ក្រសួងអប់រំ យុវជន និងកីឡា
វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យព័ត៌មានវិទ្យា និង ទំនាក់ទំនង

គម្រោងសញ្ញាបត្រវិស្វកម

ប្រធានបទ : កម្មវិធីសម្គាល់អក្សរខ្មែរសរសេរនៅលើខ្យល់

និស្សិត : យន វ៉ាន់ដា

ឯកទេស : វិស្វកម្មព័ត៌មានវិទ្យា និង ទំនាក់ទំនង

គ្រូទទួលបន្ទុក : បណ្ឌិត វ៉ាលី ដូណា

ឆ្នាំសិក្សា : ២០២២-២០២៣

MINISTERE DE L'EDUCATION, DE LA JEUNESSE ET DES SPORTS
INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTEMENT DE GENIE INFORMATIQUE ET COMMUNICATION

MEMOIRE DE FIN D'ETUDES

Titre : Khmer Air Writing Recognition (Prototype)

Étudiant : M. YORN Vanda

Spécialité : Génie Informatique et Communication

Tuteur de stage : Dr. VALY Dona

Année scolaire : 2022-2023

ក្រសួងអប់រំ យុវជន និងកីឡា
វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យព័ត៌មានវិទ្យា និង ទំនាក់ទំនង

គម្រោងសញ្ញាបត្រវិស្វករ

របស់និស្សិត: ឃន វ៉ាន់ដា

កាលបរិច្ឆេទការពារនិក្ខេបបទ: ថ្ងៃទី ១០ ខែ កក្កដា ឆ្នាំ ២០២៣

អនុញ្ញាតឲ្យការពារគម្រោង

នាយកវិទ្យាស្ថាន: _____

ថ្ងៃទី ខែ ឆ្នាំ ២០២៣

ប្រធានបទ : កម្មវិធីសម្គាល់អក្សរខ្មែរសរសេរនៅលើខ្យល់

សហគ្រាស : វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ប្រធានដេប៉ាតឺម៉ង់ : លោក ង្វៃ ហោង _____

សាស្ត្រាចារ្យដឹកនាំគម្រោង : បណ្ឌិត វ៉ាលី ដូណា _____

អ្នកទទួលខុសត្រូវក្នុងសហគ្រាស : បណ្ឌិត វ៉ាលី ដូណា _____

រាជធានីភ្នំពេញ ឆ្នាំ២០២៣

MINISTERE DE L'EDUCATION,
DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTEMENT DE GENIE INFORMATIQUE ET COMMUNICATION

MEMOIRE DE FIN D'ETUDES

DE M. YORN Vanda

Date de soutenance : le 10 juillet 2023

« Autorise la soutenance du mémoire »

Directeur de l'Institut :_____

Phnom Penh, le juillet 2023

Titre : Khmer Air Writing Recognition (Prototype)

Établissement du stage : Institut de Technologie du Cambodge

Chef du département : M. LAY Heng _____

Tuteur de stage : Dr. VALY Dona _____

Responsable de l'établissement : Dr. VALY Dona _____

PHNOM PENH, 2023

ACKNOWLEDGMENTS

As a fifth-year student studying Information and Communication Engineering at the Institute of Technology of Cambodia, we are required to complete an internship as part of our studies. The internship is crucial as it helps us become familiar with the real work environment and enables us to acquire fresh knowledge and experiences. Moreover, it provides us with an opportunity to apply the knowledge we have gained over the past five years in a practical manner. During this internship, I would like to express my gratitude to a few remarkable individuals who have greatly contributed to my success, whether through direct or indirect means.

Firstly, I want to show my gratitude to **His Excellency Dr. PO Kimtho**, the director of the **Institute of Technology of Cambodia**. He has dedicated his time and effort to building strong collaborations both locally and internationally.

Secondly, I am extremely thankful to **Mr. LAY Heng**, the head of the Department of **Information and Communication Engineering** (DICE). He has shown great leadership and wise management within the department.

Thirdly, I would like to express my thanks to my supervisor, **Dr. VALY Dona**, head of **Mechatronics and Information Technology** (MIT) Research Units and a lecturer at the **Institute of Technology of Cambodia**. Thank you for granting me a chance to contribute to this project and for providing me with numerous insights and guidance throughout my internship. I genuinely value the time I spent collaborating with him, both in a professional and personal capacity. He has invested his time and effort into guiding me. He has provided valuable suggestions and advice to help me improve my research work and report.

Last but not least, I want to express my gratitude to the younger students who are part of the research and technology transfer teams. They have worked hard and diligently to improve this project. I have gained valuable knowledge from them. Lastly, I am extremely thankful to my family for their support and encouragement.

RÉSUMÉ

Khmer Air Writing est une technologie prometteuse en vision par ordinateur qui permet aux utilisateurs de dessiner ou d'écrire des caractères dans les airs à l'aide de leur doigt. Cette technologie utilise des algorithmes d'apprentissage automatique pour reconnaître les caractères. Cependant, la reconnaissance des caractères khmers, qui ont des schémas d'écriture complexes, pose un défi pour la reconnaissance optique des caractères.

Dans cette étude, nous proposons une approche pour reconnaître les caractères khmers en utilisant la reconnaissance de l'écriture manuscrite aérienne. Notre méthode consiste à utiliser un réseau de neurones artificiels (ANN) pour apprendre les modèles de mouvements d'écriture manuscrite. Nous analysons les données de position obtenues à partir du suivi de la main pour obtenir des coups de doigt, qui sont ensuite utilisés par l'ANN pour reconnaître les caractères.

Pour former et tester notre méthode proposée, nous avons collecté un ensemble de données de 18 150 lettres khmères manuscrites. L'ensemble de données a été utilisé pour former l'ANN et évaluer la précision de la reconnaissance de notre approche. Les résultats ont montré que notre approche a atteint une précision de reconnaissance de 94,78 % sur l'ensemble de données de caractères khmers manuscrits.

Ces résultats mettent en évidence le potentiel de la reconnaissance de l'écriture manuscrite aérienne pour améliorer l'accessibilité aux appareils numériques, en particulier pour les utilisateurs qui rencontrent des difficultés avec les méthodes de saisie traditionnelles. En permettant aux utilisateurs d'écrire ou de dessiner des caractères dans l'air, cette technologie offre une méthode de saisie alternative et plus intuitive, élargissant les possibilités d'interaction avec les appareils numériques.

ABSTRACT

Khmer Air Writing is a promising technology in computer vision that allows users to draw or write characters in the air using their finger. This technology utilizes machine learning algorithms to recognize the characters. However, recognizing Khmer characters, which have complex writing patterns, poses a challenge for optical character recognition.

In this study, we propose an approach to recognize Khmer characters using Air Handwriting Recognition. Our method involves employing an Artificial Neural Network (ANN) to learn the patterns of handwriting movements. We analyze positional data obtained from hand tracking to obtain finger strokes, which are then used by the ANN to recognize the characters.

To train and test our proposed method, we collected a dataset of 18,150 hand-written Khmer letters. The dataset was used to train the ANN and evaluate the recognition accuracy of our approach. The results showed that our approach achieved a recognition accuracy of 94.78% on the dataset of handwritten Khmer characters.

These results highlight the potential of air handwriting recognition to enhance accessibility to digital devices, particularly for users who face difficulties with traditional input methods. By enabling users to write or draw characters in the air, this technology provides an alternative and more intuitive input method, expanding the possibilities for interacting with digital devices.

Keywords: Khmer Air Handwriting, Computer Vision, Machine Learning, Artificial Neural Network (ANN)

សេចក្តីសង្ខេប

កម្មវិធីសម្គាល់អក្សរខ្មែរសរសេរនៅលើខ្យល់ គឺជាបច្ចេកវិទ្យាថ្មីមួយក្នុង Computer Vision ដែលអាចឱ្យអ្នកប្រើប្រាស់គូរ ឬសរសេរតួអក្សរខ្មែរនៅលើខ្យល់ ដោយប្រើប្រាស់ម្រាមដៃរបស់ពួកគេ។ បច្ចេកវិទ្យាថ្មីនេះប្រើវិធីសាស្ត្រក្នុងការ Train Machine Learning Model ហើយយើងបានជ្រើសរើសនូវវិធីសាស្ត្រមួយក្នុងចំណោមវិធីសាស្ត្រជាច្រើនទៀតនៃ Machine Learning ដែលជាវិធីសាស្ត្រដើម្បីសិក្សាពីគំរូនៃចលនាសរសេរដោយដៃនោះគឺ Artificial Neural Network (ANN)។ ហើយយើងបានប្រមូលទិន្នន័យពីការសរសេរអក្សរខ្មែរដោយដៃដែលចាប់យកទិន្នន័យទីតាំងដែលទទួលបានពីការគូរធៀបទៅនឹង Screen Computer ហើយវាត្រូវបានប្រើប្រាស់ដើម្បីឱ្យ ANN Model ធ្វើការសិក្សារពីទិន្នន័យទាំងនោះ ដើម្បីកំណត់សម្គាល់តួអក្សរខ្មែរ។

សំណុំទិន្នន័យនៃអក្សរខ្មែរដែលត្រូបានយកសិក្សាមានចំនួន 18,150 តួអក្សរដែលចែកចេញជា 33 ក្រុមតំណាងឱ្យតួអក្សរខ្មែរហើយសសំណុំទិន្នន័យដែលត្រូវបានប្រើប្រាស់ដើម្បី Train ជាមួយនឹង ANN Model ត្រូវបានវាយតម្លៃរកភាពត្រឹមត្រូវនៃការស្គាល់តួអក្សរនៃវិធីសាស្ត្ររបស់យើង។ជាលទ្ធផលបានបង្ហាញថា វិធីសាស្ត្រនេះសម្រេចបាននូវភាពត្រឹមត្រូវគិតជាភាគរយគឺ 94.78% ទៅលើសំណុំទិន្នន័យនៃអក្សរខ្មែរទាំងអស់ ។

ដោយលទ្ធផលនេះបង្ហាញឱ្យឃើញពីសក្តានុពល នូវភាពត្រឹមត្រូវនៃការសរសេរដោយដៃលើខ្យល់ ដើម្បីបង្កើនភាពងាយស្រួលនិងទំនើបកម្មទៅដល់ឧបករណ៍ឌីជីថល ជាពិសេសសម្រាប់អ្នកប្រើប្រាស់ដែលប្រឈមនឹងការលំបាកជាមួយវិធីសាស្ត្របញ្ចូលបែបបុរាណ។ តាមរយៈការសិក្សារនេះវាបានអនុញ្ញាតឲ្យអ្នកប្រើប្រាស់សរសេរឬគូរតួអក្សរនៅលើខ្យល់ ដោយឡែកបច្ចេកវិទ្យានេះក៏បានផ្តល់នូវវិធីសាស្ត្រនិងពង្រីកលទ្ធភាពសម្រាប់អន្តរកម្មជាមួយឧបករណ៍ឌីជីថល។

ABBREVIATION LIST

ML: Machine Learning

NN: Neural Network

DL: Deep Learning

ANN: Artificial Neural Network

AI: Artificial Intelligent

NP: Numpy

PD: Pandas

OS: Operating System

VSCode: Visual Studio Code

MP: MediaPipe

CV : Computer Vision

ITC: Institute of Technology of Cambodia

TABLE OF CONTENTS

FIGURE LIST

CHAPTER I PRESENTATION OF ORGANIZATION

1.1. Company Presentation

ViLa Lab, short for "Vision and Language Lab," is a renowned research laboratory located at the Institute of Technology of Cambodia (ITC). It primarily focuses on the fields of Computer Vision and Natural Language Processing (NLP). The lab provides a conducive environment for engineering and master's students to conduct research in these areas.

Under the guidance of experienced researchers and faculty members, the students at ViLa Lab engage in a wide range of research endeavors. These projects cover diverse aspects of computer vision and NLP, such as Khmer Air-Writing, object detection, Khmer Word Spotting, Khmer plagiarism and many others.

Figure 1: ViLa Lab Logo

1.1.1. Address and Contact

ViLa Lab is located at the Institute of Technology of Cambodia (ITC) in Phnom Penh, Cambodia.

Address: Russian Conf. Blvd. Phnom Penh, Cambodia.
Tel. KH: (+855) 23 880 370 / 982 404
PO Box: 86
Faxes: (+855) 23 880 369

Email: info@itc.edu.kh

Website: https://www.itc.edu.kh

Figure 2: ViLa Lab Address

1.1.2. Activities and Opportunities

As an esteemed research facility, ViLa Lab offers a variety of resources and support to students, enabling them to delve into cutting-edge research topics and explore innovative ideas. The lab is equipped with state-of-the-art hardware and software tools, allowing students to work on sophisticated projects in computer vision and NLP. Moreover, students have the valuable opportunity to pursue their master's or PhD degree through international scholarships.

