

Fake Currency Detection Using Machine Learning and Image Processing

Pothumarthi Sriram, Paravada Teja, Pampana Veerendra, Chinthakayala Swaroop
Department of Artificial Intelligence & Machine Learning, Parul Institute of Engineering and Technology
Parul University, Vadodara, Gujarat, India

B.Tech (Computer Science & Engineering — AI&ML),
Under the Guidance of Prof. Asim Makkad, AI & ML Department

Abstract—Counterfeit currency is a serious financial crime that can shake a whole country's economy; it can be prevented, but people are often unaware of how to tell a counterfeit note from a genuine one, so spreading awareness is key. This paper presents a Fake Currency Detection system that applies machine learning and image processing to classify banknote images as genuine or fake. Statistical features (variance, skewness, kurtosis, and entropy) are extracted from the images using wavelet transforms, and classification is performed by supervised models—Random Forest, K-Nearest Neighbours (KNN), and Gradient Boosting—combined in an ensemble (Voting Classifier) and tuned with GridSearchCV and cross-validation. The paper reports a twenty-study literature survey, the software requirements specification (SRS), a layered, modular system design, the Waterfall development methodology, the implementation stack (Python, OpenCV, NumPy, Pandas, Scikit-learn, PyWavelets, Flask/Django, MySQL/MongoDB), and the testing and deployment strategy. Convolutional neural networks (CNNs), which excel at image pattern recognition, are discussed as a complementary direction. The source work also acknowledges limitations: model effectiveness depends on the quality and comprehensiveness of the training data, counterfeiters constantly adopt new techniques so the dataset must be kept up to date, and the models themselves must be secured against manipulation. Future directions include transfer learning, which adapts pre-trained models to reduce training time and resources, and generative adversarial networks (GANs) for building more comprehensive training datasets.

Keywords—Fake currency, Machine learning, Image processing, NumPy, OpenCV, Convolutional neural network.

I. INTRODUCTION

Currency counterfeiting is a serious financial crime that allows criminals to illegitimately obtain goods and services, undermining confidence in the monetary system. Highly deceptive counterfeit notes are extremely difficult to identify by visual inspection alone. Advanced methods from artificial intelligence provide powerful tools for discerning subtle differences between genuine and counterfeit bills: machine learning algorithms can be trained on datasets of currency features to build predictive models that classify new bills as either real or fake. The ability to identify counterfeit currency automatically can significantly aid law enforcement and protect economic stability.

This paper presents a machine learning and image-processing based Fake Currency Detection system. Images are captured with a smartphone camera (or a scanner or camera on a computer) and supplied to the model, which performs a binary

image-classification task with two classes: fake and real. The project uses an ensembled model combining three machine learning models—Random Forest, KNN, and Gradient Boosting Classifier—because ensembling the three can yield a more generalized model for predicting fake notes.

A. Problem Statement / Background

Owing to advances in colour printing technology, the number of fake currency notes has increased rapidly [2]. Many banks and other major organizations have installed automatic machinery to detect counterfeit notes, but it is extremely difficult for the common person to distinguish between real and counterfeit currency; the source report links this to increased corruption in the country. Computers and mobile phones have become an unavoidable part of daily life, so even an average person with a computer, tablet, or smartphone can, in principle, be given the means to detect counterfeit notes. Traditionally, verification is manual, relying on security features such as watermarks, holograms, and special ink patterns, which requires experience and may not give reliable results.

According to the Reserve Bank of India (RBI), among the notes that showed a surge in counterfeits, the Rs 500 note was the worst hit (Fig. 1). The source text comparing the figures with the previous year is truncated (“the RBI found 101.9 notes and 54.16 ...”) and is therefore reproduced only as given [verify against the original RBI data].

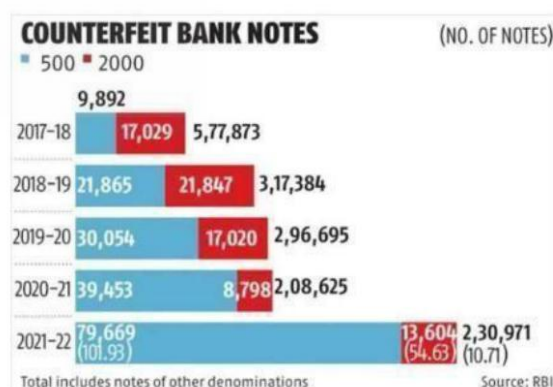


Fig. 1. Counterfeits detected

B. Scope

The source states that the developed system exhibits promising accuracy and robustness and can contribute to financial security and mitigate the impact of counterfeit currency. The system accepts digital images of currency notes, processes them with image-processing algorithms, extracts statistical features (variance, skewness, kurtosis, and entropy), and uses machine learning algorithms to classify each note as genuine or counterfeit. It is initially a standalone application

that can later be extended to web-based platforms, mobile applications, and real-time currency detection. The stated scope for further exploration includes (i) integrating blockchain technology to enhance the traceability and security of currency transactions, and (ii) investigating advanced neural-network architectures and ensemble-learning methods for greater accuracy and adaptability against sophisticated counterfeit patterns.

