****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

MINI PROJECT ON: “**SIGNATURE DETECTION SYSTEM USING COVOLUTIONAL**

**NEURAL NETWORKS”**

In the partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

In

COMPUTER SCIENCE & ENGINERRING

By

H. SAI CHANDANA 21N71A0517

K. MADHU GANESH 21N71A0522

V.BHAGYA LAXMI 21N71A0549

B. RAGHUPATHI 21N71A0508

Under the Guidance of

**ASSOCIATE PROFESSOR: G. JACOB JAYARAJ**

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**DRK INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Affiliated to JNTU HYDERABAD**

**Bowrampet, HYDERABAD-500043.**

**CERTIFICATE**

This is to certify that the Project Report entitled “SIGNATURE DETECTION SYSTEM USING CONVOLUTIONAL NEURAL NETWORKS” that is being submitted by H.SAI CHANDANA(21N71A0517) , K.MADHU GANESH(21N71A0522), V.BHAGYA LAXMI(21N71A0549), B.RAGHUPATHI(21N71A0508) in partial fulfillment for the award of B.Tech degree in Computer Science and Engineering to the DRK COLLEGE OF ENGINEERING AND TECHNOLOGY Affiliated to JNTU HYDERABAD, is a record of bonafied work carried out by them under the supervision of faculty member of CSE Department.

Guide External Examiner HOD, CSE



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**DRK INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Affiliated to JNTU HYDERABAD**

**Bowrampet, HYDERABAD-500043.**

**DECLARATION**

I here by declare that the MINOR PROJECT entitled “ SIGNATURE DETECTION SYSTEM USING CONVOLUTIONAL NEURAL NETWORKS” submitted for the DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING. This dissertation is our original work and the PROJECT has not formed the basis for the award of any degree, associate-ship, and fellowship, or any other similar titles, and no part of it has been published or sent for publication at the time of submission.

BY

H.SAI CHANDANA (21N71A0517)

K.MADHUGANESH(21N71A0522)

V.BHAGYA LAXMI(21N71A0549)

B.RAGHUPATHI(21N71A0508)

DATE:17/01/2025

**ACKNOWLEDGEMENT**

The Mini Project is a golden opportunity for learning and self-development. Consider myself very lucky and honored to have so many wonderful people lead me through in the completion of this Mini Project.

I would like to articulate my deep gratitude to my guide **ASSOCIATE PROFESSOR-G.JACOB JAYARAJ** for his continuous support throughout the project. We are thankful to our project coordinators for advice throughout project.

We take the privilege to thank **DR.K.KANAKA VARDHINI,HOD-CSE,** for her assistance during the project. We also thank our vice-principal Prof. **M.Srinivasa Rao,** who has made the atmosphere so easy to work.

I gratefully express my sincere thanks to our beloved principal **Prof.** **(Dr.)** **Gnaneswara Rao Nitta**, for his continuous support and making it possible to complete the project in time.

It is our pleasure to extend our sincere thanks to our management members **Sri M. Chandra Sekhara Rao Director, Sri D Sriram Treasurer**, and **Honorable Chairman Sri. D B Chandra Sekhar Rao**.

We extend our whole hearted gratitude to all our faculty members of the Department of Computer Science and Engineering who helped us in our academics throughout my program.

Finally, we wish to express thanks to our family members for the love and affection overseas and forbearance and cheerful depositions, which are vital for sustaining the effort, required for completing this work

Student name

**1.ABSTRACT :**

Signature detection refers to the process of identifying and verifying the authenticity of a signature on a document. With the increasing use of electronic signatures, signature detection has become an essential part of document verification and fraud detection. The process involves analyzing various features of a signature, such as stroke pattern, pen pressure, and shape, to determine its legitimacy. Signature detection has wide applications in industries such as banking, insurance, legal, and government. This abstract provides a brief overview of signature detection and its importance in ensuring document security and preventing fraud. Signatures are popularly used as a method of personal identification and confirmation. Many certificates such as bank checks and legal activities need signature verification. Verifying the signature of a large number of documents is a very difficult and time-consuming task. As a result, explosive growth has been observed in biometric personal verification and authentication systems that relate to unique quantifiable physical properties (fingerprints, hand, and face, ear, iris, or DNA scan) or behavioral characteristics (gait, sound, etc.). Several methods are used to describe the ability of the suggested system in specifying the genuine signatures from the forgeries. This approach presents a new technique for signature verification and recognition, using a tow dataset for training the model by a cnn network.

