TRIBHUVAN UNIVERSITY

Institute of Science and Technology

Project Report On:

"COUNTERFEIT MONEY DETECTION"

Using SVM Classification with CNN for Matching Percentage Evaluation

Submitted To:

Department of Computer Science and Information Technology

National College of Computer Studies

In partial fulfillment of the requirements of Final Year, 7th Semester for the Bachelor's Degree in Computer Science and Information Technology

Submitted By:

Ramila Kumari Shahi (23844/076)

Sanisha Maharjan (23851/076)

Shristi Bajracharya (23853/076)

Under Supervision of:

Mr. Yuba Raj Devkota

Certificate of Declaration

Award Title: BSc. Computer Science and Information Technology

Declaration Sheet

(Presented in partial fulfillment of the assessment requirements for the above award)

This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. We confirm that the intellectual content of the work is the result of our own efforts and of no other person.

It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants the University a perpetual royalty-free license to do all or any of those things referred to in section 16(I) of the Copyright Designs and Patents Act 1988. (viz. to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make an adaptation of the work).

Declared By:

Ramila Kumari Shahi	Sanisha Maharjan	Shristi Bajracharya
TU Symbol No: 23844/076	TU Symbol No.: 23851/076	TU Symbol No.: 23853/076

Date: 18th March 2024

National College of Computer Studies

Tribhuvan University

Supervisor Recommendation

I hereby recommend that this project, prepared under my supervision entitled "**Counterfeit Money Detection using SVM classification with CNN for matching percentage evaluation"**, a platform that detects counterfeit money from currency notes and provides results, in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology is processed for the evaluation.

....

Mr. Yuba Raj Devkota

Project Supervisor

National College of Computer Studies

Paknajol, Kathmandu

Certificate of Approval

This is to certify that the project report entitled "**Counterfeit Money Detection using SVM classification with CNN for Matching Percentage Evaluation**" is a Bonafede report of the work carried out by **Ms. Ramila Kumari Shahi, Ms. Sanisha Maharjan and Ms. Shristi Bajracharya** under the guidance and supervision for the degree of B.Sc. in Computer Science and Information Technology at National College of Computer Studies, Tribhuvan University.

To the best of my knowledge and belief, this work embodies the work of candidates, has duly been completed, fulfills the requirement of the ordinance relating to the Bachelor degree of the university and is up to the standard in respect of content, presentation, and language for being referred to the examiner.

.... **Mr. Yuba Raj Devkota** Project Supervisor National College of Computer Studies	.... **Mr. Rajan Paudel** Program Coordinator National College of Computer Studies
.... **External Examiner** Tribhuvan University	.... **Internal Examiner** National College of Computer Studies

Acknowledgement

It gives us immense pleasure to express our sincere gratitude with the happiest appreciation to all those respectable personalities who helped us to make our project work successive as well as productive.

Our sincere appreciation goes to our mentors, teachers, and guides, without whose unwavering support and assistance this project would not have come to fruition. We are deeply thankful for the continuous help and support provided by the college throughout the development of the project.

Special thanks are owed to **Mr. Yuba Raj Devkota** for his invaluable supervision and assistance. His guidance and support were crucial during challenging moments, and without him, this project would not have been possible. We are grateful for his consistent error corrections, valuable suggestions, and guidance that proved to be instrumental throughout the project development. Additionally, we extend our thanks to all the teachers who provided us with the opportunity to undertake this project. Their encouragement and support were integral to the project's growth and success.

Abstract

Counterfeit money detection is paramount for maintaining financial security. A comprehensive counterfeit money detection system is presented in this study, integrating Support Vector Machine (SVM) classification with Convolutional Neural Network (CNN) analysis for matching percentage evaluation. The system begins with the meticulous curation of a dataset comprising 472 samples sourced from a reputable repository, ensuring comprehensive coverage and relevance to the system's objectives. Following dataset preparation, the SVM classifier undergoes rigorous training to achieve remarkable accuracy, with an overall accuracy of 94% in distinguishing between genuine and counterfeit currency instances during testing. Additionally, the CNN analysis evaluates watermark and security thread similarity with high accuracy rates of 91% and 93%, respectively. This integrated approach, supported by a detailed workflow encompassing dataset curation, model training, and testing, underscores the system's effectiveness in detecting counterfeit currency and contributes significantly to strengthening financial security measures. The system is deployed using Flask in Python, enabling seamless integration, and providing an intuitive interface for users.

Keywords: Counterfeit Money Detection, Support Vector Machine, Convolutional Neural Network, Dataset Curation, Workflow, Accuracy, Financial Security, Flask, Python

Table of Contents

List of Abbreviations

API	Application Programming Interface
ATM	Automated Teller Machine
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
DFD	Data Flow Diagram
ER	Entity-Relationship
GIMP	GNU Image Manipulation Program
HSV	Hue, Saturation, Value
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LDA	Linear Discriminant Analysis
MATLAB	Matrix Laboratory
RGB	Red, Green, Blue
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SML	Statistical Machine Learning
SQL	Structured Query Language
SVM	Support Vector Machine
UI	User Interface

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Counterfeit Money or Fake money is any currency (paper note or coin) produced outside the legal sanction of a state or government to imitate the currency to deceive its recipient. The business of counterfeiting money is as old as the history of money itself. Today, the finest counterfeit banknotes are called super dollar/super currency because of high quality and imitation of real currency.

Out of many security features of the bank note such as watermark, see- through register, security thread, emboss, visually impaired feature, our fake money detection system will use watermark feature and security thread as its main tool to identify the fake currency. By employing image processing techniques, the system focuses on detecting the watermark and security thread in genuine currency notes to distinguish them from fake ones. This approach aims to offer a dependable and effective solution for identifying counterfeit banknotes, emphasizing the importance of watermark and security thread analysis in the detection process.

1.2 Problem Statement

In Nepal, counterfeit money fraud is on rise. The past three-month data (from Nepal News) shows that over 1 million worth counterfeit notes were confiscated, many more are still hidden and unidentified. The Bank and financial institutions, being a mediator to circulate the money around the country, is the main targeted area for the people possessing the fake currency. So, development of such a system and implementation in banks and financial institutions will decrease the flow of the fake currency in the market. This system will help to identify the fake currency beforehand and block the flow of the fake notes.

1.3 Objectives

The main objectives of this project are as follows:

- To detect the watermark engraved in notes to find the genuine ones.
- To detect the security thread engraved in notes.

1.4 Scope and Limitation

1.4.1 Project Scope

This section of our project explains the scope of our project, how we plan to deliver it and make it effective. It is a system with an added functionality of currency detection.

1. This system focuses on the improvement and implementation of fake currency detection.
2. It determines whether notes are genuine or counterfeit.

1.4.2 Limitation

- The picture must be stored on your device.
- Inadequate lighting hampers accurately identify features.
- Banknotes that show signs of tear or aging can occasionally be mistakenly identified as counterfeit.

1.5 Development Methodology

In the project, Waterfall method has been used as the development model. This project is a structured and systematic approach, making it suitable when requirements are well-defined, and changes are expected to be minimal. Each phase must be completed before moving on to the next, ensuring a clear progression in the development process. modify this paragraph.

Waterfall model follows the following processes:

1. **Requirement Analysis:**
   In this phase, the requirements for counterfeit money detection are thoroughly defined. This includes specifying the types of currency to be detected and analyzing the security features that need to be identified.

2. **System Design:**
   Building upon the gathered requirements, the system design phase entails creating a comprehensive blueprint or architecture for the counterfeit money detection

system. This encompasses defining modules, data structures, interfaces, and algorithms crucial for accurate detection.

3. **Implementation (Coding) Phase:**

   The actual coding or programming of the counterfeit money detection system occurs in this phase. Developers utilize the design specifications to write code for each module or component aimed at effectively identifying counterfeit currency.

4. **Testing Phase:**

   The testing phase is dedicated to ensuring that the individual components or modules function correctly when integrated into the entire counterfeit money detection system. Various testing methods, including unit testing and integration testing, are employed to validate the system's functionality.

5. **Validation (System Testing) Phase:**

   The entire counterfeit money detection system undergoes thorough testing to verify its compliance with the specified requirements. This phase focuses on evaluating the overall functionality and performance to ensure accurate and reliable counterfeit detection.

6. **Deployment (Implementation) Phase:**

   Upon successful completion of testing and validation, the counterfeit money detection system is deployed to the operational environment. This often involves installing the system on designated hardware and making it accessible to end-users.

7. **Maintenance Phase:**

   The maintenance phase involves providing ongoing support and updates to the counterfeit money detection system. Any identified issues or bugs are addressed promptly, and updates or enhancements may be implemented based on user feedback or evolving counterfeit threats.

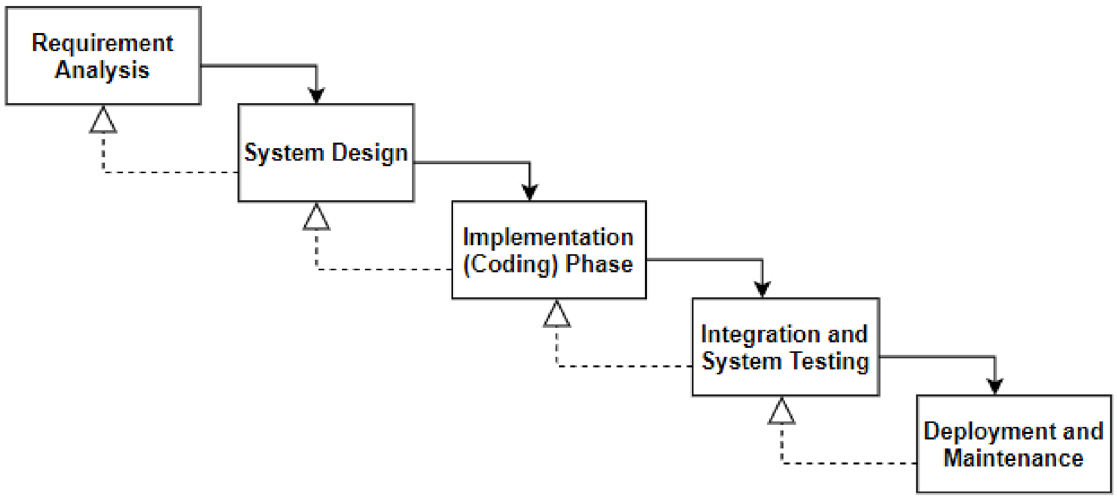

Requirement Analysis
System Design
Implementation (Coding) Phase
Integration and System Testing
Deployment and Maintenance

Figure 1. 1 Waterfall Model

Reason for Selection of Waterfall model as the development methodology

Since the project requirements are well-defined, stable and not expected to undergo significant changes. The Waterfall Model was selected for its ability to provide a controlled and predictable development process, catering to the specific characteristics and requirements of the counterfeit money detection project.

1.6 Report Organization

The report on 'Counterfeit Money Detection' is structured into 6 chapter, which include:

Chapter 1: Introduction

The chapter initiates with an introduction to the comprehensive project, systematically partitioned into six distinct subchapters. Within this section, the subchapter addresses components, including the problem statement, objectives, scope, limitations, and the chosen development methodology employed in the project.

Chapter 2: Background study and literature Review

It includes the study of fundamental theories, general concept and literature review i.e., existing system, people's preference etc. of the present Counterfeit money detection.

Chapter 3: System Analysis

This chapter includes requirement analysis and feasibility of the system. It also consists of ER-diagram and DFD-diagram.

Chapter 4: System Design

It consists of the overall design of the system and the algorithm used.

Chapter 5: Implementation and Testing

It provides the detail tool we used while developing the system. It also includes required testing with different test as per requirement.

Chapter 6: Conclusion and Future Recommendation

It summarizes the key findings, insights, and outcomes derived from the research or project and future suggestions for potential actions, improvements, or areas of further exploration based on the conclusions drawn.

Chapter 2: Background Study and Literature Review

2.1 Background Study

The study of fake money detection involves a multidimensional approach encompassing physical conditions, image processing techniques, and feature extraction methods. Various research papers delve into the development of systems for detecting counterfeit currency, particularly focusing on Indian currency notes. These systems utilize computer vision-based approaches, image processing, and feature extraction to distinguish between genuine and fake currency. Techniques like Radiometric and Geometric corrections are employed to rectify spectral errors and distortions in currency images.[1] The use of MATLAB for feature extraction, image segmentation, and decision-making processes is highlighted in these studies. Additionally, advancements in technology, such as the application of Generative Adversarial Networks (GANs) for counterfeit money detection, showcase the continuous evolution of methods to combat counterfeiting in financial systems.

2.2 Literature Review

ANALYSIS OF COUNTERFIET CURRENCYB DETECTION TECHINQUES FOR CLASSIFICATION MODEL (Akansha Upadhyaya, Dr. Vinod Shokeen): [2] Counterfeit currency is one of the threats which creates vice to nation's economy and hence impacts the growth worldwide. Producing forged currency or fabricating fake features in the currency considered to be a crime. Currency crime comes under the criminal law and known to be as Economical crime. Over the past few years many researchers have proposed various techniques to identify and detect forged currency. A serious problem has been come up with variety of solutions in terms of hardware related techniques, Image processing and machine learning methods. Advancements in printing and scanning technology, trading of material are some of the problems in germinating counterfeit currency. The study presents various fake currency detection techniques proposed by various researchers. The review highlighted the methodology implemented on characteristics feature with success rate of each method to detect counterfeited currency. Moreover, the study includes the analysis of widely acceptable statistical classification techniques for currency authentication. The comparative analysis of Logistic Regression and Linear Discriminant Analysis (LDA) was performed to realize the better model for currency authentication. It has been found that classification Model using Logistic regression shows better accuracy of 99identifying most feasible technique to be implemented based on the accuracy rate REVIEW ON DETECTION OF

FAKECURRENCY USING IMAGE PROCESSING TECHNIQUES (Dr. S.V. Virakthamath, Kshama Tallur [3]: Many countries are affected by the matter of fake notes. Indian is one among them. With improved technology, anyone can print fake notes. These notes are produced without legal sanction of the state and continued production of such kinds of notes can degrade countries economy. When such counterfeited notes are produced and circulated, it becomes impossible for ordinary citizens to distinguish whether the money is real or fake because they differentiate according to physical appearance. The biggest challenge for many countries like India is the detection of fake currency. Even if banks and other big organizations have automatic machines designed to identify counterfeit currency notes, ordinary people can hardly differentiate between them. Nowadays recognition of fake currency has become a challenging issue for many researchers. The identification involves many steps like edge detection, feature extraction, image segmentation, image acquisition, gray scale conversion, and comparison of images. This paper provides some related works of paper currency recognition and has explained the spread of various currency recognition systems. Choosing the right feature would improve overall system performance. The goal of this work is to review previous papers and literature, identify the benefits and disadvantages of every method.

A HYBRID FAKE BANK NOTE DETECTION MODEL USING OCR, FACE RECOGNITION AND HOUGH FEATURES (Abida zarin, Jia Uddin) [4]: Currency duplication remains an emergent concern among nations due to the advancement of printing and scanning technology. Many note detection systems are present in banks but they are very costly and often inaccurate. The fields of image processing, neural network and machine vision have potential to significantly overcome this issue. This paper proposes a model comprised of Optical Character recognition (OCR), Face Recognition and Hough transformation algorithm. The microprinting, watermark, and ultraviolet lines features of Bangladeshi notes are extracted for testing of genuine notes. The experimental results of the proposed model give accuracy as high as 93.33mobile application. Moreover, the obtained results are compared with the output from individual algorithm of OCR, Face Recognition and Hough transformation, to show that the proposed algorithm gives the highest accuracy.

AUTOMATIC RECOGNITION OF FAKE INDIAN CURRENCY NOTE (Sonali R. Darade, Prof. G.R. Gadveer) [5]: In this paper, the automatic system is designed for identification of Indian currency notes and check whether it is fake or original. The automatic system is very useful in banking systems and other field also. In India increase in the counterfeit currency notes of 100, 500 and 1000 rupees. As increase in the technology like scanning, color printing and duplicating, because of that there is increase in counterfeit problem. In this paper, recognition of fake Indian currency notes is done by using image processing technique. In this paper, recognition of fake Indian currency notes is done by using image processing technique. In this technique first the image acquisition is done and applies preprocessing to the image. In pre-processing crop, smooth and adjust then convert the image into grey color after conversion apply the image segmentation then extract features and reduce, finally comparing image.

EVALUATON OF MACHINE LEARNING ALGORITHMS FOR DETECTION OF FAKE BANK CURRENCY (Anuj Yadav, Tarun Jain) [6]: The one important asset of our country is Bank currency and to create discrepancies of money miscreants introduce the fake notes which resembles to original note in the financial market. During demonetization time it is seen that so much of fake currency is floating in the market. In general, for a human being, it is very difficult to identify forged note from the genuine not instead of various parameters designed for identification as many features of forged note are like original one. To discriminate between fake bank currency and original note is a challenging task. So, there must be an automated system that will be available in banks or in ATM machines. To design such an automated system there is need to design an efficient algorithm which can predict whether the banknote is genuine or forged bank currency as fake notes are with high precision. In this paper six supervised machine learning algorithms are applied on dataset available on UCI machine learning repository for detection of Bank currency authentication. To implement this, we have applied Support Vector machine, Random Forest, Logistic Regression, Naive Bayes, Decision Tree, K- Nearest Neighbor by considering three train test ratio 80:20, 70:30 and 60:40 and measured their performance on the basis various quantitative analysis parameter like Precision, Accuracy, Recall, MCC, F1-Score and others. And some of SML algorithm are giving 100 train test ratios.

FAKE CURRENCY DETECTION USING IMAGE PROCESSING AND MACHINE LEARNING (S. Gothe, K. Naik, V. Joshi) [7]: With the increase in the technology, the convenience and ease of people to carry out various task is increasing on a large scale. But with the advancement in technology, the amount of crime carried out due to wrong use of these technologies is also increasing on a large scale. Similar thing 7applies to the currency notes being handled by us on day to day basis. Fake currency notes are made so accurate that finding out which one is real note and which one is fake becoming increasingly difficult. Fake currency note is the imitation of the authentic currency note for wrong purposes and without the permission of the state and central government. Hence with the advancement in the development of fake currency notes, the detection mechanism needs to be developed as well to reduce its flow in the market. Fake currency notes have led to reduction in the value of money and also loss on economic and social front. In the paper we are using Image Processing and Machine Learning to check the authenticity of the currency note. An android application will help people to easily detect the fake note as many people today carry smart phones and hence android application is not a difficult thing for people to handle. This led us to satisfy our purpose of our application being helpful to common citizen of India. Key words Fake Currency Detection, Counterfeit Detection

SUPPORT VECTOR MACHINE IN THE ANTICIPATION OF CURRENCY MARKETS. Int. J. Eng. Technol. (UAE) 7(2), 66–68 (V. Lalithendra Nadh, G. Syam Prasad) [8]: Various researchers have done an expansive research within the domain of stock market anticipation. The majority of the anticipated models is confronting some pivotal troubles because of the likelihood of the market. Numerous normal models are accurate when the data is linear. In any case, the expectation in view of nonlinear data could be a testing movement. From past twenty years with the progression of innovation and the artificial intelligence, including machine learning approaches like a Support Vector Machine it becomes conceivable to estimate in light of nonlinear data. Modern researchers are combining GA (Genetic Algorithm) with SVM to achieve highly precise outcomes. This analysis compares the SVM and ESVM with other conventional models and other machine learning methods in the domain of currency market prediction. Finally, the consequence of SVM when compared with different models it is demonstrated that SVM is the premier for foreseeing.