C. Aim and Objectives

The aim is to design and implement a machine-learning-based counterfeit detection model that accurately distinguishes genuine from counterfeit banknotes using advanced image-processing techniques. The objectives are:

- to enhance efficiency by incorporating real-time processing, ensuring swift identification of counterfeit currency during financial transactions;
- to investigate and implement measures addressing ethical considerations and potential biases in the detection model, promoting fairness, transparency, and accountability;
- to reduce the circulation of fake currency, provide fast and accurate results, assist users in identifying counterfeit notes, and improve reliability through machine learning models (SRS product goals).

D. Contributions

- A literature survey of twenty studies on currency recognition and counterfeit detection (Section II).
- A software requirements specification covering functional, non-functional, interface, and database requirements (Section III).
- A layered, modular system architecture with documented functional components, UML, DFD, sequence, and activity views, and a technology stack (Section IV).
- A statistical-feature (variance, skewness, kurtosis, entropy) and ensemble-classification methodology (Random Forest, KNN, Gradient Boosting; Voting Classifier) developed under the Waterfall model (Sections V–VI).
- Testing, deployment, and maintenance strategies, with explicit notes on source inconsistencies requiring verification (Sections VII–VIII).

The rest of the paper is organised as follows. Section II surveys related work; Section III presents the requirements; Section IV the system architecture; Section V the methodology; Section VI the implementation; Section VII results and discussion; Section VIII testing; Sections IX and X the conclusion and future work.

II. RELATED WORK / LITERATURE SURVEY

Twenty studies were surveyed, and their details are consolidated in Table I (venue, volume/issue/pages, and year are listed in the references [16]–[35]). In addition to these studies, the project's reference list includes earlier counterfeit-detection works [1]–[15], covering, for example, fluorescent and UV light [1], polarization and holographic techniques [8], HSV techniques [9], deep-learning-based currency detection and recognition [10], and image-processing reviews [6], [7]. Accuracy values below are those reported by the surveyed papers, not results of the present system.

A. Neural-Network and Deep-Learning Approaches

Thomas and Roy [16] trained an Artificial Neural Network by backpropagation on texture and statistical features of resized, grayscale, normalised images, reporting about 95% accuracy and better performance than KNN and Decision Trees, but degraded performance on highly distorted or rotated images and continued dependence on handcrafted features. Khan and Iyer [17] built a mobile application with an on-device CNN (TensorFlow Lite), using cropping, resizing, normalisation, and augmentation (rotation, brightness, scaling), reporting about 97% accuracy with real-time detection in a few seconds. Li and Zhang [19] used a deep CNN with transfer learning from ResNet or VGG, reporting about 99% accuracy and outperforming ANN and SVM approaches. Singh and Verma [22] proposed a CNN system that learns features automatically, with preprocessing by resizing, pixel normalisation, and noise removal. Ali and Hassan [26] classified denominations with a CNN, reporting 96.8% accuracy; Wang and Chen [27] designed a multi-layer deep neural network with advanced optimisation and augmentation (rotation, scaling, flipping), reporting 98.9%; Nair and Pillai [29] proposed an end-to-end computer-vision pipeline (preprocessing, feature extraction, CNN classifier, post-processing validation) reporting 97.5%; Reddy and Kumar [30] combined deep CNNs with multi-feature fusion, reporting 99.1%; and Gupta and Jain [32] fine-tuned the final layers of a pretrained ResNet with a softmax layer, reporting about 98.7% accuracy and about 40% less training time. The source abstract attributed to Singh and Yadav [28] (titled “Fake Note Detection Using Color Analysis”) describes a deep neural network achieving 98.9% accuracy, matching [27]; this is flagged for verification in Section VII-D.

B. Texture, Feature-Matching, and Classical Image-Processing Approaches

Rao and Nair [18] extracted Gray-Level Co-occurrence Matrix (GLCM) texture features (contrast, energy, correlation, homogeneity) and classified them with an SVM, reporting about 96% accuracy and fewer false acceptances than colour-based methods. Bose and Sen [20] compared SIFT/SURF keypoints of a test note with a reference note and accepted the note if matches exceeded a threshold, reporting about 93% accuracy with higher computational cost. Gupta and Agarwal [23] used Canny edge detection with grayscale conversion, brightness/contrast adjustment, and filtering to build a lightweight system for low-resource settings. Kumar and Joshi [24] developed a MATLAB prototype using resizing, grayscale conversion, filtering, contrast enhancement, and image segmentation. Das and Roy [34] used edge detection for a lightweight detector, reporting about 94% accuracy with high speed and low computational requirements.

