CHAPTER 1
INTRODUCTION

1.1 Preamble

The rise of digital technology and automation has paved the way for innovative solutions to everyday challenges. One such critical area is currency recognition and value detection, which plays a significant role in industries like banking, vending machines, and services for the visually impaired. Traditionally, humans visually recognize and verify currency, but errors and fraud risks have prompted the need for automated systems. A currency recognition system using image processing techniques offers an efficient, reliable, and fast method to identify currency denominations and detect counterfeit notes. By capturing images of currency and analyzing its unique features, such as shape, color, texture, and embedded security markers, the system can accurately determine the currency type and its value. The use of image processing algorithms, such as edge detection, segmentation, and feature extraction, ensures precise detection even under varying conditions like lighting and wear. Integrating machine learning models further enhances the system's adaptability to new currencies and improves accuracy over time. In this project, user aim to develop a system that can recognize and evaluate various currencies using image processing. This automated approach will significantly benefit industries requiring high-speed, accurate currency validation and offer accessibility solutions for the visually impaired. In today's rapidly evolving digital era, automation has become an integral part of numerous industries, streamlining processes and enhancing productivity. One crucial area where automation can bring significant impact is in the identification and validation of currency. The ability to quickly recognize and determine the value of currency is essential not only in banking and financial sectors but also in retail, vending machines, and other self- service operations. Manual identification of currency is prone to errors, especially when dealing with different denominations, worn-out notes, or visually impaired individuals. Thus, the development of an efficient currency recognition and value detection system using image processing techniques is of paramount importance.

systems use advanced algorithms and techniques to accurately classify banknotes based on their visual attributes. Image processing, a subset of computer vision, has emerged as a powerful tool in this field, offering methods to analyze and extract critical information from currency images. The core concept of image processing revolves around converting an image into digital form and performing various operations to obtain relevant information. The use of image processing algorithms, such as edge detection, segmentation, and feature extraction, ensures precise detection even under varying conditions like lighting and wear. Integrating machine learning models further enhances the system's adaptability to new currencies and improves accuracy over time. For currency detection, features such as size, color, texture, and unique patterns are extracted and analyzed to differentiate between denominations. The accuracy of the system depends largely on the quality of the image and the efficiency of the algorithms used in the feature extraction and classification stages. Machine learning models can further enhance this process by allowing the system to "learn" and improve its recognition accuracy over time.

1.2 Motivation

One area that presents both significant challenges and promising opportunities is the development of efficient currency recognition and value detection systems with counterfeit checking property. These systems are particularly crucial in environments such as banking, automated vending machines, and services for the visually impaired, where accurate and quick identification of currency is essential. Banking and financial services manual currency handling in banks and financial institutions can be both time-consuming and prone to human errors. The process of counting, verifying, and processing cash transactions requires significant manpower and often leads to inefficiencies. By automating these processes through advanced image processing techniques, banks can significantly streamline transactions. Automated currency recognition systems reduce the likelihood of errors, minimize the risk of fraud, and enhance overall operational efficiency. With the ability to quickly and accurately recognize different denominations, banks can process cash transactions faster, improving customer satisfaction ,complete its all possible work so that it give result in less wait

times. Support for the Visually Impaired: For individuals with visual impairments, identifying currency denominations is a daily challenge that can hinder their independence and quality of life. Developing an automated currency recognition system that not only identifies but also vocalizes the currency value can significantly enhance their ability to navigate financial transactions with confidence. Such a system can be integrated into mobile applications or handheld devices, allowing visually impaired users to determine the value of their notes easily. This technology empowers individuals by providing them with greater autonomy in managing their finances and participating more fully in society. Vending Machines and Retail: As the world shifts towards contactless and automated retail experiences, the need for reliable currency recognition systems becomes increasingly vital. Vending machines, self-service kiosks, and other automated retail solutions require efficient methods for accepting cash transactions. By implementing advanced currency recognition technology, these systems can automatically detect and validate banknotes, ensuring that transactions are processed smoothly. This not only enhances the user experience by reducing the time spent on cash handling but also improves security by minimizing the chances of counterfeit notes being accepted. As contactless payment methods gain popularity, integrating currency recognition with these systems can create a seamless payment experience that caters to diverse consumer preferences. Overall, the integration of automated currency recognition and value detection systems has the potential to transform how financial transactions are conducted across various sectors. By enhancing efficiency, improving accessibility for individuals with visual impairments, and facilitating smoother retail experiences, these systems can contribute to a more inclusive and technologically advanced society. By automating these processes through advanced image processing techniques, banks can significantly streamline transactions.

1.3 Aim

The aim of this project is to develop an intelligent software system that can accurately detects counterfeit currency and categorizes different currencies using image processing and machine learning techniques.

1.4 Objectives

- To identify currency note using Image processing techniques.
- To design a scalable and updatable system that can adapt to new currencies and evolving counterfeit.
- To provide real time detection and user-friendly output.
- To make available to common people quickly.

1.5 Organization of Report

Chapter one starts with a basic introduction, explaining what the project is about and why it is important. It highlights the need for better counterfeit and currency recognition in places like ATMs and vending machines. Chapter two looks at prior art by studying different patents related to currency recognisition and counterfeit detection. This chapter helps to understand the existing technologies that influence currency recognition systems. Chapter three reviews research papers that discuss counterfeit currency recognition, focusing on different techniques that have been used. In Chapter four, the proposed approach and system architecture are explained. This chapter describes how the system works from start to finish, from capturing images to counterfeit detection. Chapter five lists the tools and technologies used in the project, including software like Python, OpenCV, and TensorFlow for capturing images and counterfeit. Chapter six focuses on the implemented work, detailing the different modules created for the project, such as counterfeit detection and image processing and feature extraction. Chapter seven discusses the results of the project and how well the system performed. It compares the outcomes with existing methods to show improvements and address any challenges. Finally, Chapter eight concludes the report by summarizing the key findings and suggesting future directions for the project, such as expanding the system to recognize more currencies and adding features for detecting counterfeit notes. This chapter emphasizes the importance of continued development in currency recognition technology.

CHAPTER 2
PRIOR ART

A prior art is a legal protection granted by a government to an inventor or creator of a unique invention, design, or idea. Prior art is a crucial concept in patent law that refers to any publicly available information about an invention or technology that existed before the filing date of a patent application. The primary purpose of prior art is to determine the novelty and non obviousness of an invention.

2.1 EP1302910A2, Paper currency recognition system

This patent describes a system used in machines to check paper money. LED (Light Emitting Diode). This is a small light that shines on the money when user put it into the machine. The light reflects off the money and goes to the next part of the system, the sensor. Phototransistor (Sensor)-This sensor catches the reflected light from the money. The amount of light it catches changes based on the type of money (different currencies reflect light differently). The sensor turns this reflected light into an electric signal called induction current. The stronger the light, the stronger the induction current, and vice versa. This signal is sent to the machine's computer to help it figure out if the money is real.CPU is the brain of the system. It controls everything: from the LED light to the sensor, and it decides if the money is real or fake based on the signal it gets. If the signal (induction current) is too strong or too weak, the CPU adjusts things to keep everything working properly. It uses other parts, like the MOSFET, to do this. MOSFET (Metal- Oxide-Semiconductor Field Effect Transistor-This is like a traffic cop for electric current. It controls how much induction current is sent to the CPU. If the signal is too strong, it reduces the current; if it’s too weak, it lets more current through. By doing this, the MOSFET makes sure the CPU always gets the right amount of current to read the money correctly. The system is a clever setup that constantly checks and adjusts itself so that it can always correctly identify real or fake money, even as the parts inside wear out over time. The patent also emphasizes the use of multi-spectral imaging for enhanced recognition capabilities.

2.2 US8600146B20, Method for the classification of Banknotes

The method is designed to help machines like ATMs, cash counters, or vending machines identify different types of banknotes, even when they are worn, dirty, or in any orientation. Banknote Scanning. The banknote is scanned as it passes through a machine. The scan creates a two-dimensional digital image of the note, capturing its surface details. Image Processing: The scanned image is divided into smaller sections or areas These areas represent different parts of the banknote, such as corners, edges, or specific design elements like serial numbers or watermarks. Classification: The machine compares the feature vector of the scanned banknote to a set of known banknotes that have already been classified. These known banknotes are stored in the machine's memory. To make this comparison, the system calculates a distance between the scanned banknote's feature vector and the feature vectors of known banknotes. This distance tells the system how similar the scanned banknote is to each known type. The most commonly used distance metric in this method is the Manhattan distance. In simple terms, it's the sum of the differences between corresponding features of the scanned banknote and a known banknote. The smaller the distance, the more similar the two banknotes are. For example, the system might check whether one area of the banknote is darker than another. If it is, a 1 might be assigned; if it's not, a 0 is assigned. This creates a simple code that helps quickly narrow down which banknotes are possible matches. The binary signature is compared to the signatures of known banknotes. The system uses a special version of Hamming distance(which counts how many bits are different between two binary numbers) to see how closely the scanned banknote matches known ones. Advantages Efficiency By using both the feature vector and the binary signature, the system can quickly narrow down possible matches and speed up the process of identifying banknotes. The method reduces the likelihood of misclassifying a worn or dirty banknote by adjusting for common wear patterns. In summary, this patent describes a detailed and efficient method for machines to recognize and classify banknotes, even if they are old or damaged. The system uses advanced image processing techniques to create a digital "fingerprint" of each banknote and compares it to a database of known banknotes. The classification process includes

the use of machine learning algorithms, which enable the system to accurately distinguish between genuine banknotes and counterfeit ones by learning from large datasets of banknote images.

2.3 US3618765, Counterfeit Currency Detector

The US3618765 patent describes an advanced counterfeit currency detector designed to authenticate banknotes by analyzing their physical and optical properties. This system relies on a combination of light sources and sensors to examine key characteristics of a banknote, such as transparency, reflectivity, and spectral absorption of the paper and ink. The detector works by passing light through the currency and measuring how much light is absorbed or reflected by the material. Since genuine banknotes are made with special inks and paper that exhibit unique optical properties, this method helps in distinguishing them from counterfeit notes. The use of photoelectric cells allows the system to measure light transmission and reflection at various wavelengths, ensuring a thorough examination of the currency. By leveraging these precise optical measurements, the detector can effectively identify fraudulent banknotes that do not match the standard characteristics of genuine currency. This counterfeit detection system is designed to be integrated into automated machines such as ATMs, vending machines, and banknote sorting systems. It operates by comparing the measured values against a predefined set of authentication standards, ensuring a high level of accuracy in detecting counterfeit money. This automated process eliminates the need for manual verification, making financial transactions more secure and efficient. Additionally, it reduces human errors and speeds up the validation process, which is crucial for businesses and financial institutions handling large volumes of cash. By preventing counterfeit banknotes from circulating in the economy, this system plays a vital role in enhancing financial security and reducing fraud risks. Its ability to provide real-time authentication makes it an essential tool in modern banking and commerce, ensuring that only genuine currency is accepted and distributed. It does this by using ultraviolet (UV) light to check for fluorescence in counterfeit bills and a magnetic field test to detect the presence of magnetized ink, which is found in genuine U.S. currency. The system requires little or no skill to operate, making it accessible for banks, businesses,

and the general public.The Patent concludes that the disclosed apparatus provides a simple and quick method to detect counterfeit U.S. paper currency.The detector works by scanning the currency note with an optical sensor that captures its visual characteristics.

2.4 US 2004/021344, Apparatus for recognising counterfeit currency

The prior art related to the US 2004/021344 patent consists of various conventional counterfeit currency detection methods and systems that were used before this invention. Earlier counterfeit detection systems primarily relied on manual verification techniques, where human inspectors examined banknotes based on visible security features such as watermarks, color-shifting ink, microprinting, and security threads. These methods were prone to human errors and inefficiencies, making them unreliable for large-scale financial transactions. Some semi-automated systems utilized UV (ultraviolet) light detection, which could identify fluorescent security features embedded in genuine banknotes. However, counterfeiters found ways to replicate UV-reactive materials, reducing the effectiveness of this method. Additionally, magnetic ink detection systems were employed to recognize magnetic patterns in authentic banknotes, but these systems were limited by their inability to detect high-quality counterfeit notes that mimicked genuine magnetic properties.

CHAPTER 3
LITERATURE REVIEW

The literature review serves as a foundational step that informs the research or development process by synthesizing existing knowledge and providing a clear understanding of the topic being addressed. Its role is to ensure that the project is well- grounded in the field, identifying the current state of research or practice, and showing where the project fits in the broader landscape. It establishes a theoretical or conceptual framework for your project by reviewing existing work, identifying relevant theories, models, or technologies, and building upon them. It establishes a theoretical or conceptual framework for your project by reviewing existing work, identifying relevant theories, models, or technologies, and building upon them.