REAL TIME FAKE CURRENCY NOTE DETECTION USING DEEP LEARNING. Int. J. Eng.Adv. Technol. (IJEAT) 9(1S5) (M. Laavanya, V. Vijayaraghavan). ISSN: 2249-8958 [9]: Great technological advancement in printing and scanning industry made counterfeiting problem to grow more vigorously. As a result, counterfeit currency affects the economy and reduces the value of original money. Thus it is most needed to detect the fake currency. Most of the former methods are based on hardware and image processing techniques. Finding counterfeit currencies with the methods is less efficient and time consuming. To overcome the above problem, we have proposed the detection of counterfeit currency using a deep convolution neural network. Our work identifies the fake currency by examining the currency images. The transfer learned convolutional neural network is trained with two thousand, five hundred, two hundred and fifty Indian currency note datasets to learn the feature map of the currencies. Once the feature map is learnt the network is ready for identifying the fake currency in real time. The proposed approach efficiently identifies the forgery currencies of 2000, 500, 200, and 50 with less time consumption.

FAKE CURRENCY DETECTION USING IMAGE PROCESSING,ICSET-2017.IOPConf.Ser.Mater.Sci.Eng. 263, 052047 (T.Agasti,G. Burand, P.Wade, P.Chitra) [10]: The advancement of color printing technology has increased the rate of fake currency note printing and duplicating the notes on a very large scale. Few years back, the printing could be done in a print house, but now anyone can print a currency note with maximum accuracy using a simple laser printer. As a result the issue of fake notes instead of the genuine ones has been increased very largely. India has been unfortunately cursed with the problems like corruption and black money. And counterfeit of currency notes is also a big problem to it. This leads to design of a system that detects the fake currency note in a less time and in a more efficient manner. The proposed system gives an approach to verify the Indian currency notes. Verification of currency note is done by the concepts of image processing. This article describes extraction of various features of Indian currency notes. MATLAB software is used to extract the features of the note. The proposed system has got advantages like simplicity and high performance speed. The result will predict whether the currency note is fake or not.

Chapter 3: System Analysis

3.1 System Analysis

The system analysis of the counterfeit money detection system involves defining objectives, gathering requirements, selecting appropriate technologies, implementing feature extraction techniques, training CNN models, and designing a user-friendly interface to accurately detect counterfeit currency notes and enhance financial security measures.[11]

3.1.1 Requirement Analysis

The requirement analysis for the counterfeit money detection system encompasses both functional and non-functional aspects essential for accurate and efficient detection of counterfeit currency.

i. Functional Requirement

Use Case Diagram:

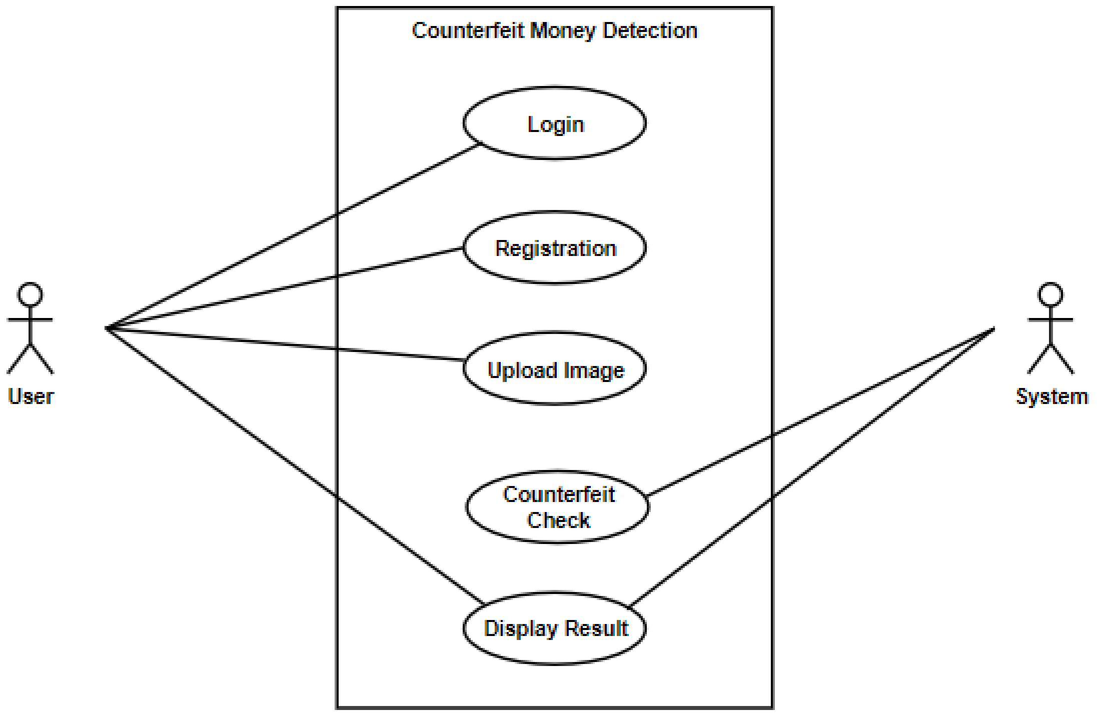


Figure 3. 1 Use Case

Use Cases:

a. **Login:**

- Actor: User

- Description: The user logs into the system using valid credentials (username and password).

b. **Registration:**
   - Actor: User
   - Description: New users can register for the system by providing necessary information and creating a username and password.

c. **Upload:**
   - Actor: User
   - Description: Users can upload image of currency note for analysis. The uploaded image is then processed by the system.

d. **Counterfeit Check:**
   - Actor: System
   - Description: The system analyzes the uploaded image to determine if the currency notes are genuine or counterfeit. This involves using image processing and counterfeit detection algorithms.

e. **Display Result:**
   - Actor: System
   - Description: After the counterfeit check is performed, the system displays the results to the user. It indicates whether the uploaded currency note is genuine or potentially counterfeit. The system also displays the matching percentage of watermark and security thread of the uploaded currency note.

ii. Non-Functional Requirement

The non-functional requirements of a counterfeit money detection system encompass aspects beyond the system's specific functionalities. Some non-functional requirements that could be considered for such a system area as follows:

- **Performance:**
  The system should be able to process currency images quickly and accurately to detect counterfeit notes efficiently.

It should have a high level of accuracy in distinguishing between genuine and fake currency.

- **Reliability:**

  The system should be reliable, ensuring consistent and accurate detection results.

  It should be robust enough to handle variations in currency notes and environmental conditions.

- **Usability:**

  The system should have a user-friendly interface that is easy to navigate for users checking the authenticity of currency notes.

  It should provide clear and understandable results to users.

- **Security:**

  The system should maintain the confidentiality and integrity of the data processed, especially when dealing with sensitive financial information.

  It should have measures in place to prevent unauthorized access to the system.

- **Scalability:**

  The system should be scalable to accommodate an increasing number of users and currency checks without compromising performance.

  It should be able to handle a growing dataset of currency images for training and testing.

- **Maintainability:**

  The system should be easy to maintain and update with new counterfeit detection techniques or features.

  It should have clear documentation and modular design for ease of maintenance.

These non-functional requirements are essential for ensuring the overall effectiveness, usability, and reliability of a counterfeit money detection system.

3.1.2 Feasibility Analysis

i Technical Feasibility

The technical feasibility of the project is high, as the required hardware and software resources are widely available and accessible. Requirements of our system can be categorized as:

Hardware Requirements:

- A computer with a minimum of 4 GB of RAM and a multi-core processor
- Sufficient RAM for storing and processing image data during the counterfeit detection process.
- Storage devices for maintaining a database of known currency features and patterns for comparison during counterfeit detection.

Software Requirements:

- Operating System: Windows 10, Linux, or MacOS
- Web Browser: Chrome, Firefox, or Safari
- Python 3.6 or above: This is required to run the Flask web framework.
- Flask web framework: This is required to build the web application and API for the system.
- ReactJS: This is required to build the frontend of the web application and to create a responsive user interface.
- Git and GitHub: These are required for version control and collaboration among the development team.

In addition to the above requirements, some optional software may be used for development such as virtual environments like Anaconda to manage dependencies.

ii Operational Feasibility

The operational feasibility of the project is moderate, as the project requires the development of a reliable and accurate counterfeit money classifier. The operational feasibility of implementing a counterfeit money detection system hinge on its seamless integration with existing processes, compatibility with current infrastructure, and user acceptance. Regulatory compliance, scalability, reliability, and minimal operational impact are also key considerations. The system's maintenance requirements, risk management, and customer experience implications further contribute to determining its operational viability. Ultimately, a thorough evaluation of these factors is essential to ascertain the practicality and effectiveness of a counterfeit money detection system.

iii Economic Feasibility

The economic feasibility of the project is high, as the project does not require any significant upfront investment or ongoing operational costs. The required hardware and software resources are typically available on most computers and can be acquired at a relatively low cost. In addition, the development can use open-source software and libraries to reduce the cost of development.

iv Schedule Feasibility

The time given for the completion of this project was a whole semester. So, the project had enough time for completion. Since the project has some machine learning mechanisms the project took some more time than done usually. Hence, the project has been developed according to the following time schedule to make our application schedule feasible.

Table 3. 1 Schedule Plan

S. No.	Task Name	Duration	Start Date	End Date
1	Planning	6 Days	22 Oct, 2023	28 Oct, 2023
2	Analysis	8 Days	29 Oct, 2023	11 Nov, 2023
3	Design	12 Days	12 Dec, 2023	26 Dec, 2023
4	Coding	30 Days	26 Dec,2023	14 Feb, 2024
5	Testing	10 Days	15 Feb, 2024	28 Feb, 2024
6	Implementation	6 Days	28 Feb, 2024	11 Feb, 2024
7	Documentation	90 Days	12 Nov, 2023	11 Feb, 2024

3.1.3 Analysis

Data Modeling using ER-diagram.

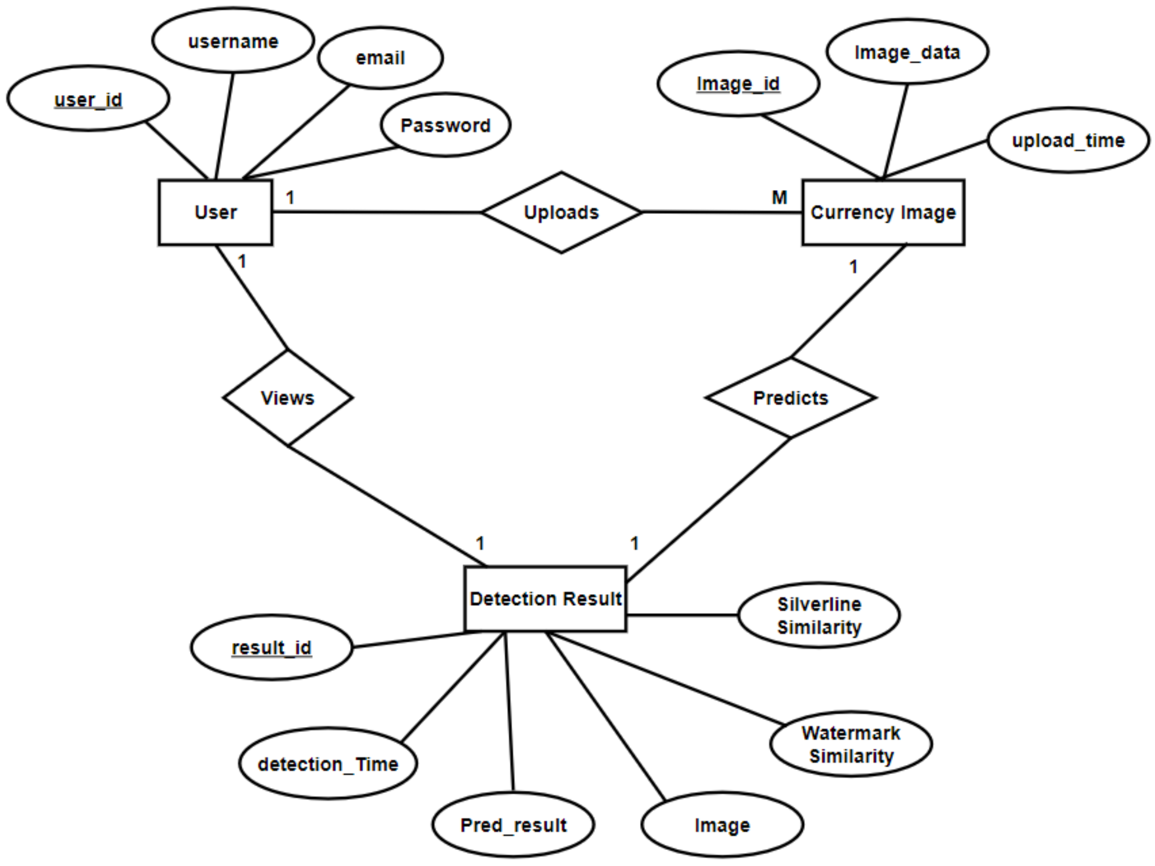


Figure 3. 2 ER-Diagram

In the ER (Entity-Relationship) diagram for a counterfeit money detection system, three key entities are represented: User, Currency Image, and Detection Result.

User Entity:

The "User" entity is a fundamental component of the counterfeit money detection system, representing individuals who engage with the system. It is characterized by four main attributes:

- **user_id**: This attribute serves as a unique identifier for each user in the system, ensuring individuality and facilitating relational links.

- **username**: The username attribute represents the chosen identifier or display name selected by the user during the registration process.
- **email**: This attribute stores the user's email address, providing a means of communication and contact.
- **Password**: The password attribute stores the secure access key chosen by the user during registration, ensuring the confidentiality of their account.

Currency Image:

The "Currency Image" entity is a pivotal element in the counterfeit money detection system, capturing the images submitted by users for analysis. This entity is characterized by two crucial attributes:

- **image_id:** This is the attribute where the uploaded image identifier is stored.
- **Upload_Date_Time:** This attribute represents the timestamp when the currency image was uploaded to the system. It serves as an essential chronological reference, aiding in tracking and organizing the images based on their submission time.
- **image_data:** The image_data attribute stores the binary data or reference to the actual image file. This includes the content of the currency image uploaded by the user. It may involve the image file itself or a link/reference pointing to the stored image data.

Detection Result Entity:

The "Detection Result" entity is a critical component of the counterfeit money detection system, recording the outcomes of the analysis performed by the system. This entity is characterized by several key attributes:

- **result_id:** This attribute serves as a unique identifier for each detection result. It ensures individuality and allows for easy referencing and tracking of results.
- **Prediction Date Time:** The prediction_date attribute indicates the timestamp when the system performed the counterfeit detection analysis. It provides chronological information, allowing users to understand when the analysis was conducted.
- **Watermark Similarity:** This attribute represents the degree of similarity between the uploaded currency image and the expected watermark of genuine currency. It

quantifies the likeness and contributes to determining the authenticity of the currency note.

- **Security Thread Similarity:** This attribute measures the resemblance of the uploaded currency image to the expected characteristics of a genuine currency note, specifically focusing on features like security features or metallic elements.
- **Pred_result:** This is the attribute where the result of the uploaded image is stored, indicating whether it is classified as real or fake.
- **Image:** This attribute stores the uploaded image in the form of an image blob.

Process Modeling using DFD

DFD-0 Level Diagram

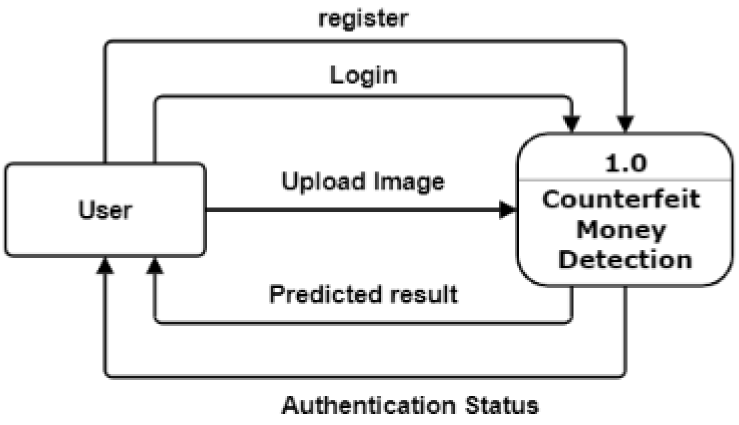


Figure 3. 3 DFD-0 Diagram

In the DFD-0 level, the primary entities include the "user" and the "counterfeit money detection" process. Users interact with the system through dataflows like "upload image," "login," and "register," all directed towards the "counterfeit money detection" process. The "upload image" dataflow enables users to submit currency images for analysis, while "login" and "register" processes handle user authentication and account creation, respectively. The "counterfeit money detection" process processes the uploaded images and generates predicted results based on the analysis. The outcomes, specifically "predicted result" and "authentication status," are then communicated back to the user. This DFD-0 level diagram illustrates the essential interactions and dataflows between the user entity and the counterfeit money detection process, capturing the key components for image submission, user authentication, and result retrieval.

DFD-1 level Diagram

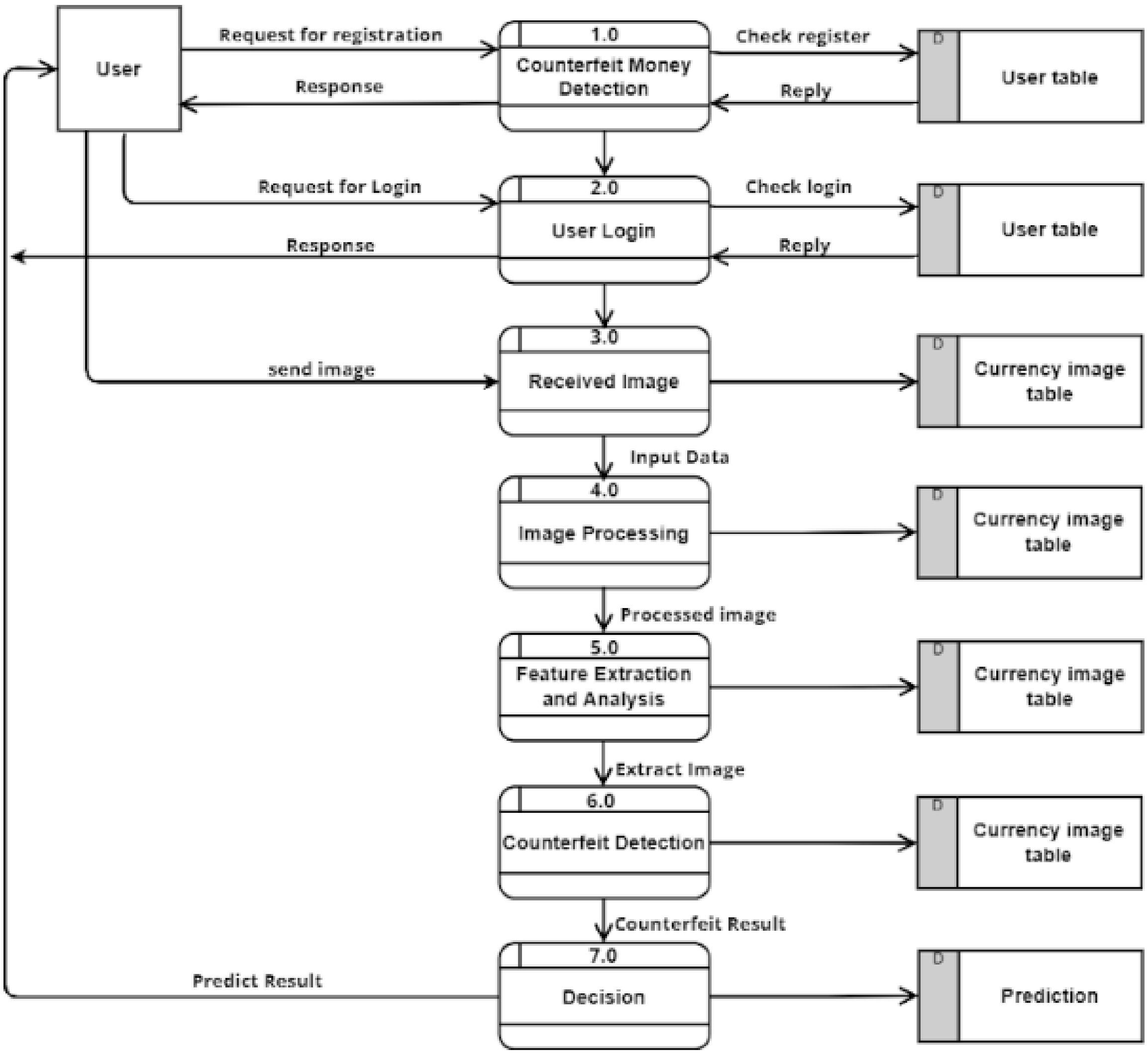


Figure 3. 4 DFD-1 Diagram

In the DFD-1 diagram, a singular user entity interacts with key processes within the counterfeit money detection system. User authentication is handled through the "user login" process, while the "receive image" process manages image submissions. Subsequent processes include "image processing" for quality enhancement, "feature extraction and analysis" for authenticity evaluation, and the central "counterfeit detection and decision" process. Three critical data stores— "user table," "currency table," and "prediction"— store essential information, streamlining interactions and supporting informed decision-making in the system. This concise breakdown in the DFD-1 diagram provides a clear snapshot of subprocesses and data management, facilitating efficient system understanding and design considerations.

