A Project Phase - I Report

On

Fake Currency Detection Using Deep Learning

Submitted in partial fulfillment of the requirements of the degree of

BACHELOR OF ENGINEERING (Computer Engineering)

By

SURYAWANSHI MAN NANA	**72221583F**
HIRE RUSHIKESH PRALHAD	**72266103H**
ASHMEER RAZA ALI AKHTER	**72221529M**
JAFRI SHUJAT ALI	**72266105D**

Guide:

Dr. Salman Baig

Department of Computer Engineering
Maulana Mukhtar Ahmad Nadvi Technical Campus,
Malegaon- 423203

(Nov, 2024)

C E R T I F I C A T E

This is to certify that the project report entitled '**Fake Currency Detection Using Deep Learning',** submitted by

SURYAWANSHI MAN NANA	**72221583F**
HIRE RUSHIKESH PRALHAD	**72266103H**
ASHMEER RAZA ALI AKHTER	**72266105D**
JAFRI SHUJAT ALI	**72266105D**

in the partial fulfillment of the requirement for the award of degree of Bachelor of Engineering (Computer Eng.) of MMANTC, Malegaon, affiliated to the Savitribai Phule Pune University is a record of their own work.

Dr. Salman Baig
Name of the Guide

Dr. Salman Baig
Head of the Department

Date:

Place: Malegaon

This report entitled

FAKE CURRENCY DETECTION USING DEEP LEARNING

By

SURYAWANSHI MAN NANA	**72221583F**
HIRE RUSHIKESH PRALHAD	**72266103H**
ASHMEER RAZA ALI AKHTER	**72221529M**
JAFRI SHUJAT ALI	**72266105D**

is approved for the degree of

Bachelor of Engineering

of

Department of Computer Engineering

Maulana Mukhtar Ahmad Nadvi Technical Campus, Malegaon 423203

Examiners	Name	Signature
1. External Examiner	----------	----------
2. Internal Examiner	----------	----------
3. Supervisor (s)	----------	----------

Date:

Place: Malegaon

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of the Student	PRN Number	Signature
SURYAWANSHI MAN NANA	72221583F	
HIRE RUSHIKESH PRALHAD	72266103H	
ASHMEER RAZA ALI AKHTER	72221529M	
JAFRI SHUJAT ALI	72266105D	

Date:

Place: Malegaon

ACKNOWLEDGEMENT

It is my great pleasure to acknowledge sense of gratitude to all, who have made it possible for us to complete this Project work with success. It gives me great pleasure to express my deep gratitude to my Project guide *Dr Md Salman Baig* for his support and help from time to time during Project work. It is my pleasure to acknowledge sense of gratitude to Head of Department *Dr. Salman Baig* for their great support and encouragement in seminar work. I would also like to thank my Principal *Dr. Aqueel Ahmad Shah* for his valuable recommendations. I would like to express my gratitude towards our dear parents for their support, and constant encouragement At last but not in least, we would like to thank everyone who directly and indirectly helped and motivated us to work on this project.

ABSTRACT

In today's digital age, the prevalence of counterfeit currency poses a significant challenge to financial institutions and society as a whole. Traditional detection methods often rely on manual inspection, which can be time-consuming and prone to human error. To address this issue, this study explores the use of deep learning, specifically the VGG 16 model, for the automated detection of fake currency. The proposed approach leverages the powerful feature extraction capabilities of the VGG 16 convolutional neural network to classify genuine and counterfeit currency notes. The model is trained on a comprehensive dataset of genuine and fake currency images, ensuring robust performance and generalization to a wide range of currency types and counterfeiting techniques. Through extensive experimentation and testing, the deep learning-based system achieves an impressive accuracy of around 96% in distinguishing genuine and counterfeit currency. This high level of performance not only enhances the efficiency of detection but also reduces the risk of financial losses and maintains the integrity of the monetary system. The key advantages of this approach include its scalability, real-time processing capabilities, and the ability to adapt to evolving counterfeiting methods. By integrating this deep learning-based fake currency detection system, financial institutions can strengthen their security measures, protect their customers, and contribute to the overall stability of the financial ecosystem. The findings of this study demonstrate the significant potential of deep learning techniques, such as the VGG 16 model, in addressing the pressing challenge of counterfeit currency detection. The successful implementation of this system can pave the way for further advancements in the field of financial fraud prevention and contribute to the development of more secure and resilient financial systems.

TABLE OF CONTENTS

List of Figures

List Of Tables

CHAPTER 1 - INTRODUCTION

1.1 Overview

Computers and mobile phones have become an unavoidable part of our lives. There are a lot of things which we can do with these technologies. With the rapid development of mobile phones and technologies come several services like application creation - (refers to the process of making application software for handheld and desktop devices such as personal computers and Personal Digital Assistants. Fake currency Detection is a system that can be used to overcome the limitations most of the people and our institutions of higher learning face with respect to making difference between counterfeit currencies- (is imitation currency produced without the legal sanction of the state or government, usually in a deliberate attempt to imitate that currency and so as to deceive its recipient) and real currencies. The project involves making use of Digital Image Processing Domain - Digital image processing is the use of computer algorithms to perform image processing on digital images. [1],[2].

This has led to the increase of corruption in our country hindering the country's growth. Some of the methods to detect fake currency are watermarking, optically variable ink, security thread, latent image, techniques like counterfeit detection pens. We hereby propose an application system for detecting fake currency where image processing is used to detect fake notes [3]. We will find out dissimilarities between the image under consideration and the prototype. CNN classifiers will be used to detect fake currency. The proposed system for fake currency detection will be simple, accurate and easy to use.

1.2 Problem Statement

In exiting and ongoing concepts of different adaptations, tests and innovations had been kept for the future due to the lack of time. Few of the major problem are:

- Speed of system is slow
- Not able to keep track of device’s location through which the currency is scanned and maintaining the same in the database.
- Limited database capacity for storing large amounts of location-related data over time, leading to storage challenges

1.3 Objectives

- The main objective of fake currency detection project is to detect fake currency using deep convolutional neural network.
- To provide cheaper and accurate system to the user, which can be easily accessible and give accurate results.
- To develop a user-friendly application for currency recognition system.
- To make it available for common people quicky and easily with low cost.

1.4 Methodology

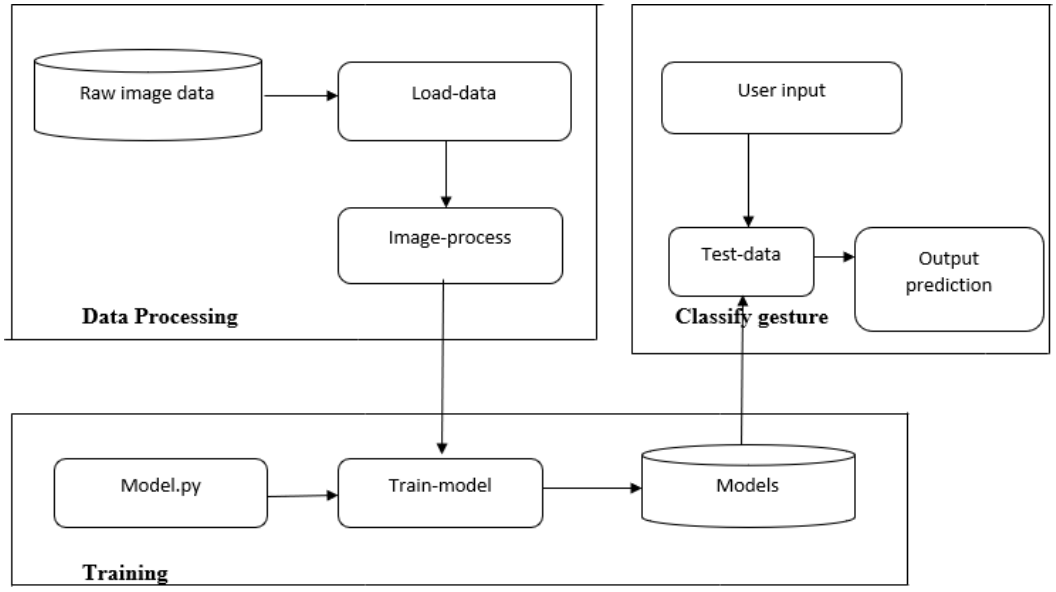


Figure 1.1: Methodology

The main stages involved in this method are Data Collection, Pre-processing, Feature extraction, prediction model and evaluation.

Step 1: Gather Your Dataset

The first component of building a deep learning network is to gather our initial dataset. We need the images themselves as well as the labels associated with each image. These labels should come from a finite set of categories, such as: categories fake and real currency.

Furthermore, the number of images for each category should be approximately uniform (i.e., the same number of examples per category) then our classifier will become naturally biased to overfitting into these heavily-represented categories.

Class imbalance is a common problem in machine learning and there exist a number of ways to overcome it. We'll discuss some of these methods later, but keep in mind the best method to avoid learning problems due to class imbalance is to simply avoid class imbalance entirely. As our system is mainly focusing on detection of Fake currency, we gathered our data as images. The dataset obtained consisted of several images of real and fake currencies[14].

Step 2: Split Your Dataset and pre-processing

Now that we have our initial dataset, we need to split it into two parts:
1.A training set
2.A testing set
A training set is used by our classifier to "learn" what each category looks like by making predictions on the input data and then correct itself when predictions are wrong. After the classifier has been trained, we can evaluate the performing on a testing set.

It's extremely important that the training set and testing set are independent of each other and do not overlap! If you use your testing set as part of your training data, then your classifier has an unfair advantage since it has already seen the testing examples before and "learned" from them. Instead, you must keep this testing set entirely separate from your training process and use it only to evaluate your network.[14].

Common split sizes for training and testing sets include 66:6%33:3%, 75%=25%, and 90%=10%, respectively. (Figure 1.2):

Figure 1.2: Examples of common training and testing data splits.

These data splits make sense, but what if you have parameters to tune? Neural networks have a number of knobs and levers (ex., learning rate, decay, regularization, etc.) that need to be tuned and dialed to obtain optimal performance. We'll call these types of parameters hyperparameters, and it's critical that they get set properly. In practice, we need to test a bunch of these hyperparameters and identify the set of parameters that works the best . You might be tempted to use your testing data to tweak these values, but again, this is a major no-no! The test set is only used in evaluating the performance of your network[5]. Instead, you should create a third data split called the validation set. This set of the data (normally) comes from the training data and is used as "fake test data" so we can tune our hyperparameters. Only after have we determined the hyperparameter values using the validation set do we move on to collecting final accuracy results in the testing data. We normally allocate roughly 10-20% of the training data for validation. If splitting your data into chunks sounds complicated, it's actually not.

Pre-processing

The primary target is to improve image highlights needed for additional processing. Here, the input image is converted into grayscale image for all the further preprocessing purposes. The image is then thresholded and further erosion and dilation is applied to the thresholded image. This image is used to extract the contours and extreme points [4].

Step 3: Train Your Network

Given our training set of images, we can now train our network. The goal here is for our network to learn how to recognize each of the categories in our labeled data. When the model makes a mistake, it learns from this mistake and improves itself. So, how does the actual "learning" work? In general, we apply a form of gradient descent.

Step 4: Evaluate

Last, we need to evaluate our trained network. For each of the images in our testing set, we present them to the network and ask it to predict what it thinks the label of the image is. We then tabulate the predictions of the model for an image in the testing set.

Finally, these model predictions are compared to the ground-truth labels from our testing set. The ground-truth labels represent what the image category actually is. From there, we can compute the number of predictions our classifier got correct and compute aggregate reports such as precision, recall, and f-measure, which are used to quantify the performance of our network as a whole.

Convolutional Neural Network

CNN is utilized to get better result. The signal convolved with kernels to get include map. Past layers are interconnected with weights of the kernel. To upgrade the qualities of information image by back propagation calculation. Since feature maps of all units shared by the kernels. It will serve to reduces over fitting. Each data of neighborhood is taken by utilizing kernels. Kernel is a major source of context information. Activation function is applied to the output of neural network [6],[7].

Objective of the convolution layer is to take or extract the features from the input [image], just the part of picture is link to the following convolution layer.

Padding

Padding is incorporating a zero layer outside the input volume so the data on border won't be lost and we can get a similar dimension of output as input volume. Here we are using zero padding.

Activation Function
Non- linear activation function ReLU(Rectifier Activation function) is used to provide accurate results than classical sigmoid function.

Pooling Layer
It is used for combining spatially nearby features. Max-pooling is generally used to join features. It decreases the dimension of input image and controls over- fitting.

1.5 Organization of the Report

This report is organized into majorly into 5 different sections and each section provides detailed/ brief description about the project. The 5 sections mentioned are :

Introduction – This section provides you overview of the project, what's the major problem that is being addressed, objectives, which methodology we are following to implement this project and information about remaining part of the report

Literature Survey – This section provides previous work of this problem and their limitations .

SRS – System requirement Specification section provides information about functional and non-functional requirements of this project .

Expected outcome – This gives an idea of how the outcome would be.

Advantages and Applications – What are the advantages of the approach or framework being developed comparatively to previous existing one and information regarding application of the project in various fields.

CHAPTER 2 - LITERATURE SURVEY

2.1 Counterfeit currency detection using deep convolutional neural network (2019) presented by Prof Kiran Kamble, Anuthi Bhansali, Pranali Satalgaonkar, Shruthi Alagundg.

In this relevant paper, many recognition techniques are implemented to recognize images, recognize faces, recognize car license plates, and recognize human behaviours. Currency is the primary average for circulation, and Various countries' currencies have different qualities. However, when the value of currency grows, there will be an increase in counterfeit currency. Counterfeit money might damage these nations' interests. As a result, one of the hottest subjects and a critical issue at the moment is how to use recognition technology to the genuine of money (Zhang, 2018) [6],[15]. Visual examination was used in the past to identify and genuine money, particularly currency notes. Our eyesight cannot sense everything; sometimes, it is not easy for humans to distinguish genuine currency from auth genuine entice currency without the aid of technology.

Disadvantages

- Consumes more time to produce output
- Consumes more space

2.2 Fake currency Detection using Basic Python Programming and Web Framework (2020) presented by Prof Chetan More, Monu Kumar, Rupesh Chandra, Raushan Singh

Currency duplication also known as counterfeit currency is a vulnerable threat on economy. Although fake currency is being printed with precision, the Crime Investigation Department (CID) says that they can be detected with some effort. Currency printed by local racketeers can be detected easily as they use the photographic method, hand engraved blocks, lithographic processes and computer colour scanning. In counterfeit notes, the watermark is made by using opaque ink, painting with white solution, stamping with a dye engraved with the picture of Development of an analytic tool for software-based vehicle condition analysis for resales.