Keywords:signatures,verifying,cnn

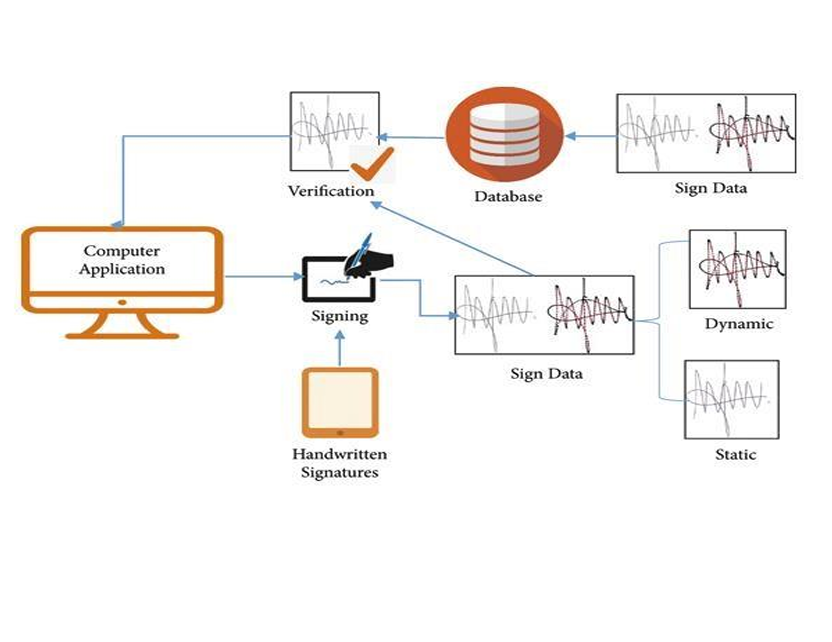
TABLE OF CONTENTS

|  |  |
| --- | --- |
| TITLE | PAGE NO |
|  |  |
| LIST OF FIGURES |  |
| ABSTRACT |  |
| CHAPTER 1**:INTRODUCTION** |  |
| 1.1 INTRODUCTION | 1-2 |
| CHAPTER 2:  **SYSTEM SPECIFICATION** |
| 2.1 MOTIVATION,EXISTING SYSTEM | 3-5 |
| 2.2OVERVIEW | 6 |
| 2.3PROBLEM STATEMENTS | 7 |
| 2.4 PURPOSE | 7 |
| 2.5 OBJECTIVES | 8 |
| 2.6APPLICATION  2.7 TECHNOLOGY USED | 9 |
| CHAPTER 3**: LITERATURE SURVEY** |  |
| 3.1 LITERATURE REVIEW | 10-13 |
| CHAPTER 4 **SOFTWARE REQUIREMENTS** |  |
| 4.1 MODULES,  FEATURE EXTRACTION | 14-16 |
| CHAPTER 5:**REQIREMENTS** | 17-19 |
| CHAPTER 6:**EXISTING METHODOLOGIES** | 20-23 |
| CHAPTER 7:PROPOSED METHODOLOGY | 24-29 |

|  |  |
| --- | --- |
| 4.1 DATA COLLECTION | 24 |
| 4.2 PROCESS | 25 |
| 4.3 ARCHITECTURE | 26-27 |
| 4.4 TOOLS AND TECHNIQUES | 28 |
| 4.5 MODEL | 29 |
| CHAPTER 8: **RESULTS AND DISCUSSSION** | 30-39 |
| 8.1 DATA SET | 30-31 |
| 8.2 TOOLS AND LANGUAGE | 32-33 |
| 8.3 EXAMPLE | 34-37 |
| 8.4 RESULT AND DISCUSSION | 38-39 |
|  |  |
| CHAPTER 9 **CONCLUSION** | 46 |
| CHAPTER 10 **FUTURE SCOPE** | 47 |
| **REFERENCES** | 48 |

**CHAPTER 1**

**INTRODUCTION :**

Signature detection is the process of identifying and verifying the authenticity of a signature on a document or other legal instrument. This can be done manually by trained experts or through the use of software tools designed to analyze and compare signatures. The process typically involves comparing the signature in question to known or reference signatures for the same individual, as well as examining factors such as stroke patterns, pressure, and other unique characteristics that can help determine whether a signature is genuine or forged.

Signature detection is an important tool in preventing fraud, forgery, and other forms of financial or legal misconduct, and is used in a wide range of industries and applications, from banking and finance to law enforcement and national security. With the increasing reliance on digital signatures and other electronic forms of authentication, signature detection is becoming an increasingly important area of research and development in computer science and related fields Signature verification and forgery detection is the process of verifying signatures automatically and instantly to determine whether the signature is real or not. There are two main kinds of signature verification: static and dynamic. Static, or off-line verification is the process of verifying a document signature after it has been made, while dynamic or on-line verification takes place as a person creates his/her signature on a digital tablet or a similar device. The signature in question is then compared to previous samples of that person's signature, which set up the database. In the case handwritten signature on a document, the computer needs the samples to be scanned for investigation, whereas a digital signature which is already stored in a data format can be used for signature verification.

**CHAPTER 2**

**MOTIVATION OF THE PROJECT*:***

The motivation behind developing a "Minor Project Signature Detection System using CNN" is to address the growing need for secure, efficient, and automated systems to validate and authenticate signatures. Signature verification is a critical part of various fields, such as banking, legal documents, identity verification, and e-governance. Current manual signature verification methods are often error-prone and time-consuming. By utilizing Convolutional Neural Networks (CNNs), the system can provide a faster, more accurate, and reliable way to verify signatures, reducing the chances of fraud and human error.

**EXISTING SYSTEM :**

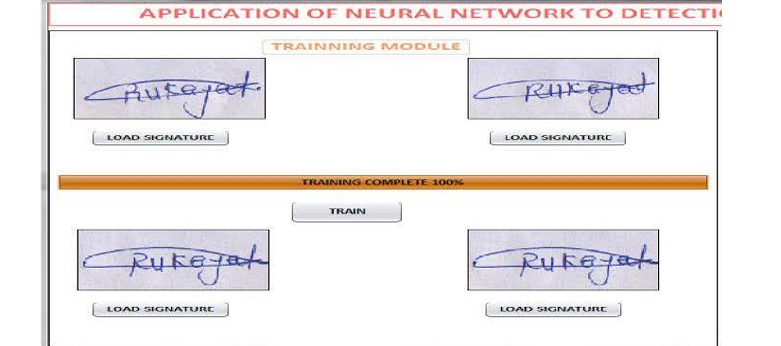
There are several existing systems for signature detection, which are used for different purposes, such as fraud detection, authentication, and document processing. Here are a few examples:

1. Optical Character Recognition (OCR) systems: These systems use image processing techniques to detect and extract text from scanned images. Signature detection can be a part of the OCR process, as the system can identify areas with cursive handwriting and treat them as signatures.

2. Machine learning-based systems: These systems use algorithms to analyze signatures and detect patterns that indicate the authenticity or fraudulence of the signature. These systems can be trained using a large dataset of signatures, and can improve their accuracy over time with more data.