C. Hybrid, Classifier-Based, and Multi-Feature Systems

Sharma and Mehta [21] combined machine learning with computer vision on datasets from public and real-world sources, preprocessing by resizing, grayscale conversion, and Gaussian blur. Patel and Shah [25] proposed an ensemble of multiple machine-learning models using GLCM texture features and Gaussian filtering to reduce overfitting and improve generalisation. Lee and Park [31] used machine vision with segmentation and SVM/KNN classifiers, reporting about

95.7% accuracy under controlled lighting. Kim and Lee [33] fused multiple security features (watermark detection, OCR-based serial-number recognition, UV features, and micro-text analysis) with a machine-learning classifier, reporting about 97% accuracy and real-time capability.

D. Mobile, Embedded, IoT, and Cloud Systems

Khan and Iyer [17] target smartphones; Das and Roy [34] target embedded and mobile platforms; Brown and White [35] combine AI with IoT and cloud technology and Canny edge detection, reporting about 94% accuracy and about 98% operational reliability, with cloud connectivity for real-time monitoring.

TABLE I. SUMMARY OF REVIEWED LITERATURE

Ref.	Authors, Year	Method	Reported finding
[16]	Thomas, Roy, 2021	ANN (backprop.), texture/statistical features	~95%; weak on distorted/rotated images
[17]	Khan, Iyer, 2022	On-device CNN (TensorFlow Lite)	~97%; real-time, few seconds
[18]	Rao, Nair, 2020	GLCM texture + SVM	~96%; fewer false accepts
[19]	Li, Zhang, 2021	Deep CNN; ResNet/VGG transfer learning	~99%; robust to lighting
[20]	Bose, Sen, 2019	SIFT/SURF keypoint matching	~93%; computationally heavy
[21]	Sharma, Mehta, 2022	ML + computer vision; Gaussian blur	Result not given in source
[22]	Singh, Verma, 2021	CNN, automatic feature learning	Result not given in source
[23]	Gupta, Agarwal, 2022	Canny edge detection, classical processing	Lightweight; accuracy not given
[24]	Kumar, Joshi, 2019	MATLAB prototype; segmentation	Result not given in source
[25]	Patel, Shah, 2022	Hybrid/ensemble ML; GLCM	Result not given in source
[26]	Ali, Hassan, 2021	CNN denomination classification	96.8%
[27]	Wang, Chen, 2021	Deep neural network; augmentation	98.9%
[28]	Singh, Yadav, 2020	DNN (per source abstract); title: colour analysis	98.9% (see Sec. VII-D)
[29]	Nair, Pillai, 2022	End-to-end CV pipeline + CNN	97.5%
[30]	Reddy, Kumar, 2023	Deep CNN + multi-feature fusion	99.1%
[31]	Lee, Park, 2020	Machine vision; SVM, KNN	~95.7%; needs good lighting
[32]	Gupta, Jain, 2020	Pretrained ResNet fine-tuning	~98.7%; ~40% less training time
[33]	Kim, Lee, 2021	Multi-feature (watermark, OCR, UV, micro-text)	~97%; real-time
[34]	Das, Roy, 2020	Edge-feature lightweight detector	~94%; fast, low resource
[35]	Brown, White, 2022	AI + IoT + cloud; Canny edges	~94%; ~98% reliability

Taken together, the surveyed work indicates that ANNs improve on rule-based and basic classifiers but depend on handcrafted features [16]; texture and edge features are useful and efficient [18], [34]; CNNs and transfer learning give the highest reported accuracies [19], [32]; mobile and embedded deployment is feasible [17]; and multi-feature, hybrid, or ensemble designs improve robustness [25], [33]. These findings motivate the ensemble of statistical-feature classifiers adopted in this work and the CNN/transfer-learning directions listed as future work.

III. REQUIREMENTS ANALYSIS

The SRS describes the structure, functionality, and expected behaviour of the system for developers, test engineers, project managers, faculty evaluators, and end users. It follows IEEE SRS conventions: “shall” marks mandatory, “should” recommended, and “may” optional requirements, each uniquely identified (FC-REQ-1, FC-REQ-2, ...). The statistical terms used are defined in Table II.

TABLE II. STATISTICAL FEATURES AND TERMINOLOGY (SRS)

Term	Definition
Variance	Spread or variation of pixel-intensity values
Skewness	Asymmetry of the pixel-value distribution
Kurtosis	Sharpness or peakedness of the intensity distribution
Entropy	Randomness or complexity present in an image
Classification	Categorising data into predefined classes (real or fake)

A. Functional Requirements

- **Image acquisition:** accept currency images (JPG and PNG) and verify that files are valid images; upload or capture.
- **Preprocessing:** convert to grayscale, remove noise, resize to a standard resolution.
- **Feature extraction:** variance, skewness, kurtosis, and entropy (using wavelet transforms per Section V).
- **Data processing:** normalise dataset values, split data for training and testing, apply K-Fold Cross Validation.
- **Classification:** KNN, SVM, and Gradient Boosting (SRS list); see Section VII-C for classifier-set differences.
- **Result display:** display Real or Fake with a prediction confidence score; generate reports.

