Industry Oriented Mini Project Report

on

FAKE CURRENCY DETECTION SYSTEM USING DEEP LEARNING

Submitted in partial fulfillment of the requirements
for the award of degree of

BACHELOR OF TECHNOLOGY

in

Information Technology

by

D. Tejaswini (20WH1A1291)

B. Akshitha (20WH1A12A0)

Ch. Dharani (20WH1A12B3)

Under the esteemed guidance of

Mr. A. Rajashekar Reddy

Assistant Professor

Department of Information Technology

BVRIT HYDERABAD College of Engineering for Women

Rajiv Gandhi Nagar, Nizampet Road, Bachupally, Hyd–500090

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

(NAAC 'A' Grade & NBA Accredited- ECE, EEE, CSE & IT)

Dec, 2023

BVRIT HYDERABAD

College of Engineering for Women

Rajiv Gandhi Nagar, Nizampet Road, Bachupally, Hyderabad – 500090
(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
(NAAC 'A' Grade & NBA Accredited- ECE, EEE, CSE & IT)

CERTIFICATE

This is to certify that the Project report on **" FAKE CURRENCY DETECTION SYSTEM USING DEEP LEARNING "** is a bonafide work carried out by **D. Tejaswini (20WH1A1291), B. Akshitha (20WH1A12A0)** and **Ch. Dharani (20WH1A12B3)** in the partial fulfillment for the award of B.Tech degree in **Information Technology , BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad** affiliated to Jawaharlal Nehru Technological University, Hyderabad, under my guidance and supervision. The results embodied in the project work have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide
Mr. A. Rajashekar Reddy
Assistant Professor
Department of IT

Head of the Department
Dr. Aruna Rao S L
Professor & HoD
Department of IT

External Examiner

DECLARATION

We hereby declare that the work presented in this project entitled **"FAKE CURRENCY DETECTION SYSTEM USING DEEP LEARNING"** submitted towards completion of in IV year I sem of B.Tech IT at "BVRIT HYDERABAD College of Engineering for Women",Hyderabad is an authentic record of our original work carried out under the esteemed guidance of **Mr. A. Rajashekar Reddy , Assistant Professor**, Department of Information Technology.

D. Tejaswini (20WH1A1291)

B. Akshitha (20WH1A12A0)

Ch. Dharani (20WH1A12B3)

This project report is dedicated to my beloved Family members and supervisor for their limitless support and encouragement and to you as a reader

ACKNOWLEDGMENTS

We would like to express our profound gratitude and thanks to **Dr. K. V. N. Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women** for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. Aruna Rao S L, Professor & Head, Department of IT, BVRIT HYDERABAD College of Engineering for Women** for all the timely support, constant guidance and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr.A. Rajashekar Reddy, Assistant Professor, Department of IT, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinators **Mr. Ch. Anil Kumar, Assistant Professor and Mr. Nemalikanti Anand, Assistant Professor**, all the faculty and staff of Department of IT who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

D. Tejaswini (20WH1A191)

B. Akshitha (20WH1A12A0)

Ch. Dharani (20WH1A12B3)

ABSTRACT

Fake currency Detection is a critical aspect of maintaining the integrity and stability of financial systems worldwide. With the advancement of color printing technology, counterfeiters have become increasingly sophisticated in producing fake currency that closely resembles genuine banknotes. As a result, there is a need for robust and reliable methods that differentiate between fake and genuine currency has grown significantly. Most of the former methods are based on hardware and image processing techniques. Finding counterfeit currencies with these methods is less efficient and time consuming. To overcome this problem, modern technological advancements in image processing, machine learning, and computer vision are used. These techniques involve the extraction of various features from currency images, including texture, color, and pattern information. Machine learning algorithms, such as Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and ensemble methods, are trained on a dataset of genuine and counterfeit currency images to learn differentiate patterns. Consequently, every person will have the capability to distinguish between fake and genuine currency independently.

Keywords: KNN, SVM, CNN.

Contents

List of Figures

Chapter 1

Introduction

The rapid evolution of technology has prompted advancements in the banking sector, necessitating the implementation of automatic fake currency detection in machines such as ATMs and automated sellers. Researchers are actively developing efficient currency detection machines capable of identifying both visible and invisible features of banknotes. Despite various proposed techniques, focusing on visible attributes like color and size proves challenging for dirty or torn notes. The Reserve Bank of India, with exclusive authority to issue banknotes, employs anti-counterfeiting measures like intricate designs and raised intaglio printing. Traditional methods of detecting counterfeit currency involved manual inspection, a time-consuming and error-prone process. However, with the advent of automated detection systems, the identification process has become faster, more efficient, and less susceptible to human error. These systems contribute significantly to the prevention of counterfeit currency circulation, thereby safeguarding the financial systems of nations and institutions.Counterfeiting, the production or use of imitation currency without legal sanction, is as old as money itself. The historical prevalence of counterfeiting involved methods like mixing base metals with precious metals. In modern times, counterfeit money negatively impacts society by reducing the value of real currency, causing inflation, decreasing paper money acceptability, and resulting in financial losses. The increase in counterfeit circulation has led to corruption and hindered a country's growth. Consequently, the development of robust automatic currency detection systems becomes crucial in addressing these challenges and ensuring the integrity of financial transactions.

Table VIII.9: Denomination-wise Counterfeit Notes Detected in the Banking System (April to March)

(Number of pieces)

Denomination (₹)	2020-21	2021-22	2022-23
1	2	3	4
2 and 5	9	1	3
10	304	354	313
20	267	311	337
50	24,802	17,696	17,755
100	1,10,736	92,237	78,699
200	24,245	27,074	27,258
500 (Specified Banknotes)	9	14	6
500	39,453	79,669	91,110
1000 (Specified Banknotes)	2	11	482
2000	8,798	13,604	9,806
Total	**2,08,625**	**2,30,971**	**2,25,769**

Source: RBI.

