Karnatak University

Dharwad

Global Educare Foundation's

Global College Of Computer Application

A Project Report on "Handwritten Document Image Forgery Detection"

Submitted in partial fulfillment for the award of the degree

Bachelor of Computer Application

Of Karnatak University, Dharwad.

Under the Guidance of

Prof.**Ayeesha**

Submitted By

Vishwajeet Kulkarni– 20U10586

Akash Kalkeri – 20U10507

Karnatak University

Dharwad

Global Educare Foundation's

Global College Of Computer Application

CERTIFICATE

This is to certify that Mr **Vishwajeet** and **Akash** has satisfactorily completed Project work entitled "**Handwritten Document Image Forgery Detection** " for the partial fulfillment of Bachelor of Computer Application by Karnatak University, Dharwad for the academic year 2022-2023.

Candidate Name: - Vishwajeet Kulkarni Candidate Name: - Akash Kalkeri

Register Number: - 20U10586 Register Number: -20U10586

Project Guide	Cordinator	Principal	Dean
Prof. Ayeesha	**Prof.Soumyashree.M**	**Prof.Ashwinikumar Koti**	**Dr.Mahesh Deshpande**

Examiner 1: ______________________

Examiner 2: ______________________

Acknowledgement

Apart from the efforts of us, the success of any project depends on the encouragement and Guidelines. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We express our gratitude and regards to our respected and beloved Dean,Dr. Mahesh Deshpande & also our beloved principal Prof.Ashwinikumar Koti and guide Prof. Ayeesha for every support extended.

The acknowledgement would be incomplete if we don’t mention our gratitude to staff members of BCA department. We are highly grateful to our external guide who helped us throughout the project.

Finally last but not least we want to forward warm regards to our beloved parents and friends because without their help and support nothing would be possible.

STUDENT NAME

Vishwajeet Kulkarni

Akash Kalkeri

CONTENTS

CHAPTER - 1

INTRODUCTION

- Handwritten documents hold significant value in various domains such as legal, historical, and administrative sectors. However, these documents are vulnerable to forgery, posing a threat to their authenticity and reliability. Detecting forged signatures, altered content, or fabricated documents is a challenging task that traditionally relies on manual inspection and expert analysis.
- In recent years, deep learning techniques have revolutionized various fields, including image recognition and natural language processing. Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have demonstrated remarkable capabilities in learning complex patterns and features from large datasets. Leveraging the power of deep learning, there is an opportunity to develop automated and efficient systems for handwritten document forgery detection.
- The study aims to explore the application of deep learning techniques for handwritten document forgery detection. By training deep learning models on a diverse dataset containing genuine and forged handwritten documents, we can leverage the models' ability to extract intricate features and patterns. This automated approach offers the potential to significantly enhance the accuracy, efficiency, and scalability of forgery detection, providing a valuable tool for document authentication.
- Further the study encompasses several key steps. First, a comprehensive dataset of genuine and forged handwritten documents will be collected, covering various types of forgeries and writing styles. Next, the collected dataset will undergo pre-processing steps, including image cropping, resizing, noise reduction, and normalization, to improve the quality and consistency of the input data. Subsequently, an appropriate deep learning architecture will be designed, considering the unique requirements of handwritten document forgery detection. The model will be trained using the prepared dataset, optimizing its parameters to minimize the difference between predicted and ground truth labels.

OBJECTIVES OF THE PROJECT :-

The objectives of the project "Handwritten Document Image Forgery Detection in Blurry and Noisy Environment using Deep Learning " are as follows:

1. Creation of new Forged and Genuine Handwritten document image database.

2. Study and analysis of existing forged handwritten document image datasets.

3. Develop a robust method for automatic classification and detection of forged area in handwritten document image especially in case of Noisy and Blurry environment.

4. Implement deep learning techniques, such as convolutional neural networks (CNNs) and Transfer learning architectures, to extract informative features of Forged Handwritten document images.

5. Enhance the robustness of the handwritten document forgery detection system by utilizing standard and benchmark forged handwritten document image datasets,

6. Evaluate the performance of the developed system using appropriate metrics, such as accuracy, precision, recall, and F1 score, comparing it with existing forged handwritten document image detection methods.

