotes. "
            "Provide detailed, structured analysis."
        )},
        {"role": "user", "content": content if image_b64 else prompt}
    ]

    try:
        async with httpx.AsyncClient(timeout=30.0) as client:
            resp = await client.post(
                f"{WEBUI_BASE_URL}/chat/completions",
                headers={"Authorization": f"Bearer {API_KEY}", "Content-Type": "application/json"},
                json={"model":DEFAULT_MODEL,"messages":messages, "max_tokens": 600, "temperature": 0.3},
            )
        if resp.status_code == 200:
            data = resp.json()
            if "choices" in data and data["choices"]:
                return data["choices"][0]["message"]["content"].strip()
        logger.warning(f"LLM returned {resp.status_code}")
    except Exception as e:
```

```
        logger.error(f"LLM call failed: {e}")
    return None

def run_llm(prompt, image_b64=None):
    try:
        loop = asyncio.new_event_loop()
        asyncio.set_event_loop(loop)
        result = loop.run_until_complete(call_llm_async(prompt, image_b64))
        loop.close()
        return result
    except Exception as e:
        logger.error(f"Loop error: {e}")
        return None

# ── Routes ─
@app.route("/")
def index():
    return render_template("index.html")

@app.route("/predict_fake_real", methods=["POST"])
def predict_fake_real():
    if "image" not in request.files:
        return jsonify({"error": "No image provided"}), 400

    file = request.files["image"]
    if not allowed_file(file.filename):
        return jsonify({"error": "Unsupported format"}), 400

    ext  = file.filename.rsplit(".", 1)[1].lower()
    fname = f"{uuid.uuid4().hex}.{ext}"
    savepathos.path.join(app.config["UPLOAD_FOLDER"], secure_filename(fname))
    file.save(savepath)

    try:
        denomination = request.form.get("denomination", "unknown")
        result       = model_manager.predict_fake_real(savepath, denomination)

        # Encode image for LLM vision analysis
        with open(savepath, "rb") as f:
            img_b64 = base64.b64encode(f.read()).decode()

        # AI Agent analysis
        llm_prompt = f"""Analyze this Indian ₹{denomination} currency note image.
Determine: 1) Is it REAL or FAKE? 2) List security features visible (watermark, security thread, microprint, color-shift ink, latent image, intaglio printing).
3) Confidence percentage. 4) Specific anomalies if fake.
FormatresponseasJSON:
{{"verdict":"REAL/FAKE","confidence":90,"features":["feature1"],"anomalies":[],"analysis":"brief text"}}"""

        llm_raw = run_llm(llm_prompt, img_b64)
        llm_data = {}
```

```
        if llm_raw:
            try:
                start = llm_raw.find("{")
                end   = llm_raw.rfind("}") + 1
                if start >= 0:
                    llm_data = json.loads(llm_raw[start:end])
            except Exception:
                llm_data = {"analysis": llm_raw}

        result["llm_analysis"] = llm_data
        result["image_path"]   = f"/static/uploads/{fname}"
        return jsonify(result)

    except Exception as e:
        logger.exception("Fake/real prediction error")
        return jsonify({"error": str(e)}), 500
    finally:
        # Keep file for report generation; clean up after 10 min handled client-side
        pass

@app.route("/predict_old_currency", methods=["POST"])
def predict_old_currency():
    if "image" not in request.files:
        return jsonify({"error": "No image provided"}), 400

    file = request.files["image"]
    if not allowed_file(file.filename):
        return jsonify({"error": "Unsupported format"}), 400

    ext   = file.filename.rsplit(".", 1)[1].lower()
    fname = f"{uuid.uuid4().hex}.{ext}"
savepath=os.path.join(app.config["UPLOAD_FOLDER"],secure_filename(fname))
    file.save(savepath)

    try:
        result = model_manager.predict_old_currency(savepath)


        # Encode image for LLM
        with open(savepath, "rb") as f:
            img_b64 = base64.b64encode(f.read()).decode()
        llm_prompt = """Identify this old/collectible Indian currency note.

