

**Final Year Project Report for the Degree of Bachelor of Computer  
Engineering**

**NEPALI MONEY RECOGNITION SYSTEM USING  
MULTILAYER PERCEPTRON**



<b>Bishal Godar</b>	<b>20070550</b>
<b>Paras Poudel</b>	<b>20070558</b>
<b>Ujjwal Bhusal</b>	<b>20070390</b>
<b>Laxmi Prasad Bhusal</b>	<b>20070556</b>
<b>Shiv Kumar Ram</b>	<b>20070569</b>

**United Technical College**  
**Faculty of Science and Technology**  
**Pokhara University, Nepal**  
**July, 2024**

Final Year Project Report for the Degree of Bachelor of Computer  
Engineering

**NEPALI MONEY RECOGNITION SYSTEM USING  
MULTILAYER PERCEPTRON**

**Supervised by: Er Pukar Neupane**

A Final year Project Report submitted in partial fulfillment of the requirements  
for the degree of Bachelor of Computer Engineering

Submitted By:

<b>Bishal Godar</b>	<b>20070550</b>
<b>Paras Paudel</b>	<b>20070558</b>
<b>Ujjwal Bhusal</b>	<b>20070390</b>
<b>Laxmi Prasad Bhusal</b>	<b>20070556</b>
<b>Shiv Kumar Ram</b>	<b>20070569</b>

**United Technical College**  
**Faculty of Science and Technology**  
**Pokhara University, Nepal**

**July, 2024**

## **Dedication**

The Project “**Nepali Money Recognition System Using Multilayer Perceptron**” is dedicated to all our respected parents and teachers who helped in completing it directly and indirectly. Our teachers and parents are our source of inspiration wisdom, knowledge and understanding. They have given us the drive and discipline to tackle the task with enthusiastic been possible. Thanking our parents and teachers with bottom of our heart for strengthening us when we thought of giving up and continually providing the moral, spiritual, emotional and financial support.

## **Declaration**

I hereby declare that this project entitled “**NEPALI MONEY RECOGNITION SYSTEM USING MULTILAYER PERCEPTRON**” is based on our original work. Related works on the topic by other researchers have been duly acknowledged. We owe all the liabilities relating to the accuracy and authenticity of the data and any other information included here under.

Name of the Students:

Bishal Godar

Shiv Kumar Ram

Ujjwal Bhusal

Laxmi Prasad Bhusal

Paras Poudel

Date:18, July,2024

## **Recommendation**

This is to certify that this project work entitled “**NEPALI MONEY RECOGNITION USING MULTILAYER PERCEPTRON**”, prepared and submitted by Bishal Godar, Shiv Kumar Ram, Paras Poudel, Laxmi Prasad and Ujjwal Bhusal in partial fulfillment of the requirements of the degree of Bachelor of Engineering (BE) in Computer Engineering awarded by Pokhara University, has been completed under my supervision. I recommend the same for acceptance by Pokhara University.

Signature:

Name of the Supervisor: Er. Pukar Neupane

Organization: United Technical College

Date: 18 July, 2024

## Certificate

This project entitled “**NEPALI MONEY RECOGNITION USING MULTILAYER PERCEPTRON**” prepared and submitted by Shiv Kumar Ram, Bishal Godar, Laxmi Prasad Bhusal, Ujjwal Bhusal and Paras Paudel has been examined by us and is accepted for the award of the degree of Bachelor of Computer Engineering by Pokhara University.

Name of Supervisor: Er. Pukar Neupane

Designation: Lecturer

Organization: United Technical College

Signature:

Date:

Name of External:

Designation:

Organization:

Signature:

Date:

Name of Head of Department: Er. Sahit Baral

Signature:

Date:

Name of Principal: Prof. Keshab Datt Awasthi (Ph.D.)

Signature:

Date:

## **Acknowledgements**

We have given our efforts in this project. However, it would not have been possible without the kind support and help of teachers and individuals. We would like to extend our sincere thanks to all of them.

We would like to express our special thanks and gratitude to our supervisor Er. Pukar Neupane and our Head of Department Er Sahit Baral, who helped us in completing the project. I sincerely thanks for the time spent proceeding and correcting our mistakes. We are overwhelmed in all humbleness and gratefulness to acknowledge our depth to all those who have helped us directly and indirectly to put these ideas well above the level of simplicity and into something concrete. Any attempt at any level can't be satisfactorily complete without the support of our parents and our friends in finalizing the project in limited time frame. We came to learn lot of things in this time and sharpen our knowledge and skills.

Name of the Students:

Ujjwal Bhusal

Bishal Godar

Shiv Kumar Ram

Laxmi Prasad Bhusal

Paras Poudel

Date:18 July, 2024

## Abstract

The primary goal of this project is to develop a web-based system for the recognition of Nepalese paper currency, making it accessible and convenient for users to identify currency anytime and anywhere. The system leverages advanced image processing techniques and a multilayer perceptron neural network for Optical Character Recognition (OCR). It processes digital images to locate and recognize characters, including letters, numbers, and symbols on currency notes. Key features of the project include edge detection, color analysis, texture recognition, and OCR, which are used to extract essential characteristics from the currency notes. The project employs the ResNET-50 model for image classification, utilizing a dataset of Nepali currency collected from Kaggle for training, validation, and testing purposes. The system is built on platforms such as Google Colab and Pycharm and integrates the ResNET-50 V2 model with the Money Recognition System.

The tasks completed include model selection, dataset collection and preparation, platform selection, model integration, training, validation, and ongoing deployment. The remaining tasks involve further data collection to balance between genuine and counterfeit currency and the development of a user-friendly interface to connect users with the recognition system. This project aims to provide accurate currency recognition with an expected accuracy of 96%, aiding users in various scenarios such as banking, shopping malls, and other commercial establishments, thus improving efficiency and reducing manual effort.

**Keywords:** *Color, Texture, ResNet50 Template matching, Image processing, OCR, Database, Feasible*



# Table of Contents

Dedication .....	ii
Declaration .....	iii
Recommendation .....	iv
Certificate.....	v
Acknowledgements.....	v
Abstract .....	vi
List Of Figures .....	ix
List Of Tables.....	x
List Of Abbreviation .....	xi
Chapter 1: Introduction .....	1
1.1. Background .....	1
1.2. Statements of problem.....	2
1.3. Objectives.....	3
1.4. Application .....	3
1.5. Scope .....	3
1.6. Limitation .....	4
Chapter 2: Literature Review .....	5
2.1 Case Study .....	5
Chapter 3: Methodology .....	10
3.2. State Diagram .....	10
3.3. Activity diagram.....	11
3.5. Data Collection Methods.....	12

3.5.1.	Data Collection .....	12
3.5.2.	Data Information .....	13
3.5.3.	Data Processing.....	13
3.6.	System Requirement Specification .....	16
3.6.1	Functional Requirements .....	16
3.6.2	Non-Functional Requirements .....	16
3.7.	Gaussian Blur Algorithm.....	17
3.7.1.	Algorithm to extract data sets .....	17
3.8.	Multi-Layer Perceptron Algorithm with Back Propagation .....	17
6.1	RESNET-50 AS a Model.....	20
Chapter 4:	Feasibility Analysis .....	30
4.1.	Technical Feasibility .....	30
4.2	Economic Feasibility .....	36
4.3	Operational Feasibility.....	36
CHAPTER 5:	RESULT & CONCLUSION.....	37
5.1	Task Done .....	37
CHAPTER 6:	CONCLUSION AND FUTURE ENHANCEMENT .....	46
6.1	Conclusion.....	46
6.2	Future Enhancement.....	47
References	.....	48
Appendix	.....	51

## List Of Figures

Figure 3.1: Methodology (Group Work, 2024).....	10
Figure 3.2:State Diagram (Group Work,2024) .....	10
Figure 3.4: Activity Diagram (Group Work, 2024) .....	11
Figure 3.5: Rs 1000 front.....	12
Figure 3.6: Rs 1000 back .....	12
Figure 3.7: Image of Denomination and Animal(Group Work, 2024).....	13
Figure 3.8: Back Propagation (Jiawei Han, 2012).....	18
Figure 3.9: Skip Connection .....	21
Figure 3.10: Architecture Of ResNet50 .....	23
Figure 3.11: Flow of ResNet 50.....	27
Figure 4.1: Anaconda navigator.....	35
Figure 5.1:Library Import .....	38
Figure 5.2: Training .....	38
Figure 5.3: Testing .....	39
Figure 5.4: Evaluation.....	40
Figure 5.5: Graph for Accuracy and Loss.....	41
Figure 5.6: Confusion Matrix .....	42
Figure 5.7: Home page.....	43
Figure 5.8: After Upload .....	44
Figure 5.9: After Classify-1 .....	44
Figure 5.10: After Classify -2 .....	45

## **List Of Tables**

Table 1: Functional Requirements(Group Work, 2024).....	16
Table 2: Data Collection .....	37

## **List Of Abbreviation**

2D	Two Dimensional
ATM	Automated Teller Machine
FR	Functional Requirement
GA	Genetic Algorithm
HTML	Hypertext Markup Language
MLP	Multi-Layer Perceptron
NN	Neural Network
OCR	Optical Character Recognition
OpenCV	Open-Source Computer Vision Library
ResNet50	Residual Network 50
SA	Simulated Annealing
SIFT	Scale Invariant Features Transform

# Chapter 1: Introduction

## 1.1. Background

The web portal will help common people for currency recognition anywhere anytime. Automatic method for detection of currency note is very important in every country. In this approach, system extract the general attributes of the paper currency like various dominant parts of image of currency note (like cut size of length, width, grammage and thickness of paper etc.). The currency recognition system is used in many scenarios such as bank, business firms, railways, shopping malls, departmental stores, government organization, etc. But recognition is done majorly using hardware device. Also, common man cannot find it feasible to use it as hardware. So, there is a need to computerize the human effort to recognize the currency (Shakya, 2015).

Consider the example of a bank; it needs to recognize the denomination every now and then they use the device which consist of ultraviolet light. The banker keeps the currency note on the device and try to find whether the watermark symbol, serial number and some other characteristics of the notes are proper to get the denomination and check its authenticity. This increases the work of the banker. If the banker uses the system and computerizes his work, the result will be much more accurate. Same is the case with areas such as shopping malls, investment firms where such systems can be used. So, there is needed to make easier way to recognize the currency notes. The identification marks help to know the denomination of currency. These marks of currency help to detect the respective value of the currency (Bahrani, 2020).

If all these procedures could be done with the help of an automated program, then it would be lot easier for the foreigners to recognize the amount and time could be saved for the tellers since they would not have to separate the money as per their amount and then feed it to the banknote counter. As a result, time and efficiency both could be increased. If a currency recognition system can be properly implemented providing the facilities to provide the denomination of a note, then acceptable level of recognition performance can be reached. In this approach system extracts the general attributes of the paper currency like various dominant parts of image of currency note (like identification marks present in 4 corners of the note, size, color etc.). The identification marks help to know the denomination of currency. The system will be developed to check 7 different currency notes of 5,10,20,50,100, 500 and 1000 rupees. The Web Application will display currency denomination after the user uploads the picture of the currency note in the system. The system simply extracts feature of currency which were match with original currency features and

immediately displays result with accuracy. The system simply extracts feature of currency which were then matched with original currency features and result are displayed immediately with accuracy of 96%. The features which were considered for currency recognition are as follows:

- Physical Dimensions
- Denomination
- Mount Everest at right corner
- Nepal Rastra Bank at top of currency note
- Nepal Rastra Bank logo

Yak, Swamp deer, Snow Leopard, Two one-horned rhinoceros in grassy plain, Tiger, Elephant (Shakya, 2015).

## **1.2. Statements of problem**

Main purpose of the system is to recognize the denomination of the paper currency. There are lots of machines that helps the people to recognize paper currency. Those machines ATM, top up machines, etc. which are not accessible by people at any time they want. Such machines are very expensive and only the business firm involved in related field own them (Ami Shah, 2015).

This system is based on image processing using neural networks. User can upload the image of currency and the system extracts the features of currency and compares with stored dataset which helps in recognizing the denomination of uploaded currency. This system is available to common people so they can easily utilize anywhere and at any time.

In the past, traditional methods of currency recognition relied heavily on manual inspection and sorting, which posed challenges in terms of efficiency, accuracy, and speed. Human errors, the need for extensive training, and the inability to handle large volumes of currency transactions efficiently were notable drawbacks. As technological advancements progressed, the introduction of early machine vision systems attempted to automate the process, but their accuracy was limited, and they struggled with variations in currency designs and security features (Fukumi, 2000).

Despite advancements in technology, the current state of money recognition systems still faces certain challenges. The proliferation of counterfeit currency, evolving security features on banknotes, and the need for rapid and accurate identification in diverse environments (such as retail, banking, and assistive technologies for the visually impaired) present ongoing obstacles. Additionally, the increasing variety of currencies globally, each with its unique design and security

features, complicates the development of universally applicable recognition models. In the present scenario, AI solutions have shown promise in overcoming these challenges by leveraging machine learning algorithms and computer vision techniques (Nanda, 2012).