Figure 1.1: Fake Currency Production

1.1 Motivation

As genuine currency plays vital role in maintaining the stability and credibility of financial systems, fostering economic growth, and upholding public confidence in the authenticity of monetary transactions.It is important to safeguard individuals money and address the issue of producing counterfeit currency among citizens is needed. The motivation behind the implementation of a Fake Currency Detection System extends to preventing economic losses for both individuals and businesses by detecting and removing counterfeit notes from circulation. Preserving public trust in the monetary system is crucial; failure to detect counterfeit currency can erode confidence in currency authenticity, leading to a loss of trust in financial institutions the system aims to protect businesses and individuals from the adverse impact of counterfeit money, ensuring regulatory compliance and safeguarding institutions from legal consequences. By minimizing fraud and associated criminal activities, particularly money laundering and illicit transactions, the system contributes to national security.These system play a crucial role in uphold-

ing the reliability and security of monetary transactions in an evolving and interconnected world.

1.2 Objective

The primary objectives of a Fake Currency Detection System are centered around preserving the financial integrity of systems by reducing the circulation of counterfeit currency, a threat to currency stability and credibility. The system strives to prevent economic losses for individuals and businesses by accurately identifying and removing counterfeit notes, thereby safeguarding the value of genuine currency. The system protects businesses and individuals from financial losses and legal consequences resulting from inadvertent dealings with counterfeit money, ensuring regulatory compliance. By minimizing fraud and associated criminal activities, enhancing transaction efficiency, and contributing to national security, the system creates a secure and trustworthy environment for monetary transactions.

1.3 Problem Statement

The persistent threat of counterfeit currency in financial systems demands the development of a robust Fake Currency Detection System. The evolution of increasingly sophisticated counterfeiting techniques, coupled with the widespread use of automated transactions, underscores the critical need for advanced detection mechanisms. Existing systems encounter challenges in accurately distinguishing between genuine and counterfeit banknotes, particularly in the face of sophisticated methods like digital manipulation and replication. Traditional approaches struggle to adapt to the dynamic landscape of counterfeit techniques, posing risks to financial institutions, businesses, and individuals. This problem statement calls for the design and implementation of an effective Fake Currency Detection System that addresses the complexity of modern counterfeiting. The envisioned system should ensure real-time and reliable detection across diverse automated transaction platforms, including ATMs and vending machines. Overcoming challenges such as variations in note conditions, ensuring regulatory compliance, and staying ahead of innovative counterfeit tactics is essential. The ultimate goal is to create a comprehensive solution that safeguards financial integrity, prevents economic losses, and upholds public trust in the face of an ever-evolving landscape of counterfeit currency threats.

Chapter 2

Literature Survey

[1] Sruthi R. used various methods for detecting fake currency notes with machine learning and image processing techniques. A transfer learned convolutional neural network has been used achieving 81.5% accuracy for identifying real notes and 75% for counterfeit notes. An ensemble of classifiers has been used, with SVM (support vector machine) achieving the highest accuracy of 82.7%. Edge detection techniques have been used , achieving an accuracy of 90.45%. Different preconfigured CNNs (convolutional neural network) have been used. KNN (K-Nearest Neighbors Algorithm), SVC (support vector machine), and GBC (Genetic Bee Colony) have been used , with KNN (KNearest Neighbors Algorithm) and GBC (Genetic Bee Colony) providing higher accuracy in the recognition task.

[2] Pallavi, S. et al. detect and identify banknotes of different denominations. This study presented a system that use convolutional neural networks with deep training using CNN (Convolutional Neural Network) model training, currency pre-processing algorithm. By acting as a feature extractor, their ML-CNN (Machine Learning Convolutional Neural Network) based project eliminated the need for image processing and the need for manually confirming the existence of security characteristics in the note. The project's audio output, which will also be useful to blind people, will be its final product. Further experiments with different ML-CNN (Machine Learning Convolutional Neural Network) architectures will increase the model's accuracy

[3] Colaco, Rencita Maria et al. chose to use the programming language Python and OpenCV for their project using Canny Edge Detection algorithm. To make comparisons and determine the outcome, a number of characteristics that define genuine currency apart from counterfeit ones were taken into account. Identification marks, see-through registers, optical vari-

able ink, currency color codes, security threads, watermarks, latent images, and micro-lettering were a few of these features. The accuracy of the proposed system was close to 80%. The goal of this research was to create a low-cost system with quick computations so that even the average person, who is unable to use more advanced resources, can identify counterfeit money.

[4] Jamkhandikar, Dayanand et al.used two different currencies, and it was discovered that the suggested method based on color and feature analysis is effective for currencies. This project was done using image processing algorithm. The accuracy found in this system was 70%. Doing this project allowed them to identify counterfeit money, which is particularly helpful in preventing high-order counterfeiting that makes use of low-cost but highquality machinery.

[5] Sangogi, Mrs Jyoti et al. made it possible for someone who is blind to tell whether money is real or phony. Based on the parameters of the HSV values of the currency note, the Python technique, which was implemented in a Raspberry Pi with a scanner, could capture the currency note and carried out the image processing techniques mandated in the project to determine whether the currency is genuine or counterfeit.