Provide the FULL descriptive name exactly as it would appear in a collector's database, including:
- Denomination (e.g. 1 Rupee, 10 Rupees)
- Issuing authority (British India / RBI / Government of India)
- Governor/Secretary name if visible
- Any special features (fancy number, star note, error note, holy number 786, rare series, UNC)
- Year if visible
```

```
Format as JSON: {"name":"full descriptive collector name","denomination":"10 Rupees","year":"1985","governor":"R.N.Malhotra","special_features":["starno","fancy number"],"historical_note":"brief"}"""
        llm_raw = run_llm(llm_prompt, img_b64)
        llm_data = {}
        if llm_raw:
            try:
                start=llm_raw.find("{")
                end =llm_raw.rfind("}") + 1
                if start >= 0:
                    llm_data = json.loads(llm_raw[start:end])
            except Exception:
        llm_data={"name":result.get("predicted_name",""),"analysis":llm_raw}

        # Lookup resale value from CSV
        identified_name = llm_data.get("name", result.get("predicted_name", ""))
        print("=== RESALE DEBUG ===")
        print(f"LLM identified_name: '{identified_name}'")
        print(f"CSV total entries: {len(OLD_CURRENCY_VALUES)}")
        print(f"CSV keys: {list(OLD_CURRENCY_VALUES.keys())}")
        resale_value = lookup_resale_value(identified_name)
        print(f"Resale result: {resale_value}")
        print("====================")
        normalized_name = normalize_name(identified_name)
        resale_value = lookup_resale_value(normalized_name)

        result["llm_analysis"]   = llm_data
        result["resale_value"]   = resale_value
        result["identified_name"] = identified_name
        result["image_path"]     = f"/static/uploads/{fname}"
        print("IDENTIFIED NAME:", identified_name)
        print("CSV KEYS:", list(OLD_CURRENCY_VALUES.keys()))
        print("RESALE:", resale_value)
        return jsonify(result)

    except Exception as e:
        logger.exception("Old currency prediction error")
        return jsonify({"error": str(e)}), 500
    print("LLM RAW:", llm_raw)
    print("LLM DATA:", llm_data)
    print("IDENTIFIED NAME:", identified_name)

def lookup_resale_value(name: str) -> dict:
    if not name or not OLD_CURRENCY_VALUES:
        return {"found": False, "value": None}

    name_lower = name.lower().strip()

    # 1. Exact match
    if name_lower in OLD_CURRENCY_VALUES:
        return {"found":True,"value": OLD_CURRENCY_VALUES[name_lower], "match_type": "exact"}
```

```
    # 2. Substring match — CSV key contains search name or vice versa
    for key, val in OLD_CURRENCY_VALUES.items():
        if name_lower in key or key in name_lower:
     return{"found":True,"value":val,"match_type":"substring","matched_name": key}

    # 3. Score-based word overlap (ignore common filler words)
 STOP_WORDS = {"note", "banknote", "rupee", "rupees", "india", "rare", "unc", "issue", "sign", "with", "serial",
"number", "fancy", "holy","inset", "star", "error", "prefix", "bundle", "notes"}

    name_words = [w for w in name_lower.split() if len(w) > 2 and w not in STOP_WORDS]

    best_score, best_key, best_val = 0, None, None
    for key, val in OLD_CURRENCY_VALUES.items():
        key_words = set(w for w in key.split() if len(w) > 2 and w not in STOP_WORDS)
        matches = sum(1 for w in name_words if w in key_words)
        # Boost score if denomination digit matches (e.g. "10", "500")
        digit_matches = sum(1 for w in name_lower.split() if w.isdigit() and w in key)
        score = matches + (digit_matches * 2)
        if score > best_score:
            best_score, best_key, best_val = score, key, val

    if best_score >= 2:
        return {"found": True, "value": best_val, "match_type": "scored",
                "matched_name": best_key, "score": best_score}

    # 4. Denomination-only fallback — match just the number (e.g. "500", "10")
    digits_in_name = [w for w in name_lower.split() if w.isdigit()]
    for key, val in OLD_CURRENCY_VALUES.items():
        if any(d in key for d in digits_in_name):
return{"found":True,"value":val,"match_type":"denomination_fallback","matched_name": key}

    return {"found": False, "value": None,
            "available_sample": list(OLD_CURRENCY_VALUES.items())[:5]}
@app.route("/api/debug_resale")
def debug_resale():
    test_name = request.args.get("name", "")
    return jsonify({
        "csv_total_entries": len(OLD_CURRENCY_VALUES),
        "csv_sample": list(OLD_CURRENCY_VALUES.items())[:10],
        "lookup_result": lookup_resale_value(test_name) if test_name else "pass ?name=... to test"})
@app.route("/api/agent_chat", methods=["POST"])
def agent_chat():
    body   = request.get_json(silent=True) or {}
    message = body.get("message", "").strip()
    context = body.get("context", "")
    if not message:
        return jsonify({"error": "No message"}), 400

    full_prompt = f"Currency detection context: {context}\n\nUser question: {message}\n\nAnswer as an expert
currency authentication agent."
```