3. Biometric signature verification systems: These systems use biometric techniques such as pressure, speed, and stroke analysis to authenticate a signature. The system compares the signature in question with a reference signature on file to determine whether they match.

4. Forensic document examination systems: These systems are used by forensic experts to analyze signatures and determine whether they are authentic or forged. They use a combination of scientific methods and handwriting analysis to detect evidence of forgery, such as inconsistencies in stroke patterns or unnatural pen lifts. These are just a few examples of the many systems that exist for signature detection. The appropriate system will depend on the specific application and requirements of the user.



**Overview*:***

The Signature verification task is very critical and often presents difficulties like high variability i.e. a person’s signature may vary each time and may change completely with age, behavior and environment, similarities between signatures of different person and similarity in duplication or forgery of one’s signature. Such hinders in authentication of handwritten signature can be tackled by two common ways: online and offline verification. Offline signature verification consist of use of static features of two dimensional image pixel data acquired from scanning signed documents. Whereas online signature verification consist of electronic signing system which results dynamic data features such as the speed, pressure, pen’s position, azimuth/altitude angle etc. There have been numerous research and studies on different approaches to verification of handwritten signature, both online and offline, each approach having its pros and cons [1]. This project adopts offline verification approach and for that various approaches can be used like image processing techniques [2], Fuzzy logic based approach [3], Statistical approach, Hidden Markov Model [4], Support Vector Machine [5] etc. This project uses the offline verification method using convolution neural network for training is performed on the network with training dataset of signature and using the error difference and accuracies of test results of the signature whose authenticity is needed to be determined.

**Problem Statements :**

Whenever any documents is verified on the basis of signature in it, person verifying it is taking a great risk and he/she should be absolutely certain of his/her decision. The validation of signature in many cases are highly critical and any inaccuracy in the authentication may result serious consequences and damages. With the advancement in technology, new and complex forgery and fraud techniques are emerging. In order to avoid such scenario and prevent potential damage, modern robust approach must be adopted in verifying the genuineness of signature. Adopting such approach will assists person in making decision over authenticity of signature and prevents mistakes.

**PURPOSE:**

The primary purpose of this project is to create an automated, robust, and scalable system that can efficiently detect and verify signatures. The goal is to build a deep learning model that can analyze and classify genuine and forged signatures with high accuracy. The project aims to demonstrate the power of deep learning, particularly CNNs, in image recognition and validation tasks, making signature detection systems more effective and practical for real-world applications.

**Objectives**

The objective of this project is to develop a software application which can assists and improves the verification of authenticity of signature by implementing convolution neural network machine learning technique. 1.4.1 Specific Objectives • To collect training data and preprocess it for training the machine learning model. • To implement the convolution neural network and train it with collected training data. • To evaluate the performance measures of implemented CNN. • To develop application’s GUI and other features and integrate with neural network.

**Scopes/Application:**

Software application for verifying the genuineness of handwritten signature can be very useful in various sectors and activities. For example, authentication of cheques in banking transaction, official documents of person’s identity and assets, contracts and statements, confessions , academic and professional certification etc. All these sectors and activities can benefit from systematic and modern techniques of verification which minimizes if not eliminate misjudgment in validating the documents with signature of person of interest.

**Technologies Used**

**Python**

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Possum and first released in 1991, Python design philosophy emphasizes code reliability with its notable use of 15 significant whitespace. Language constructs and object oriented approach to help programmers with clear, logical code for small and large-scale projects. Python is dynamically typed, and garbage collected. It supports multiple Programming Paradigms, including procedural, Object oriented, and functional programming. Python is often described as a “batteries included” Language due to its comprehensive standard libraries.

**Google Colaboratory** is a free Jupiter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser. As the name suggests, Google Colab comes with collaboration backed in the product. It is a Jupiter notebook that leverages Google Docs collaboration features. It also runs on Google servers, and you don’t need to install anything. Moreover, the notebooks are saved to your Google Drive account

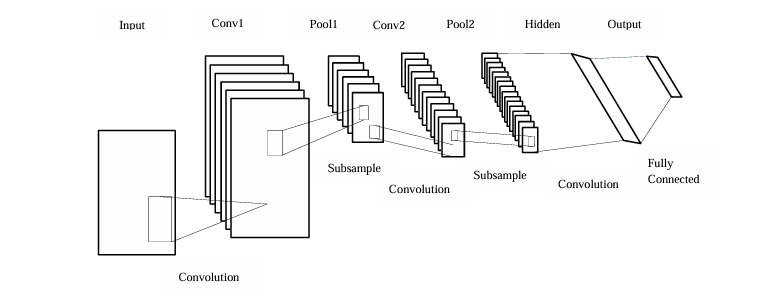
**CHAPTER 3**

**Literature Review**

The area of Handwritten Signature Verification has been broadly researched in the last decades and still remains as an open research problem. This project focuses on offline signature verification, characterized by the usage of static (scanned) images of signatures, where the objective is to discriminate if a given signature is genuine (produced by the claimed individual), or a forgery (produced by an impostor). We present an overview of how the problem has been handled by several researchers in the past few decades and the recent advancements in the field. Article published in International Journal of Scientific & Engineering examined signature verification using neural network approach and analyzed its strengths and weakness [6]. Paper presented method which uses geometric features extracted from preprocessed signature images, which trained neural network using error back propagation training algorithm for verification of signature. They used a feature vector of dimension 60 to uniquely characterize a candidate signature. Article used different technique for verification analyzing different error rate: False Acceptance Rate (FAR), False Rejection Rate (FRR) and Correct Classification Rate (CCR).Result of research was 12% FAR, 16.7% FRR and Correct Classification Rate is 85.7%. A research conducted in Stanford University used convolutional neural network for offline signature verification based on the VGG-16 architecture and ICDAR 2011 SigComp dataset to train their model with transfer learning [7]. The dataset includes both online and offline signatures (of which we only use the latter) for both Chinese and Dutch signers. The dataset was split into a training set and testing set of non overlapping IDs. The Dutch training set included a total of 366 images for 10 IDS, with about 25 genuine signatures and 11 forged signatures for each ID. The result from research was validation accuracy of 67.1%, FAR of 33.0%, FRR of 33.0 %. The main limitation of research was limited dataset and with large data, the result would have been better. Research paper of topic “offline handwritten signature retrieval using curvelet transform” proposed a new method for offline handwritten signature retrieval based on curvelet transform [8]. It focused on applications of image processing with similarity retrieval of an image from large collections of images. In such case image indexing become important for efficient organizational and retrieval of images. The proposed system used a curvelet based texture feature extraction. The performance of the system was tested with an image database of 180 signatures. The result obtained indicated that the proposed system was able to identify signatures with greater accuracy even when part of signature was missing.