[6] Kumar, Akhila et al. used three supervised machine learning (SML) methods were used in this project to authenticate banknotes. The three methods that were used are the Decision Tree (DT), KNN (K-Nearest Neighbors Algorithm), and SVM (Support Vector Machine) algorithms

[7] Aditya Sharma, Shweta Poojary et al. worked using the help of the Tensorflow and Keras libraries, this system primarily focused on the picture categorization part of deep learning using deep learning network algorithms, deep neural network algorithms, and convolutional neural network algorithms. The detection of currency tooks only a few seconds, and currency recognition was simple. The system has an excellent overall accuracy level for differentiating between real and fraudulent cash

[8] Kudalkar et al. proposed a technique which was appeared to be effective in determining whether the currency is real or fake using image acquisition, noise removal, gray scale conversion, edge detection, segmentation, feature extraction, comparison, supervised learning algorithm. According to the project's findings, the primary security features—bleed lines, security thread, and micro lettering—were needed to be precisely computed

[9] Vidhate, Shah et al. examined all relevant existing architectures, and by studying how they functioned, they found certain problems with the system that was in place at the time. With some of the extra features in their suggested system being implemented utilizing KNN (K-Nearest Neighbors Algorithm) algorithms or sophisticated machine learning techniques, they had preserved all the key aspects of the existing systems as their primary focus.

[10] Kumar, Akhila et al. used three supervised machine learning (SML) methods were used in this project to authenticate banknotes. The three methods that were used are the Decision Tree (DT), KNN (K-Nearest Neighbors Algorithm), and SVM (Support Vector Machine) algorithms

Chapter 3

System Design

3.1 Architecture

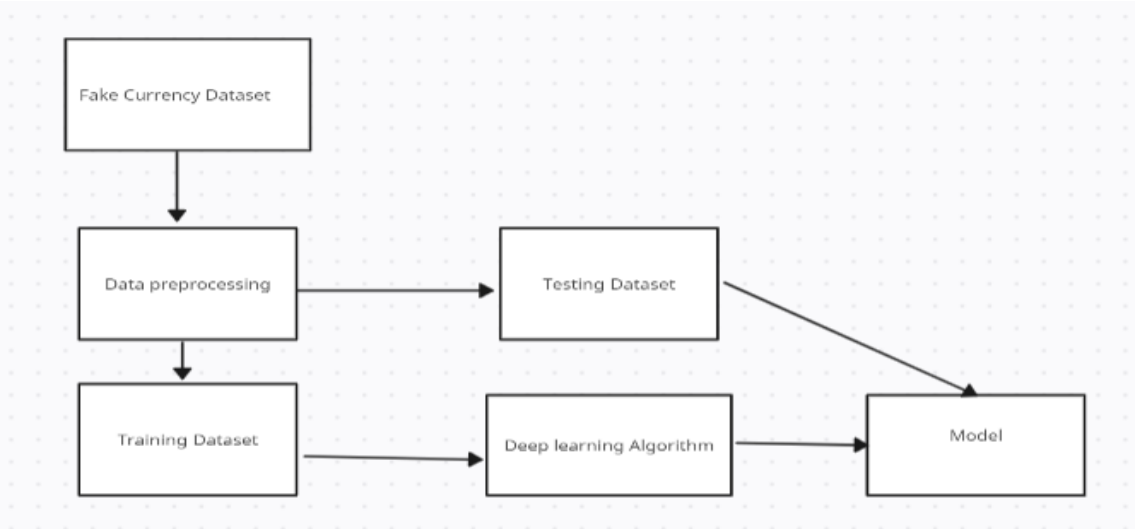

Figure 3.1.1: Architecture

The process starts with a dataset of fake currency. This dataset is then preprocessed, which may involve tasks such as cleaning the data, normalizing it, and converting it into a format that the deep learning algorithm can understand. Once the data is preprocessed, it is split into two sets: a training dataset and a testing dataset. The training dataset is used to train the deep learning algorithm, while the testing dataset is used to evaluate

the performance of the algorithm. The deep learning algorithm is a complex mathematical model that is able to learn from the data in the training dataset. Once the algorithm is trained, it can be used to make predictions on new data. In this case, the new data would be images of currency, and the algorithm would predict whether or not the currency is fake. The model is then evaluated using the testing dataset. This involves feeding the testing dataset into the model and seeing how well it performs. If the model performs well, it can be used to detect fake currency in real-world applications.

3.2 Tools Used

The following tools and libraries are used for implementing fake currency detection system:

Google Colab: Google Colab is utilized to collaboratively implement and execute code for data science and machine learning projects in a cloud-based environment, providing free access to GPUs and TPUs for accelerated computations.

NumPy: NumPy is a powerful numerical computing library in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

pandas: Pandas is a data manipulation and analysis library. It offers data structures like DataFrame, which is particularly useful for handling structured data. Pandas simplifies tasks such as data cleaning, exploration, and transformation.

Matplotlib: Matplotlib is a comprehensive 2D plotting library for creating static, interactive, and animated visualizations in Python. It is widely used for generating plots, charts, and graphs to visualize data distributions and trends.

Seaborn: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides an interface for creating informative and attractive statistical graphics. Seaborn simplifies the process of creating complex visualizations and enhances the aesthetic appeal of plots.

TensorFlow: TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive set of tools and resources

for building and deploying machine learning models, particularly neural networks. In this code, it is used as the backend for Keras.

Keras: Keras is a high-level neural networks API written in Python. It is designed to be user-friendly, modular, and extensible. Keras allows developers to build and experiment with neural network models easily. In this code, it is used for building and training neural network architectures.