```
    response    = run_llm(full_prompt)
    if not response:
        response = "I'm your Currency Detection AI Agent. I can help you understand security features of Indian currency notes, identify fake notes, and provide information about collectible old currency values. Please upload an image for detailed analysis."
    return jsonify({"response": response})

@app.route("/api/generate_report", methods=["POST"])
def generate_report():
    from utils.report_generator import generate_pdf_report
    body   = request.get_json(silent=True) or {}
    result = body.get("result", {})
    mode   = body.get("mode", "fake_real")
    try:
        pdf_path = generate_pdf_report(result, mode)
returnsend_file(pdf_path,as_attachment=True,download_name="currency_report.pdf", mimetype="application/pdf")
    except Exception as e:
        logger.error(f"Report error: {e}")
        return jsonify({"error": str(e)}), 500

@app.route("/api/old_currency_list")
def old_currency_list():
    return jsonify({"currencies": [{"name": k, "value": v} for k, v in OLD_CURRENCY_VALUES.items()]})

@app.route("/health")
def health():
returnjsonify({"status":"ok","old_currency_count":len(OLD_CURRENCY_VALUES)})

if __name__ == "__main__":
    os.makedirs(app.config["UPLOAD_FOLDER"], exist_ok=True)
    app.run(debug=True, host="0.0.0.0", port=5000)
```

Chapter 6

System Testing

6.1 Introduction

System testing is an essential phase in the development of the Fake Currency Detection System. It ensures that the developed application functions correctly and meets the specified requirements. The testing process involves verifying each module of the system, including image input, preprocessing, deep learning prediction, voice alert, report generation, and AI assistant.

In this project, testing is performed to evaluate the accuracy, performance, reliability, and usability of the system. The system is tested using different types of currency images such as genuine notes, counterfeit notes, damaged notes, and old currency notes. The goal of system testing is to ensure that the application produces correct results under various conditions and provides a seamless user experience

6.2 Objective of Testing

The main objective of testing is to ensure that the system operates correctly and efficiently. The specific objectives include verifying that the system correctly classifies currency notes as fake or original, ensuring that all features such as voice alerts, multilingual support, and report generation function properly, and validating that the system handles different types of inputs including damaged and low-quality images.