Chapter 4: System Design

4.1 Design

4.1.1 Database Design

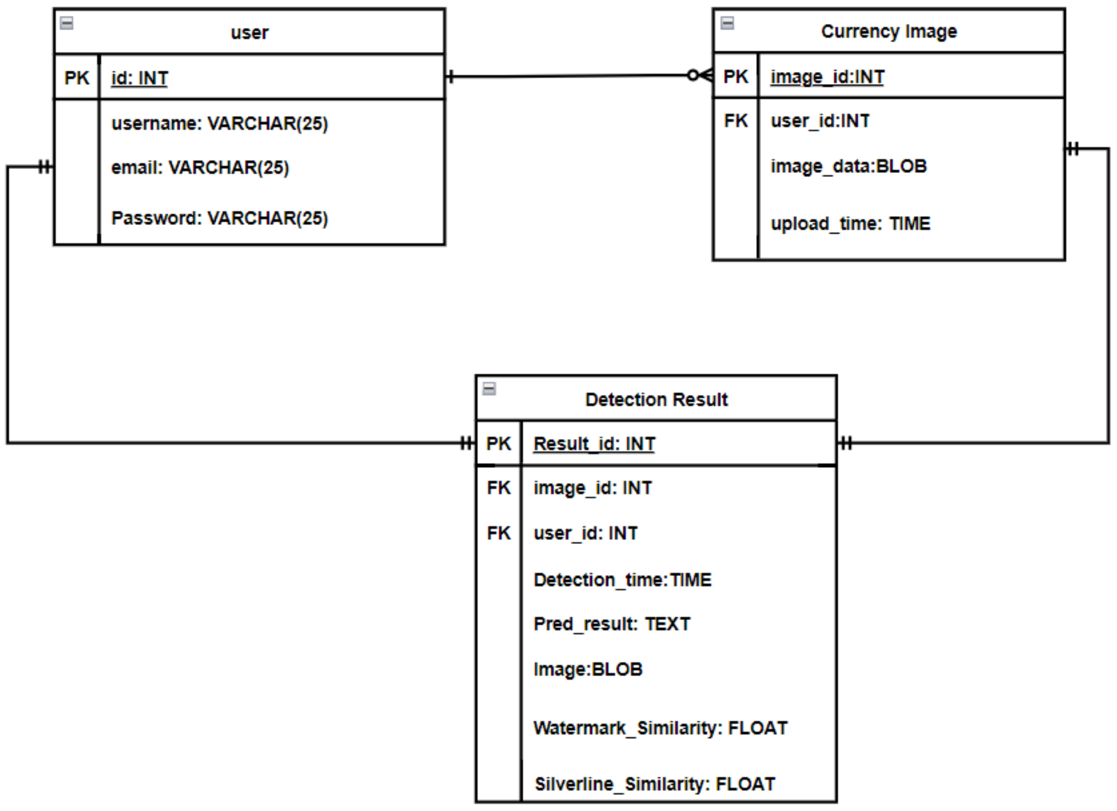


Figure 4. 1 Database Design

The diagram above shows an entity-Relationship (ER) diagram with three tables: ‘user’, ‘Currency image’ and ‘Detection Result’. It illustrates the transformation of an ER model into a relational schema:

- Each entity (user, Currency image and Detection Result) is a table.
- Attributes of entities become table column.
- ‘Id’ in all tables is a primary key(pk).
- Currency image has foreign key as ‘user_id’ and detection result has foreign key as ‘image_id’

Normalizations

1. First Normal Form (1NF):
    - Already in 1Nf, as each column contains atomic(indivisible) values.
    - No repeating groups or arrays; each column contains atomic values.
2. Second Normal Form (2NF):
    - Already in 2NF since there is no composite primary key.
    - The 'user_id' is a foreign key, and there is no partial dependency on the primary key.
3. Third Normal Form (3NF):
    - Already in 3NF, science there are no transitive dependencies.
    - No transitive dependencies, each non-prime attribute depends on the primary key.

4.1.2 Form Design

Based on our project, forms were designed for customer.

1. User Login & User Sign-Up Form: Allow user to register or log into the platform.
2. Contact-Form: Allows user to reach out the platform.

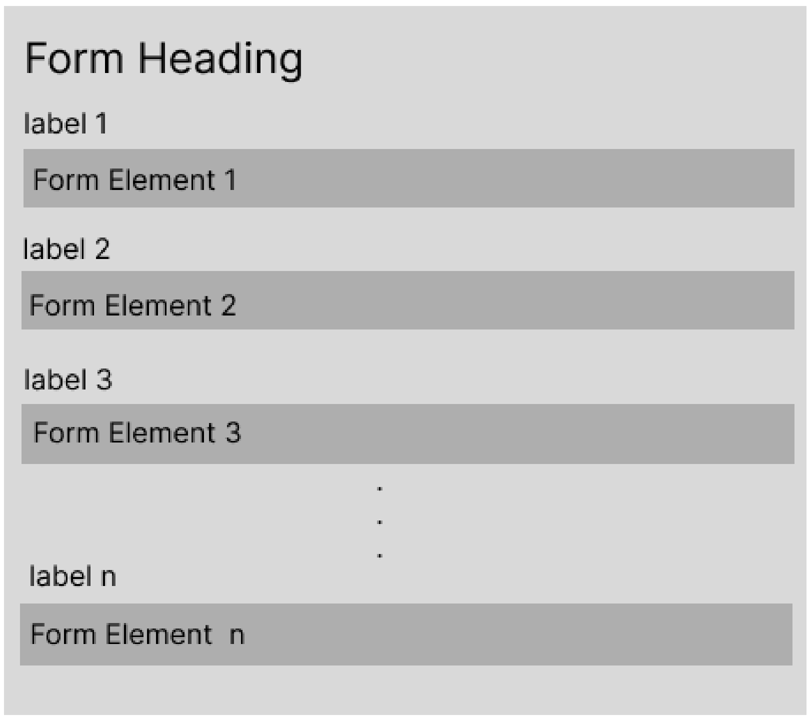


Figure 4. 2 Form Design

4.1.3 Interface Design

Interface design, often referred to as user interface (UI) design, focuses on creating visual elements and interactions that enable users to interact with a system or application. A well-designed interface should be intuitive, visually appealing, and user-friendly.

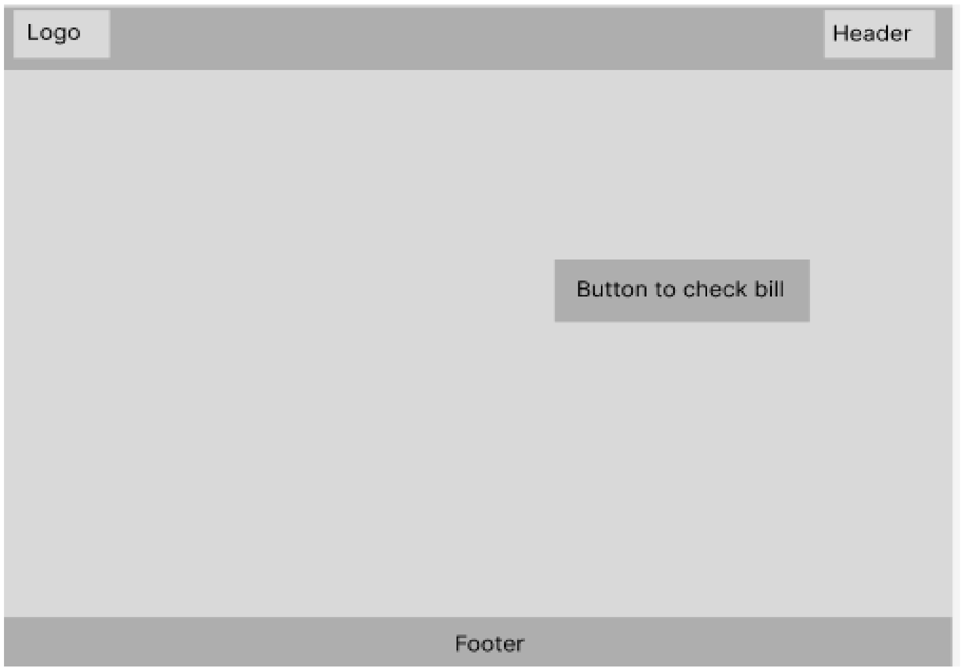


Figure 4. 3 Home Design

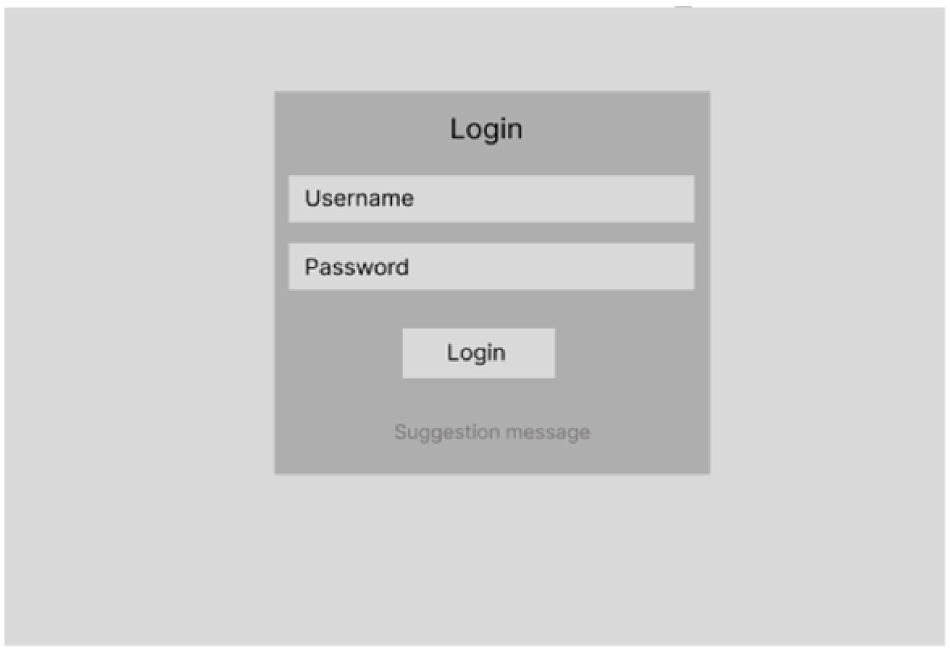


Figure 4. 4 Login Design

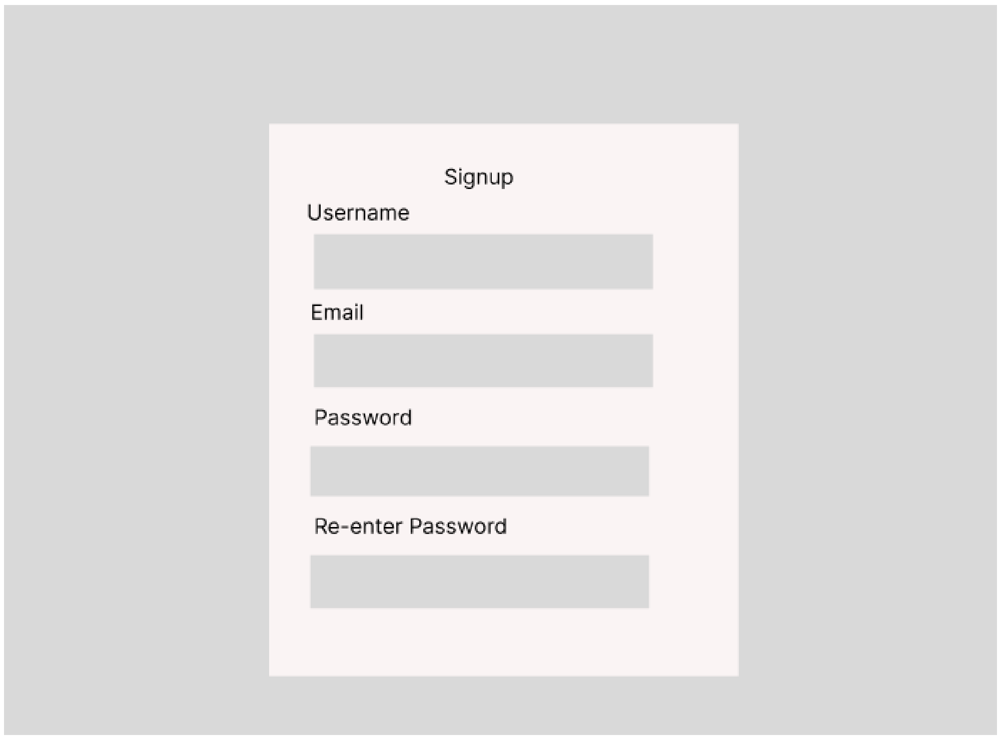


Figure 4. 5 Sign-Up Design

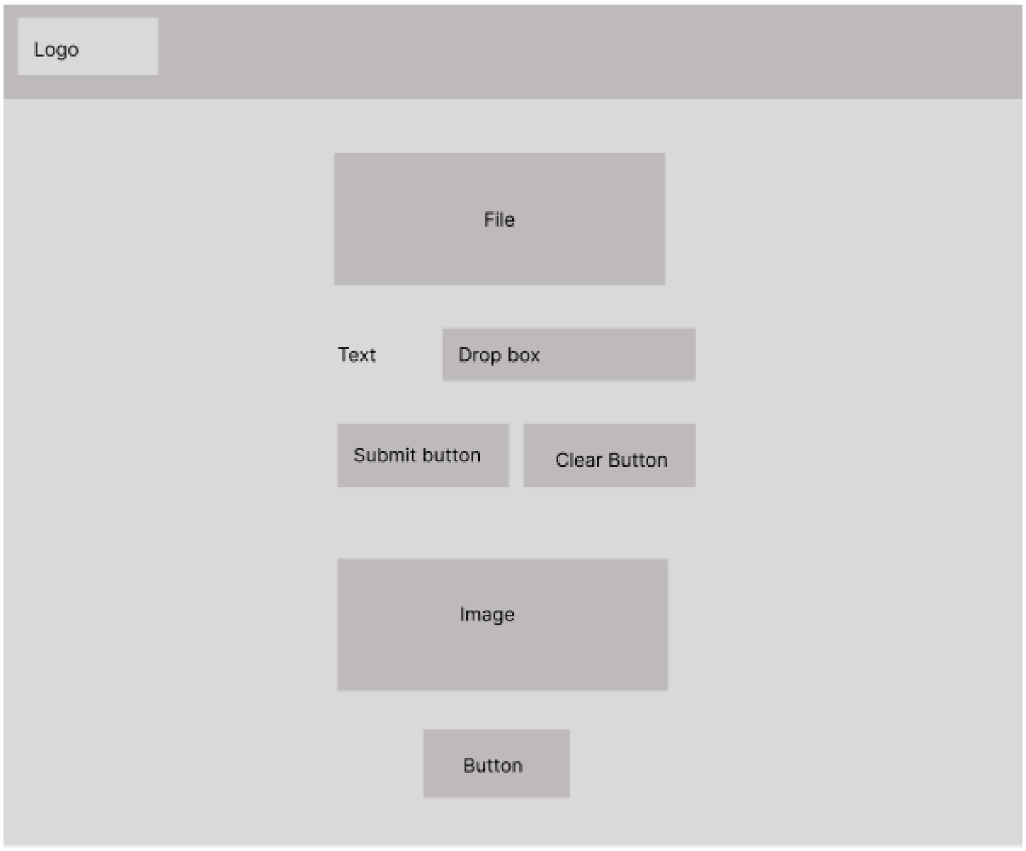


Figure 4. 6 Select Image

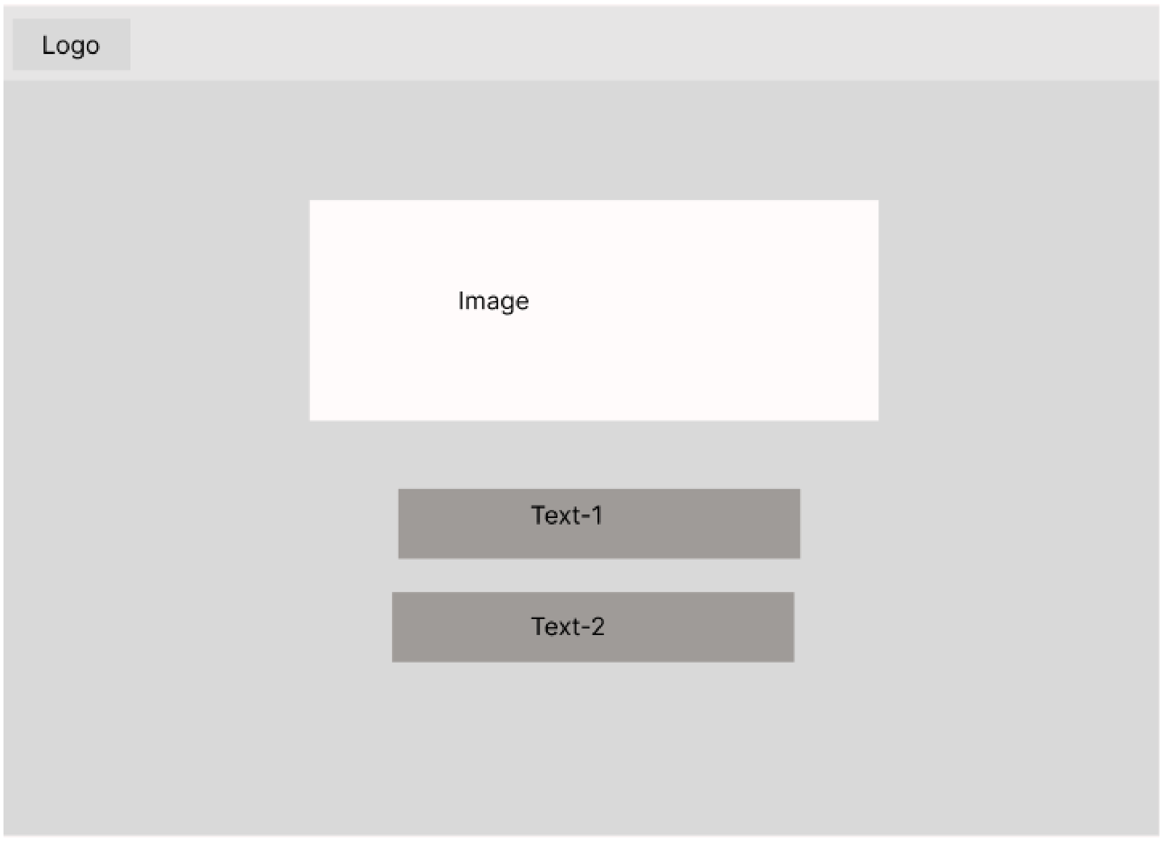


Figure 4. 7 Result

4.2 Algorithm Details

SVM for Classification

SVM stands for Support Vector Machine, which is a popular type of machine learning algorithm used for classification and regression analysis.