• 1.1 Existing System:-

The forgery we detect can be classified as hand-written signature forgery and copy-move forgery of any photo, text, or signature. We have developed a novel approach using **capsule layers** to detect a forgery in handwritten signatures. We also use ELA (Error Level Analysis) to detect any error in the compression levels of the image.

• 1.2 Problem definition:-

The problem of handwritten document forgery poses a significant challenge in ensuring the authenticity and reliability of important documents. The problem addressed in the proposed work on deep learning for handwritten document forgery detection is the accurate identification and differentiation between genuine and forged handwritten documents. Traditional manual methods for forgery detection are time-consuming, subjective, and prone to human error, making them inefficient and unreliable for large-scale document authentication. The goal is to develop a robust and automated system that can effectively detect various forms of forgery, 3 such as signature forgery, alteration of handwritten text, or the creation of entirely fabricated handwritten documents.

• 1.3 Proposed System:-

The proposed methodology for Handwritten Document Image Forgery Detection consists of several stages. The following steps outline the methodology.

1.Data Collection: Gather a diverse dataset of genuine and forged handwritten documents. Acquire a sufficient number of samples covering different types of forgeries, writing styles, and document characteristics. Ensure that the dataset represents a wide range of scenarios for robust model training.

2. Data Pre-processing: Pre-process the collected dataset to enhance the quality and consistency of the input data. Apply techniques such as image cropping, resizing, noise reduction, and normalization to improve the overall data quality and standardize the samples for effective model training. Image acquisition Pre-processing Segmentation Feature Extraction Classification Forgery detection Genuine Forged Display forged area Fig. 1. General architecture of the proposed methodology Knowledge Base 4

3. Model Selection and Architecture: Choose an appropriate deep learning architecture suitable for handwritten document forgery detection. Common choices include convolutional neural networks (CNNs) for image-based tasks or recurrent neural networks (RNNs) for sequential data. Design the architecture considering the complexity of the forgery detection task and the specific requirements of handwritten document analysis.

4. Training and Validation: Split the pre-processed dataset into training and validation sets. Train the deep learning model using the training set, optimizing its parameters through back propagation and gradient descent to minimize the difference between predicted and ground truth labels. Validate the model's performance on the validation set and iterate on the training process to improve accuracy and generalization.

5. Model Evaluation: Evaluate the trained model on a separate testing dataset that was not used during training. Measure key performance metrics such as accuracy, precision, recall, and F1 score to assess the effectiveness of the model in detecting different types of forgeries. Perform thorough analysis and comparison of the results to gain insights into the model's performance

Also this system considers 3 important parameters or parts for recognition and detection.

Logo , Handwritten Text or other text and Signature.

In most cases of important documents these 3 parts are almost always present.