1.2. Objective of Internship

During my fifth year at the Institute of Technology of Cambodia, as a student in the Department of Information and Communication Engineering, I completed a mandatory internship. This internship aimed to provide real-world experience and integrate theoretical knowledge into practical applications. It also emphasized the development of soft skills such as communication, time management, problem-solving, critical thinking, and collaboration. I secured an internship during the second semester of my fifth year and was assigned a project by a company's project manager. The objective was to study the existing system, propose innovative solutions, and apply the research outcomes. At the end of the internship, I wrote a thesis and defended my project, highlighting the findings and their practical implementation.

CHAPTER II PROJECT PRESENTATION

2.1. Overview

In recent years, technological advancements have revolutionized the way we interact with computers and digital devices, offering more intuitive and seamless experiences. One remarkable project in this realm is the Khmer Air Writing Recognition system, which combines machine learning and computer vision techniques to enable users to write Khmer characters in the air and have them recognized by the system.

The Khmer Air Writing Recognition project leverages the power of machine learning to train a dataset using an Artificial Neural Network (ANN) model. The model's optimizer, Adam, and Loss function, Cross-Entropy-Loss, contribute to the accuracy and efficiency of the recognition process. Through the training process, the model learns to recognize and interpret the handwritten Khmer characters based on the provided dataset.

In addition to the machine learning aspect, the project incorporates computer vision technology to facilitate the recognition process. MediaPipe, a computer vision library, is employed to track the user's hand and capture its position. By leveraging MediaPipe, the system precisely detects the hand movements and extracts the finger positions when the user draws in the air.

The project takes advantage of the coordinates obtained through computer vision to integrate with the machine learning model. The position data captured by MediaPipe is fed into the model, allowing it to associate the hand movements with specific Khmer characters. This integration between computer vision and machine learning enables real-time recognition of the handwritten Khmer characters written in the air.

The Khmer Air Writing Recognition project holds great promise in enhancing human-computer interaction and promoting the use of the Khmer language. By enabling users to write Khmer characters in the air, without physical contact or specialized input devices, the project offers a natural and intuitive means of interaction with computers. This innovation has the potential to facilitate communication, language education, and the preservation of the rich Khmer culture.

While challenges exist in fine-tuning the machine learning model and optimizing the computer vision algorithms, ongoing research and development efforts continue to improve the system's accuracy and robustness. As the project advances, we can anticipate its integration into

various applications and devices, ultimately transforming the way we write and interact with Khmer script in the digital realm.

In this era of technological innovation, the Khmer Air Writing Recognition project stands as a remarkable achievement, blending machine learning and computer vision to enable seamless and culturally significant human-computer interaction. By harnessing the power of these technologies, this project paves the way for intuitive and inclusive means of writing and communicating in the Khmer language.

2.3. Problem

Before the introduction of air writing technology, people faced challenges with limited accessibility, mobility constraints, writing speed, language barriers, and difficulties in preserving and sharing written information. Air writing emerged as an innovative solution, providing a more accessible, portable, and efficient way to capture information.

By enabling users to write in the air using gestures or motion-based input, it overcomes the limitations of traditional writing methods. Air writing offers enhanced accessibility, mobility, speed, and language flexibility. It also enables digital storage and sharing capabilities, making it convenient for preserving and distributing information.

Overall, air writing technology revolutionizes the way we capture and interact with written content, making it more convenient, efficient, and adaptable to diverse user needs.

2.4. Objective

The objective of this project is to build a tool that allows users to track their hand movement in front of a webcam and integrate it with an artificial neural network model to recognize and predict written characters. The tool is able to detect user's hand:

- **Detect and Tracking hand position:** To make a tool which can detect and get the information form users' hand and drawing landmark to the detected hand.
- **Drawing:** To make the possibility of drawing without physical touch by just following the condition of the tool.
- **Predict:** To make the prediction tool which could recognize Khmer character base on Drawn Character compare to the existed dataset and ML model
- **Clear Board:** Be able to clear the screen just like clearing on board in real whiteboard.

2.5. Planning

During the internship, it is crucial to identify the main objective that needs to be achieved. This objective serves as the guiding principle for the planning process. Once the objective is defined, it is essential to identify the overall project requirements and understand the scope of work involved. This information lays the foundation for developing a comprehensive and effective plan that outlines the tasks, milestones, and timelines. By conducting thorough planning after clarifying the internship's main objective and project needs, the internship period can be utilized efficiently and effectively.

Description of activities	March				April				May				June			
	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4
Learing Machine Learning and Pytorch																
Build Computer Vision Prototype																
Analyze function and Requirement to Prototype																
ANN Model and Computer Vision Integration																
Testing and Implemetation																

Table 1: Project Plan

CHAPTER III PROJECT ANALYZE AND DESIGN

3.1. System Design

3.1.1. Use Case Diagram

A use case diagram is a visual representation that depicts the interactions between users and a system, providing a clear understanding of its functionalities and their relationships. In the context of your air-writing project, a use case diagram would showcase the key actions and interactions involved in the system. It would demonstrate the process of tracking hand-drawn characters, including the recognition of gestures and movements. Additionally, the diagram would highlight functionalities such as clearing the screen to start afresh, predicting characters based on the drawn hand gestures, and the overall functioning of the system as a closed entity.

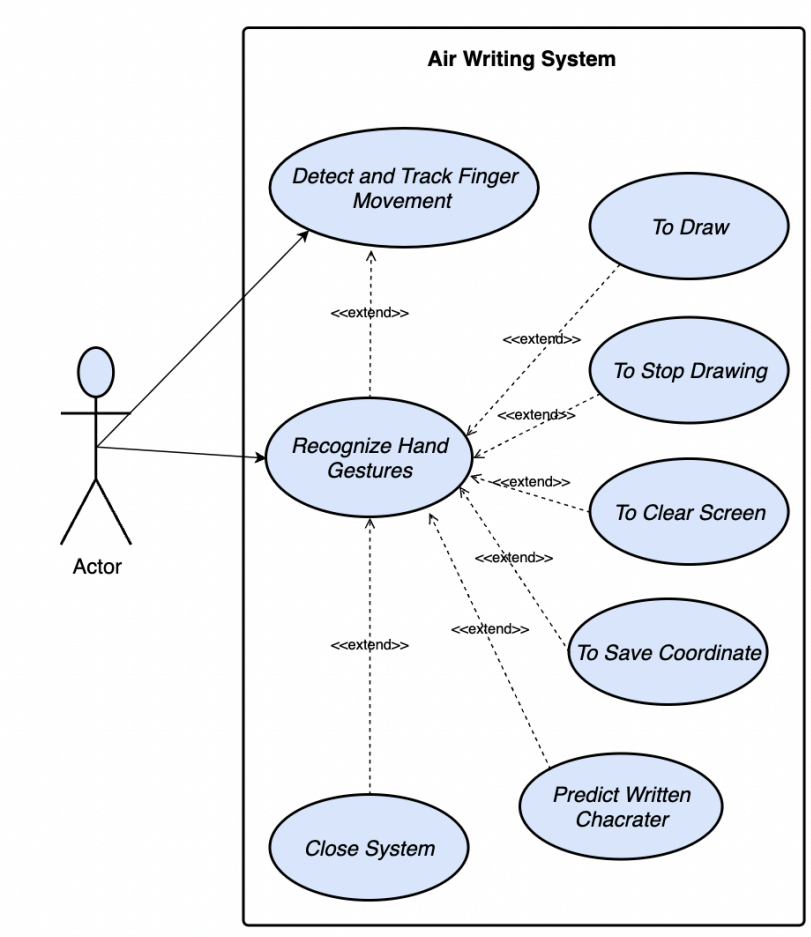


Figure 3: Use Case Diagram

Here's a description of the main elements that could be included in the use case diagram for your air-writing project:

- **Actor**:
  - User: The person interacting with the system by performing hand gestures and drawing characters in the air through webcam.

- **Use Cases:**
  - Track Hand Draw: This use case represents the system's ability to track the hand movements of the user as they draw characters in the air.
  - Clear Screen: This use case allows the user to clear the screen or reset the system, removing any previously drawn characters.
  - Predict Character: This use case involves the system's capability to predict the character being drawn by the user based on their hand movements and gestures.

3.1.2. Activity diagram

- **Hand Tracking Diagram**

The system starts by initializing the 'HandTracker' class with the provided parameters.

- It opens the webcam to capture the input video stream.
- The system searches for hands in the captured video frames.
- It retrieves the positions of the detected hand landmarks.
- The system draws landmarks and connections on the video frames based on the detected hand landmarks.
- It determines the state of each finger (whether it is up or down) based on the hand landmarks' positions.
- The system assigns each finger's position to the landmark list and stores the results.
- It returns the finger positions or the landmark list containing the finger states.
- The activity diagram represents the sequential flow of activities in the 'andTracker' class, from initializing the class to returning the finger positions.

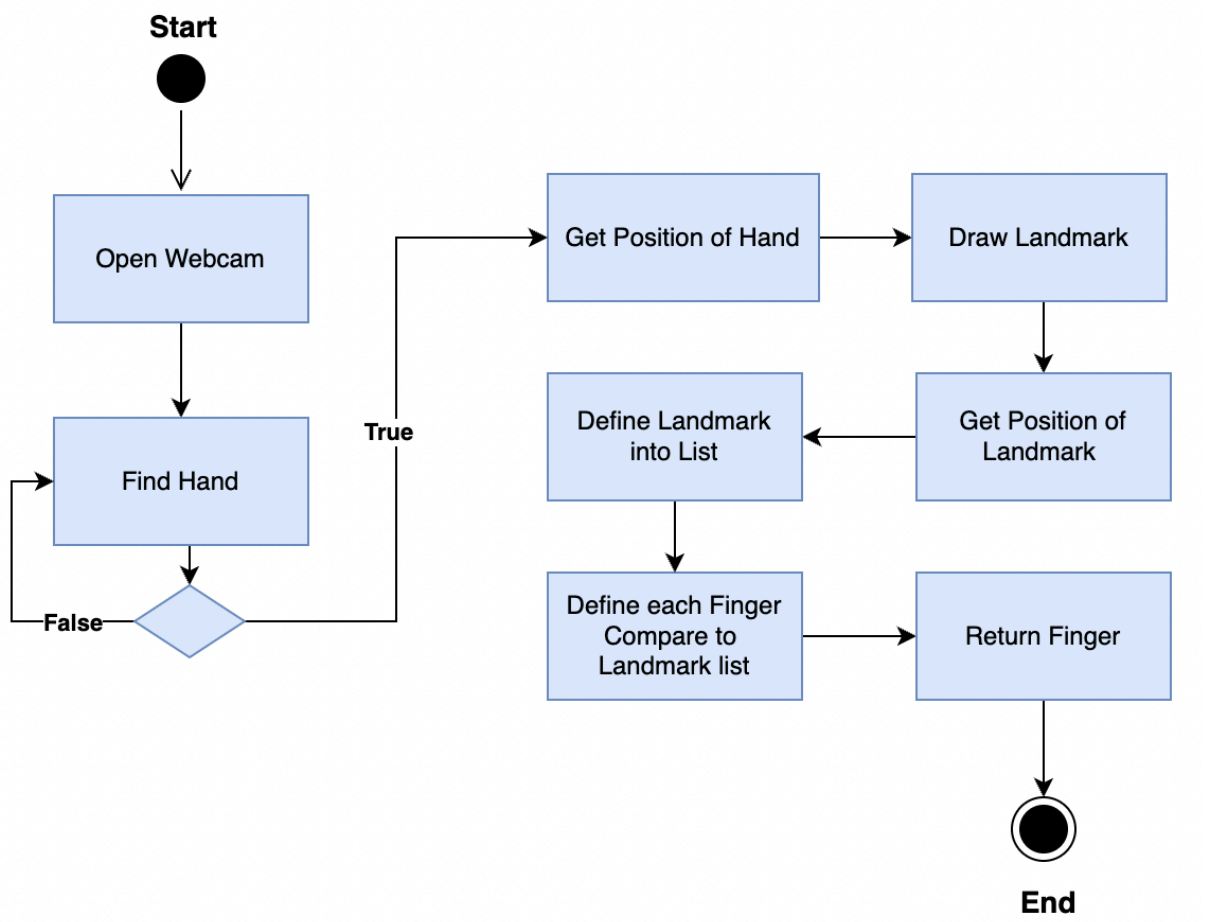


Figure 4: Hand Tracking Activity Diagram

- **Draw and Predict Diagram**

In the Drawing and Predict scenarios, the activity diagram can be summarized as follows: The user initiates the air writing software on their PC, which opens the webcam for hand tracking. The system continuously tracks the hand movements and recognizes gestures. If the hand position and finger states meet the condition **'upFingers[1] and not upFingers[2] and not upFingers[3] and not upFingers[4]'**, the drawing operation starts, and the drawing stroke is appended to the file.txt.