3.1 Indian Currency Recognition using Neural Network Pattern Recognition Too

Mr. Viranchi et. al. (2017) Handling currency notes through machines in banks became necessary with the advancement of technology. One of the main challenges in recognizing currency was the fact that notes often became damaged, blurred, or dirty due to circulation, making it difficult for machines to identify them correctly. Additionally, security features in currency, such as intricate designs and watermarks, added another layer of complexity. To address these challenges, a system was developed to select important features from the currency, including its size, shape, and the central value, while utilizing the Canny Edge Detector for segmentation. This technique proved effective in detecting edges and isolating the key parts of the image. Once the images were processed, a neural network was employed to classify the notes. These neural networks, modeled after the human brain, were capable of learning patterns from data. In this case, the neural network was trained on 540 images of Indian currency notes, with denominations ranging from 5 to 500 rupees. The process involved converting the images to grayscale, detecting the edges, and applying mathematical operations to refine the images before feeding them into the neural network for classification. This approach effectively automated the

currency recognition process, making it highly beneficial for banking applications such as ATMs and cash counters where machines handle cash.

3.2 Currency Recognition using a Smartphone Comparision between color SIFT and grayscale SIFT algorithms

Iyad Abu Doush et. al. (2016) The study focused on developing a mobile system to recognize Jordanian currency using the SIFT algorithm. This method identified key points in images that remained consistent despite changes in image scale, rotation, or lighting. Two versions of the algorithm were tested: one utilizing color features and another relying solely on grayscale images. The system was evaluated using a dataset that included both coins and banknotes. The color SIFT approach consistently outperformed the grayscale version in terms of speed and accuracy. However, difficulties arose when images were too wrinkled, folded, or captured under poor lighting conditions. The research emphasized that incorporating color information led to better results, as many objects were challenging to classify without their color features.

3.3 A Systematic literature Review of Counterfeit Currency Detection using Modern Methods

Dr. K. Sundravadivelu et. al. (2022) The paper provides an in-depth analysis of current and emerging methodologies for counterfeit currency detection, with a primary focus on image processing techniques. Key approaches discussed include Ultra Violet (UV) detection, polarization analysis, advanced edge detection, and machine learning algorithms. Specific image processing methods such as the Canny edge detection algorithm, image segmentation, and feature extraction are examined for their effectiveness in identifying subtle differences between genuine and forged currency features. A significant emphasis is placed on the integration of these techniques into automated systems to enhance accuracy, speed, and scalability in real-world applications. The study highlights how leveraging both traditional and advanced technologies can lead to the development of robust frameworks for counterfeit detection, aiming to strengthen defences against increasingly sophisticated forgeries. Furthermore, the paper underscores the importance of continuous research and development in this field to keep pace with the evolving tactics of counterfeiters.

3.4 Counterfeit Currency Recognition using Deep Learning

Sabat Muhamad et. Al. (2024) The paper begins by highlighting the limitations of traditional counterfeit detection methods, which often rely on manual inspection and basic machine-assisted techniques. These conventional approaches are increasingly inadequate against sophisticated counterfeiting methods. The authors emphasize the need for automated, accurate, and efficient systems capable of distinguishing genuine banknotes from counterfeit ones. A significant portion of the review is dedicated to exploring various deep learning architectures employed in currency recognition. Convolutional Neural Networks (CNNs) are prominently featured due to their proficiency in image processing tasks. The paper discusses studies where CNNs have been applied to detect counterfeit currency, noting their ability to learn intricate patterns and features unique to genuine banknotes. The review also examines the use of Generative Adversarial Networks (GANs) in counterfeit detection. GANs are highlighted for their capability to generate synthetic data, which can be used to augment training datasets, thereby improving the robustness of detection models. The paper references the "Deep Money" system, which employs GANs to discriminate fake notes from genuine ones, achieving an accuracy of 80% . In addition to CNNs and GANs, the paper discusses hybrid models that combine multiple deep learning techniques to enhance detection accuracy. For instance, integrating CNNs with machine learning classifiers like Support Vector Machines (SVMs) has shown promising results in some studies. These hybrid approaches leverage the strengths of both deep learning and traditional machine learning algorithms. The authors address the challenges associated with dataset acquisition and preprocessing in counterfeit detection. They underscore the importance of using diverse and representative datasets to train models effectively. Techniques such as data augmentation and transfer learning are discussed as strategies to overcome limitations posed by limited data availability. The review highlights several case studies where deep learning models have been applied to specific currencies.

CHAPTER 4
PROPOSED APPROACH AND SYSTEM ARCHITECTURE

4.1 Proposed Approach

The proposed approach consists of multiple element transactions like Image Acquisition, Feature extraction and comparison, Texture features, and Voice output. This system is divided into two parts. The first part is to identify the currency denomination through image processing. The second part is the counterfeit currency recognition. To test the authenticity of Indian currency notes by preparing a system which takes the image of currency bill as input and gives the final result by applying various image processing and computer vision techniques and algorithms.

4.2 Preparation of Dataset

The dataset will contain the following repositories Sub dataset for Rs. 500 currency notes

i. Images of real notes

ii. Images of fake notes

iii. Multiple images of each security feature (template)– Sub- dataset of Rs. 2000 currency notes (Similar structure)

The various security features that we are considering are: (for Rs. 500 currency notes- Total 10 features)–

i. Rs. 500 in Devanagari and English script (2 features)

ii. Ashoka pillar Emblem (1 feature)

iii. RBI symbols in Hindi and English (2 features)

iv. 500 rupees written in hindi (1 feature)

v. RBI logo (1 feature)

vi. Bleed Lines on Left and right side (2 features)– Number Panel (1 feature).

4.3 Image Acquisition

Next, the image of the test-currency note is taken as input and fed it into the system. The image should be taken from a digital camera or preferably, using a scanner. The image should have a proper resolution, proper brightness and should not be hazy or unclear. Blurred images and images with less detail may adversely affect the performance of the system.

4.4 Pre-processing

Next, the pre-processing of the input image is done. In this step, first the image is resized to a fixed size. A fixed size of image makes a lot of computations simpler. Next up, image smoothening is performed by using Gaussian Blurring method. Gaussian blurring removes a lot of noise present in the image and increases the efficiency of the system.

4.5 Gray- scale conversion

Gray scale conversion is mainly used because an RGB in age has 3 channels whereas a gray image has only one channel. This makes the computation and processing on images much more easier in the case of gray scaled images.

4.6 Algorithm 1: For feature 1- 7

1) Feature detection and matching using ORB: After completing the necessary processing of the image, feature detection and matching is done using ORB. Our dataset already contains the images of different security features present in a currency note (total 10). Further, we have multiple images of varying brightness and resolutions corresponding to each security feature (6 templates for each feature). Using the ORB algorithm, each security feature is detected in the test image. To make the searching of the security feature (template image) easier and more accurate, a search area will be defined on the test currency image where that template is most likely to be present. Then, ORB will be used to detect the template in the test image and the result will be highlighted properly with a marker. This process will be applied for every image of each security feature present in the data-set and every time the detected part of the test image will be highlighted properly using proper markers.

Fig.4.1 Features in 500 currency

2) Feature Extraction: Now, using ORB location of each template has been detected in the input image within the high lighted area. The highlighted area is then cropped by slicing the 3D pixel matrix of the image. Next, we apply Gray scaling and Gaussian blur to further smoothen the image and now our feature is ready to be compared with the corresponding feature in our trained model.

3) Feature comparison using SSIM : From the previous step, the part of the test currency image which matches with each of the templates will be generated. In this method, the original template will be compared with the extracted feature and then a score will be given for the similarity between the two images using SSIM.

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Fig.4.2 Formula bar

The Structural Similarity Index (SSIM) is a scoring system that quantifies the image quality degradation that is caused by processing such as data compression or by losses in data transmission. Basically it looks for similarity between two images. It is a part of skimage library and uses the above mentioned formula to calculate similarity. It returns a value between-1 and 1. Closer the SSIM to 1, higher is the similarity. So, for every security feature, the SSIM value between each image of that security feature and the corresponding extracted feature from

the test image will be calculated. Then, the mean SSIM for each security feature is calculated and stored.

4.7 Algorithm 2: For feature 8 and 9

Every currency note contains bleed lines near the left and right edges. There are 5 lines in case of 500 currency note and 7 lines in case of Rs. 2000 currency near each of the two sides. This algorithm is being used to count and verify the number of bleed lines present in the left and right sides of a currency note. (feature 8 and 9)

1) Feature Extraction: In the first step, the region in which the bleed lines are present are extracted by cropping the image. So, a part near the left and right edges of the input currency note image is carefully extracted.

2) Image Thresholding: In the 2nd step, the image is thresholded using a suitable value. This ensures that only the black bleed lines remain on a white background and makes further processing quite easy.

3) Calculation of number of bleed lines: The 3rd step involves calculation of number of bleed lines. In this step, first we iterate over each column of the thresholded image. Then we iterate over each pixel of each column. Then, we calculate the number of black regions in each column by increasing a counter whenever current pixel of the column is white and the immediate next pixel is black. Similarly we, count number of black regions for each column, but, if the number of black regions is too large (>= 10), then that column is erroneous and it is discarded. Finally, the average count of black regions is calculated by considering the non- erroneous columns only and the result is displayed as the number of bleed lines. This count should be approximately 5 for Rs 500 currency notes and 7 for Rs 2000 currency notes.

4.8 Algorithm 3: For feature 10

For feature 10 Every currency note contains a number panel in the bottom right part where the serial number of the currency note is displayed. The number of characters present in the number panel should be equal to 9 (neglecting the space between the characters). This algorithm performs various operations and finally counts the number of characters present in the number panel.

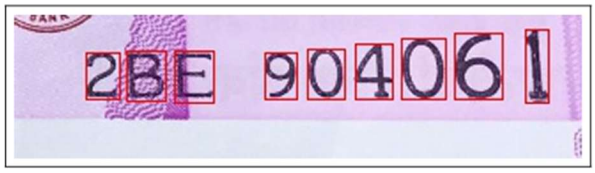

Fig.4.3 Number Panel Detection

1) Image Thresholding (with multiple values): The 1st step in this algorithm is again thresholding with suitable value so that only the black characters remain in the number panel on a white background and become easy to detect. But in this algorithm, thresholding is done using multiple values, i.e. first the image is thresholded at the initial value (90), then all other steps mentioned below are done and number of characters are calculated. After that, the threshold value is increased by 5 every time and the process of calculation of number of characters is repeated till either we reach the final value (150, in our case) or we find enough proof that 9 characters are present in the number panel.

2) Contour Detection: In the 2nd step, contour detection of the thresholded image of number panel is done.

3) Finding Bounding Rectangles: In the 3rd step, the bounding rectangle for each contour is found. The details of each rectangle is put inside a list.

4) Eliminating erroneous rectangles: The list of rectangles computed in the previous step may contain a number of erroneous and unnecessary rectangles due to noise present in the image. These erroneous rectangles need to be eliminated. So, in this step, all rectangles whose area is either too big or too small are eliminated. Then, the rectangles which are bound by a bigger rectangle are also eliminated. Finally, those rectangles which are positioned completely too high in the number panel are also eliminated.

5) Calculation of number of characters: The rectangles re maining after the previous elimination step are those rectangles which bound only each character of the number panel. The number of rectangles still remaining is calculated and this gives us the number of characters detected in that particular thresholded image. The above process is repeated for multiple threshold values (starting form 90 or 95 and increasing by 5 in each iteration). The algorithm stops if either it detects 9 characters in three consecutive iterations or if the threshold value reaches the

maximum value (150 in our case).

4.9 Displaying Output Finally: The result of all algorithms is displayed to the user. The extracted image of each feature and the various important data collected for each feature is displayed properly in a GUI window. Further, the status (Pass/ Fail) of each feature is displayed along with the details. Finally the total number of features that have passed successfully for the input image of currency note is displayed and based upon that it is decided whether the note is fake or not.

