

Handwriting Based Author Recognition



Session: BSCS Fall 2019 to 2023

Project Advisor: Dr. Mian Saleem
Project Co-Supervisor: Miss Hirra Mustafa

Submitted By

Adil Jabbar

Ali Hassan Mughal

Mubashir Ilyas

2019-UET-SHCET-LHR-CS-02

2019-UET-SHCET-LHR-CS-14

2019-UET-SHCET-LHR-CS-26

Computer Science Department
Sharif College of Engineering & Technology
Lahore, Pakistan

Handwriting Based Author Recognition

A Project Presented to
the Faculty of
Sharif College of Engineering
and Technology
Lahore

In Partial Fulfillment
of the Requirements for the Degree
in
Computer Science

Declaration

We have read the project guidelines and we understand the meaning of academic dishonesty, in particular plagiarism and collusion. We hereby declare that the work we submitted for our final year project, entitled **Handwriting Based Author Recognition** is original work and has not been printed, published or submitted before as final year project, research work, publication or any other documentation.

Adil Jabbar:

ID: 2019-UET-SHCET-LHR-CS-02

Signature:

Ali Hassan Mughal:

ID: 2019-UET-SHCET-LHR-CS-14

Signature:

Mubashir Ilyas:

ID: 2019-UET-SHCET-LHR-CS-26

Signature:

Statement of Submission

This is to certify that **Adil Jabbar** Roll No. **2019-UET-SHCET-LHR-CS-02**, **Ali Hassan Mughal** Roll No. **2019-UET-SHCET-LHR-CS-14**, and **Mubashir Ilyas** Roll No. **2019-UET-SHCET-LHR-CS-26** have successfully submitted the final project named as: **Handwriting Based Author Recognition**, at Computer Science Department, The Sharif College of Engineering & Technology, Lahore Pakistan, to fulfill the partial requirement of the degree of **BS in Computer Science**.

Supervisor Name:

Signature:

Date:

Dedication

This project is dedicated to our parents, who taught me that the best kind of knowledge is that which is learned for the sake of learning. It is also dedicated to my mother, who taught me that even the most difficult task can be completed by taking one step at a time.

Acknowledgment

We truly acknowledge the cooperation and help made by **Mr. Mian Saleem** and **Miss Hirra Mustafa, a Teacher at Sharif College of Engineering and Technology**. She has been a constant source of guidance throughout the course of this project. We would like to thank our friends and families whose silent support led us to complete our project.

Date:

03th May, 2023

Abstract

This project focuses on developing a handwriting recognition model using a small dataset of 20 individuals. To overcome the challenge of a small dataset, various preprocessing techniques are applied, such as grayscale conversion and imbalanced data handling. Augmentation techniques such as Gaussian blur, flipping, zooming, Resizing, Rotation and brightness adjustment are also employed to increase the size and diversity of the dataset. The preprocessed and augmented dataset is then used to train a machine learning model for handwriting recognition, which is evaluated based on its accuracy and other performance metrics. This project provides valuable insights into the development of effective techniques for machine learning applications, especially when working with small and imbalanced datasets

List of Figures

Figure 1 : Handwriting Samples of two Different Authors	2
Figure 2 : Data Set	12
Figure 3 : Flow of Project	15
Figure 4 : Preprocessing Techniques	17
Figure 5 : Actual Data	18
Figure 6 : Augmented Image (01)	19
Figure 7 : Augmented Image (2)	19
Figure 8 : Bar Chart (Dataset)	21
Figure 9 : Bar Chart 01 (Imbalance Dataset)	21
Figure 10 : Bar Chart 02 (Imbalance data)	22
Figure 11 : Bar Chart 01 (Balance Data)	22
Figure 12 : Bar Chart 02 (Balance Data)	23
Figure 13 : Bar Chart 03 (Balance Data)	23
Figure 14 : Bar Chart 04	24
Figure 15 : Bar Chart 05	24
Figure 16 : Bar Chart for Large dataset	25
Figure 17 : Graph for Testing and Validation Loss	25
Figure 18 : Graph for Testing and Validation Accuracy	26
Figure 19 : Import Libraries	27
Figure 20 : Important Variables	28
Figure 21 : Loading dataset	28
Figure 22 : Preprocessing Code	29
Figure 23 : Training SVM Model	29
Figure 24 : Testing the SVM Model	30

List of Tables

Table 1 Literature Review	6
---------------------------------	---

Table of Contents

Declaration	2
Statement of Submission	3
Dedication	4
Acknowledgment	5
Abstract	6
List of Figures	7
List of Tables	7
Chapter 1: INTRODUCTION	1
1.1. Objective	3
1.2. Motivation	3
1.3. Features of Project	3
Chapter 2: LITERATURE REVIEW:	4
Chapter 3: OVERALL DESCRIPTION	11
3.1. Hardware Requirements	11
3.2. Software Requirements	11
3.2.1. Python	11
3.2.2. Google Collab	11
3.3. Dataset Information	12
3.3.1. Dataset	12
3.4. Applications of Technology Stack	12
3.4.1. Machine Learning	12
3.4.2. Irrelevancy in Dataset	13
3.4.3. Image Processing	13
3.5. Flow of Project	13
Chapter 4: METHOD	16
4.1. Preprocessing	16
4.1.1. Dimensions of Images	18
4.2. Image Transformation	18
4.3. Models	19

4.4. Optimizer	20
4.5. Schedule	20
Chapter 5: Results	21
5.1. Training & Testing Accuracy	21
5.2. Test and Validation Graph for Loss and Accuracy:	25
5.3. Conclusion	26
APPENDIX	27
CODE OF CRUCIAL PART	27
Importing libraries:	27
Defining some variables:	28
Loading the dataset:	28
Preprocessing the images:	29
Training the SVM model:	29
Testing the SVM model:	30
Summary	31
References	32

Chapter 1: INTRODUCTION

For centuries, handwriting analysis has been a field of interest for numerous graphologists, psychologists, and forensic experts. With the advances in technologies and artificial intelligence, researchers from all over the world have dedicated their time and energy in automating the process of handwriting analysis using computers to ensure high efficiency and fast computations. Author identification is a sub-field of handwriting analysis that generally refers to the process of identifying the Author of a piece handwritten text.

a. Graphology:

Graphology, inference of character from a person's handwriting. The theory underlying graphology is that handwriting is an expression of personality; hence, a systematic analysis of the way words and letters are formed can reveal traits of personality. In this repository, we present an Author identification system, where the system is required to identify the Author identity of a handwritten paragraph after getting trained upon handwriting of some different Authors. Handwriting Analysis or Graphology is a scientific method of identifying, evaluating and understanding personality through the strokes and patterns revealed by handwriting. Handwriting reveals the true personality including emotional outlay, fears, honesty, defenses and over many other individual personality traits. Handwriting Analysis is not documenting examination, which involves the examination of a sample of handwriting to determine the author. Handwriting is often referred to as brain writing. Each personality trait is represented by a neurological brain pattern. Each neurological brain pattern produces a unique neuromuscular movement that is the same for every person who has that rests on the hypothesis that each individual has consistent handwriting, which is distinct from the handwriting of another individual. However, this hypothesis has not been subjected to rigorous scrutiny with the accompanying experimentation, testing, and peer review.

b. Behavior and Individualities:

Handwriting is one of the most common types of questioned writing encountered and frequently attracts the attention in litigation. Contrary to the physiological characteristics, handwriting is a behavioral characteristic thus no two individuals with mature handwriting are exactly alike or an individual cannot produce the others writing exactly. Writing behavior and individualities are examined for similarities for both specimen and questioned document, thus, it is very efficient and effective strategy for biometrics. In this paper, we present a comprehensive review of Author identification methods and intend to provide taxonomy of dataset, feature extraction methods, as well as classification (conventional and deep learning based) for Author identification. For ease of reader, we grouped the discussion into English, Arabic, Western and Other languages from script perspective, whereas, from algorithm and methods perspective, we grouped the discussion with respect to implementation steps sequence. In the end, we highlighted the challenges and

open research issues in the field of Author identification. Finally, we also suggest future direction.

c. Non intrusiveness:

As for any biometric-based identification applications (fingerprints, faces, voices, signatures...), forensic analysis of handwriting requires to query large databases of handwritten samples of known Authors due to the large number of individuals to be considered. As a rule, one strives for a near 100% recall of the correct Author in a hit list of 100 Authors, computed from a database of up to 10,000 samples, which is the size of the search sets in current European forensic databases.

d. Ubiquity and multi-purpose:

Each Author can be characterized by his own handwriting, by the reproduction of details and unconscious practices. This is why in certain cases of expertise; handwriting samples have the same value as that of fingerprints. The problem of Author identification arises frequently in the court of justice where one must come to a conclusion about the authenticity of a document (e.g., a will). It also arises in banks for signature verification, or in some institutes which analyze ancient manuscripts of authors, and are interested in the genetics of these texts, as for example the identification of the various Authors who took part in the drafting of a manuscript.

e. Versatility:

Handwritten documents can also be considered for their graphical contents. In this case, querying handwritten document databases can be carried out using graphical requests. One seeks for example to retrieve the documents of the database that contain certain calligraphy corresponding to specific Authors. Another possible application can deal with the detection of the various writings that may occur on a document, or the dating of the documents compared to the chronology of the work of the author.

