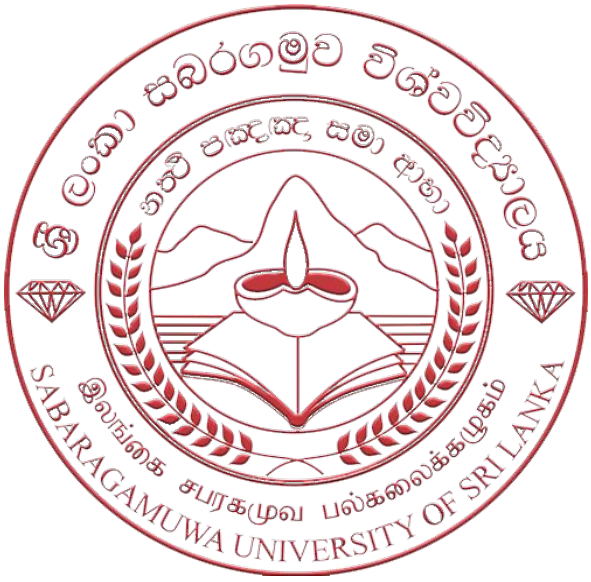
MINI PROJECT FINAL REPORT – IS4110



"D-TECT" – A SIGNATURE VERIFICATION MOBILE APPLICATION - GROUP 22

19APC3970	B.Gobihanath
19APC4010	R.M.C.P.K Rathnayake
19APC4078	A.B.T.Perera
19APC4015	T.Sajeeth

Supervised By:

Professor B.T.G.S Kumara

Mentored By:

Mr. B.A.Charun Umesh

Department of Computing and Information Systems

Faculty of Computing

Sabaragamuwa University of Sri Lanka

1 DECLARATION

We, the undersigned, solemnly affirm the originality and authenticity of this final report. It is our own work, and we have not incorporated any material without proper acknowledgment from sources previously submitted for academic purposes in any educational institution. Moreover, to the best of our collective knowledge and belief, this report contains no material previously published or authored by another person or any member of our group, except where appropriate references are provided in the text.

Furthermore, we hereby authorize Sabaragamuwa University of Sti Lanka to exercise the non-exclusive right to reproduce and distribute our final report, whether in its entirety or in part, through print, electronic, or any other medium. Each group member reserves the individual right to utilize the content, whether in its entirety or in part, for future endeavors such as articles or other publications.

Name of the Student Date	: B.Gobihanath : 2024.04.07	Signature of Student	:	
Name of the Student Date	: R.M.C.P.K Rathnayake : 2024.04.07	Signature of Student	:	
Name of the Student Date	: A.B.T.Perera : 2024.04.07	Signature of Student	:	
Name of the Student Date	: T.Sajeeth : 2024.04.07	Signature of Student	:	

2 CERTIFICATE OF APPROVAL

We hereby declare that this thesis is from the student's work and effort, and all other Sources of information used have been acknowledged. This thesis has been submitted with our approval.

Head of the Department:

Dr. L. Sugeeswari Lekamge,

Department of Computing & Information Systems,

Faculty of Computing,

Sabaragamuwa University of Sri Lanka Signature:.....

Date:

Internal Supervisor:

Professor B.T.G.S Kumara,

Dean Faculty of Computing,

Sabaragamuwa University of Sri Lanka. Signature:.....

Date:

Mentor:

Mr. B.A.Charun Umesh,

Data Scientist,

Dialog Axiata, Signature: .....

Date:

3 ACKNOWLEDGEMENT

We would like to express our profound gratitude to Professor.S.Vasanthapriyan, the dean of the Faculty of Computing, Mrs. Sugeeswari Lekamge, the head of the Department of Information Systems of Faculty of Computing, Professor. B.T.G.S.Kumara, our internal supervisor of our mini project, and Mrs.W.V.Subodhi Kalpani, the Department mini project coordinator.

Thank you to the Faculty of Computing and the Department of Information Systems at Sabaragamuwa University of Sri Lanka for your direction, strict oversight, and support of our project.

We also want to express our gratitude and great obligation to our mentor, Mr. B.A.Charun Umesh, Data scientist, for giving up his time to mentor us and support us. Special thanks to them for guiding me through every stage of the process with their expertise.

And we appreciate our family's unwavering support in helping us succeed.

Finally, we extend our sincere gratitude to everyone who has helped us in various ways with this project and send our best wishes and blessings.

Thank You,

Group 22

4 ABSTRACT

“D-Tect” Signature Verification System

Group 22

Department of Computing and Information Systems, Faculty of Computing, Sabaragamuwa University of Sri Lanka.

In today's technologically advanced world, the integrity of handwritten signatures holds paramount importance within university environments, serving as vital identifiers across a spectrum of academic activities. However, the prevalence of fraudulent signature usage poses a significant challenge, particularly in attendance tracking where manual detection methods often fall short. In response to this pressing issue, we introduce "“D-Tect”," an innovative Signature Verification Mobile App aimed at automating the verification process with precision and efficiency. By harnessing the power of advanced Machine Learning algorithms, “D-Tect” analyzes signature patterns seamlessly, enabling swift identification of genuine signatures amidst a sea of potentially fraudulent ones. Central to its functionality is a state-of-the-art Convolutional Neural Network (CNN) classifier model, meticulously trained to extract features from sample images, coupled with a contour detection-based signature cropping method that streamlines the extraction process. Moreover, the project underscores the significance of comprehensive data preprocessing techniques, ensuring the dataset's integrity and optimizing model performance. Through rigorous evaluation, “D-Tect” demonstrates remarkable accuracy rates in signature verification tasks, promising a transformative shift in administrative workflows within university environments. This abstract encapsulates the journey of conceptualization, development, and implementation of “D-Tect”, underscoring its potential to revolutionize signature authentication processes and uphold the integrity of academic institutions.

5 TABLE OF CONTENTS

6 LIST OF FIGURES

7 CHAPTER 1: INTRODUCTION

In today's technologically advanced world, ensuring the authenticity and security of handwritten signatures is of paramount importance, especially within university environments. Signatures hold immense significance as unique identifiers for students, faculty, and staff across various university-related activities, from document submissions to attendance tracking. Unfortunately, instances of fraudulent signature usage have emerged, posing a significant challenge to maintaining the integrity of these processes. Most students are doing some fraudulent activities when it comes to signing the attendance sheets. So, it's very difficult for every lecturer to detect that. Therefore, we have planned to develop a Signature Verification Mobile App named “D-Tect” for automatically and continuously determining whether a signature is genuine or illegal. This app will help the staff to determine whether the student's signature matches the original signature that has already been collected from the responsible student. While manual verification methods are time-consuming, error-prone, and fail to provide a comprehensive solution to tackle this issue effectively, we have planned to use the power of Machine Learning to automate and streamline the authentication process. Our system utilizes advanced machine learning algorithms to analyze and compare signature patterns, enabling swift and accurate verification. By implementing our Mobile Application, universities can establish a robust and efficient framework for signature verification. Faculty and administrative staff can simply upload scanned images or photos of documents containing signatures, allowing our system to instantly assess their authenticity. Leveraging a vast database of previously collected genuine signatures, the machine learning model seamlessly matches and evaluates the submitted signatures against the reference data, providing reliable results in real-time. This revolutionary app empowers lecturers to effortlessly compare sample images of students' signatures with the original ones already collected, assisting them in promptly identifying any discrepancies. With “D-Tect”, all staff need to do is scan or upload the image of the document, leaving the rest to the application's advanced capabilities.

7.1 MAJOR GOALS AND OBJECTIVES

- The primary objective of the D - Tec mobile application is to provide a reliable method for authenticating handwritten signatures particularly in the university context.

- To improve accuracy in recording and tracking student attendance.

- To alleviate the burden on lecturers who may struggle to detect fraudulent signatures manually.

- To promote a culture of punctuality and encourage students to actively engage in their academic responsibilities.

- To foster collaboration and teamwork among the group members.

- To provide analytical insights on attendance and generate statistical reports that can be used for decision-making purposes.

7.2 MOTIVATION

The motivation behind the development of our Signature Verification Mobile App, "D-Tect", stems from a pressing need within university environments for a streamlined and foolproof method of authenticating handwritten signatures. With instances of fraudulent signature usage on the rise, there's an urgent necessity for a solution that can effectively address this challenge. The gap between the manual verification methods currently in use and the need for swift, accurate, and continuous verification provided the impetus for our innovative approach. Inspired by the desire to bridge this gap and provide a reliable solution, we harnessed the power of Machine Learning to automate the authentication process.

Observing the increasing instances of fraudulent activities, particularly concerning attendance tracking and document submissions, we recognized the social concern among educators, administrators, and students alike. This collective concern prompted us to embark on a mission to develop a tool that not only addresses the immediate needs of universities but also aligns with the broader societal drive towards efficiency and integrity. Moreover, in our exploration of existing solutions, we noticed a conspicuous absence of a comprehensive mobile application catering specifically to signature verification needs in educational settings. This glaring gap in the market further fueled our determination to create a solution that not only meets but exceeds the expectations of users.

Our motivation is not merely driven by market opportunities but also by a genuine desire to contribute to the improvement of academic processes and uphold the integrity of educational institutions. By filling this void in the market with our innovative mobile application, we aim to facilitate a seamless connection between the social concerns of educators and administrators and the technological solutions that can address them effectively. In doing so, we envision "D-Tect" as not just a tool for university staff but as a catalyst for advancing the dialogue and practices surrounding signature verification and authentication in educational contexts.

7.3 THE SCOPE OF THE COMPLETED PROJECT

The completed project encompasses the development of a sophisticated Signature Verification Mobile App named "D-Tect", meticulously engineered to tackle the pressing challenge of ensuring the authenticity and security of handwritten signatures within university environments. Harnessing the power of cutting-edge machine learning algorithms, the app revolutionizes the verification process by automatically analyzing and comparing signature patterns against a vast repository of meticulously collected genuine signatures. This advanced functionality not only expedites the verification process but also delivers real-time results, enabling university staff to swiftly identify any irregularities or discrepancies in submitted signatures with unparalleled accuracy.

Crafted with a user-centric approach, the app boasts an intuitive interface that empowers faculty and administrative personnel to effortlessly scan or upload document images for verification. By offering a seamless and comprehensive solution, "D-Tect" significantly mitigates the risks associated with fraudulent signature usage, thereby bolstering the integrity of various university-related activities, from document submissions to attendance tracking. In addition to the groundbreaking Signature Verification Mobile App, the project encompasses the development of an innovative online platform designed to cater to the diverse educational needs of users. Offering a vast array of courses spanning various subjects, the platform serves as a veritable treasure trove of knowledge, allowing users to explore specific topics through meticulously curated roadmaps or conduct targeted searches tailored to their interests.

Furthermore, the platform's robust infrastructure facilitates user personalization, enabling individuals to create accounts, track their learning progress, and engage with educational resources in a manner that aligns with their unique preferences and goals. For those seeking additional guidance and support, the platform also offers the option to access tutoring services, further enriching the learning experience and fostering deeper understanding and mastery of course materials. In essence, the completed project represents a paradigm shift in how universities approach signature verification and online education, offering a comprehensive suite of tools and resources that empower users to navigate the complexities of academia with confidence and ease. From ensuring the integrity of signatures to providing access to high-quality educational content, "D-Tect" and the accompanying online platform are poised to redefine the educational landscape, fostering a culture of excellence, authenticity, and lifelong learning.

7.4 THE APPROACH AND ASSUMPTIONS WHILE CARRYING OUT THE PROJECT WORK

The approach taken in developing the "D-Tect" Signature Verification Mobile App involves a comprehensive utilization of machine learning algorithms to automate the authentication process. Our system relies on a back-end infrastructure comprising advanced machine learning models, specifically designed to analyze and compare signature patterns accurately. The primary assumption guiding our project development is that users, namely faculty and administrative staff within university environments, possess basic technological literacy and are capable of navigating mobile applications effectively. Additionally, it is assumed that these users have access to smartphones or other mobile devices required to interact with the "D-Tect" app seamlessly.