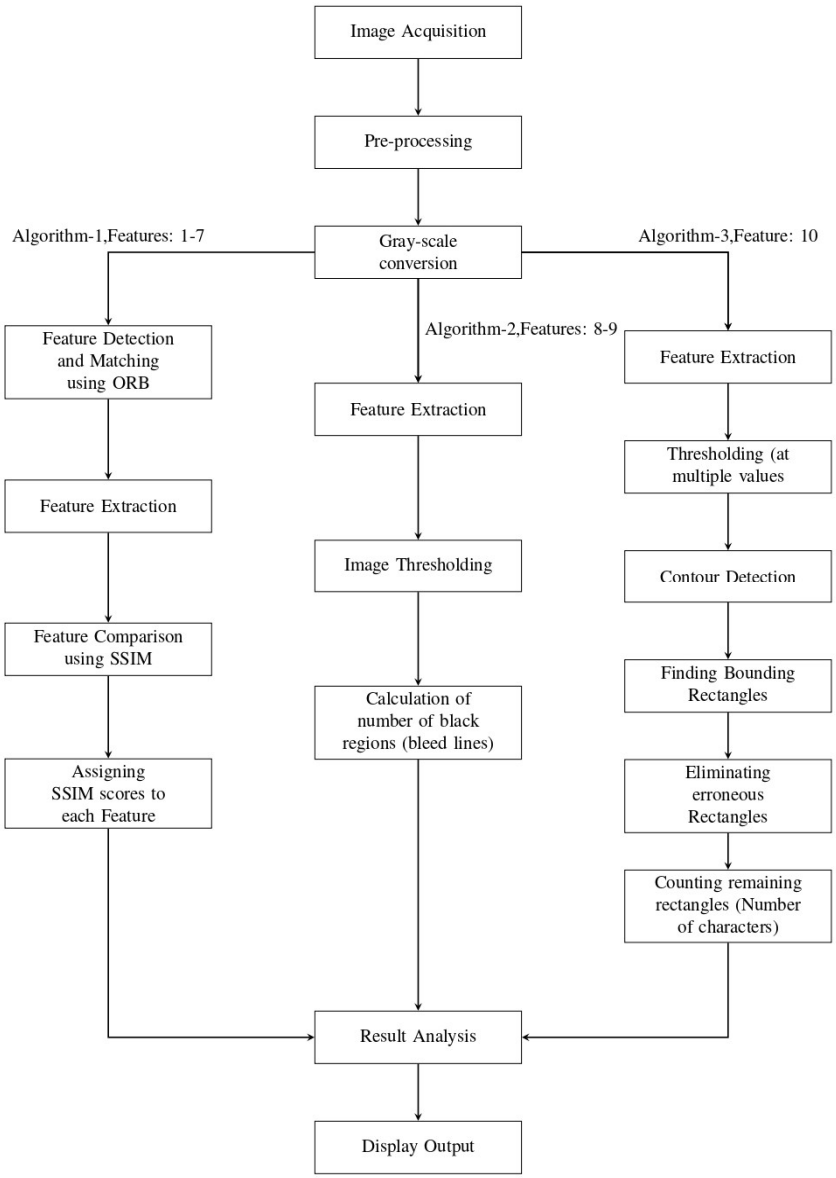

Fig.4.4 Flow diagram

4.10 Use Case Diagram Elements

Actors

- User (primary actor interacting with the system)

Use Cases

- Upload Currency Image
- Pre-process Image
- Run ORB Matching Algorithm (Algorithm-1)
- Run Bleed Line Detection Algorithm (Algorithm-2)
- Run Character Count Algorithm (Algorithm-3)
- Analyze Features
- Display Result

Explanation of Each Use Case

1. Upload Currency Image: This is the initial step where the user interacts with the system by uploading an image of a currency note (₹500, ₹2000, etc.) for authentication. To provide the system with raw input data needed for analysis. Without this step, the system has no data to process. The user may select an image file from their device using a GUI file picker or camera input.

2. Pre-processImage: Once the image is uploaded, the system performs preprocessing tasks such as noise removal, resizing, and gray-scale conversion. To prepare the image for accurate and consistent analysis across all algorithms. Preprocessing helps reduce irrelevant data and enhances features for better extraction.

3. Techniques Involved: Gray-scaling (to simplify image complexity), Filtering (to reduce noise), Resizing (to standardize dimensions)

4. Run ORB Matching Algorithm (Algorithm 1): This path involves detecting key visual features using ORB (Oriented FAST and Rotated BRIEF) and

comparing them using SSIM (Structural Similarity Index Measure). To check the structural authenticity of specific visual features like watermarks, serial numbers, or embedded security elements Steps Involved**:**

1. Detect and match features using ORB.
2. Extract descriptors from both input and reference images.
3. Use SSIM to compare the features.
4. Assign similarity scores for each matched feature.

5. Run Bleed Line Detection Algorithm (Algorithm 2): This algorithm focuses on detecting printing defects, especially "bleed lines" (unwanted ink smudges or excess print), which are common in counterfeit notes. To identify anomalies in print quality that indicate fake currency. Steps Involved:

1. Extract relevant features from the gray-scale image.
2. Apply thresholding to highlight dark regions.
3. Count the number of black regions which represent potential bleed lines.

6. Run Character Count Algorithm (Algorithm 3): This path deals with detecting and counting the number of characters (like serial numbers or printed text) using contour analysis.To verify if the expected number of characters exist in the right format and positions.

1. Extract features specific to text areas.
2. Apply multiple threshold levels for binarization.
3. Perform contour detection to locate characters.
4. Find and filter bounding rectangles.
5. Eliminate irrelevant contours (e.g., noise or symbols).
6. Count remaining valid rectangles as characters.

7. Analyze Features: Combines the results from all three algorithms and performs an integrated analysis. To make a final decision on currency authenticity by evaluating structural similarity, print quality, and text integrity. How it Works**?**

1. Aggregate SSIM scores from Algorithm 1.

2. Check black region counts from Algorithm 2.
3. Validate character count from Algorithm 3.
4. Apply decision rules (thresholds or ML logic) to determine if the note is real or fake.
5. Display Result: Outputs the result to the user (e.g., Real/Fake Note, Feature Details).

4.11 Currency Converter Module

The Currency Converter module is an optional yet valuable extension to the fake currency detection system. Its primary function is to provide users with the ability to convert the value of a validated currency note into another currency based on real-time exchange rates. This feature enhances the practical applicability of the system by integrating financial utility with image-based verification. Once the uploaded currency note is verified as genuine, the system activates the currency conversion module. The user is prompted to select a target currency (e.g., USD, EUR, GBP) from a predefined list. After selection, the system fetches the latest exchange rates using an external currency conversion API or from a locally stored exchange rate database. The conversion is computed by multiplying the denomination of the Indian currency note (e.g., ₹500 or ₹2000) with the respective exchange rate of the chosen foreign currency. The result is then displayed to the user in a readable format, along with the target currency symbol and accurate decimal precision. This module operates independently from the image processing pipeline and only activates once the classification task confirms a genuine note. It enhances user interaction and supports a broader use case, especially for travelers, traders, and educational demonstrations.

1. User uploads a currency note.
2. The note is verified using detection algorithms.
3. If the note is found to be real, the user is provided with an option to convert its value.
4. The user selects a desired foreign currency.
5. The system fetches the latest exchange rate.
6. The currency value is converted and displayed.

This module brings financial insight and practical relevance to the system by enabling real-time monetary comparison across currencies.

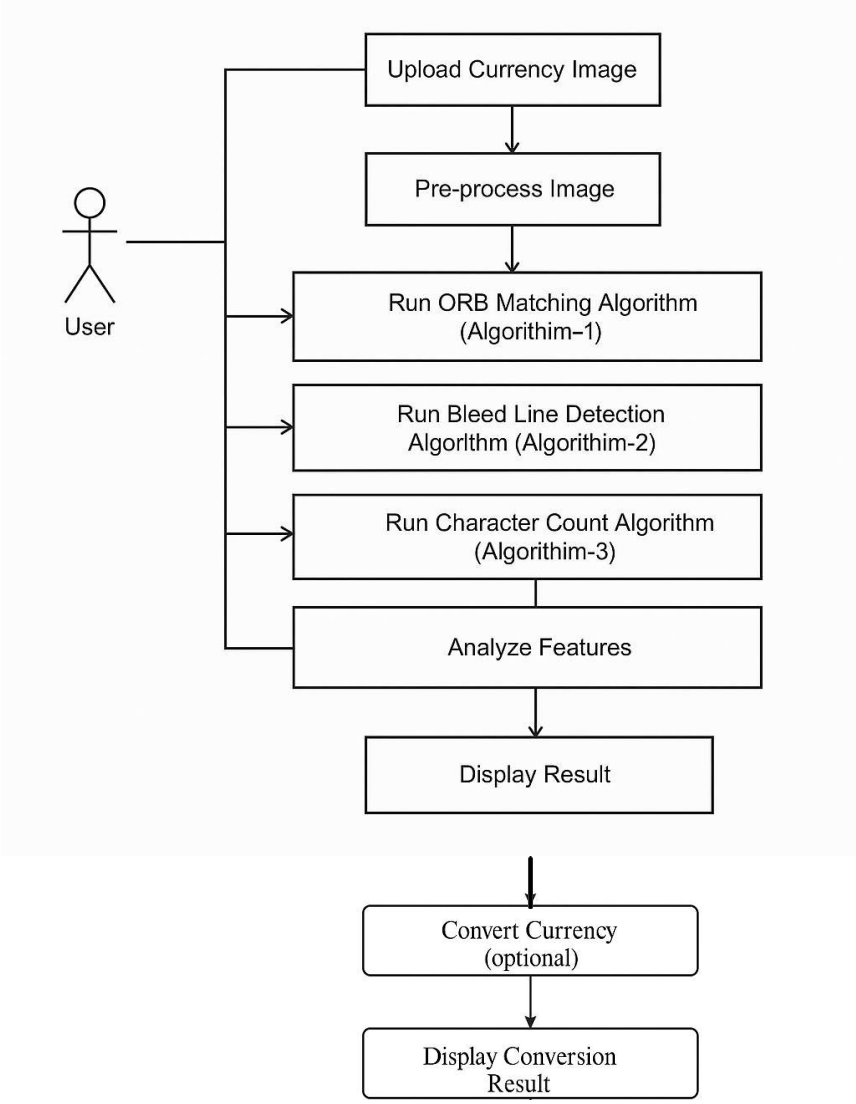

Fig.4.5 User case diagram

CHAPTER 5
TOOLS AND TECHNOLOGIES

This chapter describes the various tools and technologies employed in a project focused on currency counterfeit recognition. Python is the core programming language chosen for its versatility and extensive library support. OpenCV plays a crucial role in capturing video from webcams and executing image processing tasks, such as detecting and tracking hand gestures in real-time. Media Pipe, developed by Google, offers sophisticated hand tracking capabilities, enabling accurate gesture recognition. To interpret and classify these gestures, machine learning libraries like TensorFlow and PyTorch are utilized to train models that enhance the system's ability to comprehend a wide range of hand movements.

5.1 Python

Python is exceptionally useful for developing a system that controls cursor movement using hand gestures due to its rich ecosystem of libraries and its ease of use. Libraries like OpenCV and MediaPipe facilitate real time image and video processing, enabling the capture and analysis of hand gestures through a webcam. OpenCV allows for frame capturing, feature detection, and tracking of hand movements, while MediaPipe offers advanced hand tracking solutions. For gesture recognition, Python supports machine learning libraries such as TensorFlow and PyTorch, which can be employed to train models for identifying complex hand gestures. Once gestures are recognized, Python libraries like PyAutoGUI and Pynput can be used to translate these movements into cursor actions, including movement, clicks, and other interactions. The combination of these tools allows for rapid development and prototyping, thanks to Python's simple syntax and extensive community support. Additionally, Python’s cross-platform compatibility ensures that the system can be deployed on various operating systems with minimal adjustments. This integration of computer vision, machine learning, and system control makes Python a powerful choice for creating intuitive and responsive gesture- based cursor control systems.

5.2HTML

It acts like the building blocks for websites, telling the web browser how to display content such as text, images, links, and videos. HTML uses a system of tags, which are special codes written in angle brackets (e.g., for paragraphs or for headings), to define different parts of a web page. It is very useful for language for developing web related development. These tags come in pairs, with an opening tag to start an element and a closing tag to end it, often enclosing the content they modify. For example, to make a word bold, user would place it inside and tags. HTML by itself doesn't make web pages look visually appealing; instead, it provides the basic structure, while other technologies like CSS (Cascading Style Sheets) are used for styling, and JavaScript adds interactivity. It is a foundational skill for anyone looking to build websites or work with web technologies. This language is essential for creating web content that can be displayed correctly across different browsers and devices.

5.3 CSS

It make them visually appealing and more organized. While HTML provides the basic structure of a website, CSS controls how that structure looks, such as colors, fonts, la ts, spacing, and overall appearance. By applying CSS, user can change the background color of a web page, adjust text sizes, create borders, and arrange content into different sections. It works by selecting HTML elements through selectors (like body, h1, or p) and applying styles to them using properties (such as color, font-size, or margin). For example, if user want all the paragraphs on their page to be blue, user would use the CSS rule p color CSS is powerful because it allows user to control the look of multiple web pages from a single style sheet, making it easier to maintain a consistent design across an entire website. Additionally, CSS makes web pages responsive, ensuring they look good on different devices, like smartphones, tablets, or computers. This ability to separate content from design makes CSS an essential tool for web development, enhancing user experience and visual appeal.