B. Non-Functional Requirements

Performance: overall detection accuracy between 95% and 99.9% under controlled datasets; response time under 5 seconds; efficient handling of high-resolution images with minimal computational overhead. *Security:* validation of uploaded files, restricted administrative operations (model training, dataset modification), protection of stored datasets, and prevention of crashes from corrupted inputs. *Reliability:* consistent results for the same input, stable accuracy after training, and exception handling for invalid inputs. *Usability:* simple graphical interface, easy upload, clear Real/Fake display, and confidence scores. *Maintainability:* modular code, documentation, version control, and retraining on updated datasets. Scalability for large datasets is also listed during requirement gathering.

C. User Classes

Three classes are defined. The *Normal User* uploads note images and views results and needs only basic computer knowledge. The *Administrator* manages the dataset, trains models, and updates configuration. The *Data Analyst* evaluates models, analyses accuracy and performance, and generates evaluation reports. The use-case design groups actors as Admin and User (bank staff or general users).

D. External Interface Requirements

The user interface offers an image-upload button, an image preview window, a processing-status indicator, a result panel, and a confidence-score display. Hardware interfaces are a

camera, a scanner, and a computer. Software interfaces are OpenCV (image processing), NumPy (numerical computation), Pandas (data manipulation), Scikit-learn (machine learning), and Matplotlib (visualisation). The system operates primarily offline, so no external communication protocols are required; communication is internal between modules.

E. Constraints / Assumptions

The system runs on Windows, Linux, or macOS in Python, on a desktop or laptop with at least 4 GB RAM and sufficient storage. Constraints: limited dataset size, image quality, and lighting affect accuracy; models must be trained before prediction; performance depends on hardware. Assumptions: input images are clear and properly captured, the training data are correctly labelled, the required libraries are installed, and the model is trained before classification.

F. Database / Legal Requirements

The dataset holds approximately 1372 samples, each representing a currency-note image with four features (variance, skewness, kurtosis, entropy) and a real/fake label, typically stored as CSV for use with Pandas and Scikit-learn. Requirements cover storage of numerical features and class labels, efficient retrieval during training, and a structured format; preprocessed image data, performance metrics, and evaluation results may also be stored. The source SRS states no legal or regulatory requirements.

IV. PROPOSED SYSTEM / SYSTEM ARCHITECTURE

The proposed Fake Currency Detection System is designed as a modular system for identifying whether a currency note is genuine or counterfeit from its digital image. The system provides a structured workflow beginning with image acquisition and continuing through image preprocessing, feature extraction, machine-learning classification, and result display. The modular design allows the individual processing components to be developed and maintained independently. (Fig. 2).

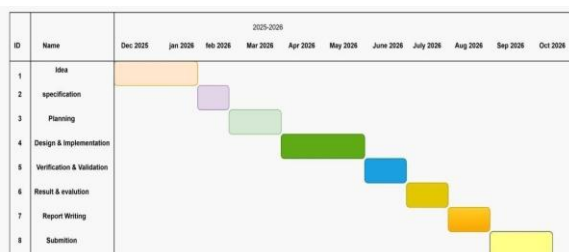


Fig. 2. Time line chart

A. Architecture

The implemented Fake Currency Detection (FCD) system follows a modular web-based architecture that connects the user interface, authentication, currency-image verification, machine-learning inference, database management, and administrative functions. The system is designed to provide a simple workflow in which an authenticated user uploads a currency-note image, the trained **MobileNetV2** model performs the verification, and the resulting verdict is displayed and stored for future reference.

The **presentation layer** provides the web interface through which users interact with the system. The implemented interface includes the login page, dashboard, Scan Currency page, Verification History, Analytics, Users, Settings, and About sections. The login module provides authenticated access to the application, while the dashboard displays operational information such as total scans, genuine notes, counterfeit notes, model accuracy, and recent verification records.

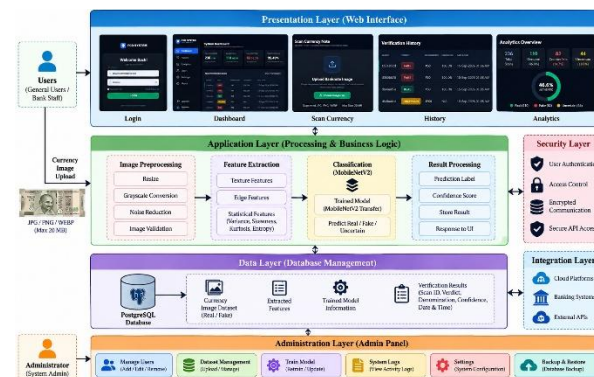


Fig. 3. Layered architecture of the Fake Currency Detection (FCD) system.