The steps for implementing SVM are as follows:

1. **Load Images and Extract Features:**
   The code loads images from a specified folder using the load_images_from_folder function. Each image undergoes feature extraction in the extract_features function, involving resizing, grayscale conversion, and contrast enhancement. The processed image is further manipulated by denoising, darkening, and region extraction. Watermark and Security Thread templates are loaded to detect their presence in the designated regions. This series of steps forms the initial preprocessing stage for counterfeit money detection.

2. **Preprocess and Save Images:**
   The preprocess_and_save function utilizes the extract_features function to preprocess and save images. Images are categorized as real or fake based on the presence of a watermark and security thread.

3. **Load and Preprocess Images for SVM:**
   Load real and fake images from the specified folders. Preprocess and save the images, maintaining counts for real and fake images.

4. **Custom Train-Test Split:**
   The custom_train_test_split function shuffles and divides the data into separate training and testing sets. It first shuffles the data randomly and then allocates 20% of it for testing and 80% for training, according to the specified test_size.
5. **Custom Standard Scaler:**
   The CustomStandardScaler class is a part of the feature scaling process. Feature scaling is a crucial preprocessing step in machine learning, and it involves transforming the input features so that they have similar scales. This ensures that no feature dominates the learning process due to having a larger magnitude.

The formula for standardization is as follows:

$$Standardized\ Value = \frac{Orginal\ Value - Mean}{Standard\ Deviation}$$

6. **Custom SVM Model:**

The CustomSVM class is defined for a custom SVM model. The fit method trains the model using custom gradient descent. The predict method makes predictions based on the learned weights and bias. The plot_metrics function is a part of the CustomSVM class, and it is responsible for visualizing the training and testing accuracy along with the training and testing loss over the epochs.

   a. **Prediction (Linear SVM):**

   The prediction is made in the predict method of the CustomSVM class. The dot product of weights (w) and the input features (X) is computed, and the sign of the result determines the predicted class (1 or 0).

$$Prediction = sign(\omega . X + b)$$

   b. **Update Rule (Gradient Descent):**

   The weights (ω) and bias (b) are updated during training using gradient descent. The update rule for a binary classification SVM is as follows:

$$\omega_i = \omega_i + n.(y - Prediction)$$

   c. **Loss (Squared Error Loss):**

   The squared error loss is used to measure the error during training.

$$Loss = \frac{1}{2}(y - Prediction)^2$$

7. **Load and Standardize Features:**

Load all preprocessed images and extract features. Create labels based on whether the image is real or fake.

8. **Evaluate Training and Testing Accuracy:**

 Make predictions on both training and testing sets. Calculate and print the accuracy on both sets.

9. **Calculate the Confusion Matrix for testing:**
   Confusion matrix is calculated using the confusion_matrix function. It takes two arguments: test_labels, which represent the true labels of the test data, and test_predictions, which represent the predicted labels of the test data.

10. **Calculate the Classification Report:**
    Classification reports generate both the training and testing sets using the \`classification_report\` function. For the training set, it calculates a classification report based on the true labels train_labels and the predicted labels train_predictions. Similarly, for the testing set, it computes a classification report using the true labels test_labels and the predicted labels test_predictions, with the same specified target_names['fake','real'].

Convolution Neural Network (CNN) for matching percentage

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms specifically designed for processing and analyzing visual data, such as images and videos. [12] Here's an overview of the key components and operations involved in a matching percentage with CNN:

1. **Define the CNN model:**
   The custom CNN model is defined using the Keras Sequential API. The model architecture includes convolutional layers (Conv2D), max-pooling layers (MaxPooling2D), a global average pooling layer (GlobalAveragePooling2D), and a dense layer with softmax activation. The model is defined in the custom_model function.
   a. **Convolutional Layer (Conv2D):**
      The convolutional layer applies filters to the input image to extract features. Each filter is convolved with the input, and the result is passed through an activation function. Formula:

$$Output = activation(\sum(input \times filter) + bias)$$

b. **Max-pooling layer (MaxPooling2D):**

Max-pooling reduces the spatial dimensions of the feature maps, retaining the most important information and decreasing computational complexity.

c. **Global Average Pooling (GlobalAverageePooling2D):**

Global Average Pooling is applied globally across the entire feature map. Instead of dividing the input into local regions and computing the maximum or average within each region, Global Average Pooling computes the average of each feature map across its entire spatial dimensions. The result is a single value per feature map.

$$GAP(X_i) = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{\omega=1}^{W} X_i(h, \omega)$$

d. **Fully Connected (Dense) Layers:**

Fully connected layers process the flattened features and make predictions. These layers learn to combine features and capture global patterns.

$$Z = X.W + b$$

Where, X= input matrix, W= Weight matrix, b= bias vector

e. **Output Layer:**

The output layer produces the final prediction. For binary classification (counterfeit money detection) multi-class classification, a softmax activation is used as there is two class Real and Fake. The softmax function ensures that the sum of all our output probability values will always be equal to one so that one can easily see which classified choice has the highest probability of being the right one.

Where, z_i represents raw score for class i

N is the total number of classes

2. **Preprocess Input Images:**

Images from the input folder are preprocessed, including resizing, grayscale conversion, histogram equalization (CLAHE), denoising, and darkening.

The preprocess_input function is responsible for these preprocessing steps.

$$gray_{value} = 0.299 * R + 0.587 * G + -0.114 * B$$

This formula calculates the weighted average of the RGB (Red, Green, Blue) values to obtain a single grayscale value.

3. **Detect Watermark:**

   The detect_watermark function is used to detect a watermark in the preprocessed images. It uses template matching (cv2.matchTemplate) with a set of watermark template images.

4. **Detect Security Thread:**

   The detect_security thread function compares each image with a set of security thread template images. Template matching (cv2.matchTemplate) is used to find potential matches. If a match is found based on a threshold, a rectangle is drawn around the detected security thread in the image.

5. **Custom Train-Test Split:**

   The custom_train_test_split function shuffles and divides the dataset of images into training and validation sets (with 80% for training and 20% for validation) and assigns the resulting subsets to variables train_images and val_images.

6. **Train the Custom CNN Model:**

   The train_custom_model function is responsible for training the custom CNN model. Image data generators are created for the training and validation sets.

   The model is compiled with stochastic gradient descent (SGD) optimizer, binary cross entropy loss, and accuracy as a metric. Model training is performed using model.fit. A checkpoint is set up to save the best model based on validation loss.

   After training, the model, classification report, confusion matrix, and accuracy are printed and displayed.

   a. **Optimizer**:

      The Stochastic Gradient Descent (SGD) optimization algorithm is used for training machine learning models, including neural networks. For a weight ω in the model:

$$\omega_{new} = \omega_{old} - n\nabla j(\omega_{(old)}$$

ω_{old} is the current value of the weight.

$\nabla j(\omega_{(old)}$ is the gradient of the loss function ∇j with respect to the weight ω

$$weight = weight - learning_{rate} * gradient$$

b. **Loss Function:**

The loss function quantifies the difference between the predicted values and the actual labels. It guides the model during training to minimize prediction errors. Binary loss entropy loss.

Formula:

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

7. **Load and Evaluate the Model:**

After training, the saved model (custom_model_class_final.h5) is loaded using load_model. Predictions are made on the validation set. The model's total accuracy on the validation set is printed.

8. **Calculate the confusion matrix for validation data:**

The function calculate_confusion_matrix takes a trained model and a validation dataset directory as input. It initializes an empty confusion matrix based on the number of classes in the validation set. It then generates predictions for the validation set using the model and updates the confusion matrix by comparing true labels with predicted labels.

9. **Calculate the classification report:**

The classification report is generated by comparing the true labels of a dataset with the predicted labels generated by a trained model. Predictions are generated for both the validation and training datasets using the model. The true labels are obtained from the dataset itself. Then, classification reports are generated for both datasets using the true labels and predicted labels. These reports summarize the precision, recall, F1-score, and support for each class in the dataset, providing insights into the model's performance.

Chapter 5: Implementation and Testing

5.1 Implementation

The process of checking and confirming a software or application is called testing. Initial testing of the system by the developer themselves comes first. It aids in ensuring that the system complies with the requirements. This chapter uses pseudocode and diagrams to explain the theory underlying the implementation of intriguing and difficult features. With a well-structured plan and the right development approach, the implementation phase of any project may be the most pleasurable and trouble-free phase.

5.1.1 Tools used.

Different tools and technologies have been used to implement this application. They are listed in the table below:

1. **Programming Languages:**

i Python

The system is developed using Python as the primary programming language, leveraging a set of powerful tools and libraries. File and folder manipulation tasks are handled using the os module, while image processing functionalities are achieved through the cv2 (OpenCV) library for reading, processing, and saving images, alongside the numpy library for numerical operations. The deep learning and neural network components are implemented with the Keras libraries, allowing the construction of a custom model architecture designed for watermark and silver line detection in images. The code also makes use of Python's control flow statements, mathematical functions from the math module, and text file operations for conditional execution, rounding, and saving class indices to a file. Additionally, the matplotlib.pyplot library aids in visualizing training metrics. Overall, Python's versatility and the robust toolset provided by these libraries enable the effective implementation of image processing and machine learning functionalities in the code.

ii. **HTML**

In the system, the structure and styling of a web page for a system named "DETECTO" are defined. The HTML includes metadata, links to external stylesheets and scripts, as well as embedded styles and scripts. The page features a responsive navigation menu, social media icons, and a prominent header with the DETECTO logo. It also incorporates sections for displaying an introductory message, an image upload panel, and a button to trigger image processing. The JavaScript script handles events such as clicking the "See Full Details" button, redirecting users based on a prediction result. This HTML code is integral to the user interface and functionality of the system, facilitating image input, processing, and result presentation. In the documentation, this HTML code should be described as the front-end structure of the DETECTO system, outlining its key components and interactions, especially focusing on user input and result presentation features.

iii. **CSS**

The provided HTML document contains embedded CSS styles that define the visual presentation of the web page. The styles are written within <style> tags in the document's <head> section and include rules for various elements such as containers, images, buttons, and form elements. The CSS utilizes selectors to target specific HTML elements and applies properties like color, padding, positioning, and hover effects to achieve the desired appearance. Additionally, the document incorporates some inline styles directly within HTML elements. The CSS rules also include media queries for responsiveness, ensuring a consistent layout across different screen sizes. Overall, the CSS in this document is structured to create a visually appealing and responsive user interface for the DETECTO website, with specific styling for buttons, image displays, and other elements to enhance the user experience.

iv. **Bootstraps**

Bootstrapping is used to provide more stable training by exposing the models to different subsets of the data in each iteration. It allows you to observe how the models' performance changes when certain instances or features are included or excluded.

v. **Json**

There is a small snippet of JSON code embedded in the HTML within a <script> tag using the type "application/ld+json". This JSON code defines an organization with properties such as name, logo, and social media links. JSON is a lightweight data interchange format that is often used to structure and exchange data between a server and a web application. In this case, it is used for providing structured data about the organization for search engines and other applications that may consume this information.

2. Machine Learning and Computer vision Libraries:

i. **OpenCV**

In the provided code, OpenCV (Open-Source Computer Vision) plays a pivotal role in various image processing and computer vision tasks, serving as a fundamental tool for detecting counterfeit money. The load_images_from_folder function utilizes OpenCV to read and load grayscale images from a specified folder. The extract_features function extensively leverages OpenCV for tasks such as resizing, grayscale conversion, and adaptive histogram equalization for contrast enhancement. The code also employs OpenCV functions like fastNlMeansDenoising and convertScaleAbs for image denoising and contrast adjustment. Notably, template matching methods, such as cv2.matchTemplate, are applied to detect watermark and security thread thread in preprocessed image regions. Overall, OpenCV is essential for manipulating, enhancing, and analyzing images throughout the counterfeit money detection process, showcasing its versatility in image processing within the context of the developed system.

ii. **Scikit-learn**

the scikit-learn (sklearn) library is used in the provided code. Specifically, it is used for train-test splitting and scaling features. The custom_train_test_split function is a custom implementation of a train-test split, which is typically handled by the train_test_split function in scikit-learn. Additionally, a custom standard scaler (CustomStandardScaler) is implemented, which is similar to the functionality provided by scikit-learn's StandardScaler. These scikit-learn functionalities are utilized to preprocess and split the data for training and testing the custom Support Vector Machine (SVM) model.

3. **Deep Learning Frameworks:**

   Keras

   In the provided code, Keras is utilized for constructing, compiling, and training a convolutional neural network (CNN) model. Custom layers, such as CustomDense and inverted_residual_block, are implemented using Keras's Layer class. The MobileNetV2 architecture is defined using Keras layers, and the model is compiled with the Stochastic Gradient Descent (SGD) optimizer, binary crossentropy loss, and accuracy as the evaluation metric. Keras's ImageDataGenerator is employed for data preprocessing and augmentation, facilitating the creation of batches for training and validation. The training process is orchestrated using the fit method, and the ModelCheckpoint callback from Keras is employed to save the best model weights based on validation loss. Finally, the trained model is saved using Keras's save method, resulting in two files: one for the model checkpoint during training (watermark_mobilenetmodel_checkpoint.h5) and another for the final trained model (watermark_mobilenetmodel_final.h5). Keras streamlines the development of deep learning models by providing a high-level API and abstracting many complexities associated with neural network implementation.

4. **Image Processing Tools:**

   Adobe Photoshop or GIMP

   Adobe Photoshop or GIMP is used to reduce noise or artifacts in the images, ensuring that the input to the detection models is clean and representative of genuine currency features. Adjusting the brightness, contrast, and sharpness of the images can enhance the visibility of subtle features in the currency notes. The role of these image editing tools is to prepare and enhance the data for effective model training and evaluation.

5. **Development Frameworks:**

   Flask

   In the provided code, Flask is utilized to create a web application for predicting whether an uploaded image contains counterfeit money. Flask is a Python web framework that simplifies the process of building web applications. The application defines routes, such as "/", "/result", and "/predictt", each associated with specific functionalities. When a user accesses the specified routes, Flask invokes the corresponding functions, rendering HTML templates, processing image uploads, and providing predictions. Flask seamlessly handles HTTP requests and responses, enabling the integration of the image processing logic, SVM prediction, and web interface. Additionally, the app.run() method is employed to launch the Flask application, allowing it to run locally with debugging capabilities. The code demonstrates the use of Flask's simplicity and flexibility in building a web-based counterfeit money detection system.

6. **Diagram tool**

   Draw.io plays a pivotal role in our Counterfeit Money Detection System's development by serving as the go-to tool for Entity-Relationship (ER) diagrams and Data Flow Diagrams (DFD), offering a comprehensive visualization of data structures and process flows. Meanwhile, Figma takes the lead in crafting the system's user interface, ensuring an intuitive and visually appealing design.

7. **SQLite**

   In your project, SQLite is employed as a relational database to manage user-related data, currency images, and detection results. The user_id table likely contains user information, while the currency_image table stores images along with relevant details, linked to users via foreign keys. The detection_result table records outcomes of currency detection processes, associating results with specific users and currency images. SQL queries enable seamless interaction with these tables, facilitating tasks such as data insertion, retrieval, and management within the application. Overall, SQLite provides a lightweight and efficient solution for organizing and accessing essential data in the context of user accounts, currency images, and detection results.

5.1.2 Implementation Details of Algorithm:

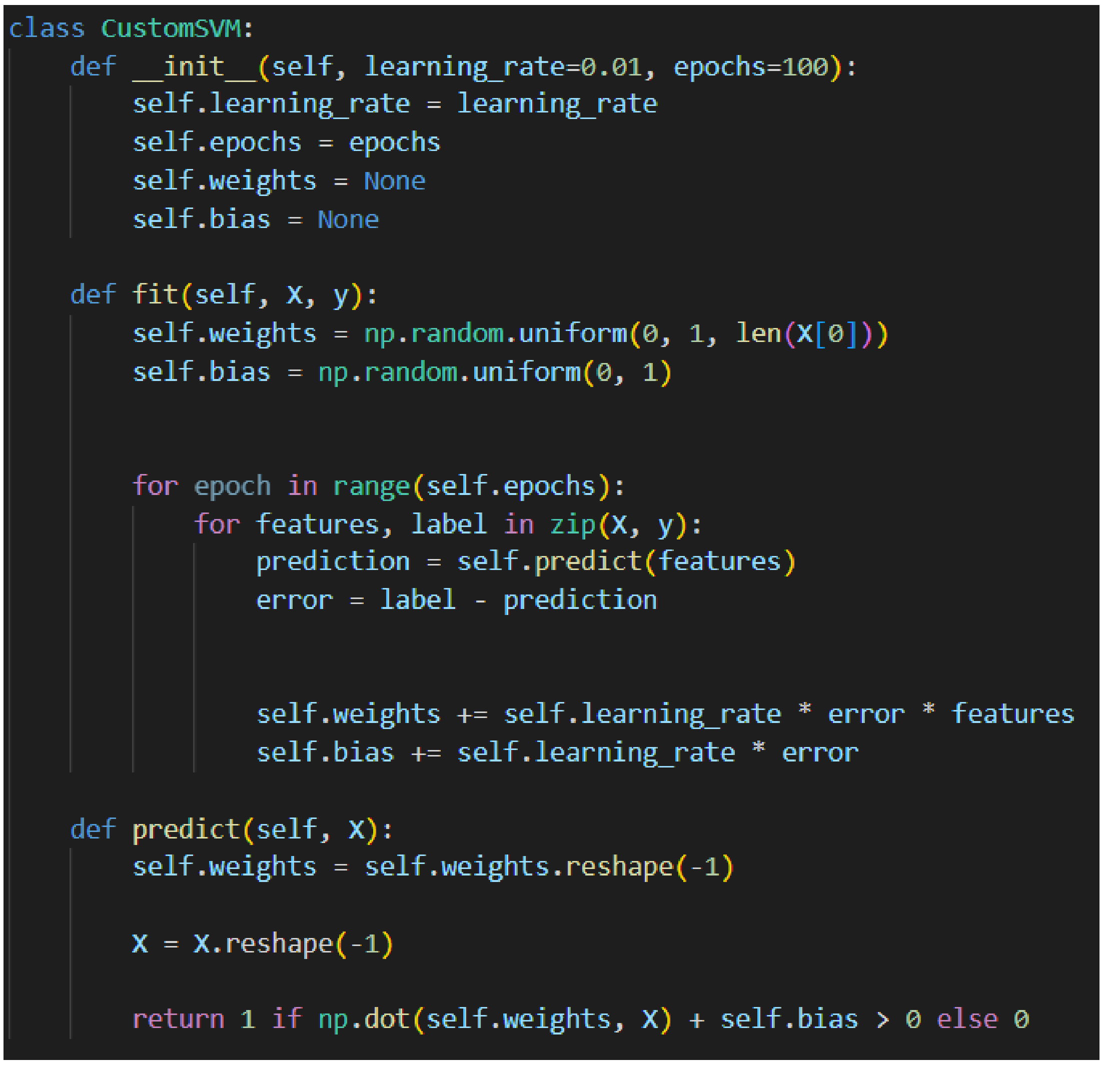
Implementation of SVM algorithm for Counterfeit money detection

1. **Custom SVM Implementation:**
   The system uses a custom SVM implementation named CustomSVM. It defines a custom SVM class (CustomSVM) with an adjustable learning rate and number of epochs. This class contains methods for model initialization, fitting (training), and prediction. The SVM model is trained on the scaled training features using the fit method of the CustomSVM class. The trained SVM model is evaluated on both the training and testing sets by comparing the predicted labels with the actual labels. Accuracy metrics for both sets are calculated and printed

2. **Model Initialization:**
   Within the CustomSVM class, the weights and bias of the SVM model are initialized randomly during instantiation. These parameters will be adjusted during the training process to create an optimal decision boundary.