CHAPTER-2

DESIGN/STRUCTURE

2.1 ACTIVITY DIAGRAM

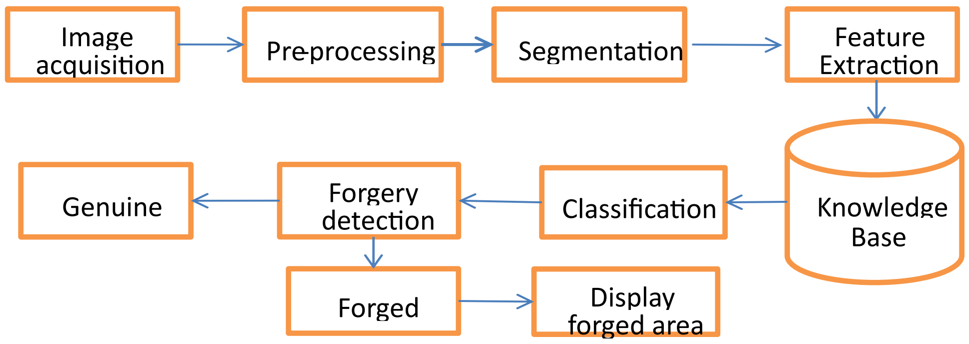
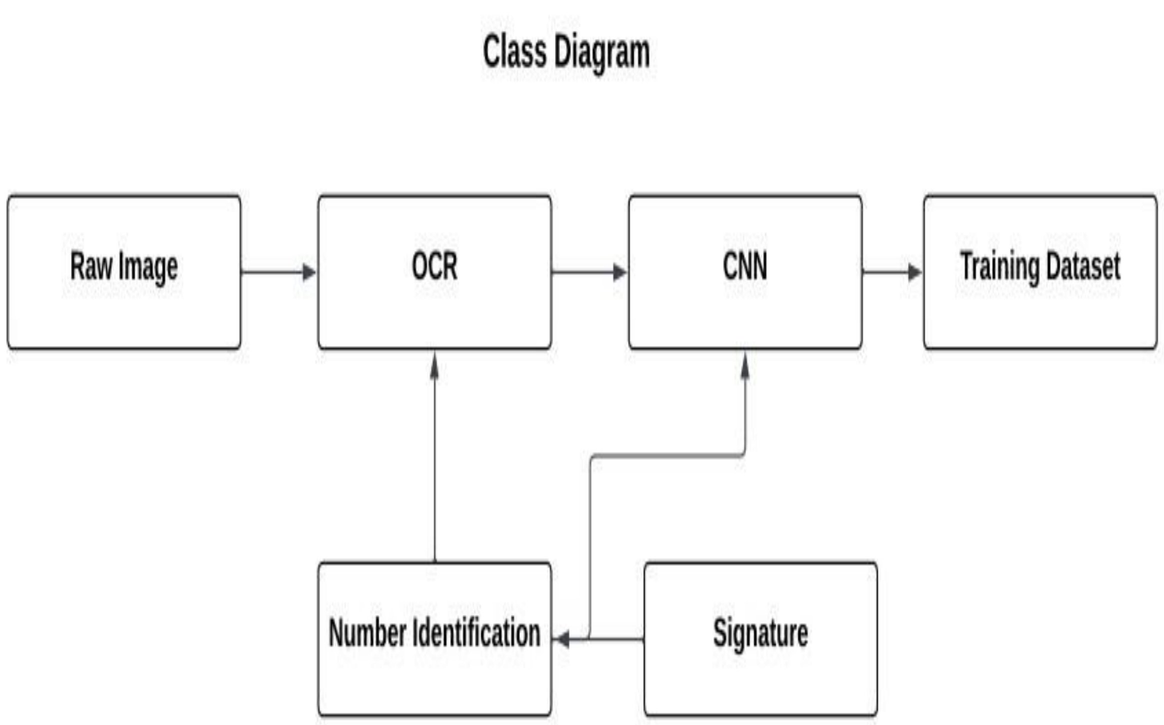
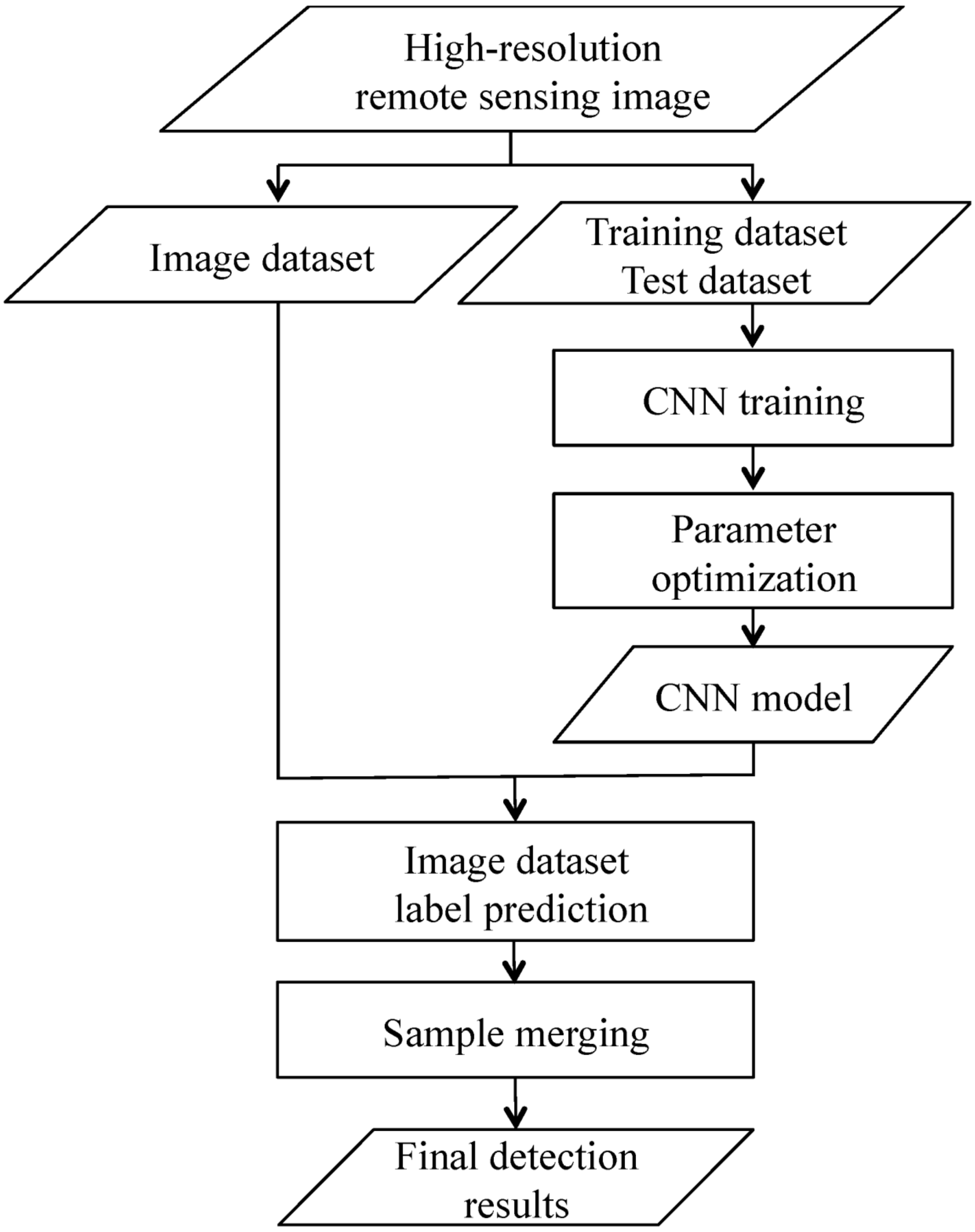