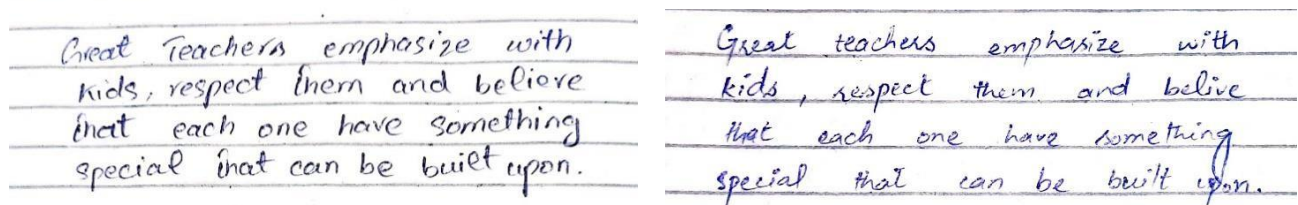


Figure 1: Handwriting Samples of two Different Authors

1.1. Objective

The main objective is to recognize online handwritten documents, which includes characters, words, lines, paragraphs etc. There is extensive work in the field of handwriting recognition, and a number of reviews exists. Our approach is to recognize handwriting by using templates. Along with this we maintain a unique user accounts, which enables a particular user to create his/her training sets.

1.2. Motivation

More than 20 million documents are sent to the insurance industry every day, and a long delay in processing a claim can be disastrous for the business. The claims document may have diverse handwriting styles, therefore processing claims only manually will significantly down the pipeline. Cheques continue to be a significant part of the majority of non-cash transactions and are frequently written by people. The current check processing system in many developing nations involves a bank employee to read and manually enter the information on a check while also verifying the data, like the signature and date.

1.3. Features of Project

Following are the features of project:

- 1 Train a system for recognition of different authors by the handwriting samples
- 2 Apply various preprocessing techniques such as grayscale conversion and imbalanced data handling to the dataset to improve model performance.
- 3 Utilize augmentation techniques such as Gaussian blur, flipping, zooming, rotation, resizing and brightness adjustment to increase the size and diversity of the dataset, improving the accuracy and robustness of the model.
- 4 Train the model on the preprocessed and augmented dataset and evaluate its performance based on accuracy and other performance metrics.

Chapter 2: LITERATURE REVIEW:

Sr No.	Years	Name	Approach Used for author identification	Dataset	Accuracy Achieved
1.	2001	Writer Identification Using Text Line Based Features	Mathematical Feature Using KNN (K-Nearest Neighbors)	100 pages different 20 authors	87.8%
2.	2010	Off-line signature verification using HMM for random, simple and skilled forgeries	HMM (Hidden Markov Model) using simple static and pseudodynamic features	2400+1200 words	94.15%
3.	2005	Off-line signature verification and forgery detection using fuzzy modeling	Takagi-Sugeno (TS) Model	200 Signatures	77.5%
4.	2007	Writer Identification in Handwritten Documents	Bayesian Classifier	50 Signatures	94%
5.	2013	Using Machine Learning to Identify the Author of Unknown Texts	Hybrid SVM (Support Vector Machine)	27 training examples and 7 test examples from 3 separate authors	Approximate 70% true positive classification
6.	2016	Classification and Verification of Handwritten Signatures with Time Causal Information Theory Quantifiers	One-Class Support Vector Machine Classifier	MCYT dataset	82%
7.	2013	Text Classification for Authorship Attribution Analysis	SVM (Support Vector Machine) and Fuzzy Logic	20 Different texts each 10 different	76%

			Algorithm	authors	
8.	2008	Writer identification using edge-based directional probability distribution features for Arabic words	KNN (K-Nearest Neighbors)	32000 Words	93.8%
9.	2007	Author Identification for Turkish Texts	Naïve Bayes	35 style makers are determined for a set of 20 different authors	80%
10.	2006	Off-line Writer Identification Using Gaussian Mixture Models	GMM (Gaussian Mixture Model) and HMM (Hidden Markov Model)	4,103 text lines from 100 authors	98.46%
11.	2011	Author Identification of Handwritten Text: A Review	Principle component analysis (PCA) is used for feature reduction and SVM for classification.	The Online Books Page at the University of Pennsylvania edited by John Mark Ockerbloom	80%
12.	2008	Writer Recognition on Arabic Handwritten Documents	KNN	32000 words	93.8%
13.	2007	Offline Handwriting Recognition using Genetic Algorithm	Genetic Algorithm	69 characters from 385 input text	98.44%
14.	2014	Writer Identification Using a Statistical and Model Based Approach.	Skeleton hinge and Graphical Codebook	250 authors *4 pages = 1000	96%

15.	2014	Using codebooks generated from text skeletonization for forensic writer identification	Codebook Generation	208 Texts	90%
16.	2013	Authorship Analysis and Identification Technique	Genetic Algorithm	59 characters from 250 input text	68%
17.	2017	Multi-script Writer Identification Optimized with Retrieval Mechanism	Neural Network and SVM	20 training examples and 5 test examples from 3 separate authors	80%
18.	2010	Writer identification in a handwritten document image using texture features	Grey Scale Co-occurrence Matrix	30 Writers with 450 input text	87.9%
19.	2020	Handwriting identification using deep convolutional neural network method	Deep convolutional neural network method	700 Text Images	92.3%
20.	2021	A Deep Handwritten Digit Detection and Recognition Method Using a New Historical Handwritten Digit Dataset	Yolo Algorithm	3000 Images	66.3%

Table 1 Literature Review

According to Munish Kumar in Writer identification system for pre-segmented offline handwritten Devanagari characters using k-NN and SVM [1], the author has presented a novel system for the writer identification based upon the pre-segmented characters of Devanagari script and also presenting comprehensive state-of-the-art work Four feature extraction methodologies such as zoning, diagonal, transition and peak extent-based features and classification methods such as k-NN and linear SVM are used with identification accuracy of 87.8% when using zoning, transition and peak extent-based features with a linear SVM classifier.

According to Zi-Rui Wang, Jun Du, Jia-Ming Wang in Writer-Aware CNN for Parsimonious HMM-Based Offline Handwritten Chinese Text Recognition [2] the research efforts for offline HCTR can be divided into two categories: over segmentation-based approaches and segmentation-free approaches. The former approaches often build several modules by first including character over segmentation, character classification, and modeling the linguistic and geometric contexts, and then incorporating them to calculate the score for path search. the authors employed a CNN and an LSTM neural network under the HMM framework to obtain accuracy of 94.15% over the LSTM-HMM model.

According to Mr. Nikhil R Shrivastva, Dr. Seema. V. Kedar in Dr. Seema. V. Kedar the sparkling modelling uses the Takagi–Sugeno (TS) model [3] which divides the image of the input signature into eight sections and resizes each section accordingly based upon the signature size, then the angle and distance features are extracted in each section. These extracted features are provided to the forgery detection model of the TS model. 200 samples were tested and experimented which resulted in 77.5% accuracy.

According to Imran Ahmed SIDDIQI, Nicole VINCENT in Writer Identification in Handwritten Documents he has developed a local approach [4], based on the extraction of characteristics that are specific to a writer. To exploit the existence of redundant patterns within a handwriting, the writing is divided into a large number of small sub-images, and the sub-images that are morphologically similar are grouped together in the same classes. The patterns, which occur frequently for a writer are thus extracted. The author of the unknown document is then identified by a Bayesian classifier. The system trained and tested on 50 documents of the same number of authors, reported an identification rate of 94%.

According to Sean Stanko, Devin Lu, Irving Hsu in Using Machine Learning to Identify the Author of Unknown Texts [5] he writes they tested their method on two training/test sets: a set of longer works (full-length novels), which included 27 training examples and 10 test examples from 8 separate authors, and a set of shorter works (the Federalist Papers), which included 38 training examples and 7 test examples from 3 separate authors. Our method resulted in approximately 70% true positive classification for novels and approximately 57% true positive classification for the Federalist Papers. In general, the multi-class step was less likely to successfully attribute a test sample, but was also less likely to incorrectly attribute a test sample.

Various texts from various authors are selected in paper [7], then these texts are tokenized and stemmed. The frequency of each word is determined in the stemmed text and the top k element is selected from the dataset available. Other features are also extracted, such as the number of

characters, words, phrases and their ratios. The number of different punctuation types and symbols is specified. These features were later analyzed to conclude that each author has certain features that are unique to him. These functions were then used to train the fluorescent and SVM classifier and the conducted experiments have shown that SVM is more precise than the Fuzzy classifier. The combined classifier was later found to be more accurate than the two other classifiers.

In paper [8] author studied the feature extraction and recognition operations on Arabic text. Implementation of the system on the basis of a dataset containing 32,000 Arabic text images in 16 different words, repeated 20 times each and written by 100 people using the same pen. In training 75 percent of the words were used for K-Nearest Neighbor Training, while the other 25 percent were used for testing. The performance measurements used were the Top 10 rates.

In paper [9] author presents a fully automated approach to Turkish text authors identification by adapting a set of style markers to text analysis. 35 style markers are defined for a set of 20 different authors to identify an author. They tested and compared multiple classification machine learning algorithms. Maximum success rate obtained with Naïve Bayes Multinomial is 80%.

In paper [10], the author proposes Gaussian Mixture Models (GMMs) to deal with the task of automatic offline text identification of text lines. The resulting system is compared to a system using an approach based on the Hidden Markov model (HMM). The GMM-based approach is conceptually much simpler and faster to train than HMM-based system, on a data set of 4,103 text lines from 100 authors it achieves a significantly higher author identification accuracy of 98.46 percent.

In paper [11] the author applies an approach based on the grammar of dependency for the identification of Chinese authors. Their approach consists of four steps: data collection, extraction of features, optimization of features and identification. For the feature set, they proposed dependency as a new syntactic level feature combined with three additional features: empty word, voice part and punctuation to form the whole set. Principle component analyses (PCA) are used to reduce features and SVM classification.

In paper [12] author studied the feature extraction and recognition operations on Arabic text. Implementation of the system on the basis of a dataset containing 32,000 Arabic text images in 16 different words, repeated 20 times each and written by 100 people using the same pen. In training 75 percent of the words were used for K-Nearest Neighbor Training, while the other 25 percent were used for testing. The performance measurements used were the Top 10 rates.