Looking ahead, the future of money recognition systems using AI aims to address the existing limitations and usher in a new era of efficiency, adaptability, and security. Anticipated challenges include the need for more robust anti-counterfeiting measures, the integration of AI into emerging financial technologies, and the establishment of standardization across diverse currencies. Future systems are expected to enhance not only accuracy and speed but also inclusivity, with improved features for assisting individuals with visual impairments. The integration of advanced technologies such as blockchain for secure digital currencies and real-time, cloud-based recognition systems may become prominent trends in the evolution of AI-based money recognition (Ami Shah, 2015).

In summary, the past struggles with manual processes and early automated systems have paved the way for the present exploration of AI solutions. As we move into the future, addressing current challenges and anticipating emerging needs will drive the development of sophisticated and comprehensive money recognition systems using artificial intelligence (Ami Shah, 2015).

### **1.3. Objectives**

1. To implement Multilayer Perceptron(Resnet-50 ) to recognize Nepali Paper Currency with highest accuracy as possible

### **1.4. Application**

1. It is used in Cash deposition system in Bank.
2. It is used in Account Section in Super Markets.
3. It is used to identify money by the foreign peoples.

### **1.5.Scope**

Nepalese Currency Denomination can be used by the financial institution were recognizing the currency and separating the currencies as per their values can optimized. This system can simplify the work of the tellers and currency can be recognized from the same machine rather than separating the currencies as per their values. It can also be used by the foreign people who have



confusion recognizing the Nepalese currency. Not Only, this system can also be used in huge supermarkets where small task like Money recognition is manually performed by the cash people and can be automated to make their task easier.

## **1.6.Limitation**

1. One must be able to operate computer
2. All the paper notes should be collected and their record must be kept so as to give appropriate result.
3. Currency value can be translated to only those languages which are available in google translate.
4. People using browser to identify the currency need to have internet connection

## Chapter 2: Literature Review

### 2.1 Case Study

#### 1. Recognition Indian Currency Fake Note Detection System Using Resnet 50.

This study leverages a deep CNN, specifically ResNet-50, to enhance the detection of counterfeit currency. ResNet-50, known for its depth and ability to mitigate the vanishing gradient problem through residual learning, is well-suited for this task. The use of transfer learning, where a pre-trained network is fine-tuned on a specific dataset, further enhances the model's performance, particularly when dealing with limited data (Riddhi Sindhe, 2023).

The dataset used in this study comprises two thousand currency notes, providing a robust foundation for training the network. By learning the feature map of genuine currency notes, the network can accurately distinguish between authentic and counterfeit notes. This approach not only improves accuracy but also significantly reduces the time required for detection, addressing a major drawback of traditional methods (Riddhi Sindhe, 2023).

In summary, the integration of deep learning techniques, particularly CNNs, into counterfeit detection systems represents a significant advancement over traditional methods. The ability to quickly and accurately identify counterfeit currency using a trained network like ResNet-50 offers a promising solution to this persistent problem. This study approach demonstrates the potential of deep learning to enhance the efficiency and reliability of counterfeit detection, thereby contributing to economic stability and security (Riddhi Sindhe, 2023).

#### 2. Deep Residual Learning for Image Recognition.

The paper "Deep Residual Learning for Image Recognition" by introduces a transformative framework aimed at overcoming the challenges associated with training very deep neural networks. The core innovation lies in the use of residual blocks, which enable layers to learn residual functions with respect to their inputs rather than learning direct mappings. This technique addresses the degradation problem, where increasing the depth of the network leads to higher training and test errors due to optimization

difficulties. By reformulating the layers in this manner, the authors successfully train networks as deep as 152 layers, surpassing the depth of previous architectures (Kaiming He, 2016).

The methodology includes standard components like convolutional layers, batch normalization, and ReLU activations, but the introduction of identity Skip is what allows the networks to effectively learn residual mappings. The residual learning framework employs identity Skip connections that bypass one or more layers, facilitating the flow of gradients during backpropagation and making it easier to optimize very deep networks. The empirical results are remarkable: the 152-layer residual network (ResNet) achieves a top-5 error rate of 3.57% on the ImageNet dataset, outperforming all preceding models. Additionally, the framework demonstrates significant performance improvements on other challenging tasks, such as object detection and segmentation in the COCO dataset, showcasing a 28% relative improvement (Kaiming He, 2016).

### 3. Paper currency recognition for color images based on Artificial Neural Network

(Ch. Ratna Jyothi, 2016) introduced four different kinds of currencies through computer vision with typical Accuracy rate was 93.84%.

In designing such a system, it considers different dimensions, Color luminance, Edge histogram using max sobel gradient, correlations as features. A different method using radial basis Function networks, is utilized for developing an intelligent system which can recognize paper currency. This research is specifically designed for recognizing paper currency from United Kingdom (“Pound”), Japan (“Yen”), Europe(“euro”) and India(“rupee”). In the proposed paper recognition technique has been designed in such a way that it can be used for recognizing paper currency of four different countries. To overcome the problem of recognizing dirty banknotes, the preprocessing stage is also considered (Ch. Ratna Jyothi, 2016).

### 4. An intelligent paper currency recognition system

(Sarfraz, 2015) proposed an Android paper currency recognition system that applied to Saudi Arabian papers. Recognizing paper currency methods that relies on some features and correlations between two currency Images.

Radial Basis Function Network for classification method uses the case of Saudi Arabian paper currency as a model. The method is quite reasonable in terms of accuracy. The system deals with 110 images, 10 of which are tilted with an angle less than  $15^\circ$ . The rest of the currency images consist of mixed including noisy and normal images 50 each. It uses fourth series (1984–2007) of currency issued by Saudi Arabian Monetary Agency (SAMA) as a model currency under consideration. The system produces accuracy of recognition as 95.37%, 91.65%, and 87.5%, for the Normal Non-Tilted Images, Noisy Non-Tilted Images, and Tilted Images respectively. The overall Average Recognition Rate for the data of 110 images is computed as 91.51% (Sarfraz, 2015).

##### 5. Design and evaluation of neural networks for coin recognition by using GA and SA

This paper way to fashion a neural network using a simulated annealing and genetic algorithm. The comparable traits of the pictures of coins (i.e., size, color, weight, and pattern) reason hassle for forex recognition. The proposed scheme located numerous capabilities and additionally the recognition rate turned into about 98%. However, as a problem becomes complex and large-scale, the amount of operation increases, hardware implementation to real systems (coin recognition machines using NNs) becomes difficult. To make a small-sized NN system conducts to simplify hardware implementation to the real systems (Fukumi, 2000).

In this paper, a new scheme which makes a small-sized NN system is proposed. Some schemes are proposed which make a small-sized NN system. However, it costs when an image is taken in with an image scanner because the perfect image of a coin is used for learning and recognition. When hardware implementation to a real system is considered, the most important points are follows:

- Recognition accuracy must be a 100 percent or close to 100 percent.
- It is a low-cost system.
- When it is applied to a real system, it doesn't consume recognition time (Fukumi, 2000).

This paper focuses on reducing costs by using a partial coin image taken by a cheap scanner, aiming to design a smaller neural network (NN) system that achieves near 100% recognition accuracy. Genetic Algorithms (GA) and Simulated Annealing (SA) are utilized to enhance

performance. GA mimics natural evolution, while SA emulates the slow cooling process in metallurgy to avoid metastable states, allowing the system to escape local minima. Using GA and SA together mitigates the drawbacks of each method used in isolation, enabling a global search and yielding desirable results quickly. The paper first discusses reducing computational effort in coin recognition, then explains NN design using GA and SA, and finally, demonstrates the scheme's effectiveness through computer simulations (Fukumi, 2000).

6. Design and implementation of Indian paper currency authentication system based on feature extraction by edge-based mentation using Sobel operator.

(Nanda, 2012) Proposed method uses three extracted functions from the banknote along with identity mark, watermark, and safety thread Results shows the system gives good performance for images with low noise with accuracy 98.3%.

The technology of currency recognition aims to efficiently classify paper currency by extracting visible and hidden marks. Traditional methods, relying on features like size and color, struggle with worn, dirty, or altered banknotes. Using edge information and a three-layer BP neural network (NN) can improve recognition, but NNs require extensive training samples and proper distribution to avoid overfitting and poor generalization. In circulation, paper currency often loses original information due to wear and damage, complicating automatic recognition. Therefore, extracting characteristic information and selecting effective recognition algorithms is crucial. This paper presents a simple, efficient method that meets practical high-speed application needs. Utilizing digital image processing, the method involves capturing the currency image with white backlighting to reveal hidden marks, followed by pre-processing, edge detection, segmentation, and characteristic extraction to improve accuracy (Nanda, 2012).

7. Evaluating color descriptors for object and scene recognition

(KEA Van De Sande, 2008) Done Comparison between the local color descriptors with grey value descriptors. They use the evaluation framework (KEA Van De Sande, 2008) Schmid to the amount of local grey value invariants. The results show the strategy which mixes color information and SIFT gives better results.

This paper examines the invariance properties and distinctiveness of color descriptors systematically. First, it presents a taxonomy of invariant properties based on the diagonal model of illumination change. This approach defines invariance to light intensity changes, light intensity shifts, light color changes, and combined light color changes and shifts. The distinctiveness of color descriptors is then experimentally analyzed using two benchmarks: one from the image domain (photographs) and one from the video domain (news broadcasts). Extensive experiments on a large set of real-world images and videos demonstrate the practical utility of these different invariant properties. (KEA Van De Sande, 2008).

This paper is organized as follows. In section 2, the reflectance model is presented. Further, its relation to the diagonal model of illumination change is discussed. In section 3, a taxonomy is given of color descriptors and their invariance properties. The experimental setup is presented in section 4. In section 5, a discussion of the results is given. Finally, in section 6, conclusions are drawn (KEA Van De Sande, 2008).

## **8. Implementation of ResNet-50 on End-to-End Object Detection (DETR) on Objects**

Object recognition in images remains a significant challenge in computer vision. End-to-end object detection, which leverages CNN and Transformer architectures, has shown promise in addressing this problem. This research implements ResNet-50 in an End-to-End Object Detection system to enhance performance. ResNet-50, known for its image recognition effectiveness, and DETR, which uses Transformers to learn object representations, were tested on the COCO dataset. The ResNet-50 + DETR system achieved higher accuracy and faster detection speeds compared to traditional CNN models and DETR without ResNet-50, improving object detection performance by about 90%. These findings highlight the potential of ResNet-50 in DETR systems to advance object detection technology, particularly in real-time applications. (Endang Suherman, 2023).

## Chapter 3: Methodology

### 3.1.System Architecture

The recognizer is based on neural network classifier that feeds on datasets. The overview of methodology is as shown in the figure below:

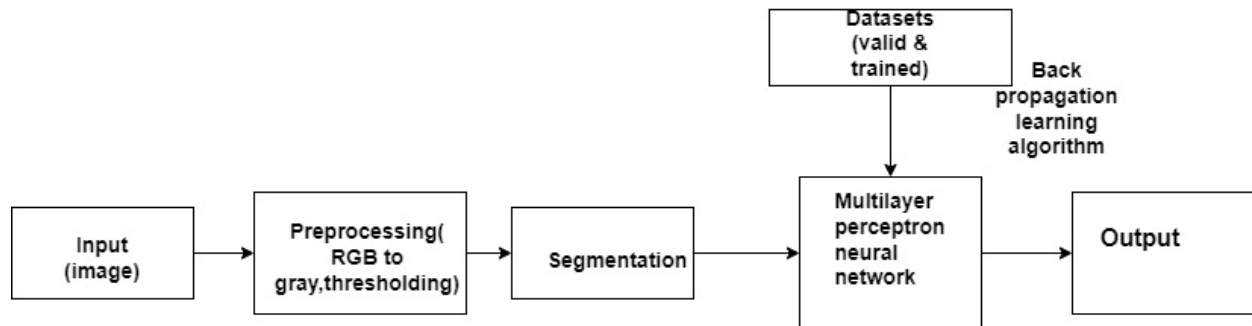


Figure 3.1: Methodology (Group Work, 2024)

### 3.2.State Diagram

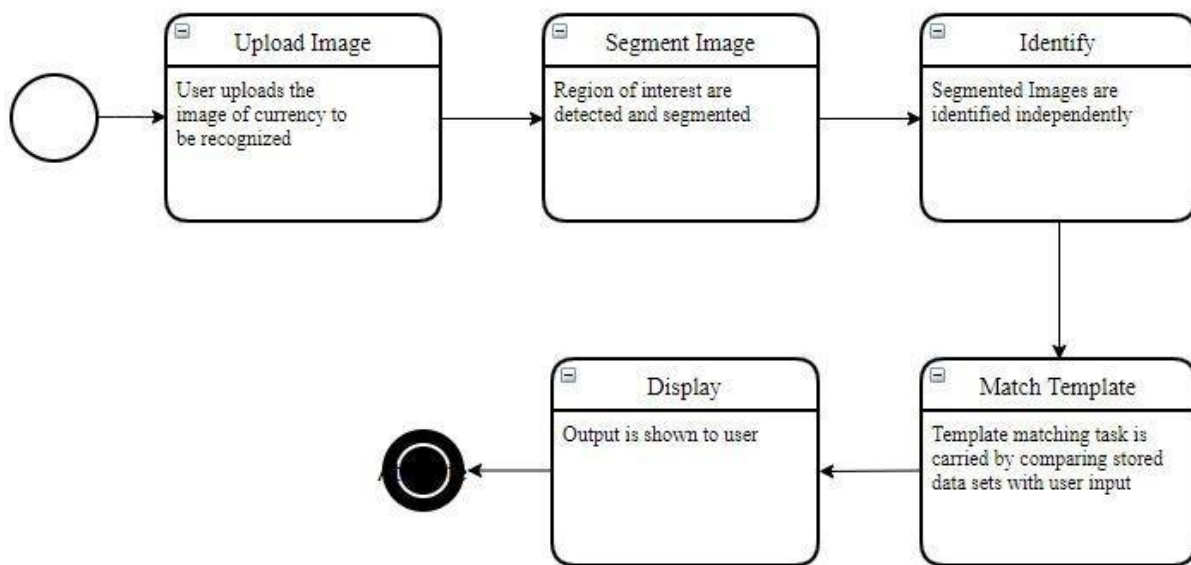


Figure 3.2:State Diagram (Group Work,2024)