Fig. 1. General architecture of the proposed methodology


Class Diagram
Raw Image
OCR
CNN
Training Dataset
Number Identification
Signature

.

2.2FLOW CHARTS


High-resolution remote sensing image
Image dataset
Training dataset Test dataset
CNN training
Parameter optimization
CNN model
Image dataset label prediction
Sample merging
Final detection results

CHAPTER-3

REQUIREMENTS

3.1 HARDWARE REQUIREMENTS.

- Hard Disk: 1.2GB or above
- RAM: 8 GB
- Processor: Intel i5 Processer or above PREPROCESSING

3.2 SOFTWARE REQUIREMENTS.

- Python 3.6.8
- Deep learning frameworks such as TensorFlow, Kera's, or Torch
- Image processing libraries such as OpenCV
- Jupiter Notebook or an Integrated Development Environment (IDE) such as PyCharm or Visual Studio Code

CHAPTER-4

CODING.

4.1 INTRODUCTION OF PROGRAMMING LANGUAGE

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limitedto only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL)natural language processing, data science etc. Python has a lot of libraries and tools for every need of this project. For Handwritten Digit Recognition the python libraries and tools such as Numpy, Keras, TensorFlow, Tkinter and Pillow are used.

Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most the time and implement just that part more efficiently in some lower-level language. This will result in much less programming and more efficient code (because you will have more time to optimize) than writing everything in a low-level language.

4.2 MODULES.

PYTHON and LIBRARIES

The python idle editor is used to write and edit code for the entire software. To run the program code, it has to be run through the terminal. To install the libraries the following command is used in the terminal- pip install numpy, tensorflow, keras, pillow.