Chapter 4

Methodology

4.1 Algorithms used

4.1.1 KNN Algorithm

The k-Nearest Neighbors (KNN) algorithm is a simple yet effective supervised machine learning algorithm used for classification and regression tasks. It operates on the principle of similarity, where it classifies new data points based on the majority class of their k nearest neighbors in a feature space. In classification, when a new data point needs to be classified, KNN identifies the k closest data points (neighbors) based on a chosen distance metric (such as Euclidean distance) and assigns the class label that occurs most frequently among these neighbors. For regression tasks, KNN predicts the output value for the new data point by averaging or considering the mean of the target values of its k nearest neighbors. The choice of 'k', the number of neighbors, is crucial and can impact the algorithm's performance, where smaller values of 'k' can lead to more flexible decision boundaries (potentially overfitting) while larger values can lead to smoother but potentially oversimplified boundaries (potentially underfitting). While KNN is straightforward and easy to understand, its computational complexity grows with the size of the training dataset since it requires calculating distances between the new data point and all existing data points. Additionally, the choice of the distance metric and the value of 'k' greatly impacts the algorithm's performance and generalization ability, and tuning these parameters is essential for optimal results.

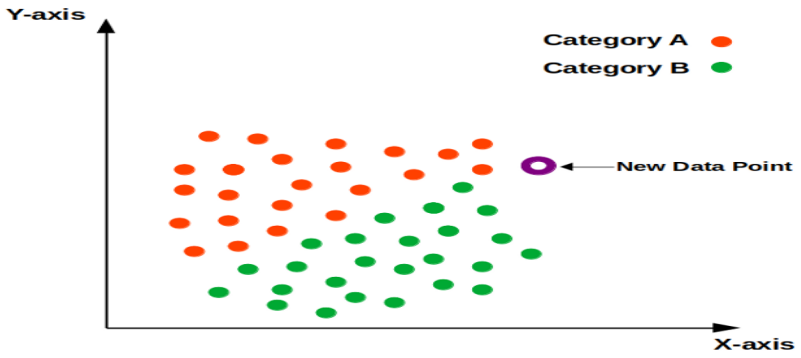

Figure 4.1.1: KNN Algorithm

4.1.2 SVM Algorithm

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for classification and regression tasks. Its primary goal is to find the optimal hyperplane that best separates classes in a high-dimensional space. In classification, SVM aims to create a decision boundary (hyperplane) that maximizes the margin, which is the distance between the closest data points of different classes known as support vectors. These support vectors are the critical data points that influence the position and orientation of the hyperplane. SVM can handle linear and nonlinear data by using different kernel functions (like polynomial, radial basis function, etc.) that map the data into higher-dimensional spaces where classes are more separable. For regression tasks, SVM employs a similar principle by finding a hyperplane that best fits the data within a certain margin of tolerance. It aims to minimize errors while maximizing the margin, thus finding the optimal fit for the given data. SVM is effective in handling high-dimensional data, works well with a clear margin of separation between classes, and is robust against overfitting when the right parameters are chosen. However, it can be sensitive to noise and might become computationally expensive with large datasets.

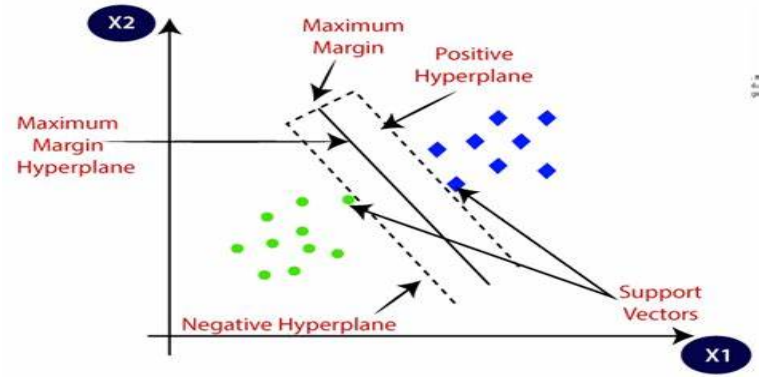

Figure 4.1.2: SVM Algorithm

4.1.3 CNN Algorithm

Convolutional Neural Networks (CNNs) are a class of deep neural networks primarily used for analyzing visual imagery in applications such as image recognition, object detection, and image classification. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from the input data. They consist of multiple layers including convolutional layers, pooling layers, and fully connected layers.

key components of cnn are:

- Input Layer
- Conventional layers (Conv2D)
- Batch normalization layers (Batch Normalization)
- Max Pooling
- Flatten Layer
- Output Layer

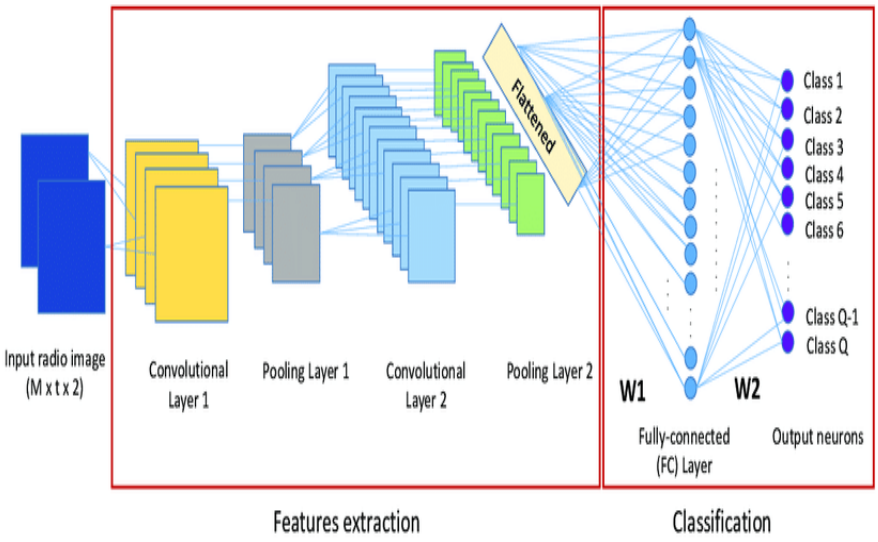

Figure 4.1.3: CNN Algorithm

Input layer: This layer takes in an image as input. In your case, the input shape is (224, 224, 3), which means the image is 224 pixels wide, 224 pixels tall, and has 3 channels (one for each color: red, green, and blue).