### 3.3. Activity diagram

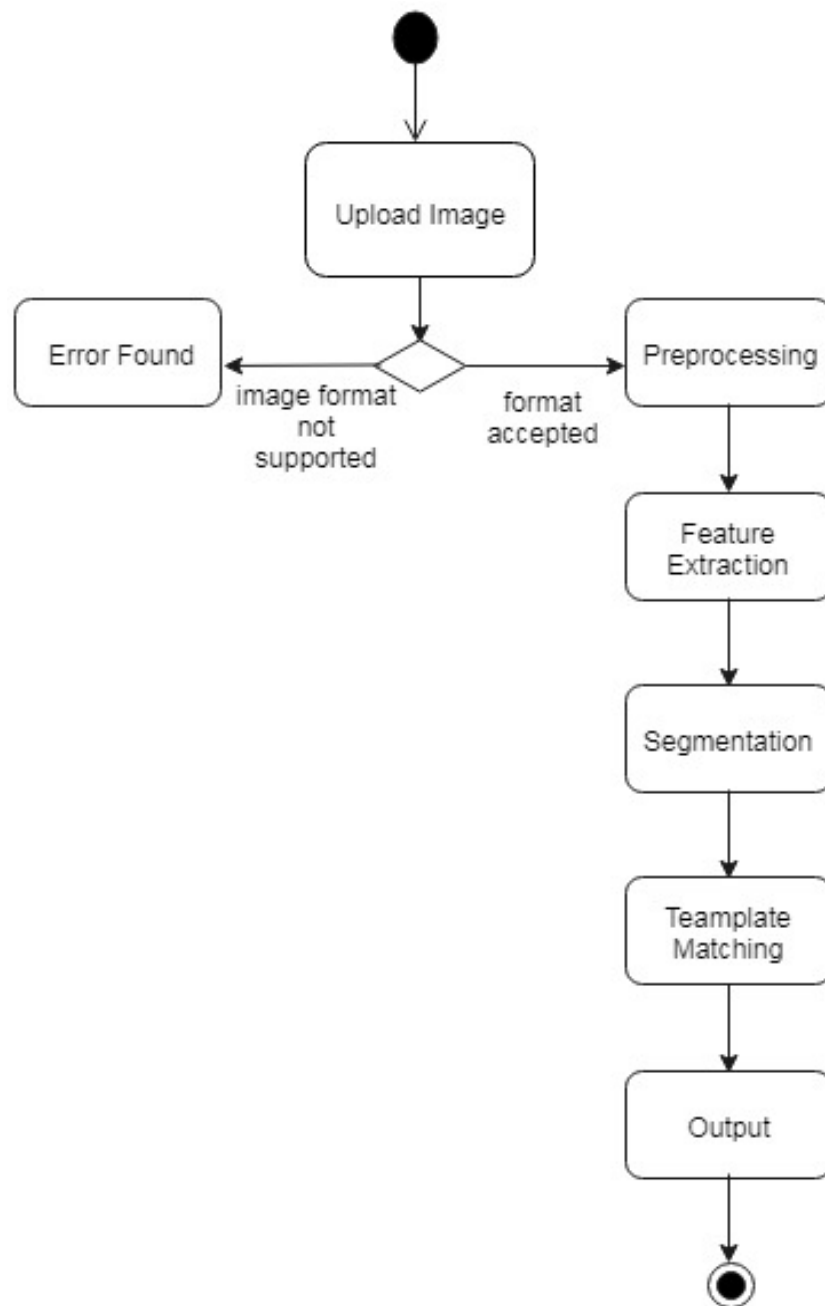


Figure 3.3: Activity Diagram (Group Work, 2024)



### 3.5.Data Collection Methods

#### 3.5.1. Data Collection

Dataset was collected from the [Kaggle](#). Altogether around 3600 data were collected which used for training purpose.



Figure 3.4: Rs 1000 front(Group Study, 2024)



Figure 3.5: Rs 1000 back(Group Study, 2024)

### 3.5.2. Data Information

This consists of different segments of currency. For example: The image of denomination, Colors, and animal looks like this:

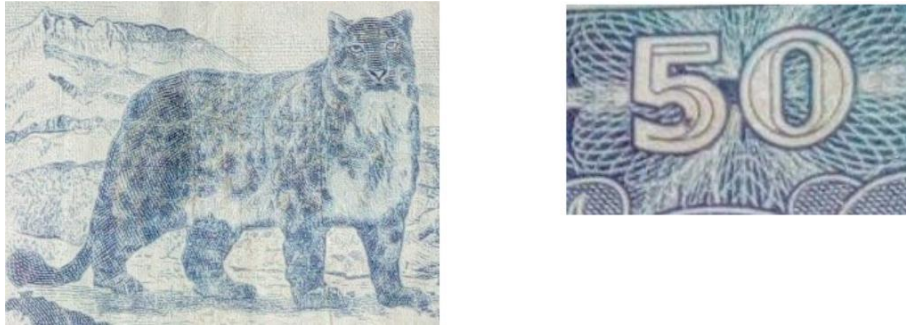


Figure 3.6: Image of Denomination and Animal(Group Work, 2024)

### 3.5.3. Data Processing

The processing was while detecting and recognizing the image uploaded by the user. The steps are as follows:

1. Preprocessing:

- Image Rotation: Images were rotated upto 40 Degree.
- Shear Range: 0.2
- Zoom Range: 0.2
- Horizontal Flip: True
- Fill Mode: Nearest
- Image Acquisition: Collection of a dataset of images containing various money notes under different lighting conditions and orientations (Rosebrock, 2021).
- Image Resizing: Resize all images to a consistent size to ensure uniformity in input dimensions (Rosebrock, 2021).
- Grayscale Conversion: Convert images to grayscale to simplify processing, reduce dimensionality, and retain important features for money recognition (Rosebrock, 2021).

- Normalization: Normalized pixel values to a standardized range i.e  $[0, 1]$  to improve convergence and stability during training (Rosebrock, 2021).

## 2. Feature Extraction:

- Flattening: Flatten the 2D grayscale images into 1D vectors. This converts the image data into a format suitable for input to the MLP (Rosebrock, 2021).
- Edge Detection (Optional): Apply edge detection techniques to highlight edges, which can be important features for recognizing currency notes (Nanda, 2012).

## 3. Segmentation:

- Localization: Identify and isolate regions of interest (ROI) within the image that likely contain currency notes. This could involve techniques such as object detection or region-based methods (Nanda, 2012).
- Bounding Box Extraction: Draw bounding boxes around the segmented regions to define the regions for further processing (Nanda, 2012).
- ROI Cropping: Crop and extract the regions of interest based on the bounding boxes (Nanda, 2012).

## 4. Template Matching:

- Template Creation: Create templates for each type of currency note to be recognized. These templates represent the distinctive features of each denomination (Nanda, 2012).
- Template Matching: Use template matching techniques to compare the extracted regions of interest with the created templates (Nanda, 2012).
- Score Calculation: Assign similarity scores to the matched templates. The template with the highest score is considered the recognized currency denomination (Nanda, 2012).

## 5. MLP Training with Backpropagation:

- Input Layer: Use the flattened, normalized, and preprocessed images as input to the MLP (Waldo, 2022).

- **Hidden Layer:** Design the architecture of the MLP with one or more hidden layers. The number of neurons in these layers and their activation functions can be tuned based on the complexity of the recognition task. Design the architecture of the MLP with one or more hidden layers. The number of neurons in these layers and their activation functions can be tuned based on the complexity of the recognition task (Waldo, 2022).
- **Output Layer:** The output layer should have neurons corresponding to the different currency denominations, and a suitable activation function (e.g. softmax) for classification (Waldo, 2022).
- **Backpropagation Algorithm:** Train the MLP using the backpropagation algorithm to minimize the difference between predicted and actual currency labels. Adjust weights and biases iteratively to improve the network's performance (Waldo, 2022).

#### 6. Evaluation:

- **Test Set Validation:** Evaluate the trained MLP on a separate test set to assess its generalization to new unseen data (Waldo, 2022).
- **Performance Metrics:** Use appropriate metrics such as accuracy, precision, recall and F1 score to quantify the system's performance (Waldo, 2022).

### 3.6.System Requirement Specification

#### 3.6.1 Functional Requirements

Functional Requirement Number	Functional Requirement Description
FR1	The system should process the input given by the user
FR2	System should detect the denomination present in the image
FR3	System should detect the boundary
FR4	System should crop the region of interests (ROI)
FR5	System should match the ROI with datasets
FR6	System shall show the error message to the user when the input given is not in the required format.
FR7	System should retrieve characters (currency denomination) present in the image and display them to the user

Table 1: Functional Requirements (Group Work, 2024)

#### 3.6.2 Non-Functional Requirements

- Performance: Currency denomination in the input image will be recognized with an accuracy about 96% and more.
- Functionality: This software will deliver on the functional requirements mentioned in this document.
- Flexibility: It provides the users to load the image easily.
- Learnability: The software is very easy to use and reduces the learning work.

### 3.7. Gaussian Blur Algorithm

In two-dimension, Gaussian function is the product of two 1D Gaussian functions, one in each dimension:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \quad (\text{Gedraite, 2011}) \dots\dots\dots (1)$$

Where, y is the distance along vertical axis from the origin,

x is the distance along horizontal axis from the origin

$2\pi$  is normalization constant

$\frac{1}{2\pi\sigma^2}$  is normalization factor

$\sigma$  is the standard deviation (Sharda, 2021).

#### 3.7.1. Algorithm to extract data sets

- Gaussian Blur of a 2D image can be defined as the convolution of that image with the 2D Gaussian function.

$$b[i, j] = \sum_{y=i-r}^{i+r} \sum_{x=j-r}^{j+r} red[y, x] * \omega[y, x] \dots\dots\dots (2)$$

Where,

i,j is pixel location in output matrix b

r is radius around pixel

y,x is indices of pixel location

red[y, x] is red channel value of image at position (y, x)

w[y, x] is weight

- Contour is used to find out the dimension of the image and to detect the desired shape of the image (Sharda, 2021).

### 3.8. Multi-Layer Perceptron Algorithm with Back Propagation

Multi-Layer perceptron (MLP) is a feed forward neural network with one or more layers between input and output layer. Feed forward means that data flows in one direction from input to output layer (forward). This type of network is trained with the back propagation learning algorithm. MLPs are widely used for pattern classification, recognition, prediction and approximation. A

multi-layer perceptron can be used to derive Back Propagation Learning Algorithm (Jiawei Han, 2012).

For back propagation algorithm, in order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (Jiawei Han, 2012).

In order to have some numbers to work with, let us take an example to specify the initial weights, the biases, and training inputs/outputs. The goal of back propagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs. The figure 3.8 below shows us the working module of Backpropagation algorithm (Jiawei Han, 2012).

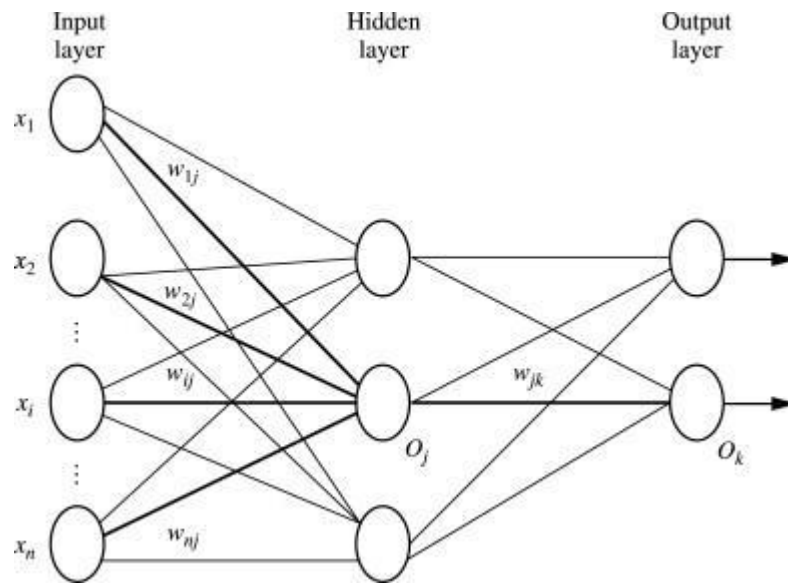


Figure 3.7: Back Propagation (Jiawei Han, 2012).