1. Keras: Keras is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models. 18 It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and

TensorFlow are very powerful libraries but difficult to understand for creating neural networks. Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

2. TensorFlow: TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow tutorial is designed for both beginners and professionals. Our tutorial provides all the basic and advanced concept of machine learning and deep learning concept such as deep neural network, image processing and sentiment analysis. TensorFlow is one of the famous deep learning frameworks, developed by Google Team. It is a free and open-source software library and designed in Python programming language, this tutorial is designed in such a way that we can easily implements deep learning project on TensorFlow in an easy and efficient way. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems. It can run on single CPU systems, GPUs as well as mobile devices and largescale distributed systems of hundreds of machines.

3. Numpy: NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. Numpy which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. It is an open source project and you can use it freely. NumPy stands for Numerical Python. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important. 19

4. Pillow: Pillow is a free and open source library for the Python programming language that allows you to easily create &s manipulate digital images. Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3. Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety

of images such as "jpeg", "png", "bmp", "gif", "ppm", "tiff". You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

.**CNN:**

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.

CNNs are trained using a large dataset of labeled images, where the network learns to recognize patterns and features that are associated with specific objects or classes. Once trained, a CNN can be used to classify new images, or extract features for use in other applications such as object detection or image segmentation.

CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, including object classification, object detection, and image segmentation. They are widely used in computer vision, image processing, and other related fields, and have been applied to a wide range of applications, including self-driving cars, medical imaging, and security systems.

- A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals.
- CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces.
- This characteristic that makes convolutional neural network so robust for computer vision.
- CNN can run directly on a underdone image and do not need any pre-processing.
- A convolutional neural network is a feed forward neural network, seldom with up to 20.
- The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer.
- CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes.
- With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces.
- The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.
It is the sequential design that give permission to CNN to learn hierarchical attributes.

In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.

Train the Model.

The model.fit() function of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size.

It takes some time to train the model. After training, we save the weights and model definition in the file.

4.3 SAMPLE CODE

To Train the models

```
import numpy as np
import os
import cv2
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.optimizers import Adam

# Set the path to the dataset
dataset_path = 'logo_dataset/'

# Set the image size for resizing
image_size = (128, 128)

def load_dataset():
    images = []
    labels = []

    # Load genuine document images
    genuine_path = os.path.join(dataset_path, 'Original_logo')
    for filename in os.listdir(genuine_path):
        file_path = os.path.join(genuine_path, filename)

        # Check if the file is an image (you can add more image formats as needed)
        if filename.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):
```

```
        img = cv2.imread(file_path)

        if img is not None:
            img = cv2.resize(img, image_size, interpolation=cv2.INTER_AREA)
            images.append(img)
            labels.append(0)  # Genuine label = 0
        else:
            print(f"Error reading image: {file_path}")

# Load forged document images
forged_path = os.path.join(dataset_path, 'Forged_logo')
for filename in os.listdir(forged_path):
    file_path = os.path.join(forged_path, filename)

    # Check if the file is an image (you can add more image formats as needed)
    if filename.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):
        img = cv2.imread(file_path)

        if img is not None:
            img = cv2.resize(img, image_size, interpolation=cv2.INTER_AREA)
            images.append(img)
            labels.append(1)  # Forged label = 1
        else:
            print(f"Error reading image: {file_path}")

# Convert lists to numpy arrays
images = np.array(images)
labels = np.array(labels)

return images, labels
```

```
def build_model(input_shape):
    model = Sequential()

    # Add convolutional layers
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())

    # Add dense layers
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    return model

def train_and_save_model():
    # Load the dataset
    images, labels = load_dataset()

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

    # Normalize the pixel values to the range [0, 1]
    X_train = X_train.astype('float32') / 255.0
    X_test = X_test.astype('float32') / 255.0

    # Build the model
    input_shape = X_train[0].shape
    model = build_model(input_shape)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

    # Train the model
    model.fit(X_train, y_train, batch_size=32, epochs=20, validation_data=(X_test, y_test))

    # Save the trained model to an h5 file
    model.save('logo_model.h5')
    print("Trained model has been saved to 'logo_model.h5'.")

if __name__ == '__main__':
    train_and_save_model()
```