Convolutional layers (Conv2D): These layers extract features from the image by convolving it with a small filter. The first convolutional layer in your model has 32 filters, each of which is 3x3 pixels in size. This means that the layer will extract 32 different features from the image. The second convolutional layer has 64 filters, and so on.

Batch normalization layers (BatchNormalization): These layers help to stabilize the training process by normalizing the activations of the previous layer.

Pooling layers (MaxPooling2D and AveragePooling2D): These layers reduce the size of the feature maps by taking the maximum or average value of a small region of the input. This helps to reduce the number of parameters in the network and prevent overfitting.

Flatten layer: This layer converts the feature maps into a one-dimensional vector.

Fully-connected layers (Dense): These layers learn complex relationships between the features extracted by the convolutional layers. The first fully-connected layer in your model has 2056 neurons, the second layer has 512 neurons, and the third layer has 256 neurons.

Output layer: This layer outputs the predictions of the network. In your case, the output layer has two neurons, one for each class in your classification task.

Chapter 5

Implemention

5.1 Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D,GlobalAveragePooling2D, Flatten, Dropout, BatchNormalization, Input,AveragePooling2D
from glob import glob
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import load_img, img_to_array
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.models import load_model
import os
```

Figure 5.1: Importing libraries

The script employs Python libraries (NumPy, pandas, matplotlib, seaborn, TensorFlow, Keras) for machine learning and visualization, focusing on image classification with a convolutional neural network. It manages data processing, splitting, and integrates callbacks for training control. The code also handles directories and pre-trained model loading, with a minor 'os' module import typo.

5.2 Loading dataset

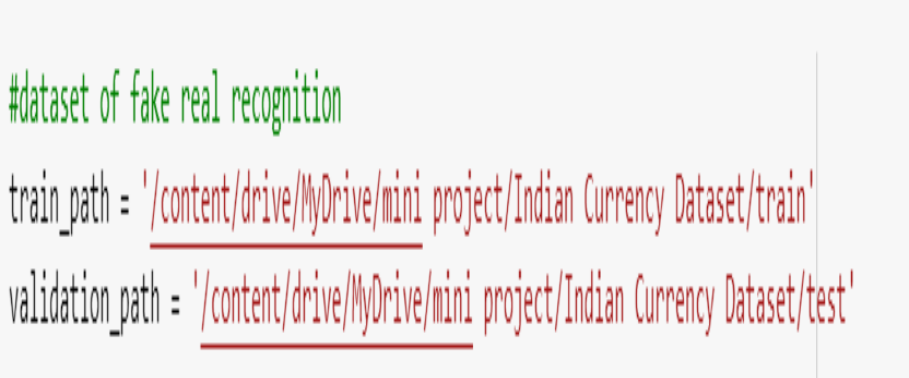

Figure 5.2: Loading Dataset

The above code defines file paths for a dataset used in a fake and real currency recognition project. The training images are located in the '/content/drive/MyDrive/mini project/Indian Currency Dataset/train' directory, and validation images are in '/content/drive/MyDrive/mini project/Indian Currency Dataset/test'. These paths likely point to directories containing labeled images for model training and evaluation.

5.3 Data Preprocessing

```
# data augumentation
train_datagen = ImageDataGenerator(rescale=1/255)
validation_datagen=ImageDataGenerator(rescale=1/255)
test_datagen=ImageDataGenerator(rescale=1/255)
```

Figure 5.3: Data generation

The above code initializes data generators for data augmentation. The 'ImageDataGenerator' from Keras is used for preprocessing images during model training. It applies rescaling to normalize pixel values between 0 and 1 ('rescale=1/255'). This normalization is a common practice in deep learning to improve model convergence and performance. The generators are likely used with image datasets for training, validation, and testing.

The below code configures image data generators with augmentation for training and validation. The 'ImageDataGenerator' for training includes shear, horizontal flip, and zoom adjustments, along with rescaling. Training and validation generators are then created using the specified paths.

```
batch_size = 10
# recognition data

train_datagen = ImageDataGenerator(rescale=1/255,
                  shear_range=0.3,
                  horizontal_flip=True,
                  zoom_range=0.3
                  )
val_datagen = ImageDataGenerator(rescale=1/255)

train_generator = train_datagen.flow_from_directory(
                train_path,
                target_size=(224,224),
                batch_size=batch_size,
                color_mode="rgb",
                class_mode="categorical"
                )

val_generator = val_datagen.flow_from_directory(
                validation_path,
                target_size=(224,224),
                batch_size=batch_size,
                color_mode="rgb",
                class_mode="categorical"
                )
```

```
Found 248 images belonging to 2 classes.
Found 107 images belonging to 2 classes.
```

Figure 5.3: Data Augmentation

5.4 Model Training

The code defines a convolutional neural network (CNN) for image classification using Keras with TensorFlow backend. The model architecture consists of convolutional layers with batch normalization, max pooling, and average pooling operations. The network is designed for binary classification (2 output classes) with a final sigmoid activation. The fully connected layers include dropout for regularization. The overall structure is organized using the Functional API of Keras, creating a model that takes input with shape (224, 224, 3) and outputs the classification result. The model architecture involves progressively reducing spatial dimensions through convolution and

pooling layers, followed by dense layers for classification.

