VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

GRADUATION THESIS

USING MACHINE LEARNING METHODS IN TRANSLATING SIGN LANGUAGE INTO VIETNAMESE

COUNCIL: SOFTWARE ENGINEERING

SUPERVISOR: Assoc. Prof. Quản Thành Thơ
REVIEWER: Trần Hồng Tài, M.Sc
—o0o—
STUDENT 1: Võ Tuấn Khanh (1810220)
STUDENT 2: Nguyễn Trí Nhân (1810390)

HO CHI MINH CITY, 05/2022

Declaration Of Authenticity

We claim that this research is our work, conducted under the supervision and guidance of Associate Professor Quản Thành Thơ. The result of our research is authentic and has never been published in any platforms before. All the materials we utilized in this research were gathered from various sources and are referenced correctly in the References section.

Furthermore, within this research, we also used the results of several other authors and organizations. All of them have been correctly cited. In any case of plagiarism, we stand by my actions and will be responsible for it. Therefore, Ho Chi Minh City University of Technology is not responsible for any copyright infringements conducted within our research.

Acknowledgment

To complete the thesis outline, I would like to express my deep gratitude to Associate Professor Quản Thành Thơ for his guidance throughout the research process.

We want to express our sincere thanks to the Faculty of Computer Science and Engineering teachers, Ho Chi Minh City University of Technology, for their dedication to imparting knowledge during our years of study at the school. The knowledge accumulated during the study process is the foundation for the research process and the future professional goals.

Finally, we wish you good health and success in your noble career.

Abstract

Presently, more than two million people with deafness and difficulty in hearing in Vietnam cannot talk freely with those who do not have those symptoms. There are a few ways for the deafness and difficulty in hearing to communicate with others, but they are ineffective and inefficient. The most used method is through notes or body language to express their thoughts to others. However, not many people comprehened sign language, making it hard to understand the deaf.

To help those people communicate more efficiently, we have built a system using Artificial Intelligence to translate sign language into Vietnamese. The design, in short, contains two physical modules, which are a camera and a smartphone installed our application. Beneath the application is an artificial-intelligence-based pipeline with six more submodules that translate sign language into text and read the word out loud through the phone's built-in speaker.

Besides, the sign language data is collected from various resources through the site `https://tudienngonngukyhieu.com/` and `https://qipedc.moet.gov.vn/`. Based on the videos instructing vocabulary, we gathered and categorized them for the primary dataset of this system.

Overall, our system currently can translate some words. However, the more time it takes to learn the new terms, the more accurate the system is. The system can translate many more words with a much more extensive sign language library. We suppose our approach not only helps people with disabilities in communicating and enriching their lives, but also increases the volume of the workforce and reduces the economic burden on the national budget since then.

Contents

1 Introduction **1**

1.1 Problem statement 1

1.2 Goals of the thesis 2

1.3 Scopes of the thesis 2

1.4 Structure of the thesis proposal 3

2 Related Work **4**

3 Theoretical Background **7**

3.1 Convolution Neural Network - CNN 7

3.1.1 Architecture 8

3.1.2 Feature extraction part 9

3.1.3 Classification part 11

3.2 Media Pipe 12

3.2.1 Introduction to Media Pipe Hands 12

3.2.2 Palm detection model 14

3.2.3 Hand landmark model 14

3.3 Distance Matrix 15

3.3.1 Create Distance Matrix 16

3.4 Beam search and Connectionist Temporal Classification 16

3.4.1 Connectionist Temporal Classification 16

3.4.2 Why we want to use CTC 17

3.4.3 Beam Search with CTC decoder 18

3.5 Technology 23

3.5.1 Java 23

3.5.2 Firebase 24

4 Design and Solution **25**

4.1 Gathering Data 25

4.2 System Structure 26

4.3 Detail Implementation 28

4.3.1 Hand pattern recognition 28

4.3.2 Direction determination 29

4.3.3 Location detection 31

4.3.4 Word decoder 34

4.3.5 Text-to-speech 39

4.3.6 The camera module 40

4.3.7 App design 42

4.3.8 Use case 46

4.3.9 Use case description 47

5 Result and evaluation **52**

5.1 Result 52

5.2 Evaluation 52

List of Figures

1 Using Microsoft Kinect to translate sign language 5

2 Glove-based approach to translate sign language 5

3 Convolution Neural Network 7

4 Different between Normal Neural Network and Convolution Neural Network . . 8

5 Convolution Neural Network Architecture 8

6 Convolution Neural Network Layer 9

7 Using padding for strike one in Convolution Layer 10

8 Max Pooling and Average Pooling 10

9 Some Active Function common used in CNN 11

10 Fully connected Layer 12

11 Media Pipe real time tracking 3D hand landmarks 13

12 21 Hand Landmarks 15

13 Distance Matrix 15

14 Calculating distance between A and B 16

15 The Distance Matrix is constructed from Raw Data 16

16 Overview of a Neural Network for handwriting recognition 17

17 Annotation for each horizontal position of the image 17

18 Basic version of Beam Search 19

19 NN output and tree of beams with alphabet = "a", "b" and BW = 2 20
20 The effect of appending a character to paths ending with blank and non-blank . 20
21 CTC beam search 22
22 Java Logo 23
23 FireBase Logo 24
24 Google sheet about words labeled 26
25 Google sheet about Hand Shape can be recognized 26
26 Overview of the old system structure 27
27 Overview of the new system structure 28
28 Hand pattern recognition pipe line 29
29 Word "You" (bạn) in sign language 29
30 Word "I" (tôi) in sign language 30
31 Steps to detect the direction of the hand 30
32 Vector(0, 8) represent the hand pointing toward the left 31
33 View from the camera module 32
34 Illustration of how the ultrasonic sensor works 32
35 The ultrasonic sensor HY-SRF05 33
36 Result of location module 34
37 Map one to one data from four component with word in database and get result 35
38 Hand State which construct from pattern, location and direction 36
39 Architecture 36
40 A queue of hand state which get from three component in section 37
41 Vectorization 37
42 Beam Search 39

43 Map to dictionary and get word 39
44 The components inside the camera module prototype 41
45 Views of the camera module prototype from the above and under 42
46 Views of the camera module prototype from the sides 42
47 The landing page of the application 43
48 The main screen of the application 44
49 The profile screen 45
50 The dictionary screen 45
51 Use case diagram 46

List of Tables

1 Structure of the thesis proposal 3

2 Use case view result after predict 47

3 Use case view profile 47

4 Use case sign up 48

5 Use case sign in 49

6 Use case sign out 50

7 Use case look up word in dictionary 50

8 Mark leaned word 51

Chapter 1

Introduction

1.1 Problem statement

"Each deaf person is like a separate world, and they feel more self-deprecated and alone when they can not interact and share with others. They still have the desire to contribute to society", said Mr. Do Hoang Thai Anh, Vice Chairman of the Hanoi Deaf Association [11].