Parameters :

$x$  = inputs training  $x=(x_1, x_2, \dots, x_n)$ .

$t$  = target vector  $t=(t_1, t_2, \dots, t_n)$ .

$\delta_k$  = error at output unit.

$\delta_j$  = error at hidden layer.

$\alpha$  = learning rate.

$O_j$  = bias of hidden unit  $j$ .

The back propagation learning routine in MLP is given as,

1. Choose and fix the architecture for the network, which will contain input, hidden and output units, all of which will contain sigmoid functions.

2. Randomly assign the weights between all the nodes.
3. Each training example is used, one after another, to re-train the weights in the network.
4. After each epoch (run through all the training examples), a termination condition is checked (also detailed below). Here, we are not guaranteed to find weights which give the network the global minimum error, i.e., perfectly correct categorization of the training examples. Hence the termination condition may have to be in terms of a (possibly small) number of mis-categorization (Jiawei Han, 2012).

As an activation function, logistic function is used which is used to derive the sigmoid function as well when  $L=1$ ,  $k=1$ ,  $x_0=0$ . The equations 3 and 4 shows Logistic Function and Sigmoid Function respectively.

$$f(x) = \frac{L}{1+e^{-k(x-x_0)}} \dots\dots\dots(3)$$

$$S(t) = \frac{1}{1+e^{-t}} \dots\dots\dots(4)$$

The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt. In this project, a popular learning algorithm Multi-layer perceptron Neural Network using Back Propagation has been implemented to classify the dominations from the given images of the currency notes. The core principle of MLP Neural Network is to utilize a supervised learning technique called back propagation for training. Its multiple layers and nonlinear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to a two-layer input-output model. Logistic function is a common activation function which is similar in shape but ranges from 0 to 1 (Jiawei Han, 2012).

$$y(vi) = \tanh(vi) \quad \text{and} \quad y(vi) = (1 + e^{-vi})^{-1} \dots\dots\dots(5)$$

Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through backpropagation, a generalization of



the least mean squares algorithm in the linear perceptron. We represent the error in output node  $j$  in the  $n$ th data point by

$$e_{\{j\}}(n) = d_{\{j\}}(n) - y_{\{j\}} \dots \dots \dots (a)$$

$$e_j(n) = d_j(n) - y_j \dots \dots \dots (b)$$

where  $d$  is the target value and  $y$  is the value produced by the perceptron.

The node weights are adjusted based on corrections that minimize the error in the entire output, given by equation 6 as,

$$\epsilon(n) = \frac{1}{2} \sum e_j^2(n) \dots \dots \dots (6)$$

Using gradient descent, the change in each weight is given in equation 7,

$$\Delta w_{ji}(n) = -\eta \frac{\partial \epsilon(n)}{\partial v_j(n)} y_i(n) \dots \dots \dots (7)$$

where  $y_i$  is the output of the previous neuron and  $\eta$  is the learning rate, which is selected to ensure that the weights quickly converge to a response, without oscillations. The derivative to be calculated depends on the induced local field  $v_j$ , which itself varies. It is easy to prove that for an output node this derivative can be simplified to

$$-\frac{\partial \epsilon(n)}{\partial v_j(n)} = e_j(n) \phi'(v_j(n)) \dots \dots \dots (8)$$

where  $\phi'$  is the derivative of the activation function described above, which itself does not vary. The analysis is more difficult for the change in weights to a hidden node, but it can be shown that the relevant derivative is given in equation 9,

$$-\frac{\partial \epsilon(n)}{\partial v_j(n)} = \phi'(v_j(n)) \sum_k -\frac{\partial \epsilon(n)}{\partial v_k(n)} w_{kj}(n) \dots \dots \dots (9)$$

This depends on the change in weights of the  $k$ th nodes, which represent the output layer. So to change the hidden layer weights, the output layer weights change according to the derivative of the activation function, and so this algorithm represents a backpropagation of the activation function (Jiawei Han, 2012)

## 6.1 RESNET-50 AS a Model

ResNet-50 is CNN architecture that belongs to the ResNet (Residual Networks) family, a series of models designed to address the challenges associated with training deep neural networks. Developed by researchers at Microsoft Research Asia, ResNet-50 is renowned for its depth and efficiency in image classification tasks. ResNet architectures come in various depths, such as

ResNet-18, ResNet-32, and so forth, with ResNet-50 being a mid-sized variant. ResNet-50 for currency detection is a practical application of deep learning in the field of image recognition. Here's a general guide on how we can use ResNet-50 for currency detection. The results of ResNet-50 were groundbreaking, achieving a 3.57% error rate on the ImageNet dataset and taking first place in several other competitions, including the ILSVRC and COCO object detection challenges (Kaiming He, 2016).

The vanishing gradient problem makes Deep learning neural networks hard to train. We use backpropagation to update the neural network weights using the chain rule of derivatives. The repeating multiplications will make the weights extremely small while reaching earlier layers (Kaiming He, 2016).

Residual networks use the concept of skip connections. Using these skip connections, we can resolve the problem of vanishing gradient (Kaiming He, 2016).

### **Skip Connections**

In a typical network, the convolutional layers are stacked one after the other. In skip connections, the traditional layers are stacked one after the other, but here we add the original input to the output of the convolutional block. This is called skip connection. Using this skip connection, the residual networks solve the vanishing gradient problem. Because some layers are skipped, the value will not reach a minimum value as we are missing some (Kaiming He, 2016).

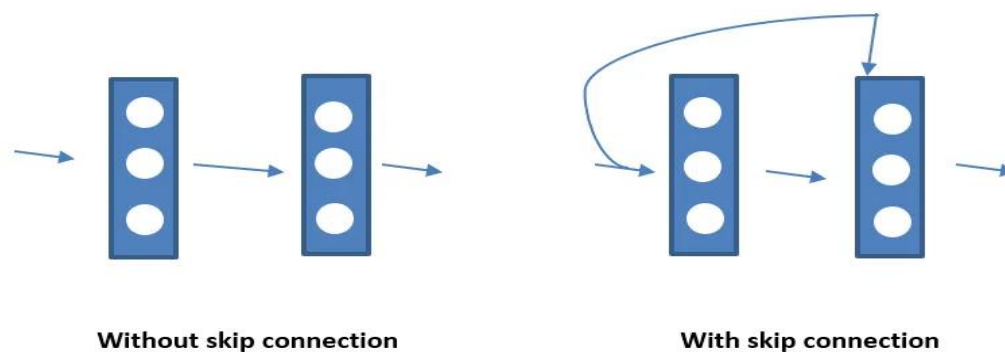


Figure 3.8: Skip Connection

## **Steps to Use ResNet-50 for Currency Detection**

### **Data Collection:**

**Images of Currency:** Gathered a diverse dataset of images of Nepali currencies to detect images taken under various lighting conditions and from different angles and rotations (Riddhi Sindhe, 2023).

**Annotations:** Labeled the images with the corresponding currency (Riddhi Sindhe, 2023).

### **Data Preprocessing:**

**Resize Images:** Resize images to the size expected by ResNet-50 i.e 224x224 pixels (Riddhi Sindhe, 2023).

**Normalization:** Normalize the pixel values to be in the range that ResNet-50 expects i.e [0, 1] (Riddhi Sindhe, 2023).

### **Model Selection:**

**Pre-trained ResNet-50:** Used a pre-trained ResNet-50 model. This model has been trained on a large dataset (ImageNet) and has learned useful features that can be transferred to your currency detection task (Riddhi Sindhe, 2023).

### **Transfer Learning:**

**Feature Extraction:** Removed the top layers of ResNet-50 and use the network as a fixed feature extractor by adding new fully connected layers on top of it for currency classification.

**Fine-Tuning:** Optionally, you can fine-tune some of the ResNet-50 layers (usually the later layers) along with the new fully connected layers (Riddhi Sindhe, 2023).

### **Training:**

**Compile Model:** Used an appropriate loss function (e.g., categorical cross-entropy) and optimizer (e.g., Adam).

**Train the Model:** Trained the model on RS 1000 currency dataset (Riddhi Sindhe, 2023).

### **Evaluation:**

**Validation:** Used a separate validation set to evaluate the model's performance.

**Metrics:** Monitored metrics like accuracy, precision, recall, and F1-score to understand performance of Model (Riddhi Sindhe, 2023).

**Deployment:**

**Export Model:** Exported the model for deployment.

**Inference:** Used the trained model to detect and classify currencies in new images (Riddhi Sindhe, 2023).

### Architecture of ResNet50

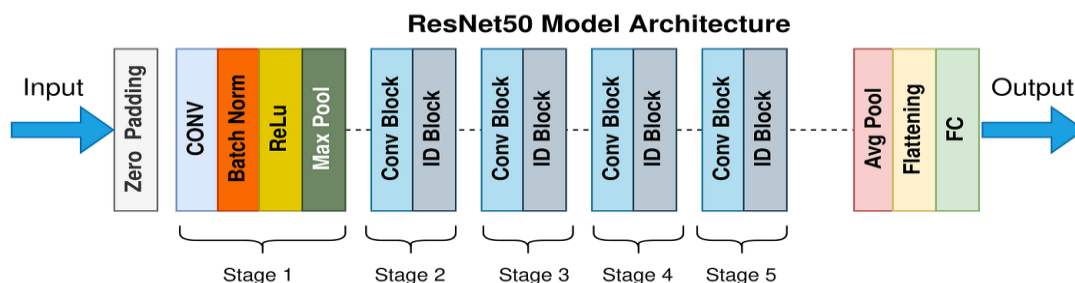


Figure 3.9: Architecture Of ResNet50 (Mukherjee, 2022).

ResNet-50 consists of 50 layers that are divided into 5 Stages, each containing a set of residual blocks. The residual blocks allow for the preservation of information from earlier layers, which helps the network to learn better representations of the input data (Mukherjee, 2022).

ResNet50 function

```
keras.applications.ResNet50(  
    include_top=True,  
    weights="imagenet",  
    input_tensor=None,  
    input_shape=None,  
    pooling=None,  
    classes=1000,  
    classifier_activation="softmax",  
)
```

Each Keras Application expects a specific kind of input preprocessing. For ResNet, call `keras.applications.resnet.preprocess_input` on your inputs before passing them to the model. `resnet.preprocess_input` will convert the input images from RGB to BGR, then will zero-center each color channel with respect to the ImageNet dataset, without scaling (Mukherjee, 2022).

## Arguments

- **include\_top:** whether to include the fully-connected layer at the top of the network (Narayan, 2024).
- **weights:** one of None (random initialization), "imagenet" (pre-training on ImageNet), or the path to the weights file to be loaded (Narayan, 2024).
- **input\_tensor:** optional Keras tensor (i.e. output of `layers.Input()`) to use as image input for the model (Narayan, 2024).
- **input\_shape:** optional shape tuple, only to be specified if `include_top` is False (otherwise the input shape has to be (224, 224, 3) (with "channels\_last" data format) or (3, 224, 224) (with "channels\_first" data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 32. E.g. (200, 200, 3) would be one valid value (Narayan, 2024).
- **pooling:** Optional pooling mode for feature extraction when `include_top` is False.
  - None means that the output of the model will be the 4D tensor output of the last convolutional block.
  - avg means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
  - max means that global max pooling will be applied (Narayan, 2024).
- **classes:** optional number of classes to classify images into, only to be specified if `include_top` is True, and if no weights argument is specified (Mukherjee, 2022).
- **Classifier\_activation:** A str or callable. The activation function to use on the "top" layer. Ignored unless `include_top=True`. Set `classifier_activation=None` to return the logits of the "top" layer. When loading pretrained weights, `classifier_activation` can only be None or "softmax" (Mukherjee, 2022).

**Zero-padding pads the input with a pad of (3, 3)**

### Stage-1

- The 2D Convolution has 64 filters of shape (7, 7) and uses a stride of (2, 2). Its name is “conv1”.
- BatchNorm is applied to the channels axis of the input.
- MaxPooling uses a (3, 3) window and a (2, 2) stride.

### Stage-2

- The convolutional block uses three sets of filters of size [64, 64, 256], “f” is 3, “s” is 1 and the block is “a”.
- The 2 identity blocks use three sets of filters of size [64, 64, 256], “f” is 3 and the blocks are “b” and “c”.

### Stage-3

- The convolutional block uses three sets of filters of size [128, 128, 512], “f” is 3, “s” is 2 and the block is “a”.
- The 3 identity blocks use three sets of filters of size [128, 128, 512], “f” is 3 and the blocks are “b”, “c”, and “d”.

### Stage-4

- The convolutional block uses three sets of filters of size [256, 256, 1024], “f” is 3, “s” is 2 and the block is “a”.
- The 5 identity blocks use three sets of filters of size [256, 256, 1024], “f” is 3 and the blocks are “b”, “c”, “d”, “e”, and “f”.

### Stage-5

- The convolutional block uses three sets of filters of size [512, 512, 2048], “f” is 3, “s” is 2 and the block is “a”.
- The 2 identity blocks use three sets of filters of size [512, 512, 2048], “f” is 3 and the blocks are “b” and “c”.