The provided code snippet represents a system that utilizes webcam tracking to detect hand gestures. It verifies if fingers **upFingers[1] and upFingers[2] and upFingers[3] and upFingers[4] and not upFingers[0]** are raised while **upFingers[0]** is not raised. If this condition is satisfied, the system saves the coordinates of the user's drawing stroke to a file.

Afterwards, the system reads the saved data from the file, applies coordinate normalization, and appends the normalized data to a list. The normalized data is then written to another file for further use.

Subsequently, the code loads a pre-trained model and processes the test data. It predicts a Khmer character based on the input using the trained model. In case there are no saved data rows available for prediction, the system notifies the user to draw again, indicating that there is insufficient data for accurate character recognition.

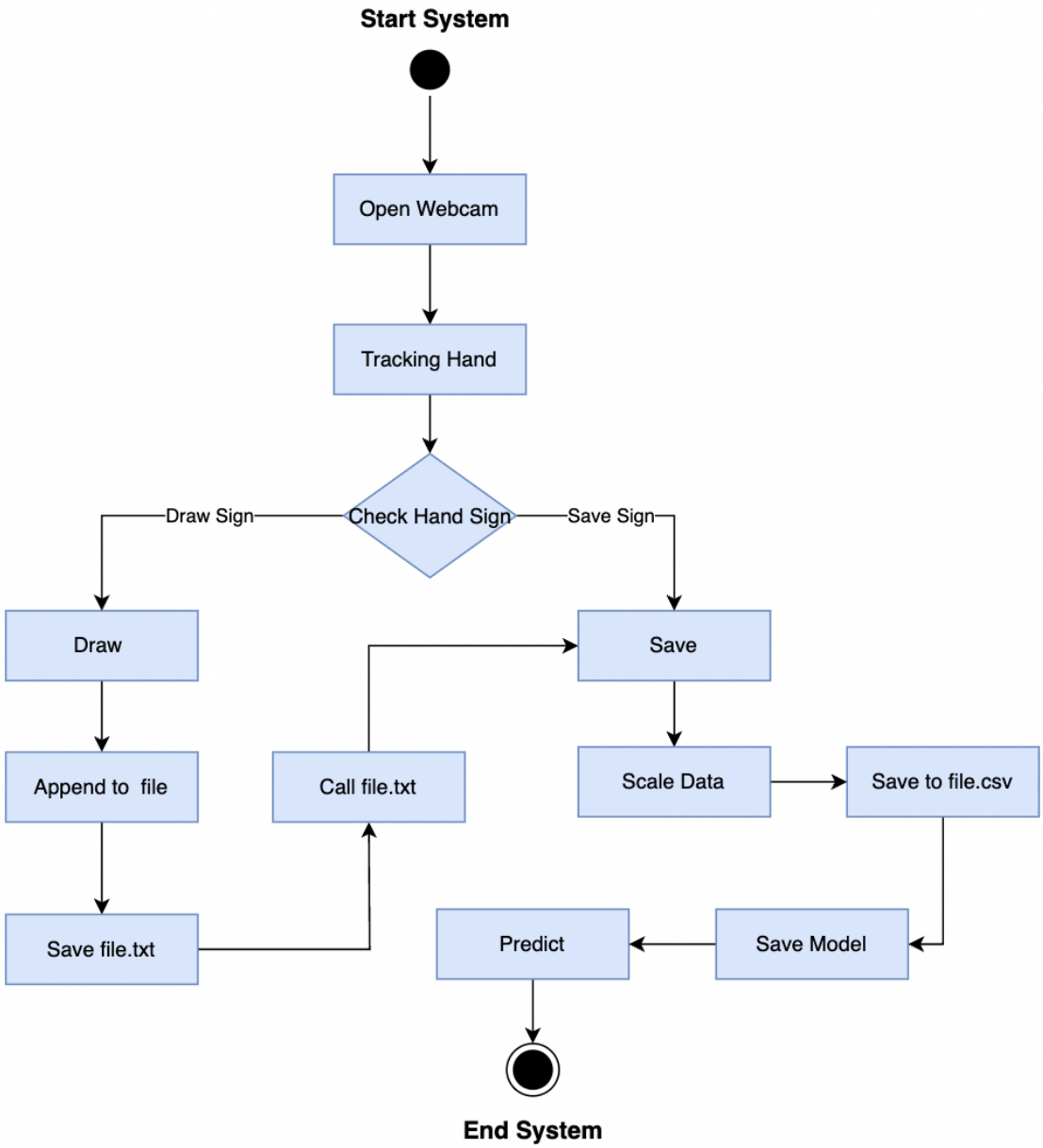


Figure 5: Drawing and Predict Activity Diagram

- Draw and Clear Diagram

In the Drawing and Clear scenarios, the activity diagram can be summarized as follows: The user initiates the air writing software on their PC, which opens the webcam for hand tracking. The system continuously tracks the hand movements and recognizes gestures. If the hand position and finger states meet the condition **'upFingers[1] and not upFingers[2] and not upFingers[3] and not upFingers[4]'**, the drawing operation starts, and the drawing stroke is appended to the file.txt. Furthermore, if the webcam tracking detects that the hand position and finger states match the condition **'upFingers[0] and not upFingers[1]'**, the Clear operation starts, and it accesses file.txt to clear the coordinate data of the user's drawing.

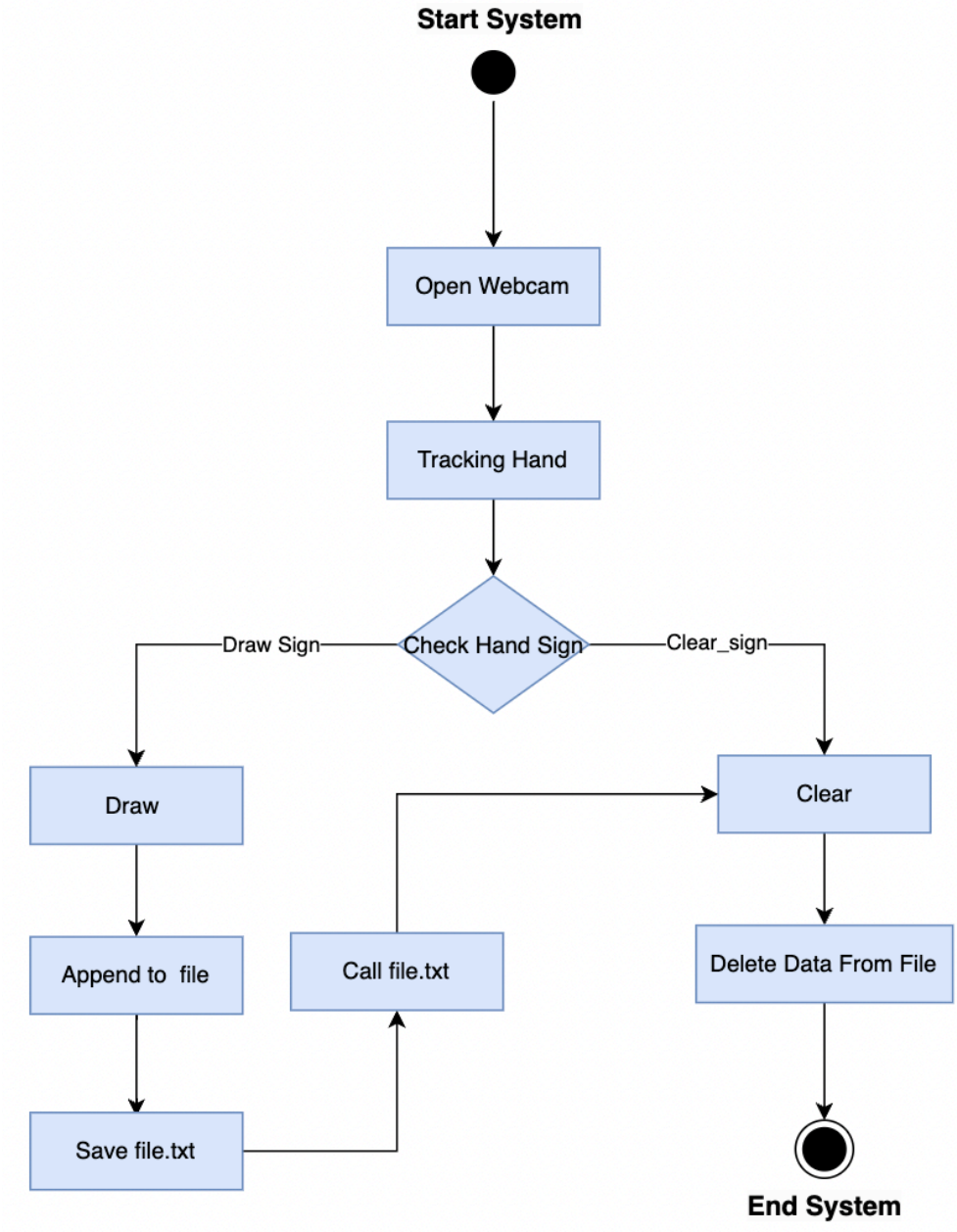


Figure 6: Drawing and Clear Activity Diagram

- **Close System Diagram**

In a closed system scenario, the activity diagram can be summarized as follows: The user initiates the air writing software on their PC, which opens the webcam for hand tracking. The system continuously tracks the hand movements and recognizes gestures. If the hand position and finger states do not match the condition "**not upFingers[1] and not upFingers[2] and not upFingers[3] and not upFingers[4] and not upFingers[0]**", the system remains active. However, if the condition is met, the system closes by turning off the webcam and terminating the software. The system processes the recognized gestures, predicts Khmer characters using a pre-trained model, and displays the predictions on the PC screen. The loop continues, allowing for real-time predictions until the user manually terminates the session.

In the condition "**not upFingers[1] and not upFingers[2] and not upFingers[3] and not upFingers[4] and not upFingers[0]**", it means that we have accessed all the endpoints of the five fingers. If the webcam tracking detects that none of them are raised, the system will automatically stop.