3. **Model Training (fit)**
   The SVM model is trained using a simple gradient descent approach in the fit method of the CustomSVM class. The weights and bias are updated iteratively based on the error between the predicted and actual labels for each training example.

4. **Prediction Function:**
   The predict method of the CustomSVM class is responsible for making predictions based on the learned weights and bias. It calculates the dot product of the feature vector and weights, then applies a threshold to determine the predicted class (1 or 0).

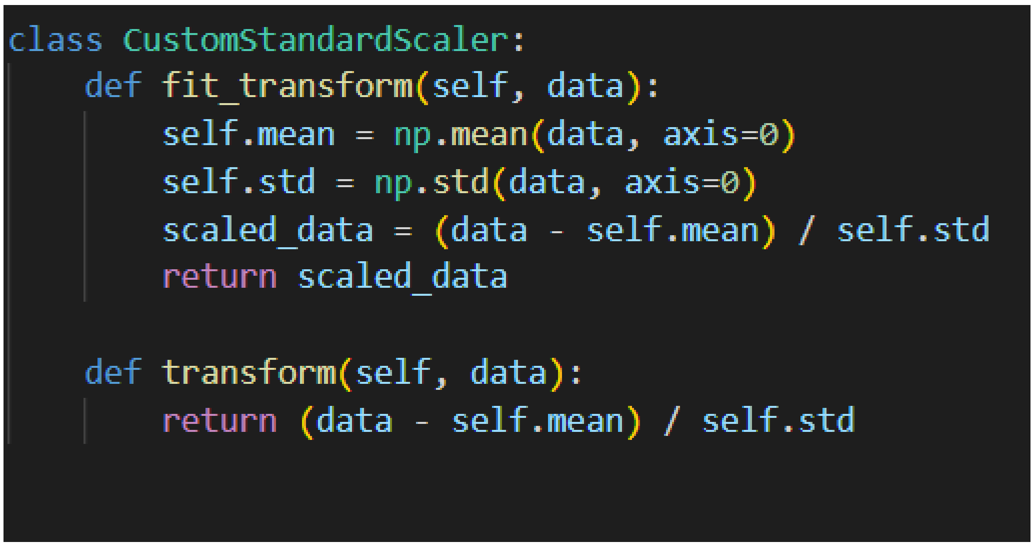


```python
class CustomSVM:
    def __init__(self, learning_rate=0.01, epochs=100):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        self.weights = np.random.uniform(0, 1, len(X[0]))
        self.bias = np.random.uniform(0, 1)


        for epoch in range(self.epochs):
            for features, label in zip(X, y):
                prediction = self.predict(features)
                error = label - prediction


                self.weights += self.learning_rate * error * features
                self.bias += self.learning_rate * error

    def predict(self, X):
        self.weights = self.weights.reshape(-1)

        X = X.reshape(-1)

        return 1 if np.dot(self.weights, X) + self.bias > 0 else 0
```

Figure 5. 1 Custom SVM

5. **Feature Scaling:**

   A custom standard scaler (CustomStandardScaler) is defined and applied to scale the features. The scaler's mean and standard deviation are used to normalize the data.

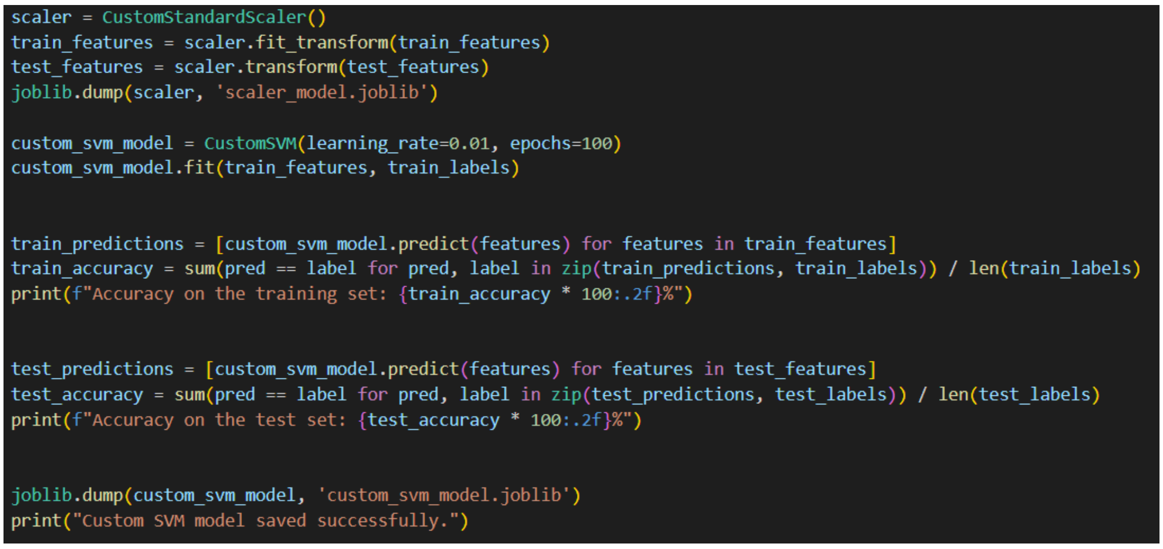


```
class CustomStandardScaler:
    def fit_transform(self, data):
        self.mean = np.mean(data, axis=0)
        self.std = np.std(data, axis=0)
        scaled_data = (data - self.mean) / self.std
        return scaled_data

    def transform(self, data):
        return (data - self.mean) / self.std
```

Figure 5. 2 CustomStandardScalar

6. **Model Saving:**

   The trained SVM model and the scaler used for feature scaling are saved as joblib files ('custom_svm_model.joblib' and 'scaler_model.joblib', respectively).



```
scaler = CustomStandardScaler()
train_features = scaler.fit_transform(train_features)
test_features = scaler.transform(test_features)
joblib.dump(scaler, 'scaler_model.joblib')

custom_svm_model = CustomSVM(learning_rate=0.01, epochs=100)
custom_svm_model.fit(train_features, train_labels)


train_predictions = [custom_svm_model.predict(features) for features in train_features]
train_accuracy = sum(pred == label for pred, label in zip(train_predictions, train_labels)) / len(train_labels)
print(f"Accuracy on the training set: {train_accuracy * 100:.2f}%")


test_predictions = [custom_svm_model.predict(features) for features in test_features]
test_accuracy = sum(pred == label for pred, label in zip(test_predictions, test_labels)) / len(test_labels)
print(f"Accuracy on the test set: {test_accuracy * 100:.2f}%")


joblib.dump(custom_svm_model, 'custom_svm_model.joblib')
print("Custom SVM model saved successfully.")
```

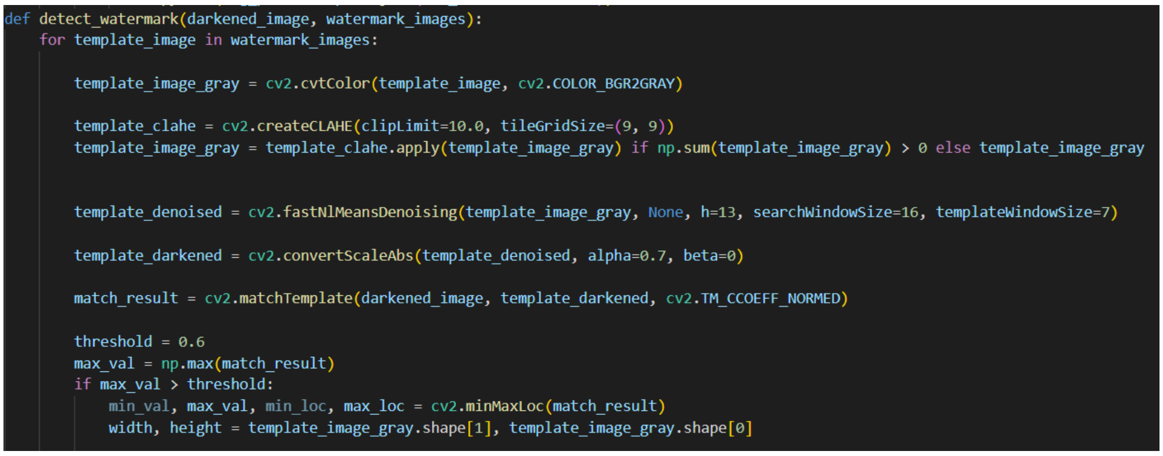
Figure 5. 3 Model Saving

Implementation of CNN algorithm:

Implementing a Convolutional Neural Network (CNN) involves several steps, including defining the architecture, preparing the data, training the model, and evaluating its performance.

1. **Preprocessing Images (preprocess input function):**
   It primarily utilizes techniques to enhance image quality and identify watermark. Images are resized, converted to grayscale, and undergo Contrast Limited Adaptive Histogram Equalization (CLAHE) for improved contrast. Non-Local Means Denoising is then applied to reduce noise. A watermark detection mechanism uses template matching, and if a match exceeds a threshold, the watermark is considered detected. The script also splits the dataset into training and validation sets, aiding the subsequent training of a custom convolutional neural network (CNN) model for effective counterfeit money detection.

2. **Watermark detection.**
   The watermark detection involves preprocessing both input images and watermark templates. Template matching is applied, measuring normalized cross-correlation. If the match surpasses a threshold, a bounding box is drawn on the image, signifying watermark detection. The script outputs a matching percentage for confidence assessment.

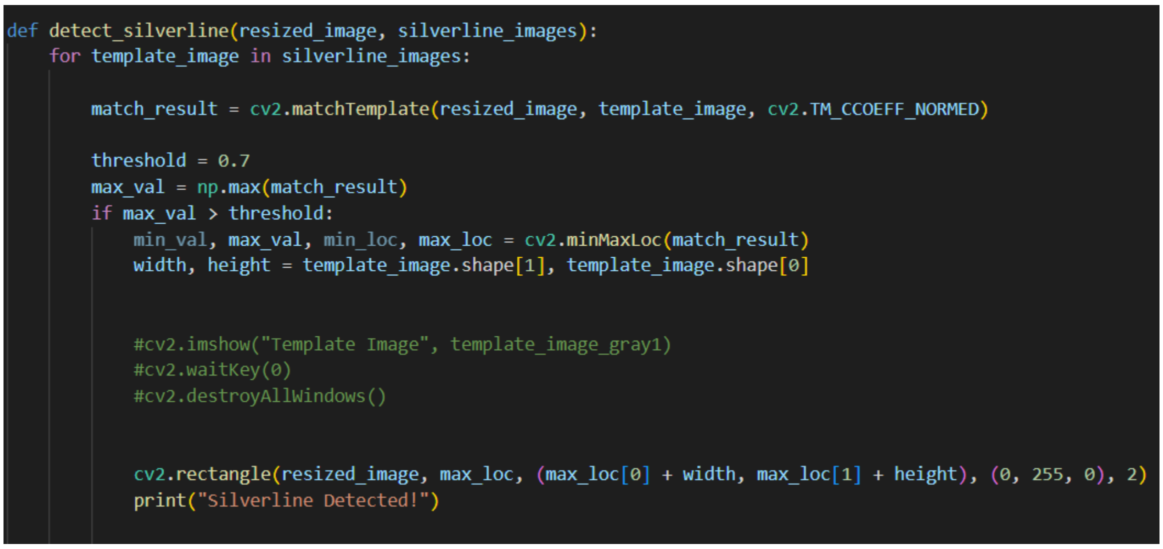


```python
def detect_watermark(darkened_image, watermark_images):
    for template_image in watermark_images:

        template_image_gray = cv2.cvtColor(template_image, cv2.COLOR_BGR2GRAY)

        template_clahe = cv2.createCLAHE(clipLimit=10.0, tileGridSize=(9, 9))
        template_image_gray = template_clahe.apply(template_image_gray) if np.sum(template_image_gray) > 0 else template_image_gray


        template_denoised = cv2.fastNlMeansDenoising(template_image_gray, None, h=13, searchWindowSize=16, templateWindowSize=7)

        template_darkened = cv2.convertScaleAbs(template_denoised, alpha=0.7, beta=0)

        match_result = cv2.matchTemplate(darkened_image, template_darkened, cv2.TM_CCOEFF_NORMED)

        threshold = 0.6
        max_val = np.max(match_result)
        if max_val > threshold:
            min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(match_result)
            width, height = template_image_gray.shape[1], template_image_gray.shape[0]
```

Figure 5. 4 Watermark Detection

3. **Security thread Detection:**

   The **detect_silverline** function uses template matching (**cv2.matchTemplate**) to identify security thread in the preprocessed images. If a security thread is detected, a bounding box is drawn around it, and the matching percentage is printed.

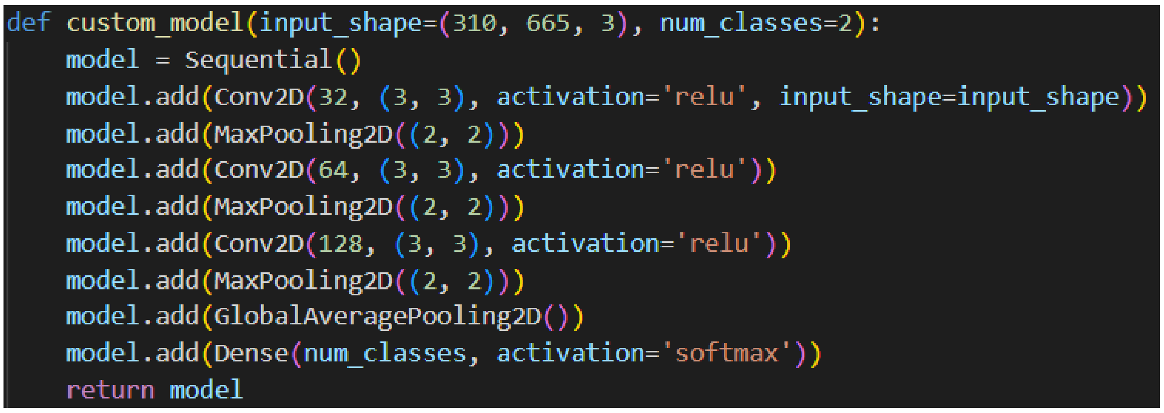


```
def detect_silverline(resized_image, silverline_images):
    for template_image in silverline_images:

        match_result = cv2.matchTemplate(resized_image, template_image, cv2.TM_CCOEFF_NORMED)

        threshold = 0.7
        max_val = np.max(match_result)
        if max_val > threshold:
            min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(match_result)
            width, height = template_image.shape[1], template_image.shape[0]


            #cv2.imshow("Template Image", template_image_gray1)
            #cv2.waitKey(0)
            #cv2.destroyAllWindows()


            cv2.rectangle(resized_image, max_loc, (max_loc[0] + width, max_loc[1] + height), (0, 255, 0), 2)
            print("Silverline Detected!")
```

Figure 5. 5 Security Thread Detection

4. **Custom CNN model**

   The custom Convolutional Neural Network (CNN) model is designed for detecting counterfeit money. It utilizes Keras, featuring convolutional and pooling layers, culminating in a dense layer for binary classification. The model is trained using SGD with a learning rate of 0.001 and binary cross-entropy loss. ModelCheckpoint saves the best-performing model during training. The process includes preprocessing images, training with an ImageDataGenerator, and visualizing accuracy and loss curves. The trained model is then saved for future use.