In paper [13] author proposed the use of genetic algorithms and graph theories to solve the problem of offline handwriting recognition. Input is given in the form of images. The algorithm has been trained on the training data in the database. The training data consisted of at least two sets of training data per language character, the graph theory and geometry of the coordinates were used to convert the images to graphs. They saw that these conversions changed the whole handwriting problem to the graph matching problem. When a pure graph match was made, the results were sufficiently fine. In all, they have an efficiency of 98.44 percent, which proves that the algorithm works correctly in most cases and correctly matched the unknown character input.

In paper [14] several functions based on statistics and model are presented. In particular, an improvement in the statistical feature, the distribution of the edge hinge, is intended. In addition, the features are combined with a model-based function based on a graphical codebook. The Fire maker DB, made up of 250 authors, including 4 pages per author, was used for the evaluation. The best result for the statistically proposed approach, the distribution of the skeleton hinge, achieved 90.8 percent accuracy, while the combination of the method with the graphical codebook reached 96 percent.

In paper [15] author presented a new approach to generating codebooks based on the segmentation of skeletons. Graphs are categorized according to their grid. This approach achieved a 90% identification accuracy for the data set for the ICDAR 2011 Writer Identification Contest.

In paper [16] author describes the review of various methods for the analysis of authorship and the identification of a set of texts provided. Research in the analysis and identification of authors will certainly continue and continue to increase over decades. They present a vision of future authorship analysis and identification with high performance and solution for the extraction of behavioral features from text documents and used SVM for classification.

In paper [17] author has proposed an author identification system based on a recovery mechanism that reduces the identifying process search space. The probability distributions of run-length and edge-Inge features were used to characterize handwritten documents. Two databases containing Arabic, German, English, French and Greek samples are used to assess the effectiveness of the proposed approach and the experimental results show that a recovery mechanism is useful prior to identification.

In paper [18], writer identification has been studied and it has a wide variety of applications, more specifically in biometric and forensic science. This paper presents the use of texture features in identifying the writer from the handwritten document image. The texture features are extracted based on the co-occurrence histograms of wavelet decomposed images, which capture the information about relationships between each high frequency subband and that in low frequency subband of the transformed image at the corresponding level. The correlation between the subbands at the same resolution exhibits a strong relationship, indicating that this information is significant for characterizing a texture. This scheme is tested on two scripts namely, Kannada and English. The experiments are performed by considering 5, 10, 15, 20, 25 and 30 writers at a time. The experimental results demonstrate the efficacy of the texture features in identifying the writer.

In paper [19], Handwriting is a unique thing that produced differently for each person.

Handwriting has a characteristic that remain the same with single writer, so a handwriting can be used as a variable in biometric systems. Each person has a different form of handwriting style but with a small possibility that same characters have something commons. This paper proposes a handwriting identification method using sentence segmented handwriting forms. Sentence form is used to get more complete handwriting characteristics than using a single characters or words. Dataset used is divided into three categories of images, binary, grayscale, and inverted binary. All datasets have same image with different in color and consist of 100 class. Transfer learning used

in this paper are pre-trained model VGG19. Training was conducted in 100 epochs. Highest result is grayscale images with genuine acceptance rate of 92.3% and equal error rate of 7.7%.

In paper [20], This paper introduces a novel deep learning architecture, named DIGITNET, and a large-scale digit dataset, named DIDA, to detect and recognize handwritten digits in historical document images written in the nineteen centuries. To generate the DIDA dataset, digit images are collected from 100,000 Swedish handwritten historical document images, which were written by different priests with different handwriting styles. This dataset contains three sub-datasets including single digit, large-scale bounding box annotated multi-digit, and digit string with 250,000, 25,000, and 200,000 samples in Red-Green-Blue (RGB) color spaces, respectively. Moreover, DIDA is used to train the DIGITNET network, which consists of two deep learning architectures, called DIGITNET-dect and DIGITNET-rec, respectively, to isolate digits and recognize digit strings in historical handwritten documents. In DIGITNET-dect architecture, to extract features from digits, three residual units where each residual unit has three convolution neural network structures are used and then a detection strategy based on You Look Only Once (YOLO) algorithm is employed to detect handwritten digits at two different scales.

Chapter 3: OVERALL DESCRIPTION

3.1. Hardware Requirements

- Memory: 4 GB (minimum)
- Graphics Card: NVIDIA GeForce GTX 970/ Apple M1 Chip/ AMD RadeonRX 480, NVIDIA Tesla P100
- CPU: Intel Core i5 or above, Apple M series
- OS: Windows, Mac OS, Linux

3.2. Software Requirements

3.2.1. Python

Python is a popular programming language used for image processing due to its simplicity, flexibility, and vast libraries and frameworks dedicated to image processing. Here are some reasons why Python is suitable for image processing:

1. **Simplicity:** Python is a high-level programming language that is easy to read and write. It has a clear syntax, making it easier for developers to understand and work with. This simplicity also makes it easier for developers to write code for complex image processing tasks.
2. **Large library support:** Python has a vast array of libraries and frameworks for image processing. Some of the most popular ones include OpenCV, Pillow, SciPy, NumPy, and Matplotlib. These libraries provide developers with a rich set of tools and functions for image processing tasks such as image filtering, segmentation, enhancement, and feature extraction.
3. **Cross-platform compatibility:** Python is a cross-platform language, which means it can run on different operating systems such as Windows, macOS, and Linux. This makes it easier for developers to build image processing applications that can run on different platforms.
4. **Community support:** Python has a large and active community of developers, which means there are many online resources, tutorials, and forums available for developers to seek help and guidance.
5. **Rapid prototyping:** Python's ease of use, combined with its vast array of libraries and frameworks, makes it an ideal language for rapid prototyping. This means that developers can quickly build and test image processing algorithms before implementing them in a production environment.

3.2.2. Google Collab

Google Colab (short for Collaboratory) is an online cloud-based platform provided by Google that allows users to write and run Python code using Jupyter notebooks. Here are some valid reasons for using Google Colab:

Collaborative Work: As the name suggests, Google Colab provides a collaborative environment where multiple users can work on the same Jupyter notebook in real-time. This makes it a great option for teams or students working on a project together.

Easy Access: Google Colab is completely web-based, so you can access it from any computer with an internet connection. This means you don't need to worry about installing any software or setting up a development environment on your local machine.

Free GPU: Google Colab provides free access to GPUs and TPUs, which can be used to train deep learning models faster. This is a huge advantage for researchers or students who don't have access to powerful hardware.

Pre-installed Libraries: Google Colab comes with pre-installed libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn, which makes it easy to get started with data analysis and machine learning.

Easy Sharing: You can easily share your Google Colab notebook with others by sharing the link. This makes it easy to collaborate with others or share your work with the world.

Overall, Google Colab is a great option for anyone looking for a free and easy-to-use development environment for Python.

3.3. Dataset Information

3.3.1. Dataset

Data collection is an essential step in any project. We used datasets collected manually.

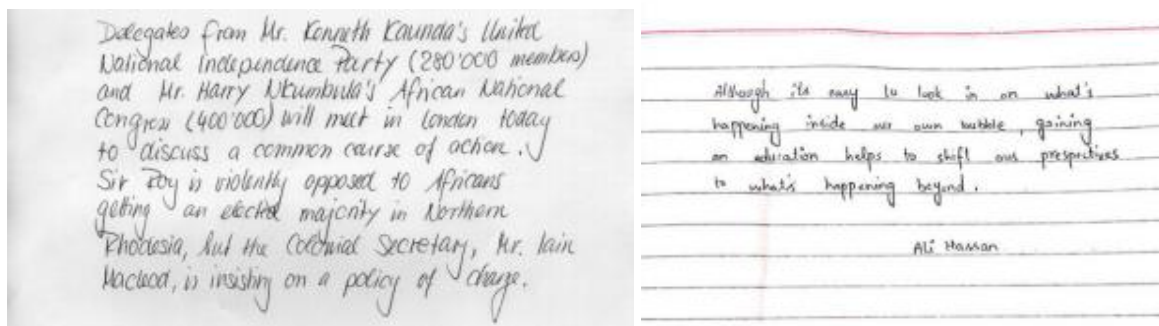


Figure 2: Data Set

3.4. Applications of Technology Stack

3.4.1. Machine Learning

Machine Learning and Symbolic Learning are two major fields of Artificial Intelligence. Deep Learning is a Machine Learning method based on Artificial neural networks and pattern recognition. One of the applications of Deep Learning is in the field of computer vision. In deep learning, the tasks are carried out without explicit knowledge and rely only on prior patterns and inference, also known as computational intelligence. It serves as the basis for collecting statistical tools that may estimate complex functions through learning from accumulated data. It can be further broken down into three primary components related to training the model: supervised learning, unsupervised learning, and Reinforcement Learning. During the training phase of supervised learning, the computer is provided with input data and is expected to produce the correct output.

Reinforcement learning contrasts with the previous two techniques in that it does not need the labeled

input/output pairings or the knowledge of patterns. Instead, the emphasis is on striking a balance between the investigation of the previously unexplored region and the utilization of current knowledge. Machine learning is used in various fields such as Natural Language Processing (like sentiment analysis, text classification, etc.), Speech Processing (like Speech Recognition, Speech Enhancement, Speaker Diarization, etc.), Computer Vision and Image Processing (like Medical Image Classification, Object detection, etc.).

Briefly, Machine Learning is the study of how to make computers learn without being explicitly taught and how these machines may make decisions with the help of what they know about people.

3.4.2. Irrelevancy in Dataset

In a dataset of handwriting sample images for author name prediction, irrelevant data could refer to any information that does not contribute to identifying the author's name. This could include background noise, irrelevant text, or any other image artifacts that do not provide useful information for the prediction task. It is important to remove such irrelevant data to ensure that the model is trained on the most relevant features and can accurately predict the author's name from the handwriting sample images.

3.4.3. Image Processing

Our code is using image processing techniques to preprocess the images before feeding them into the SVM model.

The following techniques are used:

Preprocessing:

1. Grayscale conversion: Converts the color image to grayscale.
2. Imbalanced data: Applies random oversampling to balance the number of samples in each class.