Language is a universal key that not only connects people but also builds up our society. Any disability that affects the ability to communicate is a significant disadvantage, especially for people with disabilities. They cannot integrate, have fun, learn, and communicate like ordinary people because they cannot express their thoughts, ideas, and desires to develop society as we do. That burden usually makes them fall into poverty, live a isolated life, and be exploited, apart from society. Hence, it is challenging for them to have beautiful lives.

In 2020, Vietnam had more than 2.5 million people who are deaf and mute, yet, only a tiny portion of them took part in education, had the chance to be understood, and integrated with society [5].

According to UNICEF, households with members with disabilities are often poorer, children with disabilities are at risk of having less education than their peers, and employment opportunities for people with disabilities are also lower than those without disabilities. Even though people with disabilities are beneficiaries of the policy, and poverty is not a burden to accessing health facilities, very few people with disabilities (2.3%) have access to functional rehabilitation services when being sick or injured. Besides, inequality still exists in living standards and social participation for people with disabilities. Many organizations have been founded to support, help, and create better living conditions for people with disabilities. However, this work still has many difficulties and inadequacies as there is no formal school or class. In addition, there are insufficient sign language translators while they take an essential role in helping the people with disabilities connect with society.

A quote from Cavett Robert, "Life is a grindstone, and whether it grinds you down or polishes you up is for you and you alone to decide." However, it is challenging for these people to go to school and get a good education. They have their desires and dreams, but our resources and efforts are not enough to make them a polished grindstone. Furthermore, sign language shares the same property as any other spoken language; each different region and territory has a different way of expressing sign language. These unseen differences make communication, self-expression, and information exchange even more complex and challenging for humanity.

In short, we must admit that understanding and breaking the language barrier is extremely necessary and urgent because the deaf and mute, like many other ordinary people, deserve to be assisted, understood, and acknowledged. Furthermore, we believe our system is the resolution to problems of the deaf and hard of hearing.

1.2 Goals of the thesis

the thesis aims to research, understand, and implement solutions to converting sign language into Vietnamese. In particular, the system must receive a queue of images from the camera mounted on the hat, use the implemented algorithms to process and display text on the phone screen.

We can solve the above problem by breaking it into smaller ones listed below. With each issue, we will give our solution and architect a system that can solve the whole problem of the thesis.

- Search and collect data on sign language, conduct evaluation, classification, and normalization of data
- Find out the approaches that have been implemented
- Design architecture of the model
- Plan to implement, develop a sign language conversion system
- Build an application that users can utilize

1.3 Scopes of the thesis

In this case study, we will build a system including an app and camera module to translate sign language into Vietnamese. Because of the limited time, the scope of the study is also limited as follows:

- The system can only translate Vietnamese words
- The system can only recognize the words trained with before

1.4 Structure of the thesis proposal

This proposal includes four sections and each will convey the related works and output when doing the thesis.

Chapter	Content
1	A brief introduction about plan and objectives of thesis
2	Related works that had been done and how they has helped us in doing the thesis
3	Introduction of theoretical background as foundation knowledge that are applied in the thesis
4	Solution and design approach for problem statement of the thesis
5	Result and evaluation for the thesis
6	Summary of the thesis

Table 1: Structure of the thesis proposal

Chapter 2

Related Work

Nowadays, many scholars worldwide have submitted research projects relating to turning sign language into text, using a variety of methodologies and perspectives.

Two main approaches have been proposed:

- **Glove-based approaches:** This approach requires the deaf and mute to wear a sensor glove. When users have any different actions or gestures, the gloves will record all those movements. After that, data from the sensor will be analyzed by the analyzer component. Finally, that component returns the output to the users.
- **Vision-based approaches:** With this approach, developers will apply image processing algorithms to determine hand positions, gestures, and movements. The user will not have to wear necessary glove-based methods, which is convenient. However, using image processing algorithms, we need to deal with the worst quality output affected by these algorithms.

Specifically, about the vision-based approach, the earlier method used several image processing algorithms to build feature vectors based on a single RGB image of the hand. In this paper, "Real-time sign language recognition using a consumer depth camera" [6], using multi-layered random forest (MLRF) not only allows them to recognize hand signs correctly but also minimizes training time and effort. Alternatively, in this paper, "Sign Language Translation in Urdu/Hindi Through Microsoft Kinect," sign language can be recognized by auxiliary equipment: Microsoft Kinect (see figure 1), which captures the signs of the deaf person, after that, through the computer system, they can detect what deaf people say.

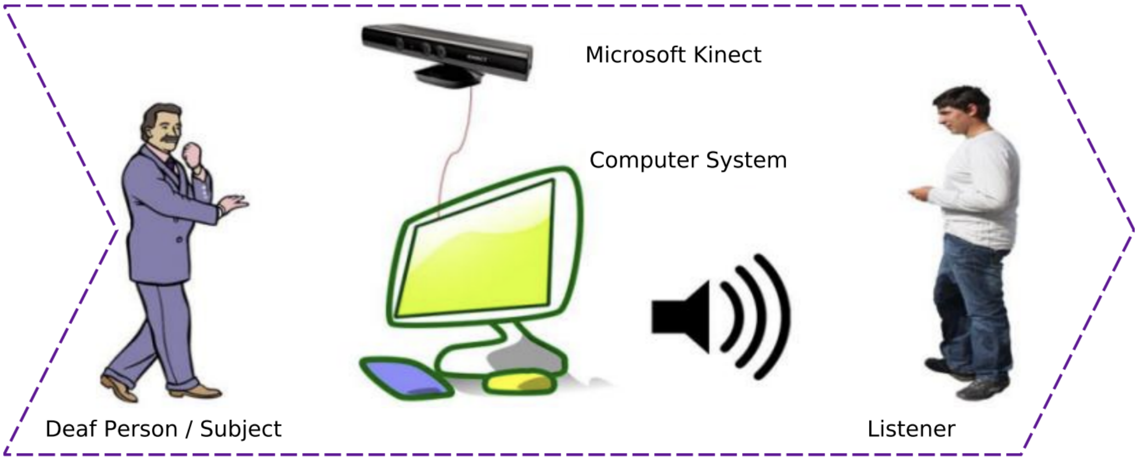


Figure 1: Using Microsoft Kinect to translate sign language

Besides the vision-based approach, glove-based system has a much relevant research. This paper, "The Language of Glove: Wireless gesture decoder with low-power and stretchable hybrid electronics" [9] introduces the way to convert American Sign Language (ASL) alphabet into text and display it on a computer or smartphone (see figure 2). They can detect which hand gesture is performed with sensor gloves and send the result to the smartphone via Bluetooth. The sign language interprets text and displays it on the digital display screen. This approach is helpful in the real world for the deaf and mute who cannot communicate with ordinary people.