Another objective is to ensure that the system provides accurate prediction results with minimal response time. Testing also aims to identify and fix errors, bugs, or inconsistencies in the system. Additionally, it ensures that the AI assistant responds correctly to user queries and that the old currency detection module provides appropriate resale value estimation.

6.3 Testing Methods

In order to ensure that the Fake Currency Detection System functions accurately, efficiently, and reliably, multiple testing methods are employed during the development process. Each testing method focuses on a specific aspect of the system and helps in identifying errors at different stages. The combination of these testing techniques ensures that both individual components and the overall system perform as expected.

1.Unit Testing
Unit Testing is the first level of testing performed in the system. In this method, individual modules or components of the application are tested independently to verify their correctness. For example, the image preprocessing module is tested separately to ensure that operations such as resizing, noise removal, and normalization are working correctly. Similarly, the feature extraction module is tested to confirm that it accurately detects features such as watermark, micro text, and denomination patterns. The deep learning model prediction function is also tested independently to ensure it returns correct classification outputs. Unit testing helps in identifying bugs at an early stage and ensures that each component functions correctly before integration.

2.Integration Testing
Integration Testing is performed after unit testing to verify that different modules of the system work together seamlessly. In this project, integration testing ensures that the frontend interface correctly communicates with the backend server, and the backend properly interacts with the deep learning model. For example, when a user uploads an image through the frontend, it should be successfully sent to the backend, processed by the model, and the result should be returned without errors. Integration testing also ensures that additional features such as voice alerts, report generation, and AI assistant are properly connected to the main system. This testing helps in identifying issues related to data flow and communication between modules.

3.System Testing