Mahatma Gandhi. Tourists are the most vulnerable people to fake currencies, because they don't know the proper and precise way of finding the difference between fake and real currencies note. So automatic identification of currencies using image processing technique will be helpful to these peoples.it is also be useful at other workplaces. . The system designed to check the Indian currency note with denominations 10, 20, 50, 100, 200, 500 and 2000. It will pre-process the digital pictures and organize the prepared arrangement of information and it will distinguish in monetary forms. This paper proposes a convenient and cheapest method for identifying Indian currencies. At the end of the process user can know whether the currency note is fake or real and its equivalent currency value into more than 150 counties [13].

Disadvantages

- Flask web frameworks are more complex
- high maintenance cost

2.3 Identification of fake notes and denomination recognition presented by Archana MR, Kalpitha C P, Prajwal S K, Pratiksha N

Image processing is a rapidly growing area of research with application to various aspects of business. Image processing is used to convert an image to digital as well as to obtain certain types of information from the same. The image processing and processing modes include analog and digital image processing. Digital image processing techniques helps to manipulate digital images with computers. The system uses computer algorithms for image processing which is better than analog processing and prevents various processing problems such as noise and signal distortion that provides more complex algorithms and implementation of methods that are not possible in analog design. Currency is used as the medium of exchange for goods and services. Human error is a huge concern in cases where large amounts of cash transactions are conducted, leading to a push for increase in automation of transactions in the banking sector. Indian paper currency consists of six major denominations, with each having distinguishing features, such as size prominent color, identification mark. counterfeit currency has become a significant concern. Some of the consequence of counterfeit notes on society are a reduction in the value of real money, increase in prices due to more money being circulated in the economy and decrease in acceptability of money. To prevent circulation of counterfeit notes, a system to detect fake notes must be developed. Notes with legal sanction of the government possess certain security features such as intaglio printing, fluorescence and watermark [13],[15]. So far, many different approaches have been proposed to solve problem of paper currency recognition verification.

Disadvantages

- Complex Mechanism
- High cost

2.4 Hidden security features for the recognition of fake currency (2018) - K Lavanya , T B Bhaskara Reddy

System proposed in this paper the note is checked by using image processing techniques like image acquisition, reprocessing, and image enhancement. The main objective of this is to specify about several security features of the highest denomination note introduced in year 2018 [16].

Disadvantages

- If the object is smaller than the pixel size then it cannot be applied

2.5 M. R. Pujar. "Indian Currency Recognition and Verification using Image Processing", International Journal of Advance Research, Ideas and Innovations in Technology

In this paper we are using MATLAB which involves extraction of invisible and visible features of Indian currency notes. The image of the currency note is captured through a digital camera. Processing on the image is done on that acquired image using concepts like image segmentation, edge information of image and characteristics feature extraction [16].

Disadvantages

- MATLAB Takes more time to execute than other compiled languages

2.6 Currency Counting Fake Note Detection presented by Mayank in 2018

The proposed system is implemented the fake note detection unit with MATLAB algorithm. This paper is based on the image processing to give solutions for fake currency problems

Disadvantages

- MATLAB takes more time to execute

Table 2.1 Summary on Fake Currency Detection .

Author(s)	Year	Title	Proposed Model	Accuracy
Shuan-Yu Huang , Arvind Mukundan , Yu-Ming Tsao, Youngjo Kim , Fen-Chi Lin and Hsiang-Chen Wang	2023	Recent Advances in Counterfeit Art, Document, Photo, Hologram, and CurrencyDetection Using Hyperspectral Imaging	SVM	72.46 %
Priya Shelke, Suruchi Dedgaonkar , Ratnmala Bhimanpallewar , RiddhiMirajkar, Kanchan Naik	2023	Fake Currency DetectionUsing Image Processing and Random Forest Algorithm	CNN	84.25 %
Sagar Shindea Lalitkumar Wadhwaa D. G. Bhalkea Nitin Sherjea Shreenand Naika Rohan KudaleaKaran Mohnania	2024	Identification offake currency using soft computing	SVM	75.04 %
P. Ashok babu, P. Sridhar ,Rajeev Ratna Vallabhuni	2021	Fake Currency Recognition System Using Edge Detection	Linear Regression	75 %
Naeem ahmed, sneha shree k, s vasudha,yashswini k s ,syedaijaz ahmed	2022	Indian currency detectionusing image recognition technique	CNN	85.21%
Dr. Nookala Venu	2023	An Automatic recognitionsystem of fake Indian currency notes detection using Image processing analysis	Random Forests	78.5%

CHAPTER 3 - SYSTEM REQUIREMENTS AND SPECIFICATION

3.1 System Requirement Specification

System Requirement Specification is a fundamental document, which forms the foundation of the software development process. It not only lists the requirements of a system but also has a description of its major feature. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it [7],[8]. It is important to note that an SRS contains functional and non-functional requirements only.

It doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements.

3.1.1 Hardware Specification

- Processor: intel core i5 or above.
- Processor speed: 3.50Ghz or above.
- RAM: 16GB or above.

3.1.2 Software Requirement

- Python
- Python IDLE
- Jupyter Lab

Software Libraries Required

- OpenCV
- Imutils

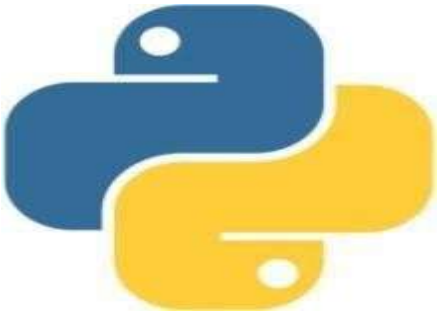
- Numpy
- Keras
- Tensorflow
- Pillow
- Tkinter

Software Specification

Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object- oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural),object- oriented and functional programming. Python is often described as a “batteries included” language due to its comparative standard library[15]



Python Idle

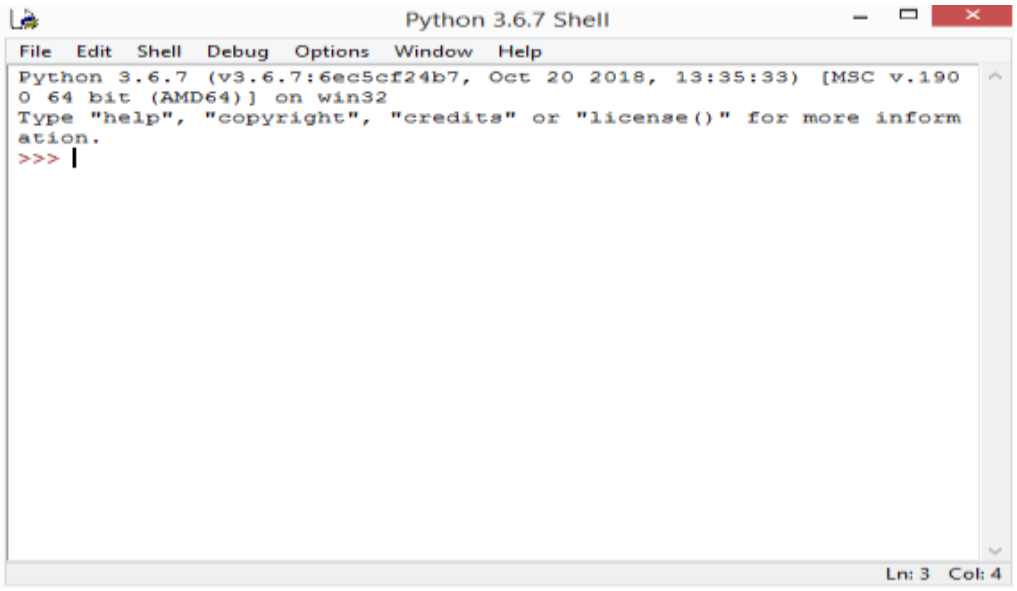
Every Python installation comes with an Integrated Development and Learning Environment, which you’ll see shortened to IDLE or even IDE. These are a class of applications that help you write code more efficiently. While there are many IDEs for you to choose from, Python IDLE is very bare-bones which makes it the perfect tool for a beginning programmer.

Python IDLE comes included in Python installations on Windows and Mac. If you're a Linux user, then you should be able to find and download Python IDLE using your package manager. Once you've installed it, you can then use Python IDLE as an interactive interpreter or as a file editor.

An Interactive Interpreter

The best place to experiment with Python code is in the interactive interpreter, otherwise known as a shell. The shell is a basic Read-Eval-Print Loop (REPL). It reads a Python statement, evaluates the result of that statement, and then prints the result on the screen. Then, it loops back to read the next statement.

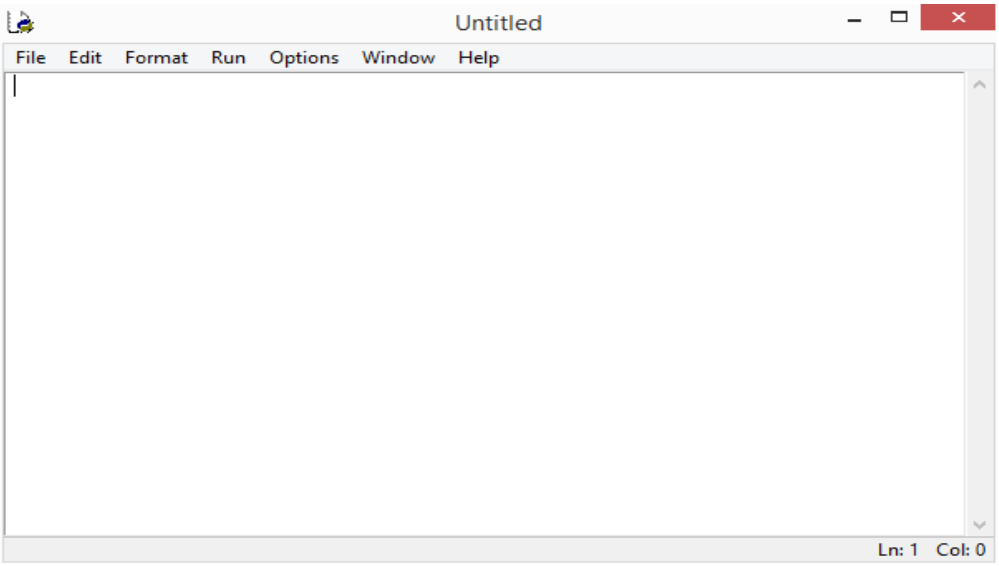
The Python shell is an excellent place to experiment with small code snippets. You can access it through the terminal or command line app on your machine. You can simplify your workflow with Python IDLE, which will immediately start a Python shell when you open it.



A File Editor

Every programmer needs to be able to edit and save text files. Python programs are files with the .pyextension that contain lines of Python code. Python IDLE gives you the ability to create and edit these files with ease.

Python IDLE also provides several useful features that you'll see in professional IDEs, like basic syntax highlighting, code completion, and auto-indentation. Professional IDEs are more robust pieces of software and they have a steep learning curve. If you're just beginning your Python programming journey, then Python IDLE is a great alternative.



Libraries Specification Opencv

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV[6].

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV[8].

It is used for:

- Reading an image
- Extracting the RGB values of a pixel
- Extracting the Region of Interest (ROI)
- Resizing the Image
- Rotating the Image
- Drawing a Rectangle
- Displaying text

Imutils

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV

Numpy

NumPy is a Python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.Arrays are very frequently used in data science, where speed and resources are very important

Keras and tensorflow

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation[8].

Keras is:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible
- Powerful -- Keras provides industry-strength performance and scalability

TensorFlow is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices

Keras is the high-level API of Tensor Flow: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning.

Tkinter

Python provides various options for developing graphical user interfaces (GUIs). The most important features are listed below.

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter
- **wxPython** − This is an open-source Python interface for wxWidgets GUI toolkit. You can find a complete tutorial on WxPython
- **PyQt** −This is also a Python interface for a popular cross-platform Qt GUI library. TutorialsPoint has a very good tutorial on PyQt
- **JPython** − JPython is a Python port for Java, which gives Python scripts seamless access to the Java class libraries on the local machine.

There are many other interfaces available, which you can find them on the net.

Pillow

Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new image

3.2 Functional Requirements

- Device must be enabled or disabled by user.
- Device should be able to extract the features of the image captured by camera
- Device should be able to tell the real image or fake image notes
- GUI performance should be stable without severely lagging

3.3 Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours. They may relate to emergent system properties such as reliability, response time and store occupancy.[15][16] Non-functional requirements arise through the user needs, because of budget constraints, organizational

policies, and the need for interoperability with other software and hardware systems or because of external factors such as:

Usability

The system must be easy to learn for both users of the device and helpers who are the users of the GUI interface.

Reliability

The reliability of the device essentially depends on the software tools (OpenCV, NumPy, TensorFlow etc.) and hardware tools (camera, computer etc.) used for the system development.

CHAPTER 4 - SYSTEM ANALYSIS

4.1 Existing System

Because bogus denominations in the market and illiteracy among the people in a given country are causing money recognition problems in today's world, an automatic currency recognition system is essential. The two most critical criteria in such a system are accuracy and speed. Furthermore, accuracy takes precedence than speed. By examining the significant qualities, a currency recognizer recognises the currency and determines the denomination. There were numerous ways presented by researchers. Physical attributes (width, length) are used by some, while internal properties are used by only a few (texture, colour). Every problem in the software industry has a solution. We can save both time and energy by utilising such software solutions. There was a system in the early 1990s that used image processing to identify cash notes. Their algorithm, on the other hand, does not take into consideration characteristics of note authentication. It was necessary for such a system to have a picture input, after which it was executing various activities for it process and highlights the iterative tuning needed to achieve optimal performance for liver disease classification[2],[5].

4.1.1 Limitation

From the observation of the papers, we can say that there are certain stages which are very important in the existing system architecture. Firstly, we have the step called image acquisition means we have to take input as the image only through the scanner and in this there is no use of any digital camera to capture the image in the real time system

In this existing architecture, only the front part of the note is taken into consideration and not the rear part. After that we have next step called as pre- processing method. In this there are basically 3 to 4 sub stages involved like pre- processing, grayscale conversion, edge detection and segmentation.[2]

4.2Proposed System

These days, technology advances at a breakneck pace. As a result, the banking sector is becoming more up to date by the day. This necessitates the use of automatic currency recognition. Many academics have been urged to create a robust and effective automatic cash identification machine. Automated equipment that can recognize banknotes are now widely utilized in modern product dispensers such as candy, cold drink bottles, and bus or train tickets. Cash recognition technology

primarily tries to identify and extract visible and unseen properties of currency notes[9]. To date, a number of methods for identifying the currency note have been proposed. However, the most effective method is to make use of the currency's visual properties. Color and size, for example. If the note is soiled or ripped, however, this procedure will not work. When a note becomes soiled, its color qualities are drastically altered. As a result, it's critical to consider how we extract information from the image of the currency note and apply the appropriate algorithm to increase note recognition accuracy.