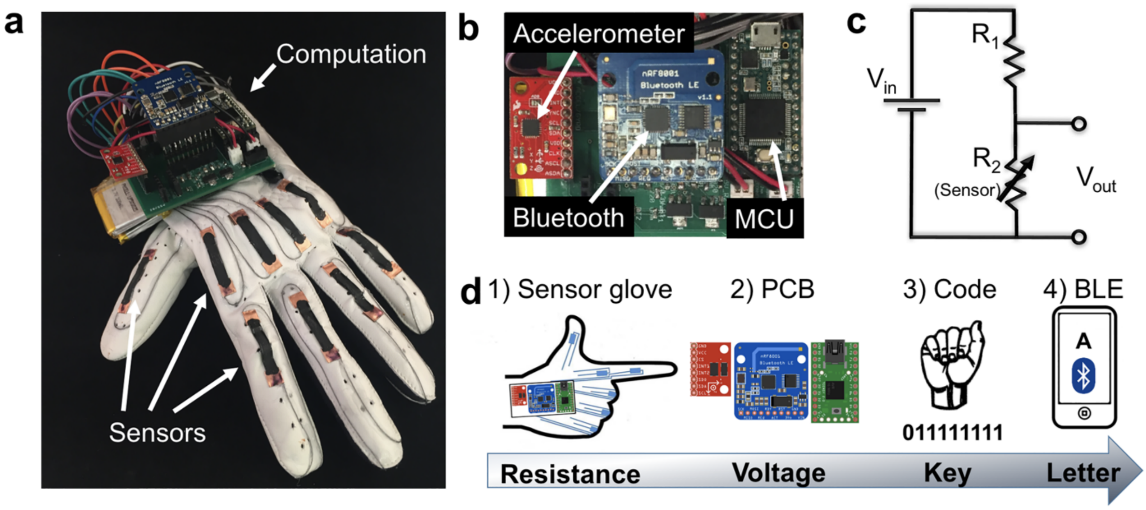


Figure 2: Glove-based approach to translate sign language

Both approaches above have some problems; they can only recognize a minimal number of words, like an alphabet, number, or some word with easy hand shape and no motion. However, it is not easy; many words will use the same hand shape but differ in many characteristics, such as positions and directions. There is currently no model that can handle the conversion of sign language flexibly and conveniently for the deaf and mute, helping them communicate effectively and naturally. Therefore, thanks to applying appropriate technologies, the authors carry out this graduation thesis to break down the barriers between deaf-and-mute people and ordinary people, helping them become self-sufficient and more confident in daily communication.

Chapter 3

Theoretical Background

3.1 Convolution Neural Network - CNN

Convolution Neural Networks are a particular class of Neural Networks [7]. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product, and optionally follows it with a non-linearity. CNN mainly consists of Convolution Layers, Pooling Layers, Activation Layers, and Fully Connected Layers. ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode specific properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network. Some of the primary uses of CNN can be mentioned as image classification, object detection, semantic segmentation, face recognition, ...

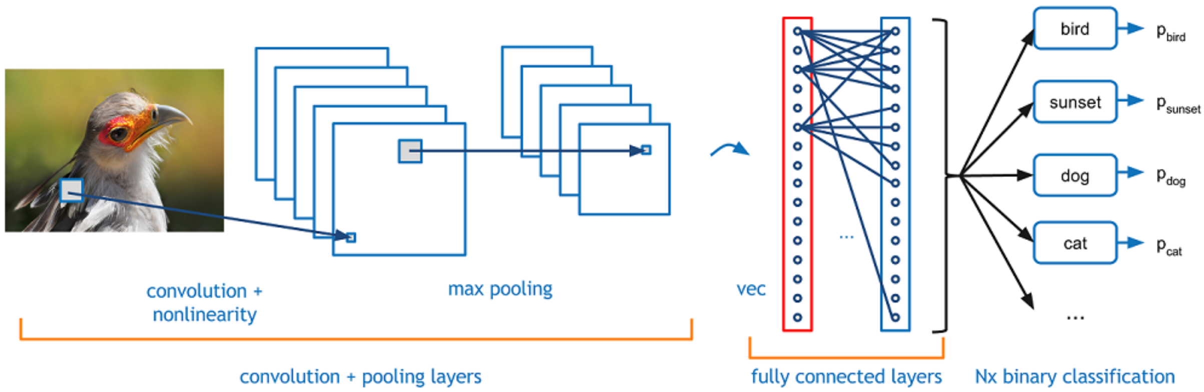


Figure 3: Convolution Neural Network

The figure 3 above shows an example of a convolution neural network, which is taking an image as input and then extracting features from it through various layers and then finally predicting the class of the object in the given image.

3.1.1 Architecture

Convolution Neural Networks have a different architecture with regular Neural Networks, and we can see this difference in figure 4 below. Regular Neural Networks transform an input through a series of hidden layers. Every layer comprises a set of neurons, where each layer is fully connected to all neurons in the previous layers. Finally, a last fully-connected output layer represents the predictions with CNN architecture. First of all, the layers are organized into three dimensions: width, height, and depth. Further, the neurons in one layer do not connect to all neurons in the next layer but only to a small region. Lastly, the system will reduce the final output to a single vector of probability scores, organized along the depth dimension.

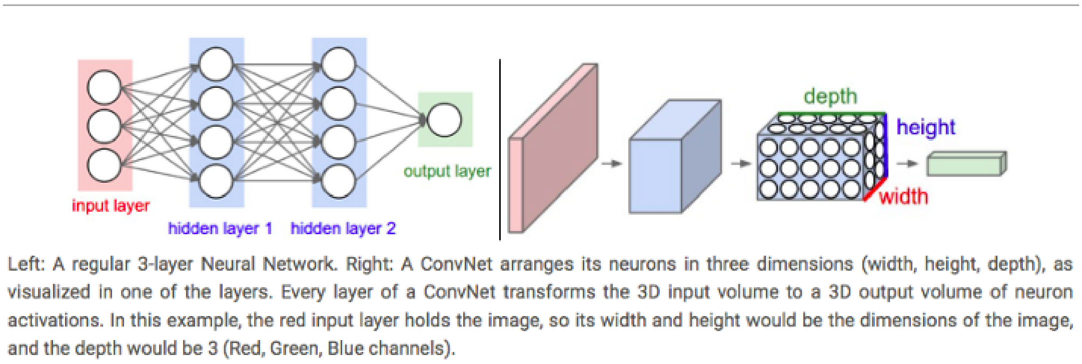


Figure 4: Different between Normal Neural Network and Convolution Neural Network

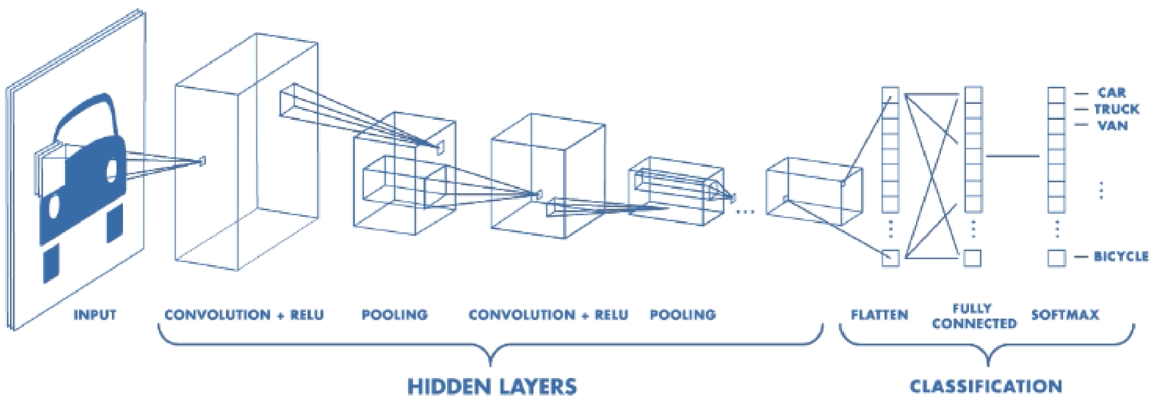


Figure 5: Convolution Neural Network Architecture

As we can see in figure 5, CNN can be divided into two parts:

1. The hidden layers/ Feature extraction part
   In this part, the network will perform a series of convolutions and pooling operations while the features are detected. Imagine you have a picture of a zebra, these are the parts where the network would recognize: its stripes, two ears, and four legs.
2. The Classification part
   The fully connected layers serve as a classifier on top of their extracted features. By using a provided algorithm, they will assign a probability for the objects on the image.

3.1.2 Feature extraction part

Convolutional Layer

The convolution layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. A convolution is executed by sliding the filter over the input. At every location, matrix multiplication is performed and it sums the result onto the feature map. This extracting features from images happen throughout the CNN's convolutional layers. This process is illustrated in figure 6

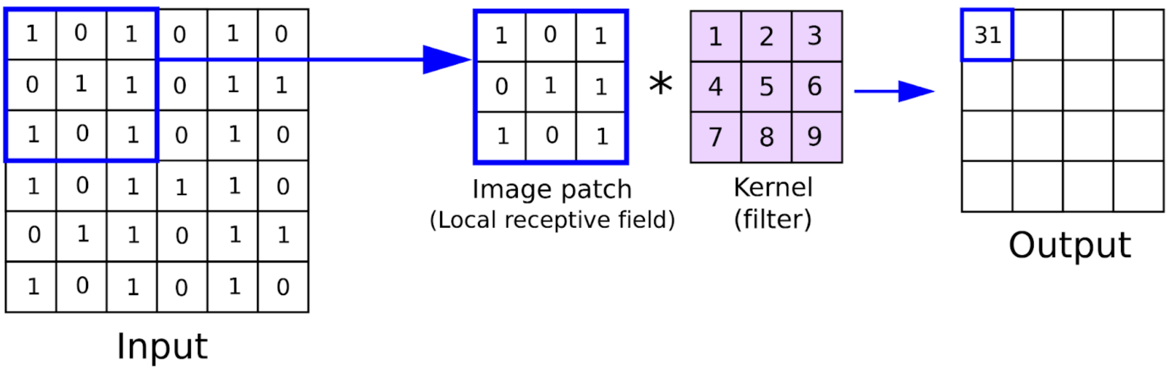


Figure 6: Convolution Neural Network Layer

When the feature map is made, we can pass each value in the feature map through a non-linearity function, such as ReLU, sigmoid before it becomes the input of the next convolution layer.

Because the size of the feature map is always smaller than the input, we have to do something to prevent our feature map from shrinking. This is where we use padding (7). A layer of zero-value pixels is added to surround the input with zeros so that our feature map will not shrink. In addition to keeping the spatial size constant after performing convolution, padding also improves performance, ensures the Kernel and strides size will fit in the input.

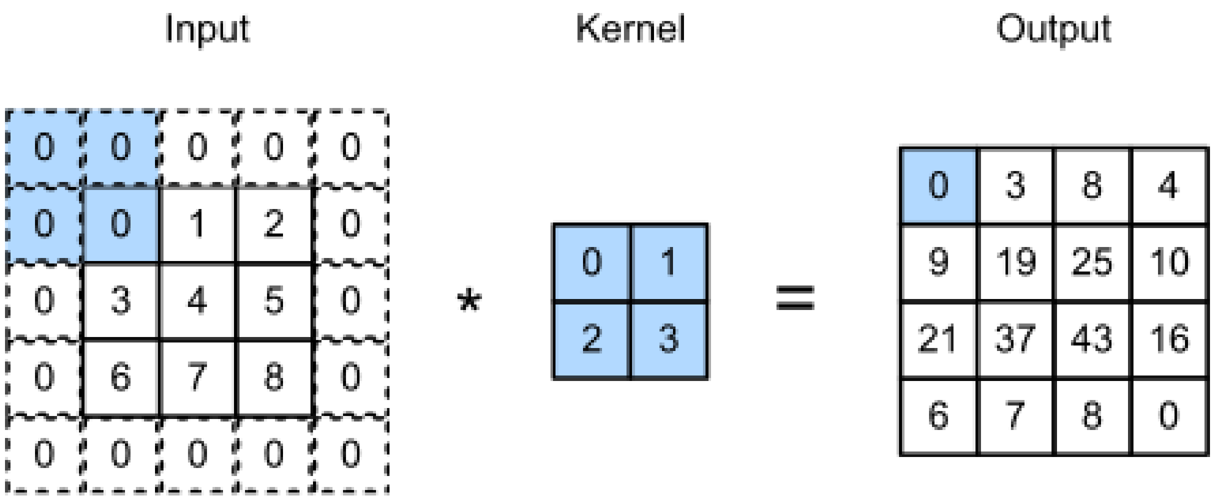


Figure 7: Using padding for strike one in Convolution Layer

Pooling Layers

After a convolution layer, it is common to add a pooling layer in between CNN layers. The function of Pooling is to continuously reduce the dimensionality to reduce the number of parameters and computation in the network. This action shortens the training time and controls overfitting.

There are two main types of Pooling Layers in a CNN: Max Pooling and Average Pooling. The functionality of these two types of layers is demonstrated in figure 8. Max Pooling restores the maximum value from the picture segment covered by the Kernel. Average Pooling converts the average values from the bit of the picture surrounded by the Kernel.

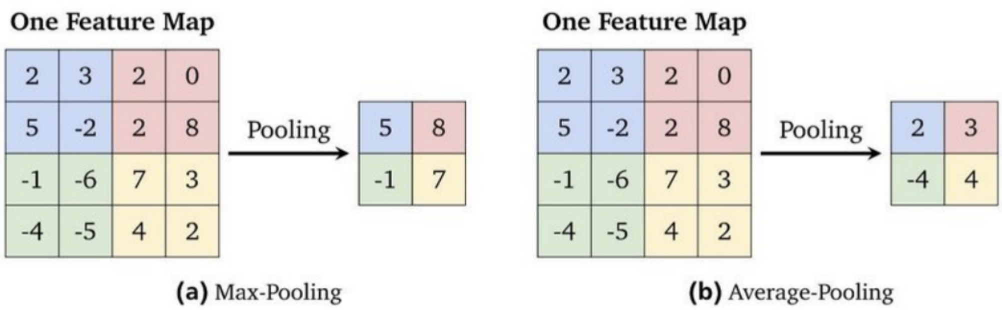


Figure 8: Max Pooling and Average Pooling

Activation Layers

In general, Neural networks and CNNs rely on a non-linear "trigger" function to signal distinct identification of likely features on each hidden layer. CNN may use a variety of specific functions (figure 9), such as rectified linear units (ReLUs) and continuous trigger (non-linear) functions—to efficiently implement this non-linear triggering.