System Testing is conducted to evaluate the complete system as a whole. In this phase, the entire application is tested under real-world conditions to ensure that all functionalities are working correctly. The system is tested using different types of inputs such as genuine currency images, counterfeit notes, damaged notes, and low-quality images. This testing verifies that the system correctly performs all operations including image processing, feature extraction, prediction, voice output, and report generation. System testing ensures that the application meets all functional requirements and behaves as expected in real usage scenarios.

4.User Acceptance Testing (UAT)

User Acceptance Testing (UAT) is performed to ensure that the system meets user expectations and is easy to use. In this phase, real users interact with the application and provide feedback based on their experience. The usability of features such as image upload, result display, voice alerts, and AI assistant is evaluated. The goal of UAT is to confirm that the system is user-friendly, intuitive, and meets the requirements specified by the end users. Any issues identified during this phase are addressed to improve the overall user experience.

5.Performance Testing

Performance Testing is used to evaluate the speed, responsiveness, and stability of the system. Since the application involves real-time image processing and deep learning prediction, it is important to ensure that the system provides results within an acceptable time frame. Performance testing measures the time taken for image upload, preprocessing, model prediction, and result display. It also evaluates how the system behaves under multiple requests or continuous usage, such as live camera scanning. This testing ensures that the system does not experience delays, crashes, or performance degradation.

6.Validation Testing

Validation Testing is also an important method used in this project to assess the accuracy of the deep learning model. The model is tested using unseen data (test dataset) to evaluate its prediction performance. Metrics such as accuracy, precision, and recall are used to measure how well the model distinguishes between fake and genuine currency notes. This testing ensures that the model is reliable and performs well in real-world scenarios.

7.Robustness Testing

Robustness Testing is performed to check how well the system handles unexpected or invalid inputs. The system is tested with blurred images, rotated images, low-light images, and non-currency images to ensure that it does not crash and still provides meaningful output or appropriate error messages. This ensures that the system is stable and can handle real-world variations in input data.

Overall, these testing methods collectively ensure that the Fake Currency Detection System is accurate, efficient, reliable, and user-friendly. By thoroughly testing each component and the entire system, the application achieves high performance and robustness in detecting counterfeit currency.