Furthermore, the project assumes the availability of scanned images or photos of documents containing signatures for verification purposes. It is expected that users can easily upload these images to the "D-Tect" app for analysis. The app is designed to provide real-time verification results, assuming timely responses from the system's database infrastructure, which houses a vast repository of previously collected genuine signatures. Additionally, the project assumes a foundational understanding of the English language among users, as the app's interface and instructions are presented in English.

In summary, the "D-Tect" Signature Verification Mobile App project is built upon the assumption of user proficiency in technology, access to requisite devices, availability of signature images for analysis, and timely database responses. These assumptions collectively underpin the development and functionality of the app, ensuring its effectiveness in addressing the challenges posed by fraudulent signature usage within university environments.

7.5 CONCISE SUMMARY OF MAJOR OUTCOMES

The “D-Tect” Signature Verification Mobile App project represents a significant advancement in ensuring the authenticity and security of handwritten signatures within university environments. By addressing the prevalent challenge of fraudulent signature usage, the project aims to enhance the integrity of various university-related activities, including document submissions and attendance tracking. Through the development of the “D-Tect” app, powered by advanced machine learning algorithms, the project offers a comprehensive solution to automate and streamline the authentication process.

One of the major outcomes of the project is the establishment of a robust and efficient framework for signature verification. By utilizing machine learning techniques, the “D-Tect” app enables swift and accurate analysis and comparison of signature patterns. This not only reduces the time and effort required for manual verification but also minimizes the potential for error, thereby enhancing the overall reliability of the authentication process.

Moreover, the “D-Tect” app empowers faculty and administrative staff by providing them with a user-friendly tool to determine the authenticity of signatures. By simply uploading scanned images or photos of documents containing signatures, users can instantly assess their authenticity using the app's advanced capabilities. Leveraging a vast database of previously collected genuine signatures, the app seamlessly matches and evaluates submitted signatures against reference data, providing reliable results in real-time. In addition to its core functionality, the “D-Tect” app facilitates prompt identification of any discrepancies between sample images of students' signatures and the original ones already collected. This capability not only assists staff in detecting fraudulent activities but also enables them to take timely action to uphold the integrity of university processes. Throughout the project development process, several assumptions have been made to ensure the effectiveness and usability of the “D-Tect” app. These include the assumption of user proficiency in technology, availability of signature images for analysis, and timely database responses. By carefully considering these assumptions and incorporating them into the app's design, the project aims to deliver a solution that meets the needs of university environments effectively and efficiently.

8 CHAPTER 2: BACKGROUND

The concept of "D-Tect", our Signature Verification Mobile App, arose from a deep understanding of the challenges faced within university environments concerning the authentication of handwritten signatures. Recognizing the pivotal role signatures play as identifiers in various university-related activities, from document submissions to attendance tracking, we observed a concerning trend of fraudulent signature usage among students. This widespread issue posed a significant obstacle to upholding the integrity of these processes, especially given the difficulty lecturers face in detecting such fraudulent activities manually.

Similar to the inception of Courseify, where we identified challenges in accessing timely updates on free courses and recognized financial constraints as barriers to education, our vision for "D-Tect" was to develop a solution that addresses these pressing issues. We aimed to leverage the power of Machine Learning to automate and streamline the authentication process, eliminating the time-consuming and error-prone nature of manual verification methods. Much like Courseify aimed to democratize education by centralizing free course offerings and mitigating financial barriers, "D-Tect" seeks to establish a robust and efficient framework for signature verification within university environments.

Just as Courseify aimed to simplify the educational journey by providing a centralized platform for individuals to explore and engage with free courses tailored to their interests, "D-Tect" aims to simplify the process of verifying signatures by allowing faculty and administrative staff to effortlessly upload scanned images or photos of documents containing signatures. By leveraging advanced machine learning algorithms and a vast database of genuine signatures, "D-Tect" empowers staff to swiftly and accurately determine the authenticity of signatures in real-time, thereby enhancing the integrity of university-related processes. In essence, "D-Tect", much like Courseify, aims to revolutionize its respective domain by providing a user-friendly solution that addresses key challenges, promotes efficiency, and ultimately enhances the overall experience for its users within university environments.

9 CHAPTER 3: SPECIFICATION AND DESIGN

UNDERSTANDING OF REQUIREMENTS

1. FUNCTIONAL REQUIREMENTS

- User Authentication
  Lecturers should be able to log in securely using their credentials.
- Signature Scanning
  The app should allow lecturers to scan signature sheets using the mobile phone camera or upload photos. The system should detect and extract individual signatures from the scanned or uploaded images.
- Signature Verification
  The app should employ signature verification algorithms to analyze the scanned signatures. It should compare the scanned signatures with the reference signatures stored in the database.
- Attendance Recording
  The system should accurately record attendance based on the verified signatures. Lecturers should be able to view and track attendance records for individual students or entire classes.
- Notification System
  The app should send notifications to lecturers regarding attendance status or any exceptions detected.
- Data Management
  The system should securely store and manage student information, including signatures and attendance records.

2. NONFUNCTIONAL REQUIREMENTS

- User-Friendly Interface
  The app should have an intuitive and user-friendly interface for both lecturers and students. It should be easy to navigate, with clear instructions for scanning and verifying signatures.
- Accuracy and Reliability
  The signature verification algorithms should deliver accurate and reliable results. The system should minimize false positives and false negatives during the verification process.
- Security and Privacy
  The app should ensure the security and privacy of student data and attendance records. Robust encryption and access control mechanisms should be implemented to protect sensitive information.
- Performance
  The app should be responsive and provide quick results during signature scanning and verification. It should handle a reasonable number of concurrent users without significant performance degradation.
- Compatibility
  The app should be compatible with popular mobile operating systems, such as iOS and android. It should support various image formats for signature scanning, ensuring flexibility for users.
- Scalability
  The system should be designed to accommodate future growth, allowing for the addition of more lecturers and students without compromising performance.

9.1 USE CASE DIAGRAMS

That shows the various actions a user can perform related to student attendance, including signing up, logging in, viewing the dashboard, checking attendance, scanning sign-in sheets, uploading sign-in sheets, and editing user profiles.

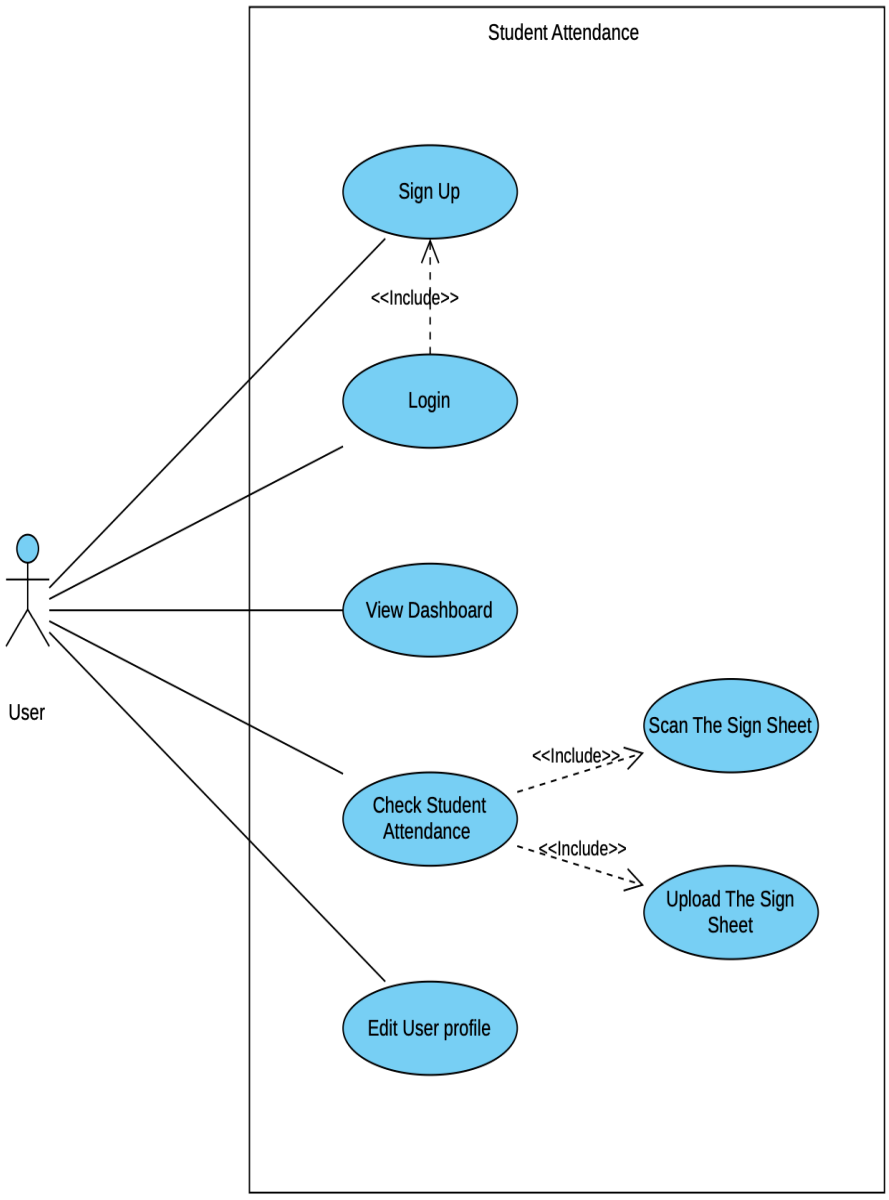


Figure 1 - Use Case Diagram

9.2 ER DIAGRAMS

It is an ER diagram that shows the relationships between user, document, and signature entities, including the attributes associated with each entity, such as user information, document details, and signature details.

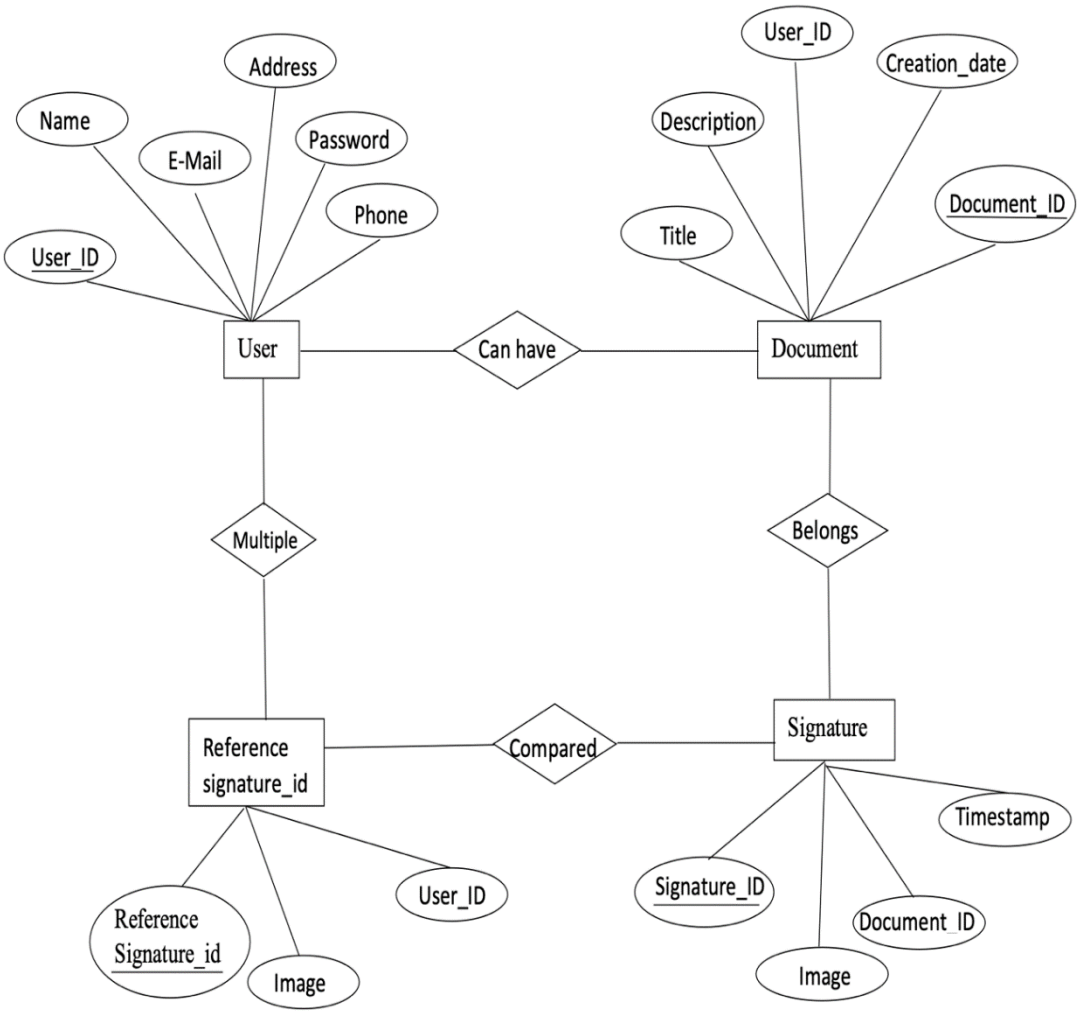


Figure 2 -ER Diagram

9.3 USER FLOW CHART

This flowchart outlines a mobile app's user journey. It starts with login options - existing users can log in directly, while new users can sign up. Logged-in users can view their profile with past reports and daily updates or scan a sign sheet and access reports.