The 2D Average Pooling uses a window of shape (2, 2) and its name is “avg\_pool”.

The flatten doesn't have any hyperparameters or names.

The Fully Connected (Dense) layer reduces its input to the number of classes using a softmax activation.

Its name should be 'fc' + str(classes) (Mukherjee, 2022).

**The following are the main components of ResNET-50.**

### **1. Convolutional Layers**

The first layer of the network is a convolutional layer that performs convolution on the input image. This is followed by a max-pooling layer that down samples the output of the convolutional layer. The output of the max-pooling layer is then passed through a series of residual blocks (Kaiming He, 2016).

### **2. Residual Blocks**

Each residual block consists of two convolutional layers, each followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. The output of the second convolutional layer is then added to the input of the residual block, which is then passed through another ReLU activation function. The output of the residual block is then passed on to the next block (Kaiming He, 2016).

### **3. Fully Connected Layer**

The final layer of the network is a fully connected layer that takes the output of the last residual block and maps it to the output classes. The number of neurons in the fully connected layer is equal to the number of output classes (Kaiming He, 2016).

**The complete flow of ResNet-50 involves:**

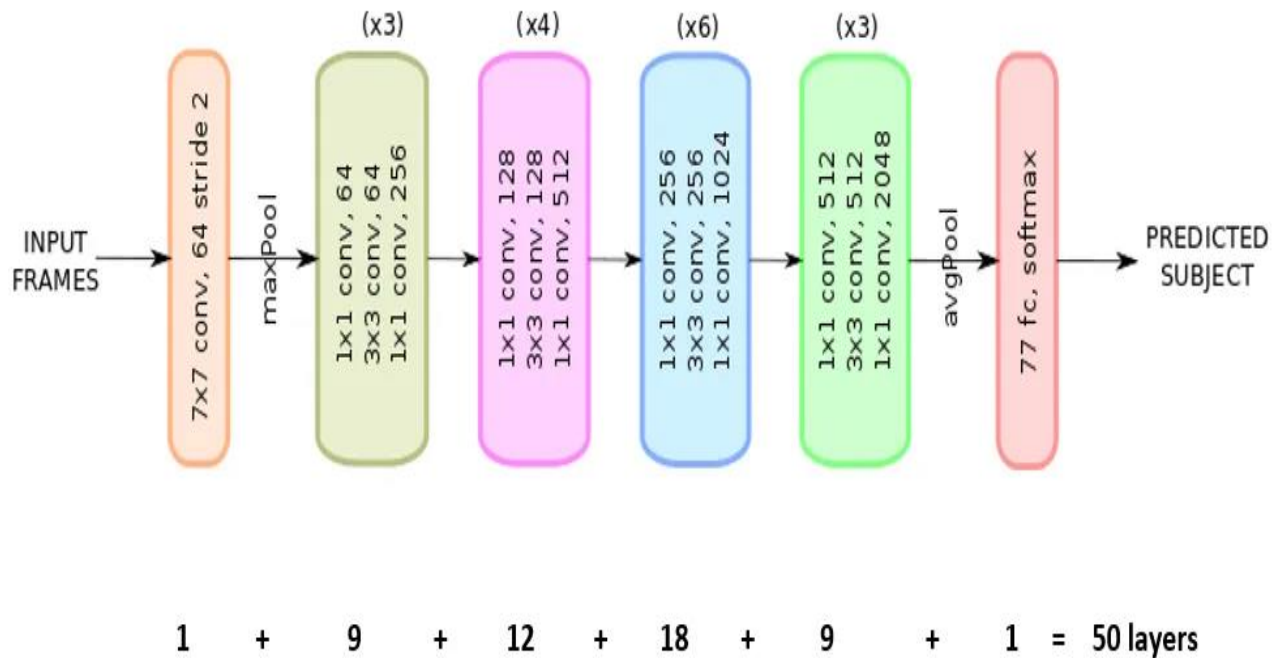


Figure 3.10: Flow of ResNet 50 (Mohammad Jahrohni, 2019)

- **Input Layer:** The journey begins with the input layer, where the raw image data is fed into the network (Mohammad Jahrohni, 2019).
- **Relu Activation:** It is an activation function commonly used in neural networks. It is a simple and effective way to introduce nonlinearity into the output of a neuron. The function is defined as:  $\text{ReLU}(x) = \max(0, x)$ . In other words, the output of the ReLU activation function is equal to the input if the input is positive, and 0 if the input is negative. This means that ReLU is a piecewise linear function, which makes it computationally efficient to compute and differentiate and able to prevent the vanishing gradient problem (Agrarap, 2019).



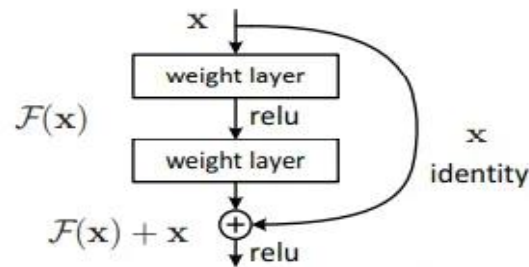


Figure 2. Residual learning: a building block.

- **Flatten layers:** The layer converts the output of the previous layer into a 1D vector, which is then fed into the fully connected layer (Agrarap, 2019).
- **Convolutional Layers:** Convolutional layers process the input image to detect features and generate feature maps (Nash, 2012).
- **Convolution and Residual Blocks:** Next are the convolutional blocks made up of multiple convolutional layers followed by residual blocks (Nash, 2012).
- **More Convolutional Blocks:** More convolutional blocks follow the residual blocks, refining features and learning intricate patterns (Nash, 2012).
- **Batch Normalization:** Batch normalization is typically applied after the convolutional or fully connected layers in a neural network but before the activation function. It is a widely used technique in deep learning and has been shown to improve the performance of many types of neural networks. Faster convergence, Improved generalization, and Regularization are some benefits of batch normalization (Sing, 2020).
- **Max Pooling:** Max pooling layers follow these convolutional layers. They down sample the spatial dimensions of the feature maps, retaining important information and reducing computational load. 3x3 max pooling operation with stride 2 and padding 1. So, the image size will be reduced to 75\*75 from 150\*150. We can see that the image size is reduced from 300\*300 to 75\*75 with 64 filters in each (Afia Zafar, 2022).
- **Global Average Pooling:** Instead of fully connected layers, ResNet typically employs global average pooling. Global average pooling reduces the spatial dimensions of the feature maps to a single value per feature, simplifying the architecture (Afia Zafar, 2022).

- **Min Pooling:** Min Pooling selects the minimum value from the pooling window. Though not as commonly used as max or average pooling, it can be useful in specific scenarios where the smallest feature values need to be retained (Afia Zafar, 2022).
- **Fully Connected Layer:** The final layer of the network is a fully connected layer that takes the output of the last residual block and maps it to the output classes. The number of neurons in the fully connected layer is equal to the number of output classes (Afia Zafar, 2022).
- **Output Layer:** The reduced feature maps undergo a final classification step in the output layer, producing the model's prediction (Mukherjee, 2022).

## Chapter 4: Feasibility Analysis

### 4.1. Technical Feasibility

This is complete web-based application. The main technologies and tools associated with this project are:

- **OpenCV**

OpenCV, short for Open Source Computer Vision Library, is a powerful toolkit designed for real-time computer vision applications. It provides a comprehensive set of algorithms for tasks like image and video processing, object detection, facial recognition, and machine learning. OpenCV's open-source nature, extensive documentation, and active community make it a popular choice for developers working on projects in robotics, self-driving cars, augmented reality, and more (Bradski, 2000).

- **Visual Studio Code**

Visual Studio Code (VS Code) is a popular source-code editor developed by Microsoft. It provides a lightweight and highly customizable environment for writing, debugging, and managing code across multiple programming languages. In normal terms, it facilitates users to write the code in an easy manner. Many people say that it is half of an IDE and an editor, but the decision is up to the coders. Any program/software that we see or use works on the code that runs in the background. Traditionally coding was used to do in the traditional editors or even in the basic editors like notepad. These editors used to provide basic support to the coders (Mustafeez, 2024).

- **Python**

Python is a high-level programming language known for its simplicity, readability, and versatility. It has gained popularity in various fields, including web development, data analysis, machine learning, and scientific computing. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed (Srinath, 2017).

## Python Libraries used in Project

### 1) **Library: NumPy**

Purpose: NumPy is a powerful numerical computing library in Python. It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. In the context of machine learning, it is often used for handling and manipulating data before feeding it into a model (Harris, 2020)..

Python Code:

```
import numpy as np
```

### 2) **Library: TensorFlow**

Purpose: TensorFlow is an open-source deep learning framework developed by Google. It is widely used for building and training neural networks. The tf alias is a common convention for importing TensorFlow (Agrawal, 2015).

Python Code:

```
import tensorflow as tf
```

### 3) **Library: imbalanced-learn (imblearn)**

Purpose: RandomOverSampler is used to handle imbalanced datasets by randomly oversampling the minority class to match the number of samples in the majority class. This is helpful in scenarios where the classes are not equally represented in the dataset (murphy, 2017).

### 4) **Library: Matplotlib**

Purpose: Matplotlib is a plotting library used for creating static, interactive, and animated visualizations in Python. plt is a common alias used for the pyplot module, which provides a MATLAB-like interface for plotting (Hunter, 2007).

### 5) **Library: Seaborn**

Purpose: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It is commonly used for visualizing the confusion matrix and other statistical plots (Waskom, 2021).

### 6) **Module: tensorflow.keras.applications**

Purpose: This module provides pre-trained models and architectures that are widely used in the deep learning community. ResNet50 is a specific architecture of a Residual Neural Network, which is 50 layers deep. It can be used for transfer learning and fine-tuning (Agrawal, 2015).

Python code

```
from tensorflow.keras.applications import ResNet50
```

#### 7) **Module: tensorflow.keras.preprocessing.image**

Purpose: ImageDataGenerator is used for data augmentation and preprocessing of images. It can generate batches of tensor image data with real-time data augmentation, which is useful for improving the robustness and performance of the model (Agrawal, 2015).

Python Code

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

#### 8) **Module: tensorflow.keras.layers**

Purpose: This module provides various layers that can be added to a Keras model.

- Dense: A fully connected layer, where each neuron is connected to every neuron in the previous layer (Agrawal, 2015).
- GlobalAveragePooling2D: A layer that performs global average pooling operation, reducing the spatial dimensions of the feature map to a single value per channel (Agrawal, 2015).

Python Code:

```
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
```

#### 9) **Module: tensorflow.keras.models**

Purpose: Model is the base class used for creating a Keras model with a functional API. It allows for the creation of complex models that have multiple inputs and outputs or non-sequential data flows (Agrawal, 2015).

Python Code:

```
from tensorflow.keras.models import Model
```

#### 10) **Module: tensorflow.keras.models**

Purpose: Sequential is a simpler model class that allows for the creation of models layer-by-layer in a linear stack. It is useful for straightforward, single-input, single-output models (Agrawal, 2015).

Python Code:

```
from tensorflow.keras.models import Sequential
```

#### 11) **Module: sklearn.metrics**

Purpose: This module provides functions to evaluate the performance of classification models.

- Classification\_report: Generates a text report showing the main classification metrics, including precision, recall, and F1-score (Pedregosa, 2011).
- Confusion\_matrix : Computes a confusion matrix to evaluate the accuracy of a classification.

Python Code:

```
from sklearn.metrics import classification_report, confusion_matrix
```

#### 12) **Module: tensorflow.keras.layers**

Purpose: Dropout is a regularization layer that helps prevent overfitting by randomly setting a fraction of input units to 0 at each update during training. This forces the network to learn more robust features (Agrawal, 2015).

Python Code:

```
from tensorflow.keras.layers import Dropout
```

#### 13) **Module: tensorflow.keras.callbacks**

Purpose: Callbacks are special functions that can be applied at different stages of training (e.g., at the end of each epoch).

- EarlyStopping: Monitors a chosen metric and stops training when it no longer improves.
- ModelCheckpoint: Saves the model after every epoch if the chosen metric improves, allowing you to keep the best version of your model (Agrawal, 2015).

Python Code:

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

#### 14) **Module: sklearn.utils**

Purpose: shuffle is used to shuffle arrays or sparse matrices in a consistent manner. It is useful for shuffling the data before splitting it into training and validation sets to ensure a random distribution (Pedregosa, 2011).

Python Code:

```
from sklearn.utils import shuffle
```

#### 15) **Library: OS**

Purpose: The os module in Python provides a way of using operating system-dependent functionality. It allows you to interact with the underlying operating system in a portable way. This includes tasks such as manipulating the file system, fetching environment variables, and handling paths (Python, 2019).

- **HTML**

HTML stands for Hypertext Markup Language. It is used to design the frontend portion of web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within the tag which defines the structure of web pages (Lee, 1993).

- **PyCharm**

PyCharm is a powerful tool for programmers who love Python. It's an IDE, or Integrated Development Environment, which means it bundles a bunch of helpful features into one program. PyCharm can analyze your code for errors, help you fix them, and even suggest ways to write better code. It also lets you run your programs and debug them if they're not working right. Plus, it integrates with other tools you might use, like version control systems, and even has special features for web development and data science. There's a free version of PyCharm that has a lot to offer, but there's also a paid version with even more bells and whistles (Qiang Hu, 2018).