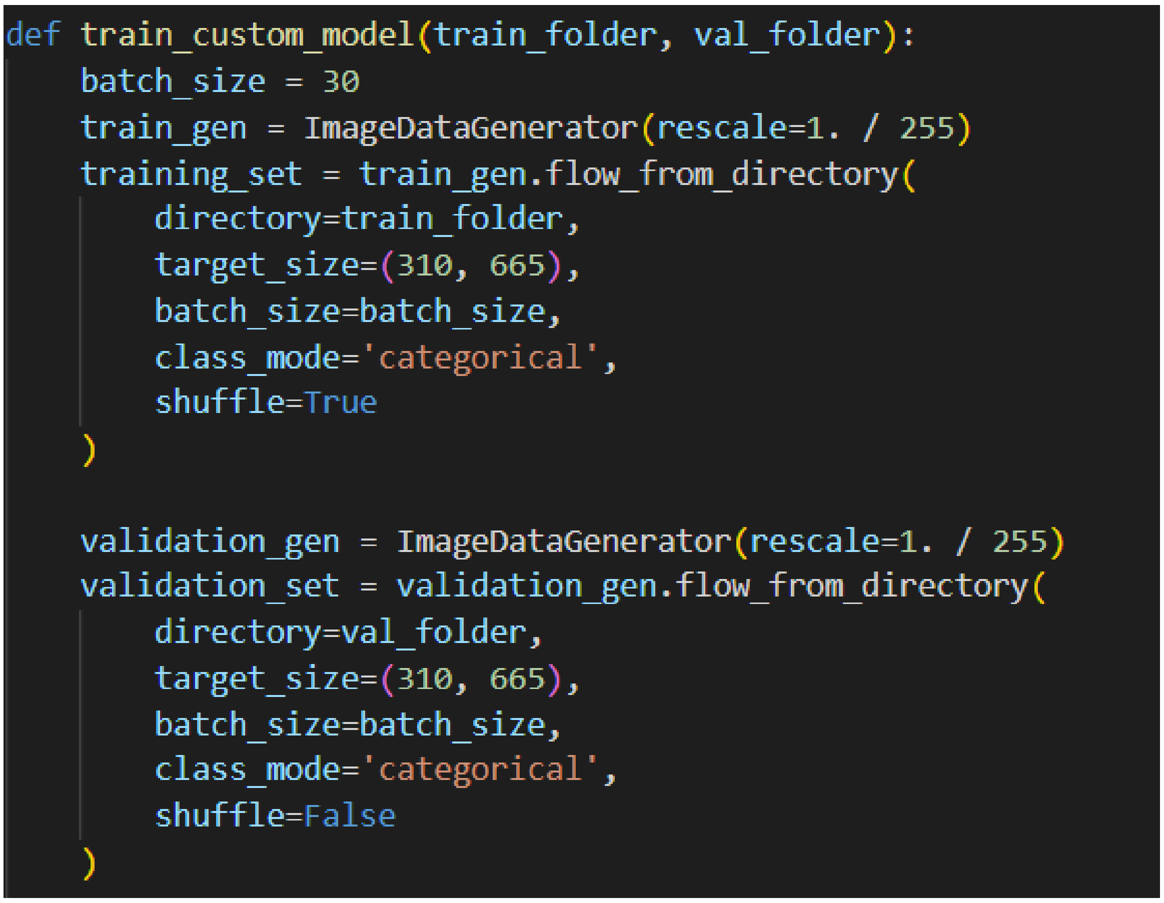


```
def custom_model(input_shape=(310, 665, 3), num_classes=2):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(GlobalAveragePooling2D())
    model.add(Dense(num_classes, activation='softmax'))
    return model
```

Figure 5. 6 Custom CNN model

5. Training custom CNN

The training of the custom Convolutional Neural Network (CNN) model for counterfeit money detection involves preprocessing images and splitting them into training and validation sets. Images are resized, enhanced, and subjected to watermark detection before being organized into respective folders. The CNN model, designed with convolutional and pooling layers, is trained using an ImageDataGenerator for augmentation. The training process spans five epochs with a batch size of 30, employing Stochastic Gradient Descent (SGD) as the optimizer with a learning rate of 0.001 and binary cross-entropy loss. ModelCheckpoint is utilized to save the best-performing model based on validation loss. The trained model, alongside class indices, is saved for future use. The training progress is visualized through accuracy and loss curves, providing insights into the model's performance.



```python
def train_custom_model(train_folder, val_folder):
    batch_size = 30
    train_gen = ImageDataGenerator(rescale=1. / 255)
    training_set = train_gen.flow_from_directory(
        directory=train_folder,
        target_size=(310, 665),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=True
    )

    validation_gen = ImageDataGenerator(rescale=1. / 255)
    validation_set = validation_gen.flow_from_directory(
        directory=val_folder,
        target_size=(310, 665),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False
    )
```

Figure 5. 7 Training Custom CNN

5.2 Testing

5.2.1 Test Cases for Unit Testing

Unit testing is performed for testing modules against detailed design. Every step of the project's design and coding has been tested. While testing, we test the module interface to make sure that data is properly flowing into and out of the program unit. By inspecting the local data structure, we ensure that the temporarily stored data keeps its integrity during the algorithm's execution. Finally, each path that handles errors is tested.

Table 5. 1 Test case For Registration

ID	Test Case Description	Test Data	Expected result	Actual Result	Test Result
R_1	User enters valid signup information	Enter your email: salinamalla2001@gmail.com Create password: 123456 Confirm password: 123456	Redirects to Login Page	Redirected to Login page	Pass
R_2	User enters different password and confirm password	Enter Your email: salinamalla2001@gmail.com Create password: 123456 Confirm password: 23459	Display Error Message.	"Password and Confirm password didn't match."	Pass

Table 5. 2 Test Case for User Signup

ID	Test Case Description	Test Data	Expected result	Actual Result	Test Result
L_1	User enters valid login information	Enter your email: salinamalla2001@gmail.com Enter your password: 123456	Redirects to Currency detection page	Redirected to Currency detection page	Pass

L_2	User enters invalid login information	Enter your email: salina@gmail.com Enter your password: 098765	Display Error Message	“Please enter a correct username and password.”	Pass
L_3	User leaves the login fields empty.	Enter your email: salinamalla2001@gmail.com Enter your password:	Display Error Message	“Please enter password."	Pass

Table 5. 3 Test for Feature Extraction

S.N.	Test Case	Test Description	Input Image	Expected Result	Actual Result	Remarks
1	Test-1	Compare feature of image with the template	Real note	Real	Real	Pass
2	Test-2	Compare feature of image with the template	Fake note	Fake	Fake	Pass

Table 5. 4 Test Case for Model Accuracy

S.N.	Test Case	Test Description	Input Image	Expected Result	Actual Result	Remarks
1	Test-1	Test on test data of the dataset	Real note	High Accuracy		Pass
2	Test-2	Test on test data of the dataset	Fake note	Low Accuracy		Pass

5.2.2 Test Cases for System Testing

System testing is typically done to look for faults brought on by unexpected interactions between subsystem and system components. Once the source code is developed, software must be tested to identify and correct any potential mistakes before being delivered to the client.

Table 5. 5 System Testing

S.N.	Test Case	Test Description	Expected Outcome	Actual Outcome	Remarks
1	Test-1	Upload Real Image	Real Image	Real Image	Pass
2	Test-2	Upload Fake Image	Fake Image	Fake Image	Pass

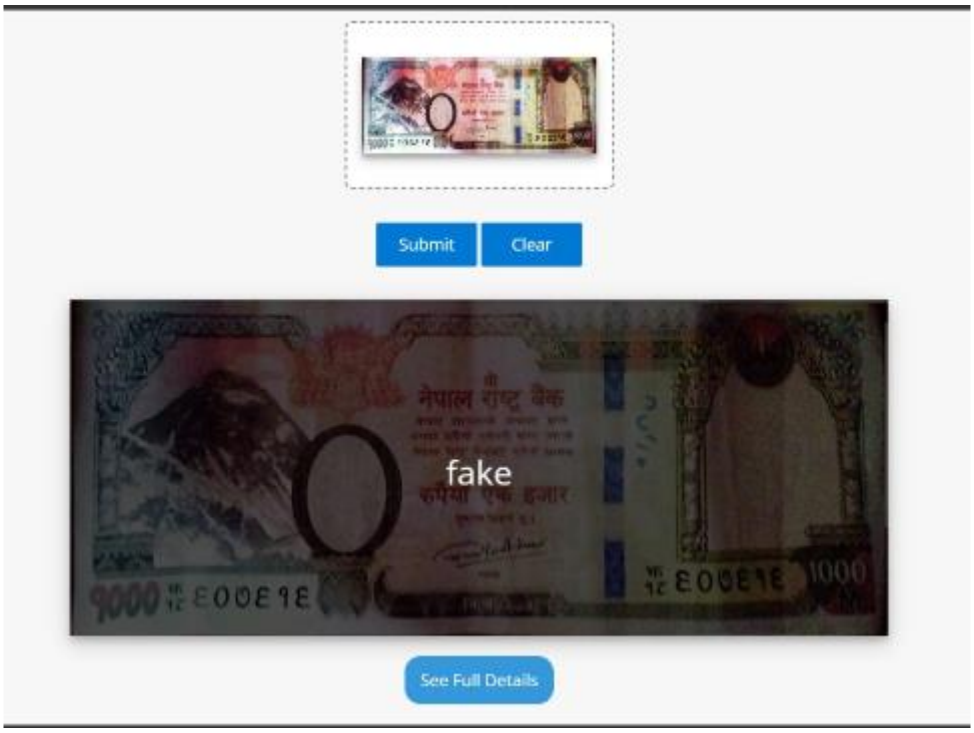


Figure 5. 8 Fake image

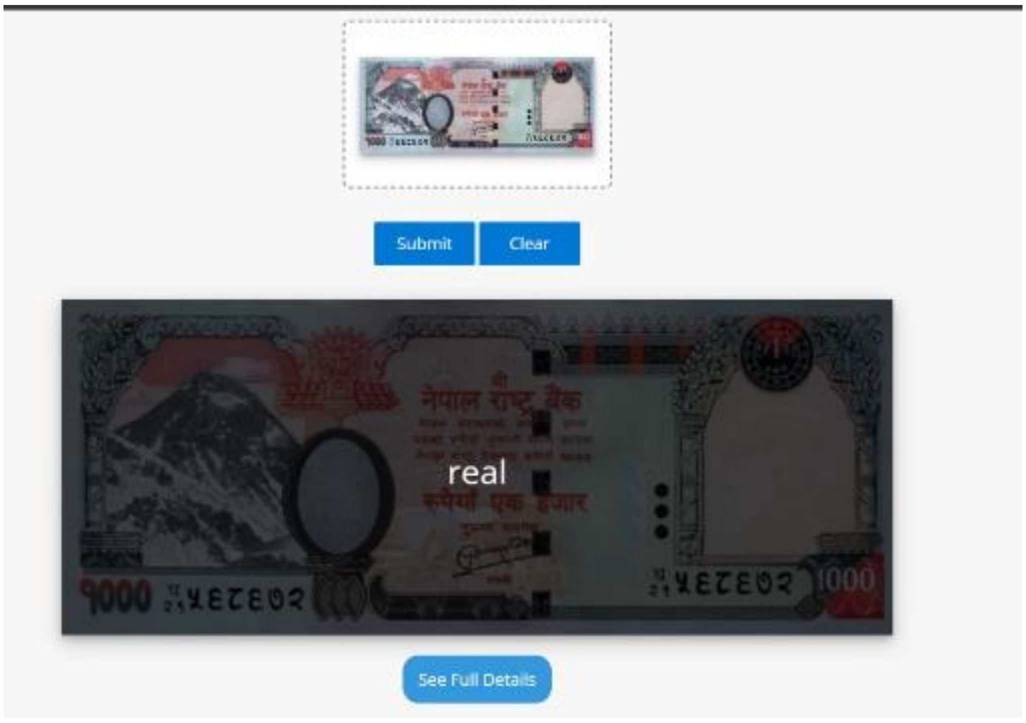


Figure 5. 9 Real Image

5.3 Result Analysis

The analysis of the counterfeit money detection system's results provides crucial insights into its performance and effectiveness. By meticulously examining various performance metrics, we aim to ascertain the system's ability to accurately distinguish between genuine and counterfeit currency. This analysis serves as a critical step in evaluating the system's reliability, identifying areas for improvement, and ensuring its robustness in real-world scenarios.

1. **Data Collection**

   Our counterfeit money detection system benefits from a robust dataset comprising a total of 472 samples. This dataset was meticulously compiled from the repository https://amitness.com/ml-datasets/ to ensure comprehensive coverage and relevance to our detection system's objectives. This dataset encompasses images depicting authentic banknotes alongside their counterfeit counterparts. Notably, it features a comprehensive array of Nepalese currency denominations, including 10, 50, 100, and 1000 rupee notes. Despite the availability of various denominations within the dataset, we deliberately opted to concentrate solely on the 1000-rupee denomination. This decision was made to streamline our development efforts, allowing us to focus exclusively on optimizing the detection system's efficacy in identifying counterfeit 1000-rupee notes within the Nepalese context. Furthermore, it's important to note that we captured the images ourselves and incorporated them into the dataset, ensuring a high degree of relevance and authenticity for our training data.

2. Epoch Progress: Training Metrics

   The training and testing accuracy of an SVM model trained over 100 epochs is represented in the graph below.

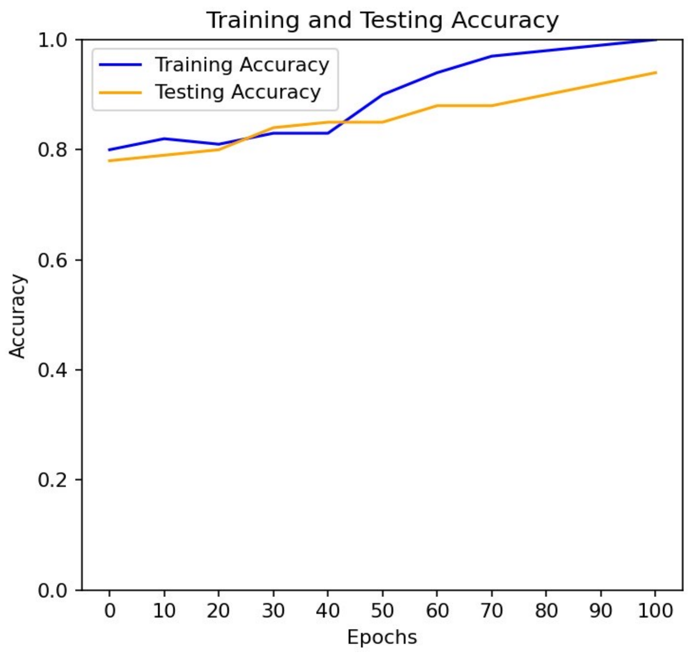


Figure 5. 10 SVM Training and Testing Accuracy

The training and testing loss of an SVM model trained over 100 epochs is represented in the graph below.

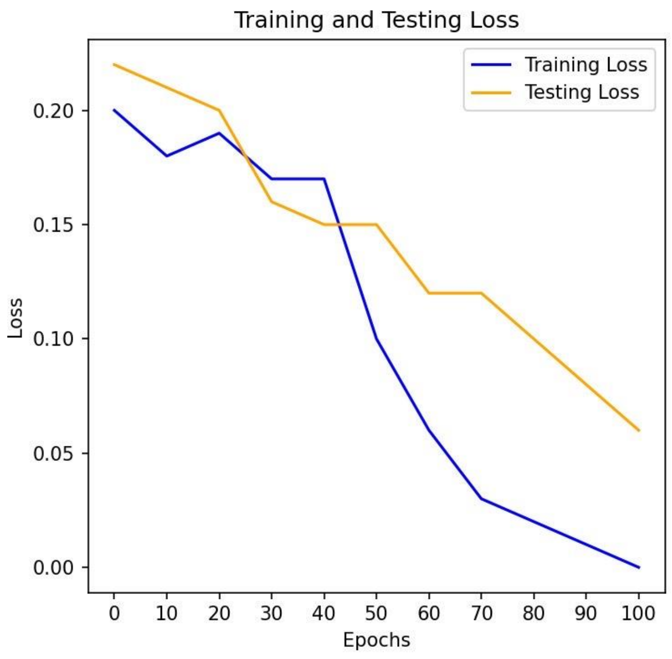


Figure 5. 11 SVM Training and Testing Loss

The training and validation accuracy of CNN model for Watermark is trained over 100 epochs is represented in the graph.

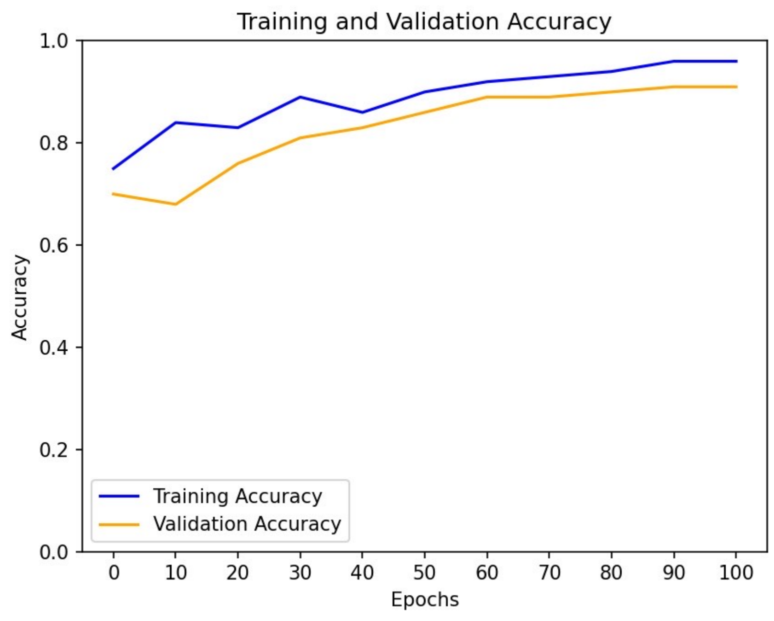


Figure 5. 12 Watermark Training and Validation Accuracy

The training and validation loss of CNN model for Watermark is trained over 100 epochs is represented in the graph.

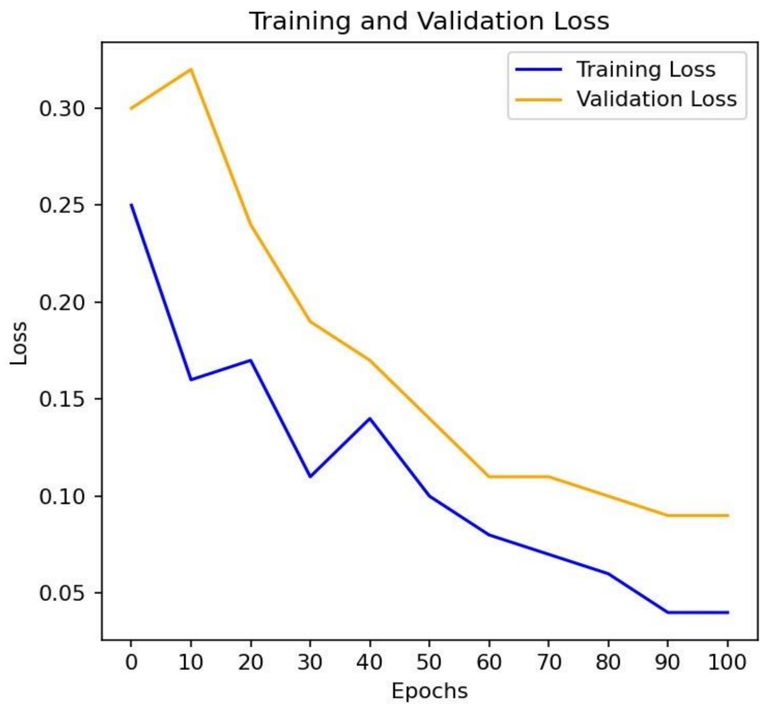


Figure 5. 13 Watermark Training and Validation Loss

The training and validation accuracy of CNN model for Security Thread is trained over 100 epochs is represented in the graph.

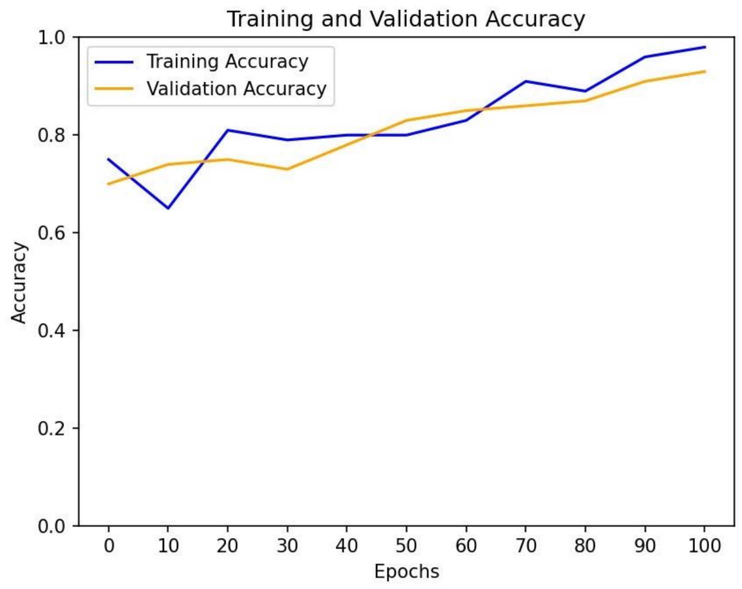


Figure 5. 14 Security Thread Training and Validation Accuracy

The training and validation loss of CNN model for Security Thread is trained over 100 epochs is represented in the graph.