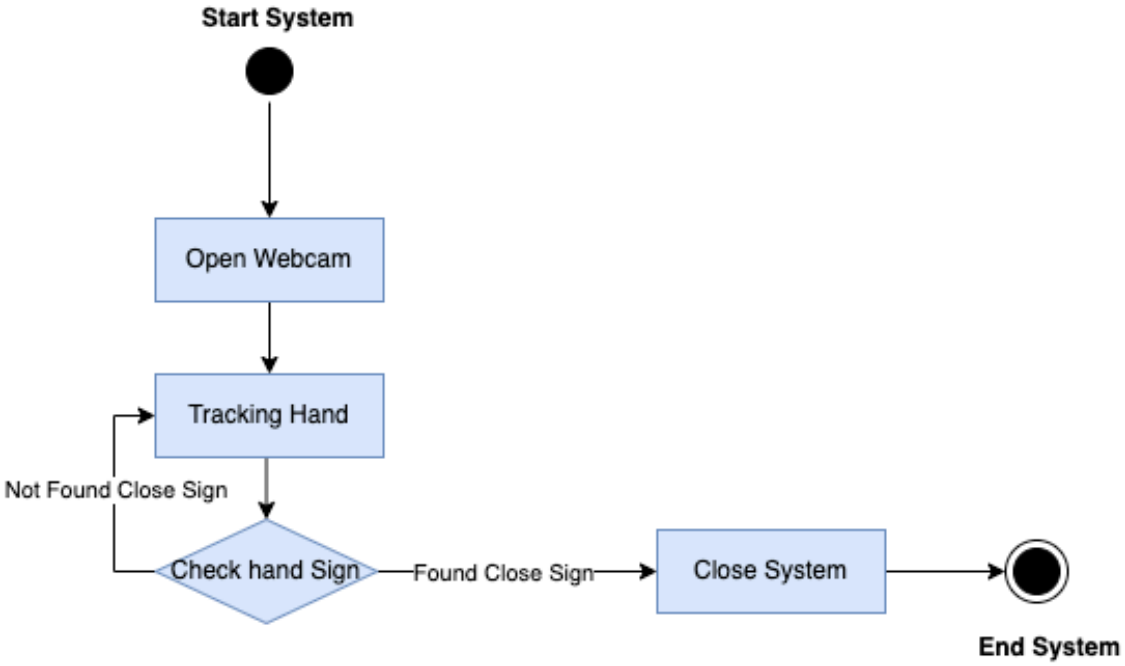


Figure 7: Close System Activity Diagram

3.2. Architecture of the Application

3.2.1. Physical Architecture

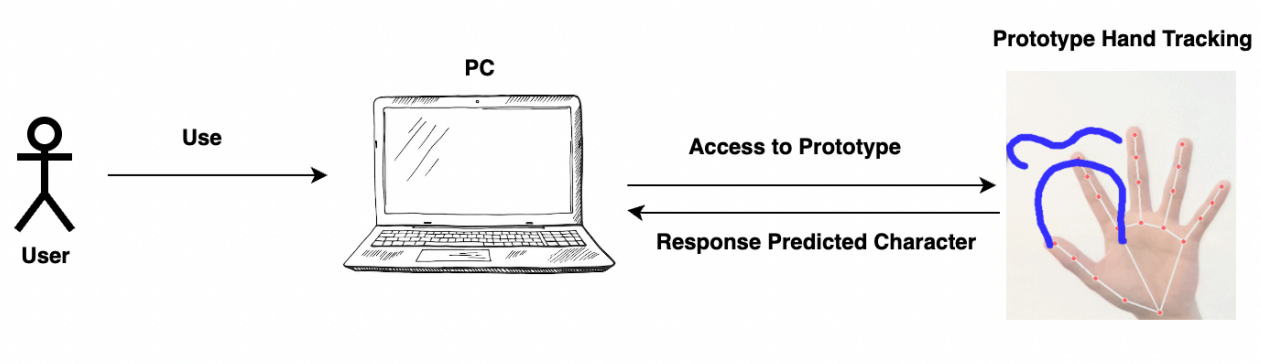


Figure 8: Physical Architecture

The physical architecture of the Air Writing project consists of a user's personal computer (PC) and a webcam, which are essential components for interacting with the air writing prototype software. The user accesses the software through their PC, initiating the webcam to capture their hand movements as they perform air writing gestures. The prototype software employs sophisticated algorithms to track the position of the hand in real time, utilizing computer vision techniques to analyze the captured data. By leveraging a pre-trained machine learning model, the software accurately predicts the Khmer characters corresponding to the hand movements. The predicted characters are then promptly displayed on the PC's screen, providing immediate visual feedback to the user. This physical architecture enables users to interact with the air writing system seamlessly, empowering them to express their thoughts and ideas through air writing with the aid of technology.

3.2.2. Logical Architecture

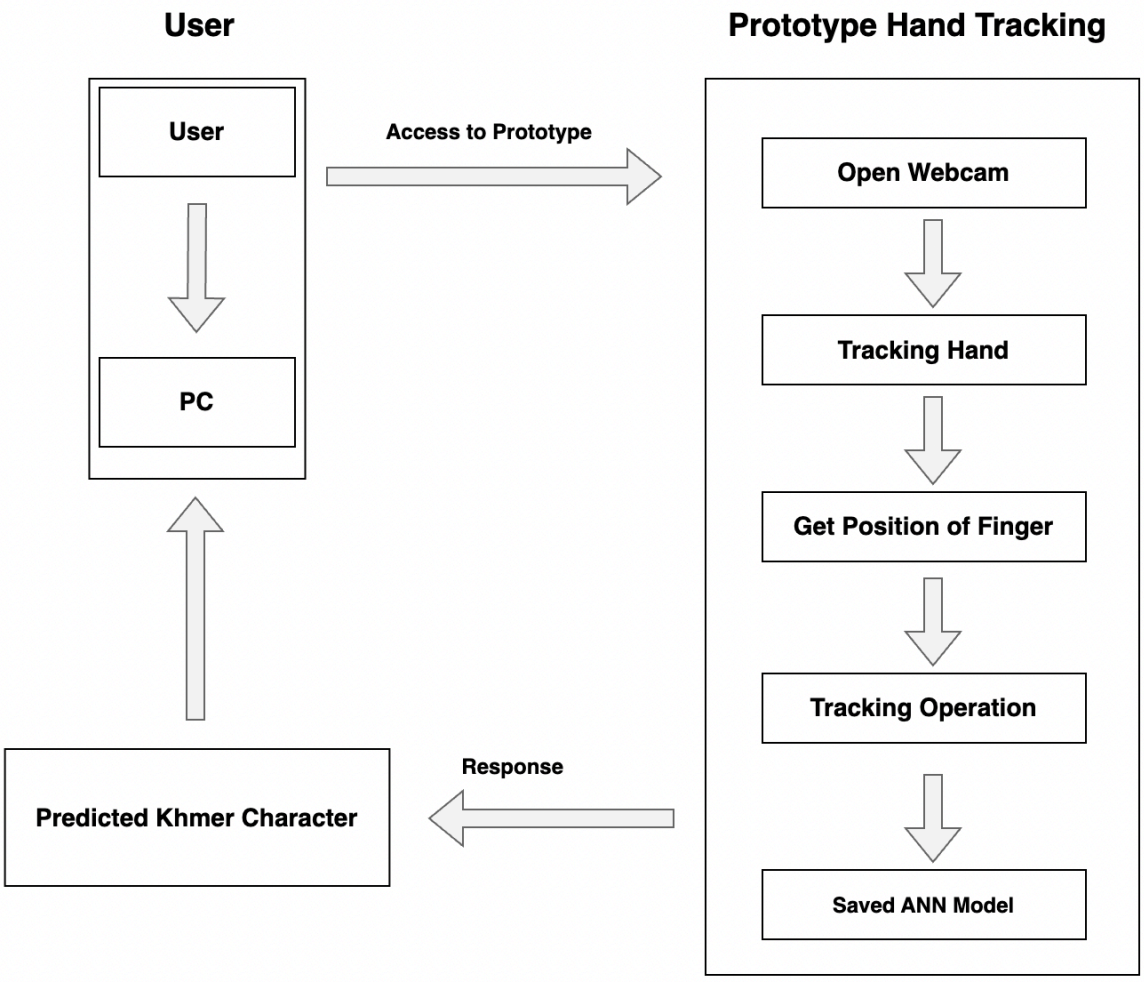


Figure 9: Logical Architecture

The logical architecture of the Air Writing project consists of a user utilizing a PC to access the air writing prototype. The prototype initiates the camera functionality, enabling it to track the movements of the user's hand. As the hand moves, its position is captured and undergoes hand tracking operations. The data obtained from the hand tracking process is then fed into a pre-trained model, which analyzes the input and predicts the corresponding Khmer characters.

Once the prediction is made, the results are transmitted back to the PC interface. The PC screen (OpenCV Frame) displays the predicted Khmer characters as a response to the user's air writing input. This logical architecture allows users to interact with the system, enabling them to write in the air and receive the corresponding Khmer character output on their PC screen (OpenCV Frame).

3.3. Frameworks and Technology Choice

3.3.1. Python

Python is a popular high-level programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability, making it easier to write and understand compared to other programming languages. Python supports multiple Integrated Development Environments (IDEs) like PyCharm, Visual Studio Code, Atom, and Jupyter Notebook, which provide additional features for code editing, debugging, and project management.

Figure 10: Python Logo

3.3.2. Microsoft Excel

Microsoft Excel is a widely used spreadsheet program developed by Microsoft. It is a part of the Microsoft Office suite and provides a powerful tool for data organization, analysis, and visualization. Excel is commonly used in various industries, including finance, accounting, marketing, and project management.

Figure 11: Excel Logo

3.3.3. Pytorch

PyTorch is a popular open-source machine learning framework developed by Facebook's AI Research (FAIR) lab. It provides a dynamic and flexible platform for building and training various types of deep learning models. PyTorch is widely used for tasks such as image classification, natural language processing, computer vision, and reinforcement learning.Once installed, you can use PyTorch to build, train, and evaluate deep learning models. It's helpful to have a good understanding of Python programming and basic machine learning concepts before diving into PyTorch. There are online courses, tutorials, and books available that specifically focus on learning PyTorch.

Figure 12: Pytorch Logo

3.3.4. Pandas

Pandas is a powerful open-source library for data manipulation and analysis in Python. It provides easy-to-use data structures and data analysis tools, making it popular among data scientists, analysts, and researchers. Pandas is built on top of NumPy and is often used in conjunction with other libraries like Matplotlib and scikit-learn. Pandas simplifies many data-related tasks and allows you to perform data analysis tasks efficiently. It provides an extensive set of functionalities and can handle datasets of various sizes, making it suitable for both small and large-scale data analysis projects.

Figure 13: Pandas Logo

3.3.5. Numpy

NumPy (Numerical Python) is a powerful open-source library for numerical computing in Python. It provides efficient and high-performance multidimensional array objects, along with a collection of mathematical functions, to facilitate numerical operations. NumPy serves as fundamental building block for various scientific and data-related libraries in the Python ecosystem.

Figure 14: Numpy Logo

3.3.6. OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning library that offers a wide range of tools and functions for image and video processing, object detection and tracking, and various other computer vision tasks. It is written in C++ and provides bindings for Python and other programming languages.

Computer vision involves the extraction, analysis, and understanding of information from digital images or videos. OpenCV simplifies the development of computer vision applications by providing a rich set of functions and algorithms. Here are some key features and functionalities of OpenCV:

Figure 15: OpenCV Logo

3.3.7. Pillow

Pillow is a powerful Python library for image processing and manipulation. It provides a wide range of functions for opening, manipulating, and saving many different image file formats. Pillow is a fork of the Python Imaging Library (PIL) and offers a more active and maintained development community.

Figure 16: Pillow Logo

3.3.8. MediaPipe

MediaPipe is an open-source framework developed by Google that provides a platform for building real-time multimedia processing pipelines. It offers a wide range of pre-built components and tools for processing and analyzing various forms of media, including video, audio, and 3D data. MediaPipe is designed to be efficient, portable, and flexible, making it suitable for a variety of applications. MediaPipe provides extensive documentation, including tutorials, code samples, and API references, which can be found on the official MediaPipe website(mediapipe.dev). The documentation guides you through the process of building MediaPipe pipelines, integrating machine learning models, and deploying applications on different platforms.

Figure 17: MediaPipe Logo

3.3.9. Anaconda

Anaconda is an open-source distribution of the Python and R programming languages, primarily used for data science and machine learning tasks. It provides a convenient and comprehensive platform for managing and working with various libraries, packages, and environments. Anaconda includes the Anaconda Navigator graphical user interface (GUI), as well as command-line tools, to simplify package and environment management.

Figure 18: Anaconda Logo

3.3.10. Visual Studio Code

Visual Studio Code (VSCode) is a lightweight and highly extensible source code editor developed by Microsoft. It offers a clean and intuitive user interface, cross-platform support, and a rich set of features including code editing with IntelliSense, debugging capabilities, version control integration, task automation, and a marketplace for extensions. VSCode enhances productivity through customization options, an integrated terminal, and various productivity features. It is widely used and supported by a vibrant community of developers.

Figure 19: Visual Studio Code Logo

CHAPTER IV PROJECT IMPLEMENTATION

4.1. Model Implementation

4.1.1. Related Technology

- **Air Writing Techniques and Applications**

Air writing techniques refer to the various methods and approaches used to write or create visible characters in mid-air without physical contact with a writing surface. These techniques rely on motion and gesture tracking systems to capture and interpret the movements of the user's hand or fingers. Some common air writing techniques include:

- **Trajectory-based Techniques**: Trajectory-based air writing techniques track hand movements in 3D space to recognize characters or symbols. Accelerometers, gyroscopes, or cameras can be used to capture and interpret hand movements.

- **Gesture-based Techniques**: Gesture-based air writing techniques use predefined gestures to input characters or commands. Computer vision and machine learning are used to recognize and interpret the gestures.