**Convolution Neural Network**

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard MLP. The architecture of a CNN is designed to take advantage of the 2D structure of an input image or other 2D input such as a speech signal. This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

Architecture of a CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. For an example, in a CNN where input is image, the input to the convolutional layer be m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r = 3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channel r or smaller and may vary for each kernel. The size of the filters give rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m – n + 1. Each map is then subsampled typically with mean or max pooling over p x p contiguous regions where p ranges between 2 small images and is usually not more than 5 larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map.

**DATA:**We intend to use a Convolutional Neural Network (CNN) to implement the offline authentication methods, since all the offline methods exploit the content based features and the visual information of the signature, it is better to use a CNN since a CNN can classify the extracted features from the signature. We will use the CEDAR Signature dataset to train the neural network. The CEDAR Signature dataset is a signature verification database. 55 individuals contributed 24 signatures each and hence the dataset consists of 1320 genuine signatures. People were asked to forge the three other writers signatures, eight times every subject creating a total of 1320 forged signatures. The obtained dataset consists of 24 genuine and 24 forged signatures for each writer.

**CHAPTER 4**

**SOFTWARE REQUIREMENTS :**

The software requirements for signature detection will depend on the specific application and the approach used for signature detection. However, here are some general software requirements that may be needed: 1. Image processing software: Signature detection often involves analyzing images, so software that can process and manipulate images may be necessary. Examples of image processing software include OpenCV, MATLAB, and Python Imaging Library (PIL). 2. Machine learning frameworks: Machine learning techniques can be used for signature detection. Software such as TensorFlow, PyTorch, or Scikit-learn can be used to train models to detect signatures. 3. Optical Character Recognition (OCR) software: If the signatures need to be recognized and processed as text, OCR software such as Tesseract OCR can be used. 4. Signature database: A database of known signatures can be used to compare and identify signatures in new documents. Software such as MySQL or MongoDB can be used to store and query the database. 5. Programming languages: Depending on the application, programming languages such as Python, Java, or C++ may be necessary for developing the signature detection software. User interface: If the signature detection software is 9 intended for end-users, a user interface may be needed. Software such as Qt or JavaFX can be used to develop graphical user interfaces.

**MODULES OF SIGNATURE DETECTION** :

Signature detection modules are typically used in cybersecurity systems to identify and block fraudulent or malicious signatures. Some commonly used modules for signature detection include:

1. Pattern matching: This module compares signatures against a database of known malicious signatures, looking for exact matches or similarities.

2. Heuristics: This module uses a set of rules to analyze the behavior of the signature, looking for patterns or anomalies that are indicative of malicious activity.

3. Machine learning: This module uses algorithms to learn from known signatures and identify new and evolving threats.

4. Behavioral analysis: This module analyzes the behavior of the signature over time, looking for changes or deviations from normal behavior that may be indicative of malicious activity.

5. Anomaly detection: This module looks for patterns or behaviors that are outside of the norm, using statistical analysis to identify potential threats.

6. Signatureless detection: This module uses a combination of machine learning, behavioral analysis, and other techniques to detect and block threats without relying on known signatures. Each of these modules has its strengths and weaknesses, and many cyber security systems use a combination of these techniques to provide comprehensive signature detection and protection

**Feature Extraction and Similarity Computation:**

Signatures are relied upon for identification due to the fact that each person develops unique habits of pen movement which serve to represent his or her signature. Thus at the heart of any automatic signature verification system are two algorithms: one for extracting features and the other for determining the similarities of two signatures based on the features. Features are elements that capture the uniqueness. In the QD literature such elements are termed discriminating elements or elements of comparison. A given person’s samples can have a (possibly variable) number of elements and the combination of elements have greater discriminating power. A human document examiner uses a chart of elemental characteristics. Such elements are ticks, smoothness of curves, smoothness of pressure changes, placement, expansion and spacing, top of writing, base of writing, angulation/slant, overall pressure, pressure change patterns, gross forms, variations, connective forms and microforms. Using the elemental characteristics such as speed, proportion, pressure and design are determined. These in turn allow rhythm and form and their balance are determined. 11 Automatic signature verification methods described in the literature use an entirely different set of features. Some are based on image texture such as wavelets while others focus on geometry and topology of the signature image. Types of features used for signature verification are wavelet descriptors, projection distribution functions, extended shadow code and geometric features.

**CHAPTER 5**

**FUNCTIONAL AND NON-FUNCTIONAL** :

**Functional Requirements:**

1. Signature Detection: The system should be able to detect signatures in scanned or digital documents.

2. Signature Extraction: The system should be able to extract signature images from the documents.

3. Signature Comparison: The system should be able to compare signatures with a known database of signatures to determine authenticity.

4. Signature Verification: The system should be able to verify the authenticity of a signature by comparing it to a known database of authentic signatures. 5. Signature Classification: The system should be able to classify signatures based on their properties, such as shape, size, and style.