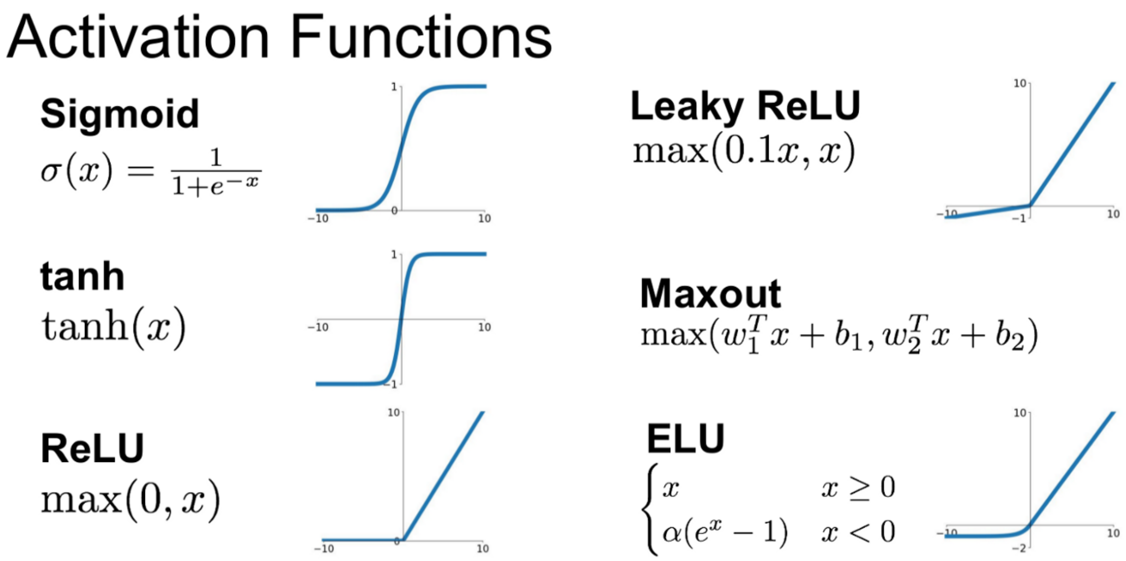


Figure 9: Some Active Function common used in CNN

3.1.3 Classification part

Fully connected layers

The last layers of a CNN are fully connected. Neurons in a fully connected layer have complete connections to all the activations in the previous layer. This part is, in principle, the same as a regular Neural Network.

Figure 10 illustrates the way of input value stream into the fully connected layer. Because these fully connected layers can only accept one-dimensional data, we need to convert our 3D data to 1D data. After passing through some FC, we will get the data classification result.

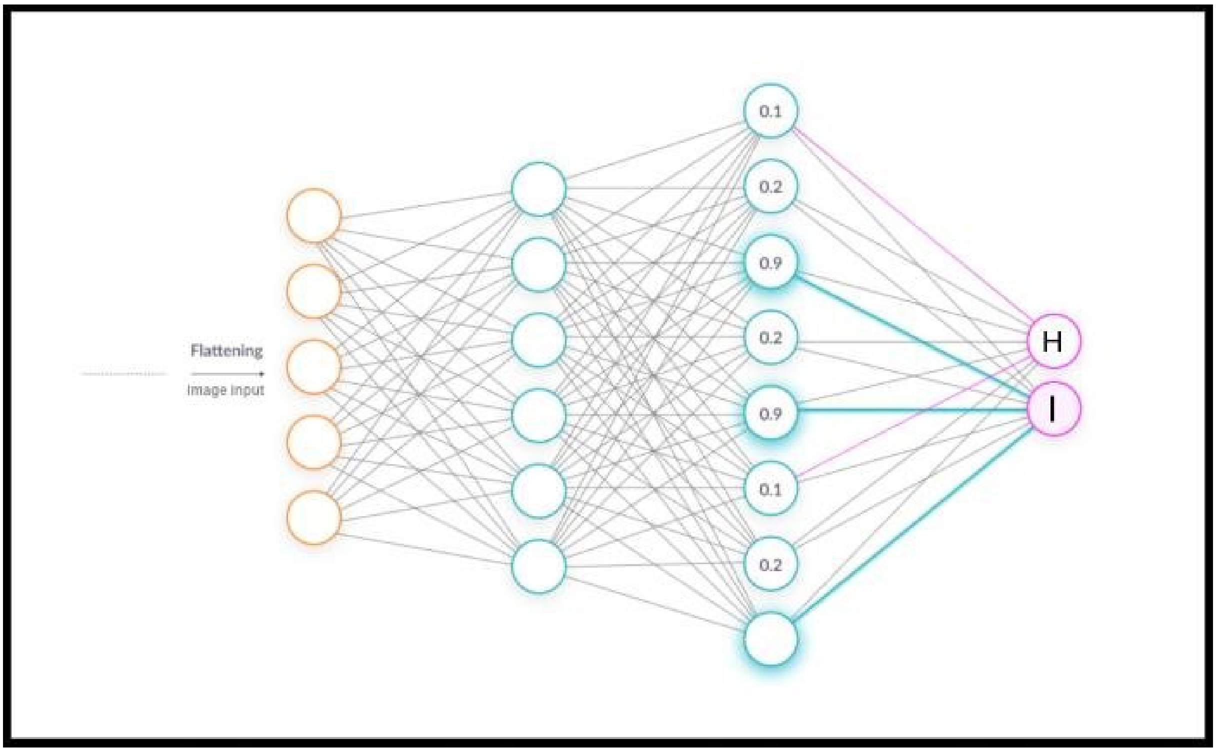


Figure 10: Fully connected Layer

3.2 Media Pipe

3.2.1 Introduction to Media Pipe Hands

MediaPipe Hands (11) is a high-resolution tracking system for hands and fingers [13]. It uses machine learning to infer 21 3D hand landmarks from a single frame. This solution delivers real-time performance on a cell phone and even scales to many hands, whereas current state-of-the-art systems rely primarily on powerful desktop environments for inference.