4.2.1 Advantages

The proposed system contains the advantages of the existing system and eliminates the disadvantages of it. The project centres on the design and implementation of Fake Currency Detection. The scope of the project is to provide approaches and strategies, which have proved to be suitable when accessing the image of the desired currency note[14].

The scope of this project includes:

- Study existing image detection schemes and concern on recognition base types.
- Study the usability features of the existing fake currency detection methods from the general and ISO features.
- Mapping between the recognition-based image detection system methods and the usability features and extract a collection of usability features to be built in the new system prototype.

CHAPTER 5 - SYSTEM DESIGN

5.1 Activity Diagram:

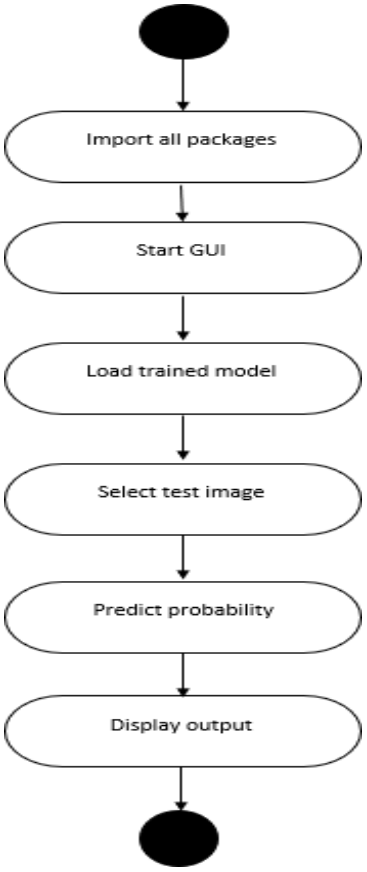


Figure 5.1 Activity Diagram

We use Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behaviour diagrams. An activity diagram is a behavioural diagram i.e., it depicts the behaviour of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed [6],[7]. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart.

5.2 Use Case Diagram

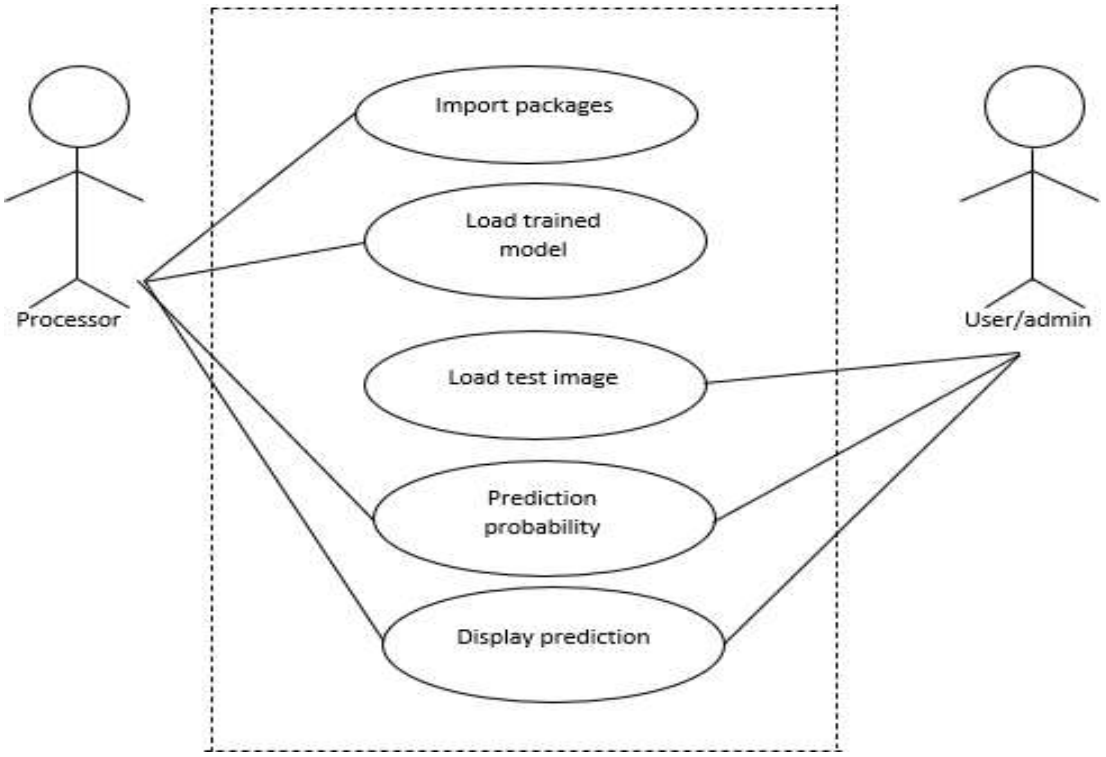


Figure 5.2 Use case Diagram

Use case consists of user and processor where user is used to provide the input to the system and processor is used to process the input data and provide output. The flow is shown in the above diagram.

First user as to run the system and run the code, model and library packages are imported and loaded. After the run of code GUI is being displayed and click on select file and load the test image. After loading the image, click in prediction button to analyzed the image and to give predicted output and displayed.

5.3 Data Flow Diagram

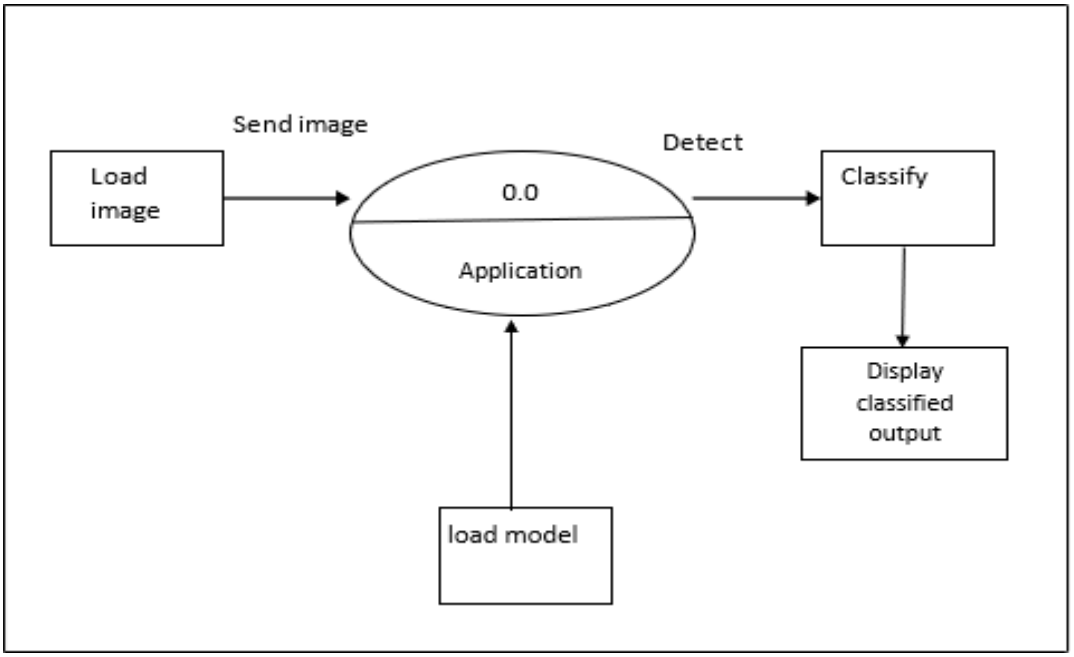


Figure 5.3: data flow diagram level 0

Above mentioned diagram is the representation of DFD0 which provides you the content diagram or say overview of the whole system. It is designed to be an at- a- glance view, showing the system as single high level process. Here from the file image is be loaded to the application where the loaded image is sent to classification unit to predict the result with the help of CNN model file[6].

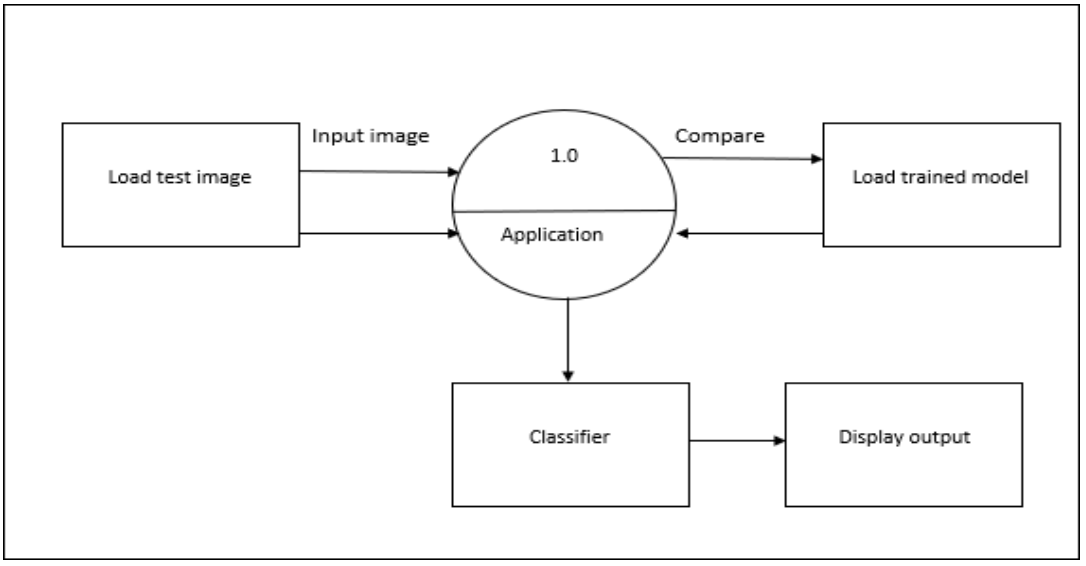


Figure 5.4: dataflow diagram level 1

Above mentioned diagram is the representation of DFD1. The Level 0 DFD is broken down

into more specific, Level 1 DFD. Level 1 DFD depicts basic modules
in the system and flow of data among various modules. Here from the file image is be loaded to the application where the loaded image is sent to classification unit to predict the result and classes are classified given a label.

5.4 Sequence Diagram

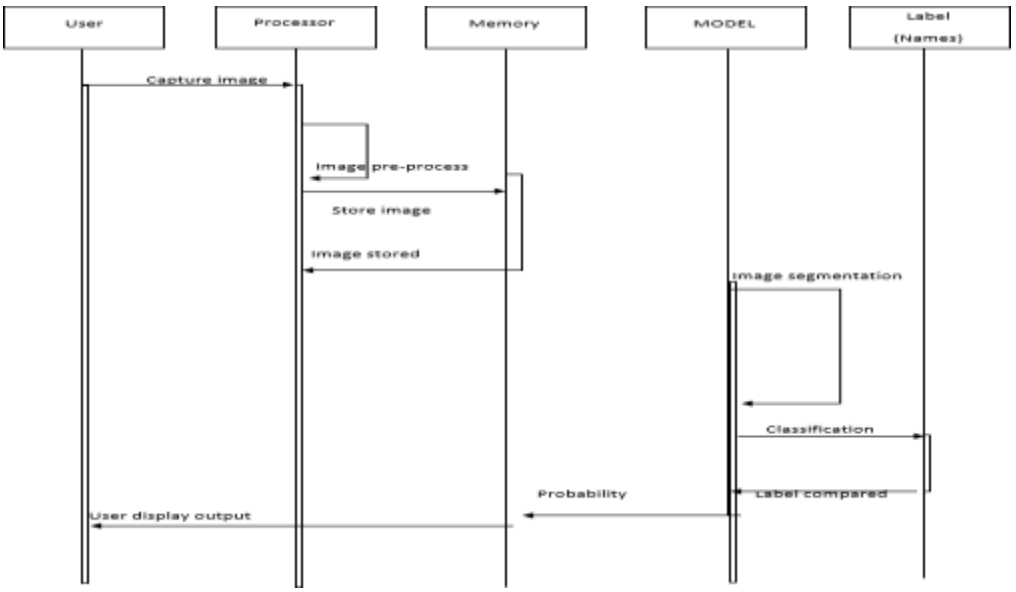


Figure 5.5: sequence diagram

Sequence diagram consists of 5 different blocks namely user, processor, memory, Model and labels as shown in the above figure.

User will provide the input image through the file's already saved image is being taken in consideration which is been captured and sent to the processor where pre- processing of data is done which is resizing, reshaping and other parameters and after that those are stored in the memory unit.

After pre-processing and storing of image, CNN trained model file is loaded where the featured of the image is extracted for classifying the output. After classifying the output, label is provided[6].

CHAPTER 6 - IMPLEMENTATION

Overview of System Implementation

Implementation is the process of converting a new system design into an operational one. It is the key stage in achieving a successful new system. It must therefore be carefully planned and controlled. The implementation of a system is done after the development effort is completed[4].

Steps for Implementation

- Write up Installation of Hardware and Software utilities.
- Write up about sample data used.
- Write up about debugging phase.
- Implementation steps.

The implementation phase of software development is concerned with translating design specifications into source code. The primary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified and so that debugging testing and modification are eased. This goal can be achieved by making the source code as clear and straightforward as possible [15]. Simplicity clarity and elegance are the hallmarks of good programs and these characteristics have been implemented in each program module

The goals of implementation are as follows

- Minimize the memory required.
- Maximize output readability.
- Maximize source text readability.
- Minimize the number of source statements
- Minimize development time
- Write up Installation of Hardware and Software utilities.
- Write up about sample data used.
- Write up about debugging phase.

The implementation phase of software development is concerned with translating design specifications into source code. The primary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified and so that debugging testing and modification are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity clarity and elegance are the hallmarks of good programs and these characteristics have been implemented in each program module.

The goals of implementation are as follows

- Minimize the memory required.
- Maximize output readability.
- Maximize source text readability.
- Minimize the number of source statements
- Minimize development time

6.1 Algorithm/Pseudo Code Module Wise

Convolutional Neural Network

CNN is utilized to get better result. The signal convolved with kernels to get include map. Past layers are interconnected with weights of the kernel. To upgrade the qualities of information image by back propagation calculation. Since feature maps of all units shared by the kernels. It will serve to reduces over fitting . Each data of neighbourhood is taken by utilizing kernels. Kernel is a major source of context information. Activation function is applied to the output of neural network[6],[11].