Fig. 3. Layered architecture of the system

he **presentation layer** provides the web interface through which users interact with the system. The implemented interface includes the login page, dashboard, Scan Currency page, Verification History, Analytics, Users, Settings, and About sections. The login module provides authenticated access to the application, while the dashboard displays operational information such as total scans, genuine notes, counterfeit notes, model accuracy, and recent verification records.

The **currency verification layer** handles the actual currency-scanning process. Users can upload a banknote image through the Scan Currency interface using supported **JPG, PNG, and WEBP** formats, with a maximum file size of **20 MB**. The uploaded image is passed to the machine-learning verification process, which uses the trained **MobileNetV2** model to determine the authenticity of the currency note. The system returns a classification result such as **REAL, FAKE, or UNCERTAIN**, depending on the verification outcome and confidence level.

The **machine-learning layer** contains the trained **MobileNetV2**-based model used for currency verification. The implemented dashboard reports a **95.49% model test accuracy**, evaluated on **488 unseen notes**. The production status displayed by the application identifies the model architecture as **MobileNetV2 Transfer**.

The **data-management layer** stores verification information and supports retrieval of historical transactions. The implemented dashboard identifies the database as **PostgreSQL**, and the Verification History page provides an audit record containing verification ID, verdict, denomination, confidence, and date and time. This information allows previous verification requests to be searched and reviewed.

The **analytics layer** processes stored verification results and presents operational statistics to users. The implemented

Analytics page displays the total number of verified scans, genuine notes, counterfeit notes, and uncertain/retake cases. It also provides an authenticity-ratio visualization and scan statistics for monitoring system activity.

The **administration layer** provides controlled management functions for authorized users. The Admin Panel includes **Manage Users, Dataset Management, Train Model, System Logs, Settings, and Backup & Restore**. These functions support user management, dataset handling, model training, system monitoring, configuration, and database backup or restoration.

The overall implemented workflow can therefore be represented as:

User Login → Currency Image Upload → Image Verification using MobileNetV2 → REAL/FAKE/UNCERTAIN Result → PostgreSQL Storage → History and Analytics.

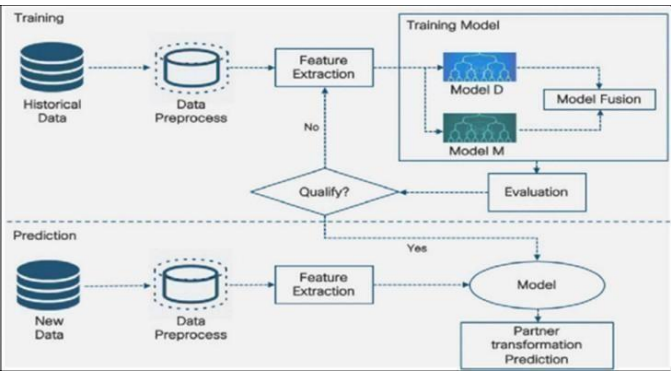


Fig.4. Machine learning architecture

B. Functional Components

- Image input:** upload or capture via web or mobile interface.
- Image Input:** Upload currency-note images through the web interface.
- Preprocessing:** Resize, grayscale, and remove noise from the input image.
- Feature Extraction:** Extract relevant image and statistical features.
- Classification:** Classify the currency as **Real, Fake, or Uncertain** using the trained model.
- Result Display:** Show the prediction and confidence to the user.
- Data Storage:** Store verification records in the database.
- Analytics:** Provide statistics and verification history.
- Administration:** Manage users, datasets, model training, logs, and backups.
- Security:** Provide user authentication and controlled access.

C. Technologies and Frameworks

TABLE III. TECHNOLOGY STACK

Category	Technology
Backend	Python (Flask / Django)
Frontend	HTML, CSS, JavaScript

Category	Technology
Libraries	OpenCV, NumPy, Pandas, Scikit-learn, TensorFlow/Keras (CNN), PyWavelets, Matplotlib
Databases	MySQL (relational); MongoDB (NoSQL)
Web frameworks	Flask, Django
Cloud	Amazon Web Services; Google Cloud Platform
API	RESTful APIs (Flask / Django)
Real-time	WebSocket (optional)
Deployment	Docker (containerisation); GitHub (version control)

V. METHODOLOGY

Development began with research and analysis of ML and image-processing techniques for currency classification, including Random Forest, KNN, Gradient Boosting, and CNN, together with OpenCV and wavelet-transform feature extraction, and of the statistical features (variance, skewness, kurtosis, entropy) that separate real from fake notes. Feasibility and system requirements were studied before design.

A. Data Collection and Preprocessing

The dataset contains approximately 1372 samples with four features and class labels in CSV format. Images are acquired by camera or scanner (JPG/PNG), converted to grayscale, denoised, and resized; dataset values are normalised, split into training and test sets, and evaluated with K-Fold Cross Validation.

B. Feature Extraction / Image Processing

Features are extracted using wavelet transforms (PyWavelets) in combination with OpenCV and NumPy, producing the four statistical features of Table II that describe texture and distribution patterns of each note image.