5.4 Java Script

Making them more engaging and functional for users. Unlike HTML, which structures content, and CSS, which styles it, JavaScript brings web pages to life by enabling elements to change, move, or respond to user actions, like clicks, mouse movements, or keyboard inputs. For example, with JavaScript, user can create image slideshows, show or hide content when a button is pressed, validate form inputs before they are submitted, or even build interactive games. JavaScript runs directly in the web browser, meaning it doesn't need special software to work, making it a widely used language on almost every website. It uses a set of instructions called "scripts" that tell the browser how to react to different events. Developers often use JavaScript along with. Additionally, JavaScript can connect with external data sources, allowing websites to display up-to- date information without needing to refresh the page. It's a very useful language that's easy to learn for beginners but powerful enough to create complex features.

5.5 Jupyter Notebook

Jupyter Notebook as the primary development environment due to its interactive nature and ease of use. It allowed me to write and test Python code in a modular fashion using code cells, which helped in debugging and experimenting with different approaches efficiently. it used it for loading and preprocessing datasets, performing exploratory data analysis (EDA) using libraries like pandas, matplotlib, and seaborn, and for visualizing trends and patterns in the data. Jupyter's ability to display inline charts and tables made it easy to understand the underlying structure of the data and identify key features. During the machine learning phase, they trained and evaluated models inside the notebook using scikit-learn and TensorFlow, observing performance metrics in real time. it also documented each step using Markdown cells, which made the notebook self-explanatory and easy to revisit or share with collaborators. For example, in my Fake Currency Detection System, that used Jupyter Notebook to run and test image classification models, visualize accuracy/loss graphs, and perform real-time predictions on uploaded currency images.

5.6 Modules

1.Chardet

Chardet, short for "character detection," is a Python library used to detect the character encoding of text files. When working with text data, especially from different sources like websites, documents, or emails, it's essential to know the character encoding used, as it tells the computer how to interpret the bytes in a file into readable text. If user try to read a file with the wrong encoding, user might see strange symbols or errors. This is where Chardet comes in handy. Chardet helps by analyzing the byte patterns of a text file and guessing its encoding, making it easier to read the content correctly. For example, if user have a text file from an unknown source, user can use Chardet to identify how it's encoded, so user can read or process the file without any issues. This is especially useful when handling text data from multiple languages, as different languages may use different encodings. To use Chardet, user start by importing the library in their Python script. Then, user can read the file as binary data that guess. User can then use this information to decode the file properly. For instance, if user have a web scraper that collects text data from various websites, the data might be in different encodings. Chardet helps user handle this inconsistency, ensuring the text is read correctly regardless of its source. Overall, Chardet is a helpful tool when dealing with diverse text data, making it easier to manage and process files from different encodings without manual guessing.

2. Scikit-Learn

Scikit-learn (also known as sklearn) is one of the most widely used open-source Python libraries for machine learning and data analysis. Built on top of core scientific libraries like NumPy, SciPy, and matplotlib, it provides a robust and efficient set of tools for tasks such as classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. Its simple and consistent API makes it beginner-friendly while still being powerful enough for complex tasks in professional projects. Scikit-learn is widely adopted in academia, research, and industry due to its ease of use and versatility

Scikit-learn also excels in data preprocessing, which is a critical step in any machine learning pipeline. It offers tools for splitting datasets into training and testing sets (train_test_split), scaling features (StandardScaler, MinMaxScaler), handling missing values, encoding categorical data (LabelEncoder, OneHotEncoder), and constructing data pipelines.

3. Matplotlib

Matplotlib is a popular Python library used for creating visualizations like graphs, charts, and plots. It's widely used in data analysis and scientific computing to help present data in a way that's easy to understand. Instead of just looking at raw numbers or data tables, Matplotlib allows user to turn this information into visual formats, making it simpler to identify trends, patterns, and insights. The library is very flexible and supports various types of plots, such as line graphs, bar charts, scatter plots, pie charts, histograms, and more. This variety makes it suitable for visualizing all kinds of data, whether user's working with financial figures, scientific results, or general statistics. For example, if user have data about monthly sales figures, user can use Matplotlib to create a line chart that shows how sales have changed over time, helping user quickly spot increases or decreases. Using Matplotlib is straightforward. First, user need to import the library in their Python code, typically using import matplotlib. pyplot as plt. Then, user can create their plot by feeding their data into Matplotlib functions. For example, plt.plot() is used for line graphs, while plt.bar() creates bar charts. After defining their data and customizing the plot (such as adding labels, titles, or changing colors), user can display it using plt.show().Matplotlib also allows user to customize their visualizations, making it easy to change the style, size, or color of their plots. User can add titles, labels, legends, and even combine multiple plots into one figure for comparison. Overall, Matplotlib is a versatile tool that transforms complex data into clear, visual stories. It's widely used by data scientists, researchers, and analysts to present findings, making it easier for others to understand the data without diving into raw numbers.

4. Numpy

NumPy, short for "Numerical Python," is a powerful Python library used for handling and performing mathematical operations on large sets of data, especially with arrays and matrices. At its core, NumPy provides an efficient way to work with multi- dimensional arrays, which are similar to lists but much faster and more capable when dealing with large amounts of data. This makes it an essential tool for tasks in data science, machine learning, engineering, and scientific computing. One of the main advantages of NumPy is that it can handle big datasets more efficiently than standard Python lists. It allows user to perform complex calculations like addition, subtraction, multiplication, or division across entire arrays without the need for loops, making the process faster and simpler. For example, if user have two lists of numbers representing monthly sales for two different years, user can quickly use NumPy to find the difference between them, sum them up, or calculate averages. NumPy also offers a wide range of built-in mathematical functions such as trigonometry, statistics, and linear algebra, which makes it easier to carry out complex calculations. Additionally, it provides tools to generate random numbers, reshape data, and perform various mathematical transformations, all while using much less memory and processing power than standard Python methods. To use NumPy, user first import it into their Python script with import numpy as np. User can then create an array using np.array() and start performing calculations. For example, if user create two arrays, a and b, user can easily add them together with a + b or find their average using np.mean(a).Overall, NumPy is a foundational tool for data manipulation and analysis, providing a way to handle complex mathematical tasks efficiently. It’s widely used in data science, research, and any field that requires working with numerical data.

5. Pyaudio

python-dateutil is a Python library that makes working with dates and times much easier and more flexible. It builds on Python's built-in datetime module, adding extra features to handle date and time tasks that are often tricky or complex. This library is especially useful when user need to work with different date formats, time zones, or perform

operations like adding months to a date, handling recurring events, or parsing various date string formats. One of the most helpful features of python-dateutil is its ability to recognize and convert almost any date string into a proper datetime object. For example, if user have a date written as "March 25, 2024," "2024-03-25," or even "25th March 2024," the library can correctly interpret and convert it into a format that Python can understand and use for calculations. Another powerful feature is handling time zones. python-dateutil allows user to easily convert dates and times between different time zones, which is particularly useful if user're working with global data or applications that need to show the correct local time for users in different regions. This feature is built on top of the tz module within dateutil. Additionally, python-dateutil provides tools for handling recurring events. User can create rules for events that happen regularly, like weekly meetings or monthly reminders, and the library can generate all the future dates based on those rules.To use python-dateutil, user first need to install it, then import it into their script. For example, using from dateutil import parser, user can convert different date strings into datetime objects with ease.Overall, python-dateutil simplifies working with dates and times, making it a handy tool for tasks like scheduling, data analysis, time zone conversions, or any application that involves complex date manipulations. It's widely used by developers who need more flexibility and power than Python's basic datetime module provides.

6. Python-dateutil

python-dateutil is a Python library that makes working with dates and times much easier and more flexible. It builds on Python's built-in datetime module, adding extra features to handle date and time tasks that are often tricky or complex. This library is especially useful when user need to work with different date formats, time zones, or perform operations like adding months to a date, handling recurring events, or parsing various date string formats. One of the most helpful features of python-dateutil is its ability to recognize and convert almost any date string into a proper datetime object. For example, if user have a date written as "March 25, 2024," "2024-03-25," or even "25th March 2024," the library can correctly interpret and convert it into a format that Python can understand and use for calculations. Another powerful feature is handling

time zones. python-dateutil allows user to easily convert dates and times between different time zones, which is particularly useful if user're working with global data or applications that need to show the correct local time for users in different regions. This feature is built on top of the tz module within dateutil. Additionally, python-dateutil provides tools for handling recurring events. User can create rules for events that happen regularly, like weekly meetings or monthly reminders, and the library can generate all the future dates based on those rules. To use python-dateutil, user first need to install it, then import it into their script. For example, using from dateutil import parser, user can convert different date strings into datetime objects with ease. Overall, python-dateutil simplifies working with dates and times, making it a handy tool for tasks like scheduling, data analysis, time zone conversions, or any application that involves complex date manipulations. It's widely used by developers who need more flexibility and power than Python's basic datetime module provides.

7. Kiwisolver

Kiwisolver is a software library used in computer programming to help solve mathematical problems, especially those involving constraints. It is written in the Python programming language and is used to manage and solve equations where there are certain conditions (called constraints) that must be met. Kiwisolver is often used in applications where things need to be arranged in a specific way, such as user interfaces (UI) for websites, mobile apps, or desktop applications. Let's break it down more simply. Imagine user have different objects that need to be placed on a screen, like buttons or images. User want to place them in a certain order or make sure they don't overlap. But user also want to make sure that the objects can move or change size if the screen gets bigger or smaller. Kiwisolver helps make these decisions automatically by solving the equations that describe how things should behave. For example, if user're creating a website, and user want to ensure a picture stays centered, Kiwisolver can help keep the picture in the right spot, no matter how much the screen size changes. It can figure out the best way to resize and move items based on the rules user give it. Kiwisolver works behind the scenes, so user usually don't interact with it directly.

Instead, it’s used by programmers when they are creating software. It makes sure that everything follows the rules user set, like keeping certain objects in a specific position or size while the rest of the lausert adjusts smoothly. In short, Kiwisolver is a tool that helps manage the lausert and arrangement of items on screens, making sure everything fits and behaves as expected when changes occur.

8. Pywin32

Pywin32 is a Python library that allows Python programs to interact with Windows operating system features. It acts as a bridge between Python and Windows, helping developers control Windows functions using Python code. Pywin32 makes it easier to do things like automate tasks, control applications, or manage system settings on a Windows computer. Let’s break it down simply. Normally, Windows has a lot of built- in tools that let user do things like open files, create folders, send emails, or even control Microsoft Office applications like Word and Excel. But to use these tools, user usually need to write code in languages like C++ or Visual Basic. Pywin32 simplifies this by letting user use Python, which is easier to learn and write. For example, with Pywin32, user can automate tasks like sending an email through Outlook, opening a Word document, or reading data from an Excel spreadsheet—all using Python. It also allows user to interact with the Windows system, like reading system logs, checking installed software, or working with printers. One common use of Pywin32 is for automation. Imagine user need to send the same report every day. Instead of manually opening Excel, updating the data, and sending the email, user can write a Python script using Pywin32 to do all of this for user automatically. It can open the Excel file, update the information, and send the email without user having to do anything. Another use is for system administration tasks. If user're managing multiple computers, user can use Pywin32 to interact with the Windows system to control processes, manage files, or even handle Windows settings across different computers. In short, Pywin32 is a powerful tool that lets user use Python to control and automate Windows features, making tasks easier and saving time.

9. Opencv-python

OpenCV-Python is a Python library that helps user work with images and videos. It's built on OpenCV, which is a set of tools used in computer vision. Computer vision is the field that enables computers to "see" and understand images or videos. With OpenCV- Python, user can easily use these tools to make programs that recognize objects, process images, or analyze video footage. Let's make it simple: Imagine user have a picture or a video, and user want to make changes to it or find specific things, like faces, shapes, or colors. OpenCV-Python makes this easy by giving user functions user can use in Python code. It's used a lot in projects that involve artificial intelligence (AI), robotics, or anything where computers need to analyze visual information. Here are some of the ways OpenCV-Python is commonly used for Image Editing: User can load an image and apply different changes, like resizing, rotating, or changing the colors (for example, turning it black and white). This is useful if user want to clean up or prepare images for further analysis and second use is Object Detection: OpenCV can help detect specific objects in pictures or videos. For example, user can write a program to automatically find and highlight faces in a picture or identify cars in a video. This is important in security systems, self-driving cars, and other technologies that need to understand the real world and third for Video Analysis User can use it to work with video files, processing them frame by frame. For example, if user're analyzing a video from a security camera, user can detect when something moves or identify specific actions. In summary, OpenCV-Python is a powerful tool that lets user work with images and videos in Python. It helps user do things like image editing, object detection, and video analysis, making it an essential tool for projects involving computer vision and AI. OpenCV (Open Source Computer Vision Library) is a powerful tool for implementing computer vision and image processing tasks, including the recognition of currency notes. In currency recognition, OpenCV offers a variety of image processing techniques and algorithms that can be applied to accurately detect and classify different denominations of notes based on visual features. Contour detection is used to find the boundaries of objects in an image.