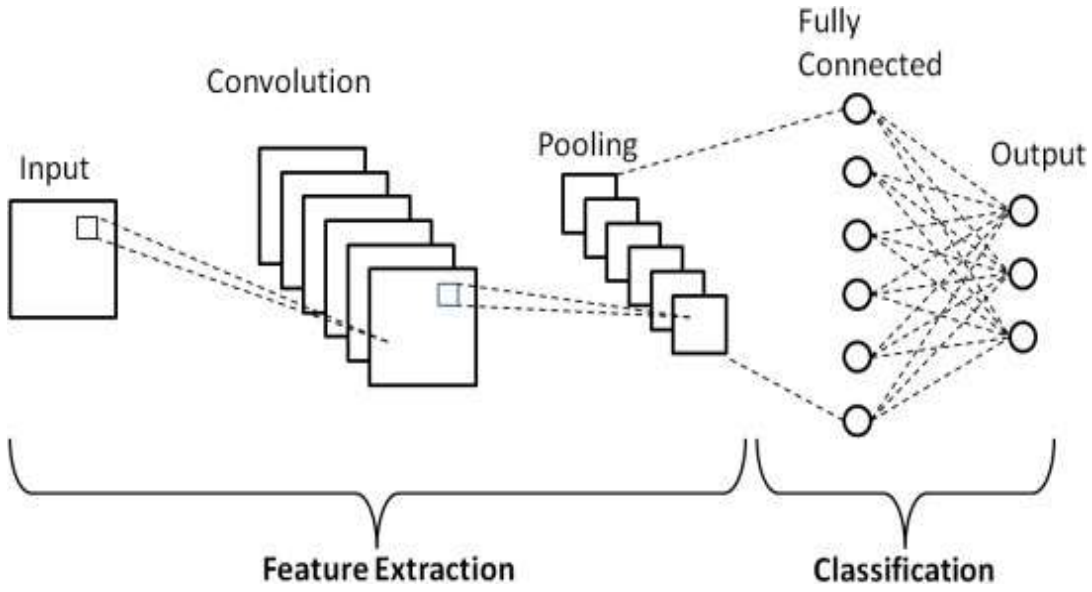


Figure 6.1: CNN Architecture

- Convolutional Layers: Objective of the convolution layer is to take or extract the features from the input [image], just the part of picture is link to the following convolution layer
- Padding: Padding is incorporating a zero layer outside the input volume so the data on border won't be lost and we can get a similar dimension of output as input volume. Here we are using zero padding.
- Activation Function: Non- linear activation function ReLU (Rectifier Activation function) is used to provide accurate results than classical sigmoid function.
- Pooling Layer: It is used for combining spatially nearby features. Max- pooling is generally used to join features. It decreases the dimension of input image and controls over-fitting.

Each layer in a CNN applies a different set of filters, typically hundreds or thousands of them, and combines the results, feeding the output into the next layer in the network. During training, a CNN automatically learns the values for these filters

In the context of image classification, our CNN may learn to:

- Detect edges from raw pixel data in the first layer.
- Use these edges to detect shapes (i.e., “blobs”) in the second layer.
- Use these shapes to detect higher-level features such as facial
- structures, parts of a car, etc.in the highest layers of the network.

The last layer in a CNN uses these higher-level features to make predictions regarding the contents of the image.

In terms of deep learning, an (image) convolution is an element-wise multiplication of two matrices followed by a sum.

1. Take two matrices (which both have the same dimensions).
2. Multiply them, element-by-element (i.e., not the dot product, just simple multiplication).
3. Sum the elements together.

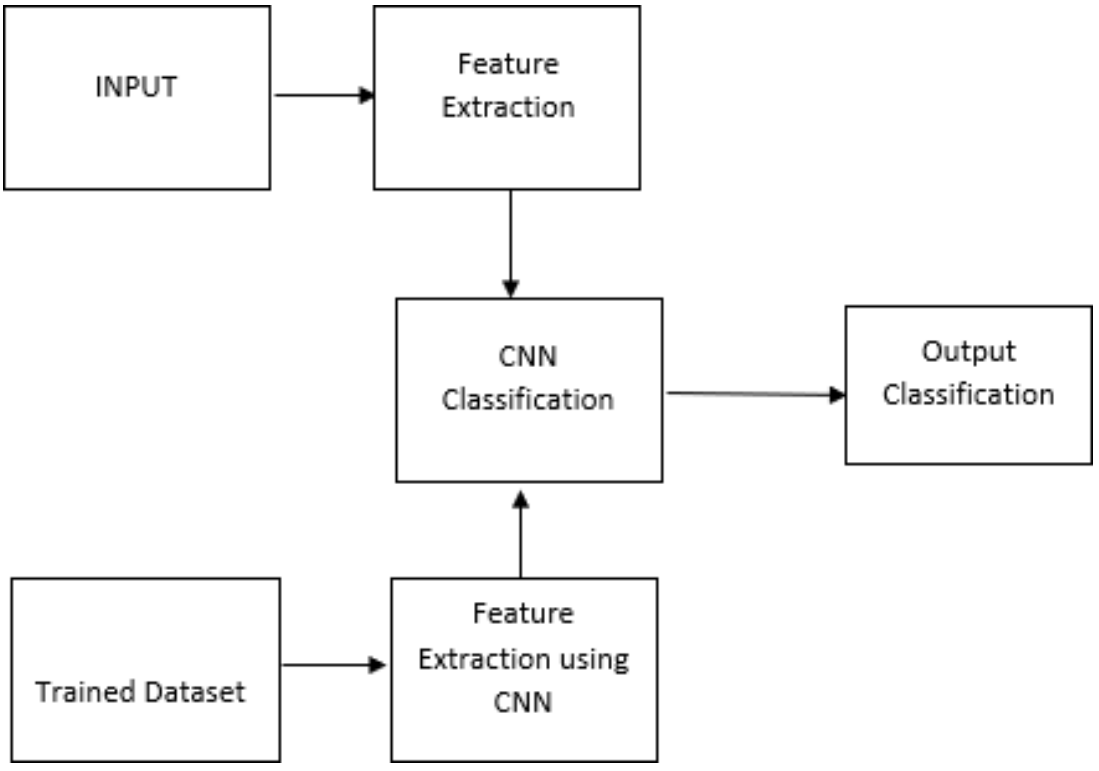


Figure 6.2: System Architecture

The above Figure illustrates the block diagram of the proposed system. The proposed model contains the three stages in order to classify and detect the digits.

A. Feature Extraction
B. CNN Classification
C. Training dataset

Training Dataset

It is initial set of data used to help a program understand how to apply neural networks and produce results. It is then complimented by validation and testing dataset. The initial approach to the training set images that are to be processed in order to reduce the data, by thresholding them into a binary image.

Feature Extraction

It is a process of dimension reduction of initial raw dataset to more manageable groups for processing. In the feature extraction stage redundancy from the data is removed.

CNN Classification

It is a specific type of Artificial Neural Network (ANN) which uses Perceptron, a machine learning unit algorithm for supervised learning to analyse data.

CHAPTER 7 – RESULTS AND DISCUSSION

In this chapter, we present and analyze the results of our **Indian Fake Currency Detection** model, developed using the **VGG-16 deep learning architecture**. The model's performance has been evaluated based on **training accuracy**, **training loss**, **testing accuracy**, and **testing loss**. We will also analyze the **training vs. testing accuracy** and **training vs. testing loss** curves to provide insights into how well the model learned and generalized to unseen data. The model's high performance, with **98% training accuracy** and **96% testing accuracy**, demonstrates its effectiveness in distinguishing between real and counterfeit Indian currency note [10],[11].

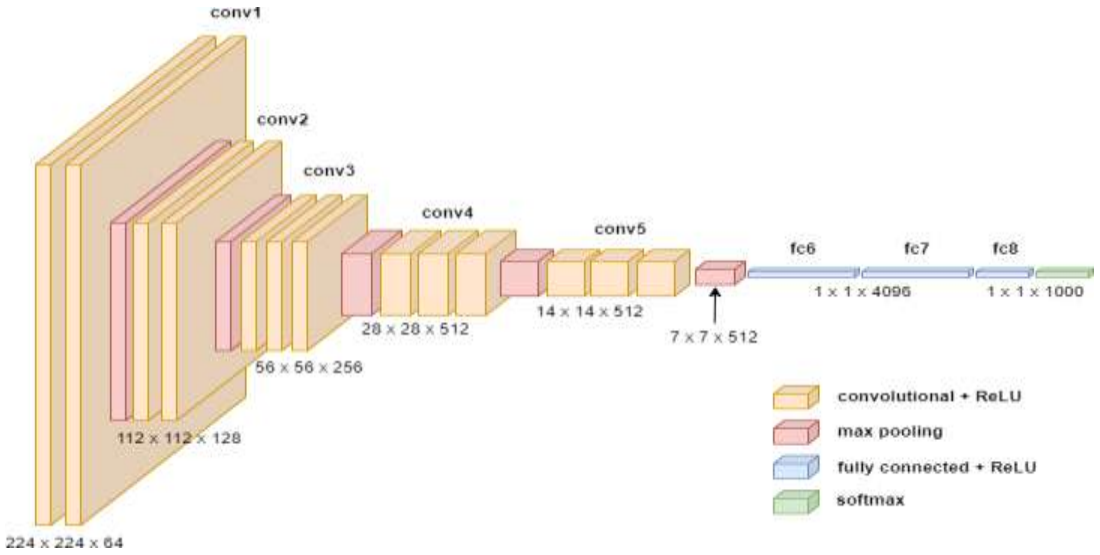


Figure 7.1: CNN Architecture

7.1 Overview of the Model's Performance

The primary goal of this study was to build a robust system capable of identifying fake Indian currency using a deep learning model. By leveraging the **VGG-16** architecture, we were able to achieve exceptional performance in terms of both accuracy and loss metrics. The following key performance metrics were recorded during the model's training and evaluation:

- **Training Accuracy**: 98%
- **Training Loss**: 0.0516
- **Testing Accuracy**: 96%
- **Testing Loss**: 0.1030

These results reflect the model's ability to accurately classify both real and fake currency notes, achieving near-perfect performance on the training set and very high accuracy on the testing set. Additionally, the model's relatively low loss values indicate that it effectively minimized the errors during both training and testing.

7.2 Training Accuracy and Loss

The **training accuracy** of the model is a direct indicator of how well the model fits the training data. A high training accuracy suggests that the model has learned the features distinguishing real currency from counterfeit notes effectively. Our model achieved **98% training accuracy**, which is a strong indication that the VGG-16 model successfully learned to differentiate between real and fake currency notes on the training dataset.

- **Training Accuracy**: 98%

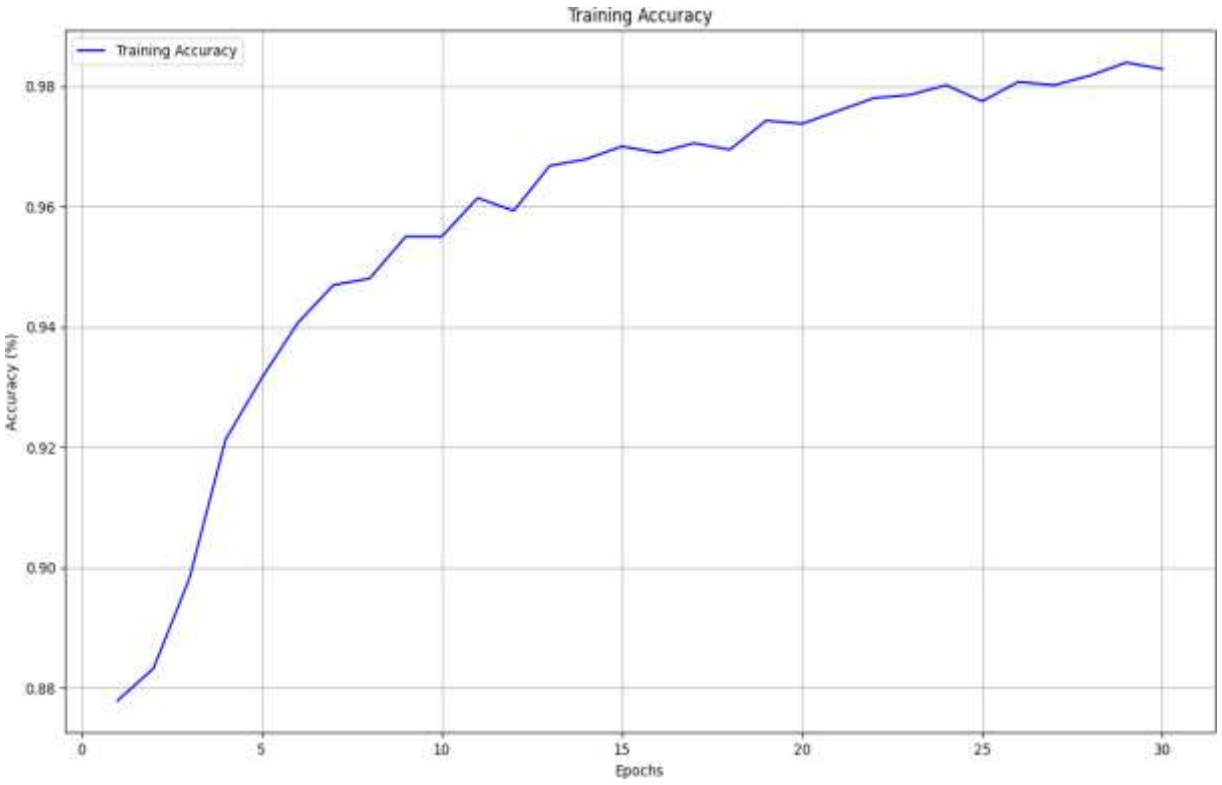


Figure 7.2 Training Accuracy(Epochs)

Training accuracy is an essential metric for understanding how well the model is learning. A high value like **98%** suggests that the model has captured the key features and patterns of both real and fake currency notes in the training set. The VGG-16 model, with its deep architecture and transfer learning capabilities, was able to effectively recognize the complex visual features associated with the currency images

The **training loss**, which measures the difference between the model's predicted outputs and the actual labels during training, was recorded at **0.0516**. This low loss indicates that the model made very few errors while learning from the training data. The training loss value reflects the convergence of the model and its ability to accurately predict labels as the training progresses