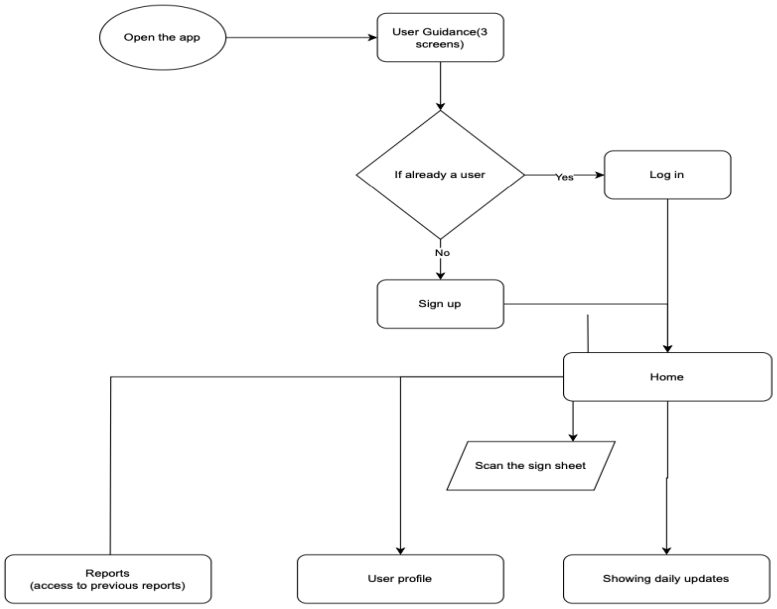


Figure 3 - User Flow Chart

9.4 CLASS DIAGRAM

This class diagram a Handwritten Signature Verification System, illustrating how various elements like User and Document interact with each other. It details attributes and methods associated with each class.

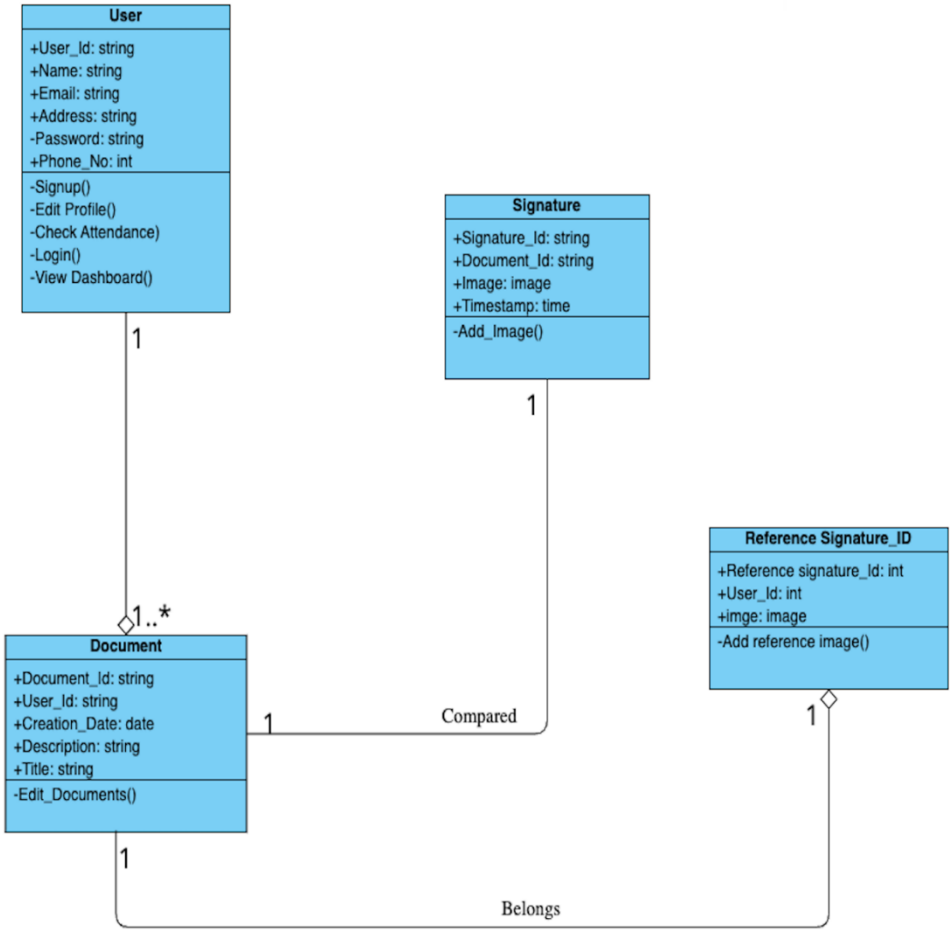


Figure 4 - Class Diagram

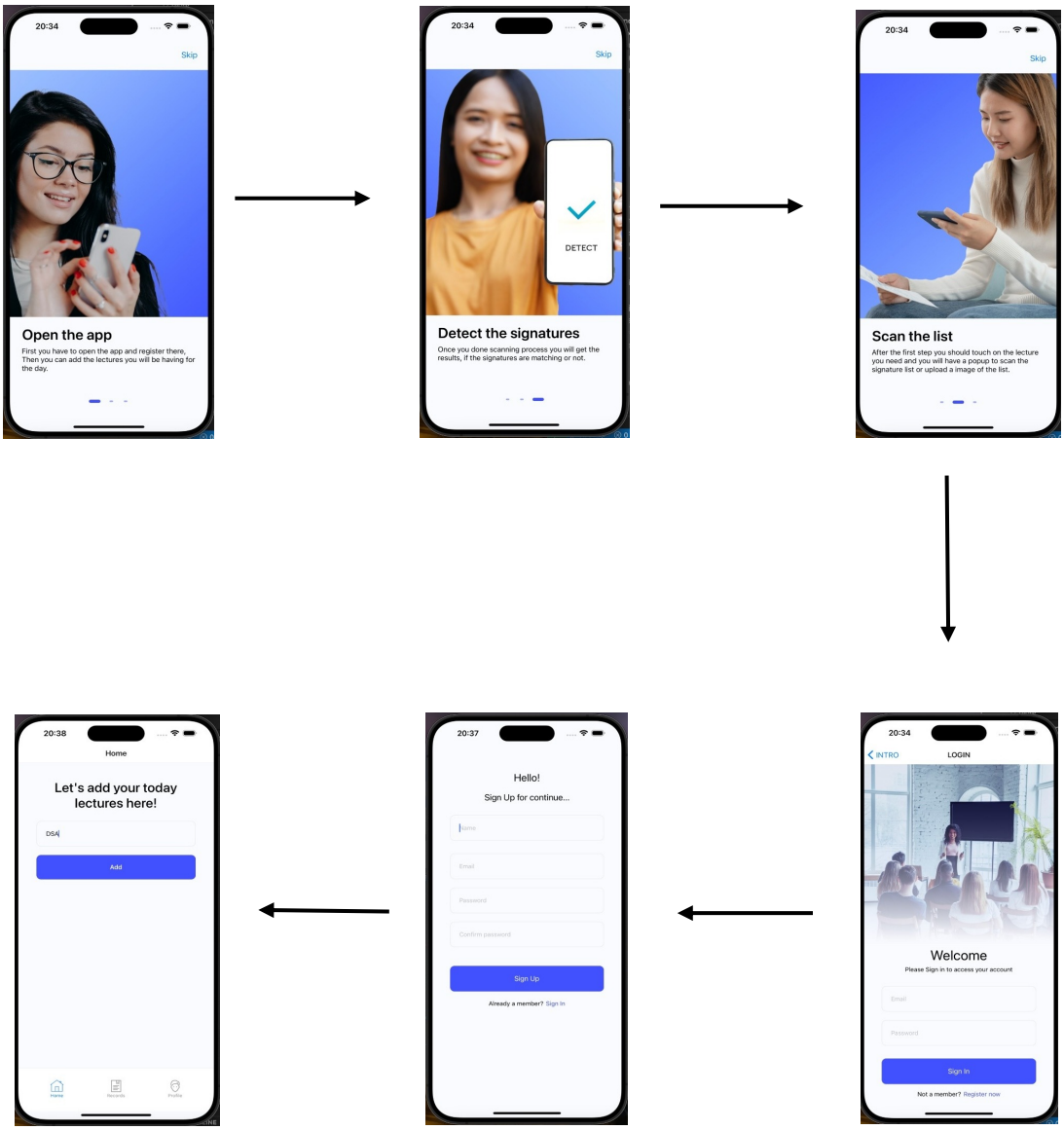
9.5 GANNT CHART

This image shows a project schedule timeline. It's a horizontal bar chart divided into sections representing project phases. Each section has a duration labeled in months, with markers potentially indicating weeks.

Activity	Time Duration															
	Month 1				Month 2				Month 3				Month 4			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Learning Techniques																
Create Project proposal Document																
Gathering Requirements & Analyze Data																
Design																
Development																
Implementation																
Testing																
Deployment & Maintenance																
End System																