10. Tkinter

It is one of the most widely used libraries in Python for developing Graphical User Interfaces (GUIs). It is included in the standard Python distribution, which means it can be used without installing any additional packages. Tkinter acts as a wrapper around the Tcl/Tk GUI toolkit, providing a Pythonic interface for creating windows, buttons, labels, text inputs, menus, and various other GUI elements. Because of its simplicity and ease of use, Tkinter is often the first choice for beginners as well as for building small to medium-scale desktop applications.The fundamental idea behind Tkinter is to make user interactions more visual and intuitive by allowing users to interact with an application through graphical components rather than command-line text. With Tkinter, developers can design applications that display windows with customized sizes, icons, and layouts. These windows can contain various widgets such as labels to display text, buttons to perform actions, entry fields for user input, checkboxes, radio buttons, and even more complex elements like canvas for drawing and tree views for displaying hierarchical data. Each of these widgets can be styled and configured with various options like font, color, size, and positioning to create a pleasant and functional user experience.

- Tkinter follows an event-driven programming model. This means that the program responds to user inputs (or events) such as mouse clicks, key presses, or window resizing. The mainloop() method is at the core of any Tkinter application—it starts an infinite loop that waits for events and dispatches them to the appropriate widgets and handlers. Developers can bind specific functions or commands to GUI elements to define the behavior when a user interacts with them. For instance, clicking a button can trigger a function that processes data or updates part of the interface.

- One of the reasons for Tkinter's popularity is its simplicity. A basic GUI application can be created with just a few lines of code. For example, to create a window with a label and a button, only a handful of lines are needed. This makes it very approachable for new developers and ideal for building quick prototypes. Moreover, Tkinter supports multiple layout management techniques like pack(), grid(), and place() to position widgets within the window. Each method offers flexibility in designing interfaces

according to the needs of the application.

- Despite its simplicity, Tkinter is also capable of handling more advanced GUI requirements. It supports building multi-window applications, creating menu bars with dropdown options, incorporating message boxes for alerts, file dialog boxes for browsing the file system, and much more. It also supports the integration of images and custom fonts, allowing developers to design visually appealing interfaces. For applications that require drawing or visualizing data, the Canvas widget provides a powerful area where shapes, graphs, and even animations can be implemented.

- Tkinter is platform-independent, meaning applications built with it can run on Windows, macOS, and Linux with little or no modification. This portability makes it a reliable choice for creating desktop applications across different operating systems. Additionally, since Tkinter is a built-in library, there is strong community support and abundant documentation available online. Whether you are creating a calculator, a to-do list manager, a currency converter, or even a small game, Tkinter provides the tools necessary to bring your ideas to life.

- In conclusion, Tkinter is a highly accessible and versatile GUI library that comes bundled with Python. It enables developers to build interactive and user-friendly applications without needing to rely on external tools or complex frameworks. Its straightforward syntax, combined with powerful features, makes it a perfect choice for both beginners and experienced developers looking to add a graphical interface to their Python applications.

5.7 Algorithms

1) Feature Detection using ORB:- Feature detection is a fundamental step in image processing and computer vision that involves identifying distinct and stable patterns in an image, known as keypoints, which are used to recognize and match specific objects or regions between different images. In the context of fake currency detection, feature detection is crucial because each genuine currency

note contains unique visual elements—such as the RBI seal, Mahatma Gandhi watermark, Ashoka Pillar, and denomination scripts—that can be used to authenticate it. To detect these features in an input image of a currency note, the system employs the ORB algorithm, which stands for Oriented FAST and Rotated BRIEF. ORB is a powerful and efficient method for keypoint detection and feature description, widely recognized for its speed and robustness in real-time applications. It is particularly well-suited for this project because it is both rotation and scale invariant, meaning it can detect features accurately even if the note is tilted or appears at different sizes in the image.

- The ORB algorithm combines two well-established computer vision techniques: FAST (Features from Accelerated Segment Test) for detecting keypoints, and BRIEF (Binary Robust Independent Elementary Features) for describing those keypoints. The process begins by applying the FAST algorithm to both the input currency note image and the reference template images. FAST quickly identifies keypoints by evaluating the intensity of a pixel compared to its surrounding pixels in a circular pattern. Keypoints are typically corners or edges in the image that remain stable under various transformations. However, FAST alone is not rotation invariant. To address this, ORB adds an orientation component by computing the direction of the intensity gradient around each keypoint, allowing the system to detect features even when the image is rotated.
- Once keypoints are identified, ORB proceeds to the feature description phase using BRIEF. BRIEF creates a binary string descriptor for each keypoint by comparing the intensity of pairs of pixels in a predefined pattern around the keypoint. This descriptor effectively summarizes the local visual characteristics of the keypoint region, making it easy to compare with descriptors from other images. ORB enhances BRIEF by rotating the sampling pattern to match the keypoint's orientation, ensuring rotation invariance. The descriptors from the input image and the templates are then compared using a feature matcher—commonly the Brute Force Matcher in OpenCV—which calculates the Hamming distance between each pair of descriptors. The Hamming distance counts the number of bits that differ between two binary strings, and a lower value indicates a better match.

- In this project, ORB is applied to match specific security features of the note, such as the Ashoka Pillar or RBI symbol, between the input image and pre-stored templates. Each template is compared to the corresponding region of the test image using ORB, and if a sufficient number of keypoints are successfully matched with low error, it is concluded that the feature exists in the test note. To improve accuracy and speed, the system defines a specific region of interest (ROI) on the currency note where each feature is expected to be, thereby reducing unnecessary search and computation. Once a match is detected, the location is marked visually on the test image using colored rectangles or markers. This provides both a visual confirmation and data point for further comparison using metrics like SSIM.
- Overall, ORB-based feature detection plays a central role in verifying the authenticity of a note by checking for the presence and correctness of critical security features. Its speed, reliability, and resistance to rotation, scale, and brightness changes make it an ideal choice for currency validation systems. ORB allows the system to function effectively with a wide variety of input conditions, such as different lighting, camera angles, or image quality, ensuring that the fake currency detection remains both accurate and robust.

2) Detection of Bleed Lines :- Bleed lines are one of the distinct security features embedded in Indian currency notes, particularly in denominations of ₹500 and ₹2000. These lines are a form of raised printing and can be felt by touch. They appear as a series of short, thick vertical lines located near the left and right edges of the note. In ₹500 notes, there are typically 5 bleed lines on both the left and right sides, whereas in ₹2000 notes, there are 7 on each side. Their consistent presence, number, shape, and position make them a reliable feature for verifying the authenticity of a currency note. Algorithm 2 in this project is specifically designed to detect and count these bleed lines using computer vision techniques such as image thresholding and pixel-wise analysis.

- The algorithm starts with the extraction of the region of interest (ROI) from the input image. Since the approximate positions of the bleed lines are fixed on any Indian currency note, the algorithm defines a cropping

region near the left and right edges of the note where bleed lines are expected to be present. This significantly reduces computational overhead and improves accuracy by narrowing down the search area. The cropped region, which ideally contains the bleed lines, is then passed to the next step—image thresholding.

- Image thresholding is a basic but powerful technique in image processing that converts a grayscale image into a binary image. The purpose of thresholding here is to isolate the dark bleed lines from the lighter background of the note. The grayscale image of the cropped region is analyzed pixel by pixel, and each pixel is assigned a value of either black (0) or white (255), depending on whether its intensity falls below or above a certain threshold. A carefully chosen threshold value ensures that the bleed lines, which appear darker, remain black, while the rest of the image becomes white. This transformation makes it significantly easier to detect and analyze the lines as distinct black regions on a white background.
- Once the binary image is obtained, the algorithm performs pixel-wise counting of black regions to identify the actual bleed lines. This is accomplished through column-wise scanning of the binary image. Each column of pixels in the image is scanned from top to bottom, and the algorithm counts the number of transitions from white to black pixels. Specifically, it increments a counter every time a white pixel is followed immediately by a black pixel vertically. These transitions represent the edges or boundaries of the black regions (bleed lines). By counting how many such black regions exist in each column, the algorithm gathers data on where and how frequently these lines appear.
- However, since the image might contain noise or defects (due to lighting, shadows, or quality of the note), not all columns provide accurate readings. To handle this, the algorithm incorporates an error elimination mechanism. Columns with an unusually high number of black transitions—typically caused by noise or textured background—are flagged as erroneous and ignored in the final calculation. This ensures that only clean and valid data contributes to the line count. For instance, if a

column has more than 10 black regions, it is likely affected by noise and is excluded from further analysis.

- After filtering out noisy columns, the algorithm computes the average number of black regions across the remaining valid columns. This average provides a reliable estimate of how many bleed lines are present in the scanned region. If the average count is close to 5, the note is likely a ₹500 denomination; if it is close to 7, it likely represents a ₹2000 note. Any significant deviation from these expected counts may indicate a forged or tampered note, leading the system to flag it as potentially fake.
- The algorithm's strength lies in its simplicity, speed, and adaptability. It does not require complex machine learning or feature matching models, making it efficient for real-time use. At the same time, its reliance on well-defined visual patterns ensures accuracy, especially when the input image is clear and properly aligned. Nevertheless, some limitations exist. The method may struggle with extremely blurred or poorly lit images, or if the bleed lines are smudged, torn, or occluded. In such cases, preprocessing steps such as brightness adjustment, image sharpening, or morphological filtering could be applied to improve results.
- In conclusion, Algorithm 2 plays a vital role in the overall fake currency detection system by focusing on the bleed line feature—a tactile and visual marker of genuine Indian currency. Using image thresholding to simplify the image and a column-wise scanning approach to count black regions, the algorithm accurately identifies the number of bleed lines present on the note. This result is then used as one of the ten feature checks that determine whether a note is real or counterfeit. The integration of this method into a larger system provides both robustness and confidence in detecting fraudulent currency with minimal manual intervention.

3) Feature Comparison Using SSIM (Structural Similarity Index Measure):

In the process of fake currency detection, accurately determining whether a specific security feature is genuine or counterfeit requires a reliable and perceptually relevant comparison method. Simple image comparison

techniques, such as pixel-by-pixel subtraction or correlation, often fall short due to their sensitivity to noise, lighting variations, and slight misalignments. These limitations make them ineffective in real-world scenarios where test images may be captured under different environmental conditions. To overcome these challenges, the Structural Similarity Index Measure (SSIM) is used in this system. SSIM is a powerful and widely accepted image quality assessment metric that evaluates the visual similarity between two images by focusing on the aspects of human perception that matter the most: brightness, contrast, and structure. It serves as a reliable tool to compare extracted features from a test currency image with known templates of authentic notes.

- The core idea behind SSIM is to model the human visual system's sensitivity to structural information in images. Unlike traditional metrics that measure absolute differences, SSIM aims to quantify how closely two images resemble each other in a way that aligns with how a human would judge the similarity. It does so by decomposing image comparison into three independent components—luminance, contrast, and structural comparison. Luminance refers to the brightness level of the image and captures the mean pixel intensity. Contrast assesses how pixel values are spread or varied across an image. Lastly, structure focuses on patterns or textures formed by the spatial relationship between pixels. These three components are then combined into a single similarity score.
- Mathematically, SSIM is defined by the following equation:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Where μx\mu_xμx and μy\mu_yμy are the average pixel intensities (luminance) of images xxx and yyy, σx\sigma_xσx and σy\sigma_yσy represent their standard deviations (contrast), and σxy\sigma_{xy}σxy is the covariance between the two images (structure). Constants c1c_1c1 and c2c_2c2 are included to stabilize the division in case the denominator is

very small. The output of this function is a score between -1 and 1, although in practice it usually ranges between 0 and 1, where 1 indicates perfect similarity and 0 or negative values indicate low or no similarity.