6.4 Validation

Validation is the process of ensuring that the system produces accurate and reliable results. In this project, validation is performed using a test dataset that contains both genuine and counterfeit currency images. The trained deep learning model is tested on unseen data to evaluate its performance.

The system calculates prediction accuracy based on correct classifications. The validation process also checks whether the system correctly identifies features such as micro text, watermark, and denomination. Additionally, validation ensures that the system can handle damaged currency notes and still provide reasonable predictions.

The results obtained from validation help in improving the model by fine-tuning parameters and retraining with additional data if necessary.

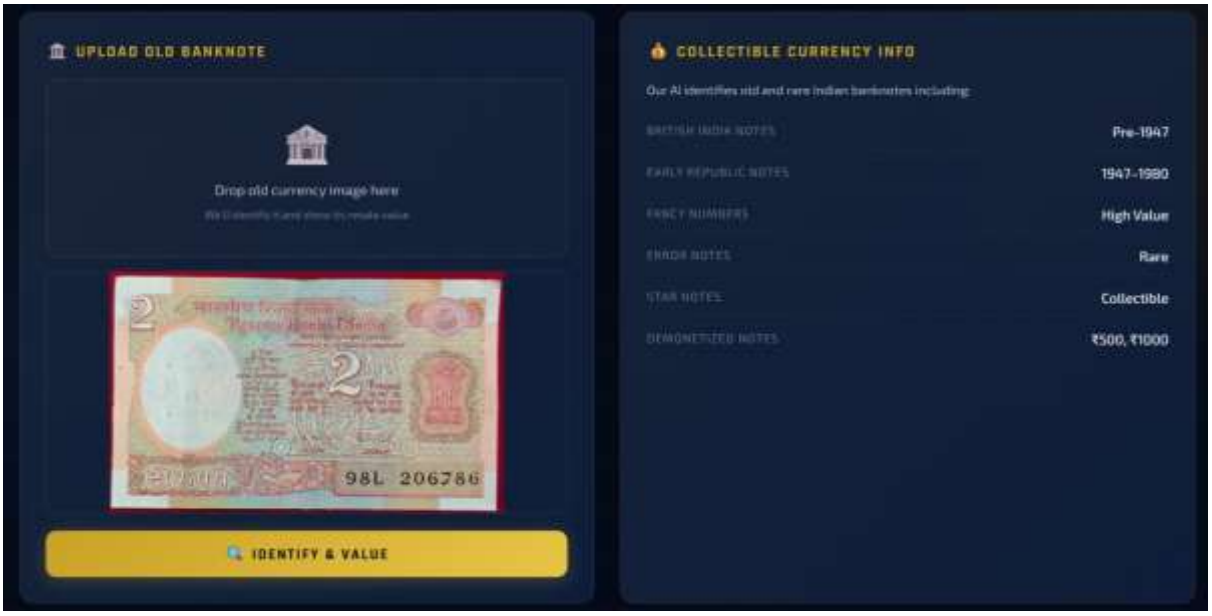
6.5 Test Reports

Below are the test cases for the Fake Currency Detection System:

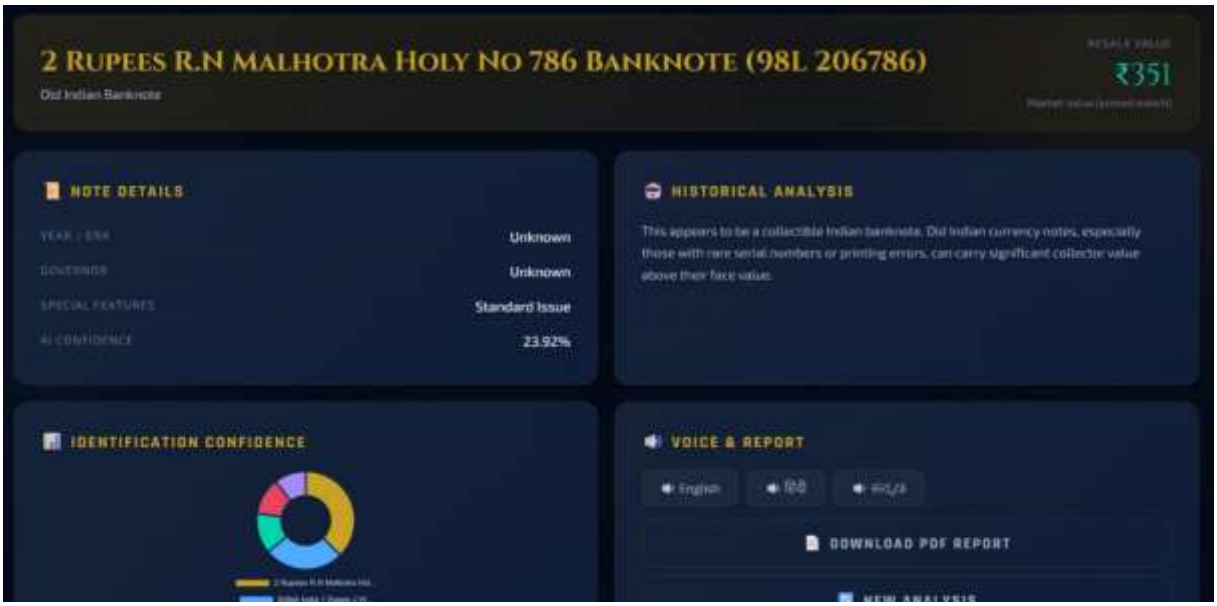
Test Case ID	Test Purpose	Test Condition	Expected Outcome	Actual Result	Status
TC01	Upload valid genuine currency	Upload clear original note image	System detects as Original	Detected as Original	Pass
TC02	Upload fake currency image	Upload counterfeit note image	System detects as Fake	Detected as Fake	Pass
TC03	Detect damaged note	Upload torn/folded currency	System still predicts result	Predicted correctly	Pass
TC04	Multiple denomination detection	Upload ₹10, ₹50, ₹500 notes	Correct denomination identified	Correct output	Pass
TC05	Micro text detection	Upload high-quality image	Micro text extracted correctly	Text detected	Pass
TC06	Voice alert system	Enable voice output	System announces result	Voice working	Pass
TC07	Multi-language support	Select Kannada/Hindi	Output in selected language	Language switched	Pass
TC08	Pie chart accuracy	Perform prediction	Pie chart displayed	Chart displayed	Pass
TC09	Report generation	Click download report	PDF report generated	Report downloaded	Pass
TC10	Old currency detection	Upload old note	Detect note + resale value	Value displayed	Pass
TC11	AI assistant response	Ask question	Correct response generated	Response accurate	Pass
TC12	Invalid image input	Upload non-currency image	Show error message	Error displayed	Pass
TC13	Low-quality image	Upload blurred image	System attempts prediction	Prediction given	Pass
TC14	Real-time camera scan	Use live camera	Detect currency instantly	Working correctly	Pass
TC15	System performance	Multiple inputs	No lag or crash	Stable performance	Pass

CHAPTER 7

SNAPSHOTS



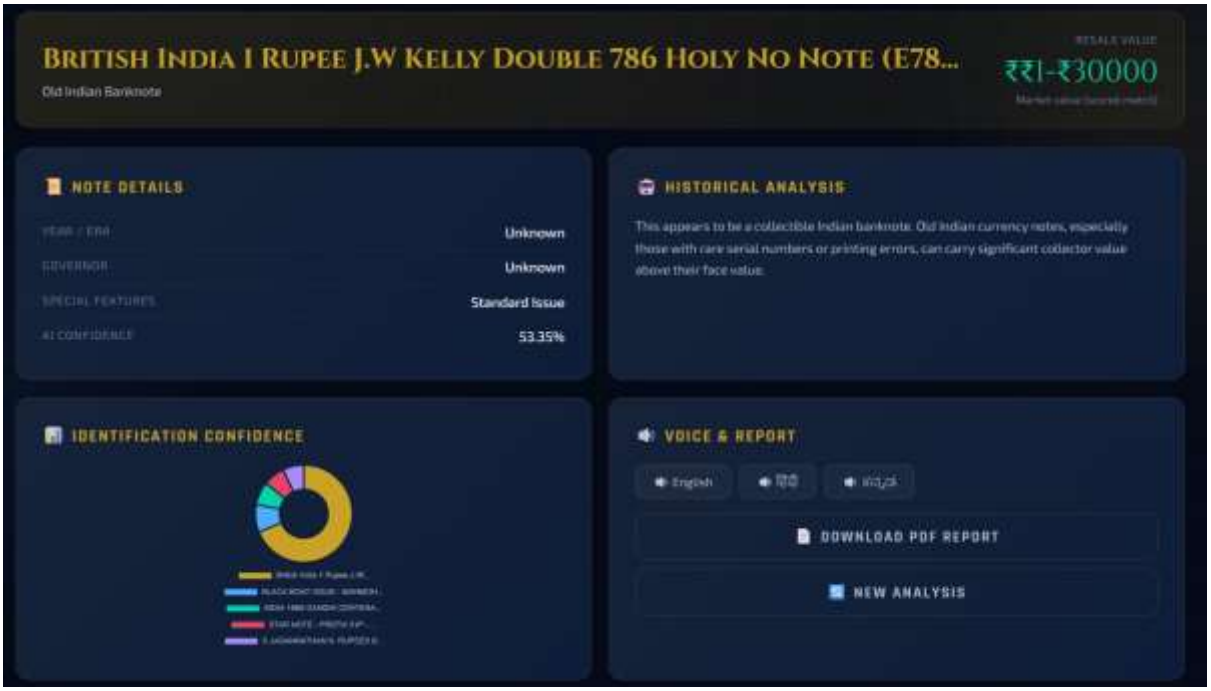