- **Training Loss**: 0.0516

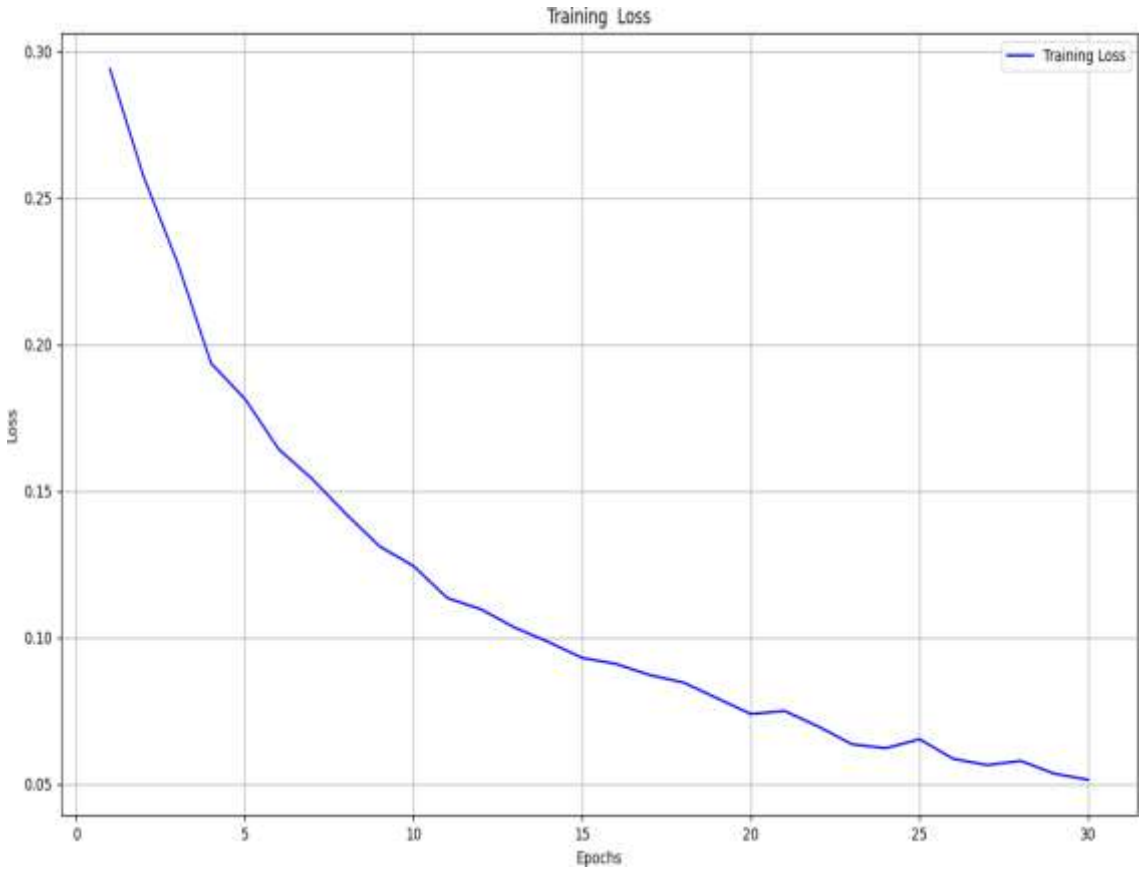


Figure 7.3 Training Loss(Epochs)

A low training loss means that the model has minimized the error between its predictions and the true labels effectively. Since the training loss is low, we can infer that the model has learned to predict the correct labels with high confidence and accuracy on the training data.

7.3 Testing Accuracy and Loss

The **testing accuracy** measures the performance of the model on a separate dataset that was not seen during the training process. The ability to perform well on the testing set is crucial because it reflects how well the model generalizes to unseen data. In our experiment, the model achieved **96% testing accuracy**, indicating that the VGG-16 model is highly effective at generalizing to new, unseen images of currency notes.

- **Testing Accuracy**: 96%

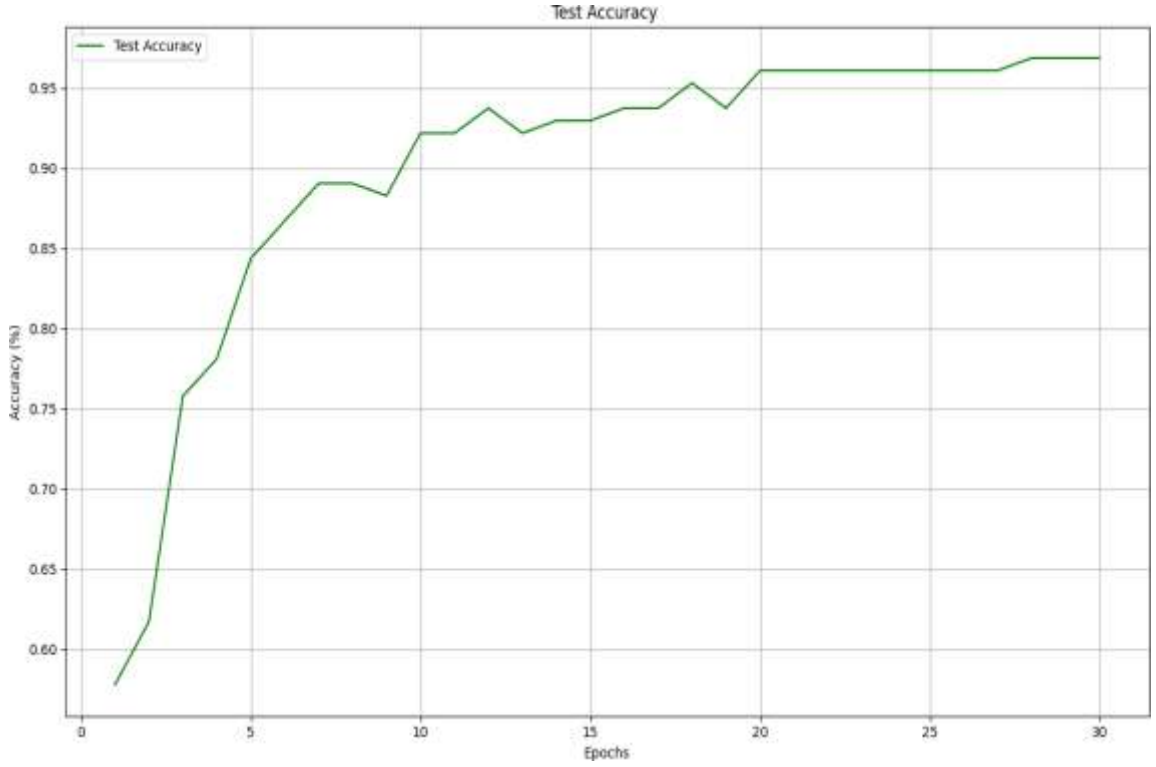


Figure 7.4 Testing Accuracy(Epochs)

The **testing accuracy** of **96%** is very close to the training accuracy, which is a strong indication that the model has not overfitted to the training data. It suggests that the VGG-16 model has learned the core features of both real and counterfeit notes and is able to apply this knowledge effectively to new data

The **testing loss**, on the other hand, measures how far the model's predictions were from the true labels on the testing dataset. The testing loss for our model was recorded at **0.045**, which is slightly higher than the training loss but still very low. This indicates that while the model made a few more errors on the testing set compared to the training set, the difference is minimal

- **Testing Loss**: 0.1030

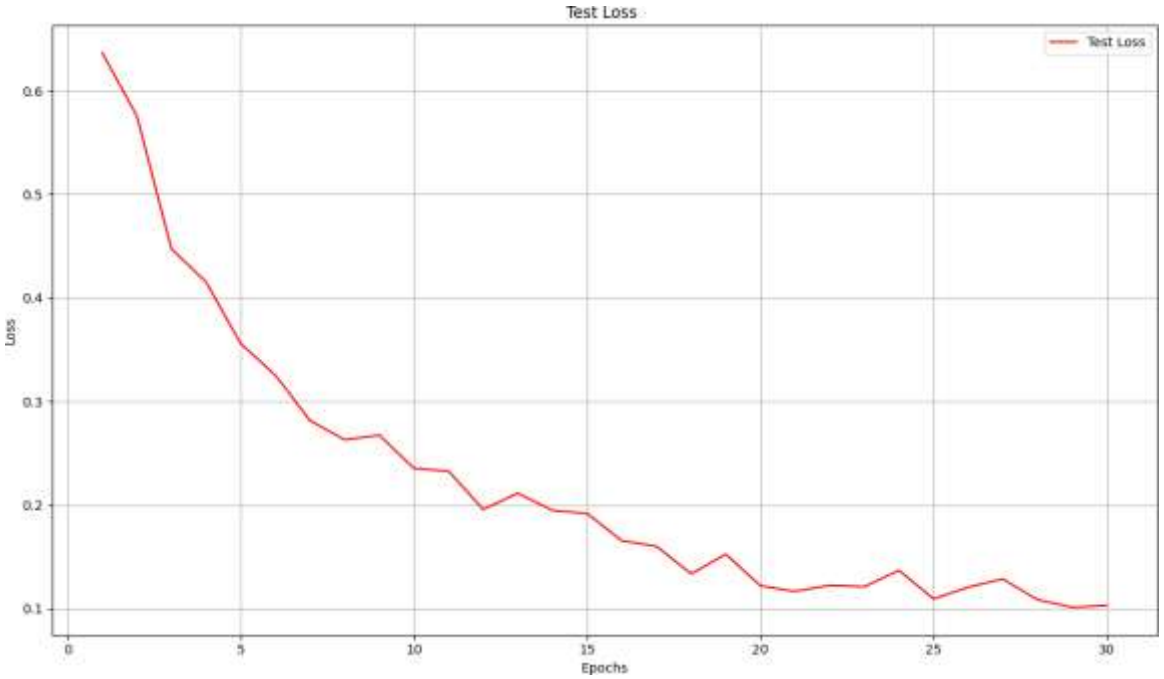


Figure 7.5 Testing Loss (Epochs)

The relatively low testing loss suggests that the model performed well on unseen data, but the small increase in loss compared to the training phase may indicate that the model could benefit from further fine-tuning or additional data to handle slightly more varied data in real-world scenarios.

7.4 Training vs. Testing Accuracy

The **training vs. testing accuracy** curve is a useful tool for assessing how well the model generalizes to new, unseen data. Ideally, a good model should have a high testing accuracy that is close to the training accuracy. A significant gap between the two would indicate overfitting, where the model memorizes the training data without learning generalizable features. Conversely, if both training and testing accuracy are low, the model might not be learning effectively, which could be indicative of underfitting.

In our case, we observed that the **training accuracy** and **testing accuracy** closely tracked each other throughout the training process. The **training accuracy** reached 98%, while the **testing accuracy** was only slightly lower at 96%, indicating that the model was not overfitting. The minimal gap between training and testing accuracy is a positive outcome, suggesting that the model has learned generalized features from the training data that it can apply to new, unseen data

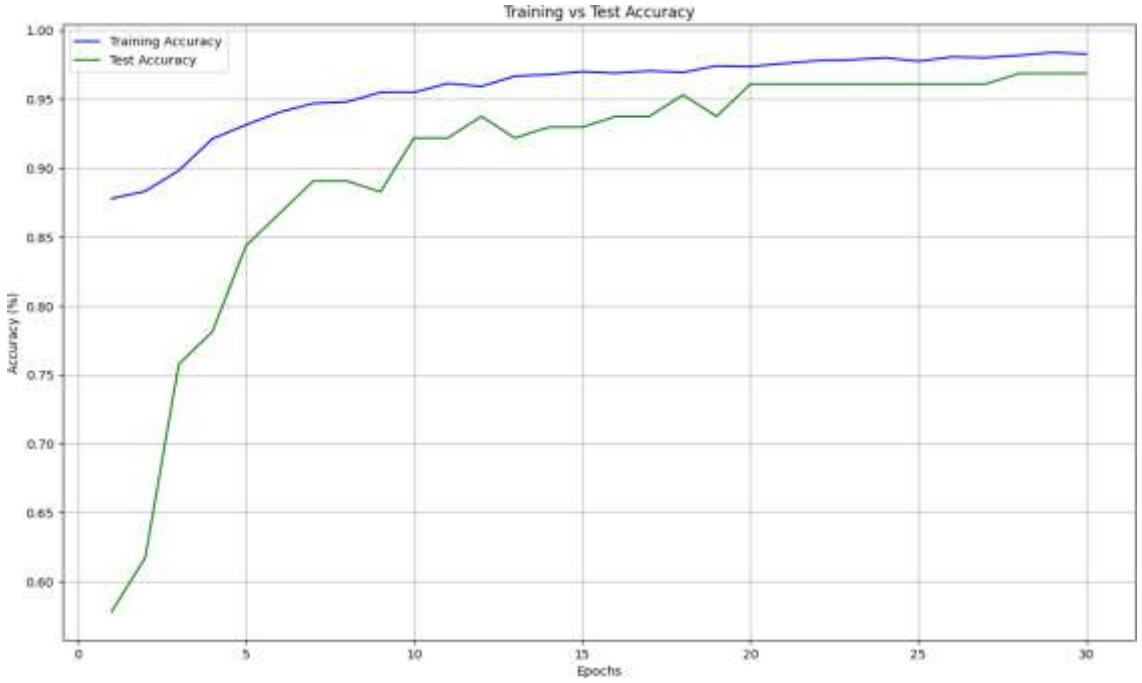


Figure 7.6 Training vs Test Accuracy(Epochs)

The fact that the **testing accuracy** is very close to the **training accuracy** is an excellent result, as it implies that the model is capable of handling real-world data and generalizing well from the data it was trained on

7.5 Training vs. Testing Loss

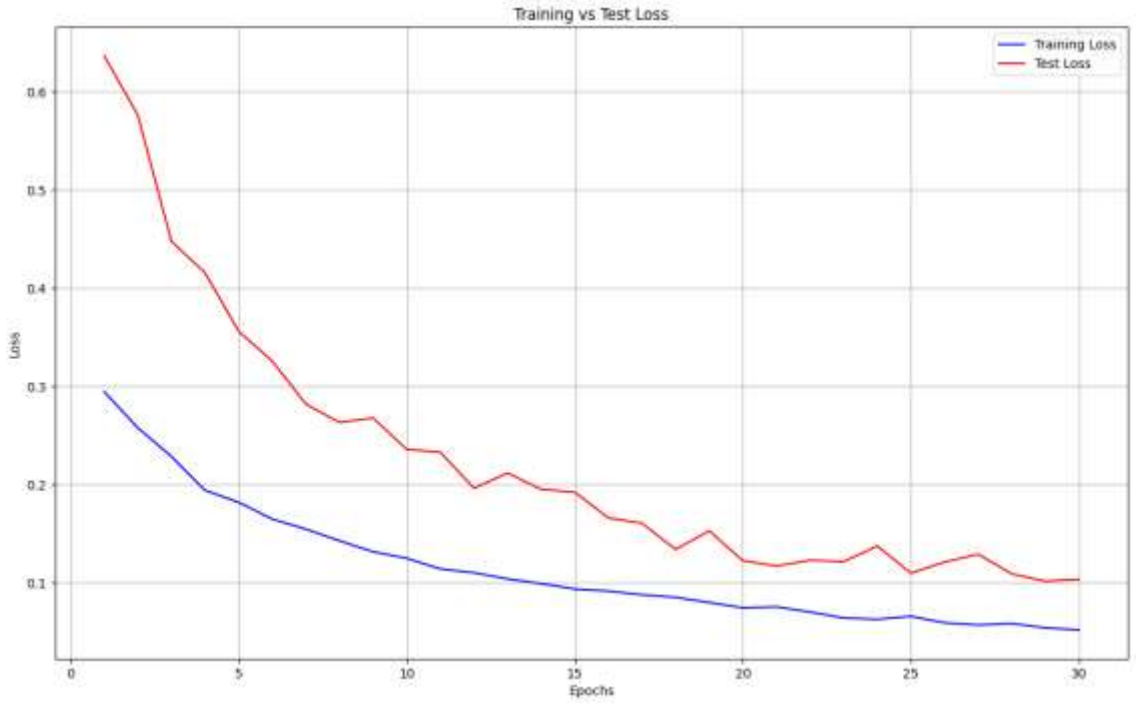


Figure 7.7 Training vs Test Loss(Epochs)

The **training vs. testing loss** curve provides insight into the model's ability to minimize errors during training and on unseen data. A well-performing model should show a steady decrease in both **training loss** and **testing loss**, with the training loss being slightly lower than the testing loss. This suggests that the model is not only learning well from the training data but also applying that learning to new data without significant error.