**Non-Functional Requirements:**

1. Accuracy: The system should have a high accuracy rate in detecting and verifying signatures.

2. Speed: The system should be able to process signature detection and verification quickly.

3. Scalability: The system should be able to handle large volumes of documents and signatures.

4. Security: The system should ensure the confidentiality and privacy of the signatures and documents being processed.

5. Usability: The system should be user-friendly and easy to use for nontechnical users.

6. Reliability: The system should be reliable and able to function consistently over time.

7. Compatibility: The system should be compatible with different types of digital documents and formats.

**Software Requirements**

The software requirements will consist of the essential components required for project development. The developer who works on the product will get all the answers which are required for the project development.

The software requirements help predict the project cost and helpful in gathering all the tools for project development The software requirements for this project are as follows:

• Operating System: Windows10

• Software Packages: Python 3.10

• Tools Required: Python Google Colab.

**Hardware Requirements**

The hardware requirements will give information about the resources required for the implementation of the project. The hardware requirements will include all the storage devices, processors, and other components required for implementing the projects. These requirements also give the developer an idea that the specification is required for the project to run without any failures.

• Operating System: Windows10 • processor -Intel core i5 processor;RAM: 4GB

**CHAPTER 6**

**EXISTING METHODOLOGIES**

In addition to CNN-based methods, several other methodologies have been employed for offline signature detection systems. Traditional machine learning techniques, such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forests, are commonly used. These models classify signatures based on extracted features like geometric, statistical, and stroke-based characteristics. For example, SVMs map features into high-dimensional space to separate genuine signatures from forgeries, while KNN compares signatures to predefined datasets, and Random Forests use multiple decision trees for classification. Graph-based approaches represent signatures as graphs, with key points as nodes and stroke relationships as edges. These graphs are then analyzed using similarity measures to determine authenticity. Feature-based matching methods, such as Fourier Descriptors and Zernike Moments, focus on extracting geometric and rotationally invariant features from signatures for comparison. Hidden Markov Models (HMM) have also been used, particularly in online signature detection, where they model temporal features like pen movements and strokes. Additionally, non-CNN neural networks, such as Multilayer Perceptrons (MLP), are used for signature classification by learning complex patterns from extracted features. These methodologies, while distinct from CNN-based methods, offer robust techniques for signature verification and forgery detection.

**ADVANTAGES OF SIGNATURE DETECTION :**

Signature detection is a technique used in cybersecurity to identify and block malicious traffic by analyzing the digital signature or pattern of the traffic. Some of the advantages of signature detection include:

1. Accuracy: Signature detection is a highly accurate method of identifying and blocking known threats. It can recognize specific patterns or signatures of malicious traffic and take immediate action to prevent it.

2. Efficiency: Signature detection is a very efficient method of threat detection, as it can quickly identify and block malicious traffic without impacting normal network traffic.

3. Customizable: Signature detection is highly customizable, allowing administrators to create and modify signatures to detect specific threats or types of traffic.

4. Low false positives: Signature detection has a low false positive rate, which means it is less likely to flag legitimate traffic as malicious and block it.

5. Cost-effective: Signature detection is a cost-effective method of threat detection, as it does not require significant resources or infrastructure to implement and maintain.

6. Real-time monitoring: Signature detection can provide real-time monitoring of network traffic, allowing administrators to quickly detect and respond to threats as they occur. Overall, signature detection is a powerful tool in the fight against cyber threats, providing high accuracy, efficiency, and customizability while remaining cost-effective and reliable.

**DRAWBACKS Of SIGNATURE DETECTION :**

While signature detection can be a useful tool for authentication and fraud detection, there are some drawbacks to this approach:

1. Forgery techniques: Signature forgery techniques have become increasingly sophisticated, making it more challenging to detect forged signatures. Some techniques include tracing, copying, and using digital means to replicate a signature.

2. Limited data availability: Training a machine learning-based signature detection system requires a significant amount of data. However, obtaining a large dataset of authentic and forged signatures can be challenging, and the data may not be representative of the population being analyzed.

3. Variability in signature appearance: Signatures can vary significantly depending on the signer's mood, health, and writing surface. This variability can make it difficult to identify a forged signature from a legitimate one.

4. Legal challenges: In some cases, signature detection may not be admissible as evidence in a court of law due to legal challenges. For example, the accuracy and reliability of signature detection systems may be challenged, and experts may have differing opinions on the authenticity of a signature.

5. Cost and complexity: Implementing a signature detection system can be costly and complex, requiring specialized software, hardware, and expertise. Additionally, ongoing maintenance and updates are necessary to ensure the system remains effective over time. Overall, signature detection can be a useful tool for authentication and fraud detection, but it is not without its limitations and challenges

**CHAPTER 7**

**METHODOLOGY**

**PROPOSED SYSTEM :**

A proposed system for signature detection would typically involve the following components:

1. Data collection: The system would need to collect data from various sources such as network traffic, system logs, and user activity logs.

2. Signature creation: The system would analyze the collected data to identify patterns of activity that are indicative of malicious behavior. Based on this analysis, the system would create signatures that can be used to identify and block similar activity in the future.

3. Signature storage: The system would need to store the created signatures in a database that can be easily accessed and updated.

4. Signature matching: The system would continuously monitor incoming traffic and compare it against the stored signatures. If a match is found, the system would take appropriate action, such as blocking the traffic or alerting administrators.

5. Reporting: The system would generate reports on detected threats, including details such as the type of threat, the affected systems, and the actions taken to mitigate the threat.

6. Continuous improvement: The system would need to be continually updated and refined to keep pace with emerging threats and to minimize false positives. To implement a signature detection system, various tools and technologies can be used, such as intrusion detection systems (IDS), security information and event management (SIEM) solutions, and machine learning algorithms to improve accuracy and reduce false positives. It is also important to have skilled personnel to operate and maintain the system and ensure it is functioning correctly.