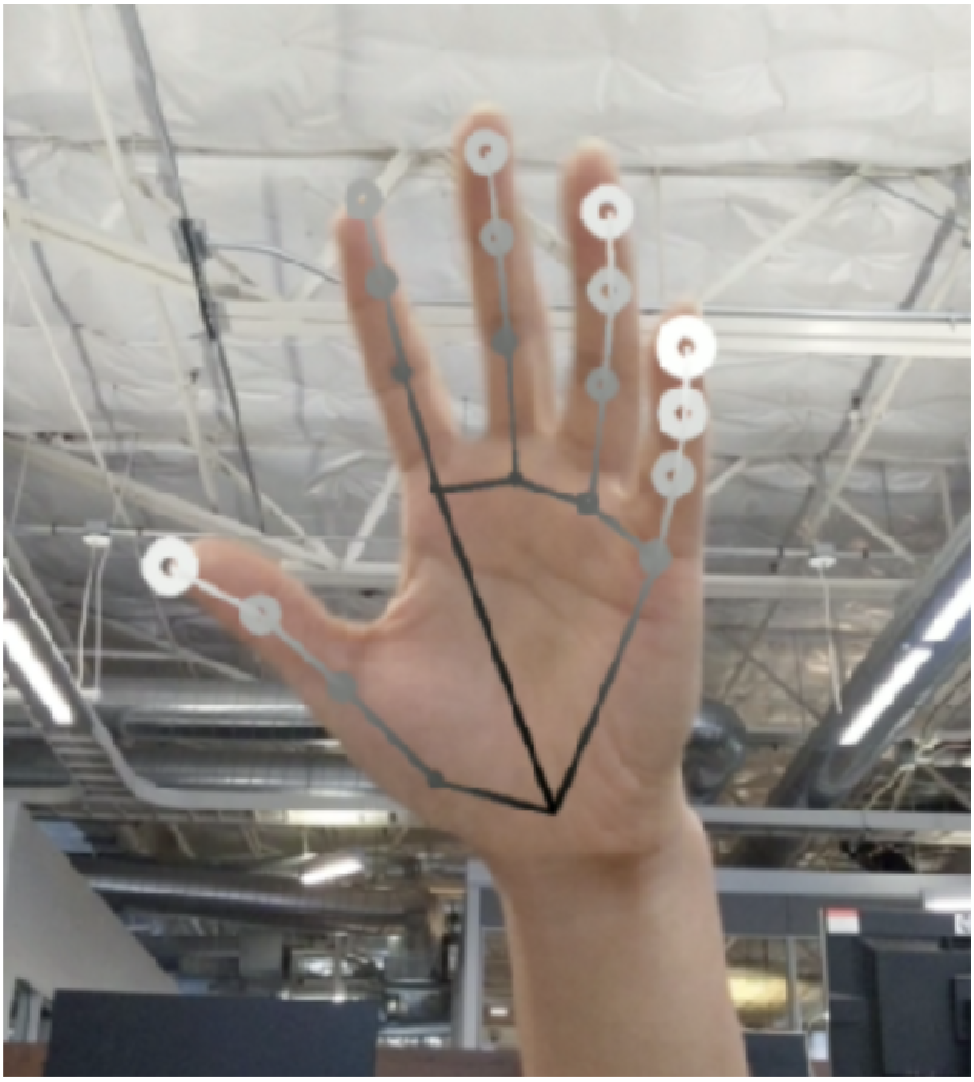


Figure 11: Media Pipe real time tracking 3D hand landmarks

MediaPipe Hands makes use of a machine learning pipeline that consists of several models that work together: A palm detection model, which acts on the entire image, will return an orientated hand bounding box. A hand landmark model that returns high-fidelity 3D hand key points from the cropped image region determined by the palm detector.

However, providing the hand landmark model with a correctly cropped hand image minimizes the requirement for data augmentation drastically (such as rotations, translations, and scaling) and instead, allows the network to focus on coordinate prediction accuracy. Furthermore, in this ML pipeline, crops can be created based on the hand landmarks recognized in the previous frame, and palm detection is only used to localize the hand when the landmark model can no longer detect its presence.

3.2.2 Palm detection model

The Media Pipe team provides the palm detection model to detect initial hand locations and distinguish whether the hand recognized is left or right, which is very useful as each sign goes along with a different side will result in different meanings. They created a single-shot detector model, comparable to the face detection model in MediaPipe Face Mesh [8], tailored for mobile real-time applications. Hand detection is difficult: our model must detect occluded and self-occluded hands and work across many hand sizes with a significant scale span relative to the image frame.

According to their statements, the methods they used to address the above challenges vary in many strategies. First, instead of training a hand detector, they trained a palm detector because estimating bounding boxes of inflexible objects like palms and fists was much easier than recognizing hands with articulated fingers. Furthermore, the non-maximum suppression method performs effectively even in two-hand self-occlusion situations such as handshakes because palms are small objects. Furthermore, palms can be simulated using square bounding boxes (anchors in ML language) that ignore other aspect ratios, reducing 3-5 anchors. Second, even for tiny objects, an encoder-decoder feature extractor is used for more extensive picture context awareness (similar to the Retina Net approach). Finally, the significant scale variance limits focus loss during training to support many anchors.

Using the strategies described above gives an average precision of 95.7 percent in palm detection. With no decoder and a regular cross-entropy loss, the baseline is just 86.22 percent.

3.2.3 Hand landmark model

Following palm detection over the entire image, our next hand landmark model uses regression to accomplish exact key point localization of 21 3D hand-knuckle coordinates (see figure 12) within the detected hand regions, i.e., direct, coordinate prediction. Even with partially visible hands and self-occlusions, the model develops a consistent internal hand posture representation.

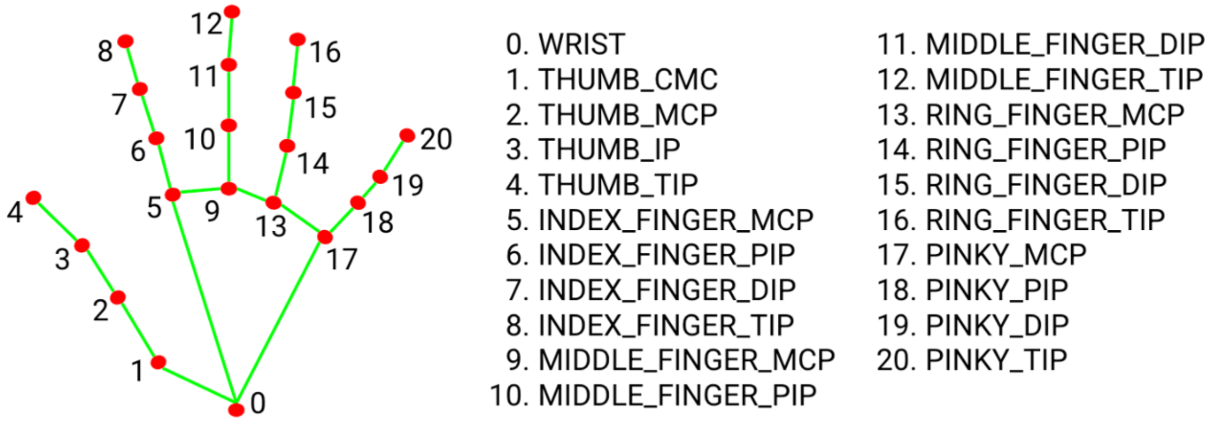


Figure 12: 21 Hand Landmarks

3.3 Distance Matrix

A distance matrix [1] is a table that shows the distance between pairs of objects. For example, in the figure 13., we can see the length between A and B is 16, B and C is 37, and so on. The diagonal of the table is the distance to the object from itself, so the value, as we can see, is 0. Distance matrices are sometimes called dissimilarity matrices.