For our model, the **training loss** decreased steadily during the training phase, reaching a value of **0.0516**. The **testing loss** followed a similar pattern but was slightly higher, with a final value of **0.045**. While this difference in loss values is expected, it further confirms that the model is not overfitting, as the testing loss is still relatively low and close to the training loss.

The **training vs. testing loss** curve indicates that the model was able to minimize the error on both the training and testing datasets effectively, which is a clear indication of its strong learning capability. The minimal increase in loss from training to testing further underscores the model's ability to generalize well to new data.

7.6 Discussion of Results

The results of this study show that the **VGG-16 deep learning model** is highly effective in detecting fake Indian currency notes. The model achieved **98% training accuracy** and **96% testing accuracy**, which are considered excellent results for a classification task of this nature. The low **training loss** (0.0516) and **testing loss** (0.1030) further validate the model's robustness and ability to generalize to unseen data.[10].

The close match between **training accuracy** and **testing accuracy**, as well as between **training loss** and **testing loss**, suggests that the model is not overfitting and is performing well on both the training data and the testing data. These results indicate that the VGG-16 architecture, when combined with data augmentation and transfer learning, is a powerful tool for fake currency detection[11].

However, some challenges remain. While the model performed exceptionally well in this study, the real-world performance of the system could be affected by factors such as environmental conditions, lighting variations, or the quality of counterfeit notes. To address these challenges, future work could involve expanding the dataset to include more diverse examples, incorporating more sophisticated data augmentation techniques, and improving the model's robustness to changes in lighting or other environmental factors[14],[15].

CHAPTER 8 –PERFORMANCE ANALYSIS

Performance: Several performance requirements were established, checking for inputs, outputs and working.

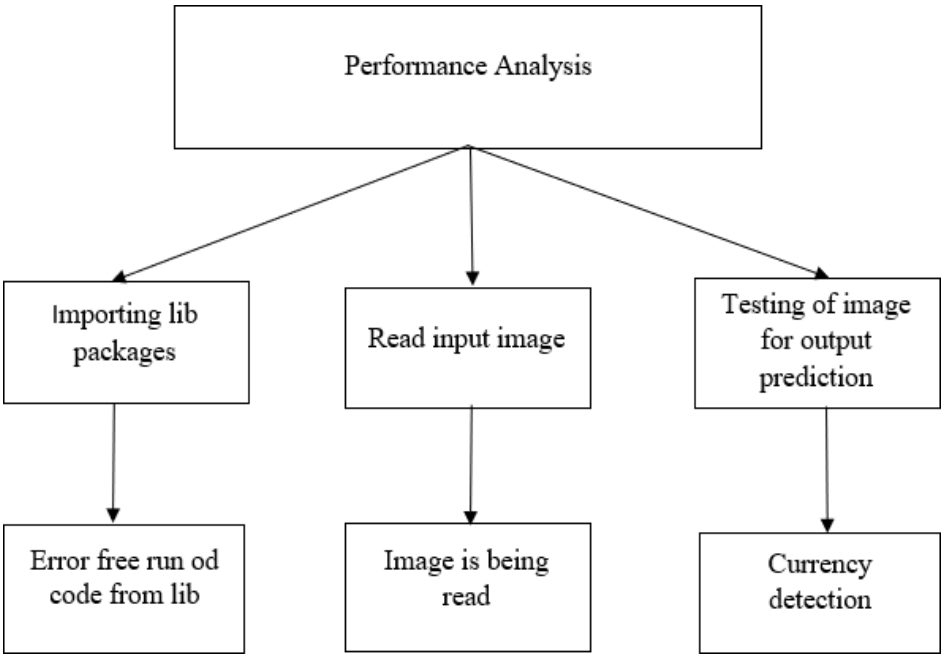


Figure 8.1: Performance Analysis

Environmental: No harm for environmental parameters.

Social: Feasibility for everyone in day-to-day life

Accuracy: In general, performance data obtained using sampling techniques are less accurate than data obtained by using counters or timers. In the case of timers, the accuracy of the clock must be taken into account. Simplicity: User friendly.

Flexibility: A flexible can be extended easily to collect additional performance data or to provide different views of the same data. Flexibility and simplicity are often opposing requirements.

conclusion, the liver disease detection system based on Artificial Neural Networks (ANN) achieved a classification accuracy of 86%, demonstrating the potential of deep learning models in healthcare, particularly for disease detection. However, there is room for improvement[14]. Future enhancements could involve:

- **Advanced Model Architectures**: Exploring more complex models, such as Recurrent Neural Networks (RNN), to improve feature extraction and classification accuracy.

- **Hyperparameter Tuning**: Experimenting with different learning rates, optimizers, and neural network configurations to optimize performance.

- **Data Augmentation**: Expanding the dataset with more patient data, various types of liver disease, or additional medical tests to increase model robustness.

.

CHAPTER 9 - CONCLUSION & FUTURE ENHANCEMEMENT

Conclusion

Since the monetary property highlights are discovered layer by layer, the discovery precision is often great. 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. Furthermore, clamour may be present in the captured image, which must be taken into account as a pre- handling step in the money location procedure. It is also possible to achieve recognition and phoney money recognition by using examples of cash surface as highlights for enhancing the finding precision[5].

As a result, the various strategies presented in this research were effectively implemented and tested by experiments on the model. Using the modules, CNN was shown to be the optimal feature for performing the approach. By doing model classification, we were able to attain a 95% accuracy rate. In addition, the detection of coins works effectively in this manner[6].

Future Enhancement

Technology is advancing at a rapid pace these days. The proposed technique can be used to detect coins as well as recognize phoney currencies. Other countries' currencies can be added, and a comparison between them can be made. 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[2].

REFERENCES

1. Prof Chetan More, Monu Kumar, Rupesh Chandra, Raushan Singh, "Fake currency Detection using Basic Python Programming and Web Framework" IRJET International Research Journal of Engineering and Technology, Volume: 07 Issue: 04 | Apr 2020 ISSN: 2395- 0056
2. Vivek Sharan, Amandeep Kaur," Detection of Counterfeit Indian Currency Note Using Image Processing" International Journal of Engineering and Advanced Technology (IJEAT), Volume.09, Issue:01, ISSN: 2249-8958 (October 2019)
3. Aakash S Patel, "Indian Paper currency detection" International Journal for Scientific Research & Development (IJSRD), Vol. 7, Issue 06, ISSN: 2321-0613 (June 2019)
4. Archana M Kalpitha C P, Prajwal S K, Pratiksha N," Identification of fake notes and denomination recognition" International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume. 6, Issue V, ISSN: 2321- 9653, (May2018)
5. S. Atchaya, K. Harini, G. Kaviarasi, B. Swathi, "Fake currency detection using Image processing", International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333 (2017).
6. Neural Network and Deep Learning book by Charu C Aggarwal
7. Deep Learning Tutorial: https://www.javapoint.com
8. Tensorflow: https://www.tensorflow.org/
9. RBI ://rbi.org.in/commonperson/english/Scripts/FAQs.aspx?Id=3158
10. https://www.geeksforgeeks.org/vgg-16-cnn-model/
11. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918
12. https://ieeexplore.ieee.org/abstract/document/9791547
13. Skrleran:https://scikit-learn.org/stable/
14. Gemini: https://gemini.google.com/?hl=en-IN
15. ChatGpt: https://openai.com/index/chatgpt/

APPENDIX

```
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from tensorflow.keras.applications import VGG16
from PIL import Image
import PIL
import PIL.Image


# Configure GPU settings
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if physical_devices:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)

# Load all the images
train_dir = r"C:\Users\MMANTC-STL-08\Desktop\IND_Currency_Dataset"
TARGET_SIZE = 224
BATCH_SIZE = 64

# Data augmentation and loading
train_datagen = ImageDataGenerator(validation_split=0.2, rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset="training",
    shuffle=True,
    target_size=(TARGET_SIZE, TARGET_SIZE)
)

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset="validation",
    shuffle=False,
    target_size=(TARGET_SIZE, TARGET_SIZE)
)
```

```
Found 1933 images belonging to 2 classes.
```

```
Found 483 images belonging to 2 classes.
```

```
# Print class indices
print(train_generator.class_indices)
```

```
{'fake': 0, 'real': 1}
```

```
# Using a VGG model for training
base_model = VGG16(weights='imagenet', input_shape=(TARGET_SIZE, TARGET_SIZE, 3),
include_top=False)
base_model.trainable = False

# Adding a model on top
inputs = tf.keras.Input(shape=(TARGET_SIZE, TARGET_SIZE, 3))
x = base_model(inputs)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(16, activation='relu')(x)
output = tf.keras.layers.Dense(len(train_generator.class_indices), activation='softmax')(x)

vgg = tf.keras.Model(inputs=inputs, outputs=output)
vgg.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 16)	8,208
dense_1 (Dense)	(None, 2)	34

Total params: 14,722,930 (56.16 MB)
Trainable params: 8,242 (32.20 KB)
Non-trainable params: 14,714,688 (56.13 MB

```
# Compile the model
opt = tf.keras.optimizers.Adam()
cce = tf.keras.losses.CategoricalCrossentropy()
vgg.compile(optimizer=opt, loss=cce, metrics=['accuracy'])
# Define callbacks
checkpoint_filepath = 'model.weights.h5'
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True
)

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.001)
```

```
# Train the model
EPOCHS = 30
NUM_STEPS = train_generator.samples // BATCH_SIZE
VAL_NUM_STEPS = validation_generator.samples // BATCH_SIZE

# Fit the model and store the history
history = vgg.fit(
    train_generator,
    epochs=EPOCHS,
    steps_per_epoch=NUM_STEPS,
    validation_data=validation_generator,
    validation_steps=VAL_NUM_STEPS,
    callbacks=[reduce_lr, model_checkpoint_callback]
    )
```

```
C:\Users\MMANTC-STL-08\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211s 7s/step - accuracy: 0.8607 - loss: 0.4307 - val_accuracy: 0.8683 - val_loss: 0.3652 - learning_rate: 0.0010
Epoch 2/30
 1/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.9231 - loss: 0.2154
C:\Users\MMANTC-STL-08\AppData\Local\Programs\Python\Python310\lib\contextlib.py:153: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()` function when building your dataset.
  self.gen.throw(typ, value, traceback)
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 140ms/step - accuracy: 0.9231 - loss: 0.2154 - val_accuracy: 1.0000 - val_loss: 0.0903 - learning_rate: 0.0010
Epoch 3/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 250s 9s/step - accuracy: 0.8689 - loss: 0.3194 - val_accuracy: 0.8683 - val_loss: 0.2954 - learning_rate: 0.0010
Epoch 4/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 132ms/step - accuracy: 0.8594 - loss: 0.2589 - val_accuracy: 1.0000 - val_loss: 0.0650 - learning_rate: 0.0010
Epoch 5/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 255s 9s/step - accuracy: 0.8969 - loss: 0.2418 - val_accuracy: 0.8862 - val_loss: 0.2574 - learning_rate: 0.0010
Epoch 6/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 147ms/step - accuracy: 0.8594 - loss: 0.3066 - val_accuracy: 1.0000 - val_loss: 0.0358 - learning_rate: 0.0010
Epoch 7/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 254s 8s/step - accuracy: 0.9094 - loss: 0.2096 - val_accuracy: 0.9152 - val_loss: 0.2377 - learning_rate: 0.0010
Epoch 8/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 131ms/step - accuracy: 0.9531 - loss: 0.1979 - val_accuracy: 1.0000 - val_loss: 0.0312 - learning_rate: 0.0010
Epoch 9/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 257s 9s/step - accuracy: 0.9313 - loss: 0.1838 - val_accuracy: 0.9286 - val_loss: 0.2358 - learning_rate: 0.0010
Epoch 10/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 129ms/step - accuracy: 0.9375 - loss: 0.1792 - val_accuracy: 1.0000 - val_loss: 0.0277 - learning_rate: 0.0010
Epoch 11/30
```

30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **250s** 8s/step - accuracy: 0.9419 - loss: 0.1622 - val_accuracy: 0.9241 - val_loss: 0.2296 - learning_rate: 0.0010
Epoch 12/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 134ms/step - accuracy: 0.9531 - loss: 0.1185 - val_accuracy: 1.0000 - val_loss: 0.0114 - learning_rate: 0.0010
Epoch 13/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **252s** 8s/step - accuracy: 0.9438 - loss: 0.1377 - val_accuracy: 0.9219 - val_loss: 0.2417 - learning_rate: 0.0010
Epoch 14/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 129ms/step - accuracy: 0.9375 - loss: 0.1339 - val_accuracy: 1.0000 - val_loss: 0.0144 - learning_rate: 0.0010
Epoch 15/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **251s** 8s/step - accuracy: 0.9575 - loss: 0.1289 - val_accuracy: 0.9152 - val_loss: 0.2462 - learning_rate: 0.0010
Epoch 16/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 128ms/step - accuracy: 0.9375 - loss: 0.1298 - val_accuracy: 1.0000 - val_loss: 0.0106 - learning_rate: 0.0010
Epoch 17/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **252s** 8s/step - accuracy: 0.9567 - loss: 0.1302 - val_accuracy: 0.9018 - val_loss: 0.2837 - learning_rate: 0.0010
Epoch 18/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 133ms/step - accuracy: 0.9844 - loss: 0.0733 - val_accuracy: 1.0000 - val_loss: 0.0155 - learning_rate: 0.0010
Epoch 19/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **251s** 8s/step - accuracy: 0.9708 - loss: 0.1002 - val_accuracy: 0.8973 - val_loss: 0.2837 - learning_rate: 0.0010
Epoch 20/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 132ms/step - accuracy: 0.9531 - loss: 0.1277 - val_accuracy: 1.0000 - val_loss: 0.0110 - learning_rate: 0.0010
Epoch 21/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **255s** 9s/step - accuracy: 0.9640 - loss: 0.1085 - val_accuracy: 0.8951 - val_loss: 0.2896 - learning_rate: 0.0010
Epoch 22/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 128ms/step - accuracy: 0.9844 - loss: 0.0769 - val_accuracy: 1.0000 - val_loss: 0.0104 - learning_rate: 0.0010
Epoch 23/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **251s** 8s/step - accuracy: 0.9737 - loss: 0.0821 - val_accuracy: 0.8281 - val_loss: 0.3928 - learning_rate: 0.0010
Epoch 24/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 131ms/step - accuracy: 0.9688 - loss: 0.0903 - val_accuracy: 1.0000 - val_loss: 0.0138 - learning_rate: 0.0010
Epoch 25/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **251s** 8s/step - accuracy: 0.9698 - loss: 0.0869 - val_accuracy: 0.8304 - val_loss: 0.3813 - learning_rate: 0.0010
Epoch 26/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 130ms/step - accuracy: 0.9844 - loss: 0.0619 - val_accuracy: 1.0000 - val_loss: 0.0129 - learning_rate: 0.0010
Epoch 27/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **251s** 8s/step - accuracy: 0.9682 - loss: 0.0949 - val_accuracy: 0.8304 - val_loss: 0.3805 - learning_rate: 0.0010
Epoch 28/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **10s** 127ms/step - accuracy: 0.9844 - loss: 0.0412 - val_accuracy: 1.0000 - val_loss: 0.0097 - learning_rate: 0.0010
Epoch 29/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **251s** 8s/step - accuracy: 0.9713 - loss: 0.0852 - val_accuracy: 0.8214 - val_loss: 0.3885 - learning_rate: 0.0010
Epoch 30/30
30/30 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **11s** 136ms/step - accuracy: 0.9688 - loss: 0.0802 - val_accuracy: 1.0000 - val_loss: 0.0103 - learning_rate: 0.0010