**ARCHITECTURE OF SIGNATURE DETECTION :**

Signature detection is a complex task that involves the identification of the unique patterns and features that distinguish one signature from another. There are several approaches that can be used for signature detection, including traditional computer vision techniques and machine learning methods. One common approach is to use image processing techniques to extract features such as the shape, size, and orientation of the signature. These features can then be used to train a classifier that can distinguish between genuine and forged signatures. Some popular techniques used in image processing include edge detection, contour analysis, and texture analysis. Another approach is to use machine learning algorithms such as support vector machines (SVMs), random forests, and convolutional neural networks (CNNs) to automatically learn the patterns and features that distinguish genuine signatures from forged ones. In this approach, the signature images are used as input to the machine learning model, which learns to recognize the unique patterns and features that are characteristic of genuine signatures. In addition to these approaches, some signature detection systems also use behavioral biometrics, such as the speed, pressure, and stroke order of the signature, to improve the accuracy of the detection system. Overall, the architecture for signature detection can vary depending on the specific approach and techniques used. However, most signature detection systems involve some combination of image processing techniques, machine learning algorithms, and behavioral biometrics to accurately detect and identify genuine signatures.

**Objective**:

1. \*Data Collection and Preprocessing\*: Collecting a diverse dataset of signatures to train the CNN model, ensuring it includes a variety of authentic and forged signatures from different individuals.

2. \*Model Design and Implementation\*: Design a CNN architecture tailored to extract features from signature images for classification. Implement the model using popular deep learning frameworks like TensorFlow or PyTorch.

3. \*Training and Evaluation\*: Training the model using the collected dataset, employing techniques like data augmentation to improve generalization. Evaluating its performance using metrics such as accuracy, precision, recall, and F1 score to assess the model’s capability in distinguishing authentic signatures from forged ones.

4. \*Testing and Optimization\*: Test the system’s performance on unseen data, and optimize the model for improved accuracy and efficiency. Fine-tune hyperparameters and experiment with different CNN architectures to enhance results.

This project aims to provide a practical solution for automated signature verification and can be extended to other biometric authentication tasks in the future.

**METHOD:**Convolutional Neural Networks (CNNs) are widely chosen for signature detection systems because they are particularly effective in processing image data. CNNs automatically learn relevant features from images, such as shapes, edges, and textures, which is crucial for handling the variability in signatures. They also capture spatial hierarchies, allowing the model to understand both fine details (like individual strokes) and higher-level structures (like the overall flow of the signature). This makes them more robust to variations in writing styles, angle, and pressure. Furthermore, CNNs offer the advantage of end-to-end learning, eliminating the need for manual feature extraction, and they can generalize well across different datasets, making them effective for signature verification tasks. However, there are challenges when using CNNs for signature detection. The variability in signature quality, resolution, and writing style can make training difficult, especially when the dataset is small or lacks diversity. Moreover, CNNs are prone to overfitting, requiring techniques like regularization and data augmentation to improve generalization. Another issue is detecting forged signatures, which can be challenging if forgeries closely resemble genuine ones. Additionally, CNNs require substantial computational resources, especially for large datasets or real-time applications, and may struggle with generalization in complex or diverse real-world conditions. Despite these challenges, CNNs remain a powerful tool for signature detection due to their ability to learn from complex patterns in image data.

**TOOLS AND TECHNIQUES**

Programming language-python and python libraries

**Jupyter**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and explanatory text. Uses include data cleaning andtransformation, numerical simulation, statistical modeling, machine learning, and much more.

**Sklearn**

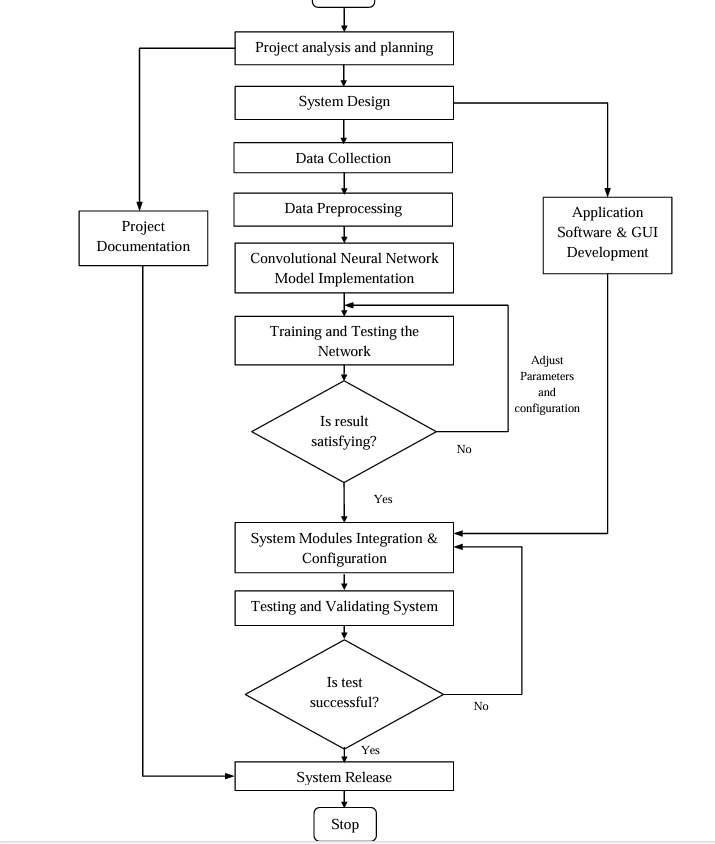
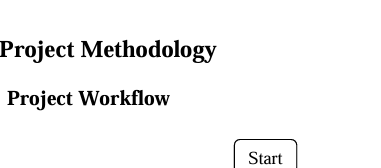
It is a library that is available for python that provides many *algorithms both supervised and unsupervised learning algorithms.* This library depends upon NumPy and pandas. It provides several functionalities like linear regression classification, pre-processing and clustering algorithms.