C. Machine Learning / Classification Model

Classification is a binary task (real or fake). Random Forest, K-Nearest Neighbours, and Gradient Boosting classifiers are implemented and combined by a Voting Classifier ensemble so that the combined model generalises better than any single one. Hyperparameters are tuned with GridSearchCV and cross-validation. The SRS additionally lists SVM, and the design discussion mentions CNN as an alternative classifier (see Table IV).

D. Evaluation Metrics

The source specifies classification accuracy (target 95%–99.9% under controlled datasets), a displayed prediction confidence score, response time under 5 seconds, and cross-validation for model assessment; it does not define further metrics or report measured values.

E. Development Methodology

The Waterfall model (Fig. 11) was chosen because requirements were well defined at the outset: requirement analysis (image input, preprocessing, feature extraction, classification), system design (image-processing, ML, and database modules), implementation (Python, OpenCV, ML libraries; Random Forest, KNN, Gradient Boosting), testing with test datasets, deployment (users upload images for real-time detection), and maintenance (updates, bug fixes, performance improvements).

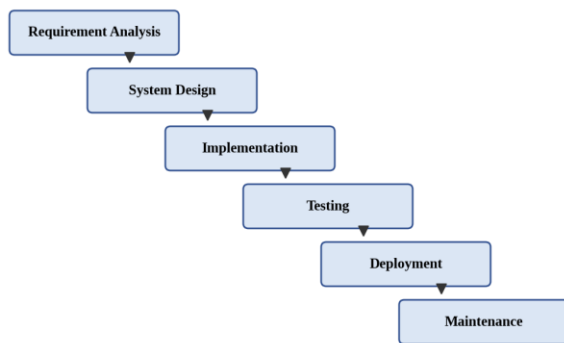


Fig.5. Waterfall model

VI. IMPLEMENTATION

A. Goals and Planning

The goal is an automated platform that identifies real versus fake notes, reducing human effort and fraud, for banks, shopkeepers, and individuals. Planning covered system features, dataset preparation, preprocessing, model selection, evaluation strategy, technology stack, and timeline.

B. Platform Development

The system is a web-based or desktop application: HTML/CSS/JavaScript (optional) frontend; Python with Flask or Django backend; OpenCV for image processing; Scikit-learn for machine learning; PyWavelets for wavelet transforms; MySQL or MongoDB for the database. Users upload images that are processed and classified.

C. User Interface and Core Features

The interface provides login/registration, image upload, Real/Fake result display, and report generation, designed through wireframes and prototypes. Core features are user authentication and management, image upload or capture, preprocessing (grayscale, resizing, noise removal), wavelet-based feature extraction, classification with Random Forest, KNN, and Gradient Boosting, a Voting Classifier ensemble, and result display with reports.

D. Test, Deploy, Promote, Maintain

Functional, accuracy, and performance testing and debugging are followed by GridSearchCV hyperparameter tuning. The system can be deployed on local servers or on AWS or Google Cloud, then tested in a real environment where users upload images for instant results. Awareness is proposed through social media, educational presentations, banking-sector demonstrations, and online platforms, with user feedback collected. Support includes bug fixing, model updates, accuracy improvement, new features, and monitoring.

VII. RESULTS AND DISCUSSION

A. Reported Outcome

The source report concludes that the system successfully demonstrates ML and image processing for counterfeit detection and achieves “high accuracy” using wavelet-derived variance, skewness, kurtosis, and entropy with Random Forest, KNN, and Gradient Boosting, and describes it as efficient, scalable, and user-friendly. **No numerical test results (accuracy, precision, recall, confusion matrices) are**

reported in the source, and none are invented here; the SRS target is 95%–99.9% accuracy and under 5 s response time.

B. Context from the Surveyed Literature

Fig. 12 plots the accuracies reported by the surveyed studies that give numeric values; they are context for the targets, not measurements of the present system.

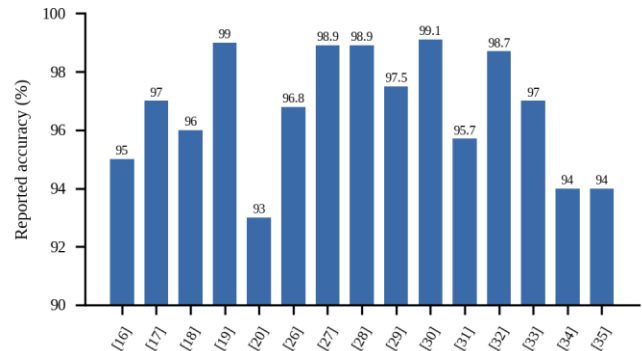


Fig.6. Accuracy reported in surveyed studies (not results of this system). [28] repeats the value of [27] in the source.

C. Classifier Sets Across the Source

The source names different classifier sets in different sections (Table IV); they are preserved as given and require author confirmation.