Figure 5- Gantt chart

9.6 USER INTERFACE



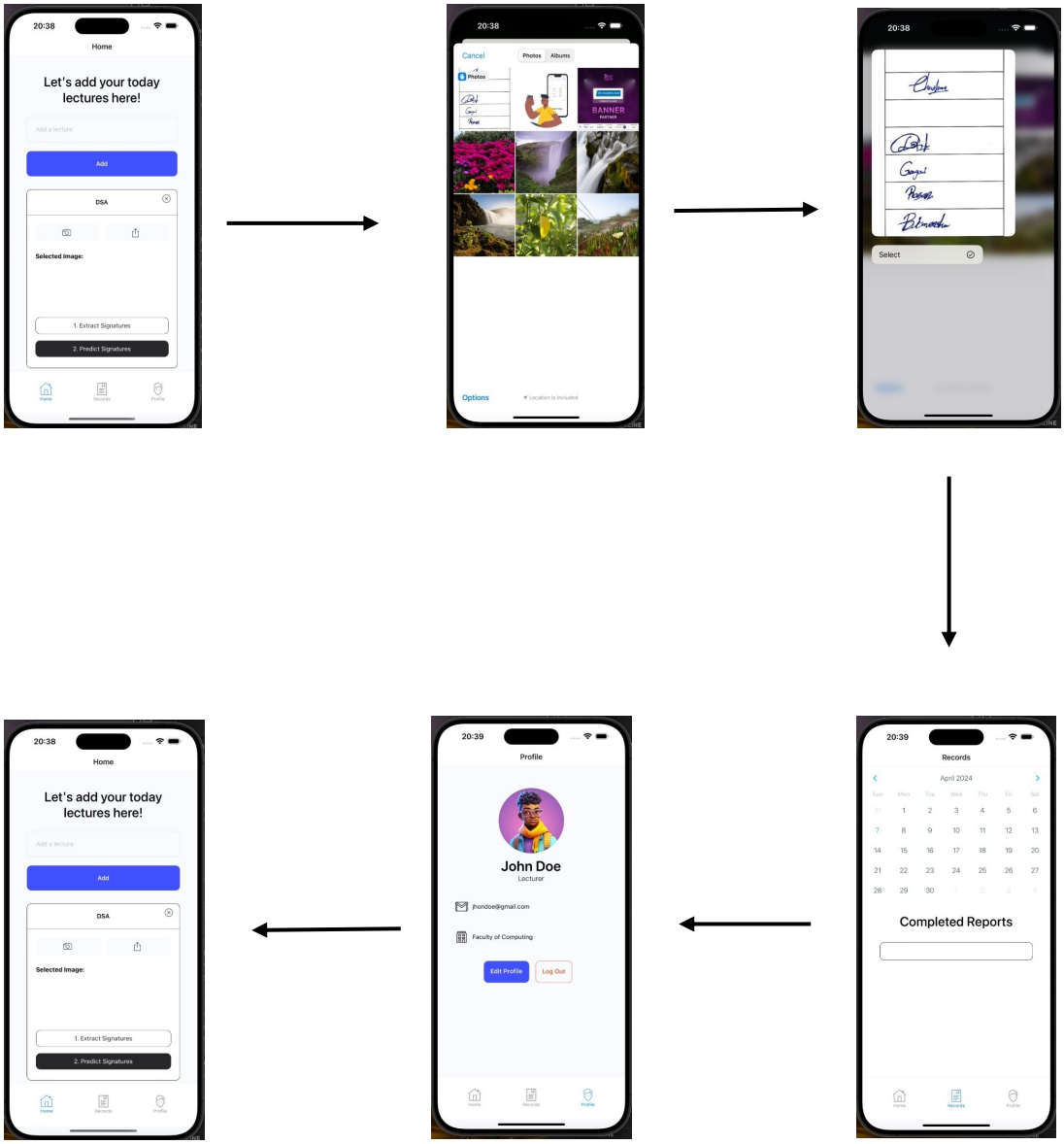


Figure 6 - User Interface

10 CHAPTER 4: IMPLEMENTATION

10.1 SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

- Visual Studio Code
- Google Colaboratory
- PyCharm & Jupyter Notebooks
- Figma
- Firebase

HARDWARE REQUIREMENTS

- Standard laptop or computer
- Minimum of 8 GB RAM (but more RAM would be beneficial, especially for handling larger datasets.)
- Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz 2.50 GHz [64-bit operating system, x64-based processor]
- Android Mobile with good camera quality
- Technologies and Tools Used
- Python
- React Native
- Flask
- Machine Learning : Tensorflow, Keras, Convolutional Neural Networks, OpenCV

10.2 ILLUSTRATION OF A NON-STANDARD OR INNOVATIVE WAY OF IMPLEMENTING AN ALGORITHM AND DATA STRUCTURE

An innovative aspect of the implementation was the integration of an CNN classifier model for extracting features from the sample images and a contour detection based signature cropping method for cropping signatures from an attendance sheet.

1. CNN Classifier

   Our effort has resulted in the development of a state-of-the-art Convolutional Neural Network (CNN) classifier model, marking a noteworthy progression in the field of image recognition technology. Our CNN model, which uses cutting-edge deep learning algorithms, excels in classifying complicated visual data with remarkable efficiency and accuracy. Our unique design combines cutting-edge attention methods to dynamically focus on important elements in images, improving classification performance. This is what makes our model unique. In addition, our model uses sophisticated regularization methods to reduce over fitting and enhance generalization, guaranteeing stable results on a variety of datasets.

```
model = Sequential()
model.add(Rescaling(1./255, input_shape=input_shape))

model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D())

model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D())

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

Figure 7 - Model Summary

2. Contour Detection and Canny Edge Detection

In order to automatically extract individual signatures from attendance sheets, we have created a sophisticated signature cropping model in our project utilizing contour detection and canny edge detection algorithms. This creative method is a major step towards automating the processing of attendance information. Our approach makes use of computer vision techniques to detect edges and contours, which allows it to precisely identify signature regions and segment them from the surrounding backdrop. Through the integration of these approaches into our workflow, we have successfully separated signatures with minimum manual intervention, yielding very trustworthy results. The ability to automatically extract signatures simplifies the attendance monitoring procedure, resulting in time savings and increased productivity. Our application shows how computer vision technology may improve administrative work and expedite procedures related to document processing.

```
gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blurred, 50, 150)
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
largest_contour = max(contours, key=cv2.contourArea)
x, y, w, h = cv2.boundingRect(largest_contour)
largest_box_image = resized_image[y:y + h, x:x + w]

# Divide the largest_box_image into 10 equal parts
num_parts = 10
part_height = h // num_parts

for i in range(num_parts):
    ext_img = largest_box_image[i * part_height: (i + 1) * part_height, :]

    # Resize small box and encode it in base64
    ext_image = cv2.resize(ext_img, (256, 256))
    _, buffer = cv2.imencode('.jpg', ext_image)
    ext_image_encoded = base64.b64encode(buffer).decode('utf-8')