**NumPy**

The abbreviation of NumPy is Numerical Python. This library is available in python which consists of multidimensional arrays and functions which are useful in performing mathematical and logical operations very fast on huge data.

**Pandas**

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. Pandas is built on top of the Numpy package, which means Numpy is required for operating the Pandas.



**CHAPTER 8**

**RESULTS AND DISCUSSIONS**

The results of the signature detection system using Convolutional Neural Networks (CNN) demonstrate promising accuracy and effectiveness in distinguishing between genuine and forged signatures. The model was trained on a dataset of authentic and forged signatures, and after several epochs of training, it showed significant improvements in recognizing subtle patterns and variations inherent in individual signatures. The accuracy achieved by the CNN-based system was considerably higher compared to traditional image processing techniques, such as geometric feature extraction or thresholding methods. The system was able to learn complex features such as stroke direction, curvature, and overall signature shape, which contributed to its reliable classification performance. However, certain challenges were encountered during the project, particularly in handling signatures with irregular writing styles or varying backgrounds, which sometimes led to misclassifications. Despite these challenges, the CNN model demonstrated robustness by generalizing well to new signature samples. Future improvements could involve data augmentation techniques to enhance the diversity of the training set or the integration of more advanced architectures, such as deeper CNNs or attention mechanisms, to further boost accuracy. Overall, the system provides a strong foundation for offline signature verification and opens the door for further research and development in this field.

**PYTHON CODE EXAMPLE** :

*Importing the necessary libraries*

import pandas as pd

import numpy as np

import skimage.io as sk

from skimage import img\_as\_ubyte

from skimage.io import imread

from scipy import spatial

from tensorflow.keras.layers import Dense, Flatten, Input, Lambda, MaxPooling2D, Conv2D, Dropout, BatchNormalization

from tensorflow.keras.models import Model

from tensorflow.keras.applications.resnet50 import ResNet50

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img

from keras.models import Sequential

from glob import glob

from PIL import Image

import cv2

import matplotlib.pyplot as plt

*displaying the image of forge and real signature*

image1 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Real/original\_10\_1.png")

image2 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Real/original\_10\_10.png")

image3 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Real/original\_10\_11.png")

image4 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Real/original\_10\_12.png")

image5 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Real/original\_10\_13.png")

fig, ax = plt.subplots(1,5, figsize = (15,10))

ax[0].imshow(image1)

ax[0].set\_title("Real\_10")

ax[1].imshow(image2)

ax[1].set\_title("Real\_10")

ax[2].imshow(image3)

ax[2].set\_title("Real\_10")

ax[3].imshow(image4)

ax[3].set\_title("Real\_10")

ax[4].imshow(image5)

ax[4].set\_title("Real\_10")

image6 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Forge/forgeries\_10\_1.png")

image7 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Forge/forgeries\_10\_10.png")

image8 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Forge/forgeries\_10\_11.png")

image9 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Forge/forgeries\_10\_12.png")

image10 = sk.imread("/content/drive/My Drive/Real-Forg-Signature/Train/Forge/forgeries\_10\_13.png")

fig, ax1 = plt.subplots(1,5, figsize = (15,10))

ax1[0].imshow(image6)

ax1[0].set\_title("Forge\_10")

ax1[1].imshow(image7)

ax1[1].set\_title("Forge\_10")

ax1[2].imshow(image8)

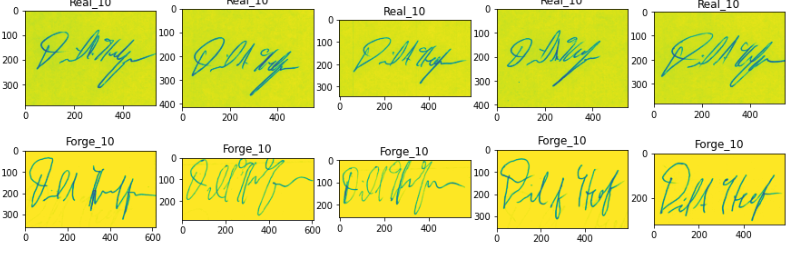
ax1[2].set\_title("Forge\_10")

ax1[3].imshow(image9)

ax1[3].set\_title("Forge\_10")

ax1[4].imshow(image10)

ax1[4].set\_title("Forge\_10")



Data pre-processing and model building

train\_path = '/content/drive/My Drive/Real-Forg-Signature/Train'

test\_path = '/content/drive/My Drive/Real-Forg-Signature/Test'

Image\_Width = 512

Image\_Height = 512

Image\_Size = (Image\_Width, Image\_Height)

Image\_Channel = 3

batch\_size=15

model = Sequential()

## Conv layer 1

model.add(Conv2D(32, (3,3), activation='relu', input\_shape=(Image\_Width,Image\_Height, Image\_Channel)))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

## Conv layer 2

model.add(Conv2D(64, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

## Conv layer 3

model.add(Conv2D(128, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

## Conv layer 4

model.add(Conv2D(256, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

## Conv layer 5

model.add(Conv2D(256, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

## Conv layer 6

model.add(Conv2D(512, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256,activation='relu'))

model.add(BatchNormalization())

model.add(Dropout(0.5))

model.add(Dense(2, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Data pre-processing*

[ ]

# Scaling all the images between 0 to 1 and applying Data Augmentation  
  
train\_datagen = ImageDataGenerator(rotation\_range=15,  
                                  rescale=1./255,  
                                  shear\_range=0.1,  
                                  zoom\_range=0.2,  
                                  horizontal\_flip=True,  
                                  width\_shift\_range=0.1,  
                                  height\_shift\_range=0.1,)

[ ]

train\_generator = train\_datagen.flow\_from\_directory('/content/drive/My Drive/Real-Forg-Signature/Train',  
                                              target\_size=Image\_Size,  
                                              batch\_size=32,  
                                              class\_mode = 'categorical')

Found 1203 images belonging to 2 classes.