Distance Matrix

	A	B	C	D	E	F
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

Figure 13: Distance Matrix

3.3.1 Create Distance Matrix

A distance matrix is computed from a raw data table. In the example below (figure 14), we can use high school math (Pythagoras) to work out the distance between A and B.

$$\sqrt{(24-9)^2+(54-49)^2} = 15.81 \approx 16$$

Figure 14: Calculating distance between A and B

We can use the same formula with more than two variables, known as the Euclidean distance. As a result, we have the distance matrix represented like figure 15.

Raw Data

	X	Y
A	9	49
B	24	54
C	51	28
D	81	54
E	81	23
F	86	32

Distance Matrix

	A	B	C	D	E	F
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

Figure 15: The Distance Matrix is constructed from Raw Data

3.4 Beam search and Connectionist Temporal Classification

3.4.1 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) [4] is a type of Neural Network output helpful in tackling sequence problems like handwriting (figure 16) and speech recognition where the timing varies. Using CTC ensures that one does not need an aligned dataset, which makes the training process more straightforward.

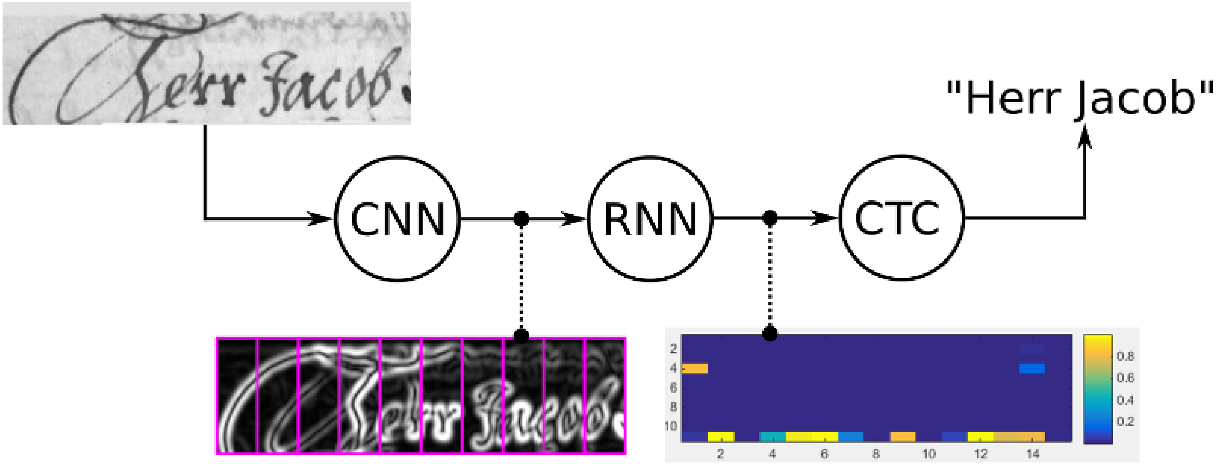


Figure 16: Overview of a Neural Network for handwriting recognition

3.4.2 Why we want to use CTC

In the context of handwritten recognition, we could create a dataset with images of text-lines, and then specify for each horizontal position of the image the corresponding character as shown in figure 17 Then, we could train a model to output a character-score for each horizontal position. However, there are two problems with this solution.

- It takes much time, and annotating the dataset at the character level is tiresome.
- What if the character takes up more than one time-step ?. We could get "tooo" because the "o" is a wide-character as shown in figure 17. We must remove all duplicate characters like "t" and "o".

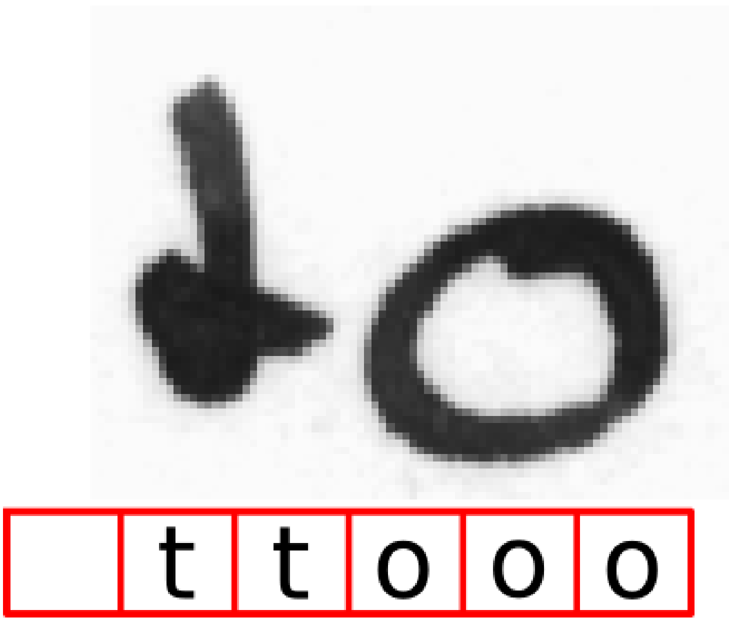


Figure 17: Annotation for each horizontal position of the image

CTC can solve both problems for us:

- We can ignore both the position and width of the character in the image and only requires the text that occurs in the picture.
- Using decode techniques, we can directly get the result of the network, and no further post-processing of the recognized text is needed.

3.4.3 Beam Search with CTC decoder

CTC has more than the Decoding phase, it can have the Encoding, Loss calculation, but we don't need it in the thesis scope anymore. So, here, we only mention to CTC decoder, but in the way, it combines with Beam Search [10]. Because CTC in decoding context can connect with another algorithm like best-path decoding, ...

Beam search

In computer science, beam search [2] is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set. Beam search is an optimization of best-first search the reduces its memory requirements. Best-first search is a graph search that orders all partial solutions (states) according to some heuristic. But in beam search, only a predetermined number of the best partial solutions are kept as candidates. Pseudocode for the basic version of beam-search is shown in figure 18

```
Data: NN output matrix mat, BW
Result: decoded text
1  beams = {∅};
2  scores(∅, 0) = 1;
3  for t = 1...T do
4      bestBeams = bestBeams(beams, BW);
5      beams={};
6      for b ∈ bestBeams do
7          beams = beams ∪ b;
8          scores(b, t) = calcScore(mat, b, t);
9          for c ∈ alphabet do
10             b′ = b + c;
11             scores(b′, t) = calcScore(mat, b′, t);
12             beams = beams ∪ b′;
13         end
14     end
15 end
16 return bestBeams(beams, 1);
```

Figure 18: Basic version of Beam Search

The beam search algorithm will be implemented through the following steps, with two parameters will be included: output matrix and beam width (BW), which specifies the number of beams to keep. First, the beam list and corresponding score are initialized (lines 1 and 2). After that, from 3-15, the algorithm will loop over all time-steps of the matrix output. At this point, only the best scoring beams (equal BW) from the previous time-step are kept (line 4). For each beam, we calculate the score and get a result (line 8); we will cover this step in more details later. Further, each beam is extended by all possible characters from the alphabet (line 10), and again, a score is calculated (line 11). After the last time-step, the best beams are returned (line 16).