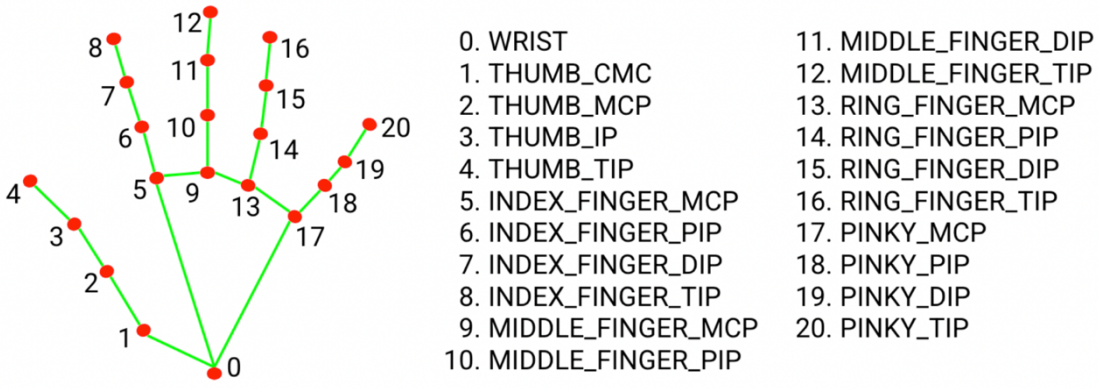


Figure 20: Hand Tracking Detail

Air writing serves as an alternative input method for computers and digital devices, offering intuitive and natural interaction. It enhances user experiences in virtual and augmented reality, enabling virtual annotations, 3D drawing, and object manipulation. In education and training, air writing facilitates interactive learning in areas such as handwriting, foreign languages, and spatial visualization, providing immediate feedback and personalized guidance.

- **Computer Vision (CV)**

Computer vision is a multidisciplinary field that focuses on enabling computers to acquire, process, analyze, and understand visual information from images or videos. It involves developing algorithms and techniques that mimic human vision capabilities, allowing machines to interpret and make sense of visual data.

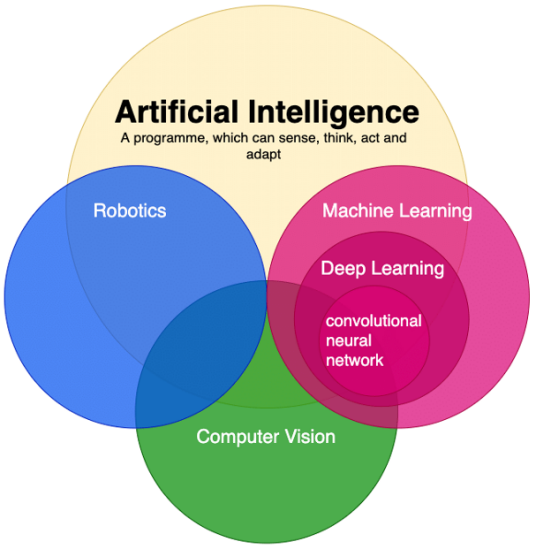


Figure 21: Relation AI and ML

Computer vision is applied in many domains, including object recognition, image analysis, surveillance, AR/VR, autonomous vehicles, medical imaging, robotics, and human-computer interaction. It plays a crucial role in tasks like object categorization, image

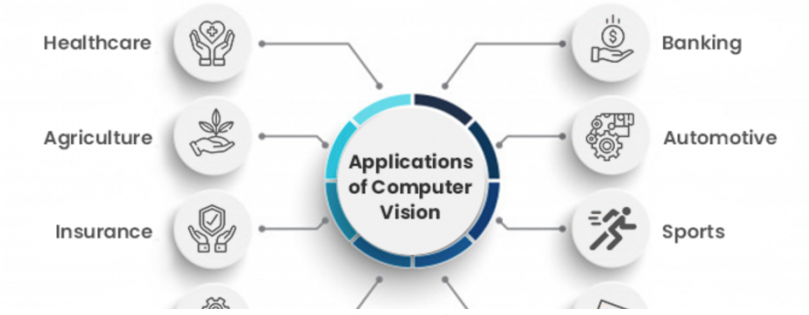


Figure 22: Application of Computer Vision

segmentation, tracking, AR/VR mapping, perception in autonomous vehicles, medical image analysis, surgical assistance, robotic vision, and natural human-computer interaction.

- **Machine Learning (ML)**

Machine learning is a subfield of AI that uses statistical modeling to identify patterns and improve performance based on data. It does not rely on direct programming commands, but rather on algorithms that process input data to perform specific tasks.

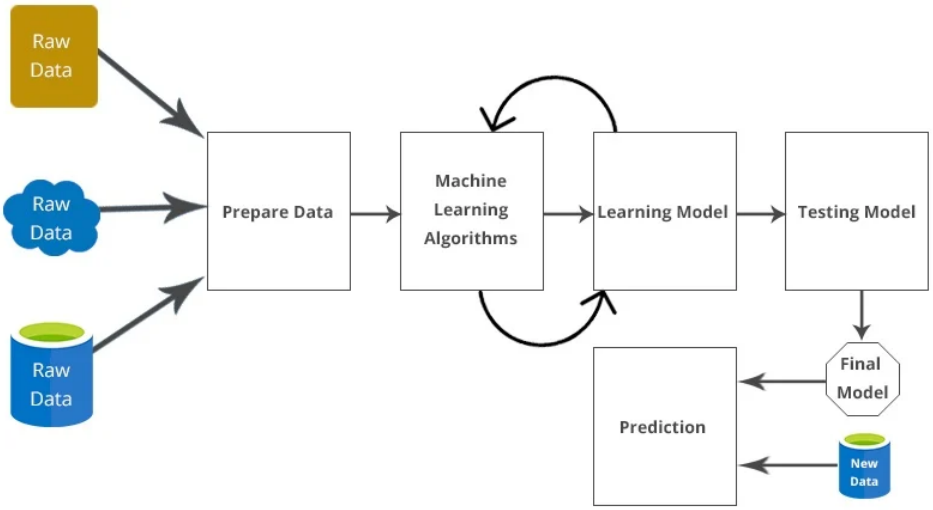


Figure 23: ML Work Flow

Machine Learning (ML) encompasses four fundamental approaches: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

- **Supervised learning** trains algorithms to correlate input variables with labeled output variables, enabling them to make predictions on new, unseen data.
- **Unsupervised learning** explores unlabeled data to discover patterns, structures, or relationships, aiming to gain insights and understanding.
- **Semi-supervised learning** combines labeled and unlabeled data, allowing the model to benefit from both types of data.
- **Reinforcement learning** teaches machines to complete tasks by programming algorithms to perform actions and receive feedback in the form of rewards or penalties, enabling them to autonomously determine the steps that maximize their cumulative reward.

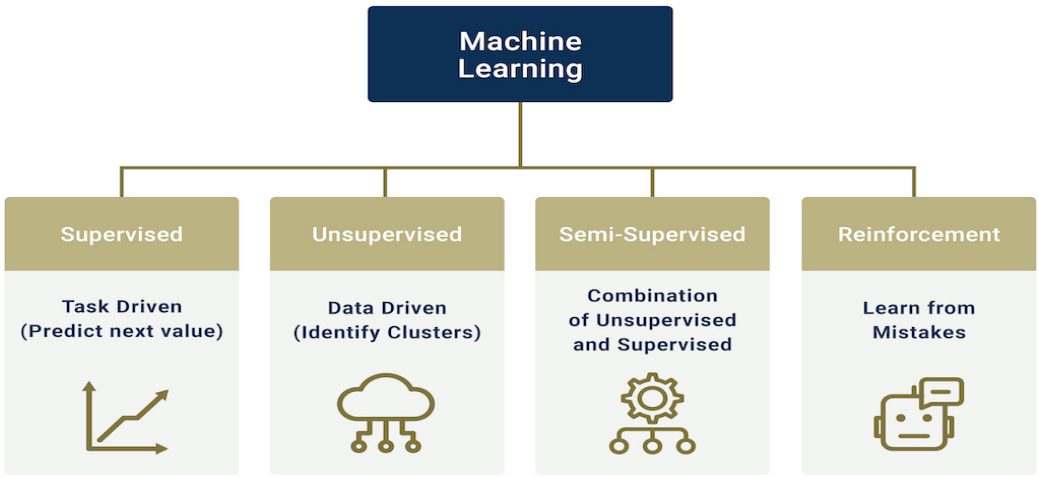


Figure 25: Types of ML

- **Deep Learning (DL) and Neural Network (NN)**

Deep learning is a subset of machine learning that uses artificial neural networks to learn from data. It requires more computational power than machine learning, but it can learn from large datasets with minimal human intervention. Deep learning excels at analyzing images, videos, and unstructured data.

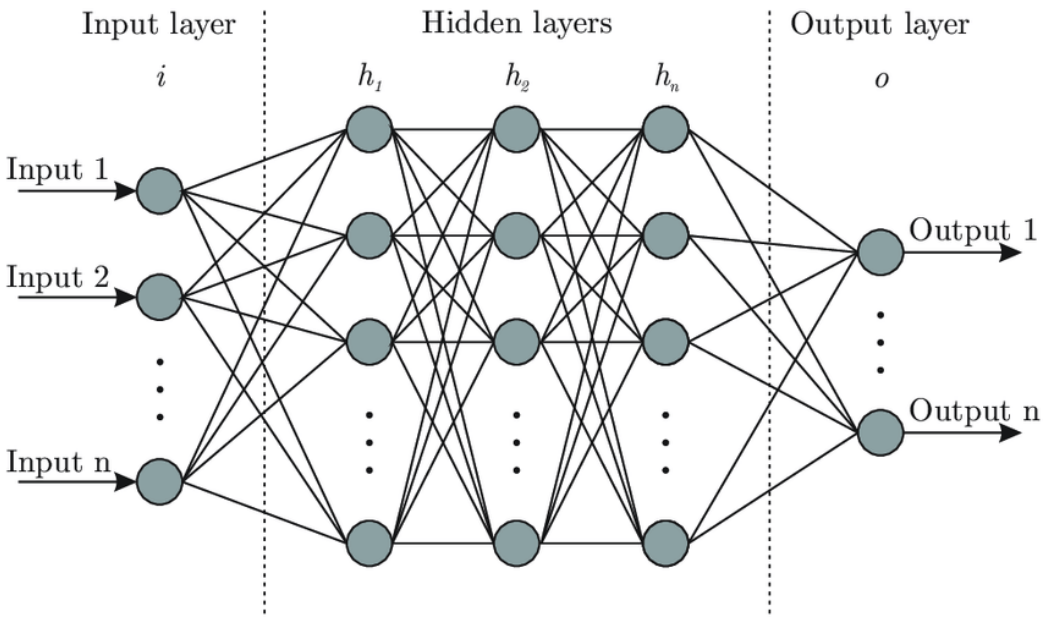
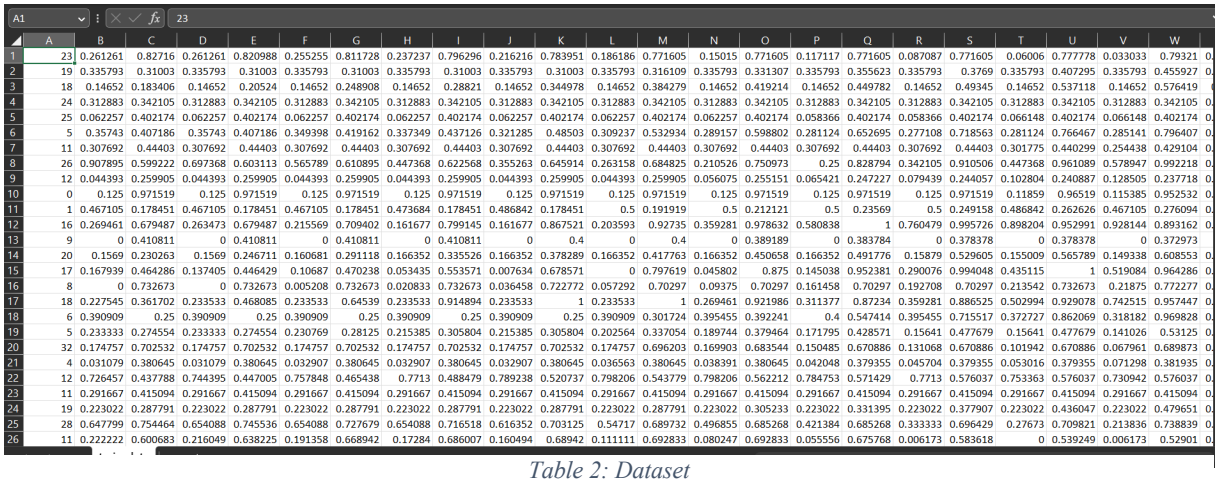