- In the fake currency detection system, SSIM is used during Algorithm 1, which is responsible for verifying Features 1 to 7 of the note—such as the Ashoka Pillar, the RBI logo, denomination in Devanagari script, and other security elements. After feature detection is performed using ORB (Oriented FAST and Rotated BRIEF), the system extracts the region of the test image where a particular feature is likely to be present. This region is then preprocessed—converted to grayscale, resized, and smoothed using Gaussian blur—to prepare it for similarity comparison. The system has a collection of template images for each feature, captured under various lighting conditions, scales, and angles, to mimic the diversity of real-world input.
- Each extracted feature region from the test image is then compared to every corresponding template using the SSIM function. This results in a set of SSIM scores, one for each comparison. To make the comparison more robust, the system does not rely solely on a single SSIM score. Instead, it calculates two important metrics: the maximum SSIM score and the average SSIM score across all comparisons for that feature. The maximum SSIM score indicates the best possible match among the templates, while the average SSIM score gives an overall measure of consistency and reliability in the comparison. If either of these values crosses a certain predefined threshold—typically around 0.75 to 0.85—the system concludes that the feature in the test note matches the genuine template, and it is marked as "Pass."
- This process is repeated for all the features under Algorithm 1. For instance, the RBI logo on a test ₹500 note is compared to six different templates of the same logo stored in the dataset. Each template is evaluated against the extracted logo using SSIM, and a decision is made based on the computed scores. This method provides a more flexible and tolerant approach than rigid pixel-wise matching, making the system

resilient to minor noise, shadows, and imperfections that often appear in scanned or photographed images.

- The use of SSIM also allows the system to incorporate tolerance thresholds. For example, if a feature has a maximum SSIM of 0.88 but an average of 0.72, the system might still pass it, understanding that one comparison was exceptionally good. On the other hand, if all SSIM values are consistently low (e.g., below 0.5), it indicates the feature is likely missing or significantly distorted, possibly due to forgery or damage, and the feature fails the check. This scoring-based logic allows the system to make intelligent and human-like decisions in evaluating feature authenticity.
- The choice of SSIM over other metrics is justified by its strong correlation with perceptual quality and structural fidelity. For fake currency detection, structural accuracy is crucial. A counterfeit note may replicate color or intensity to some degree, but it often fails to reproduce the exact structural detail of genuine features. SSIM, by comparing patterns and textures, can expose such differences more effectively than mean squared error (MSE) or peak signal-to-noise ratio (PSNR), which are purely numerical and often misrepresent perceived quality.
- In conclusion, SSIM plays an essential role in enhancing the accuracy and reliability of the currency verification process. By evaluating luminance, contrast, and structure in a perceptual manner, it bridges the gap between raw computational image comparison and human-like visual assessment. In the fake currency detection system, it serves as the final validation step for key features, ensuring that only the most accurate and structurally correct matches are accepted. Its use of multiple templates, average and maximum scoring, and tolerance for minor variations makes it highly effective in real-world applications. Thus, SSIM not only contributes to the system's robustness but also significantly improves its overall decision-making capability, ensuring higher accuracy in detecting both genuine and counterfeit notes.

CHAPTER 6

IMPLEMENTATION

The implementation of the Fake Currency Detection System is structured around a GUI-based workflow that guides the user from image acquisition to final classification. The system begins with image acquisition, where the user selects a currency note image using a Tkinter interface (gui_1.ipynb). The selected image is read using OpenCV and displayed on the canvas after proper resizing and color channel conversion. Once the image is loaded, it undergoes preprocessing which includes operations such as resizing, grayscale conversion, and noise removal to prepare it for feature extraction. In the feature extraction phase, the system utilizes advanced techniques (possibly CNN-based layers or keypoint descriptors like SIFT, depending on the model) to identify essential patterns and textures on the currency note. These features are then fed into a trained deep learning model—developed and tested in the controller.ipynb or 500_Testing.ipynb/2000_Testing.ipynb—that performs classification to determine whether the currency note is genuine or counterfeit. The result is then displayed to the user through the GUI. The system also supports multiple denominations, allowing the user to specify whether the note is ₹500 or ₹2000, ensuring the correct model is used for evaluation. This modular implementation not only improves user interaction but also simplifies the backend logic by separating GUI handling, image processing, and model prediction into distinct notebooks and functions.

6.1 Methodolgy

6.1.1 Image Acquisition: Image acquisition is the process of capturing or obtaining an image to be used for further analysis in an image processing or computer vision system. It serves as the first step in such systems, where a camera or file selection interface is used to gather the input image. In the context of a fake currency detection system, image acquisition typically involves selecting a scanned or photographed image of a currency note through a graphical user interface (GUI). This image is then passed through various stages like pre-

processing, feature extraction, and analysis to determine its authenticity. Accurate image acquisition ensures that the input is clear, correctly formatted, and suitable for processing, making it a critical component for reliable system performance.

```
def select_image():
    global canvas
    global path

    canvas.delete("all")
    path = tkFileDialog.askopenfilename()

    if len(path) > 0 and path[-4:] == '.jpg':
        image = cv2.imread(path)
        original_image = image.copy()
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = cv2.resize(image, (675, 300))
        image = Image.fromarray(image)
        image = ImageTk.PhotoImage(image)

        canvas.image = image
        canvas.create_image(0, 0, anchor=NW, image=image)
    else:
        messagebox.showinfo("Warning", "Please choose an image! (It should be a .jpg file)")
```

Fig.6.1 Image aquisition

6.1.2 Preprocessing

Preprocessing plays a vital role in transforming the raw input image into a cleaner and standardized format, which is essential for reliable feature analysis and detection. The process starts when the user selects an image, typically a .jpg file of a ₹500 or ₹2000 currency note. This image is then resized to a fixed dimension (e.g., 1165x455 pixels) to maintain consistency across all samples. Such uniformity is important for accurate comparison and feature alignment. After resizing, a Gaussian blur is applied to the image to smooth out noise, such as minor creases or ink irregularities, which might otherwise interfere with detecting key currency features. The blurred image is then converted to grayscale, simplifying the image by focusing only on intensity values rather than color information. This grayscale image is more suitable for subsequent stages like thresholding, contour detection, and ORB (Oriented FAST and Rotated BRIEF) feature extraction. Additionally, in your project, the preprocessed image is displayed on the GUI canvas, and a progress bar is updated to provide visual

feedback to the user, indicating that the preprocessing stage is successfully completed. This step ensures that all currency images undergo a uniform preparation process, which enhances the accuracy and robustness of the fake currency detection algorithms applied later in the pipeline.

```
# Resizing the image
test_img = cv2.resize(test_img, (1165, 455))

# Gaussian Blur to smooth the image
blur_test_img = cv2.GaussianBlur(test_img, (5, 5), 0)

# Convert the image to grayscale
gray_test_image = cv2.cvtColor(blur_test_img, cv2.COLOR_BGR2GRAY)

# Function to visualize the preprocessing result
def preprocessing():
    # Display the processed image
    plt.imshow(gray_test_image, 'gray')
    plt.title('Input image after pre-processing')
    plt.show()

    # Update progress bar
    progress['value'] = 5

    ProgressWin.update_idletasks()
```

Fig.6.2 Preprocessing

6.1.3 Gray-scale conversion

significantly enhances the efficiency and accuracy of the detection process. By converting the input color image to gray-scale, the system reduces the image to a single channel containing only intensity values, which simplifies computations and speeds up processing. This is especially important for the ORB (Oriented FAST and Rotated BRIEF) feature matching used in your project, as it relies solely on intensity differences rather than color information. Additionally, gray-scale images improve the effectiveness of techniques like thresholding and contour detection, which are used to identify critical features such as bleed lines and printed characters on the currency note. Since color does not play a major role in identifying fake notes, removing it helps focus the analysis on structural and textual patterns that are vital for accurate detection. Overall,

```
# Apply Gaussian Blur before conversion
blur_test_img = cv2.GaussianBlur(test_image, (5, 5), 0)

# Convert the image to grayscale
gray_test_image = cv2.cvtColor(blur_test_img, cv2.COLOR_BGR2GRAY)

# Show the grayscale image (optional visualization)
cv2.imshow("GrayScale Image", gray_test_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig.6.3 Gray-scale-conversion

6.1.4 Scale Invariant Feature Transform

In counterfeit detection, the SIFT (Scale-Invariant Feature Transform) algorithm plays a crucial role in identifying genuine currency notes by extracting and matching distinctive features. Each currency note contains specific patterns, textures, and symbols that are unique and difficult to replicate exactly. SIFT detects keypoints on a note image—such as fine lines, watermarks, micro-text, or unique logos—and creates robust descriptors that remain consistent even if the image is scaled, rotated, or partially obscured. By comparing the keypoints of a test note with those from a database of genuine notes, the algorithm can determine whether the features align accurately. A genuine note will show a high number of good matches, while a counterfeit note will have few or mismatched keypoints. This makes SIFT a reliable tool in fake currency detection systems, especially when combined with real-time image processing and machine learning models to automate and enhance accuracy. It analyzes the note by detecting keypoints such as embedded designs, security threads, and watermark patterns that are typically consistent in genuine notes. These features are then described using robust vectors that are invariant to scale, rotation, and lighting, making the system reliable under different scanning or capturing conditions. When a currency note is scanned or photographed, SIFT compares its keypoint descriptors with those stored in a database of authentic notes.

6.1.5 Oriented FAST and Rotated BRIEF

Fake Currency Detection System, Algorithm 1 is responsible for verifying Features 1 to 7, which are critical visual elements found on genuine Indian currency notes. These features include the Mahatma Gandhi image, RBI seal, Ashoka Pillar, currency number, guarantee clause, governor's signature, and the watermark area. The system uses the ORB (Oriented FAST and Rotated BRIEF) feature detection technique to identify and compare these regions between the test currency image and reference images of authentic notes. Each region corresponding to a feature is first preprocessed by resizing, applying Gaussian blur, and converting it to grayscale. ORB then detects keypoints and computes descriptors in both the test and reference images. These descriptors are compared using a Brute Force Matcher to find the number of matching features. If the number of matches for a given feature exceeds a defined threshold, it is marked as "matched." This process is repeated for all seven features, and the matching results are stored. If most of the features are successfully matched, the note is likely to be genuine; otherwise, it is classified as counterfeit. This algorithm provides a robust structural comparison approach, helping the system detect fake notes based on physical design consistency.

```
import cv2
# Load and preprocess the test image
test_image = cv2.imread("test_note.jpg")
resized_test = cv2.resize(test_image, (640, 300))
gray_test_image = cv2.cvtColor(resized_test, cv2.COLOR_BGR2GRAY)

# List of original feature images (templates for features 1-7)
original_features = [
    "Features/1_gandhi.jpg",
    "Features/2_rbi_seal.jpg",
    "Features/3_ashoka.jpg",
    "Features/4_currency_number.jpg",
    "Features/5_guarantee.jpg",
    "Features/6_signature.jpg",
    "Features/7_watermark.jpg" ]
```

Fig.6.4 Oriented FAST and Rotated BRIEF

6.1.6 Bleed Line Detection Algorithm

Features 8 and 9 are evaluated using traditional image processing techniques rather than feature matching. Feature 8, the Bleed Lines Check, focuses on detecting the presence of thin, vertical black lines typically located on the edge of genuine ₹2000 currency notes. The system extracts a specific region from the test image where these lines should appear and converts it to grayscale. It then applies thresholding using cv2.THRESH_BINARY_INV to highlight dark elements on a light background. Contours are detected, and the count of these lines is analyzed. If the number of contours exceeds a defined threshold, it confirms the presence of bleed lines, marking the feature as genuine. Feature 9, the Characters Check, works similarly by extracting another predefined region of interest from the note that usually contains printed alphanumeric characters. After grayscale conversion and thresholding, the number of character-like contours is counted. If sufficient characters are detected, this feature is also marked as present. Together, these features add another layer of verification by checking for fine-print details that are difficult to replicate in counterfeit notes, thus improving the system's robustness.

```
# Crop region where bleed lines are expected
bleed_roi = gray_test_image[0:150, 600:640]  # Example coordinates

# Threshold the region
_, thresh_bleed = cv2.threshold(bleed_roi, 127, 255, cv2.THRESH_BINARY_INV)

# Find contours
contours, _ = cv2.findContours(thresh_bleed, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

# Decision logic
if len(contours) > 10:  # Threshold may vary
    print("Bleed Lines Detected")
    feature_8 = 1
else:
    print("Bleed Lines Not Detected")
```

Fig.6.5 Bleed Line Detection Algorithm

6.1.7 Character Count Algorithm

Feature 10 focuses on detecting the presence of a latent image—a subtle visual security element printed on genuine Indian currency notes, particularly the ₹2000 note. This latent image is generally visible only when the note is tilted at a specific angle or viewed under specific lighting conditions. Since capturing such physical effects digitally is challenging, the system simulates this process by isolating the latent image region using predefined coordinates and applying image enhancement techniques such as grayscale conversion and thresholding. The goal is to highlight any latent text or symbols that may be faintly visible. After processing, contour detection is applied to count the number of distinct features in the region. If the number of detected contours crosses a set threshold, the latent image is considered present, indicating authenticity. This step adds another layer of verification by checking a detail that is hard to forge accurately in counterfeit notes.