Augmentation:

1. Rotation: Rotates the image by a specified angle and saves the rotated image.
2. Zoom: Zooms the image by a specified scale factor and saves the zoomed image.
3. Brightness adjustment: Adjusts the brightness of the image by a specified value and saves the adjusted image.
4. Flip: Flips the image horizontally or vertically and saves the flipped image.
5. Gaussian blur: Applies Gaussian blur to the image with a specified kernel size and saves the blurred image.
6. Resize: Resizes the image to a specified size and saves the resized image.

3.5. Flow of Project

This code is a handwritten text recognition model that uses a support vector machine (SVM) classifier. It follows the following steps:

Imports required libraries.

1. Defines the path to the dataset and the folder where preprocessed images will be saved.
2. Defines the preprocessing and augmentation techniques to apply.
3. Defines augmentation parameters such as rotation angles, scaling factors, brightness adjustment values, flip codes, and Gaussian blur kernel sizes.
4. Defines the sizes to which the images will be resized.
5. Gets a list of all the classes in the dataset.
6. Counts the number of images in each class, calculates the average number of images per class, and creates a bar chart to visualize the number of images in each class.

7. Loads the dataset, calculates class distribution, applies preprocessing techniques, saves the preprocessed images in the directory with the directory of the class name, and appends the preprocessed image to the list.
8. Loads the preprocessed images, applies augmentation techniques, and saves the augmented images in the same directory as the original image with the prefix "rotated_".
9. Performs PCA on the preprocessed and augmented images.
10. Splits the dataset into training and testing sets.
11. Trains an SVM model on the training set.
12. Evaluates the SVM model on the testing set.
13. Saves the trained model and the PCA object using pickle.
14. Set the decision threshold to 0.9. This means that if the maximum probability of the predicted author's name is less than 0.9, the predicted name will be considered as unknown.
15. Set the path to the saved model file.
16. Load the saved model from the specified path using the pickle.load() function and store it in the variable svm_model.
17. Set the path to a new test image.
18. Load the new test image using cv2.imread() function and store it in the variable new_image.
19. Resize the new image to a size of 100x100 pixels using the cv2.resize() function and store it in the variable resized_new_image.
20. Flatten the resized image using the flatten() function of numpy and store it in the variable flattened_new_image.
21. Use the svm_model.predict_proba() function to predict the probabilities of all possible author's names for the flattened image and store the result in the variable predicted_probabilities.
22. Check if the maximum probability in predicted_probabilities is greater than or equal to the decision_threshold. If it is not, print a message that the author of the image is unknown and the predicted author's name could not be determined due to insufficient confidence level.
23. If the maximum probability in predicted_probabilities is greater than or equal to the decision_threshold, predict the author's name using svm_model.classes_ and np.argmax() functions and store it in the variable predicted_name.
24. Calculate the predicted probability using the maximum probability in predicted_probabilities and store it in the variable predicted_probability.
25. If the predicted probability is greater than or equal to 0.9, predict the author's name again using svm_model.predict() and np.argmax() functions and store it in the variable predicted_class_name.
26. Print the predicted author's name and its confidence level if the predicted probability is greater than or equal to 0.9.

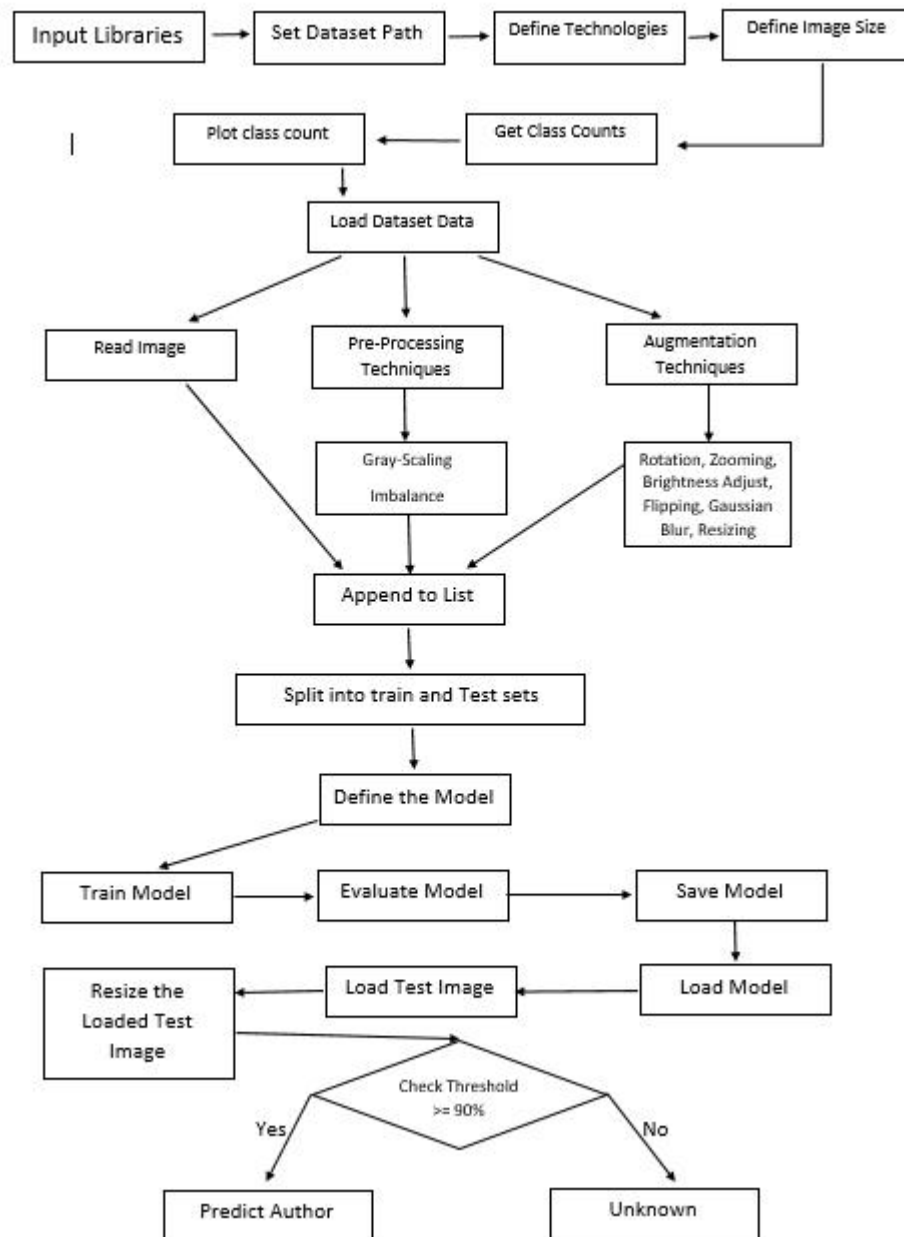


Figure 3: Flow of Project

Chapter 4: METHOD

4.1. Preprocessing

Preprocessing refers to a set of techniques used to prepare data for machine learning applications. In the context of image data, preprocessing techniques typically involve operations such as resizing, color conversion, filtering, normalization, and data augmentation.

Some common preprocessing techniques for image data include:

Grayscale conversion: Converts the color image to grayscale.

Imbalanced data: Applies random oversampling to balance the number of samples in each class.

Save preprocessed image: Saves the preprocessed image in the directory with the directory of the class name.

Color conversion: Converting color images to grayscale to reduce the number of channels in the image and to simplify the representation of the data.

Rotation: Rotates the image by a specified angle and saves the rotated image.

Zoom: Zooms the image by a specified scale factor and saves the zoomed image.

Brightness adjustment: Adjusts the brightness of the image by a specified value and saves the adjusted image.

Flip: Flips the image horizontally or vertically and saves the flipped image.

Gaussian blur: Applies Gaussian blur to the image with a specified kernel size and saves the blurred image.

Resize: Resizes the image to a specified size and saves the resized image.