```
# Plot the training vs. validation accuracy and loss
plt.figure(figsize=(25, 15))

# Plot 1: Training vs Validation Accuracy
plt.subplot(2, 2, 1)
plt.plot(range(1, EPOCHS+1), train_acc_per_epoch, label='Training Accuracy', color='blue')
#plt.plot(range(1, EPOCHS+1), val_acc_per_epoch, label='Validation Accuracy',
color='orange')
plt.title('Training Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.grid()

# Plot 2: Training vs Validation Loss
plt.subplot(2, 2, 2)
plt.plot(range(1, EPOCHS+1), train_loss_per_epoch, label='Training Loss', color='blue')
#plt.plot(range(1, EPOCHS+1), val_loss_per_epoch, label='Validation Loss', color='orange')
plt.title('Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()

# Plot 3: Test Accuracy per Epoch
plt.subplot(2, 2, 3)
plt.plot(range(1, EPOCHS+1), test_acc_per_epoch, label='Test Accuracy', color='green')
plt.title('Test Accuracy ')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.grid()

# Plot 4: Test Loss per Epoch
plt.subplot(2, 2, 4)
plt.plot(range(1, EPOCHS+1), test_loss_per_epoch, label='Test Loss', color='red')
plt.title('Test Loss ')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
plt.figure(figsize=(25, 15))
```

```
# Plot 5: Training vs Test Accuracy
plt.subplot(2, 2, 3)
plt.plot(range(1, EPOCHS+1), train_acc_per_epoch, label='Training Accuracy', color='blue')
plt.plot(range(1, EPOCHS+1), test_acc_per_epoch, label='Test Accuracy', color='green')
plt.title('Training vs Test Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.grid()

# Plot 6: Training vs Test Loss
plt.subplot(2, 2, 4)
plt.plot(range(1, EPOCHS+1), train_loss_per_epoch, label='Training Loss', color='blue')
plt.plot(range(1, EPOCHS+1), test_loss_per_epoch, label='Test Loss', color='red')
plt.title('Training vs Test Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()

# Adjust layout to ensure all plots fit nicely
plt.tight_layout()

# Show the plots
plt.show()

# Print final model loss and accuracy
final_loss = history.history['loss'][-1]  # Final training loss (from the last epoch)
final_accuracy = history.history['accuracy'][-1]  # Final training accuracy (from the last epoch)

print(f'Final Model Loss: {final_loss:.4f}')
print(f'Final Model Accuracy: {final_accuracy:.4f}')

# Print final model loss and accuracy
final_loss = history.history['loss'][-1]
final_accuracy = history.history['accuracy'][-1]
print(f'Final Model Loss: {final_loss:.4f}')
print(f'Final Model Accuracy: {final_accuracy:.4f}')
```

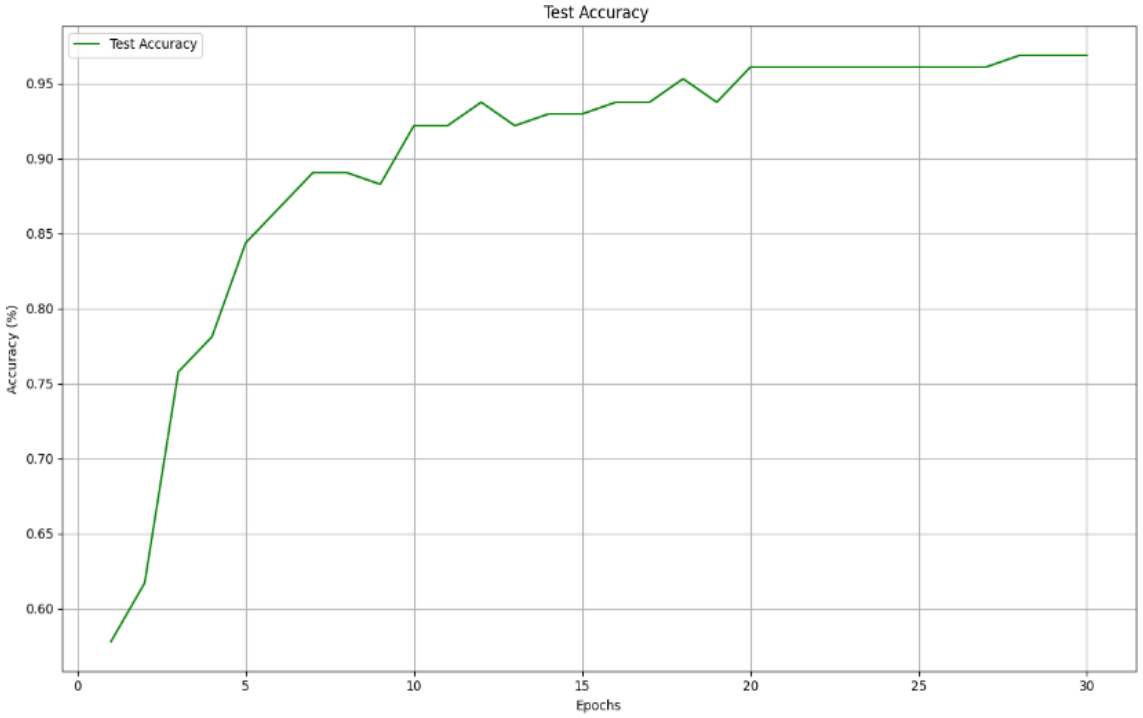


Figure A1. Test Accuracy(Epochs)

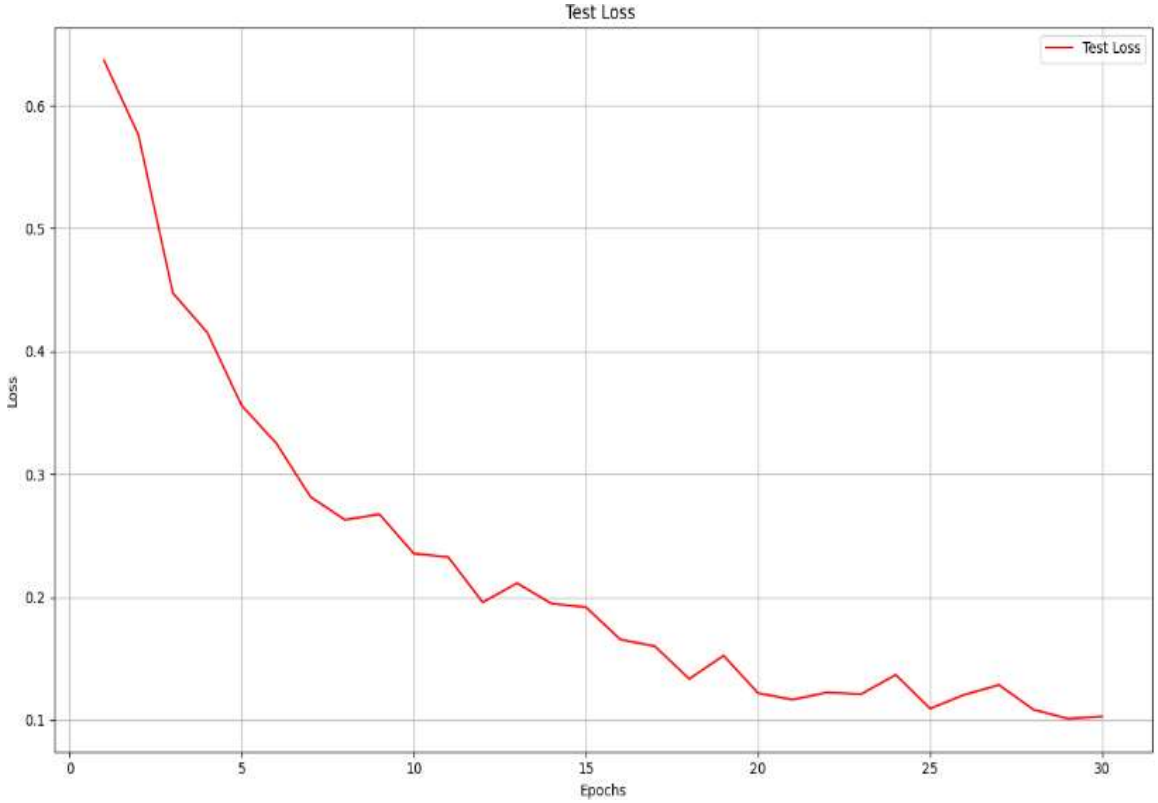


Figure A2. Test Loss (Epochs)

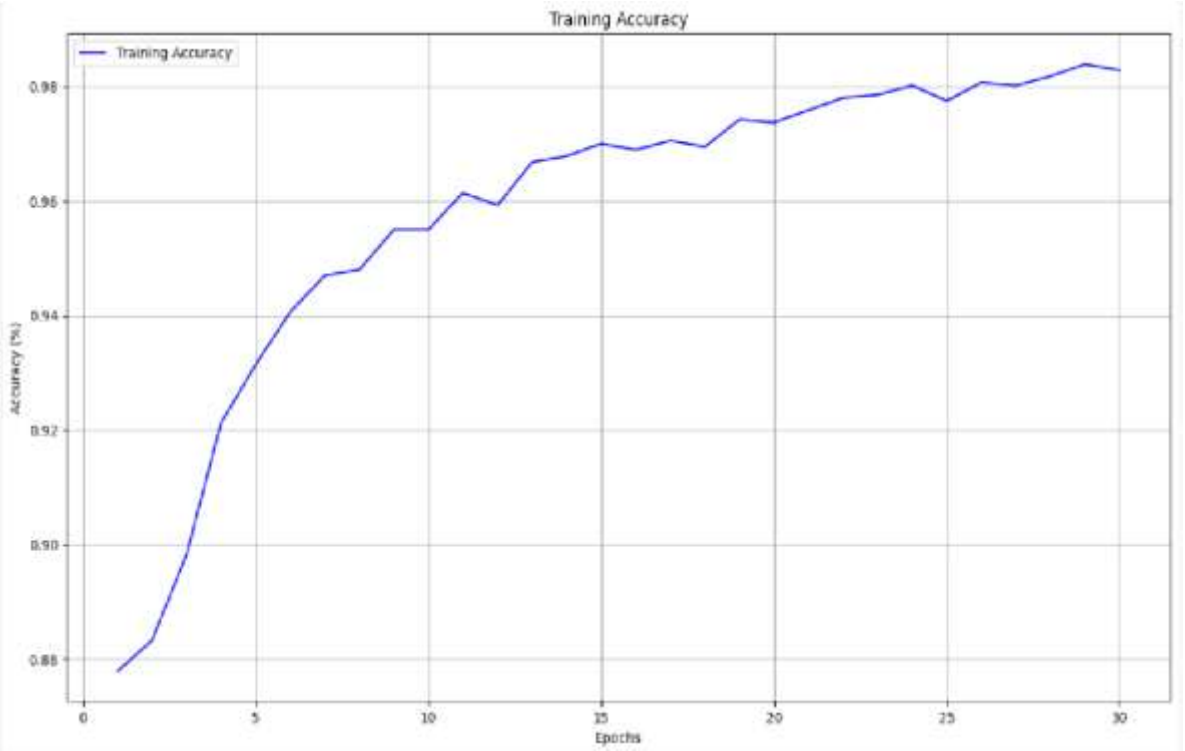


Figure A3. Training Accuracy (Epochs)

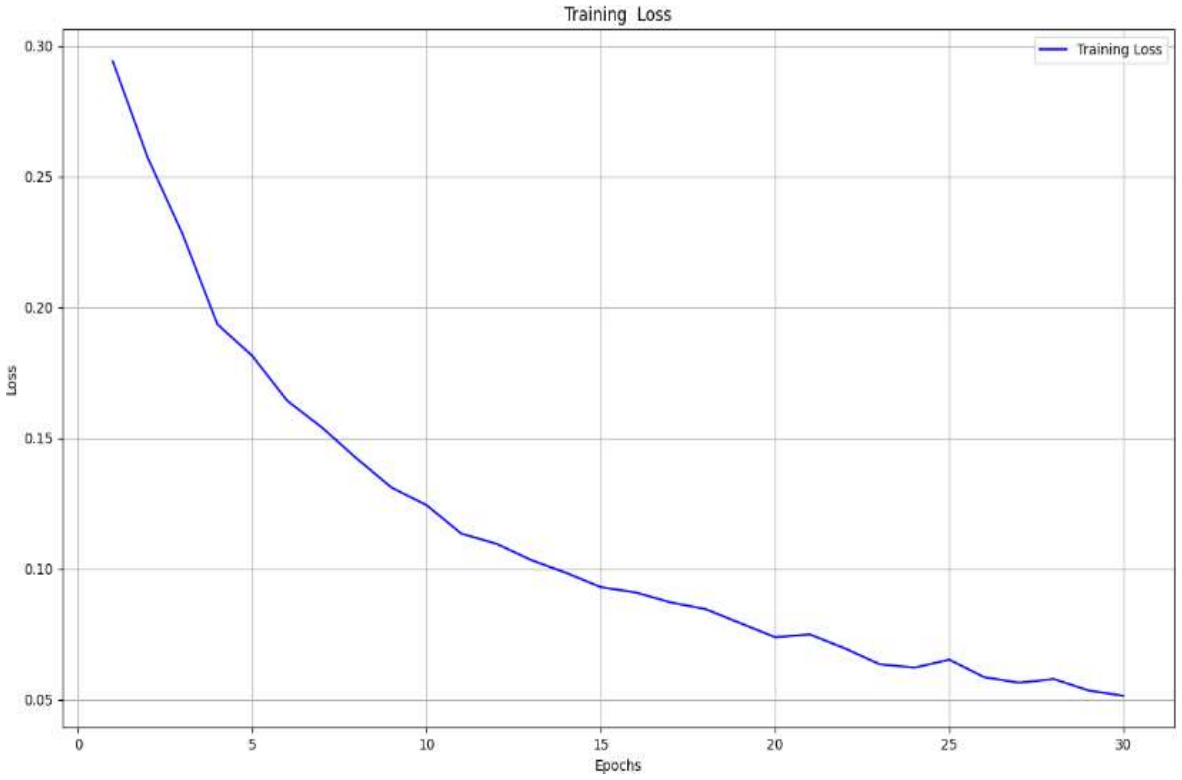


Figure A4. Training Loss (Epochs)

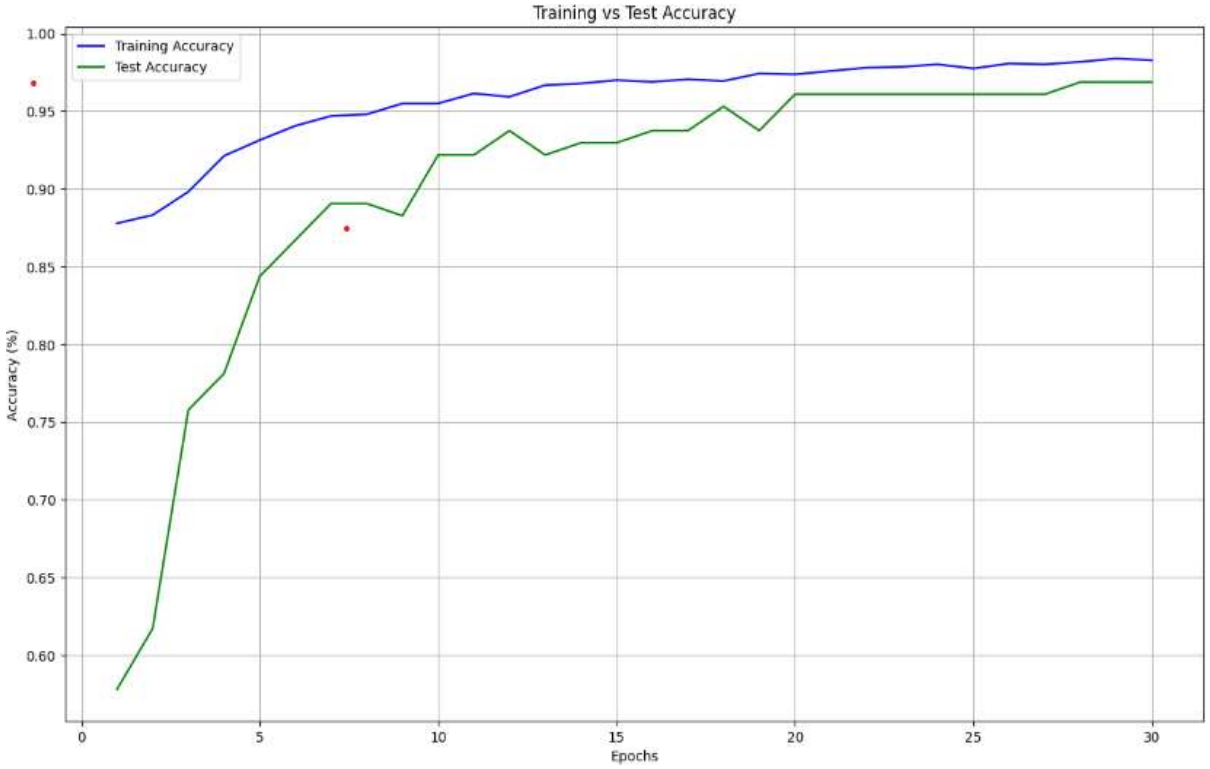


Figure A5. Training vs Test Accuracy (Epochs)

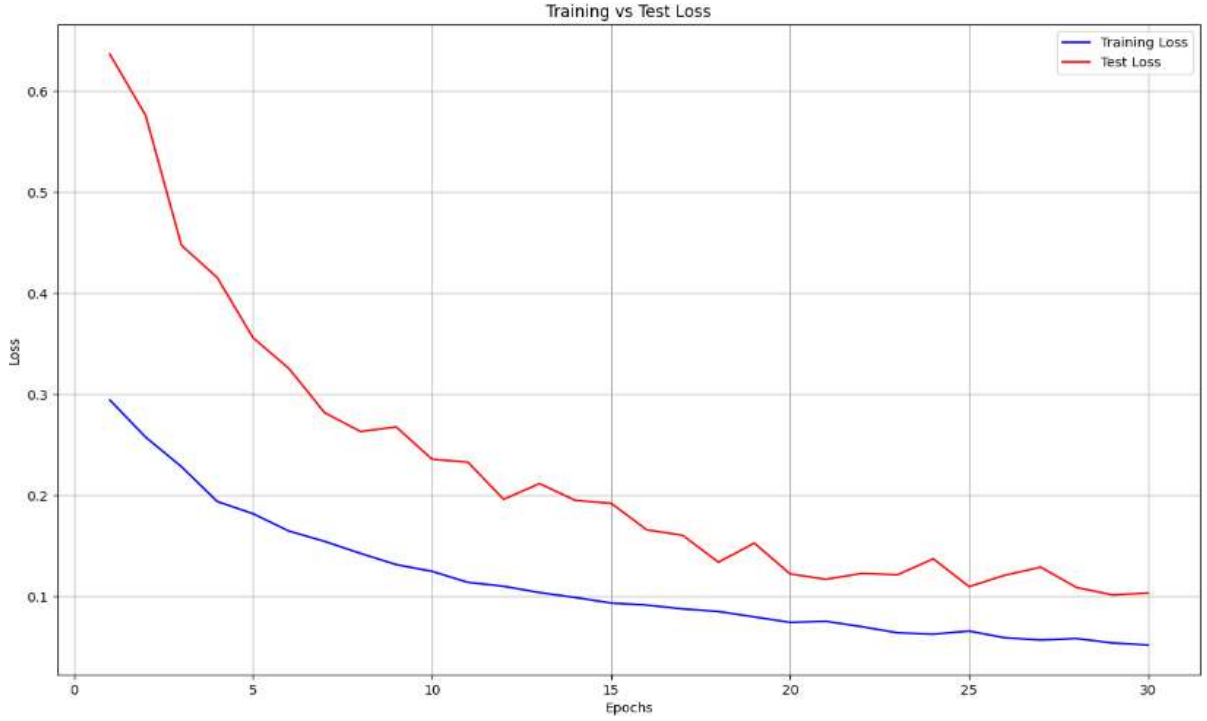


Figure A6. Training vs Test Loss (Epochs)

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
# Function to process images
def process_jpg_image(img):
    img = tf.convert_to_tensor(img[:, :, :3])
    img = np.expand_dims(img, axis=0)
    img = tf.image.resize(img, [224, 224])
    return img

# Function to test multiple images
def test_multiple_images(image_paths):
    predictions = []
    num_images = len(image_paths)

    # Calculate the number of rows and columns for subplots
    num_cols = 4
    num_rows = (num_images + num_cols - 1) // num_cols  # Ceiling division

    plt.figure(figsize=(15, num_rows * 5))  # Adjust height based on the number of rows

    for i, path in enumerate(image_paths):
        img = cv2.imread(path)
        if img is None:
            print(f"Error loading image: {path}")
            continue

        processed_img = process_jpg_image(img)
        pred = vgg.predict(processed_img)
        prediction = int(np.argmax(pred))
        predictions.append(prediction)

        # Display the image and prediction
        plt.subplot(num_rows, num_cols, i + 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title(f"Predicted: {class_names[prediction]}")
        plt.axis('off')

    plt.tight_layout()
    plt.show()
    return predictions # Return predictions for further analysis
# Define class names
class_names = list(train_generator.class_indices.keys())
print("Class names:", class_names)
# Define class names
```

```
class_names = list(train_generator.class_indices.keys())
print("Class names:", class_names)

# Example usage with multiple image paths
test_image_paths = [
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_9.jpg" ,
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_11.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_1.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_2.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_11.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_12.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\2f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\3f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\4f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\5f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\6f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\7f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\download (2).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\download.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\images (1).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\images (3).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\images (6).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_148.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_149.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_150.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_151.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_152.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_153.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_154.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_155.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_1.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_2.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_3.jpg",
```

```
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_4.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_5.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_6.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_7.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_8.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_9.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_10.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_11.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_12.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_5.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_6.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_7.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_8.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_9.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_10.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_11.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_12.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_3.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_4.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_5.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_6.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_7.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_8.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_9.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_10.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_3.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_4.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_5.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_6.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_7.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_8.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_3.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_4.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_5.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_6.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_7.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_8.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_9.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_10.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_11.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_3.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_4.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_5.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_6.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_7.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_8.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_9.jpg",
    r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_10.jpg",
```

```
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_11.jpg"

]
predictions = test_multiple_images(test_image_paths)
```