7.1 Two Rupee Old Banknote



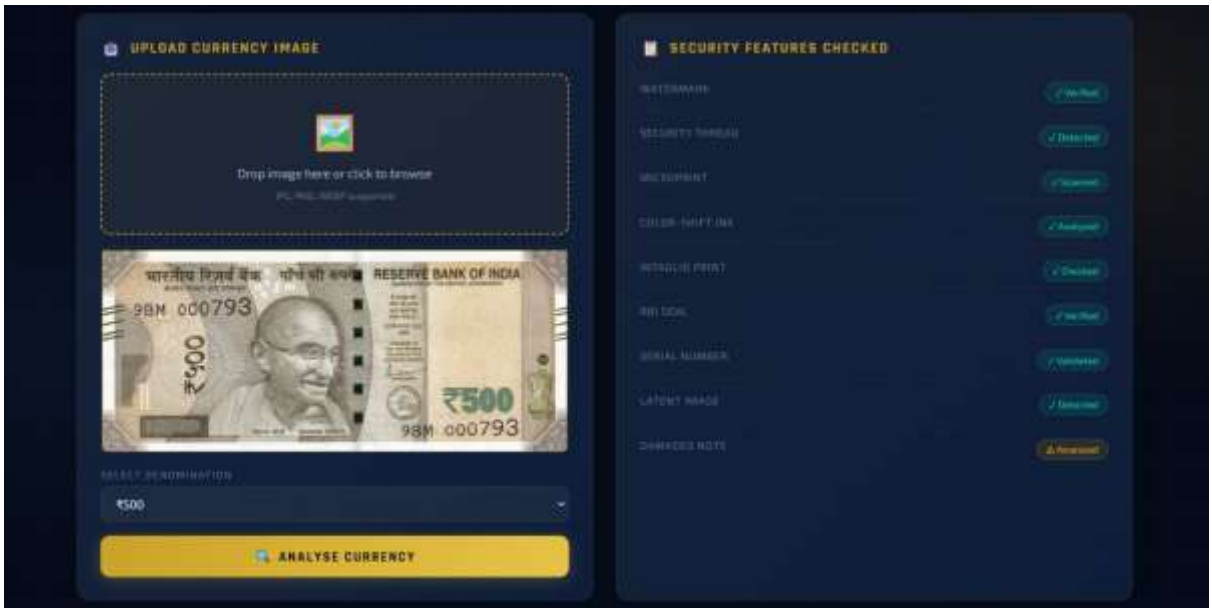
7.2 Two Rupee Old Banknote Result



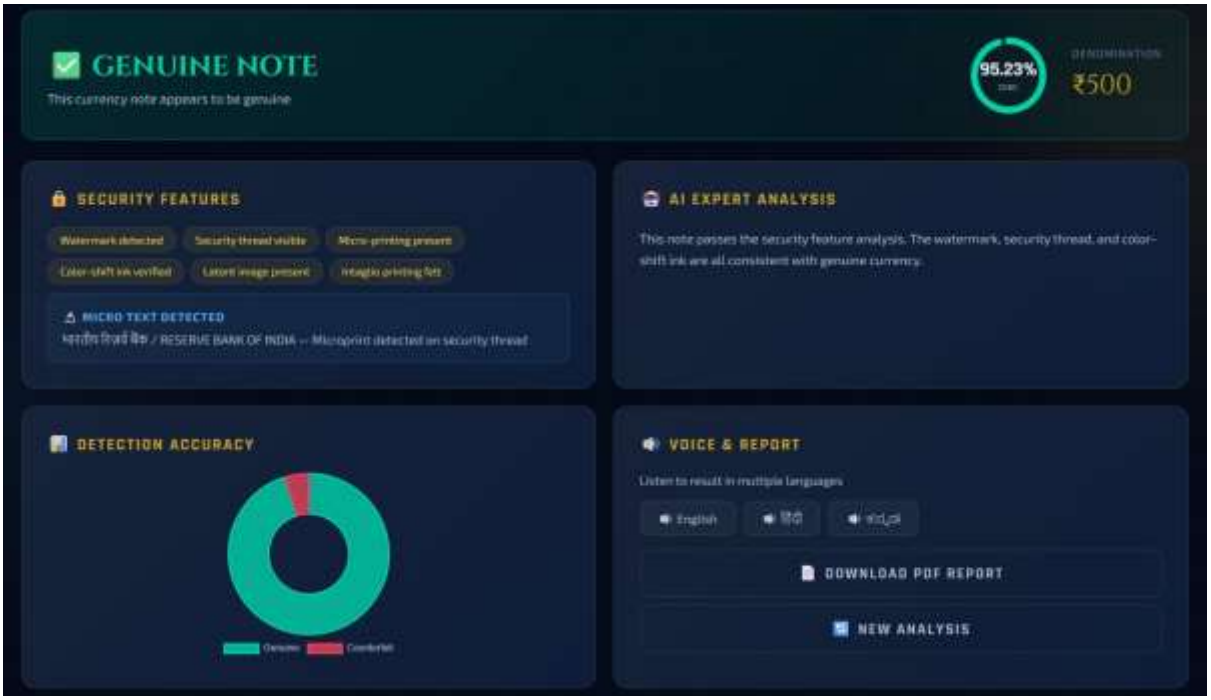
7.3 One Ruppe Old Banknote



7.4 One Rupee Old Banknote Result



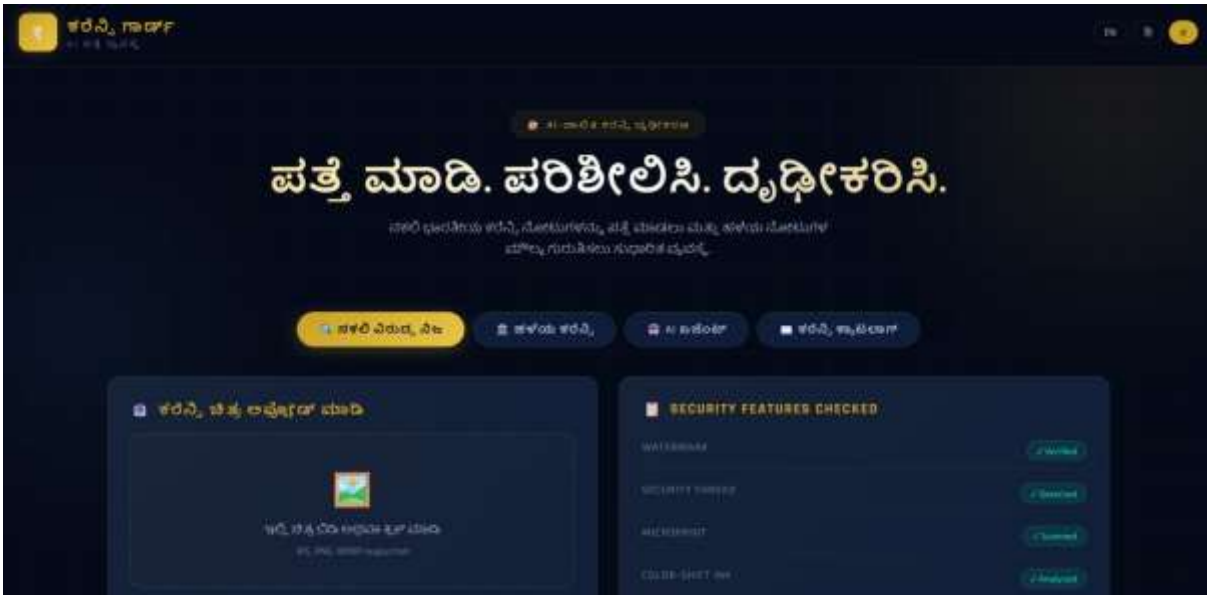
7.5 500 Rupee Currency Image



7.6 Results



7.7 Multilingual(Hindi)



7.8 Multilingual(Kannada)



7.9 AI Agent



7.10 Currency Catalog

CONCLUSION

The Fake Currency Detection System developed in this project demonstrates the effective use of Artificial Intelligence and Deep Learning techniques in solving a real-world problem. The system successfully identifies whether a currency note is genuine or counterfeit by analyzing visual features such as patterns, textures, micro text, and security elements using a Convolutional Neural Network (CNN). The implementation proves that automated detection is more efficient, faster, and reliable compared to traditional manual methods.

The system is designed to handle multiple denominations of currency and is capable of detecting even damaged or partially visible notes. It provides accurate prediction results along with a confidence score, enabling users to understand the reliability of the output. The inclusion of feature-based analysis further improves transparency by showing how the system arrived at its decision.

In addition to the core detection functionality, the project incorporates several advanced and user-friendly features such as multilingual voice alerts, pie chart visualization of prediction accuracy, and downloadable report generation. These features enhance usability and make the system accessible to a wider range of users, including those with limited technical knowledge.

The project also extends its capabilities by introducing an old currency detection module, which identifies old currency notes and estimates their resale value. This adds practical value to the system beyond counterfeit detection. Furthermore, the integration of an AI-based assistant enables users to interact with the system, ask questions, and receive guidance, thereby improving user engagement and understanding.

Through comprehensive testing and validation, the system has shown reliable performance under various conditions, including different image qualities and input types. The modular architecture ensures that the system is scalable and can be further improved by incorporating additional features such as real-time detection using advanced sensors or expanding support to other currencies.