```
input_shape=(224,224,3)
def building_the_model(input_shape):
    inputs = Input(shape=input_shape)
    model = (Conv2D(32, (3, 3),strides=2,padding='same', activation='relu',data_format='channels_last' ,input_shape=input_shape))(inputs)
    model = (BatchNormalization())(model)
    model = (Conv2D(64, (3, 3),strides=2,padding='same', activation='relu',data_format='channels_last', input_shape=input_shape))(model)
    model = (BatchNormalization())(model)
    model = (MaxPooling2D(pool_size=(2, 2), strides=2,data_format='channels_last'))(model)
    model = (Conv2D(128, (3, 3),strides=2,padding='same', activation='relu',data_format='channels_last', input_shape=input_shape))(model)
    model = (BatchNormalization())(model)
    model = (AveragePooling2D(pool_size=(2, 2), strides=2,data_format='channels_last'))(model)
    model = (Flatten())(model)
    model = (Dense(2056))(model)
    model = tf.keras.layers.Dropout(0.2)(model)
    model = (Dense(512))(model)
    model = tf.keras.layers.Dropout(0.2)(model)
    model = (Dense(256))(model)
    model = (Dense(2,activation="sigmoid"))(model)

    #     outputs = (Dense(16))(model)
    X = model

    # Bui4ld model
    model = Model(inputs=inputs, outputs=X)

    return model
model = building_the_model(input_shape)
```

Figure 5.4: Model training

```
model.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
```

```
history = model.fit(train_generator,
                    validation_data = val_generator,
                    batch_size = 128,
                    epochs = 50)
```

Figure 5.4: Model Training

The code compiles a neural network for binary classification using Adam optimizer and binary cross entropy loss. It then trains the model for 50 epochs on training data with validation data, using a batch size of 128. The training history is stored in the 'history' variable.

5.5 Model Evaluation

```
from sklearn.metrics import classification_report, confusion_matrix

val_predictions = model.predict(val_generator)
val_predicted_labels = np.argmax(val_predictions, axis=1)
val_true_labels = val_generator.classes