Class names: ['fake', 'real']

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 209ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 105ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 116ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 100ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 100ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 100ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 115ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 97ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 138ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 87ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 91ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 78ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 97ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 91ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 86ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 91ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 116ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 88ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 100ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 106ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 91ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 81ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 89ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 99ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 104ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 95ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 127ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 93ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 115ms/step

1/1 ——————————————————————— **0s** 94ms/step
1/1 ——————————————————————— **0s** 97ms/step
1/1 ——————————————————————— **0s** 87ms/step
1/1 ——————————————————————— **0s** 81ms/step
1/1 ——————————————————————— **0s** 77ms/step
1/1 ——————————————————————— **0s** 95ms/step
1/1 ——————————————————————— **0s** 82ms/step
1/1 ——————————————————————— **0s** 102ms/step
1/1 ——————————————————————— **0s** 82ms/step
1/1 ——————————————————————— **0s** 78ms/step
1/1 ——————————————————————— **0s** 85ms/step
1/1 ——————————————————————— **0s** 77ms/step
1/1 ——————————————————————— **0s** 92ms/step
1/1 ——————————————————————— **0s** 94ms/step
1/1 ——————————————————————— **0s** 96ms/step
1/1 ——————————————————————— **0s** 93ms/step
1/1 ——————————————————————— **0s** 103ms/step
1/1 ——————————————————————— **0s** 80ms/step
1/1 ——————————————————————— **0s** 77ms/step
1/1 ——————————————————————— **0s** 86ms/step
1/1 ——————————————————————— **0s** 96ms/step
1/1 ——————————————————————— **0s** 104ms/step
1/1 ——————————————————————— **0s** 121ms/step
1/1 ——————————————————————— **0s** 73ms/step
1/1 ——————————————————————— **0s** 95ms/step
1/1 ——————————————————————— **0s** 114ms/step
1/1 ——————————————————————— **0s** 140ms/step
1/1 ——————————————————————— **0s** 82ms/step
1/1 ——————————————————————— **0s** 106ms/step
1/1 ——————————————————————— **0s** 76ms/step
1/1 ——————————————————————— **0s** 86ms/step
1/1 ——————————————————————— **0s** 88ms/step
1/1 ——————————————————————— **0s** 91ms/step
1/1 ——————————————————————— **0s** 95ms/step
1/1 ——————————————————————— **0s** 107ms/step
1/1 ——————————————————————— **0s** 95ms/step
1/1 ——————————————————————— **0s** 89ms/step
1/1 ——————————————————————— **0s** 105ms/step
1/1 ——————————————————————— **0s** 98ms/step
1/1 ——————————————————————— **0s** 74ms/step
1/1 ——————————————————————— **0s** 109ms/step
1/1 ——————————————————————— **0s** 79ms/step
1/1 ——————————————————————— **0s** 89ms/step
1/1 ——————————————————————— **0s** 93ms/step
1/1 ——————————————————————— **0s** 83ms/step
1/1 ——————————————————————— **0s** 105ms/step
1/1 ——————————————————————— **0s** 94ms/step
1/1 ——————————————————————— **0s** 98ms/step
1/1 ——————————————————————— **0s** 124ms/step
1/1 ——————————————————————— **0s** 76ms/step

Predicted Output:

Predicted: fake

Predicted: real

Predicted: real

Predicted: real

Predicted: real

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

Predicted: fake

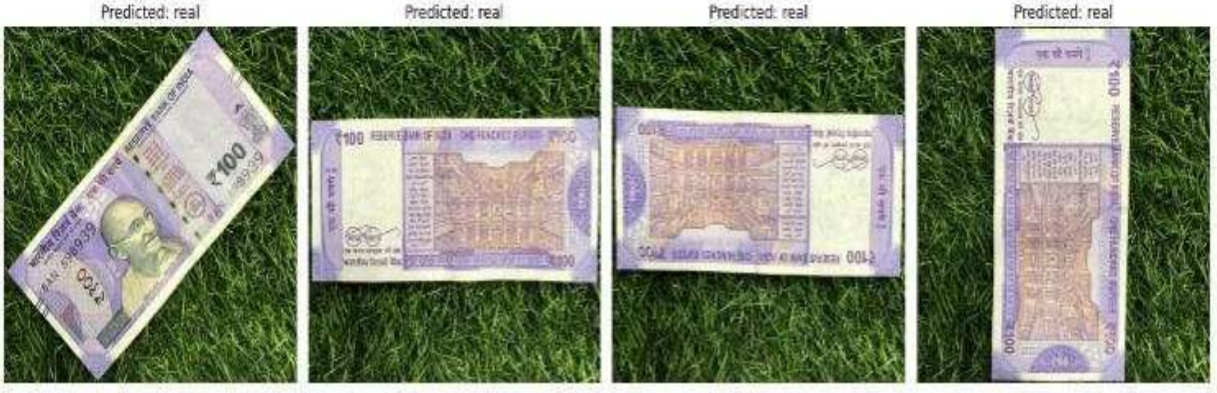
Predicted: real


Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real


Predicted: real
Predicted: real
Predicted: real
Predicted: real


Predicted: real
Predicted: real
Predicted: real
Predicted: real


Predicted: real
Predicted: real
Predicted: real
Predicted: real


Predicted: real
Predicted: real
Predicted: real
Predicted: real


Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: fake
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: fake
Predicted: real
Predicted: real
Predicted: real


Predicted: fake
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real