    # Store the base64 encoded small box image
    temp_ext_sign.append(ext_image_encoded)
```

Figure 8 - Extraction Model

10.3 DIFFICULTIES INVOLVING EXISTING SOFTWARE

- **Integration Challenges**
  D-Tect with existing university systems, such as document management platforms or student databases, may be complex due to differences in data formats, protocols, and APIs.

- **Compatibility Issues**
  Making sure that different devices, operating systems, and software versions work together might be difficult, especially if users come from different technological backgrounds.

- **Data Migration Concerns**
  Maintaining data security and integrity during the transfer of existing signature data from legacy systems to "D-Tect" may be challenging, particularly if the data formats or architectures are incompatible.

- **Scalability Constraints**
  If the underlying software design lacks enough scalability capabilities, scaling "D-Tect" to meet expanding user bases or rising needs for signature verification may prove difficult.

- **Performance Optimization**
  It could be necessary to carefully adjust algorithms, manage resources, and configure systems in order to maximize "D-Tect"'s performance in order to efficiently process high numbers of signature verification requests.

- **Security Vulnerabilities:**
  To protect the confidentiality and integrity of critical signature data, it is essential to find and address any potential security flaws in "D-Tect"'s dependencies or current software components.

- **Regulatory Compliance**
  Adding functionality or making significant changes to the software may be necessary to ensure that "D-Tect" conforms to all applicable industry standards and data protection laws, such as GDPR and HIPAA.

10.4 LACK OF APPROPRIATE SUPPORTING SOFTWARE

1. REACT NATIVE

- Unable to add the Mobile Scanner
  The absence of native support for mobile scanning in React Native should be addressed by using third-party libraries/plugins such as React Native Camera or by developing bespoke native modules to access device camera functionality.

- Unable to connect the Flask API
  One potential solution to the problem of linking a Flask API with React Native is to use tools such as Axios or Fetch API to transmit and receive data between the two platforms via asynchronous HTTP calls. To accept queries from the React Native app, make sure the Flask API is set up correctly to support CORS (Cross-Origin Resource Sharing).

2. TENSORFLOW/KERAS FOR MACHINE LEARNING

- Restricted Pre-trained Models for Verification of Signatures:
  There are several pre-trained models available in TensorFlow/Keras, but it might be difficult to locate one that is especially designed for signature verification.
  Solution: Using a specific dataset of signatures, custom train models. Use transfer learning techniques to fine-tune a model for signature verification that was learned on a comparable problem.

- Sophisticated Model Deployment on Mobile Devices:
  Due to resource limitations, deploying sophisticated TensorFlow/Keras models directly on mobile devices may cause performance problems.
  Solution: Reduce latency and compute requirements by converting and optimizing models for on-device execution using TensorFlow Lite for mobile devices.

3. PYTHON FLASK FOR BACKEND SERVICES

- The micro web framework Flask may initially provide scalability issues when processing large numbers of requests for signature verification.
- Asynchronous Request Handling: Flask's synchronous operation by default may cause performance issues for real-time signature verification when there is a high volume of requests.

4. OpenCV FOR IMAGE PROCESSING

- Although OpenCV offers a wide range of image processing tools, developers may find it difficult to perform intricate preprocessing tailored to signature photos.
- On mobile devices, executing extensive OpenCV operations for real-time signature verification might be resource-consuming.

10.5 OVER-AMBITIOUS PROJECT AIMS

- Instantaneous Verification
  Because of processing limitations and accuracy concerns, it may be impractical to expect an instantaneous and ongoing authenticity judgment.

- 100% Reliability
  Although great accuracy is desired, it may not be possible to get perfect results because of variances in handwriting and other reasons.

- Complete Automation
  Taking into account the intricacy of signature verification, it may not be possible to anticipate total automation at first.

- Immediate and Universal Implementation
  It might be difficult to roll out the app instantly across all university departments; considerable preparation and adaption would be needed.

- Universal Solution
  It may be too much to expect "D-Tect" to handle every university signature verification demand at first, as various departments may have distinct needs and difficulties. Although great accuracy is the goal, handwriting variances and other variables may make faultless outcomes unattainable.

11 CHAPTER 5: RESULTS AND EVALUATION

11.1 METHODOLOGY

Figure 1 depicts the flow diagram which have been used for this project. The figure depicts the methods and methodology used in this handwritten signature verification system.

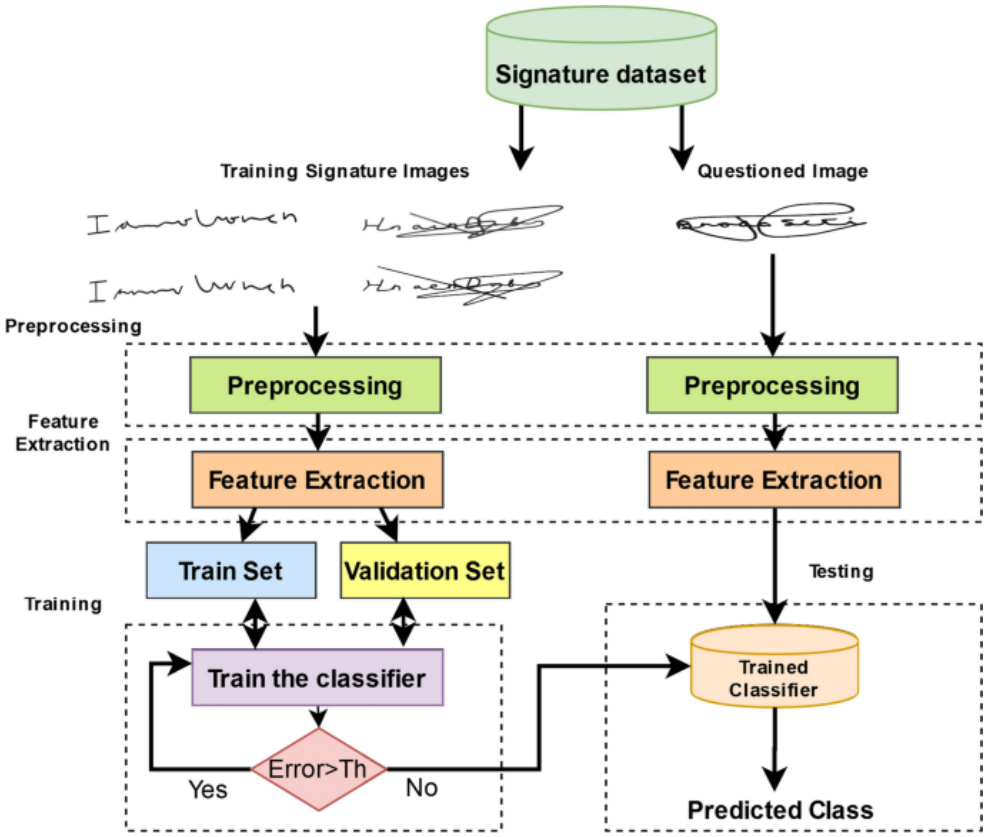


Figure 9-Development Method

11.1.1 DATA COLLECTION

The data collection process aimed to acquire a diverse and comprehensive dataset of handwritten signatures from students within our faculty. In the data collection phase, we gathered signatures from 10 students. Each participant provided 500 signature samples, totaling 5000 images across all participants. We divided these signatures into three groups: 80% (4000 signatures) for training the model, 10% (500 signatures) for fine-tuning it, and another 10% (500 signatures) for testing its accuracy. With a batch size of 128, we're processing 128 signatures at a time during training.

11.1.1.1 PARTICIPANTS

Ten students from our faculty voluntarily participated in the data collection process. These participants were selected to ensure a representative sample of signature styles and variations commonly encountered in university-related activities. Strict adherence to ethical guidelines was maintained throughout the data collection process. Participants were informed about the purpose of data collection, confidentiality measures, and their rights regarding data usage

11.1.1.2 IMAGE ACQUISITION

The data collection procedure involved capturing handwritten signature images using a mobile camera. Each participant provided 500 signature samples, totaling 5000 images across all participants. To maintain consistency and organization, images were acquired individually, with participants instructed to sign on blank paper sheets.

11.1.1.3 DATA STORAGE

Signature images were securely stored in a cloud-based repository, specifically Google Drive. Separate folders were created for each participant, labeled as "Person1" through "Person10". This organized structure enables efficient data management and retrieval throughout the project lifecycle. Figure showcases the arrangement of signature data storage.

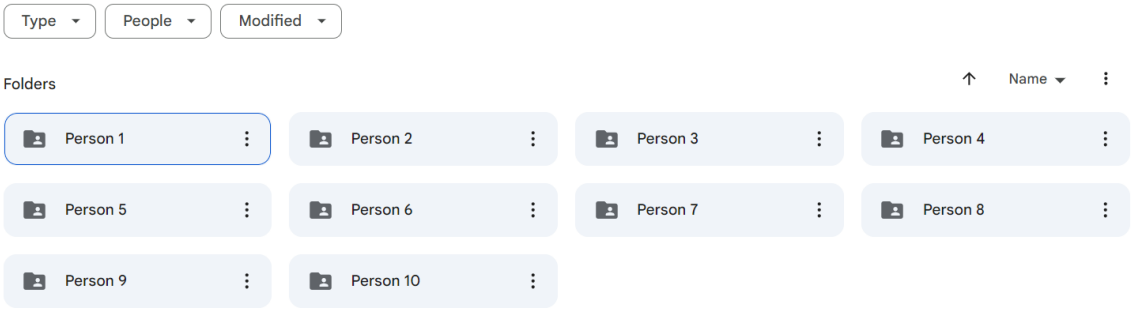


Figure 10-Arragement of Data Storage

11.1.1.4 FOLDER ORGANIZATION

Within each participant's folder, signature images were meticulously organized to ensure clarity and ease of access. Figure illustrates the folder organization within each directory, showcasing the systematic organization implemented for each participant's signature images.

Figure 11-Collected Data

11.1.1.5 QUALITY ASSURANCE

Participants were instructed to sign clearly and consistently within the designated space, minimizing variations due to extraneous factors. Additionally, images were reviewed for clarity, ensuring adequate resolution and visibility of signature details.

11.1.2 DATA PREPROCESSING

In image pre-processing, the different techniques like cropping the image, scaling to appropriate dimension and shuffling are performed to enhance the quality and suitability of the signature images for training the model. These techniques include cropping the images to remove unnecessary background, scaling them to a standardized dimension of 256 pixels × 256 pixels for consistency, and discarding any irrelevant images to ensure the dataset's integrity. Additionally, the dataset is divided into batches, with each batch containing 128 images to facilitate efficient training. Furthermore, rescaling is applied to the pixel values of the images, typically by dividing them by 255, to normalize the data and ensure uniformity in the input features. These pre-processing steps collectively optimize the dataset for subsequent model training, enabling more effective learning and improved accuracy in signature verification. In the context of signature verification, shuffling ensures that the model encounters a diverse and representative mix of genuine and forged signatures during training. This randomization of the input data sequence offers several benefits. Firstly, it accelerates the training process by facilitating faster convergence, thus reducing the number of epochs required to train the model effectively. Additionally, shuffling helps mitigate bias in the model by presenting data in a randomized order, preventing over fitting to specific patterns or sequences present in the input data. Integrating shuffling into the pre-processing pipeline contributes to improved efficiency, effectiveness, and reliability of the signature verification model.

11.1.3 FEATURE EXTRACTION

11.1.3.1 INTRODUCTION TO DEEP NEURAL NETWORKS

Deep neural networks (DNNs) comprise multiple non-linear computational units or neurons organized in a layer-wise fashion to extract high-level, deeper, robust, and discriminative features from the underlying data. DNNs have been used widely for data-driven modeling. Deep neural networks are frequently used for their accuracy and adaptive nature in the research field of automatic classification tasks. State-of-the-art architectures and pretrained networks exist and can be converted and fine-tuned to handle newer classification tasks. These networks are usually composed of several layers connected to each other with many parameters each. By using a training dataset, the training phase computes these parameters to handle classification of newer data [1].

11.1.3.2 INTRODUCTION TO CNN

One of the most popular deep neural networks is Convolutional Neural Networks (also known as CNN or ConvNet) in deep learning, especially when it comes to Computer Vision applications. They consist of a series of convolutional and pooling layers that extract relevant features from the input image, followed by one or more fully connected layers that use these features to make a prediction [2]. In the model-building process, feature extraction is crucial. Convolutional neural networks were utilized mostly for feature extraction. The convolution layer is the primary component of the CNN. Convolution layers are employed for obtaining several features from an input picture. The result of this convolution layer is known as the "Feature Map". Later, this feature map is sent to further layers, which learn various different features from the input picture. Pooling layers can be utilized to minimize the size or dimensions of a feature map. The pooling layer is built by using the pooling operators to aggregate information inside each tiny region of the input feature channels, and then down sampling the results [3]. Different pooling approaches, such as max pooling, min pooling, average pooling, gated pooling, tree pooling, etc., are employed in different pooling layers. By far the most widely used technique is max pooling [4].

The last section (or layers) of any CNN design (used for classification) often consists of fully-connected layers, in which every neuron is linked to every other neuron in the layer above it. In fully connected layers, neurons are linked between layers by means of weights, biases, and other components. The CNN architecture's output layer, or classifier, is the

last Fully-Connected layer [4]. An activation function can be used to decide whether a neuron should be activated or not. ReLu, SoftMax and Sigmoid are most commonly used Activation function. A flatten layer is often used to reduce multidimensional data to a one-dimensional array. This is frequently required when moving from convolutional to fully connected layers (which require one-dimensional input). This layer reshapes the previous flatten layer's output into a one-dimensional vector. The last layer is frequently made up of neurons that correspond to the number of classes in the dataset. Each neuron belongs to a class, and the neural network's output provides the probability distribution for these classes.

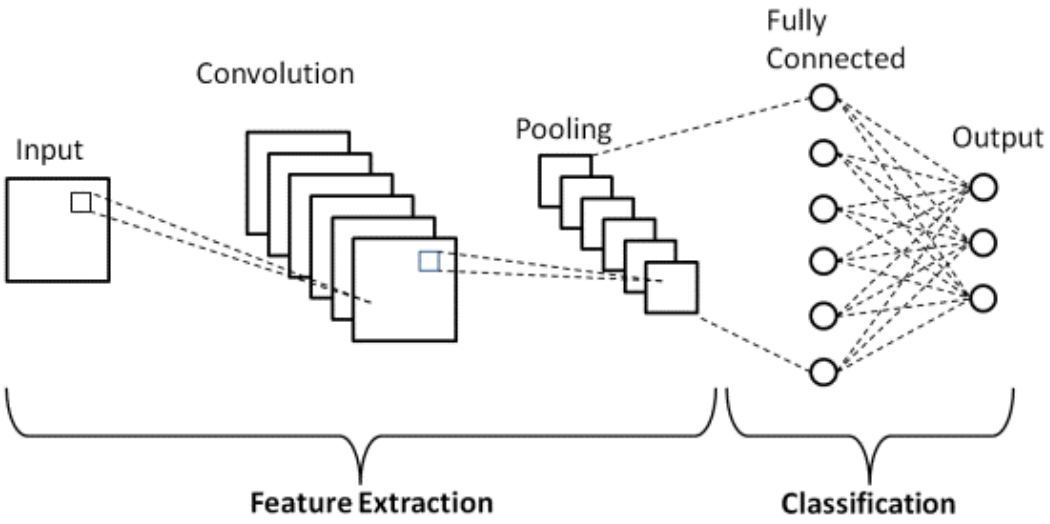


Figure 12-Sample CNN Architecture

11.1.3.3 PROPOSED CNN MODEL

We have constructed a Sequential model, with 1 rescaling layer, 3 convolutional layers consisting of 16, 32, and 64 filters respectively, 3 max-pooling layers, one dropout layer, one flattened layer and 2 fully-connected dense layers. Before training or the input images are resized to a reasonable size (256 pixels × 256 pixels).

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 256, 256, 3)       0

 conv2d_3 (Conv2D)           (None, 254, 254, 16)      448

 max_pooling2d_3 (MaxPoolin  (None, 127, 127, 16)      0
 g2D)

 conv2d_4 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_4 (MaxPoolin  (None, 62, 62, 32)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 60, 60, 64)        18496

 max_pooling2d_5 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 dropout_1 (Dropout)         (None, 30, 30, 64)        0

 flatten_1 (Flatten)         (None, 57600)             0

 dense_2 (Dense)             (None, 128)               7372928

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 7397802 (28.22 MB)
Trainable params: 7397802 (28.22 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 13-Summary of Develop CNN Model

Input pixel values were normalized using the Rescaling layer. Convolutional layers with 16, 32, and 64 filters respectively were employed, each comprising 3x3 filters with a default stride of 1 pixel followed by ReLU activation functions. Three MaxPooling layers were utilized for reducing the spatial dimensions (width and height) of an image or feature map. A dropout rate of 25% was applied after the third convolutional layer to prevent overfitting. Following the convolutional and pooling layers, the feature maps were flattened into a one-dimensional vector via the Flatten layer, thus preparing them for input into the fully connected layers. The neural network's dense layers comprised 128 neurons each, augmented by ReLU activation functions. The output of the last fully connected layer is fed into an N-way SoftMax activation function, resulting in a distribution across 10 class labels.

11.1.3.4 MODEL TRAINING

The model training process commenced with the division of the dataset into three subsets: 80% for training, 10% for validation, and an additional 10% for testing. The model was trained over 20 epochs, with around 32 batches for training and 4 batches for validation, ensuring efficient computation over the dataset. Additionally, to monitor the model's performance and prevent overfitting, an early stopping callback was integrated into the training process. The training of the model took about 5 to 6 hours. Throughout the training process we monitored key performance metrics including training accuracy, validation accuracy, training loss, and validation loss. Following the completion of training, the model achieved a training accuracy of approximately 97.91% and a validation accuracy of approximately 98.62%. These metrics were obtained after meticulously training the model over 20 epochs. Following the training process, the trained model was saved into a .h5 file format, ensuring its preservation for future use and deployment.