Figure 26: Work Flow of NN

4.1.2. Architecture of the Proposed Model

- **Process of Dataset**

The data collection procedure involved simulating air writing by drawing Khmer characters on a screen using a cursor. Our data collection program tracked the cursor movements of the users and captured the corresponding stroke coordinates. These coordinates were then saved in text files and later consolidated into a single CSV file. Each row in the CSV file represented a sample, with columns indicating the (x, y) coordinates and the corresponding Khmer character label. To ensure diversity in writing styles and variations, multiple writers from our team, consisting of 11 members, participated in generating the dataset. In total, we collected 18,150 samples (550 samples per letter) for the experiment. To facilitate further analysis, the dataset was transformed into a CSV file format. In this transformation, the coordinates were scaled to a range of 0-1, ensuring uniformity, and each row in the resulting CSV file represented a distinct sample.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	23	0.261261	0.82716	0.261261	0.820988	0.255255	0.811728	0.237237	0.796296	0.216216	0.783951	0.186186	0.771605	0.15015	0.771605	0.117117	0.771605	0.087087	0.771605	0.06006	0.777778	0.033033	0.79321
2	19	0.335793	0.31003	0.335793	0.31003	0.335793	0.31003	0.335793	0.31003	0.335793	0.31003	0.335793	0.316109	0.335793	0.331307	0.335793	0.355623	0.335793	0.3769	0.335793	0.407295	0.335793	0.455927
3	18	0.14652	0.183406	0.14652	0.20524	0.14652	0.248908	0.14652	0.28821	0.14652	0.344978	0.14652	0.384279	0.14652	0.419214	0.14652	0.449782	0.14652	0.49345	0.14652	0.537118	0.14652	0.576419
4	24	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105	0.312883	0.342105
5	25	0.062257	0.402174	0.062257	0.402174	0.062257	0.402174	0.062257	0.402174	0.062257	0.402174	0.062257	0.402174	0.062257	0.402174	0.058366	0.402174	0.058366	0.402174	0.066148	0.402174	0.066148	0.402174
6	5	0.35743	0.407186	0.35743	0.407186	0.349398	0.419162	0.337349	0.437126	0.321285	0.48503	0.309237	0.532934	0.289157	0.598802	0.281124	0.652695	0.277108	0.718563	0.281124	0.766467	0.285141	0.796407
7	11	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.307692	0.44403	0.301775	0.440299	0.254438	0.429104
8	26	0.907895	0.599222	0.697368	0.603113	0.565789	0.610895	0.447368	0.622568	0.355263	0.645914	0.263158	0.684825	0.210526	0.750973	0.25	0.828794	0.342105	0.910506	0.447368	0.961089	0.578947	0.992218
9	12	0.044393	0.259905	0.044393	0.259905	0.044393	0.259905	0.044393	0.259905	0.044393	0.259905	0.044393	0.259905	0.056075	0.255151	0.065421	0.247227	0.079439	0.244057	0.102804	0.240887	0.128505	0.237718
10	0	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.125	0.971519	0.11859	0.96519	0.115385	0.952532
11	1	0.467105	0.178451	0.467105	0.178451	0.467105	0.178451	0.473684	0.178451	0.486842	0.178451	0.5	0.191919	0.5	0.212121	0.5	0.23569	0.5	0.249158	0.486842	0.262626	0.467105	0.276094
12	16	0.269461	0.679487	0.263473	0.679487	0.215569	0.709402	0.161677	0.799145	0.161677	0.867521	0.203593	0.92735	0.359281	0.978632	0.580838	1	0.760479	0.995726	0.898204	0.952991	0.928144	0.893162
13	9	0	0.410811	0	0.410811	0	0.410811	0	0.410811	0	0.4	0	0.4	0	0.389189	0	0.383784	0	0.378378	0	0.378378	0	0.372973
14	20	0.1569	0.230263	0.1569	0.246711	0.160681	0.291118	0.166352	0.335526	0.166352	0.378289	0.166352	0.417763	0.166352	0.450658	0.166352	0.491776	0.15879	0.529605	0.155009	0.565789	0.149338	0.608553
15	17	0.167939	0.464286	0.137405	0.446429	0.10687	0.470238	0.053435	0.553571	0.007634	0.678571	0	0.797619	0.045802	0.875	0.145038	0.952381	0.290076	0.994048	0.435115	1	0.519084	0.964286
16	8	0	0.732673	0	0.732673	0.005208	0.732673	0.020833	0.732673	0.036458	0.722772	0.057292	0.70297	0.09375	0.70297	0.161458	0.70297	0.192708	0.70297	0.213542	0.732673	0.21875	0.772277
17	18	0.227545	0.361702	0.233533	0.468085	0.233533	0.64539	0.233533	0.914894	0.233533	1	0.233533	1	0.269461	0.921986	0.311377	0.87234	0.359281	0.886525	0.502994	0.929078	0.742515	0.957447
18	6	0.390909	0.25	0.390909	0.25	0.390909	0.25	0.390909	0.25	0.390909	0.25	0.390909	0.301724	0.395455	0.392241	0.4	0.547414	0.395455	0.715517	0.372727	0.862069	0.318182	0.969828
19	5	0.233333	0.274554	0.233333	0.274554	0.230769	0.28125	0.215385	0.305804	0.215385	0.305804	0.202564	0.337054	0.189744	0.379464	0.171795	0.428571	0.15641	0.477679	0.15641	0.477679	0.141026	0.53125
20	32	0.174757	0.702532	0.174757	0.702532	0.174757	0.702532	0.174757	0.702532	0.174757	0.702532	0.174757	0.696203	0.169903	0.683544	0.150485	0.670886	0.131068	0.670886	0.101942	0.670886	0.067961	0.689873
21	4	0.031079	0.380645	0.031079	0.380645	0.032907	0.380645	0.032907	0.380645	0.032907	0.380645	0.036563	0.380645	0.038391	0.380645	0.042048	0.379355	0.045704	0.379355	0.053016	0.379355	0.071298	0.381935
22	12	0.726457	0.437788	0.744395	0.447005	0.757848	0.465438	0.7713	0.488479	0.789238	0.520737	0.798206	0.543779	0.798206	0.562212	0.784753	0.571429	0.7713	0.576037	0.753363	0.576037	0.730942	0.576037
23	11	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094	0.291667	0.415094
24	19	0.223022	0.287791	0.223022	0.287791	0.223022	0.287791	0.223022	0.287791	0.223022	0.287791	0.223022	0.287791	0.223022	0.305233	0.223022	0.331395	0.223022	0.377907	0.223022	0.436047	0.223022	0.479651
25	28	0.647799	0.754464	0.654088	0.745536	0.654088	0.727679	0.654088	0.716518	0.616352	0.703125	0.54717	0.689732	0.496855	0.685268	0.421384	0.685268	0.333333	0.696429	0.27673	0.709821	0.213836	0.738839
26	11	0.222222	0.600683	0.216049	0.638225	0.191358	0.668942	0.17284	0.686007	0.160494	0.68942	0.111111	0.692833	0.080247	0.692833	0.055556	0.675768	0.006173	0.583618	0	0.539249	0.006173	0.52901

Table 2: Dataset

- **Proposed Method**

Architecture:

- The input to the model is a vector of size in_sz.
- The model consists of multiple fully connected layers with ReLU activation functions, and a final fully connected layer for the output.
- The output of the model is a vector of size out_sz, representing 33 predicted output of Khmer characters.

Settings/Configurations:

- The fully connected layers are defined using the nn.Linear module in PyTorch.
- The number of layers and units in each layer can be set using the layers_sz argument in the __init__ method of the SimpleNet class. The first layer has in_sz input units and the output of each layer is passed through a ReLU activation function.
- The final fully connected layer has out_sz output units and does not use an activation function.
- The nn.ModuleList is used to create a list of PyTorch modules, which allows the model to be trained end-to-end using automatic differentiation.
- The forward method is used to define how input is propagated through the model.
- The input vector is passed through the fully connected layers to produce the output vector.

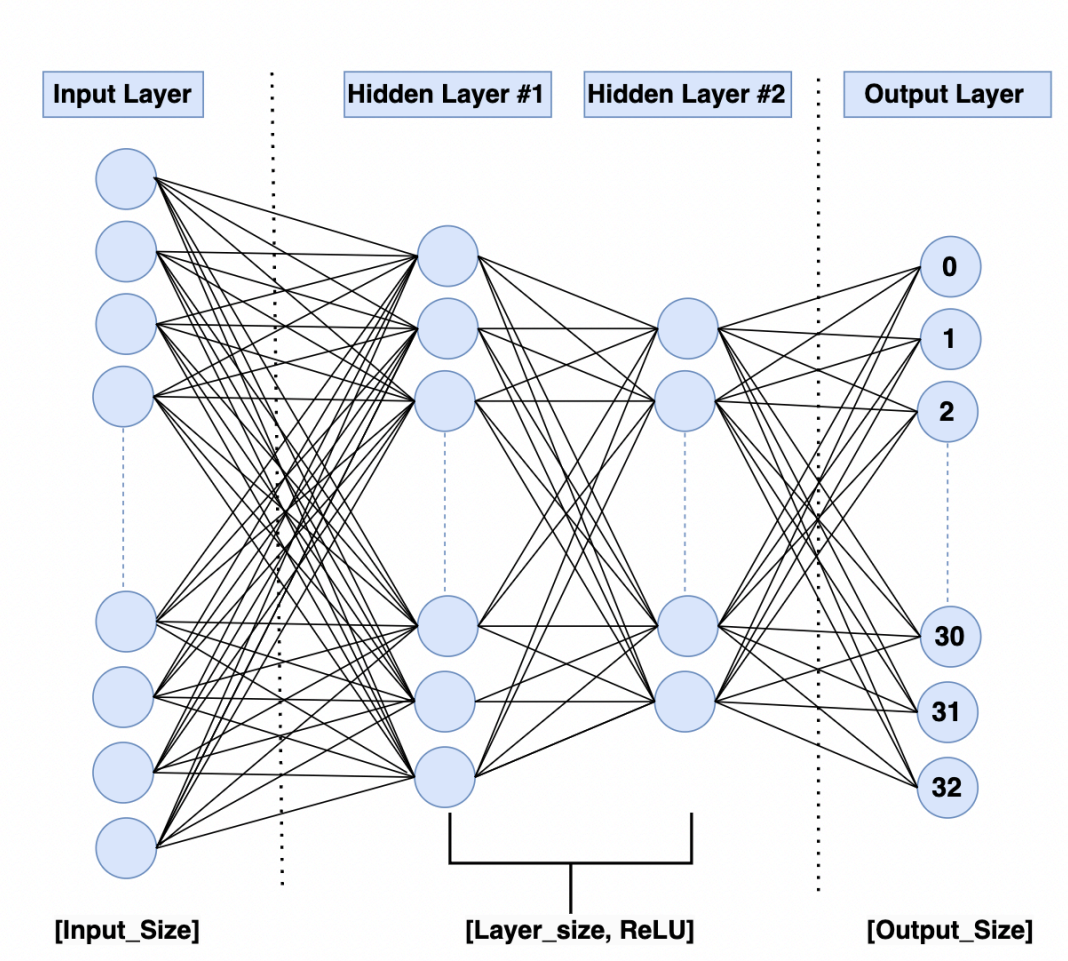


Figure 27: Flow of ANN Work in Proposed Method

- **Results**

Hyper parameter			Highest accuracy of each experiment
Layer size	**N-epoch**	**B_size**	
1024,512	15	100	**94.78%**
1024,512,256	15	30	88.52%
512,256	15	30	87.85%
1024,512,256,128	15	30	85.32%
256,128	15	30	85.26%
512,256,128	15	30	85.02%
1024,512,256,128,64	15	30	82.96%
512,256,128,64	15	30	82.54%
128,64,32	15	30	78.97%

Table 3: Results Table

Figure 28: Results

4.2. System Implementation

In this system, we have divided the project into four main classes: HandTracker, ColorRect, Point, and. These classes play distinct roles and contribute to different aspects of the system's functionality. Libraries and Modules are imported for functionality, a ColorRect Class is defined, Hand Tracking is initialized, Canvas and Drawing are set up, Coordinate and File Handling are established, User Interface (UI) Elements are defined, Camera is initialized, and a Main Loop continuously captures video frames, performing tasks such as Hand Tracking, Finger Detection, Gesture Recognition, Drawing on Canvas, Displaying UI Elements, Text Display, File Saving and Data Processing, Prediction, User Interface handling, and Loop Control until the program is exited.