```
# Crop the region where the latent image is usually found
latent_roi = gray_test_image[90:160, 260:350]  # Adjust these coordinates as

# Apply Gaussian Blur to smooth the region
blur_latent = cv2.GaussianBlur(latent_roi, (5, 5), 0)

# Apply binary thresholding
_, thresh_latent = cv2.threshold(blur_latent, 127, 255, cv2.THRESH_BINARY_

# Find contours in the thresholded latent region
contours, _ = cv2.findContours(thresh_latent, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

# Decision logic based on contour count
if len(contours) > 5:  # You may tweak the threshold for accuracy
    print("Latent Image Detected")
    feature_10 = 1
else:
    print("Latent Image Not Detected")
    feature_10 = 0
```

Fig.6.6 Character Count Algorithm

6.1.8 Result Analysis

Result Analysis section (as described in the final pages of your report PDF), the system's performance is evaluated by testing multiple genuine and counterfeit currency notes, particularly ₹2000 and ₹500 denominations. Each test case involves extracting and verifying a set of 10 distinct security features such as Gandhi watermark, RBI seal, bleed lines, latent image, and more. For each note, the system runs a sequence of image processing and feature matching algorithms to identify the presence or absence of these features. The results showed a high accuracy in detecting genuine notes when most or all features were present, while counterfeit notes were flagged due to missing or poorly replicated features. The feature-wise binary output (1 for detected, 0 for not detected) is totaled, and if a predefined threshold (e.g., 6 or more features) is met, the note is marked genuine; otherwise, it is declared fake. The analysis also emphasizes how combining ORB-based visual feature matching with classical image processing techniques enhances the reliability of detection.

```
# Collect all feature results (1 for detected, 0 for not detected)
 features = [feature_1, feature_2, feature_3, feature_4, feature_5,
        feature_6, feature_7, feature_8, feature_9, feature_10]

 # Count how many features were correctly detected
 genuine_features = sum(features)

 # Define the threshold for a note to be considered genuine
 threshold = 6  # This can be adjusted based on accuracy testing

 # Decision logic
 if genuine_features >= threshold:
    result = "Genuine Currency Note"
 else:
    result = "Fake Currency Note"

 # Display the final result
 print("Feature Match Summary (1 = Detected, 0 = Not Detected):", features)
 print("Total Features Detected:", genuine_features)
 print("Final Verdict:", result)
```

Fig.6.7 Result analysis

6.1.9 Display Output

section, as outlined in the report PDF, the system presents a clear and structured result after analyzing an input currency note image. Once all ten features have been processed—through both ORB feature matching and traditional image processing techniques—their detection statuses are compiled into a list of binary values, where 1 denotes a successfully detected feature and 0 denotes a missing one. This binary list is then used to compute the total number of valid security features found in the test image. If the number of detected features crosses a predefined threshold (such as 6 out of 10), the note is declared "Genuine Currency Note"; otherwise, it is flagged as "Fake Currency Note." This result is printed clearly in the console or GUI window along with the list of feature matches, allowing the user to visually confirm which security checks passed or failed. This display output approach not only informs the user of the final verdict but also enhances transparency by showing detailed detection insights that contributed to the decision, making the system more user-friendly and reliable.

```
# Final feature results collected from each detection step
features = [feature_1, feature_2, feature_3, feature_4, feature_5,
           feature_6, feature_7, feature_8, feature_9, feature_10]

# Count the number of features detected (value 1)
genuine_features = sum(features)

# Set a threshold: minimum number of features required to consider the note
genuine
threshold = 6

# Determine the result based on the number of detected features
if genuine_features >= threshold:
   result = "Genuine Currency Note"
else:
   result = "Fake Currency Note"