- **ANACONDA**

Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment. Package versions in Anaconda are managed by the package management system, conda, which analyzes the current environment before executing an installation to avoid disrupting other frameworks and packages. The Anaconda distribution comes with over 250 packages automatically installed. Over 7500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI (graphical user interface), Anaconda Navigator, as a graphical alternative to the command line interface. Anaconda Navigator is included in the Anaconda distribution, and allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages, install them in an environment, run the packages and update them.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example TensorFlow, can find that it stops working after using pip to install a different package that requires a different version of the dependent NumPy library than the one used by TensorFlow. In some cases, the package may appear to work but produce different results in execution. In contrast, conda analyzes the current environment including everything currently installed, and together with any version limitations specified (e.g., the user may wish to have TensorFlow version 2.0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done (Matt Ross, 2020).

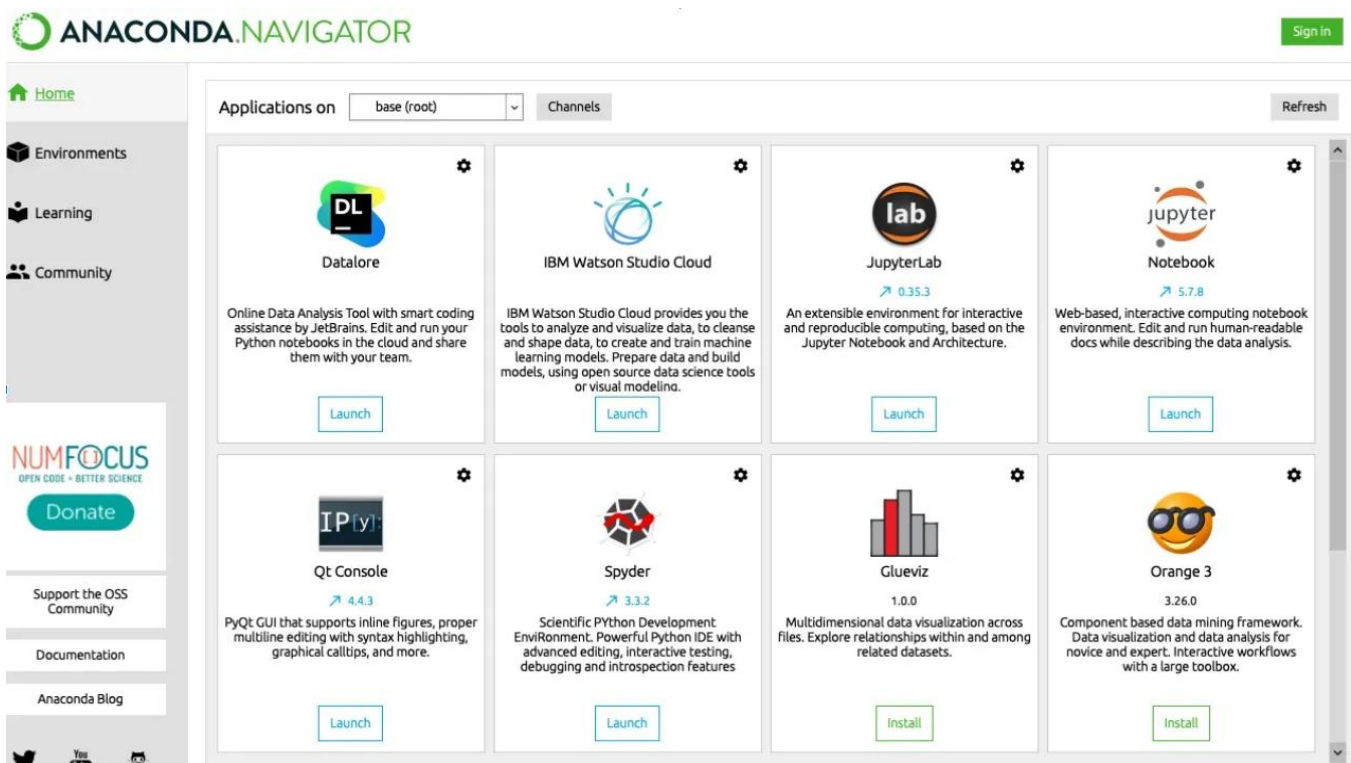


Figure 4.1: Anaconda navigator (Matt Ross, 2020)

- Diagrammatic Tools

Diagramming tools are software applications that help you create visual representations of information.

Each of the technologies mentioned above are freely available and technical skills required are manageable. Thus, this project is technically feasible.



## **4.2 Economic Feasibility**

As the project is web-based application, there will be hosting cost. Though system involves multimedia (image) data transfer, bandwidth required for operation of application is very low. The system will follow the freeware software standards. No cost will be charged from potential customer. Bug fixes, feature addition and maintaining tasks will have an associated cost which too will be low. So, the project is economically feasible.

## **4.3 Operational Feasibility**

The currency recognition process is widely carried out by different machines. Since the proposed system is web-based, anyone with browsing knowledge can use the application. Also, the application consists of image upload field and recognize button, it is easy to use. Thus, the proposed project is operationally feasible.

## CHAPTER 5: RESULT & CONCLUSION

### 5.1 Task Done

- **Model Selection**

The project is based on Image classification based on different parameters like color, edge etc. So, the Model Best prepared by Microsoft Research Team is ResNET-50 that capable for our requirement.

- **Datasets Collection and Preparation**

Nepali Currency Dataset is collected from Kaggle prepared by Gaurav Neupane for Training, Validation and Testing Purpose related 7 classes of currency(5, 10, 20, 50, 100, 500 and 1000).

Table 2: Data Collection(Group Work, 2024)

Currency Name(Class)	Train Number of data	Valid Number of data
Five	1499	501
Ten	3019	445
Twenty	2537	350
Fifty	2579	515
Hundred	2478	521
Five Hundred	2357	501
Thousand	2423	501

- Platform Selection: For Model preparation(Machine Learning) used Google Colab and Pycharm.
- Resnet-50 Model V2 integrated with System: We imported different Library and ResNet 50 V2 Model and in our system called Money Recognition System.

```

import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import RandomOverSampler
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.utils import shuffle

# For visualization
import matplotlib.pyplot as plt
import seaborn as sns

```

Figure 5.1: Library Import(Group Work, 2024)

- Training and Validation

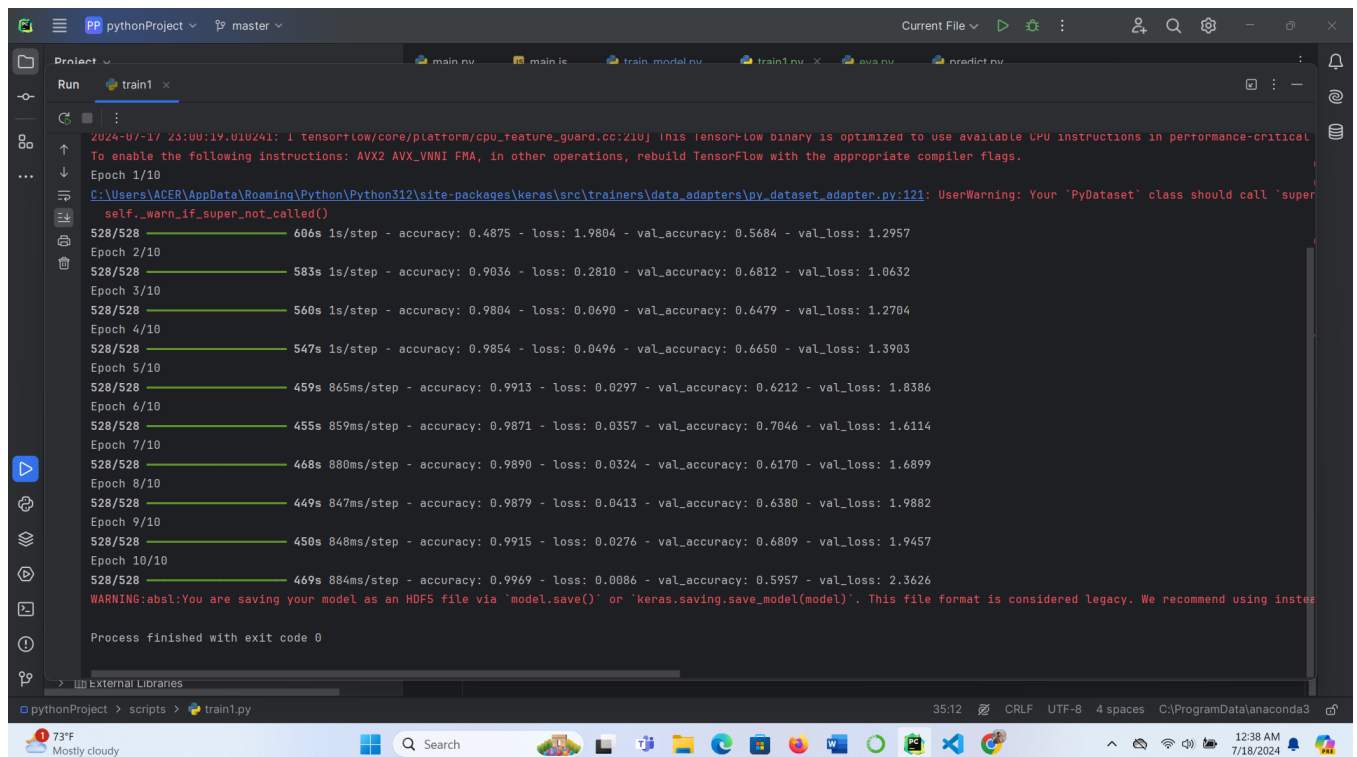
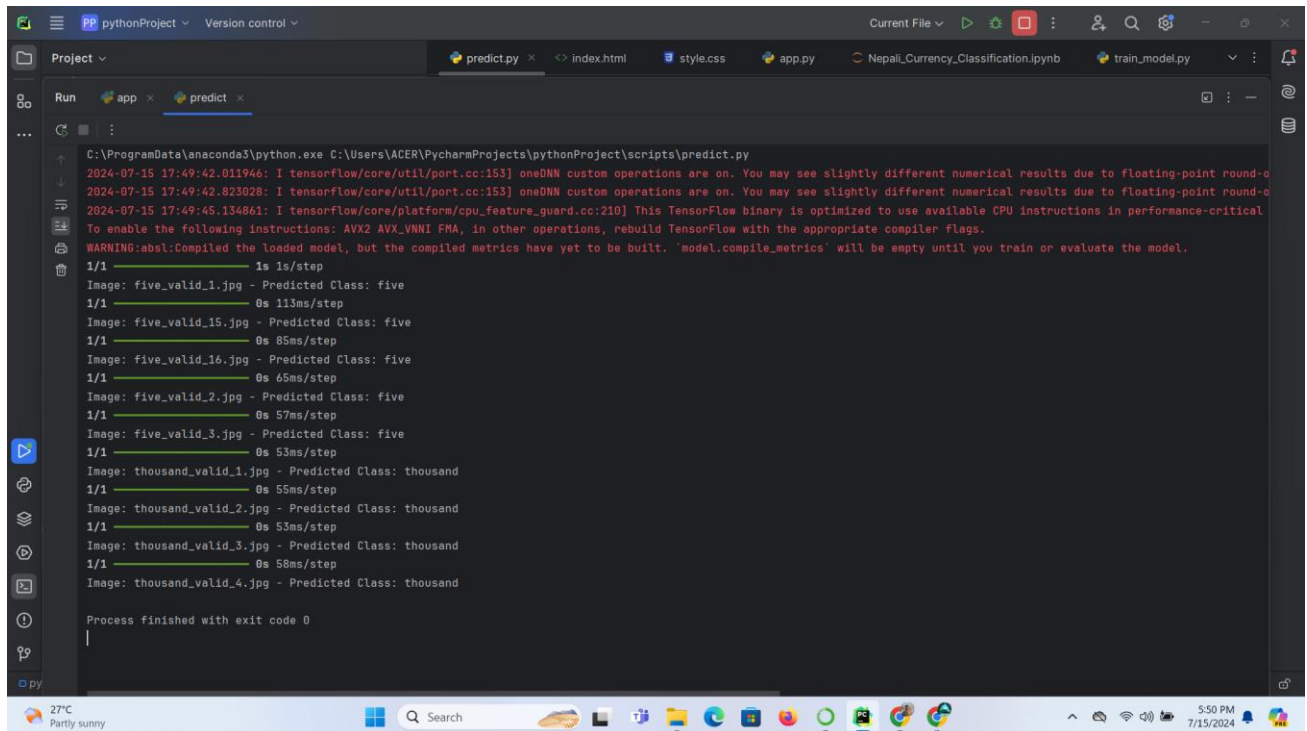


Figure 5.2: Training(Group Work, 2024)