HandTracker class: The HandTracker class is responsible for tracking and analyzing hand movements or gestures, enabling hand-based interaction within the system. It likely utilizes computer vision techniques to accurately track the position and movements of the user's hand. The HandTracker class contains several functions that contribute to its functionality in tracking and analyzing hand movements. Here are some functions that could be included in the HandTracker class:

- __init__ (self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):

  - Initializes a `HandTracker` object with the specified parameters.
  - Sets the mode, maximum number of hands, detection confidence, and tracking confidence.
  - Initializes the 'mpHands.Hands' object from the MediaPipe library for hand tracking.
  - Initializes the 'mpDraw' object for drawing hand landmarks on images.

```
def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
    self.mode = mode
    self.maxHands = maxHands
    self.detectionCon = detectionCon
    self.trackCon = trackCon

    self.mpHands = mp.solutions.hands
    self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.detectionCon, self
    self.mpDraw = mp.solutions.drawing_utils
```

Figure 29: Initialized Function

- findHands(self, img, draw=True):
  - Takes an image `img` and performs hand detection on it.
  - Converts the image to RGB format.
  - Processes the image using the 'hands.process()' method, which returns the hand landmarks if any are detected.
  - If `draw` is True, it draws the hand landmarks and connections on the image using 'mpDraw.draw_landmarks ()'.

- Returns the image with or without the drawn hand landmarks.

```
def findHands(self, img, draw=True):
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imgRGB)

    if self.results.multi_hand_landmarks:
        for handLm in self.results.multi_hand_landmarks:
            if draw:
                self.mpDraw.draw_landmarks(img, handLm, self.mpHands.HAND_CONNECTIONS)
    return img
```

Figure 30: Find Hand Function

- **getPostion (self, img, handNo=0, draw=True):**
  - Takes an image 'img' and retrieves the hand landmarks for a specified hand number 'handNo'.
  - Extracts the hand landmarks from the results object.
  - For each landmark ('lm') in the hand, it calculates the landmark's pixel coordinates ('cx', 'cy') relative to the image dimensions.
  - Appends the landmark coordinates ('cx', 'cy') to the 'lmList' list.
  - If 'draw' is True, it optionally draws circles at the landmark positions on the image.
  - Returns the list of landmark positions ('lmList').

```
def getPostion(self, img, handNo = 0, draw=True):
    lmList =[]
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for lm in myHand.landmark:
            h, w, c = img.shape
            cx, cy = int(lm.x*w), int(lm.y*h)
            lmList.append((cx, cy))

            if draw:
                cv2.circle(img, (cx, cy), 5, (255,0,255), cv2.FILLED)
    return lmList
```

Figure 31: Get Position

- **getUpFingers (self, img):**
  - Takes an image img and determines the state of each finger (up or down).
  - Calls the getPostion () method to retrieve the hand landmarks positions.
  - Initializes an empty upfingers list to store the finger states.
  - For each finger, it checks the position of specific landmarks to determine if the finger is up or down.

- Appends a Boolean value indicating the state of each finger to the ‘upfingers’ list.
- Returns the list of finger states (‘upfingers’).

```
def getUpFingers(self, img):
    pos = self.getPostion(img, draw=False)
    self.upfingers = []

    if pos:
        #thumb
        self.upfingers.append((pos[4][1] < pos[3][1] and (pos[5][0]-pos[4][0]> 10)))
        #index
        self.upfingers.append((pos[8][1] < pos[7][1] and pos[7][1] < pos[6][1]))
        #middle
        self.upfingers.append((pos[12][1] < pos[11][1] and pos[11][1] < pos[10][1]))
        #ring
        self.upfingers.append((pos[16][1] < pos[15][1] and pos[15][1] < pos[14][1]))
        #pinky
        self.upfingers.append((pos[20][1] < pos[19][1] and pos[19][1] < pos[18][1]))

    return self.upfingers
```

Figure 32: Get Fingers

ColorRect class: The ColorRect class is designed to create colored rectangles with optional text. This class provides a means to display graphical elements, such as rectangles, on the system's interface. It allows customization of the rectangles' colors and the ability to add text, making it suitable for visual representation purposes.

- __init__ (self, x, y, w, h, color, text=' ', alpha=0.5):
  - Initializes a ‘ColorRect’ object with the specified parameters.
  - Sets the position (‘x’, ‘y’), width (‘w’), height (‘h’), color (‘color’), text (‘text’), and alpha transparency ‘`alpha’) values.

```
def __init__(self, x, y, w, h, color, text='', alpha = 0.5):
    self.x = x
    self.y = y
    self.w = w
    self.h = h
    self.color = color
    self.text=text
    self.alpha = alpha
```

Figure 33: Initialized Board

- drawRect (self, img, text_color = (255,255,255),fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.8, thickness=2):
  - Takes an image 'img' and draws a colored rectangle with optional text on it.
  - Calculates the weighted sum of the image background (bg_rec) and a white rectangle (white_rect) to achieve the desired color and transparency effect.
  - Overlays the resulting rectangle (res) onto the original image at the specified position ('x', 'y') with the specified width ('w') and height ('h').
  - Uses the OpenCV 'cv2.putText()' function to draw the specified 'text' at the center of the rectangle.
  - The color, font face, font scale, and thickness of the text can be optionally customized using the provided parameters.

```
def drawRect(self, img, text_color=(255,255,255), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.8, thickness=2):
    #draw the box
    alpha = self.alpha
    bg_rec = img[self.y : self.y + self.h, self.x : self.x + self.w]
    white_rect = np.ones(bg_rec.shape, dtype=np.uint8)
    white_rect[:] = self.color
    res = cv2.addWeighted(bg_rec, alpha, white_rect, 1-alpha, 1.0)

    # Putting the image back to its position
    img[self.y : self.y + self.h, self.x : self.x + self.w] = res

    #put the letter
    tetx_size = cv2.getTextSize(self.text, fontFace, fontScale, thickness)
    text_pos = (int(self.x + self.w/2 - tetx_size[0][0]/2), int(self.y + self.h/2 + tetx_size[0][1]/2))
    cv2.putText(img, self.text,text_pos , fontFace, fontScale,text_color, thickness)
```

Figure 34: Draw Rectangle

- isOver (self, x, y):
  - Takes coordinates 'x' and 'y' as input and checks if they fall within the boundaries of the rectangle.
  - If the input coordinates are within the rectangle ('self.x', 'self.y', 'self.w', 'self.h'), it returns 'True'.
  - Otherwise, it returns 'False'.

The 'ColorRect' class provides a convenient way to create colored rectangles with optional text and transparency effects. The 'drawRect ()' method allows you to easily draw the rectangle on an image, while the 'isOver()' method helps determine if a point is inside the rectangle's boundaries

```
def isOver(self,x,y):
    if (self.x + self.w > x > self.x) and (self.y + self.h> y >self.y):
        return True
    return False
```

Figure 35: Is Over Function

By incorporating contributions from various classes, functions, and imported libraries, we can effectively work on the project. This collaborative approach allows us to harness the expertise and capabilities offered by different components of the codebase, leading to enhanced functionality and efficiency. To ensure clarity, please provide me with the initial steps you have in mind for your project.

- Initialize the camera capture using `cv2.VideoCapture`.
- Start a while loop to continuously capture frames from the camera.
- Inside the loop, perform the following steps:
  - Reduce the cooling counter to prevent rapid consecutive actions.
  - Read a frame from the camera and resize it.
  - Flip the frame horizontally using `cv2.flip`.
  - Use the hand detector to find hands in the frame and get hand positions.
  - Check for the state of the fingers to perform different actions: If the index finger is up and not over the whiteboard, update the brush size or color based on the selected option. If only the index finger is up and over the whiteboard, draw on the canvas using the current brush size and color. If the thumb finger is up and the index finger is down, clear the canvas and reset the file. If all fingers are down, end the program.
  - Update the canvas with the drawn lines.
  - Draw the whiteboard, color options, and pen size options on the frame.
  - Perform character recognition if coordinates are available:Write the coordinates to a file. Scale the coordinates and save them to a separate file. Load the trained neural network model. Use the model to predict the Khmer character based on the scaled coordinates.
  - Display the frame with the drawn lines, whiteboard, and character predictions.
- End the while loop and exit the program when the user closes the window.

4.3. Prototype and Model Integration

The integration of the model and prototype in this project involves the following steps:

- **Model Training**: A machine learning model is trained using a dataset of scaled coordinate data and corresponding Khmer characters. The model learns to recognize patterns and make predictions based on the input data. After the model is trained, it will be saved into another file.
- **Model Loading in Prototype**: In the main loop of the prototype, the pre-trained model is loaded into memory. This allows the prototype to utilize the trained model for predicting Khmer characters.
- **Data Processing in Prototype**: Before making predictions, the prototype performs data processing on the coordinate data. This typically involves scaling the coordinates to a format compatible with the input requirements of the model.
- **Prediction**: Using the loaded model and processed data, the prototype predicts the Khmer character corresponding to the current hand gesture. The model applies its learned knowledge to make an educated guess based on the input data. Displaying Prediction: The predicted Khmer character is displayed on the video frame using the Text Display functionality. This provides visual feedback to the user regarding the character recognized by the system.

By integrating the trained model into the prototype, the system can leverage the power of machine learning to recognize and interpret hand gestures accurately, enabling the real-time prediction of Khmer characters as the user interacts with the whiteboard.

CHAPTER V CONCLUSION AND FUTURE WORKS

In conclusion, the Air Writing project has demonstrated the feasibility and potential of using hand gestures as a means of text input. Through the development and evaluation of our air writing system, we have successfully showcased how users can input text by simply gesturing in mid-air. The project utilized computer vision techniques, specifically the OpenCV library, to capture and process hand movements, converting them into meaningful text characters.

5.1. Completed Task

- Tracking Hand through Webcam
- Drawing Landmark on hand
- Get Position of Hand based on Landmark
- Drawing Board on Frame
- Put Image on Frame
- Put and Display Khmer Character on Frame
- Drawing Khmer Characters in the Air by Tracking Hand
- Be able to recognize Khmer drawn characters by integrating the prototype with the model
- Clearing the Drawing on Air by Tracking Hand
- Predict What We Drew
- Closing Frame Without Any Physical Touch.

5.2. Future Works

- Deploy to server
- Improve Smoothness
- Collect more dataset including vowel and special character of Khmer Characters.
- Train model to understand Khmer words and sentences
- Build an application for writing Khmer language and covert to other types of Khmer font

Reference:

[1] S. T. Ramya, R. Sakthi, B. Rohitha and D. Praveena, "Air-Writing Recognition System," 2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC), Bengaluru, India, 2022, pp. 910-913, doi: 10.1109/IIHC55949.2022.10059943.

[2] Al Abir, Fuad & Siam, Md & Sayeed, Abu & Hasan, Md. Al & Shin, Jungpil. (2021). Deep Learning Based Air-Writing Recognition with the Choice of Proper Interpolation Technique. Sensors. 21. 8407. 10.3390/s21248407.

[3] M. Arsalan and A. Santra, "Character Recognition in Air-Writing Based on Network of Radars for Human-Machine Interface," in IEEE Sensors Journal, vol. 19, no. 19, pp. 8855-8864, 1 Oct.1, 2019.doi: 10.1109/JSEN.2019.2922395

[4] S. Skaria, A. Al-Hourani and R. J. Evans, "Deep-Learning Methods for Hand-Gesture Recognition Using Ultra-Wideband Radar," in IEEE Access, vol. 8, pp. 203580-203590, 2020. doi: 10.1109/ACCESS.2020.3037062

[5] Y. Fang, Y. Xu, H. Li, X. He and L. Kang, "Writing in the air: Recognize Letters Using Deep Learning Through WiFi Signals," 2020 6th International Conference on Big Data Computing and Communications (BIGCOM), Deqing, China, 2020, pp. 8-14.doi: 10.1109/BigCom51056.2020.00008

[6] M. Chen, G. AlRegib and B. -H. Juang, "Air-Writing Recognition—Part I: Modeling and Recognition of Characters, Words, and Connecting Motions," in IEEE Transactions on Human-Machine Systems, vol. 46, no. 3, pp. 403-413, June 2016.
doi: 10.1109/THMS.2015.2492598

[7] M. Chen, G. AlRegib and B. -H. Juang, "Air-Writing Recognition—Part II: Detection and Recognition of Writing Activity in Continuous Stream of Motion Data," in IEEE Transactions on Human-Machine Systems, vol. 46, no. 3, pp. 436-444, June 2016. doi: 10.1109/THMS.2015.2492599

[8] Chen, X., Xu, M., Zeng, J., & Cai, A. (2019). Air-Writing Gesture Recognition Based on Convolutional Neural Network and Hidden Markov Model. Sensors, 19(21), 4648.