# Generate and print classification report
class_names = list(train_generator.class_indices.keys())
print("Classification Report:")
print(classification_report(val_true_labels, val_predicted_labels, target_names=class_names))
```

Figure 5.5: Evaluation

The model's predictions are generated for the validation dataset. The predict function returns raw output probabilities for each class. The predicted labels are then extracted by selecting the class index with the highest probability using np.argmax. True labels are obtained from the generator's classes attribute.

Chapter 6

Results and Discussions

6.1 Experimental Results-1

The results showcased promising accuracy rates, with the K-Nearest Neighbors (KNN) algorithm achieving a respectable 83% accuracy in distinguishing fake currency. Surpassing this, the Support Vector Machine (SVM) exhibited a commendable accuracy of 89%, indicating its efficacy in detecting counterfeit notes. However, the Convolutional Neural Network (CNN) emerged as the frontrunner, boasting an impressive accuracy of 91%. This outcome underscores the CNN's robust ability to discern intricate patterns and features within currency images, signifying its potential as a highly reliable method for counterfeit detection in financial systems.

Classification Report:

	precision	recall	f1-score	support
fake	0.82	0.90	0.85	59
real	0.86	0.75	0.80	48
accuracy			0.83	107
macro avg	0.84	0.82	0.83	107
weighted avg	0.83	0.83	0.83	107

Figure 6.1.1: Classification report of KNN

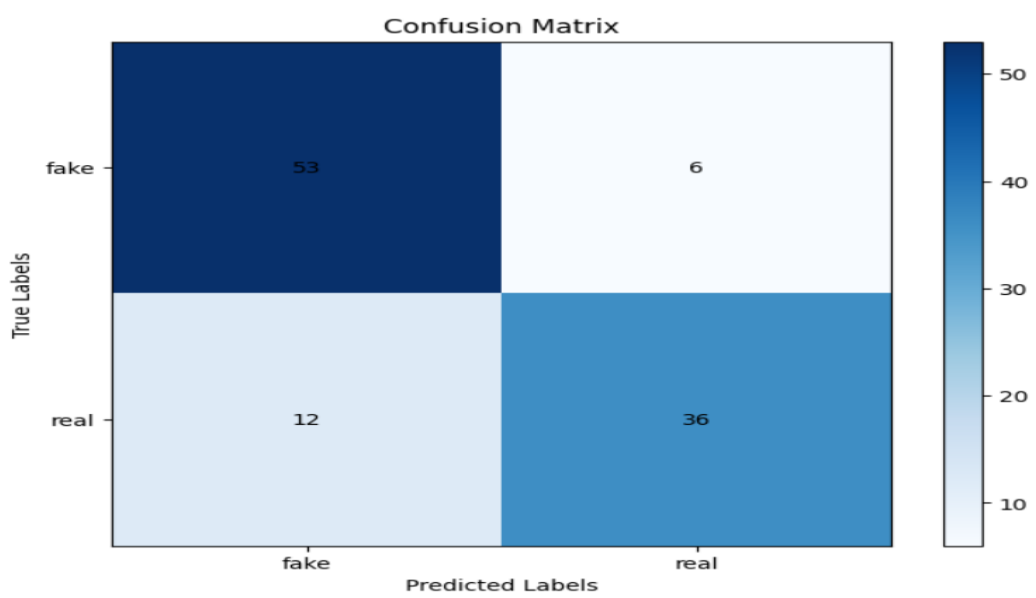

Figure 6.1.2: Confusion Matrix of KNN

Classification Report:

	precision	recall	f1-score	support
fake	0.87	0.90	0.88	59
real	0.87	0.83	0.85	48
accuracy			0.87	107
macro avg	0.87	0.87	0.87	107
weighted avg	0.87	0.87	0.87	107

Figure 6.1.3: Classification report of SVM

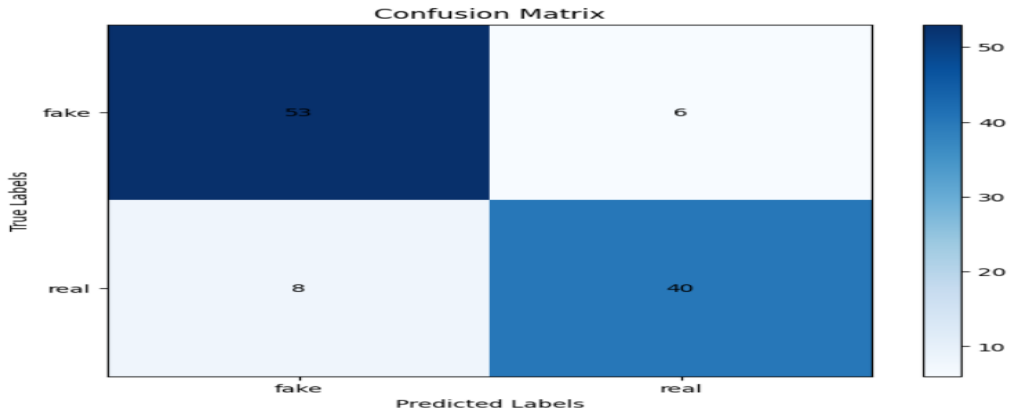

Figure 6.1.4: Confusion Matrix of SVM

Classification Report:

	precision	recall	f1-score	support
fake	1.00	0.83	0.91	59
real	0.83	1.00	0.91	48
accuracy			0.91	107
macro avg	0.91	0.92	0.91	107
weighted avg	0.92	0.91	0.91	107

Figure 6.1.5: Classification report of CNN

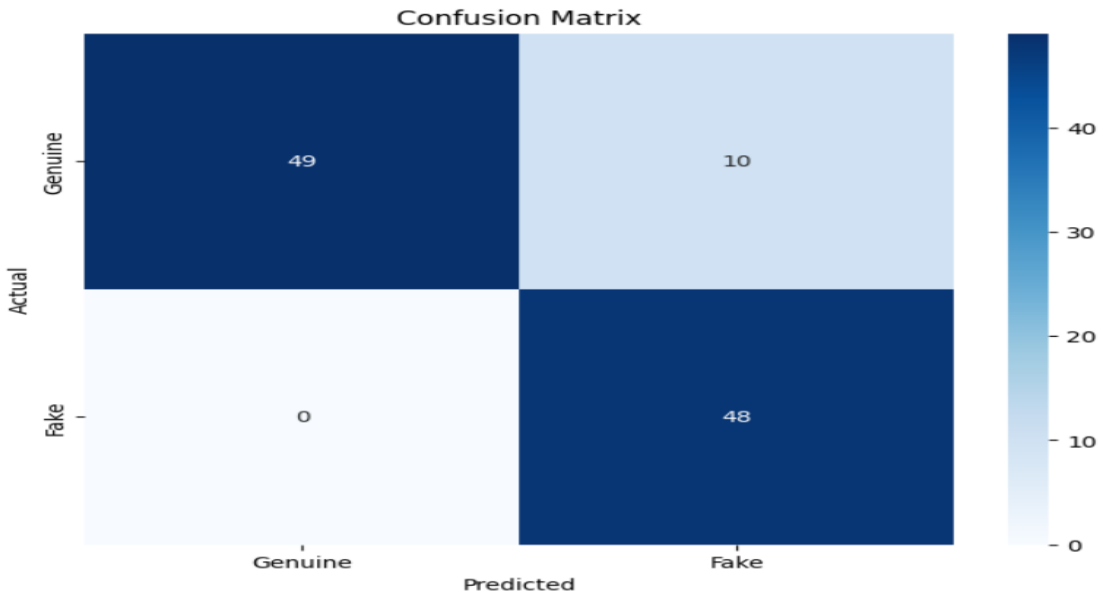

Figure 6.1.6: Confusion Matrix of CNN

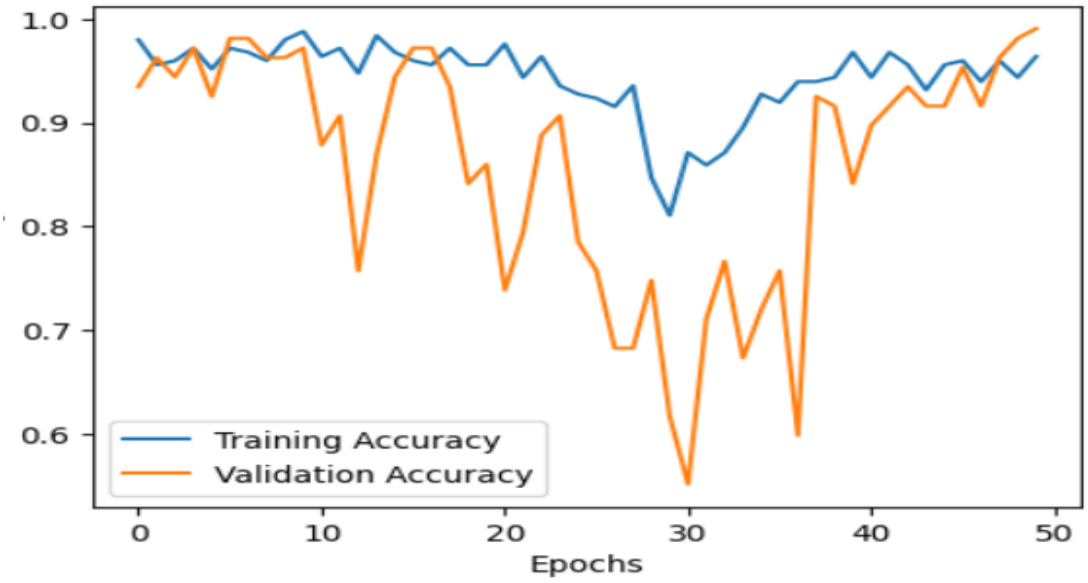

Figure 6.1.7: Modal Accuracy of CNN

Figure 6.1.8: Prediction by Model

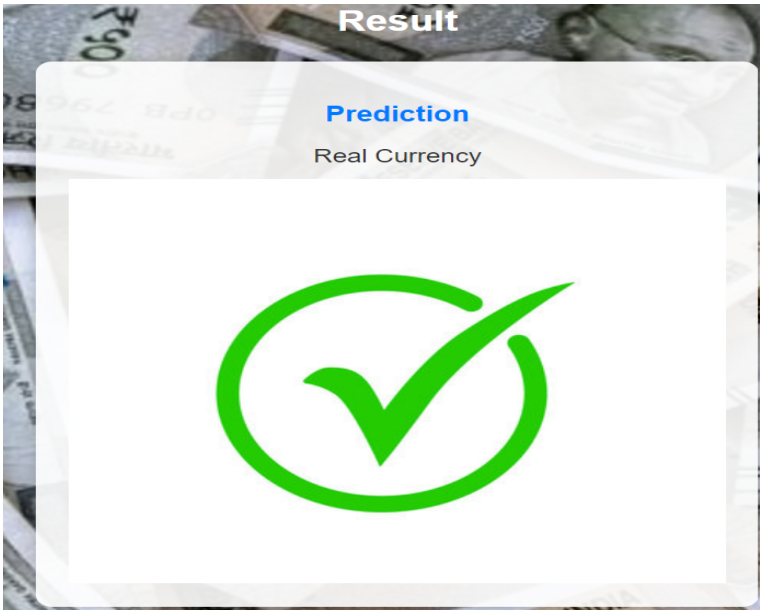

Figure 6.1.9: Prediction by Model

Chapter 7

Conclusions and future works