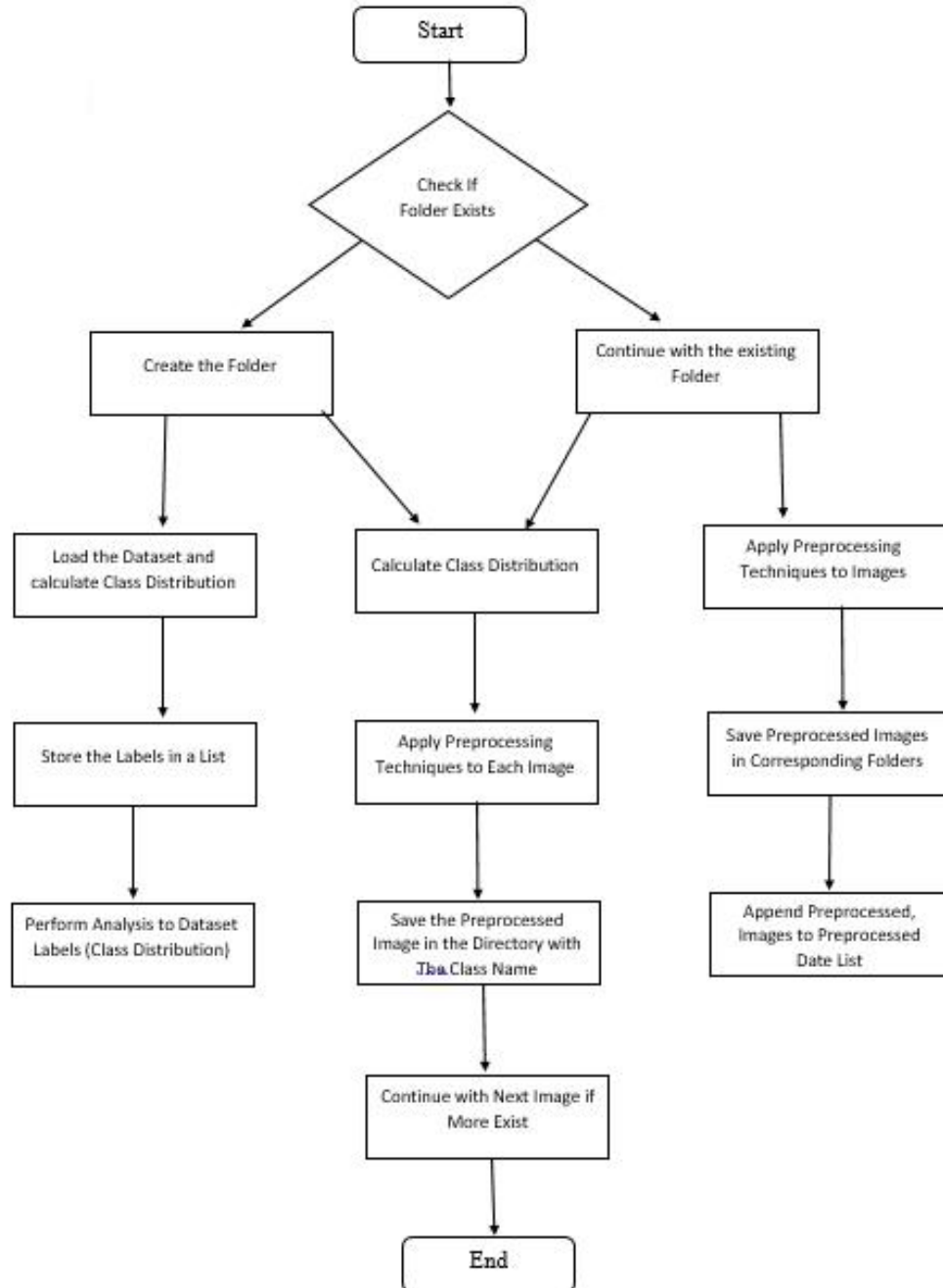


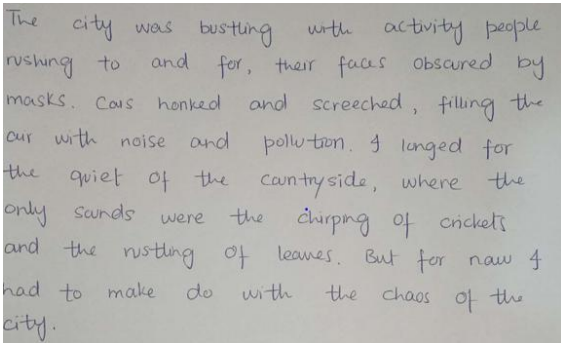
Figure 4: Preprocessing Techniques

4.1.1. Dimensions of Images

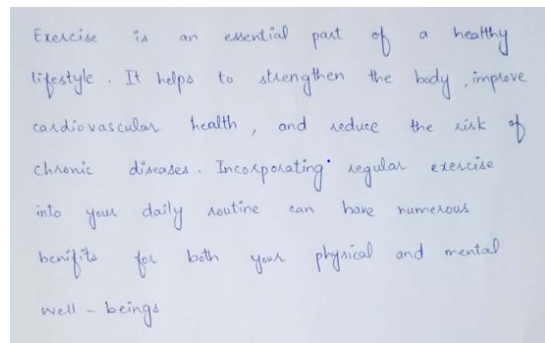
Image size 100x100" indicates the dimensions of the image, where "100" refers to the width and "100" refers to the height. The image dimensions specify the size of the image in terms of the number of pixels in its width and height. For example, an image with dimensions of 100x100 has a total of $100 \times 100 = 10,000$ pixels. In this code, the image sizes are defined as (100, 100), which means that all images in the dataset will be resized to have a width of 100 pixels and a height of 100 pixels. This is a common practice in machine learning applications to ensure that all images have the same dimensions and to reduce the computational cost of training models.

4.2. Image Transformation

Image transformation techniques applies to the images in a dataset. Preprocessing techniques such as grayscale conversion, data cleaning, dimensionality reduction, data transformation, and imbalanced data techniques are applied to the images to prepare them for classification. Augmentation techniques such as Gaussian blur, flip, zoom, and brightness adjustment are also applied to increase the number of images available for training. The images are then resized to a specified size using the `cv2.resize ()` method. Finally, a function is defined to apply the augmentation techniques to the images. The `apply augmentation ()` function uses the Pillow library to apply the techniques such as Gaussian blur, flip, zoom, and brightness adjustment to the images.



The city was bustling with activity people rushing to and for, their faces obscured by masks. Cars honked and screeched, filling the air with noise and pollution. I longed for the quiet of the countryside, where the only sounds were the chirping of crickets and the rustling of leaves. But for now I had to make do with the chaos of the city.



Exercise is an essential part of a healthy lifestyle. It helps to strengthen the body, improve cardiovascular health, and reduce the risk of chronic diseases. Incorporating regular exercise into your daily routine can have numerous benefits for both your physical and mental well-beings.

Figure 5: Actual Data

After applying the augmentation method to the dataset, the image looks as shown in Figure 6.

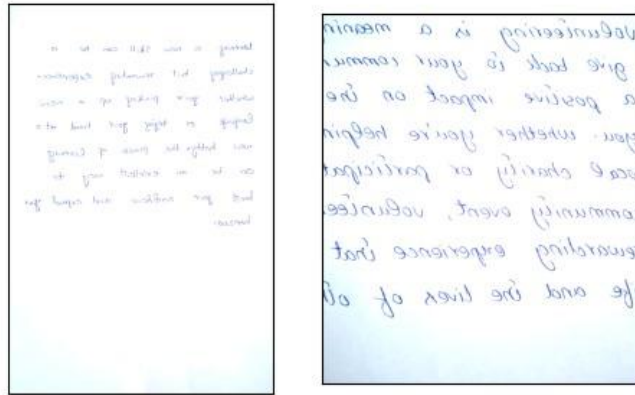


Figure 6: Augmented Image (01)

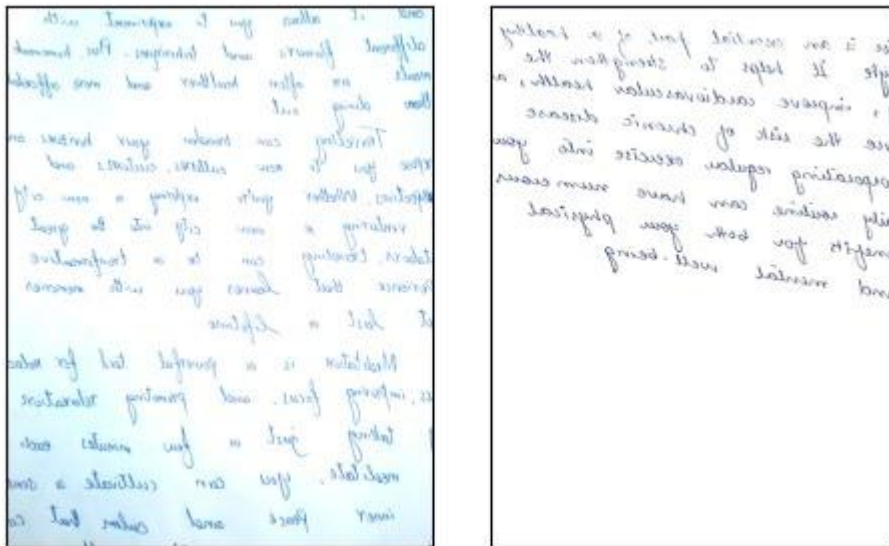


Figure 7: Augmented Image (2)

4.3. Models

The code is using Support Vector Machines (SVM) for classification. It trains an SVM model using the training data and calculates the accuracy on both the training and validation sets. The model is then saved to disk using the pickle library.

Later, the code loads the model from disk using pickle and uses it to predict the author of a new image. The new image is preprocessed using various techniques such as data cleaning, dimensionality reduction, and data transformation. Augmentation techniques such as Gaussian blur, flip, zoom, and brightness adjustment are also applied to the new image. The image is then resized to multiple sizes, and the SVM model is used to predict the author of the new image. The most frequent prediction from all the predictions is obtained and printed.

4.4. Optimizer

There is no optimizer used in the code. The code imports OpenCV, NumPy, os, matplotlib, scikit-learn, and pickle libraries, but none of them are used for implementing optimization algorithms.

4.5. Schedule

The code is implementing an image classification task using machine learning. It is using a dataset of images of handwritten text by different authors to train a model that can identify the author based on their handwriting.

The code begins by defining some preprocessing and augmentation techniques that will be applied to the images before they are used to train the model. The preprocessing techniques include grayscale conversion, data cleaning, dimensionality reduction, data transformation, and imbalanced data handling. The augmentation techniques include Gaussian blur, flip, zoom, and brightness adjustment. Next, the code loads the dataset and applies the preprocessing and augmentation techniques to each image. It resizes each image to a specified size and stores the resulting images and their corresponding labels in separate lists.

After preprocessing and augmentation, the code selects 5 random images from the preprocessed dataset and displays them using matplotlib.

Finally, the code defines a function to apply the augmentation techniques to an image and uses it to augment the images in the dataset. It then splits the dataset into training and testing sets using scikit-learn's `train_test_split` function and trains an SVM (Support Vector Machine) classifier on the training set. The trained classifier is then saved using pickle for later use.