In conclusion, the project successfully achieves its objective of developing an intelligent, efficient, and user-friendly Fake Currency Detection System. It highlights the potential of deep learning in image-based classification tasks and provides a strong foundation for future enhancements and real-world deployment.

FUTURE ENHANCEMENT

- The Fake Currency Detection System provides a strong base for detecting fake notes using deep learning, but it can be improved in many ways.
- In the future, the system can include **UV and IR scanning** to detect hidden security features that normal images cannot capture, improving accuracy.
- It can also be expanded to support **international currencies**, making it useful in places like banks and airports.
- A **mobile application** can be developed so users can scan currency using their phones, making the system more accessible.
- The model can be improved by using **advanced deep learning techniques** and a larger dataset to increase accuracy.
- More features like **serial number checking, color analysis, and hologram detection** can be added for better verification.
- The AI assistant can be enhanced to provide **better interaction and guidance**, including voice support.
- The old currency module can be connected to **real-time market data** to give accurate resale values.
- The system can also use **cloud technology** for better performance and handling multiple users.
- Security can be improved with **user authentication and data protection**.
- Finally, the system can be integrated with **ATMs and banking systems** to automatically detect fake currency.

In conclusion, the system has significant potential for future growth and can be enhanced with advanced technologies, improved models, and broader applications to make it more accurate, scalable, and widely usable in real-world scenarios.

REFERENCE

[1]. Vinay Kumar & Muni Kumar(2025). Fake Indian currency detection uses Convolutional Neural Network (CNN) and MobileNet . International Journal of Scientific Research in Computer Science, Engineering and Information Technology
https://ijsrcseit.com/index.php/home/article/view/CSEIT25113338

[2]. Deekshitha K, Dr. Kumar Siddamallappa U., & Anusha Jaju J. (2025), Fake Indian Currency Detection Using Machine Learning, International Journal for Multidisciplinary Research.

[3]. Vivek Sharma, Amrendra Kumar, & Rajendra Singh (2025), Identification of Indian Currency Note Authenticity Using Ensemble Learning and Image Processing, International Journal of Advanced Technology and Engineering Exploration.

[4]. Nikita Gauda, Ruchi Bhati, Savi Band, & Kumud Wasnik (2025), AUTHENTINOTE: A Counterfeit Currency Detection System, International Research Journal of Engineering and Technology.