```
Epoch 1/20
32/32 [==============================] - 996s 9s/step - loss: 2.3642 - accuracy: 0.1516 - val_loss: 1.9954 - val_accuracy: 0.2715
Epoch 2/20
32/32 [==============================] - 382s 8s/step - loss: 1.7058 - accuracy: 0.4388 - val_loss: 1.3748 - val_accuracy: 0.5273
Epoch 3/20
32/32 [==============================] - 386s 8s/step - loss: 1.2031 - accuracy: 0.5783 - val_loss: 0.9703 - val_accuracy: 0.6621
Epoch 4/20
32/32 [==============================] - 388s 9s/step - loss: 0.8844 - accuracy: 0.6827 - val_loss: 0.8832 - val_accuracy: 0.6602
Epoch 5/20
32/32 [==============================] - 403s 9s/step - loss: 0.7367 - accuracy: 0.7507 - val_loss: 0.6719 - val_accuracy: 0.7735
Epoch 6/20
32/32 [==============================] - 386s 9s/step - loss: 0.5122 - accuracy: 0.8240 - val_loss: 0.4482 - val_accuracy: 0.8398
Epoch 7/20
32/32 [==============================] - 394s 9s/step - loss: 0.4534 - accuracy: 0.8413 - val_loss: 0.3631 - val_accuracy: 0.8730
Epoch 8/20
32/32 [==============================] - 381s 9s/step - loss: 0.5330 - accuracy: 0.8122 - val_loss: 0.4567 - val_accuracy: 0.8340
Epoch 9/20
32/32 [==============================] - 390s 9s/step - loss: 0.3874 - accuracy: 0.8698 - val_loss: 0.2688 - val_accuracy: 0.9043
Epoch 10/20
32/32 [==============================] - 394s 9s/step - loss: 0.2646 - accuracy: 0.9165 - val_loss: 0.2422 - val_accuracy: 0.9180
Epoch 11/20
32/32 [==============================] - 407s 9s/step - loss: 0.1900 - accuracy: 0.9366 - val_loss: 0.1087 - val_accuracy: 0.9746
Epoch 12/20
32/32 [==============================] - 408s 9s/step - loss: 0.1784 - accuracy: 0.9422 - val_loss: 0.1982 - val_accuracy: 0.9336
Epoch 13/20
32/32 [==============================] - 388s 9s/step - loss: 0.3576 - accuracy: 0.8841 - val_loss: 0.2874 - val_accuracy: 0.9062
Epoch 14/20
32/32 [==============================] - 390s 9s/step - loss: 0.2134 - accuracy: 0.9276 - val_loss: 0.1548 - val_accuracy: 0.9473
Epoch 15/20
32/32 [==============================] - 393s 9s/step - loss: 0.1508 - accuracy: 0.9495 - val_loss: 0.0842 - val_accuracy: 0.9824
Epoch 16/20
32/32 [==============================] - 391s 9s/step - loss: 0.0849 - accuracy: 0.9749 - val_loss: 0.0760 - val_accuracy: 0.9824
Epoch 17/20
32/32 [==============================] - 397s 9s/step - loss: 0.0832 - accuracy: 0.9726 - val_loss: 0.0515 - val_accuracy: 0.9902
Epoch 18/20
32/32 [==============================] - 413s 9s/step - loss: 0.0582 - accuracy: 0.9857 - val_loss: 0.0532 - val_accuracy: 0.9883
Epoch 19/20
32/32 [==============================] - 388s 9s/step - loss: 0.0703 - accuracy: 0.9794 - val_loss: 0.0730 - val_accuracy: 0.9824
Epoch 20/20
32/32 [==============================] - 390s 9s/step - loss: 0.1053 - accuracy: 0.9629 - val_loss: 0.0494 - val_accuracy: 0.9902
```

Figure 14-Model Training

11.1.4 PREDICTIONS

After the completion of the training process predictions were made using the model's learnt insights and parameters, with an emphasis on accuracy and reliability. The saved model is loaded to start making predictions on both tested and unknown data. Testing data, a subset of the original dataset, was used as a benchmark to assess the model's effectiveness and its capacity for prediction. Furthermore, predictions were extended to previously unseen data, allowing for evaluation of the model's generalizability to unexpected situations. In the prediction phase, a custom predict function is used to evaluate the trained model's predictive capabilities. This function takes the trained model and a questioned signature image as input arguments. Then the image is first translated into an array representation, and subsequently widened into a specific dimension to meet the model's input specifications. The model's predict method is used for making predictions, producing class probabilities for each possible result. The function finds the highest value in the prediction matrix along with its index.

The highest probability index is used to determine the estimated class, and the greatest probability multiplied by 100 and rounded to two decimal places is used to establish the related confidence level.

```
Predictions Matrix:
 [[5.4968208e-01 2.5169855e-01 1.0464207e-01 2.1638053e-03 1.5148951e-03
  7.4203959e-04 7.7473393e-05 5.8136757e-02 2.6954627e-02 4.3876306e-03]]


Predictions Matrix Elements:
0.5496821
0.25169855
0.10464207
0.0021638053
0.0015148951
0.0007420396
7.747339e-05
0.058136757
0.026954627
0.0043876306