To test the models

```
import cv2
import numpy as np
from keras.models import load_model

def preprocess_image(image_path):
    # Load the image
    img = cv2.imread(image_path)

    # Check if the image was loaded successfully
    if img is None:
        print(f"Error: Unable to read the image from {image_path}")
        return None

    # Resize the image to the model's input shape
    img = cv2.resize(img, (128, 128), interpolation=cv2.INTER_AREA)

    # Normalize the pixel values to the range [0, 1]
    img = img.astype('float32') / 255.0

    # Expand the dimensions to match the model's input shape (add batch dimension)
    img = np.expand_dims(img, axis=0)

    return img

def predict_genuine_or_forged(image_path):
    # Load the trained model from the file
    model = load_model('logo_model.h5')

    # Preprocess the image
    img = preprocess_image(image_path)

    # Check if image preprocessing was successful
    if img is None:
        return None

    # Use the model to predict the label
    prediction = model.predict(img)
    print("prediction",prediction)

    # Get the predicted class (genuine or forged)
    if prediction[0][0] >= 0.5:
        result = 'Forged_logo'
    else:
        result = 'Original_logo'

    return result

if __name__ == '__main__':
    # Specify the path to the image you want to classify
    image_path_to_classify = 'logotest1.png'
    # Get the result (genuine or forged)
    result = predict_genuine_or_forged(image_path_to_classify)
    print(f'The document is {result}')
```