The utilization of Convolutional Neural Networks (CNNs) for fake currency detection stands as a highly promising and effective approach based on the attained results. With a remarkable accuracy of 91% in discerning counterfeit currency, the CNN proves its prowess in analyzing and identifying intricate patterns and distinguishing features within currency images. This exceptional accuracy signifies the CNN's robustness in detecting minute discrepancies that might indicate counterfeit notes, showcasing its potential as a reliable and precise tool for counterfeit currency detection in financial systems. As a result, employing CNNs in the realm of currency authentication holds significant promise for bolstering security measures and safeguarding against fraudulent practices in the monetary domain.

The current model's effectiveness in detecting fake currency, achieving a commendable accuracy with major denomination notes, lays a solid foundation for its potential expansion. The dataset's focus on major denominations signifies a valuable starting point, but the model's capacity can be significantly enhanced by incorporating minor denominations as well. Introducing data pertaining to smaller currency values into the training process could fortify the model's ability to discern a wider range of counterfeit notes, ensuring a more comprehensive understanding of the intricacies across various denominations. This expansion would not only augment the model's accuracy but also render it more robust and adaptable to real-world scenarios, thereby bolstering its applicability in combating counterfeit practices across a broader spectrum of currency values. As such, further training with minor denomination currency stands as a promising avenue to fortify the model's efficacy and bolster its real-world utility in counterfeit currency detection.

Bibliography

[1] A. Rajee, "Fake currency detection," *IEEE Access*, pp. 28645–28657, 2022.

[2] S. Pallavi, N. Pooja, H. Yashaswini, and N. Varsha, "Fake currency detection," *International Research Journal of Modernization in Engineering Technology and Science (4076-4081)*, vol. 4, no. 06, 2022.

[3] R. M. Colaco, R. Fernandes, S. Sowmya, *et al.*, "Efficient image processing technique for authentication of indian paper currency," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–8, IEEE, 2021.

[4] D. Jamkhandikar, S. Kaveri, and V. P. Sudharani, "Indian currency recognition system," 2021.

[5] M. J. Sangogi, M. P. Patil, and M. A. Jadhav, "Fake currency detection using image processing,"

[6] P. Kumar, V. Akhila, B. Sushmitha, and K. Anusha, "Detection of fake currency using knn algorithm,"

[7] D. P. Patil, G. Varma, S. Poojary, S. Sawant, and A. Sharma, "Counterfeit currency detection based on ai,"

[8] S. Kudalkar, P. Patil, and N. Shirdhone, "Fake currency detection using image processing," *AIJR Preprints*, 2022.

[9] A. Vidhate, Y. Shah, R. Biyani, H. Keshri, and R. Nikhare, "Fake currency detection application-know your currency!," *PCE JCE_*, vol. 25.