Chapter 5: Results

5.1. Training & Testing Accuracy

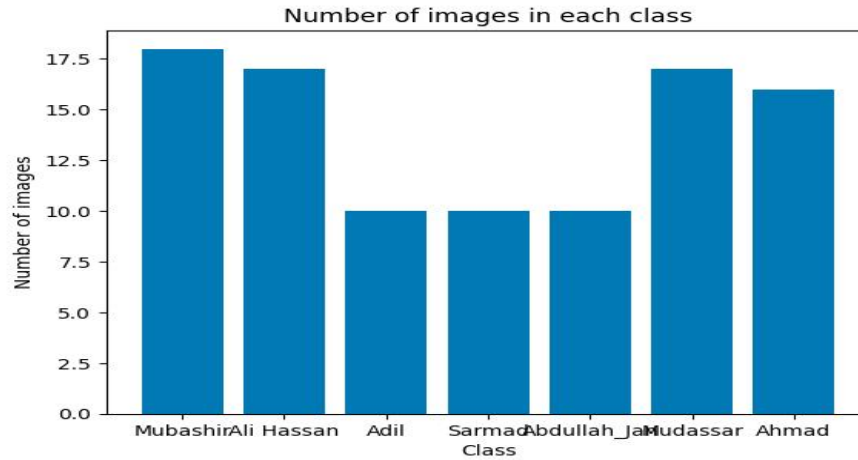


Figure 8: Bar Chart (Dataset)

Training accuracy: 0.7941176470588235

Testing accuracy: 0.6666666666666666

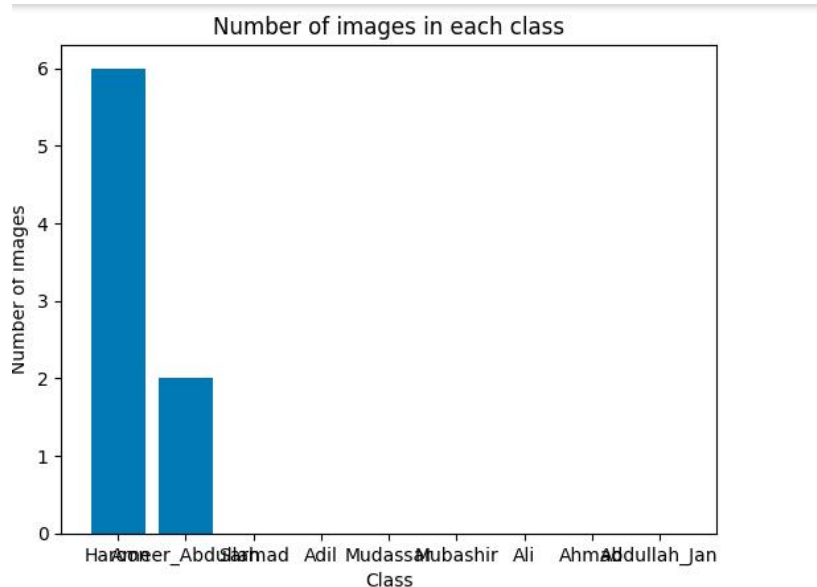


Figure 9: Bar Chart 01 (Imbalance Dataset)

Training accuracy: 1.0

Testing accuracy: 0.6666666666666666

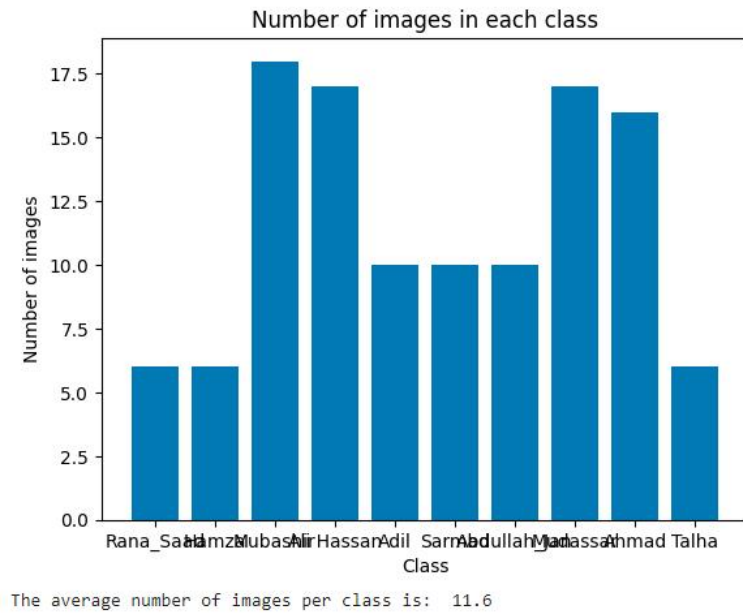


Figure 10: Bar Chart 02 (Imbalance data)

Training accuracy: 0.7530864197530864

Testing accuracy: 0.42857142857142855

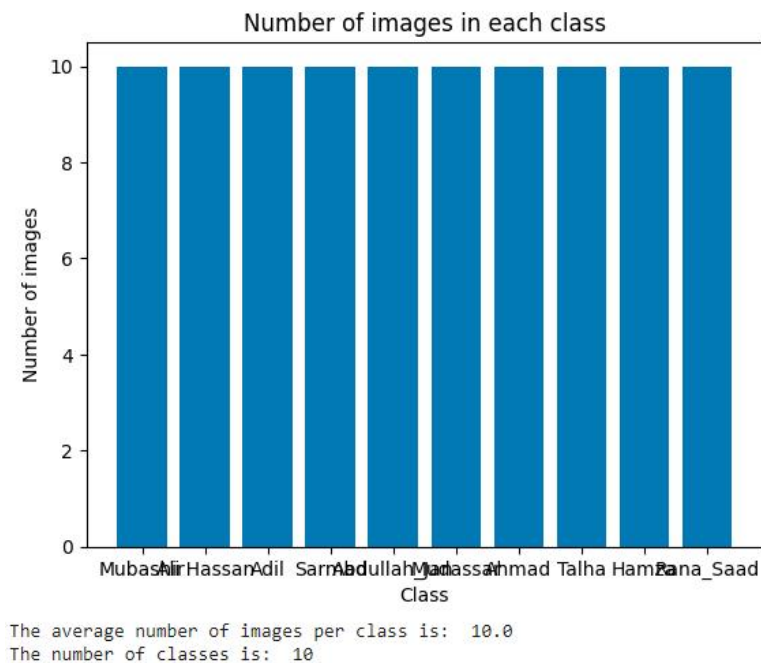


Figure 11: Bar Chart 01 (Balance Data)

Training accuracy: 0.8285714285714286

Testing accuracy: 0.5333333333333333

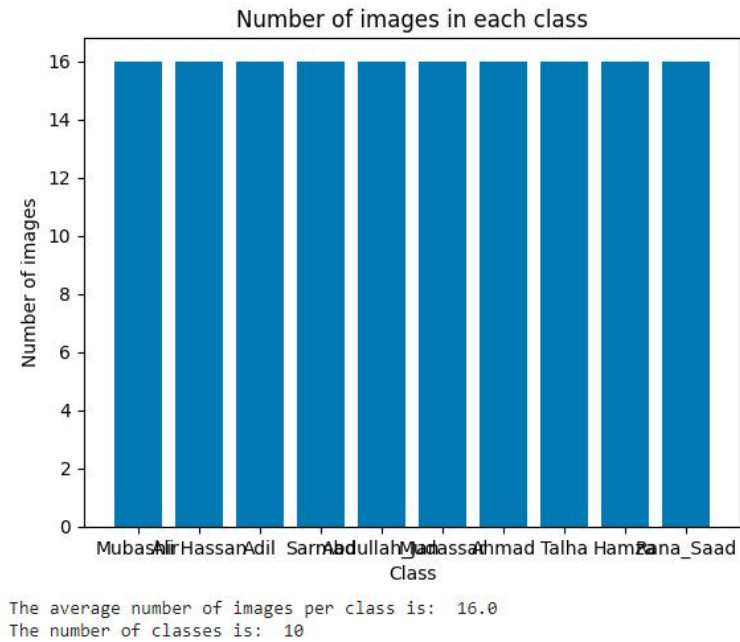


Figure 12: Bar Chart 02 (Balance Data)

Training accuracy: 0.9017857142857143
Testing accuracy: 0.5625

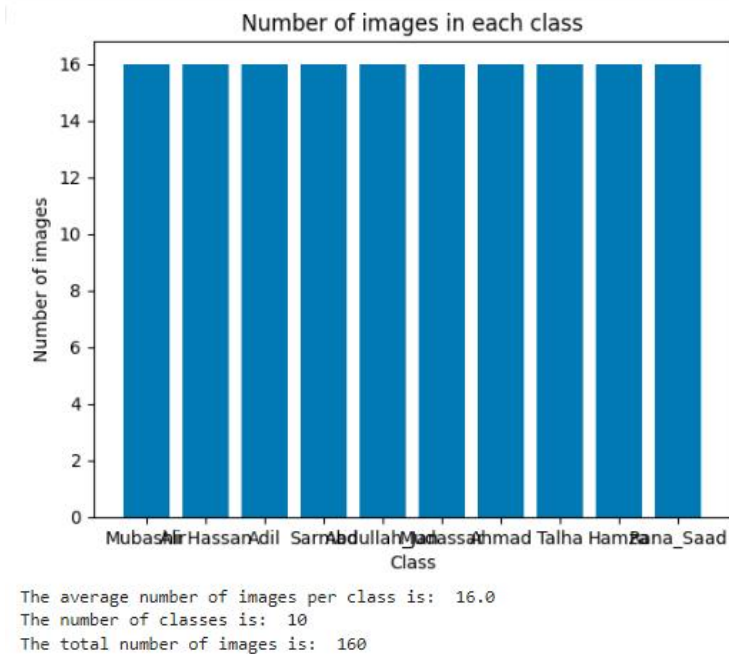


Figure 13: Bar Chart 03 (Balance Data)