Maximum Value of Predictions: 0.5496821
Index of Maximum Value: 0
Confidence: 0.5496821
Confidence Percentage: 54.97%
Predicted Class: Person1
```

Figure 15-Predictions

11.1.5 CROPPING SIGNATURES FROM AN ATTENDANCE SHEET

An extraction model was created to automatically extract signatures from attendance sheets, simplifying the procedure and reducing the need for human intervention. We have created a sample attendance sheet for our project. Figure 4 displays the sample attendance sheet we created for our project. The extraction approach uses contour detection and canny edge detection as primary techniques for recognizing and isolating signatures from attendance sheets. Contour detection is a key operation in computer vision. It includes determining the borders of objects within an image.

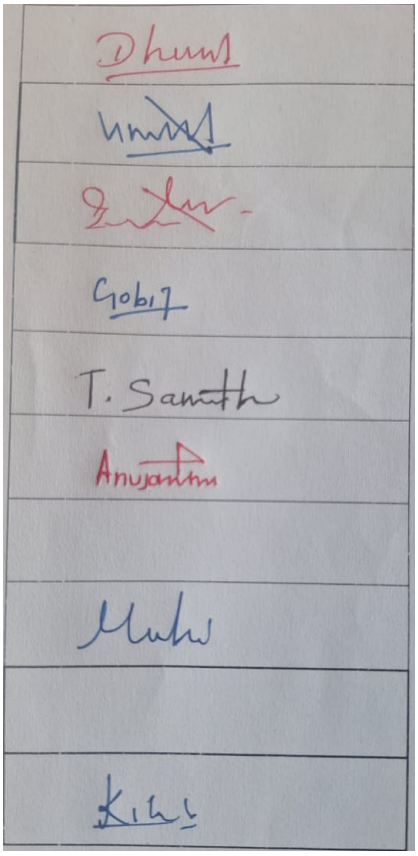


Figure 16-Sample Attendance Sheet

11.1.5.1 INTRODUCTION TO CONTOUR DETECTION AND CANNY EDGE DETECTION

Contour detection and Canny edge detection are essential techniques in computer vision that are commonly used for object detection, shape analysis, and segmentation of images. Using contour detection, we can recognize the edges of objects and easily locate them in an image. Contours, which are curves that unite continuous points along an item's boundary, encapsulate its shape and structure, making activities like object recognition and shape analysis more efficient. Meanwhile, the Canny edge detection method, known for its accuracy and noise elimination abilities, thoughtfully extracts edges by smoothing the image with a Gaussian filter to remove noise [5]. Canny edge detection is frequently used as a phase of preprocessing to improve the edges in an image, and contour detection is then used to detect and extract the contours or borders of objects based on these enhanced edges [5].

11.1.5.2 DEVELOPMENT OF EXTRACTION MODEL

This extraction model, extracting signatures from images using the OpenCV library, a powerful tool for image processing tasks. The process begins by loading the target image and resizing it to a specified width and height to ensure consistency in further analysis. Conversion to grayscale simplifies subsequent operations, followed by the application of Gaussian blur to reduce noise and enhance edge detection. Canny edge detection identifies prominent edges in the image, facilitating the extraction of signature contours. By finding contours and selecting the largest one, presumed to correspond to the signature, the script isolates the signature region within its bounding rectangle. This region is then partitioned into ten equal parts, each representing a segment of the signature. Finally, the script saves each segment as a separate image file, facilitating further analysis or processing. Through a combination of image manipulation and contour analysis, the script provides an automated solution for extracting signatures from images, with potential applications in signature verification systems or document processing workflows.

11.1.5.3 RESULTS OF THE EXTRATION MODEL

The extraction model presented in our project showcases a systematic approach to automating the process of extracting signatures from images. Through advanced image processing techniques such as Gaussian blur, Canny edge detection, and contour analysis, the model demonstrates robustness in detecting and delineating signature contours, even in the presence of noise or variations in image quality. The effectiveness of the extraction model is evident in its ability to accurately identify signature regions within images and segment them into individual parts for further analysis. By dynamically adapting to different image sizes and resolutions, the model enhances its versatility and applicability across various scenarios. The successful extraction of signatures from the provided images, as demonstrated in the attached cropped images, further validates the effectiveness of the extraction model. However, it's important to note that the model's performance may be influenced by factors such as the complexity of signature patterns and variations in handwriting styles. Further evaluation and optimization of the extraction model could enhance its performance and efficiency, contributing to its potential as a valuable tool for automating signature extraction tasks in practical applications. Here is the results of extraction model, after extract the scanned image we provide.

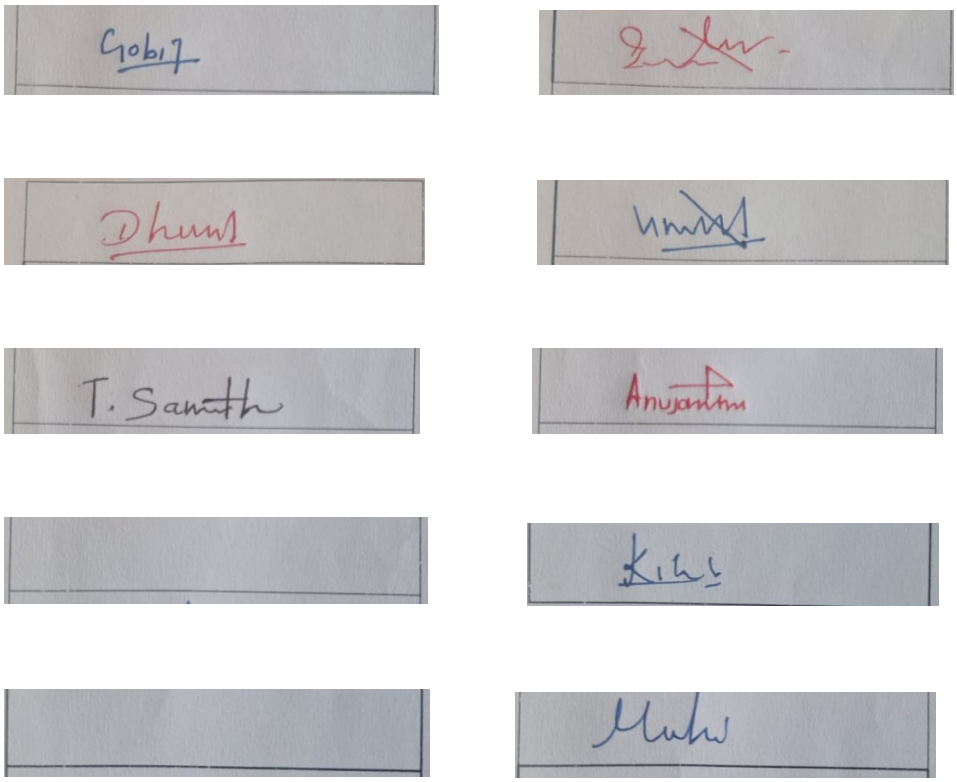


Figure 17-Output of the Extraction Model

11.2 RESULTS AND EVALUATION

Several crucial tasks were included in our evaluation process, all of which were intended to give readers an in-depth understanding of the model's capabilities. First, using graph plotting, we carefully investigated training and validation accuracy. These graphs demonstrated how the model's accuracy evolved as it gained knowledge from the data over time. Furthermore, we examined the changing patterns of training and validation loss, keeping an eye on the model's training process' convergence and spotting both over fitting and under fitting situations. Furthermore, we analyzed confusion matrices on both the validation and test datasets to gain further insight into the model's effectiveness in classification. These matrices gave an in-depth analysis of the model's accuracy in classification, showing its strengths and weaknesses for different classes. Lastly, we demonstrated the model's effectiveness and practicality by using it to forecast sample photos from the training dataset.

11.2.1 ANALYZING TRAINING AND VALIDATION ACCURACY

Throughout training, the model learns from the data and adjusts its parameters to produce more accurate predictions. Validation happens apart from training and serves as a test to see how effectively the model can generalize its learnings to new, previously unknown data. Graph plotting enables us to see how the model's accuracy evolves over time, especially throughout training epochs or iterations. Each epoch represents a full run through the whole dataset during the training phase. Through graphing accuracy over epochs, we may determine if the model's performance increases or levels with time. Figure shown below describes how the model's accuracy changes over time.

Figure 18-Traning Validation Accuracy

Here we can see the training and validation accuracies grow concurrently, it means that the model is performing well not only on the data it was trained on, but also on new, previously unknown data. This indicates that the model is effectively learning from the training data and can generalize its knowledge to make correct predictions on new cases.

11.2.2 ANALYZING TRAINING AND VALIDATION LOSSES

Apart from assessing accuracy, we carried out an in-depth examination of training and validation loss over epochs. These graphs gave us a thorough understanding of how the model's performance varied between training and validation. We were able to identify the times when the model experienced struggles with learning or was over fitting to the training set by closely observing the training and validation loss. The model's training and validation losses changes over time shown in figure.

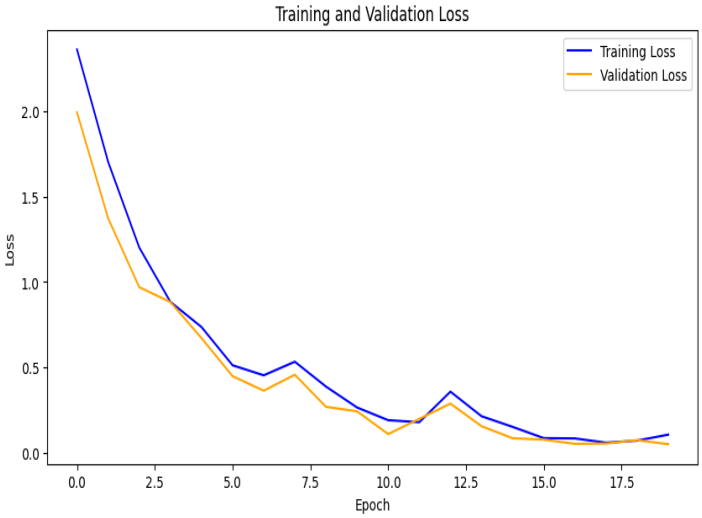


Figure 19-Tranning and Validation Loss

Here both training and validation losses are decreasing in a similar manner. It means in addition to only memorizing the training set, the model is also identifying the underlying patterns in the data, which enable it to predict new, unknown cases with accuracy. This indicates that the model is not over fitting to the training data. Our investigation of the training-validation loss generalization gap demonstrates a balanced model performance, showing that our model is not over fitted or under fitted.

11.2.3 CONFUSION MATRIX

Confusion matrix is an effective method for assessing a classification model's performance. It offers a thorough analysis of the discrepancies between the actual labels for each class and the model's predictions. By using this, we can observe which classes the model predicts well and which ones give it trouble. The number of instances that are expected to belong to a certain class in relation to the ground truth labels is represented by each cell in the matrix. We examined confusion matrices on the validation and test datasets to gain further insight into the model's classification performance.

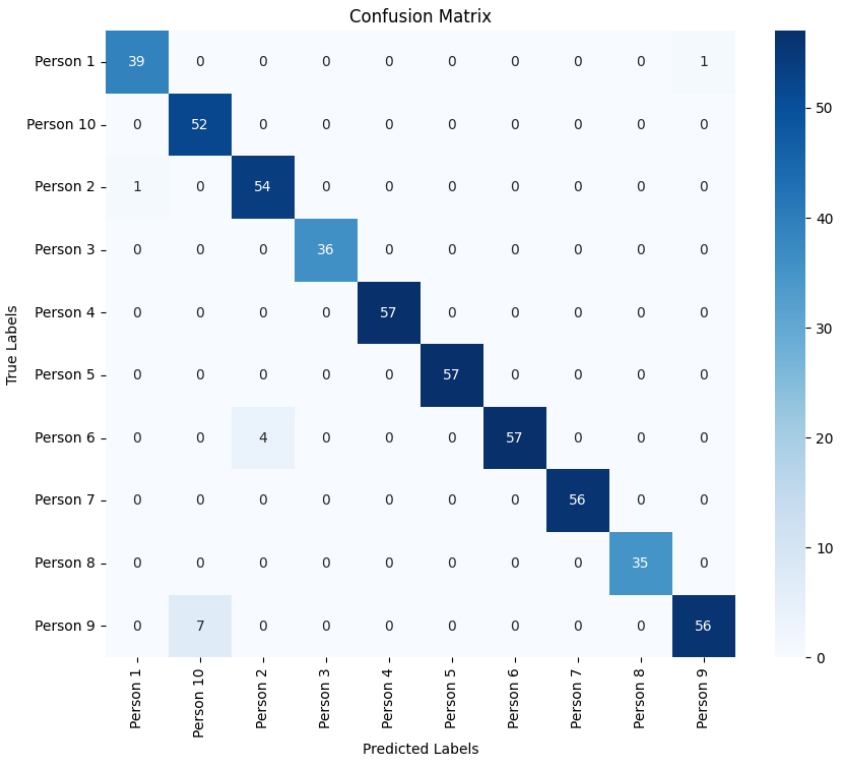


Figure 20-Confusion Matrix on Validation Dataset

Out of 512 photos from validation dataset, 499 were found to be positive overall, while 13 were found to be false positives. In the same way, the test dataset's confusion matrix provided an ultimate evaluation of the model's classification abilities. We obtained an in-depth knowledge of the model's generalization abilities on data that had not been seen before by looking at the distribution of predictions among various classes. This gave us

confidence in the model's potential to function well in real-world circumstances and enabled us to confirm its performance beyond the training and validation phases. The confusion matrices depicted in Figure offer a thorough analysis of the model's classification performance using the test dataset.

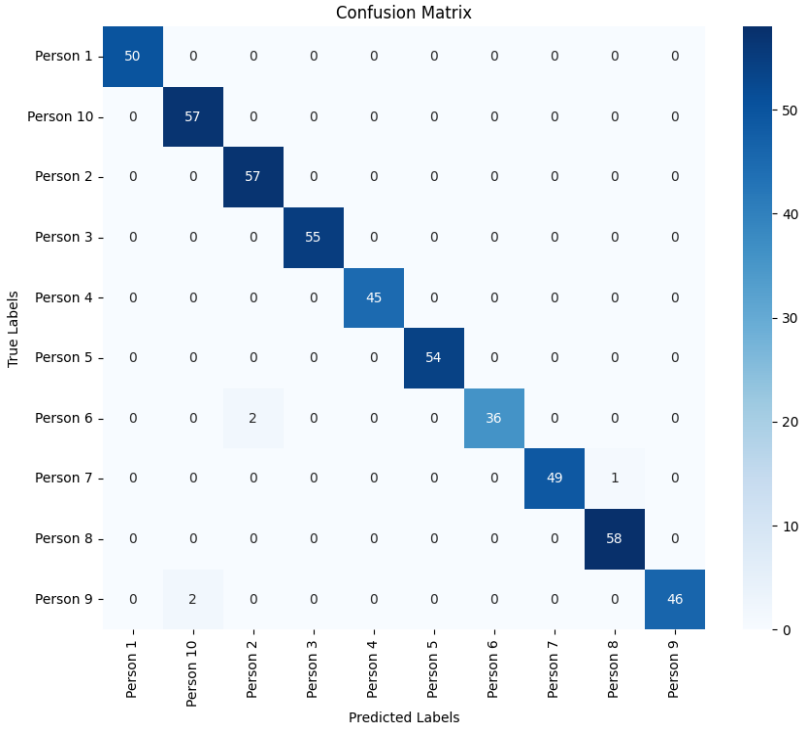


Figure 21-Confusion Matrix for the Testing Dataset

Our algorithm accurately identified 507 out of the 512 photographs we examined in the validation dataset as positive.

11.2.4 PREDICTIONS FROM TEST DATASET

We employed our trained model to make predictions on randomly chosen images from our testing dataset in order to assess the accuracy of our model even further. We showed the predicted class next to the actual class for every image, allowing users to compare the predictions of the model directly with the labels that correspond to the actual labels. We also provided confidence scores for every prediction, which represent the degree of assurance the model has for its classification. The following figure illustrates the predictions made by our model on a random selection of images taken from the testing dataset. The predicted classes are shown alongside the actual classes.

Figure 22-Predictions of Random Images Selected

Out of the 9 randomly selected images from the testing dataset, our model correctly predicted the classes of 8 images. The majority of photos were successfully classified, demonstrating the model's capacity to reliably distinguish between distinct classes and extend effectively to previously unknown data.

11.2.5 PREDICTION WITH UNSEEN DATA

We wanted to see how well our model could recognize signatures from people it's already familiar with versus those it's never seen before. So, we picked one signature from someone whose signature was part of the training data, which our model is used to seeing. Then, we chose another signature from someone not in the training data, making it new for the model. The goal was to test if our model could accurately predict both familiar and unfamiliar signatures. By doing this, we could understand how well our model adapts to different kinds of signatures and if it can handle new ones it hasn't encountered during training. We began our evaluation by testing the model with a new signature provided by Person 2 and assessed its prediction accuracy and confidence level. Remarkably, the model correctly predicted the image, providing a high confidence score of 95.54%. Figure 1 illustrates the test image. Figure showcases the model's prediction and associated confidence score.

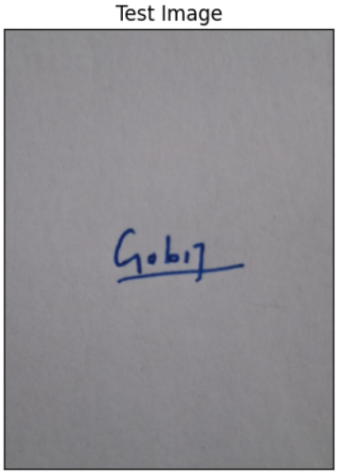


Figure 23-Test Image 01

```
1/1 [==============================] - 0s 55ms/step
Confidence: 0.9531952
Confidence Percentage: 95.32%
Predicted Class: Person2
```

Figure 24-Predicton of Test Image

Subsequently, we tested the model with a signature from an individual not present in our training dataset. Figure shows the signature provided for testing, which does not correspond to any of the ten classes represented in our training dataset.

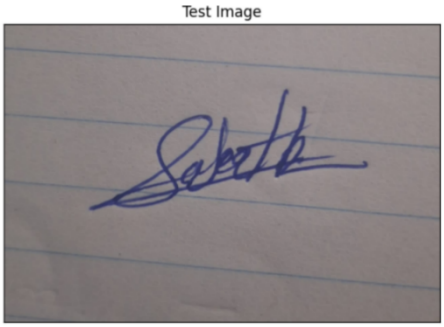


Figure 25-Test Image 02