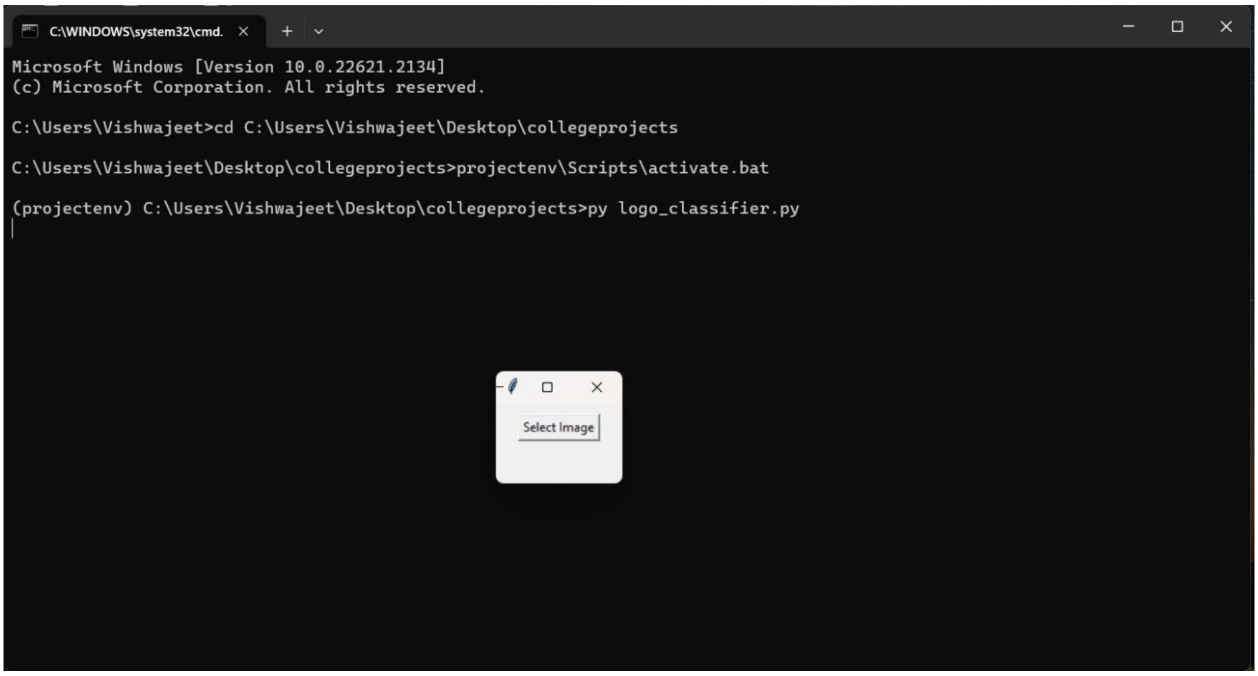
CHAPTER-5

SCREENSHOTS

5.1 TERMINAL COMMANDS

Changing Directory and activating virtual environment

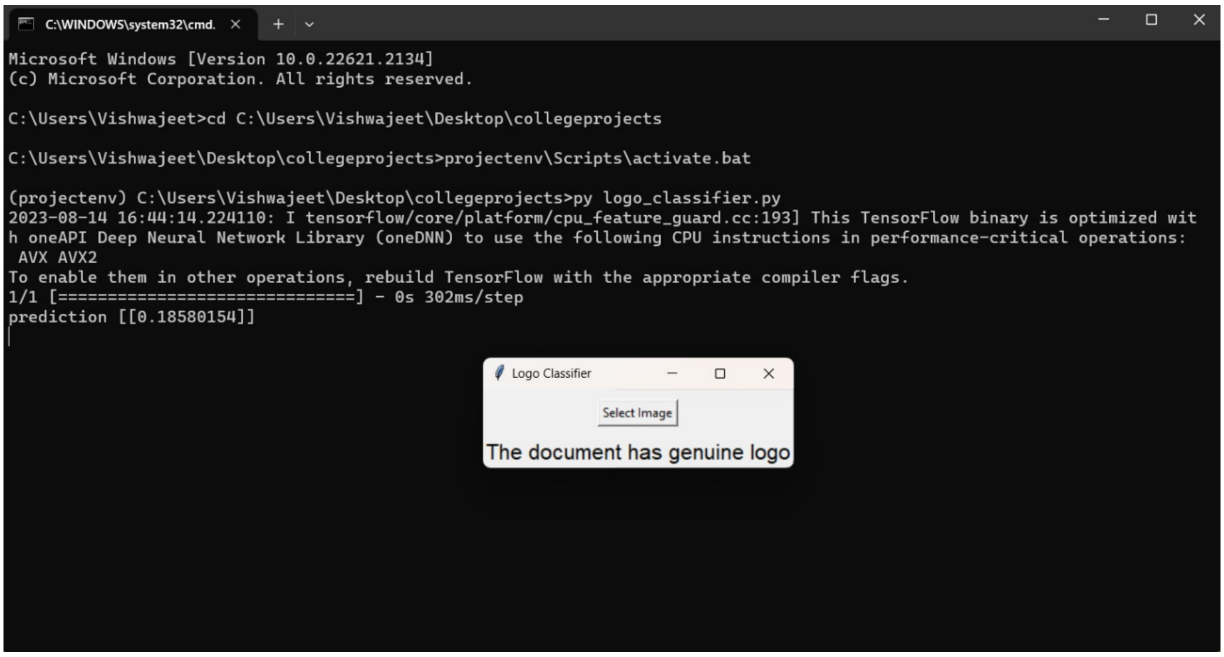
First running logo_classifier which starts a tkinter ui to select image to verify.



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vishwajeet>cd C:\Users\Vishwajeet\Desktop\collegeprojects