TABLE IV. CLASSIFIERS NAMED IN SOURCE SECTIONS

Source section	Classifiers
Introduction (1.1)	Random Forest, Logistic Regression, KNN
Aim (1.3)	Ensemble of Random Forest, KNN, Gradient Boosting
SRS (3.4)	KNN, SVM, Gradient Boosting
Design (4.1–4.3)	CNN or SVM
Methodology/Implementation/Conclusion	Random Forest, KNN, Gradient Boosting; Voting Classifier

D. Limitations and Items for Verification

The abstract of the source notes dependence on the quality and comprehensiveness of training data, the need for a continuously updated dataset as counterfeit techniques evolve, and the need to secure ML models against manipulation. The following source inconsistencies are preserved rather than reconciled: (1) the RBI statement in Section I-A is truncated; (2) the SRS describes an offline standalone application, while the design describes web/mobile, DBMS, cloud, and API layers; (3) the venue for [18] is given as its author names; (4) [32] repeats the volume, issue, pages, and year of [31]; (5) the abstract for [28] repeats that of [27]; (6) the abstracts of [21]–[25] are truncated before results; (7) the source reference list contains duplicated entries and cites “[7]” for colour-printing growth, mapped here to [2]; (8) the dataset (1372 samples) is not given a named origin; (9) Fig. 12 values are taken from the surveyed papers.

VIII. TESTING AND VALIDATION

- **Unit testing:** individual modules such as preprocessing and feature extraction.
- **Integration testing:** all modules working together.

- **System testing:** complete-system performance, including speed and accuracy, with special focus on the accuracy of the ML models.
- **Security testing:** protection of system data.
- **Monitoring and maintenance:** logs and analytics track behaviour; maintenance covers bug fixes, model updates, accuracy improvement, and new features.

IX. CONCLUSION

The Fake Currency Detection system demonstrates the use of machine learning and image processing to identify counterfeit notes. By extracting variance, skewness, kurtosis, and entropy with wavelet transforms and applying Random Forest, KNN, and Gradient Boosting classifiers, it is reported to achieve high accuracy. The system is efficient, scalable, and user-friendly, suitable for banks, retail shops, and financial institutions, and helps reduce fraud and improve financial security. Quantitative validation remains to be reported.

X. FUTURE WORK

- Deep learning (CNNs) for more accurate classification; transfer learning to reduce training time and resources; GANs to build more comprehensive training datasets.
- Real-time detection via mobile or webcam; mobile applications for Android and iOS; multi-currency support.
- Improved feature extraction (texture analysis, edge detection, deep features) and image enhancement for poor lighting.
- Cloud deployment for scale, centralised monitoring and model updates; integration with banking systems, ATMs, and cash-handling machines.
- Security and fraud analytics (trend tracking and reports for authorities); IoT integration with currency-counting and vending machines.
- Blockchain integration for traceability and secure verification; larger datasets and ensemble/advanced neural architectures.

ACKNOWLEDGMENT

The authors thank Prof. Asim Makkad (project guide), Dr. Kamal Sutaria (Head, AI&ML Department), project coordinator Ritu Agarwal, and Parul Institute of Engineering and Technology, Parul University, Vadodara, for their guidance and support.