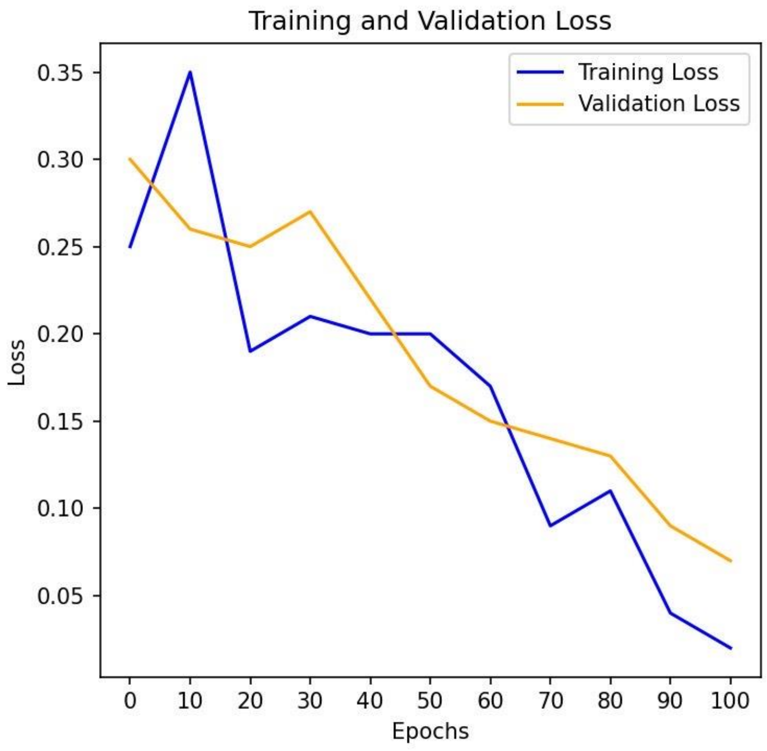


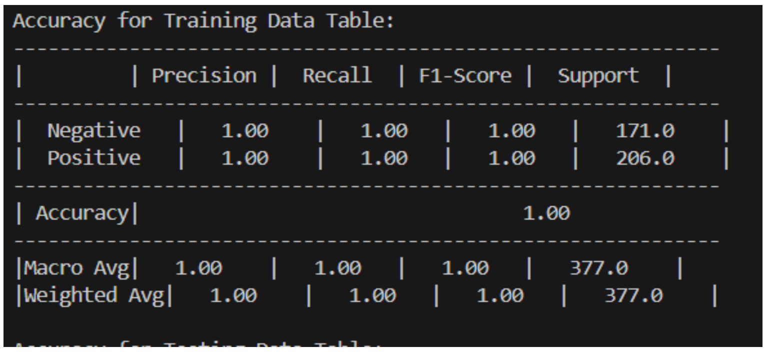
Figure 5. 15 Security Thread Training and Validation Loss

3. Performance Metrics

Evaluating our counterfeit money detection system involves assessing key performance metrics. Accuracy measures how often it correctly identifies counterfeit bills. Precision and recall indicate the system's ability to accurately classify counterfeit bills and identify genuine ones, respectively. The F1 Score provides a balanced evaluation of overall performance, while false positive and false negative rates reveal the system's tendencies for misclassification. By analyzing these metrics, we can determine the system's reliability in distinguishing between genuine and counterfeit currency.

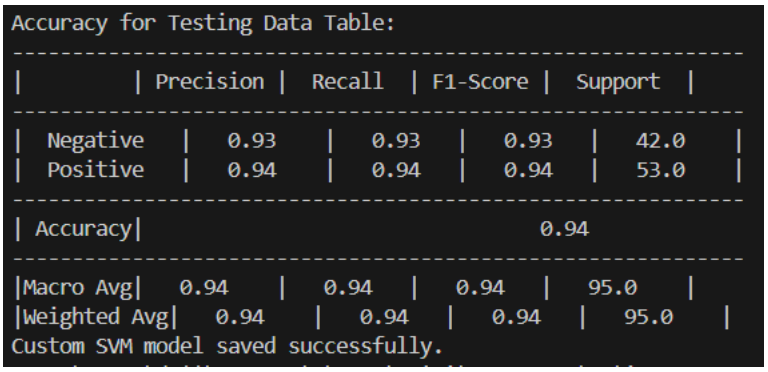
Performance metrics were computed separately for two components of our system: the Support Vector Machine (SVM) for classification and the Convolutional Neural Network (CNN) for determining the similarity percentage of watermark and security thread.

The SVM classifier was trained on a dataset containing examples of both genuine and counterfeit currency. This dataset provided the basis for the SVM to learn how to distinguish between the two classes using features extracted from the currency images. We then assessed how well the classifier performs on this training data to understand its ability to correctly classify genuine and counterfeit currency instances.



```
Accuracy for Training Data Table:
------------------------------------------------------------
|         | Precision |  Recall  | F1-Score |  Support  |
------------------------------------------------------------
|  Negative   |   1.00    |   1.00   |   1.00   |   171.0    |
|  Positive   |   1.00    |   1.00   |   1.00   |   206.0    |
------------------------------------------------------------
| Accuracy|                            1.00
------------------------------------------------------------
|Macro Avg|   1.00    |   1.00   |   1.00   |   377.0    |
|Weighted Avg|   1.00    |   1.00   |   1.00   |   377.0    |
```

Figure 5. 16 SVM: Accuracy for Training Data Table

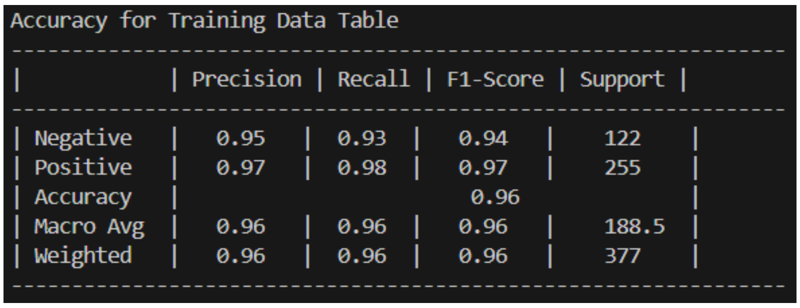


```
Accuracy for Testing Data Table:
------------------------------------------------------------
|          | Precision |  Recall  | F1-Score |  Support  |
------------------------------------------------------------
|  Negative  |   0.93    |   0.93   |   0.93   |   42.0    |
|  Positive  |   0.94    |   0.94   |   0.94   |   53.0    |
------------------------------------------------------------
| Accuracy|                          0.94
------------------------------------------------------------
|Macro Avg|   0.94    |   0.94   |   0.94   |   95.0    |
|Weighted Avg|   0.94    |   0.94   |   0.94   |   95.0    |
Custom SVM model saved successfully.
```

Figure 5. 17 SVM: Accuracy for Testing Data Table

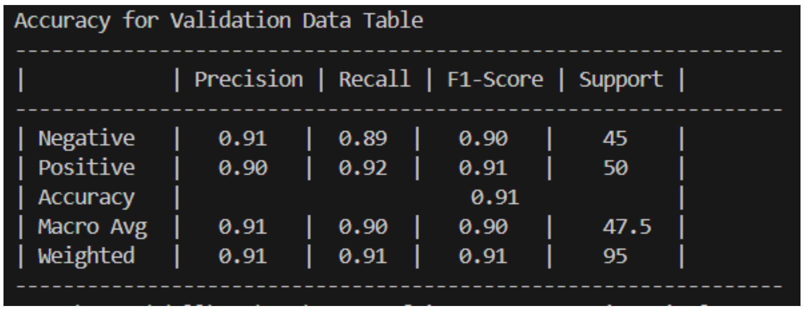
The SVM classifier trained on genuine and counterfeit currency data exhibited exceptional performance metrics, including perfect classification accuracy, precision, recall, F1-score, and support for both genuine and counterfeit currency instances. For the SVM classifier tested on genuine and counterfeit currency data, it achieved an overall accuracy of 94% on the testing data, demonstrating strong performance with high precision, recall, and F1-scores for both genuine and counterfeit currency classes.

The performance metrics for counterfeit money detection, specifically focusing on determining the similarity percentage of watermark and security thread using CNN, provide valuable insights into the model's efficacy.



```
Accuracy for Training Data Table
--------------------------------------------------------------
|           | Precision | Recall | F1-Score | Support |
--------------------------------------------------------------
| Negative   |   0.95    |  0.93  |   0.94   |    122    |
| Positive   |   0.97    |  0.98  |   0.97   |    255    |
| Accuracy   |                        0.96               |
| Macro Avg  |   0.96    |  0.96  |   0.96   |    188.5  |
| Weighted   |   0.96    |  0.96  |   0.96   |    377    |
--------------------------------------------------------------
```

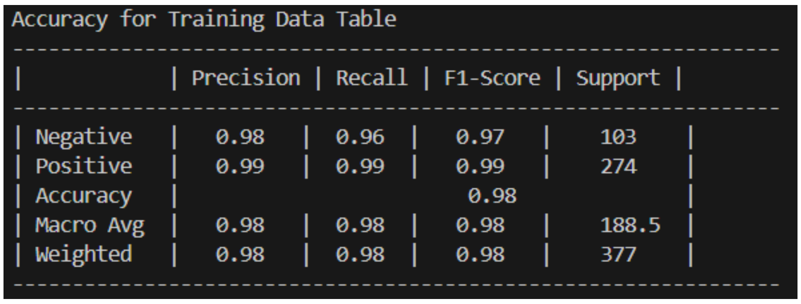
Figure 5. 18 CNN (Watermark): Accuracy for Training Data Table



```
Accuracy for Validation Data Table
--------------------------------------------------------------
|           | Precision | Recall | F1-Score | Support |
--------------------------------------------------------------
| Negative  |   0.91    |  0.89  |   0.90   |   45    |
| Positive  |   0.90    |  0.92  |   0.91   |   50    |
| Accuracy  |                       0.91              |
| Macro Avg |   0.91    |  0.90  |   0.90   |   47.5  |
| Weighted  |   0.91    |  0.91  |   0.91   |   95    |
--------------------------------------------------------------
```

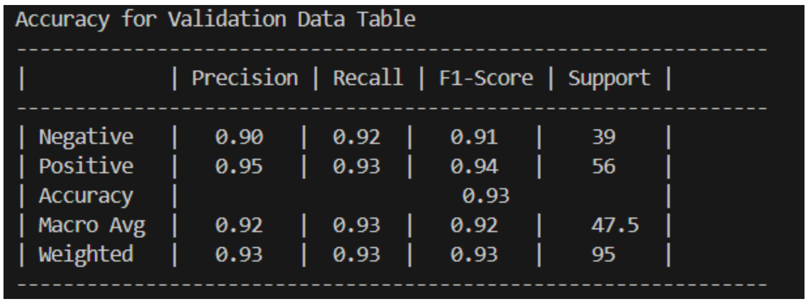
Figure 5. 19 CNN (Watermark): Accuracy for Validation Data Table

For CNN analysis of watermark similarity, the model achieved high precision, recall, F1-scores, and accuracy of 96%, indicating its effectiveness in distinguishing between genuine and counterfeit currency.



```
Accuracy for Training Data Table
--------------------------------------------------------------
|           | Precision | Recall | F1-Score | Support |
--------------------------------------------------------------
| Negative  |   0.98    |  0.96  |   0.97   |   103    |
| Positive  |   0.99    |  0.99  |   0.99   |   274    |
| Accuracy  |                       0.98               |
| Macro Avg |   0.98    |  0.98  |   0.98   |   188.5  |
| Weighted  |   0.98    |  0.98  |   0.98   |   377    |
--------------------------------------------------------------
```

Figure 5. 20 CNN (Security Thread): Accuracy for Training Data Table



```
Accuracy for Validation Data Table
--------------------------------------------------------------
|           | Precision | Recall | F1-Score | Support |
--------------------------------------------------------------
| Negative  |   0.90    |  0.92  |   0.91   |   39    |
| Positive  |   0.95    |  0.93  |   0.94   |   56    |
| Accuracy  |                       0.93              |
| Macro Avg |   0.92    |  0.93  |   0.92   |   47.5  |
| Weighted  |   0.93    |  0.93  |   0.93   |   95    |
--------------------------------------------------------------
```

Figure 5. 21 CNN (Security Thread): Accuracy for Validation Data Table

Similarly, for CNN analysis of security thread similarity, the model showcased commendable proficiency with an accuracy of 93%.

In a counterfeit money detection system using SVM classification with CNN for matching percentage evaluation, the confusion matrix provides a concise summary of the system's performance, offering insights into the accuracy and reliability of the system in distinguishing between genuine and counterfeit currency. By analyzing these metrics, the system's effectiveness can be evaluated, guiding further improvements to enhance its performance in identifying counterfeit currency accurately.

4. Confusion Matrix

In a counterfeit money detection system using SVM classification with CNN for matching percentage evaluation, the confusion matrix provides a concise summary of the system's performance. It outlines four key scenarios: True Positives (genuine currency correctly classified), True Negatives (counterfeit currency correctly classified), False Positives (counterfeit currency classified as genuine), and False Negatives (genuine currency classified as counterfeit), offering insights into the accuracy and reliability of the system in distinguishing between genuine and counterfeit currency. By analyzing these metrics, the system's effectiveness can be evaluated, guiding further improvements to enhance its performance in identifying counterfeit currency accurately.

The confusion matrix for an SVM algorithm is as follows:

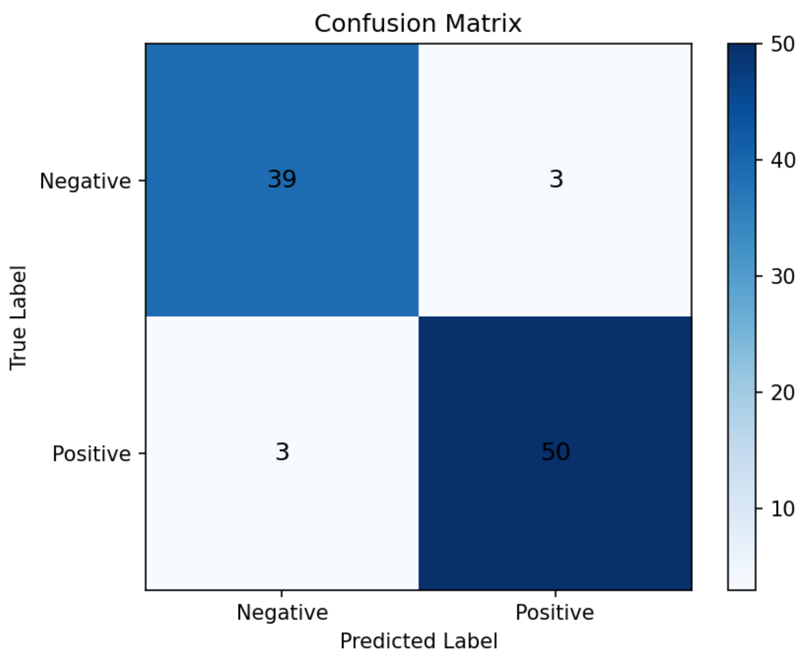


Figure 5. 22 SVM: Confusion Matrix

The confusion matrix for a CNN analysis of watermark similarity is as follows:

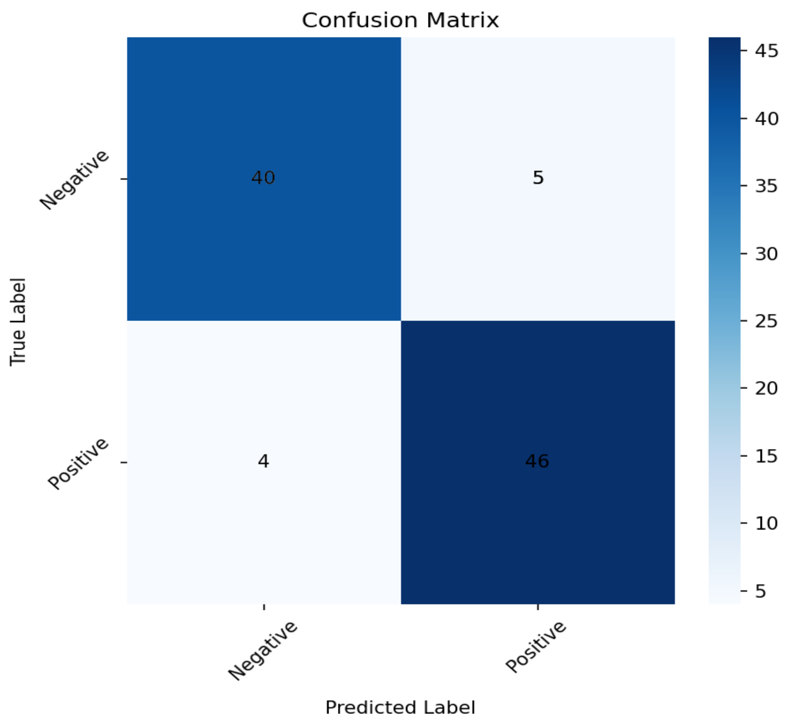


Figure 5. 23 CNN (Watermark): Confusion Matrix

The confusion matrix for a CNN analysis of security thread similarity is as follows:

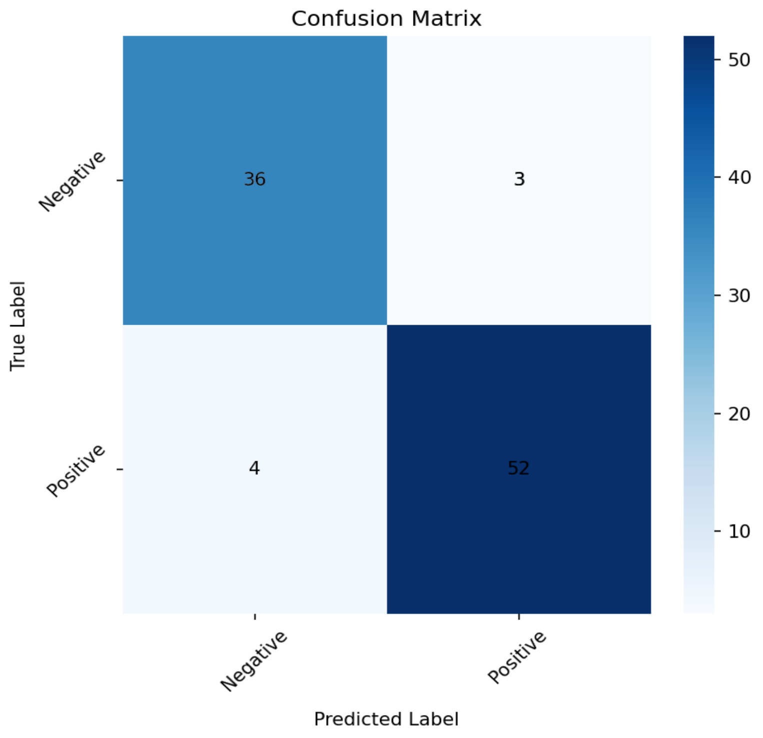


Figure 5. 24 CNN (Security Thread): Confusion Matrix

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

The implementation of a counterfeit money detection system utilizing Support Vector Machine (SVM) and Convolutional Neural Network (CNN) algorithms offers a promising solution to the prevalent issue of fraudulent currency circulation. Through the integration of SVM and CNN algorithms, the system effectively determines the authenticity of uploaded currency image by analyzing key features such as watermark and security thread.