```
1/1 [==============================] - 0s 104ms/step
Confidence: 0.38176778
Confidence Percentage: 38.18%
Predicted Class: Person1
```

Figure 26-Prediction of Test Images

Surprisingly, the model's prediction, shown in Figure 2, incorrectly classified the signature as belonging to Person 1, resulting in a confidence score of 38.18%. Although the confidence level was comparatively lower, the model's ability to correctly identify the class suggests a reasonable level of prediction accuracy. This outcome supports our conclusion that the model exhibits satisfactory performance in recognizing signatures, even when presented with data from individuals not encountered during training. The image may contain visual elements or patterns that are similar to those found in training courses. Although the image isn't related to any of the trained classes, the model may recognize some recognizable patterns within it [6]. Considering the prediction's confidence level is critical, especially when dealing with scenarios when the model assigns a class to an input that may not correspond to any of the previously trained classes. When the confidence score for a prediction is extremely low, it shows that the model is unsure about its classification. In such circumstances, it is acceptable to assume that the input does not belong to any of the trained classes [7].

After carefully considering the model's predictions and confidence scores, we decided to use a 90% confidence level to verify a student's signature. This threshold acts as a cutoff, ensuring that only predictions with high confidence levels are accepted as authenticated signatures.

11.2.6 EVALUATION OF EXTRACTION

The extraction model presented in the provided Python script offers a systematic approach to automating the process of extracting signatures from images. The model's effectiveness is evident in its ability to accurately identify signature regions within images and segment them into individual parts for further analysis. By leveraging advanced image processing techniques such as Gaussian blur, Canny edge detection, and contour analysis, the model achieves robustness in detecting and delineating signature contours, even in the presence of noise or variations in image quality. Additionally, the model's ability to dynamically adapt to different image sizes and resolutions enhances its versatility and applicability across various scenarios. However, the model's performance may be influenced by factors such as the complexity of signature patterns, variations in handwriting styles, and the presence of overlapping elements in the image. Further evaluation of the model's accuracy and robustness across diverse datasets and real-world scenarios would be essential to assess its suitability for practical applications. Additionally, optimization of parameters and fine-tuning of algorithms could potentially enhance the model's performance and efficiency. Overall, the extraction model demonstrates promise as a valuable tool for automating signature extraction tasks, with scope for refinement and improvement through continued research and development efforts.

FINAL RESULT CHECKING WITH BOTH EXTRACTION AND PREDICTION WITH MOBILE APP.

As the project came to an end, we carried out a thorough final assessment to measure the overall effectiveness of our finished application. Our main goal was to test the efficiency and reliability of the whole workflow, through uploading a new attendance sheet photo to getting the results via our mobile app. To start the evaluation process, we entered a new attendance sheet picture using the mobile app interface. This photo was then sent seamlessly to the Flask API, which acted as the backend for our signature extraction model. At this point, the submitted picture was automatically cropped, with signatures recognized and segregated as distinct entities. This phase replicated real-world circumstances in which users would submit updated attendance sheets for processing. The retrieved signatures were then sent through our trained prediction model. Each signature was assessed and classified separately, and the prediction model made real-time predictions about the signatures identified.

Finally, the prediction model's outputs were smoothly sent back to the mobile app interface, where they were presented to the user in an understandable and intuitive manner. Users may examine the recognized signatures, as well as relevant data such as the signatory's name and the confidence of the prediction, on the mobile app's screen. This user-friendly interface made the outputs simply accessible and understandable, allowing for quick decisions and actions.

Figure below illustrates the final outcome displayed on the mobile app screen following the evaluation process. As depicted, the app presents the identified signatures along with the corresponding signatories' names and the prediction accuracy.

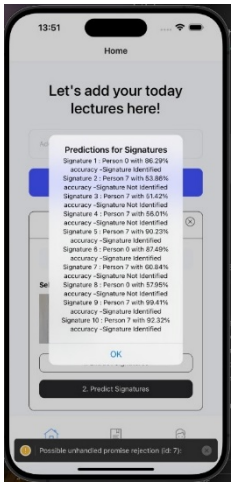


Figure 27- Final Output

12 CHAPTER 6: FUTURE WORK

- As a future work, we may also aim at boosting our system's accuracy by experimenting with new and improved parameter coefficients. These coefficients will help increase the gap between real and forged signatures, making our system more reliable.
- We are expecting to make our signature verification system work for more than 100 students.
- Instead of collecting numerous signatures, we intend to acquire a modest set of 5-10 authentic signatures from each individual. By using data augmentation techniques, we will then expand this limited dataset to generate additional training samples [8].
- Expecting to fix a problem where our existing signature extraction technique isn't correctly clipping signatures from attendance sheets. We intend to use the YOLOv8 algorithm to create a new real-time object detection model in order to get beyond this issue [9].
- Whenever a new student joins the classroom, training the model again from scratch can be time-consuming and challenging. We're thinking of introducing a new function to our mobile app to address this by allowing users (academic staff) to upload sample signature images of new student and automatically train the model using the mobile app.
- Also planning to include an existing feature for Bluetooth based attendance tracking. This feature will serve as an additional layer of security and verification by enabling double-checking of the student's presence [10]. Our application will be able to identify and verify the presence of students in the classroom by accessing their mobile Bluetooth address.

12.1 GAPS OF THE PROJECT

The future work outlined for the “D-Tect” Signature Verification Model presents a promising trajectory for enhancing the system's accuracy, scalability, and usability. Experimentation with new parameter coefficients signifies a proactive approach towards improving the model's reliability and effectiveness in distinguishing between genuine and forged signatures. The aspiration to extend the system's capacity to handle verification for over 100 students reflects a commitment to scalability and broader applicability within educational settings. Moreover, the adoption of data augmentation techniques to expand the training dataset demonstrates a strategic utilization of resources to enhance the model's performance without excessive data collection efforts. Addressing existing challenges, such as inaccuracies in signature extraction, through the implementation of advanced algorithms like YOLOv8 showcases a dedication to overcoming technical hurdles and ensuring robustness in the system's functionality. Additionally, proposed features such as mobile-based model training and Bluetooth-enabled attendance tracking underscore an innovative approach to streamlining operations and enhancing user experience. By integrating these future developments, the “D-Tect” Signature Verification Model aims to evolve into a comprehensive and reliable solution for signature authentication within university environments.

However, despite the promising advancements outlined in the future work, certain gaps in the project's implementation and functionality remain to be addressed. For instance, the reliance on a modest set of authentic signatures for data augmentation may introduce limitations in capturing the diversity of signature styles and variations encountered in real-world scenarios. Additionally, the potential challenge of ensuring the accuracy and efficiency of the model training process when incorporating new student signatures underscores the need for robust and user-friendly mechanisms for updating and maintaining the model. Furthermore, while the inclusion of Bluetooth-based attendance tracking adds an additional layer of security, potential privacy concerns and technical complexities associated with Bluetooth address verification may require careful consideration and mitigation strategies. Overall, addressing these gaps will be crucial for ensuring the effectiveness, reliability, and ethical integrity of the “D-Tect” Signature Verification Model as it continues to evolve and expand its capabilities.

12.2 PROPOSAL FOR ENHANCEMENT OR RE-DESIGN

- In alignment with the future works outlined for the “D-Tect” Signature Verification Model, several proposals for enhancement or redesign can be considered to further augment the capabilities and effectiveness of the system.

- Firstly, to bolster the system's accuracy and reliability, an emphasis can be placed on experimenting with new and improved parameter coefficients. These adjustments aim to increase the distinction between genuine and forged signatures, thereby enhancing the system's overall performance and robustness.

- Secondly, in anticipation of scaling up the signature verification system to accommodate more than 100 students, efforts can be directed towards optimizing system architecture and resource allocation. This may involve refining algorithms and implementing parallel processing techniques to ensure seamless performance even with increased workload.

- Thirdly, to streamline the data acquisition process and mitigate the need for extensive signature collection, a shift towards acquiring a modest set of 5-10 authentic signatures from each individual can be pursued. Leveraging data augmentation techniques, such as generative adversarial networks (GANs) or variational autoencoders (VAEs), can then help expand the training dataset, thereby enriching the model's learning capabilities without compromising on authenticity.

- Fourthly, addressing existing challenges with signature extraction by adopting advanced algorithms like YOLOv8 for real-time object detection presents an opportunity to overcome technical limitations and improve accuracy. By leveraging state-of-the-art object detection techniques, the system can achieve more precise and efficient signature localization within attendance sheets.

- Furthermore, to mitigate the time-consuming process of retraining the model from scratch whenever a new student joins the classroom, the introduction of a feature within the mobile app for automatic model training using uploaded sample signature images offers a practical solution. This functionality empowers academic staff to efficiently integrate new student signatures into the system, ensuring continuous adaptation and improvement without significant manual intervention.

13 CHAPTER 7: CONCLUSIONS

In conclusion, the "D-Tect" Signature Verification Mobile App project has successfully demonstrated the potential of utilizing advanced machine learning techniques for automating the authentication of handwritten signatures within university environments. Through meticulous data collection, preprocessing, and model development, we have achieved a robust framework capable of accurately verifying signatures with high efficiency. The project's outcomes include the creation of a comprehensive dataset, the development of a deep neural network-based model, and the implementation of an extraction model for automating signature extraction from documents. Furthermore, the evaluation results showcase the effectiveness and reliability of the developed system, with high accuracy rates achieved in signature verification tasks. Moving forward, there are several avenues for future work, including system scalability, optimization of model parameters, and integration of additional features for enhanced functionality.

13.1 THE IMPORTANCE OF THE RESULT

The results obtained from the "D-Tect" Signature Verification Mobile App project underscore the significance of leveraging machine learning technologies to address challenges related to signature authentication in university environments. By automating the verification process, the project not only enhances efficiency but also strengthens the integrity of university-related activities. The successful implementation of the system highlights the potential for technology-driven solutions to streamline administrative tasks and improve overall operational effectiveness within educational institutions.

13.2 VALIDITY OF THE RESULT

The validity of the results obtained from the project is supported by rigorous data collection, preprocessing, and model evaluation procedures. The utilization of a diverse dataset, adherence to ethical guidelines, and meticulous evaluation of model performance contribute to the credibility and reliability of the findings. Furthermore, the extensive evaluation conducted on both validation and test datasets provides comprehensive insights into the effectiveness and generalizability of the developed system, ensuring the validity of the results.

13.3 GAPS AND LIMITATIONS OF THE FINDINGS

Despite the success of the project, there are certain gaps and limitations that warrant consideration. One limitation is the reliance on a relatively small dataset, which may affect the model's ability to generalize to a wider range of signature styles and variations. Additionally, the extraction model's performance may be influenced by factors such as image quality and the complexity of signature patterns. Furthermore, the project primarily focuses on signature verification within university environments, limiting its applicability to other domains. Addressing these gaps and limitations through further research and development efforts will be essential to enhance the robustness and applicability of the developed system.

14 REFERENCES

[1] J. Benois-Pineau, R. Bourqui, D. Petkovic, and G. Quenot, *Explainable Deep Learning AI: Methods and Challenges*. Elsevier, 2023.

[2] M. Krichen, "Convolutional Neural Networks: A Survey," *Computers,* vol. 12, no. 8, p. 151, 2023. [Online]. Available: https://www.mdpi.com/2073-431X/12/8/151.

[3] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning pooling for convolutional neural network," *Neurocomputing,* vol. 224, pp. 96-104, 2017.

[4] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, "Fundamental concepts of convolutional neural network," *Recent trends and advances in artificial intelligence and Internet of Things,* pp. 519-567, 2020.

[5] J. F. Canny, "Finding edges and lines in images," 1983.

[6] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," *Advances in neural information processing systems,* vol. 31, 2018.

[7] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136,* 2016.

[8] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*, 2018: IEEE, pp. 117-122.

[9] B. Liang, J. Li, and J. Huang, "We can always catch you: Detecting adversarial patched objects with or without signature," *arXiv preprint arXiv:2106.05261,* 2021.

[10] A. Puckdeevongs, N. Tripathi, A. Witayangkurn, and P. Saengudomlert, "Classroom attendance systems based on bluetooth low energy indoor positioning technology for smart campus," *Information,* vol. 11, no. 6, p. 329, 2020.

15 GLOSSARY

- CNN : Convolutional Neural Networks
  A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data.
- Contour detection and Canny edge detection
  Essential techniques in computer vision that are commonly used for object detection, shape analysis, and segmentation of images.
- API : Application Programming Interface
  APIs are functions that developers can call on to access specific features by calling upon programs, code, and services that others have written.
- Gantt chart: A visual representation of a project schedule, showing the start and finish dates of elements.
- Confusion matrix: An effective method for assessing a classification model's performance. It offers a thorough analysis of the discrepancies between the actual labels for each class and the model's predictions