- Testing and evaluation



```
C:\ProgramData\anaconda3\python.exe C:\Users\ACER\PycharmProjects\pythonProject\scripts\predict.py
2024-07-15 17:49:42.011946: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-o
2024-07-15 17:49:42.823028: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-o
2024-07-15 17:49:45.134861: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
1/1 ----- 1s 1s/step
Image: five_valid_1.jpg - Predicted Class: five
1/1 ----- 0s 113ms/step
Image: five_valid_15.jpg - Predicted Class: five
1/1 ----- 0s 85ms/step
Image: five_valid_16.jpg - Predicted Class: five
1/1 ----- 0s 65ms/step
Image: five_valid_2.jpg - Predicted Class: five
1/1 ----- 0s 57ms/step
Image: five_valid_3.jpg - Predicted Class: five
1/1 ----- 0s 53ms/step
Image: thousand_valid_1.jpg - Predicted Class: thousand
1/1 ----- 0s 55ms/step
Image: thousand_valid_2.jpg - Predicted Class: thousand
1/1 ----- 0s 53ms/step
Image: thousand_valid_3.jpg - Predicted Class: thousand
1/1 ----- 0s 58ms/step
Image: thousand_valid_4.jpg - Predicted Class: thousand

Process finished with exit code 0
```

Figure 5.3: Testing(Group Work, 2024)

```
Run app x evaluate_model x
Found 3334 images belonging to 7 classes.
C:\ProgramData\anaconda3\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your 'PyDataset' class should call 'super().__init__()'
self._warn_if_super_not_called()
105/105 ----- 122s 1s/step - accuracy: 0.8692 - loss: 0.9689
Test Loss: 1.279980182647705
Test Accuracy: 0.8653269410133362
105/105 ----- 115s 1s/step
Confusion Matrix:
[[392  0  1 114  5  2  1]
 [ 1497  2  0  1  0  0]
 [  0  3491  4  0  3  0]
 [  6  0  0472 17 10 16]
 [ 93  1 13 45269 14 10]
 [  0  0  9  7 1484  0]
 [  6  3  1 58  0 2280]]
Classification Report:
      precision    recall  f1-score   support

   fifty         0.79      0.76      0.77        515
     five         0.99      0.99      0.99        501
 fivehundred      0.95      0.98      0.96        501
   hundred      0.67      0.91      0.77        521
       ten      0.92      0.60      0.73        445
 thousand      0.94      0.97      0.95        501
   twenty      0.91      0.80      0.85        350

   accuracy              0.87        3334
  macro avg              0.88      0.86      0.86        3334
 weighted avg              0.88      0.87      0.86        3334

Process finished with exit code 0
```

Figure 5.4: Evaluation(Group Work, 2024)

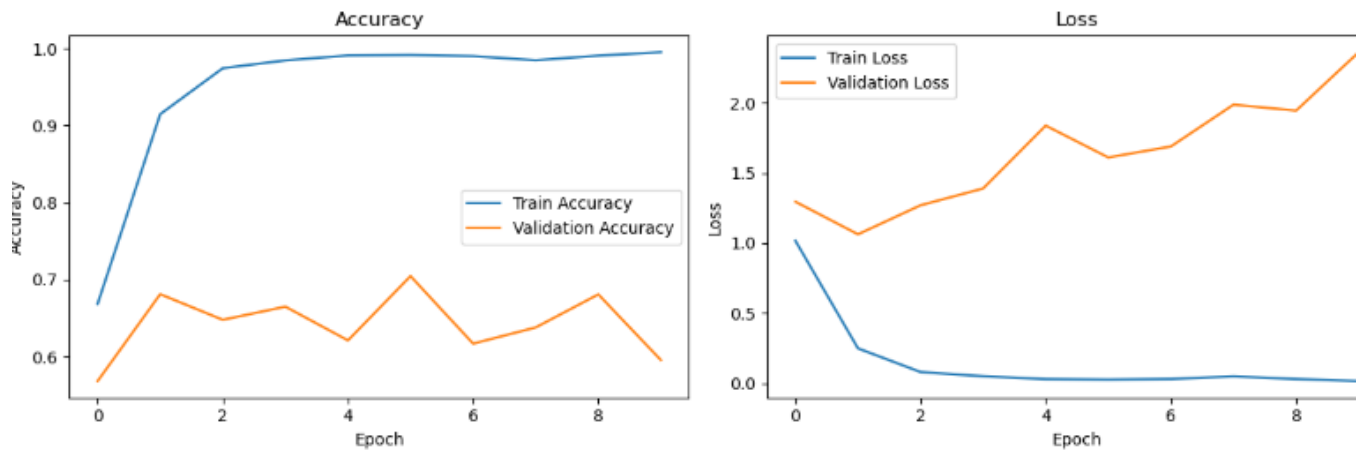


Figure 5.5: Graph for Accuracy and Loss(Group Work, 2024)



Figure 5.6: Confusion Matrix(Group Work, 2024)

## 5.2 Web App Development

### 1. Home Page(User Interface Upload Page)

Flask API is used to connect with system and user-interface. Users can upload image in this page and then classify option will appear.

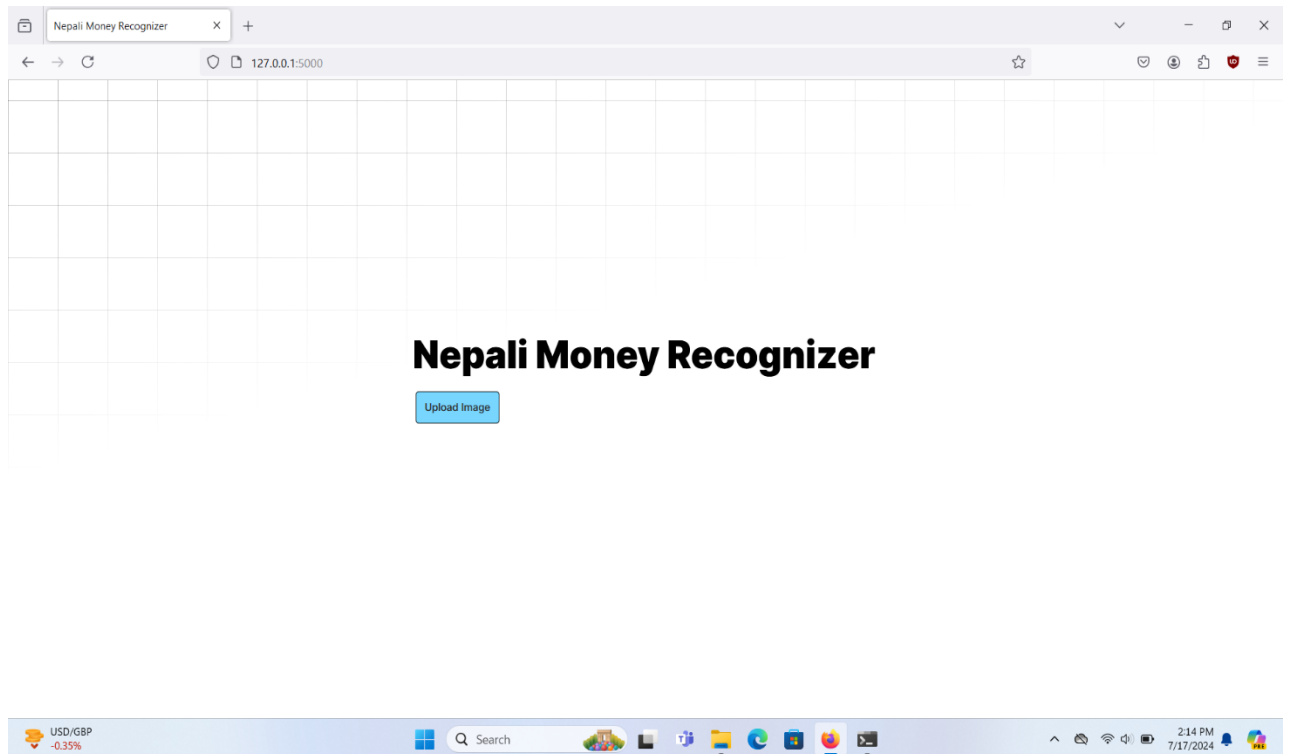


Figure 5.7: Home page(Group Work, 2024)

### 2. After Upload

After uploaded image by users, classify option appeared and then users can click on classify option to check currency value.



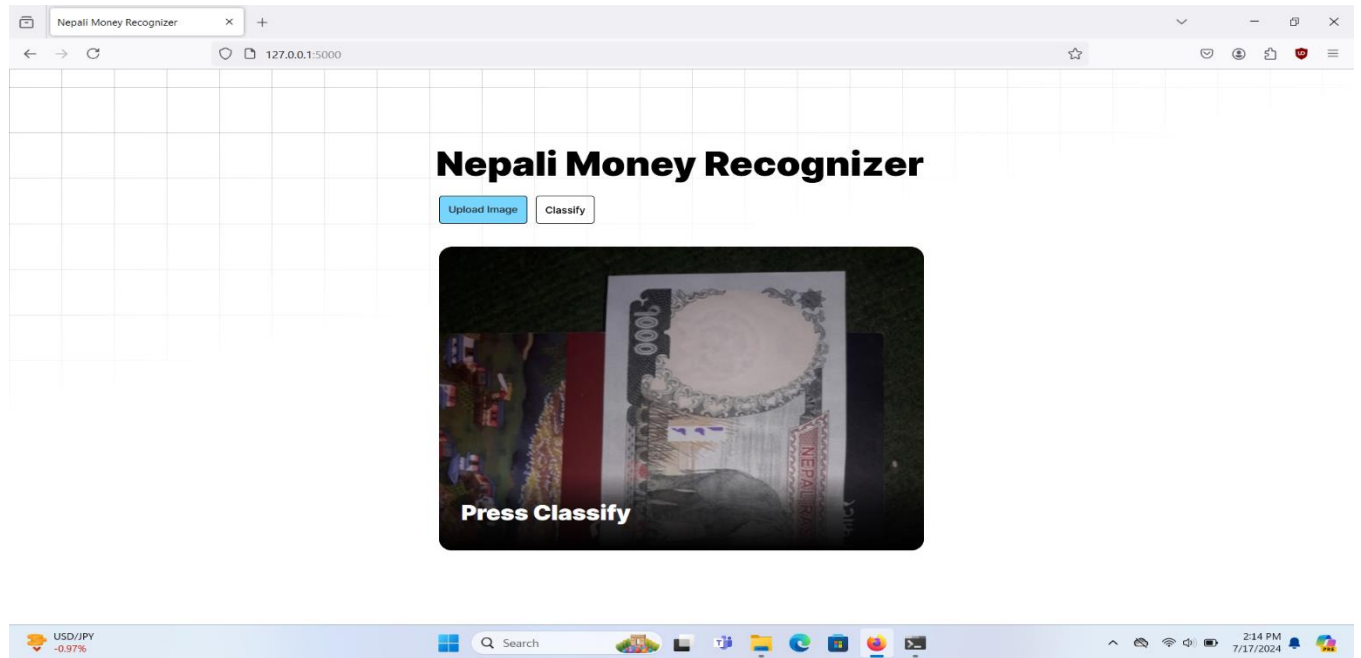


Figure 5.8: After Upload(Group Work, 2024)

### 3. After Classify

After click on Classify option, Currency value appeared.

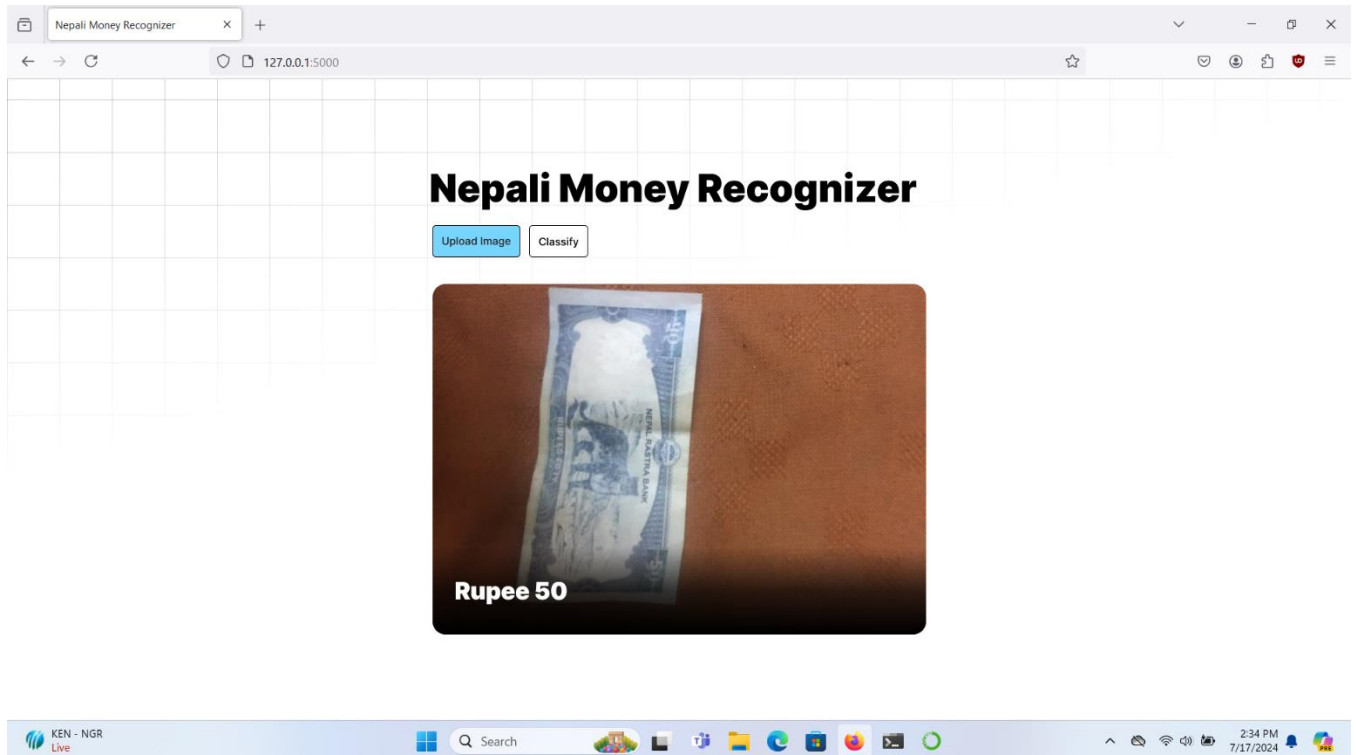


Figure 5.9: After Classify-1(Group Work, 2024)