C:\Users\Vishwajeet\Desktop\collegeprojects>projectenv\Scripts\activate.bat

(projectenv) C:\Users\Vishwajeet\Desktop\collegeprojects>py logo_classifier.py
```

5.2 Output

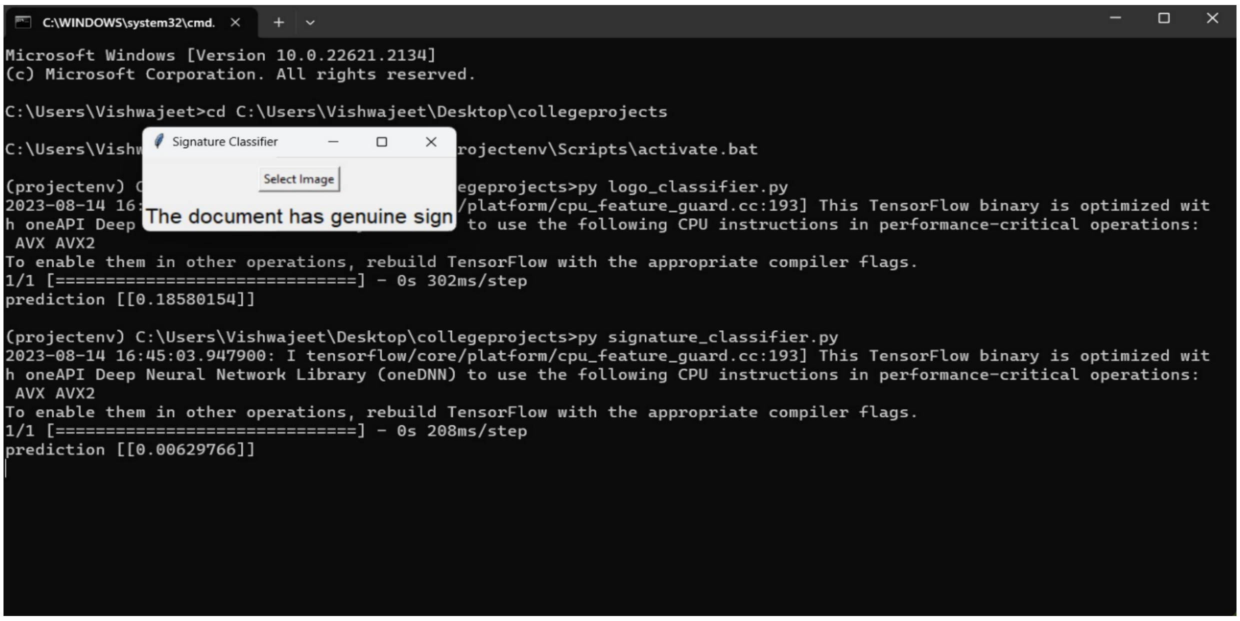
Output for logo detection



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vishwajeet>cd C:\Users\Vishwajeet\Desktop\collegeprojects

C:\Users\Vishwajeet\Desktop\collegeprojects>projectenv\Scripts\activate.bat

(projectenv) C:\Users\Vishwajeet\Desktop\collegeprojects>py logo_classifier.py
2023-08-14 16:44:14.224110: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [==============================] - 0s 302ms/step
prediction [[0.18580154]]
```

The document has genuine logo

Output for signature detection



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vishwajeet>cd C:\Users\Vishwajeet\Desktop\collegeprojects

C:\Users\Vishw                                 rojectenv\Scripts\activate.bat

(projectenv) C                                 egeprojects>py logo_classifier.py
2023-08-14 16:                                 /platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized wit
h oneAPI Deep                                  to use the following CPU instructions in performance-critical operations:
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [==============================] - 0s 302ms/step
prediction [[0.18580154]]

(projectenv) C:\Users\Vishwajeet\Desktop\collegeprojects>py signature_classifier.py
2023-08-14 16:45:03.947900: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [==============================] - 0s 208ms/step
prediction [[0.00629766]]
```

The document has genuine sign

CHAPTER-6

CONCLUSION

6.1 OUTCOME

In this project as the document image is considered to be of 3 parts the logo, text and signature all three components are subjected to reading and the logo and signatures are tested for forgery detection

The model can detect forgeries regarding only the logos and signatures which have already been stored in its dataset.

6.2 FUTURE SCOPE/ENHANCEMENT

The present project deals with using the cnn to train and identify only logos text and signatures after cropping them manually.

But the base has been set to improve the project to another level by using Custom Object Detection to identify the components of any kind of document like cheques, certifcates and many other documents which is possible by providing a sufficient and thorough dataset.

CHAPTER-7

REFERRNCES AND BIBLIOGRAPHY

- Websites referred www.stackoverflow.com

  www.pythonprogramming.net www.codecademy.com

  www.tutorialspoint.com www.geeksforgeeks.com www.wikipedia.org

  www.youtube.com

- Meritude Skills, Haegl Technology-guidance

.

YOUTUBE CHANEL NAME

- Itsourcecode