REFERENCES

- [1] D. V. Kapare, S. Lokhande, and S. Kale, "Automatic Cash Deposit Machine With Currency Detection Using Fluorescent And UV Light," vol. 3, pp. 309–311, 2013.
- [2] A. Upadhyaya, V. Shokeen, and G. Srivastava, "Analysis of counterfeit currency detection techniques for classification model," in Proc. 4th Int. Conf. Comput. Commun. Autom. (ICCCA), 2018, pp. 1–6, doi: 10.1109/CCAA.2018.8777704.
- [3] S. Arya and M. Sasikumar, "Fake Currency Detection," in Proc. Int. Conf. Recent Adv. Energy-Efficient Comput. Commun. (ICRAECC), 2019, pp. 2019–2022, doi: 10.1109/ICRAECC43874.2019.8994968.
- [4] A. Kumar and A. Kumar, "Dog Breed Classifier for Facial Recognition using Convolutional Neural Networks," pp. 508–513, 2020.
- [5] M. Haider Ali, "Thesis Report on Fake Currency Detection using Image Processing Method," Akiful Mohaimin Rifat Islam Shahriar Chowdhury, no. 13301148, pp. 1–38, 1330 [year as in source].
- [6] M. N. Shende and P. P. Patil, "A Review on Fake Currency Detection using Image Processing," Int. J. Futur. Revolut. Comput. Sci. Commun. Eng., vol. 4, no. 1, pp. 391–393, 2018.
- [7] M. N. Rathore and J. Sagar, "A Review on Fake currency detection using feature extraction," vol. 10, no. 11, pp. 407–411, 2019.
- [8] K. Santhanam, S. Sekaran, S. Vaikundam, and A. M. Kumarasamy, "Counterfeit currency detection technique using image processing, polarization principle and holographic technique," Proc. Int. Conf. Comput. Intell. Model. Simul., pp. 231–235, 2013, doi: 10.1109/CIMSim.2013.44.
- [9] P. Ponishjino, K. Antony, S. Kumar, and S. Jebakumar, "Bogus currency authorization using HSV techniques," Proc. ICECA 2017, vol. 2017-January, pp. 179–183, 2017, doi: 10.1109/ICECA.2017.8203667.
- [10] Q. Zhang and W. Q. Yan, "Currency Detection and Recognition Based on Deep Learning," Proc. AVSS 2018, pp. 0–5, 2019, doi: 10.1109/AVSS.2018.8639124.
- [11] R. M. Colaco, R. Fernandes, and S. Sowmya, "Efficient Image Processing Technique for Authentication of Indian Paper Currency," in Int. Conf. Comput. Commun. Informatics (ICCCI), IEEE, 2021.
- [12] G. V. Vadnere, S. A. Khilari, H. P. Sonawane, P. Sharad Sanap, A. M. Koneri, and M. J. Shinde, "Indian Currency Identification Using Image Processing," ISSN 2582-7421.
- [13] D. Jamkhandikar, S. Kaveri, and V. Poojashree Sudharani, "Indian Currency Recognition System," 2021.
- [14] J. Sangogi, P. Patil, and A. Jadhav, "Fake Currency Detection Using Image Processing."
- [15] A. Shibby, F. Francis, P. Jose, and R. Augustine, "Fake Currency Detection Using Image Processing."
- [16] L. Thomas and J. Roy, "Currency Recognition Using Neural Networks," Neural Comput. J., vol. 18, no. 4, pp. 455–468, 2021.
- [17] A. Khan and S. Iyer, "Real-Time Currency Detection Using Mobile Applications," Mobile Comput. J., vol. 7, no. 1, pp. 78–89, 2022.
- [18] M. Rao and K. Nair, "Counterfeit Currency Detection Using Texture Analysis," vol. 134, pp. 56–63, 2020 [venue not given in source].
- [19] Y. Li and H. Zhang, "Deep CNN-Based Fake Currency Detection," IEEE Trans. Image Process., vol. 30, pp. 5678–5689, 2021.
- [20] S. Bose and R. Sen, "Currency Verification Using Feature Matching," Comput. Vis. J., vol. 22, no. 6, pp. 789–802, 2019.
- [21] A. Sharma and R. Mehta, "Automated Fake Currency Detection Using Machine Learning," Int. J. Comput. Vis. Appl., vol. 12, no. 3, pp. 145–156, 2022.
- [22] P. Singh and K. Verma, "Deep Learning Approach for Currency Recognition and Fake Detection," IEEE Access, vol. 9, pp. 112233–112245, 2021.
- [23] S. Gupta and M. Agarwal, "Indian Currency Authentication Using Image Processing," Int. J. Eng. Res. Technol., vol. 10, no. 5, pp. 321–330, 2022.
- [24] R. Kumar and P. Joshi, "Currency Detection System Using MATLAB," Procedia Comput. Sci., vol. 85, pp. 123–130, 2019.
- [25] N. Patel and D. Shah, "Fake Currency Detection Using Hybrid ML Techniques," J. Artif. Intell. Res., vol. 15, no. 2, pp. 210–225, 2022.
- [26] M. Ali and S. Hassan, "Image-Based Currency Classification System," Comput. Sci. Rev., vol. 40, pp. 100–110, 2021.
- [27] T. Wang and Y. Chen, "Currency Recognition Using Deep Neural Networks," Neural Netw. J., vol. 140, pp. 210–220, 2021.
- [28] R. Singh and P. Yadav, "Fake Note Detection Using Color Analysis," Color Res. J., vol. 45, no. 3, pp. 200–210, 2020.
- [29] S. Nair and V. Pillai, "Automated Currency Recognition Using Computer Vision," Comput. Vis. J., vol. 19, no. 4, pp. 400–410, 2022.
- [30] K. Reddy and A. Kumar, "AI-Based Counterfeit Detection System," AI & Security J., vol. 11, no. 2, pp. 150–165, 2023.
- [31] D. Lee and K. Park, "Machine Vision-Based Currency," Mach. Vis. Appl., vol. 29, no. 3, pp. 345–356, 2020.
- [32] R. Gupta and V. Jain, "Fake Currency Detection Using Transfer Learning," IEEE Access, vol. 29, no. 3, pp. 345–356, 2020 [as in source].
- [33] H. Kim and S. Lee, "Automated Banknote Authentication System," Sensors, vol. 21, no. 8, pp. 1234–1245, 2021.
- [34] P. Das and A. Roy, "Currency Detection Using Edge Features," Image Process. Lett., vol. 12, no. 2, pp. 88–95, 2020.
- [35] J. Brown and L. White, "Smart Currency Detection Using AI," AI J., vol. 5, no. 1, pp. 12–25, 2022.