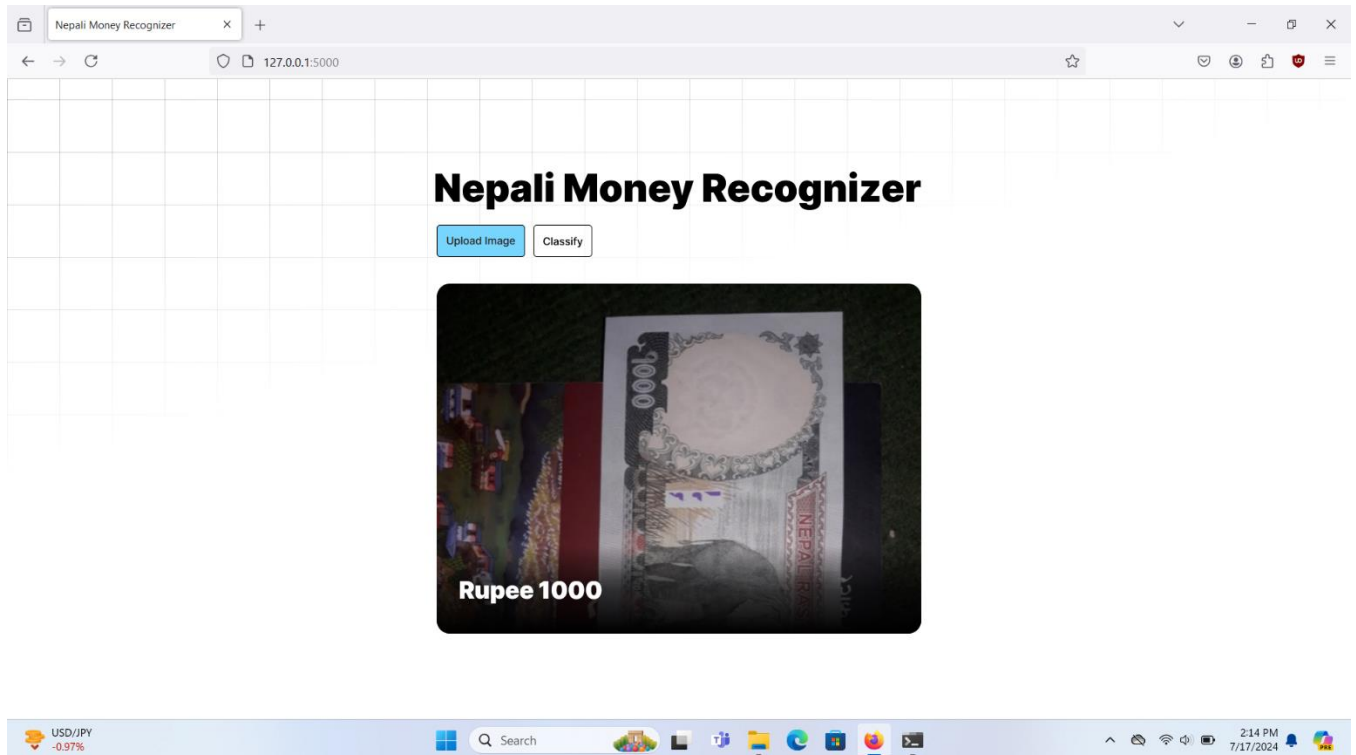


Figure 5.10: After Classify -2(*Group Work, 2024*)

## **CHAPTER 6: CONCLUSION AND FUTURE ENHANCEMENT**

### **6.1 Conclusion**

The Nepali Money Recognition System using Multilayer Perceptron has successfully demonstrated its capability to recognize Nepalese paper currency through a web-based platform. The project integrated advanced image processing techniques such as edge detection, color analysis, texture recognition, and optical character recognition (OCR) to provide accurate and reliable currency identification. By incorporating the ResNet-50 V2 model, the system achieved 99.69% accuracy in distinguishing various denominations(5, 10, 20, 50, 100, 500 and 1000) of Nepalese currency. This comprehensive approach highlights the potential of leveraging Deep learning(Resnet50 V2) and image processing technologies to address practical problems, ensuring a robust and effective solution for currency recognition.

Throughout the development process, significant emphasis was placed on data collection and preparation, model training, validation, and deployment. These steps were crucial in creating a system that not only performs well in controlled environments but also maintains its accuracy and reliability in real-world scenarios. The project underscored the importance of a detailed and methodical approach to developing Deep learning applications, demonstrating how thorough preparation and rigorous testing can lead to a highly functional end product. By addressing the technical, economic, and operational feasibility, the project ensured that the solution is not only effective but also practical and accessible for a wide range of users.

In conclusion, the Nepali Money Recognition System represents a significant advancement in the field of currency recognition, offering a practical tool that can greatly benefit various sectors, including financial institutions, businesses, and individuals. The project's success opens up opportunities for future enhancements, such as real-time recognition via mobile applications, counterfeit detection, and multilingual support. These improvements would further increase the system's versatility and usability, making it an even more valuable asset for everyday currency handling needs. By continuing to refine and expand upon this foundation, the system can adapt to evolving challenges and user requirements, solidifying its role as a critical tool in the efficient management of Nepalese currency.

## 6.2 Future Enhancement

For future enhancements, the Nepali Money Recognition System using Multilayer Perceptron can be expanded in several ways to improve its functionality and user experience:

1. **Extensive Datasets:** Incorporate a more extensive dataset to include various conditions of currency notes, such as worn-out or damaged notes, to enhance robustness.
2. **Real-Time Recognition:** Integrate a real-time recognition feature using mobile applications to significantly increase accessibility. This would allow users to identify currency on the go without needing to upload images through a web interface.
3. **Counterfeit Detection:** Train the system to recognize counterfeit notes by including more sophisticated detection algorithms and security feature analysis.
4. **Multilingual Support:** Add multilingual support to assist non-Nepali speakers in understanding the currency denominations.
5. **Cloud-Based Services:** Leverage cloud-based services for processing and storing data to improve the system's performance and scalability.
6. **User Feedback Mechanisms:** Incorporate user feedback mechanisms to help continually refine the system based on real-world usage

These enhancements will make the system more versatile, user-friendly, and secure, meeting the evolving needs of its users.

## References

1. Afia Zafar, S. R. (2022). A Comparasion of Pooling Methods for Convolution Neural Networks. *Computng and qartificial Intelligence*, 8643. doi: <https://doi.org/10.3390/app12178643>
2. Agrarap, A. f. (2019). Deep Learning using Rectified Linear Units(RELU).
3. Agrawal, A. H. (2015). *Large-Scale Machine Learning on Heterogeneous Systems*. doi:10.5281/zenodo.4724125
4. Ami Shah, K. V. (2015). A Review Paper on Currency Recognition System. *nternational Journal of Computer Applications*, 115(20), 1-4. doi:10.5120/20264-2669
5. Bahrani, S. M. (2020). DEEP LEARNING APPROACH FOR INDIAN CURRENCY CLASSIFICATION. *International Journal of Engineering Applied Sciences and Technology*, 5(6), 335-340.
6. Bradski, G. (2000). The Opencv Library. *Dr. Dobb's Journal of Software Tools*.
7. Ch. Ratna Jyothi, Y. S. (2016). Paper currency recognition for color images based on Artificial Neural Network. *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, (pp. 3402-3406). doi: 10.1109/ICEEOT.2016.7755335
8. Endang Suherman, B. R. (2023).
9. Fukumi, M. a. (2000). Design and evaluation of neural networks for coin recognition by using GA and SA. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 5, pp. 178-183. doi:10.1109/IJCNN.2000.861454
10. Gedraite, E. (2011). Investigation on the effect of a Gaussian Blur in image Filtring and Segmentation.
11. Harris, T. r. (2020). Array programming with NumPy. *Nature*. *Nature*, 585, 357-362. doi:10.1038/s41586-020-2649-2
12. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9, 90-95. doi:10.1109/MCSE.2007.55
13. Jiawei Han, M. K. (2012). Classification: Advanced Methods. In M. K. Jiawei Han, *Data Mining* (Third Edition ed., pp. 393-442). doi:<https://doi.org/10.1016/B978-0-12-381479-1.00009-5>

14. Kaiming He, X. Z. (2016). Deep Residual Learning for Image Recognition. *Microsoft Research*.
15. KEA Van De Sande, T. G. (2008). Evaluating color descriptors for object and scene recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 32, 32, pp. 1-8. doi:10.1109/CVPR.2008.4587658
16. Lee, T. B. (1993). HTML.
17. Matt Ross, K. C.-m. (2020). Introduction to Ananconda and Python: Installation and setup. *The Quantitative Mehods for Pyschology*, 16. doi:10.20982/tqmp.16.5.S003
18. Mohammad Jahrohni, P. B.-C. (2019). Privacy Constrained Biometric System for Non Cooperative Users. *Entropy*, 21, 1033. doi:10.3390/e21111033
19. Mukherjee, S. (2022). Resnet50.
20. murphy, K. e. (2017). Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *The journal of Machine Learning Research*.
21. Mustafeez, A. Z. (2024). *What is Visual Studio Code?* Retrieved from Educative: <https://www.educative.io/answers/what-is-visual-studio-code>
22. Nanda, R. M. (2012). Desing and Implementation of Indian paper Currency Authentication Sysytem Based on Feature Extraction by Edge Based Segmentation Using Sobel Operator. *International Journal of Engineering research and Development*, 3(2), 41-46. Retrieved from <https://www.ijerd.com/paper/vol3-issue2/F03024146.pdf>
23. Narayan, G. D. (2024). Enhancement of Image Processing based on Deeep Learning Backpropagation approch. *international Journal of Intelligent systems and application in Engineering*, 12, 9. Retrieved from <https://ijisae.org/index.php/IJISAE/article/view/4198>
24. Nash, K. O. (2012). An Introduction to Convoltion Neural Network. *Neural and Evolutionary Computing*, 2. Retrieved from <https://arxiv.org/abs/1511.08458>
25. Pedregosa, F. a. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 2825-2830.
26. Python. (2019). *Welcome to Python.org*. . Retrieved May 23, 2023, from <https://www.python.org/>
27. Qiang Hu, L. M. (2018). DeepGraph: A Pycharm tool for Visualizing and Understanding Deep Learning Models. *5th Asia-Pacific Software Engineering Conference (APSEC)*. doi:10.1109/APSEC.2018.00079

28. Riddhi Sindhe, A. T. (2023). Indian Currency Fake Note Detection System Using Resnet 50. *International Journal of Innovative Science and Research Technology*, 8(4).
29. Rosebrock, A. (2021). *Adaptive Thresholding with OpenCV*. Retrieved from pyimagesearch: <https://pyimagesearch.com/2021/05/12/adaptive-thresholding-with-opencv-cv2-adaptivethreshold/>
30. Sarfraz, M. (2015). An Intelligent Paper Currency Recognition System. *Procedia Computer Science*, 65, 538-545. doi:<https://doi.org/10.1016/j.procs.2015.09.128>
31. Shakya, S. L. (2015). Counterfeit Paper Banknote Identification Based on Color and Texture. *Proceedings of IOE Graduate Conference*, (pp. 160-168). Retrieved from <http://conference.ioe.edu.np/ioegc2015/papers/IOEGC-2015-022.pdf>
32. Sharda, A. (2021). *Image Filters: Gaussian Blur*. Retrieved from Medium: <https://aryamansharda.medium.com/image-filters-gaussian-blur-eb36db6781b1#:~:text=TLDR%3A%20A%20Gaussian%20blur%20is,values%20for%20the%20blurred%20image.>
33. Sing, D. S. (2020). Investigation the impact of data normalization on classification performance. *Applied Soft Computing*, 97. doi:<https://doi.org/10.1016/j.asoc.2019.105524>
34. Srinath, K. R. (2017). Python-The fastest Growing Programming Language. *International Research Journal of Engineering and Technology (IRJET)*, 4(12).
35. Waldo, J. (2022). A Comparative study of back propagation and its alternatives on multilayer perceptron. *Neural and Evolutionary Computing*. doi:<https://arxiv.org/abs/2206.06098>
36. Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6, 3021. doi:10.21105/joss.03021

## Appendix

The System Code Is Available at [Github](#)