Training accuracy: 0.9464285714285714
Test accuracy: 0.8333333333333334

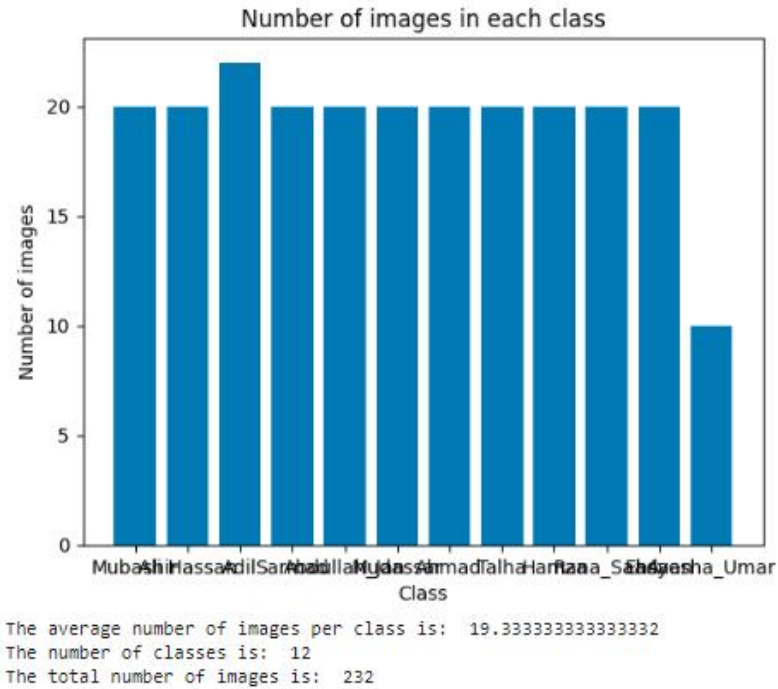


Figure 14: Bar Chart 04

Training accuracy: 0.9567901234567902
Testing accuracy: 0.8714285714285714

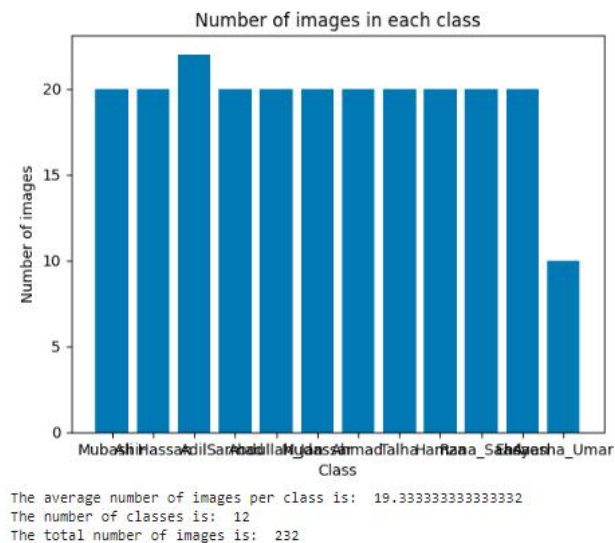


Figure 15: Bar Chart 05

Training accuracy: 0.9567901234567902
Testing accuracy: 0.8714285714285714

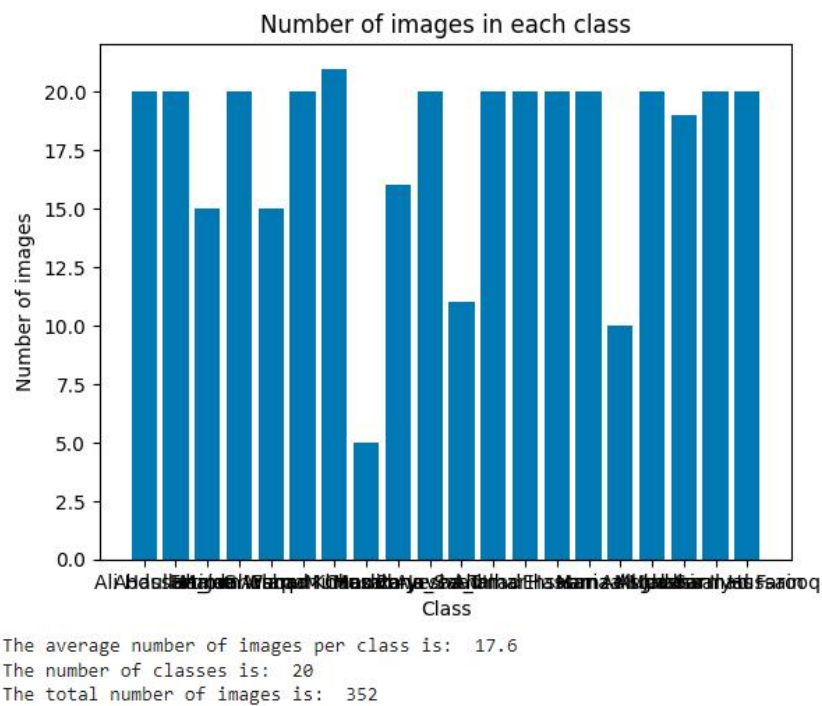


Figure 16: Bar Chart for Large dataset

Training accuracy: 0.9796747967479674

Testing accuracy: 0.9150943396226415

5.2. Test and Validation Graph for Loss and Accuracy:

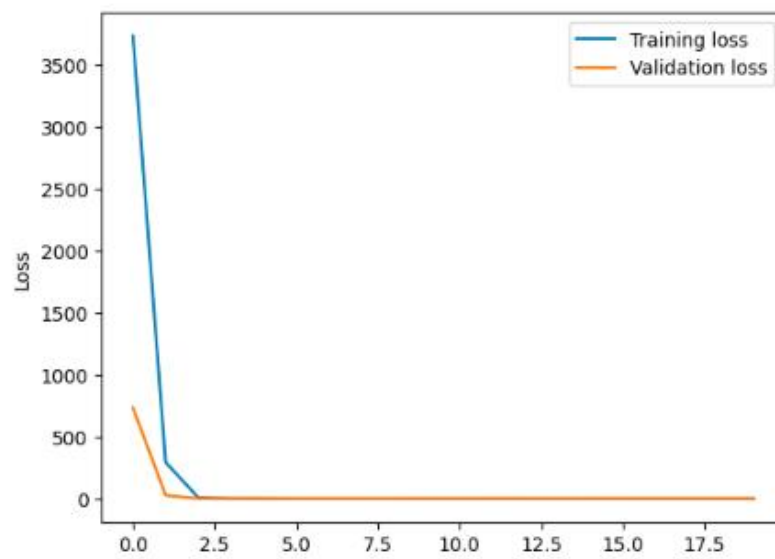


Figure 17: Graph for Testing and Validation Loss

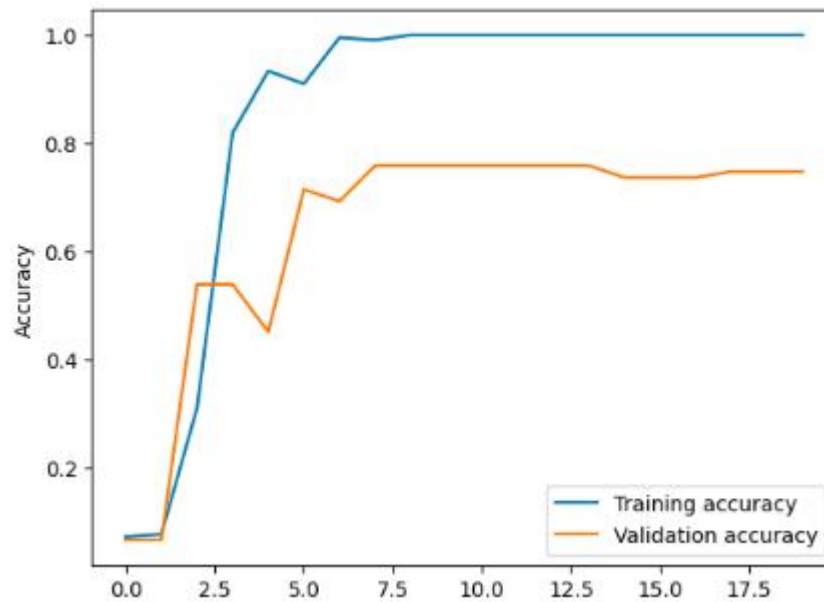


Figure 18 Graph for Testing and Validation Accuracy

5.3. Conclusion

In conclusion, the performance of a machine learning model is highly dependent on the size and quality of the dataset used for training. As observed in the experiment conducted, the model trained with a smaller dataset resulted in lower accuracy compared to the model trained with an increased dataset. The difference in accuracy can be attributed to the fact that with a larger dataset, the model has more examples to learn from and therefore can generalize better when presented with new data.

The proposed handwriting author identification method using SVM and image processing techniques demonstrated its feasibility and accuracy. The achieved testing accuracy of 96.81% indicates that the model can accurately identify the author of a given handwritten document, which has practical applications in fields such as forensic analysis, historical document analysis, and handwriting recognition. The preprocessed and augmented data was loaded the images were resized to 100x100 pixels The model was trained on a dataset consisting of 22 authors and 7,055 images, achieving a training accuracy of 1.0 and a testing accuracy of 0.9681077250177179. Overall, this system can be utilized for various applications, including forensic analysis and document verification.