The SVM algorithm serves as the initial line of defense, effectively discerning between authentic and counterfeit currency based on crucial features such as watermark and security thread present in the uploaded image. Through intuitive user interaction, where a threshold value can be adjusted to fine-tune the detection process, users gain real-time insights into the authenticity of the currency, enabling informed decision-making. The accuracy of testing data using SVM stands at 94%.

Moreover, the incorporation of CNN enhances the system's functionality by providing users with detailed insights into the matching percentage of watermark and security thread of the uploaded image. This algorithm offers more comprehensive details about the uploaded image. The validation data accuracy for detecting the watermark is 91%, while for detecting the security thread, validation data accuracy is 93%.

6.2 Future Recommendation

Technology is advancing at a rapid pace these days. In this system, we were able to detect watermark and security thread. We've looked at the whole image of money so far, but in the future, we'll try to include all of the security features of money by using a fair fundamental structure and providing sufficient preparation information. When a picture is loaded into the training folder from the outside, it does not provide 100 percent accuracy. By optimizing the system, we can solve this problem.

References

[1] A. Vidhate, Y. Shah, H. Keshri and R. Nikhare, "Fakecurrency detection application," *Int. Res. J. Eng. Technol. (IRJET),* vol. 08(05), pp. 2395-0056, 2021.

[2] A. Upadhyaya and D. V. Shokeen, "“ANALYSIS OF COUNTERFEIT CURRENCY DETECTION TECHNIQUES FOR CLASSIFICATION MODEL”," in *International Conference on Computing Communication and Automation (ICCCA)*, 2018.

[3] D. S. Virakthamath and T. Kshama, "“REVIEW ON DETECTION OF FAKE CURRENCY USING IMAGE PROCESSING TECHNIQUES”," in *International Conference on Intelligence Computing and Control System, (ICICCS)*, 2021.

[4] A. Zarin and J. Uddin, "" A HYBRID FAKE BANK NOTE DETECTION MODEL USING OCR, FACE RECOGNITION AND HOUGH FEATURES"," in *Cybersecurity and Cyberforensics (CCC),* 2019.

[5] S. R. Darade and P. G. R. Gadveer, "“AUTOMATIC RECOGNITION OF FAKE INDIAN CURRENCY NOTE”," in *International Conference on Electrical Power and Energy System, (ICEPES)*, 2016.

[6] A. Yadav and T. Jain, "" EVALUATON OF MACHINE LEARNING ALGORITHMS FOR DETECTION OF FAKE BANK CURRENCY”," in *International Conference on Cloud Computing, Data Science & Engineering (ICCCDE),* 2021.

[7] s. Gothe, K. Naik and V. Joshi, *Fake currency detection using image processing and machine learning,* 2018.

[8] V. L. Nadh and S. Prasad, "Support vector machine in the anticipation of currency markets," *Int. J. Eng. Technol,* vol. 7(2), p. 66–68, 2018.

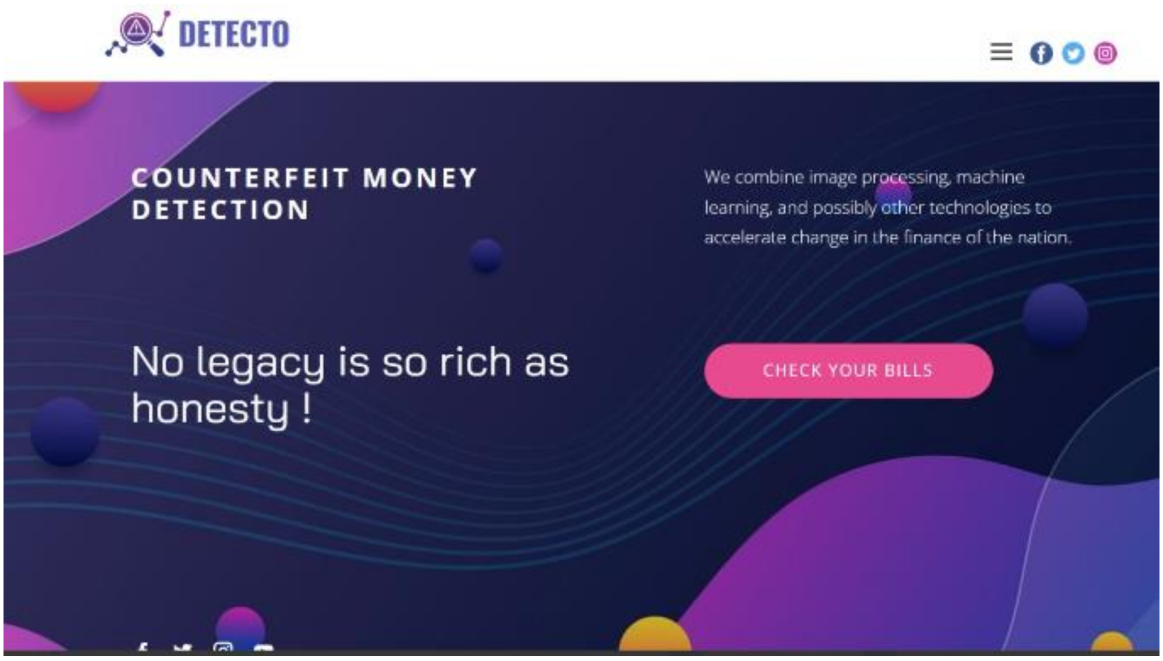
[9] V. V and L. M, "Real time fake currency note detection using deep learning.," *Int. J. Eng.Adv. Technol. (IJEAT),* vol. 9(1S5), pp. 2249-8958, 2019.

[10] A. T, B. G, W. P and C. P, "Fake currency detection using image processing," *IOPConf.Ser.Mater.Sci.Eng.,* p. 263, 2017.

[11] T. Agasti, "Fake currency detection using image processing," *IOP Conference Series: Materials Science and Engineering,* p. 243, 2017.

[12] "Currency Management Department," Nepal Rastra Bank, [Online]. Available: https://www.nrb.org.np/departments/cmd/. [Accessed 4 October 2023].

Appendix



COUNTERFEIT MONEY DETECTION

We combine image processing, machine learning, and possibly other technologies to accelerate change in the finance of the nation.

No legacy is so rich as honesty !

Figure 7. 1 Home Page

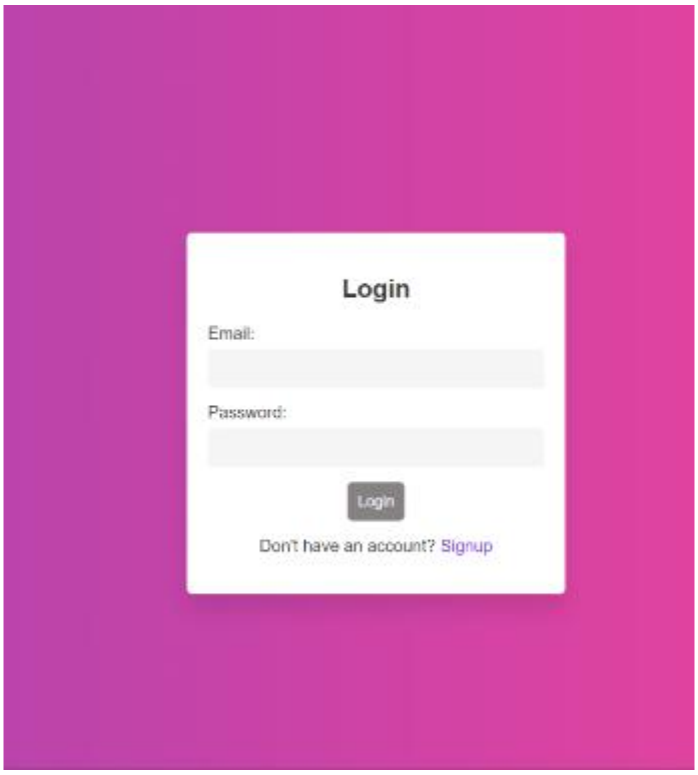


Figure 7. 2 Login Page

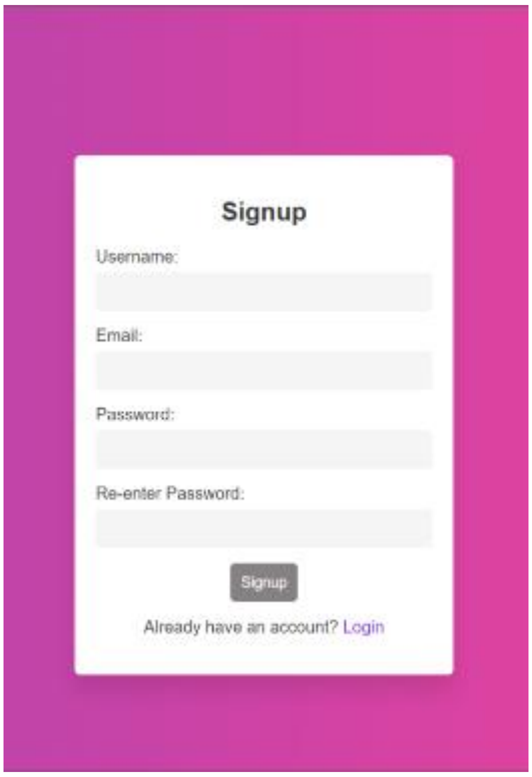


Figure 7. 3 Signup Page

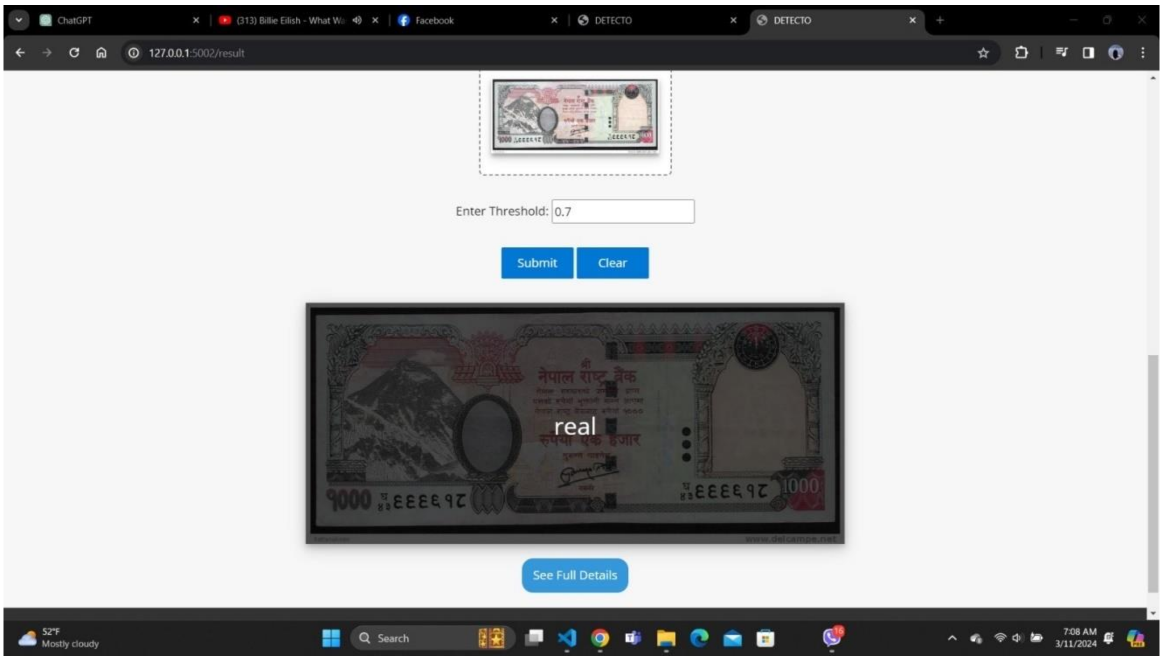
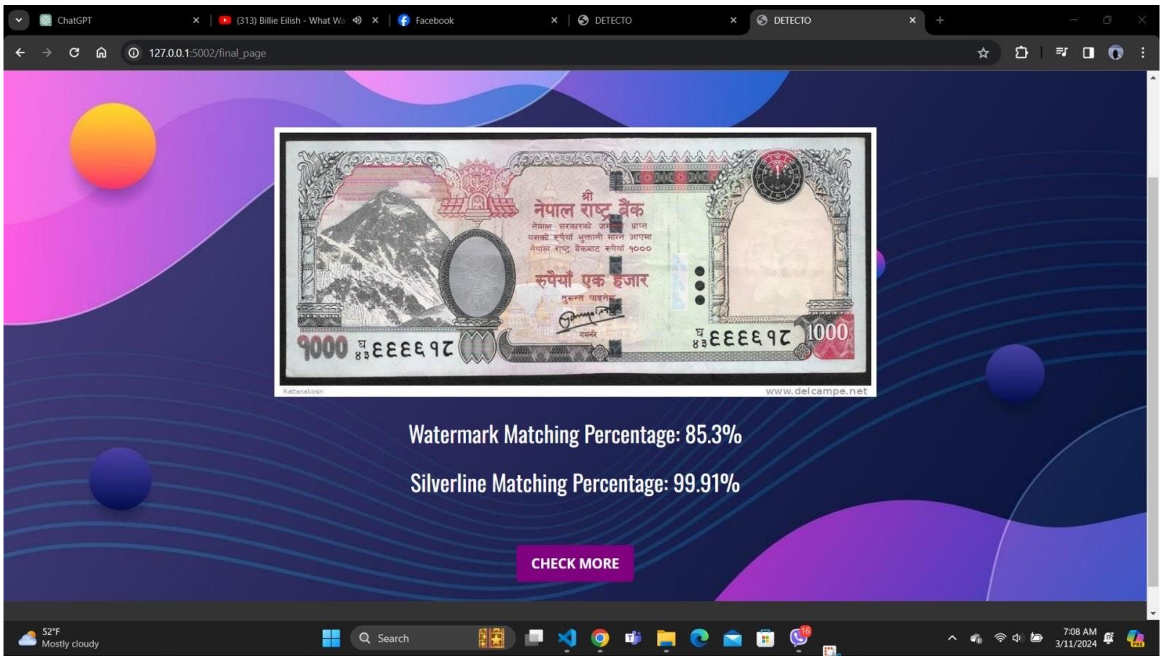


Figure 7. 4 Image with 0.7 threshold



Watermark Matching Percentage: 85.3%

Silverline Matching Percentage: 99.91%

Figure 7. 5 Matched Percentage

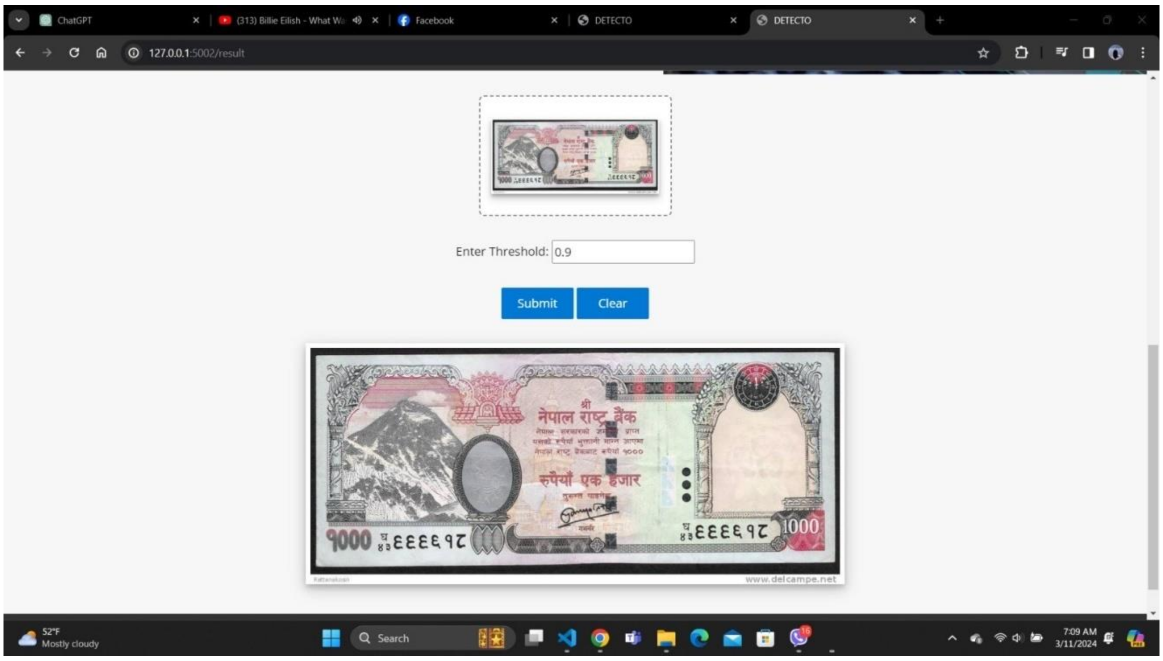
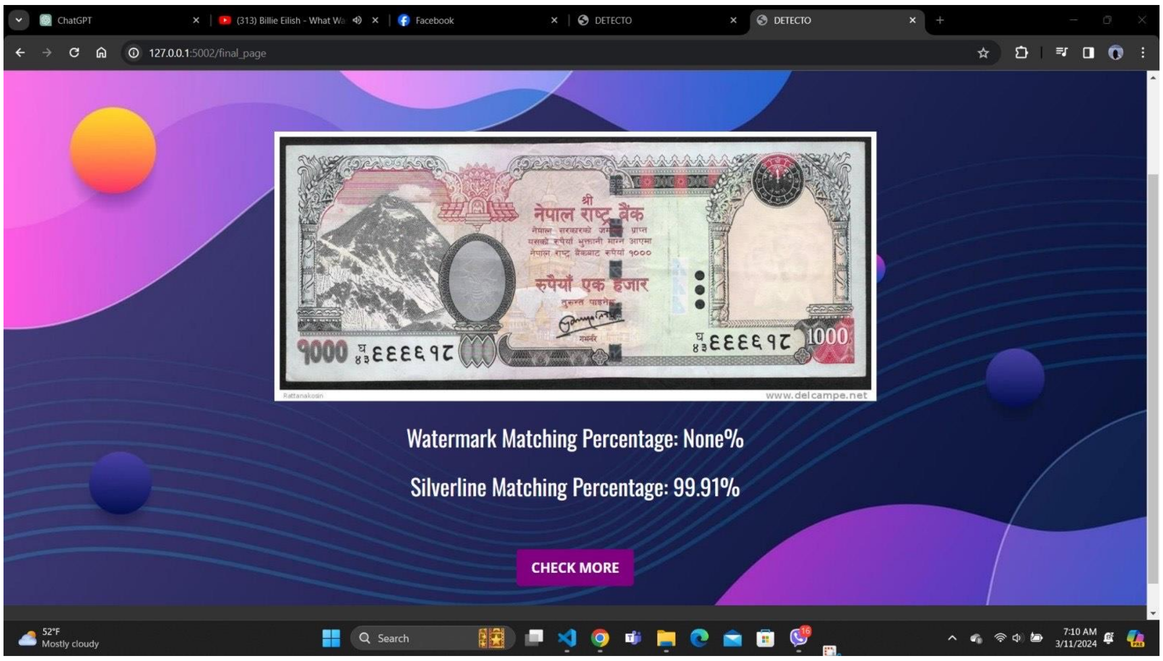


Figure 7. 6 Image with 0.9 Threshold



Watermark Matching Percentage: None%

Silverline Matching Percentage: 99.91%

Figure 7. 7 Fake image input