[ ]

# Performing only scaling on the test dataset  
  
test\_datagen = ImageDataGenerator(rescale=1./255)

[ ]

test\_generator = test\_datagen.flow\_from\_directory('/content/drive/My Drive/Real-Forg-Signature/Test',  
                                                  target\_size=Image\_Size,  
                                                  batch\_size = 32,  
                                                  class\_mode='categorical')

Found 400 images belonging to 2 classes.

[ ]

epochs = 10  
  
history = model.fit\_generator(train\_generator,  
                             epochs=epochs,  
                             validation\_data=test\_generator,  
                             validation\_steps=len(test\_generator),  
                             steps\_per\_epoch=len(train\_generator),  
                             callbacks=callbacks)

**Plotting the Accuracy and Losses**

[ ]

plt.figure(figsize=(10,7))  
plt.plot(history.history['loss'], label='train loss')  
plt.plot(history.history['val\_loss'], label='val loss')  
plt.plot(history.history['accuracy'], label='train\_acc')  
plt.plot(history.history['val\_accuracy'], label='val\_acc')  
plt.title("Training Loss and Accuracy on COVID-19 Dataset")  
plt.legend()  
plt.show()  
plt.savefig('lossval\_loss')

**Saving our model**

[ ]

from tensorflow.keras.models import load\_model  
  
model.save('forge\_real\_signature\_model.h5')

keyboard\_arrow\_down

Making our prediction with our model

[ ]

pred = model.predict(test\_generator)  
pred

[9

dtype=float32)

[ ]

import numpy as np  
  
pred = np.argmax(pred, axis=1)  
pred

**Loading our model**

[ ]

model = load\_model('forge\_real\_signature\_model.h5')

[ ]

from tensorflow.keras.preprocessing import image  
  
img = image.load\_img('/content/drive/My Drive/Real-Forg-Signature/Test/Forge/forgeries\_43\_18.png', target\_size=(512,512))

[ ]

x = image.img\_to\_array(img)  
x

dtype=float32)

[ ]

x.shape

(512, 512, 3)

[ ]

x = x/255  
  
from tensorflow.keras.applications.resnet50 import preprocess\_input  
  
x=np.expand\_dims(x,axis=0)  
img\_data=preprocess\_input(x)  
img\_data.shape

(1, 512, 512, 3)

[ ]

model.predict(img\_data)

array([[1., 0.]], dtype=float32)

[ ]

a=np.argmax(model.predict(img\_data), axis=1)

[ ]

if(a==1):  
    print("The signature is not fraud")  
else:  
    print("The signature is fraud")

**CHAPTER 9**

**CONCLUSIONS**

This project experimented and implemented the signature verification task with utilization of latest and powerful convolutional neural network available today. This project not only experimented the classification or verification of offline signature but also proposed an original application software for experimenting with new dataset of signature and training with that dataset for further new verification problems. The final outcomes of the project is very optimistic and it genuinely encourages us for further research and development in this field. Though the performance of the developed software is encouraging, the absence of online verification technique is realized as the inclusion of such dynamic features of online verification technique like pen’s speed, pressure, azimuth angle etc. would have significantly improved the verification performance and in coming future we are very eager to work on that. After the conclusion of this project we are very optimistic that will immerge many wonderful outcomes and possibilities in this field in the coming future.

**CHAPTER 10**

**Limitations and Future Works**

• More advanced techniques such as online verification where dynamic data features such as the speed, pressure, pen’s position, azimuth/altitude angle etc. were not used in the system such technique would have improved result of application. Online signature verification saves time and offers better reliability and such technique can be added in future.

• Data that we collected were limited in quantity and the data collected were itself conflicting with each other due to variability nature of human signature. If the data are collected in large quantity and with appropriate nature it could have provided better accuracy.

• Preprocessing of dataset may not have been sufficient and further operation in data cleaning and preprocessing will ensure enhanced outcomes.

• Data for training was collected in hard copy format which limits the capability of collecting large number of training data. So data collection with improved technique such as with electronic signature capturing device can facilitates large number of sample collection in comparatively less time.

**REFERENCES**

[1] Surabhi Garhawal and N. Shukla, "A Study on Handwritten Signature Verification Approaches," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 2, no. 8, 2013.

[2] W. Hussein, A. S. Mostafa and O. Ibrahim, "Image Processing Based Signature Verification Technique to Reduce Fraud in Financial Institution," MATEC Web of Conferences, p. 76, 2016.

[3] M. Hanmandlu, "Neuro-Fuzzy Approaches to Signature Verification," 2006.

[4] D. S. A. Daramola and P. T. S. Ibiyemi, "Offline Signature Recognition using Hidden Markov Model (HMM)," International Journal of Computer Applications , vol. 10, 2010.

[5] B. Zhang, "Offline signature verification and identification by hybrid features and Support Vector Machine," International Journal of Artificial Intelligence and Soft Computing, vol. 2, pp. 302-320, 2011.

[6] A. Pansare and S. Bhatia, "Off-line Signature Verification Using Neural Network," Internatio nal Jo urnal o f Scientific Engineering Research , vol. 3, no. 2, 2012.

[7] G. Alvarez, B. Sheffer and M. Bryant, "Offline Signature Verification with Convolutional Neural Networks," Stanford.edu, Stanford, California, 2016.

[8] M. Shirdhonkar and M. Kokare, "Off-Line Handwritten Signature Retrieval using Curvelet Transforms," International Journal of Computer Science and Information Security, vol. 8, 2010.

[9] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," Department of Computer Science and Technology,University of Peloponnese, Peloponnese, Greece, 2007.