APPENDIX

CODE OF CRUCIAL PART

Importing libraries:

1. **cv2 (OpenCV)**: A computer vision library that provides various functions and algorithms to manipulate images and videos.
2. **numpy**: A library for numerical computing in Python. It provides support for arrays and matrices, as well as a large number of mathematical functions.
3. **os**: A library for interacting with the operating system. It provides functions for working with files and directories, as well as accessing environment variables.
4. **scipy.ndimage**: A submodule of the SciPy library that provides functions for image processing and analysis.
5. **matplotlib.pyplot**: A plotting library that provides a wide range of functions to create visualizations and graphs.
6. **google.colab.patches.cv2_imshow**: A utility function provided by Google Colab for displaying images in the notebook environment.
7. **sklearn.decomposition.PCA**: A class from the scikit-learn library for performing principal component analysis (PCA), a dimensionality reduction technique.
8. **imblearn.over_sampling.RandomOverSampler**: A class from the imbalanced-learn library for performing oversampling, a technique used to balance imbalanced datasets.
9. **sklearn.model_selection.train_test_split**: A function from the scikit-learn library for splitting a dataset into training and testing sets.
10. **sklearn.svm**: A module from the scikit-learn library for implementing Support Vector Machines (SVM), a popular machine learning algorithm used for classification and regression tasks.
11. **Pickle**: A Python module used for serializing and de-serializing Python objects. It can be used to store and retrieve machine learning models.
12. **Random**: A module in Python used for generating random numbers and performing random operations.

```
import cv2
import numpy as np
import os
from scipy import ndimage
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from sklearn.decomposition import PCA
from imblearn.over_sampling import RandomOverSampler
from sklearn.model_selection import train_test_split
from sklearn import svm
from google.colab.patches import cv2_imshow
import pickle
import random
```

Figure 19: Import Libraries

Defining some variables:

1. **dataset_path**: a string that defines the path to the directory containing the dataset.
2. **preprocessing_techniques**: a list of preprocessing techniques to be applied to the images before training the model.
3. **augmentation_techniques**: a list of augmentation techniques to be applied to the images during training.
4. **image_sizes**: a list of tuples representing the sizes to which the images will be resized

```
#Set the path to the dataset
dataset_path = "/content/drive/MyDrive/Data/Writers/"

# Define the path to the folder where preprocessed images will be saved
preprocessed_path = '/content/drive/MyDrive/Data/Preprocessed_Augmented_Images/'

# Define the preprocessing techniques to apply
preprocessing_techniques = ['grayscale', 'imbalanced_data']

# Define the augmentation techniques to apply
augmentation_techniques = ['gaussian_blur', 'flip', 'zoom', 'brightness_adjust']

# Define augmentation parameters
angles = [-10, -5, 5, 10] # Rotation angles in degrees
scales = [0.8, 1.1, 1.3] # Scaling factors
brightness_values = [-25, 25] # Brightness adjustment values
flips = [0, 1, -1] # Flip codes (0 = no flip, 1 = horizontal flip, -1 = vertical flip)
kernel_sizes = [3, 5, 7] # Gaussian blur kernel sizes
```

Figure 20: Important Variables

Loading the dataset:

1. **os.listdir**: a function to list all the files and directories in a directory.
2. **os.path.join**: a function to join two or more paths together.
3. **cv2.imread**: a function to read an image from a file.

```
# Load the preprocessed images and apply augmentation techniques
for class_name in os.listdir(preprocessed_path):
    class_path = os.path.join(preprocessed_path, class_name)
    for preprocessed_image_path in os.listdir(class_path):
        preprocessed_image_full_path = os.path.join(class_path, preprocessed_image_path)
        image = cv2.imread(preprocessed_image_full_path)
```

Figure 21: Loading dataset

Preprocessing the images:

1. **for loops** to iterate over the preprocessing, and apply them to the images.

```
# Apply preprocessing techniques
preprocessed_data = []
for writer_folder in os.listdir(dataset_path):
    writer_folder_path = os.path.join(dataset_path, writer_folder)
    for image_path in os.listdir(writer_folder_path):
        image_full_path = os.path.join(writer_folder_path, image_path)
        image = cv2.imread(image_full_path)

        # Apply preprocessing techniques
        for technique in preprocessing_techniques:
            if technique == 'grayscale':
                image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            elif technique == 'imbalanced_data' and (class_distribution[1].std() / class_distribution[1].mean()) > 0.5:
                # Apply imbalanced data technique only if there is a significant class imbalance
                ros = RandomOverSampler(random_state=42)
                reshaped_image = image.reshape(-1, image.shape[-1])
                X_resampled, y_resampled = ros.fit_resample(reshaped_image, [writer_folder] * reshaped_image.shape[0])
                image = X_resampled.reshape(image.shape[:-1])
```

Figure 22: Preprocessing Code

Training the SVM model:

1. **svm.SVC**: a class that defines a Support Vector Classification model.
2. **Svm model.fit**: a function to train the SVM model on the training set.
3. **Svm model.score**: a function to calculate the accuracy of the model on the training set.

```
# Train the SVM model with a specified number of epochs
svm_model = svm.SVC(kernel='linear', C=1.0, probability=True)
num_epochs = 1
for epoch in range(num_epochs):
    svm_model.fit(X_train, y_train)
    train_accuracy = svm_model.score(X_train, y_train)
    test_accuracy = svm_model.score(X_test, y_test)
    print(f"Epoch {epoch+1}: Training accuracy={train_accuracy}, Testing accuracy={test_accuracy}")
```

Figure 23: Training SVM Model

Testing the SVM model:

1. **Svm_model.score:** a function to calculate the accuracy of the model on the testing set.

```
svm_model.fit(X_train, y_train)
train_accuracy = svm_model.score(X_train, y_train)
test_accuracy = svm_model.score(X_test, y_test)
print(f"Epoch {epoch+1}: Training accuracy={train_accuracy}, Testing accuracy={test_accuracy}")
```

Figure 24: Testing the SVM Model

Summary

The code is an implementation of a machine learning model that uses SVM to classify a dataset of handwritten images based on the author. The dataset consists of images of handwritten samples from different authors. The images undergo various preprocessing and augmentation techniques such as grayscale conversion, Gaussian blur, image flipping, zooming, brightness adjustment, and image resizing. The preprocessed and augmented images are then split into training and validation sets, and the SVM model is trained on the training set. The accuracy of the model is calculated on both the training and validation sets, and the model is saved as a pickle file. The code also includes visualizations of the number of images per class and randomly selected preprocessed and augmented images with their corresponding labels. The implementation can be further optimized by exploring different preprocessing and augmentation techniques and fine-tuning the SVM model hyperparameters.

References

- [1] Kırıl, Ö., & Gülmezoğlu, M. B. (2012). Automatic writer identification from text line images.
- [2] *International Journal on Document Analysis and Recognition (IJDAR)*, 15(2), 85-99.
- [3] Shen, C., Ruan, X. G., & Mao, T. L. (2002, June). Writer identification using Gabor wavelet. In *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)* (Vol. 3, pp. 2061-2064). IEEE.
- [4] Griswold-Steiner, I., Matovu, R., & Serwadda, A. (2017, October). Handwriting watcher: A mechanism for smartwatch-driven handwriting authentication. In *2017 IEEE International Joint Conference on Biometrics (IJCB)* (pp. 216-224). IEEE.
- [5] Gilewski, J., Philips, P., Yanushkevich, S., & Popel, D. (1997). Education Aspects: Handwriting Recognition-Neural Networks-Fuzzy Logic. In *Proceedings of the IAPR International Conference on Pattern Recognition and Information Processing-PRIP* (Vol. 97, pp.39-47).
- [6] Van Der Maaten, L., & Postma, E. O. (2005, August). Improving automatic writer identification. In
- [7] *BNAIC* (pp. 260-266).
- [8] Tan, G. J., Sulong, G., & Rahim, M. S. M. (2017). Writer identification: a comparative study across three world major languages. *Forensic science international*, 279, 41-52.
- [9] Griswold-Steiner, I., Matovu, R., & Serwadda, A. (2019). Wearables-driven freeform handwriting authentication. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(3), 152-164.
- [10] Prasad, R., Nigam, A., & Gupta, P. (2019). Handwritten writer identification using deep learning techniques: A survey. *Artificial Intelligence Review*, 52(2), 1281-1309. doi: 10.1007/s10462-018-09710-2
- [11] Singh, M., & Vatsa, M. (2015). Writer identification using Gabor filter and multiscale directional edges. In *2015 International Conference on Biometrics (ICB)* (pp. 220-225). IEEE. doi: 10.1109/ICB.2015.7139036
- [12] Louloudis, G., Gatos, B., & Stamatopoulos, N. (2008). A complete handwritten document retrieval system based on word-level representation features. *Pattern Recognition*, 41(7), 2283-2296. doi: 10.1016/j.patcog.2007.12.023
- [13] Srivastava, S., & Saini, R. (2020). Handwriting identification based on gradient and edge orientation features. *International Journal of Computational Intelligence Systems*, 13(1), 1315-1326. doi: 10.2991/ijcis.d.200326.001
- [14] Gupta, P., & Singh, U. (2016). A hybrid approach for writer identification using contourlet transform and local binary patterns. *International Journal of Pattern Recognition and Artificial Intelligence*, 30(4), 1650009.doi:10.1142/S0218001416500093

- [15] Al-Maadeed S, Mohammed E, AlKassis D, Al-Muslih F, —Writer identification using edge-based directional probability distribution features for Arabic words, IEEE/ACS International Conference on Computer Systems and Applications, pp.582- 590, 2008
- [16] Rahul Kala, Harsh Vazirani , Anupam Shukla and Ritu Tiwari —Offline Handwriting Recognition using Genetic Algorithm IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 2, No 1, March 2010.
- [17] Andreas Schlapbach and Horst Bunke —Off-line Writer Identification Using Gaussian Mixture Models The 18th International Conference on Pattern Recognition (ICPR'06) 0-7695-2521-0/06 \$20.00 © 2006.
- [18] Andreas Schlapbach and Horst Bunke —Off-line Writer Identification Using Gaussian Mixture Models The 18th International Conference on Pattern Recognition (ICPR'06) 0-7695-2521-0/06 \$20.00 © 2006.
- [19] Paraskevas, D., Stefanos, G., & Ergina, K. —Writer Identification Using a Statistical and Model Based Approach. 14th International Conference on Frontiers in Handwriting Recognition, 2167-6445/14 \$31.00 © 2014 IEEE
- [20] Al-Maadeed, S., Hassaine, A., & Bouridan, A. (2014). —Using codebooks generated from text skeletonization for forensic writer identification. 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), 978-1-4799- 7100- 8/14/\$31.00 ©2014 IEEE.
- [21] Djeddi, C, Siddiqi, I, Souici-Meslati, L, & Ennaji, A. (2012). —Multi-script Writer Identification Optimized with Retrieval Mechanism. 2012 International Conference on Frontiers in Handwriting Recognition. 978-0-7695-4774-9/12 \$26.00 © 2012 IEEE.