Appendix

- **Tracking Hand**

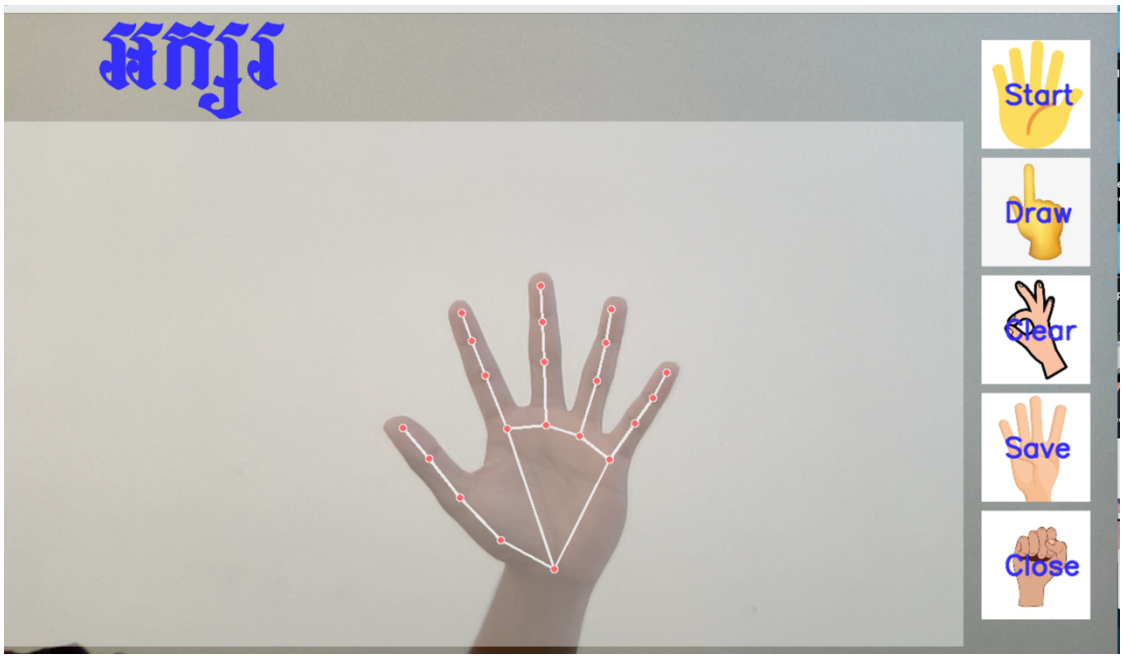


Figure 36: Tracking in Prototype

- **Drawing By Tracking Hand**

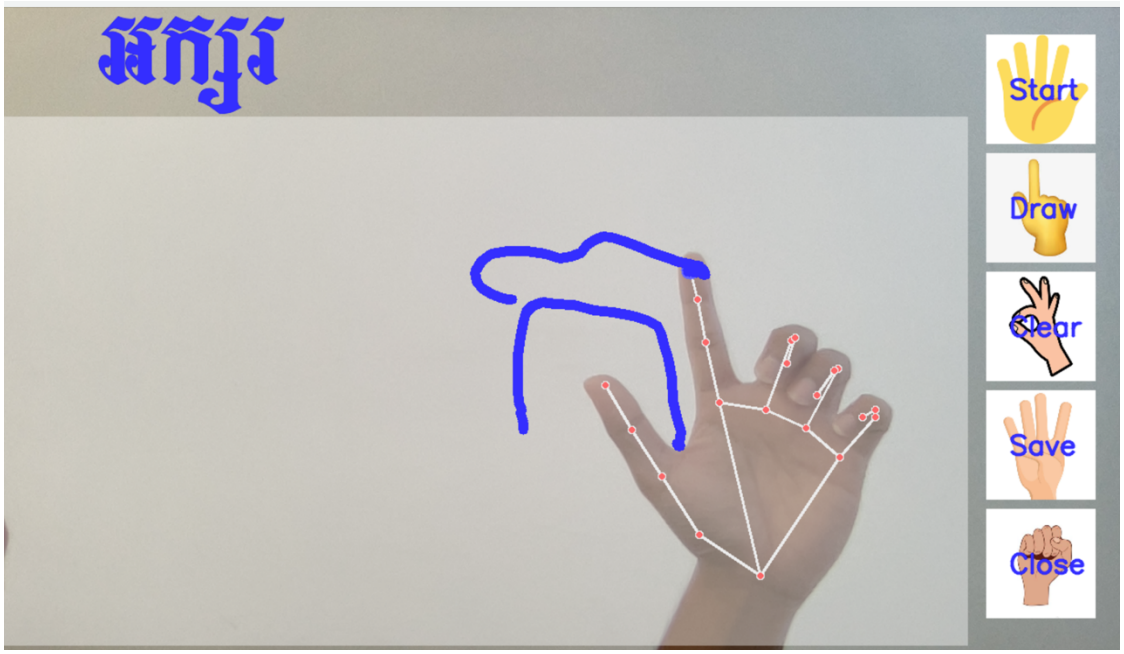


Figure 37: Drawing in Prototype

- **Save and Prediction**

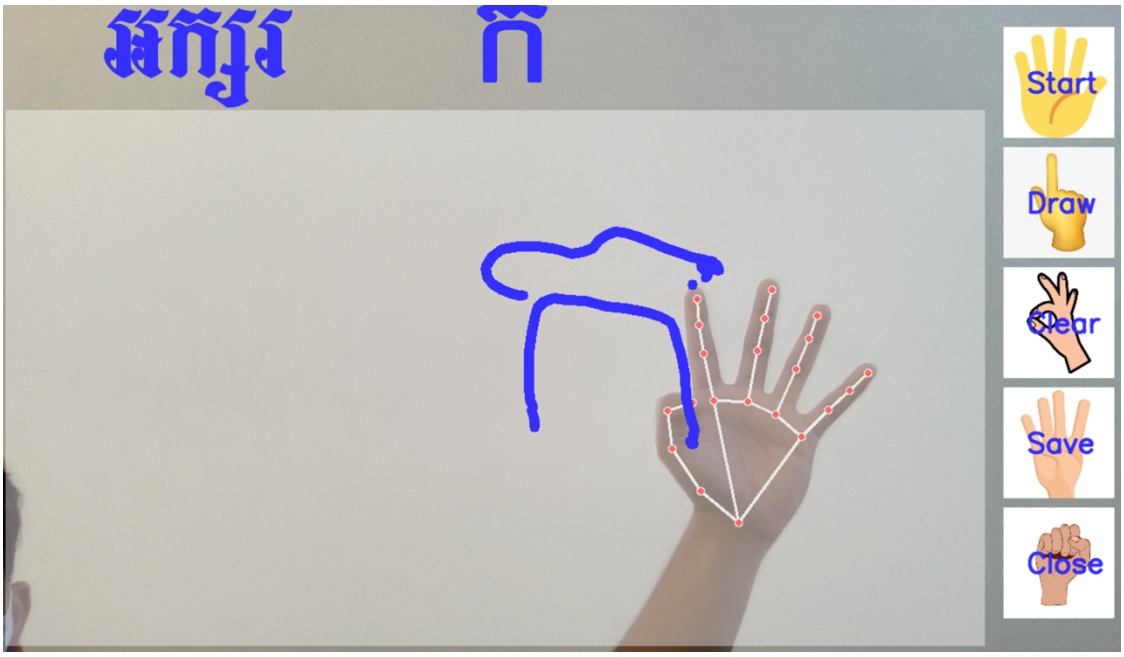


Figure 38: Save and Predict in Prototype

- **Clear Drawn Stroke**

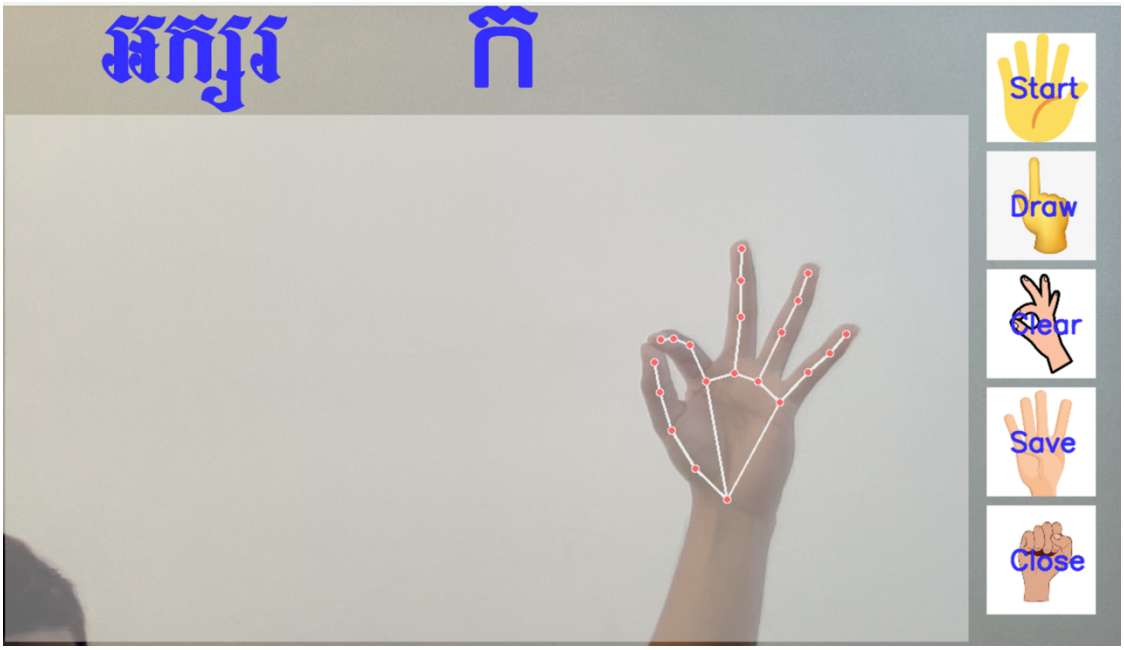


Figure 39: Clear Stroke in Prototype

- **Close System (Webcam)**

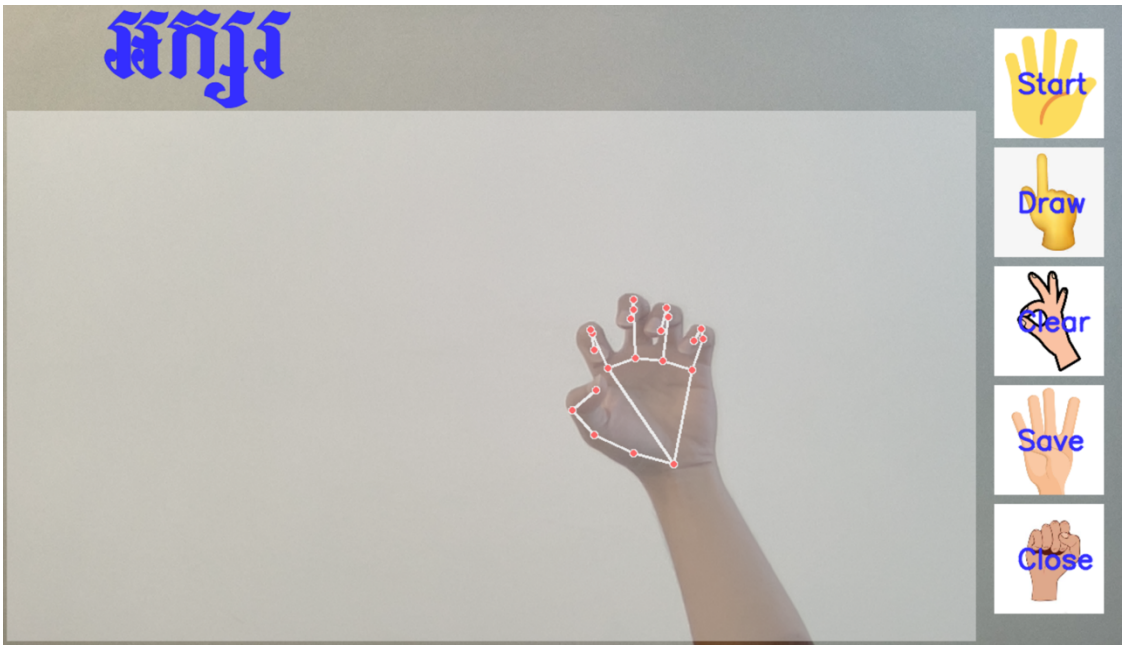


Figure 40: Close Prototype