# Display detailed output
print("🔍 Feature Detection Status (1 = Detected, 0 = Not Detected):", features)
print("✅ Total Features Detected:", genuine_features)
print("📝 Final Result:", result)
```

Fig.6.8 Diplay output

CHAPTER 7
RESULTS AND DISCUSSION

The results section provides the final phase of the currency recognition system, user primarily focused on building a robust model capable of classifying different counterfeit denominations of Indian currency notes and convert one currency value to another.

7.1 Model Performance

Performance The proposed system authenticates the input image of cur rency note through image processing. The input image passes through various algorithms in which the image is processed and each extracted feature is thoroughly examined. The results are calculated in the following manner.

7.1.1 ORB Matching Algorithm: This algorithm finally col lects the average SSIM score and the max. SSIM score for each feature. A feature passes the test and is real if the average SSIM score is greater than a minimum value (this value has to be decided after proper testing). A feature also passes the test if the max. SSIM score is too high (probably greater than 0.8).

7.1.2 Bleed Line Detection Algorithm: This algorithm finally returns the average number of bleed lines present in the left and right sides of a currency note. Each feature passes the test if the average number of bleed lines is closer to 5, in case of Rs 500 currency note, and 7, in case of 2000 currency note.

7.1.3 Character Count Algorithm: This algorithm finally returns the number of characters present in the number panel of the currency note. This feature passes the test if the number of characters detected is equal to 9 (for at least one threshold value).

7.2 Performance Analysis

The performance analysis of the proposed system was carried out using various images of currency notes. As we had already created a dataset for both fake and real currency notes of denominations of 500 and 2000, all the notes were tested and then the accuracy was calculated. For the sake of calculating the accuracy, it was assumed that if the currency note passed at least 9 features out of 10 then the

note is real otherwise it is fake. Testing of both real andfake notes was done separately.

1. For testing of real notes, 9 real notes for Rs.2000 and 10 real notes for Rs. 500 were considered, out of which 15 of the total 19 notes gave the correct desired results. Accuracy: 79%.

2. Similarly, for testing fake notes, 6 fake notes were taken into consideration for each denomination (12 notes in total), for which 10 of the 12 notes the correct required output. Accuracy: 83%.

Finally, the result of all algorithms is displayed to the user. The extracted image of each feature and the various important data collected for each feature is displayed properly in a GUI window. Further, the status (Pass/ Fail) of each feature is displayed along with the details. Finally the total number of features that have passed successfully for the input image of currency note is displayed and based upon that it is decided whether the note is fake or not. The entire GUI is programmed in python itself using tkinter library.

7.3 Result analysis

1) A large rectangular frame is provided in the center to display the uploaded currency note image for visual confirmation. At the bottom of the interface, there are three buttons: “Select an image” to upload a currency image from the device, “Submit” to process the selected image for verification, and “Exit” to close the application.

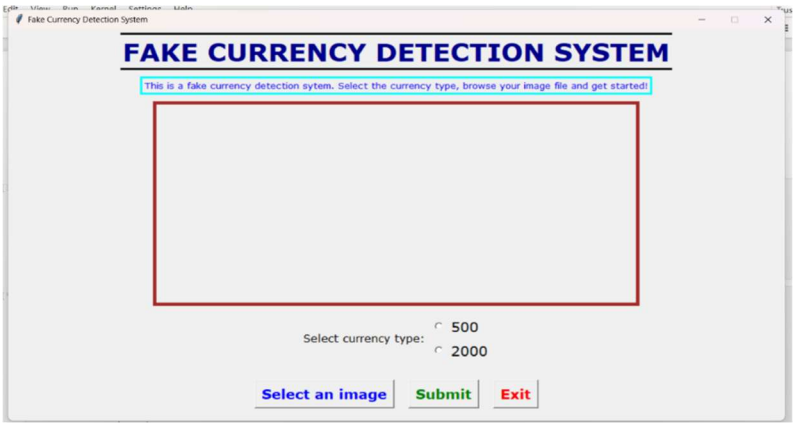

Fig.7.1 Initial image

2) The user can choose any one of these images to open and analyze within the

application. The “Open” and “Cancel” buttons at the bottom let the user either proceed with the selected file or cancel the operation..

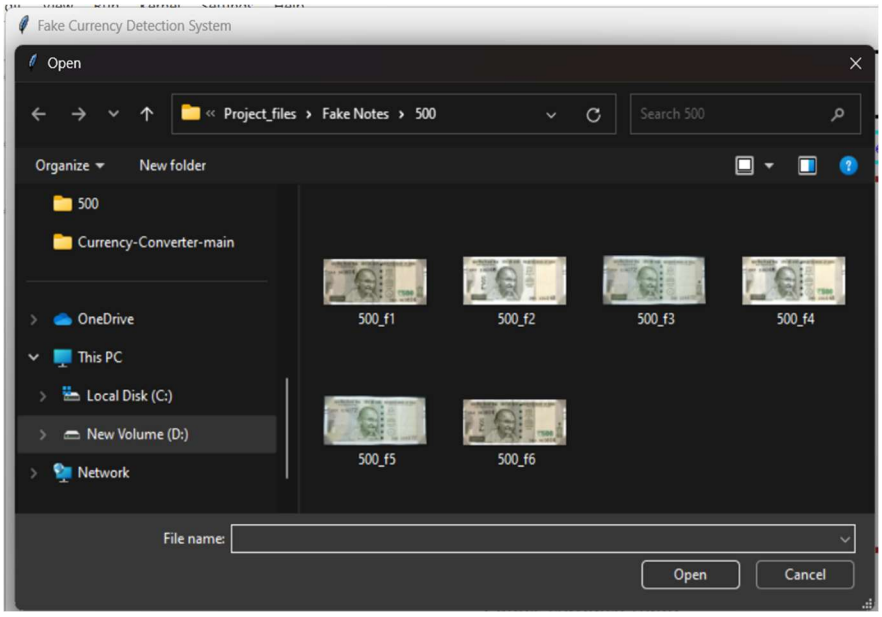

Fig.7.2 Window page

3) The radio buttons allow users to specify which denomination they are testing for authenticity. Once everything is set, the user can click the "Submit" button to begin the fake currency detection process..

Fig.7.3 Input image

4) this window appears during the analysis phase of the Fake Currency Detection System. It informs the user that the system is currently processing the uploaded currency image. A green progress bar visually indicates that the detection process is ongoing. The text "Processing! Please wait..." assures the user that the system is actively working. Once processing is complete, the user can click the "Click to continue" button to view the result..

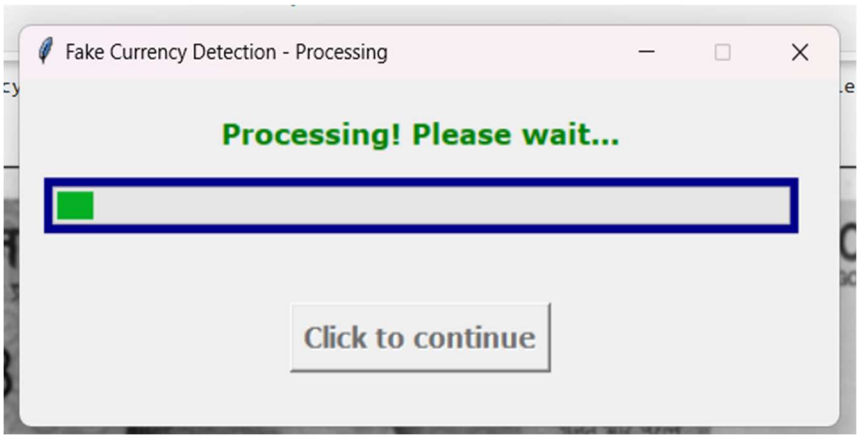

Fig.7.4 Processing

5) This is the result window of the Fake Currency Detection System, showing the outcome of the analysis. The selected ₹500 note is displayed at the center for confirmation. The result clearly states that all 10 out of 10 features used for validation have passed, indicating the note is genuine. The use of bold green text reinforces the positive outcome visually. This window provides users with clear, easy-to-understand feedback on the authenticity of the uploaded currency note..

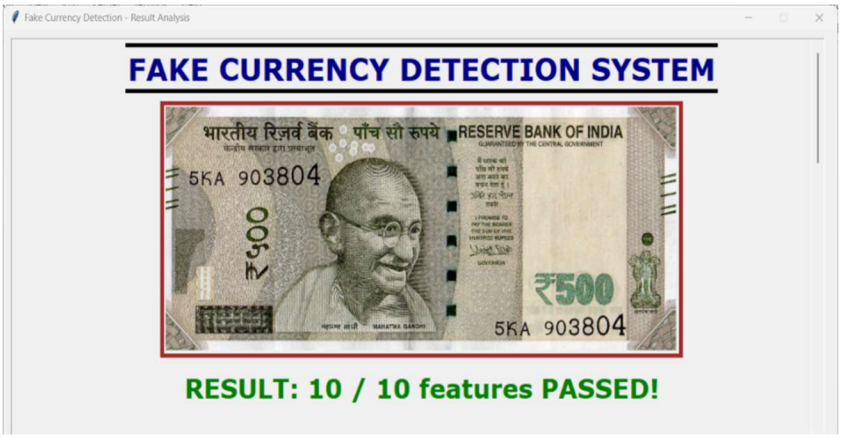

Fig.7.5 Feature extraction

6) This window presents a feature-wise analysis of the currency note using SSIM (Structural Similarity Index Measure). Each feature is compared to a reference to determine authenticity.

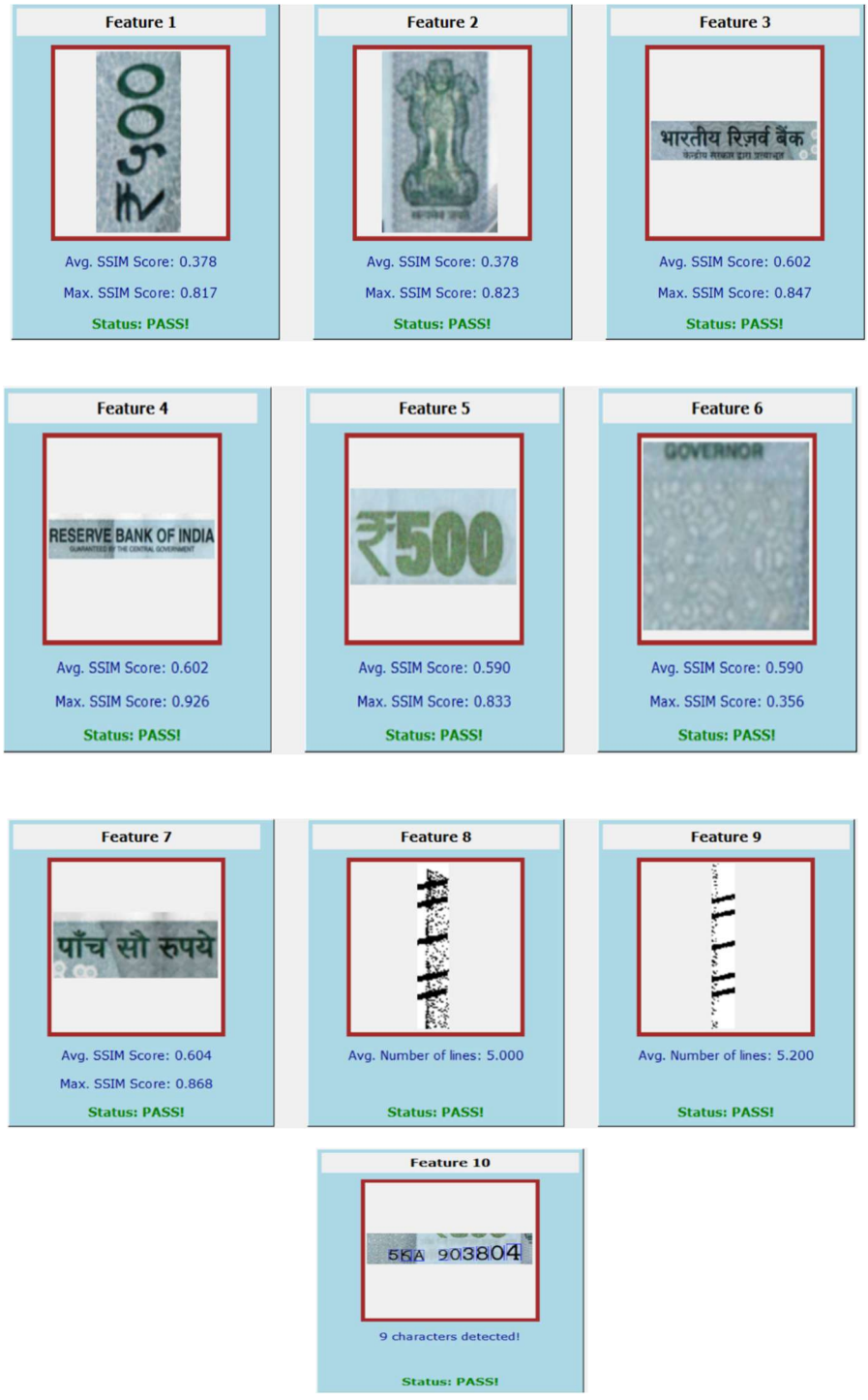

Fig.7.6 Pass features

7) "fails to match all features" typically means that the algorithm is unable to identify or correspond all the key points between two images (or between an image and a reference model).

Fig.7.7 Fail features

7.4 Currency conversion output:

1) A currency converter is a tool or system that helps users convert the value of one currency into another, using the current exchange rate.

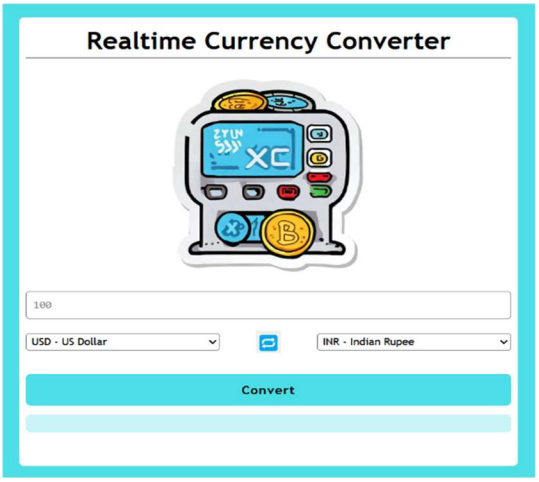

Fig:7.8. Frontened

2) A dropdown or input field for selecting currencies will open from that choose whatever you want to convert its exchange rate will show:

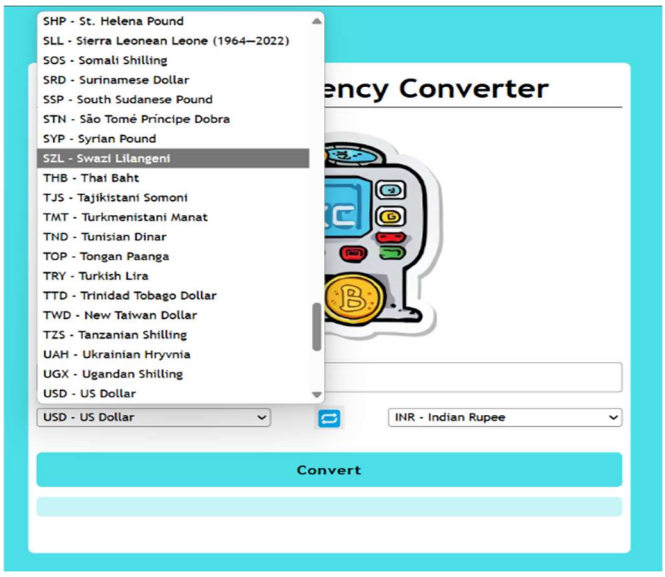

Fig.7.9 Left drop-down

3) Finally choose the current in which you want see the exchange value of that particular country.

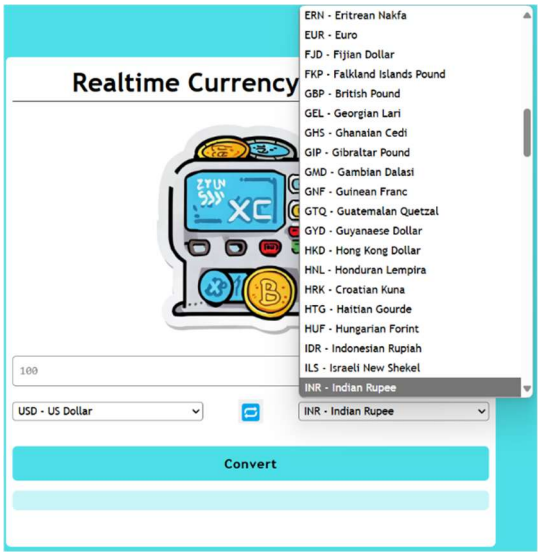

Fig.7.10 Right drop

7.5 Analysis

selecting The model takes estimate time (about 5sec-when only final results are shown leaving unnecessary details) for processing an input image. The results are also quite decent giving almost 79% accuracy in detecting genuine currency and 83% accuracy in detecting counterfeit currency.

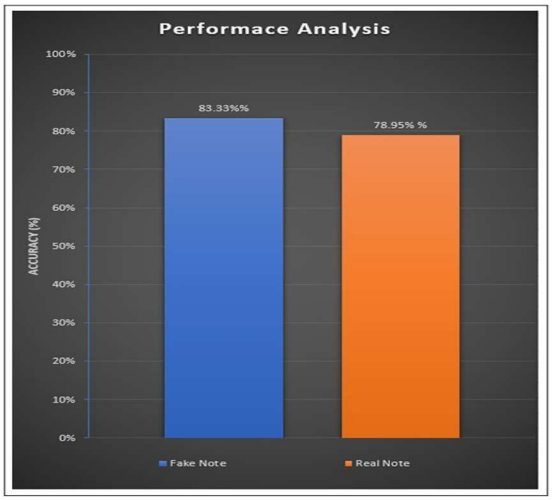

Fig.7.11 Bar analysis

CHAPTER 8

CONCLUSION

The effective application of image processing and machine learning techniques in identifying forged currency notes. By analyzing key features extracted from currency images, the system is capable of distinguishing between genuine and counterfeit notes with considerable accuracy. The developed model has shown reliable performance across various denominations, making it suitable for practical use in financial institutions, retail environments, and security checkpoints. Real-time detection enables quick assessment of notes at point-of-sale locations or during large cash transactions, while the currency conversion feature allows users to instantly convert the recognized currency value into other denominations based on current exchange rates. These enhancements not only expand the system's functionality but also improve its usability in international settings. The integration of a user-friendly interface and optional features such as real-time detection and currency conversion enhances the system's overall functionality and accessibility. Overall, the integration of image processing and machine learning for counterfeit detection presents a powerful and innovative solution for modern currency authentication. With continued refinement and the potential for mobile app integration or deployment on embedded systems like Raspberry Pi, such solutions can be made more portable and scalable, ensuring widespread adoption and a safer financial ecosystem.

8.1 Limitations of work

- They often rely on image quality, lighting conditions, and camera resolution, which can affect detection accuracy, especially in real-time scenarios.
- These systems may struggle with worn-out, folded, or partially damaged notes that deviate from the training dataset.
- The models used are typically trained on a specific set of currency denominations and may not adapt well to newly issued or redesigned notes without retraining.

8.2 Future scope of work

- Mobile and Embedded Integration: The system can be developed into a mobile app or embedded into ATMs, cash counting machines, and retail POS systems for real-time detection.
- Multi-Currency Support: Extend the model to detect counterfeit notes of various international currencies, making it suitable for global use.
- Detection of Security Features: Incorporate recognition of advanced security elements like watermarks, microtext, color-shifting ink, and holograms for stronger counterfeit resistance.

.

References

1. S. R. Darade and G. Gidveer. (2016). “Automatic recognition of fake indian currency note”, *international conference on Electrical Power and Energy Systems (ICEPES)*, *4(1), 41-46.*
2. B. P. Yadav, C. Patil, R. Karhe, and P. Patil. (2014). “An automatic recognition of fake indian paper currency note using matlab”, *Int. J. Eng. Sci. Innov. Technolvol. 3, pp. 560–566, 2014.*
3. A. Zarin and J. Uddin. M. (2014). “A hybrid fake banknote detection model using ocr, face recognition and hough features”, *Cybersecurity and Cyberforensics Conference (CCC). IEEE, pp. 91–95.*
4. M. S. Veling, M. J. P. Sawal, M. S. A. Bandekar, M. T. C. Patil, and M. A. L. Sawant (2019). “Fake indian currency recognition system by using matlab”, *International Journal of Science and Research (IJSR), 5(3), 506-510*.
5. F. A. B, P. Mangayarkarasi, Akhilendu, A. A. S, and M. K(2020), “Fake indian currency note recognition.”, *International Journal of Advanced Research in Computer and Communication Engineering, 4(4), 526-530.*
6. Joshi R.C., & Yadav S. (2020). “Yolo-V3 based currency detection and recognition system for visually impaired persons”, *International Conference on Contemporary Computing and Applications (IC3A).*
7. Rajendran & Anithaashri (2020). “CNN based framework for identifying the Indian currency denomination for physically challenged people”, *IOP Conference Series: Materials Science and Engineering*, *992(1), 012016*.
8. Jain A. (2016). “Indian Currency Recognition Using Image Processing and Artificial Neural Networks”, *International Journal of Advanced Research in Computer and Communication Engineering, 5(3), 681-684*.