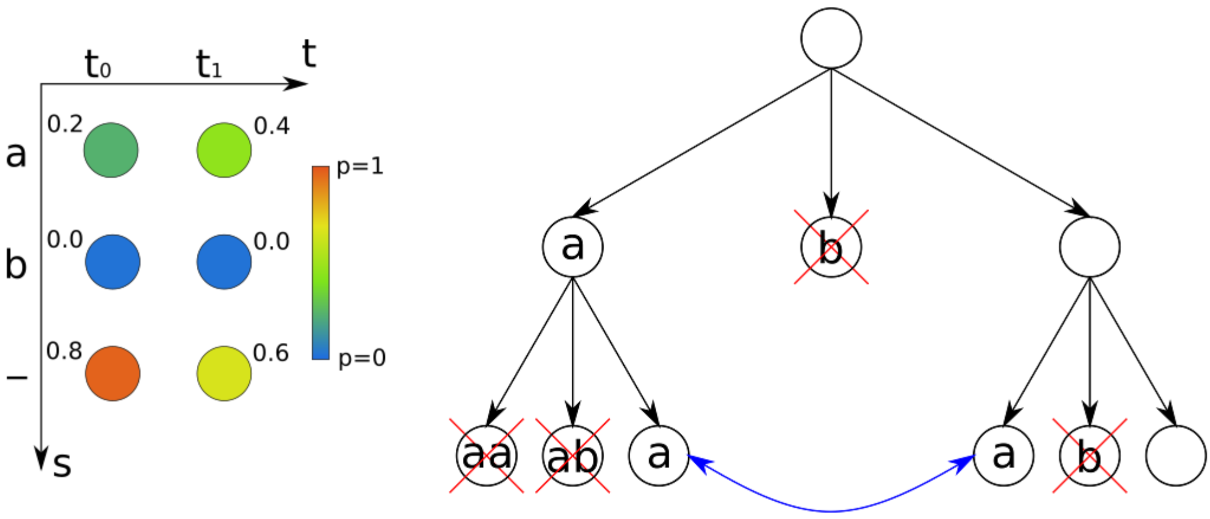


Figure 19: NN output and tree of beams with alphabet = "a", "b" and BW = 2

As we can see, in figure 19, the output matrices are decoded, and the tree of beams is shown. Beam search algorithm extended as possible and keep exactly BW candidates. Finally, we finished the last iteration, and the final step of the algorithm is to return the beam with the highest score, which is "a" in this example.

Calculating the score

As discussed above, in this part, we will talk about how to score the beam. We will split the beam-score into the score of paths ending with a blank(e.g. 'aa-') and paths ending with non-blank (e.g. 'aaa').

- We denote the probability of all paths ending with a blank and corresponding to a beam b at time-step t by $P_b(b,t)$ and by $P_{nb}(b,t)$ for the non-blank case.
- The probability $P_{tot}(b,t)$ of a beam b at time-step t is simply the sum of P_b and P_{nb}, for example: $P_{tot}(b,t) = P_b(b,t) + P_{nb}(b,t)$

$$\textbf{blank: } \text{'aa-'} + \begin{cases} \text{'-'} = \text{'aa--'} \rightarrow \text{"a" (copy)} \\ \text{'a'} = \text{'aa-a'} \rightarrow \text{"aa" (extend)} \\ \text{'b'} = \text{'aa-b'} \rightarrow \text{"ab" (extend)} \end{cases}$$

$$\textbf{non-blank: } \text{'aaa'} + \begin{cases} \text{'-'} = \text{'aaa-'} \rightarrow \text{"a" (copy)} \\ \text{'a'} = \text{'aaaa'} \rightarrow \text{"a" (copy)} \\ \text{'b'} = \text{'aaab'} \rightarrow \text{"ab" (extend)} \end{cases}$$

Figure 20: The effect of appending a character to paths ending with blank and non-blank

In figure 20, we will see what happens when we extend a path. Three main cases we can mention is:

- Extended by blank ('a' + '-' = 'a-')
- Extended by repeating last character ('aa' + 'a' = 'aaa' or 'aa-' + 'a' = 'aa-a')
- Extended by some other character ('aa' + 'b' = 'aab')

And when we collapse the extended paths, two results we will get and some cases we needed to handle:

- The unchanged (copied) beam ('a' $\rightarrow$ 'a'):
  - To copy a beam, we can extend corresponding paths by a blank and get paths ending with a blank: $P_b(n,t)+=P_{tot}(b,t-1)*mat(blank,t)$
  - Besides, with the non-blank ending paths case, if we extend it by the last character (the beam is not empty): $P_{nb}(b,t)+=P_{nb}(b,t-1)*mat(b[-1],t)$ with -1 indexes the last character in the beam
- An extended beam ('a' $\rightarrow$ 'aa' or 'ab'):
  - To extend a beam. With the last character is different from the character we need to extend, then there is no need for separating blanks ('-') in the paths: $P_{nb}(b+c,t)+=P_{tot}(b,t-1)*mat(c,t)$
  - Or the last character of beam is repeated, we must ensure that the paths end with a blank: $P_{nb}(b+c,t)+=P_b(b,t-1)*mat(c,t)$
  - We don't need to care about $P_b(b+c,t)$ because we added a non-blank character

Putting it all together

The CTC beam search algorithm is depicted in the figure 21. It is similar to the basic version that was previously shown. It does, however, contain the code for scoring the beams: copied beams (lines 7–10) and extended beams (lines 15–19). Finally, in order to find the best scoring beams, the program rates them using the Ptot (line 4) and then selects the best beams (BW).

Data: NN output matrix mat, BW and LM

Result: decoded text

```
beams = {∅};
P_b(∅, 0) = 1;
for t = 1...T do
    bestBeams = bestBeams(beams, BW);
    beams={};
    for b ∈ bestBeams do
        if b != ∅ then
            P_nb(b, t) += P_nb(b, t − 1) · mat(b(−1), t);
        end
        P_b(b, t) += P_tot(b, t − 1) · mat(blank, t);
        beams = beams ∪ b;
        for c ∈ alphabet do
            b′ = b + c;
            P_txt(b′) = applyLM(LM, b, c);
            if b(t) == c then
                P_nb(b′, t) += P_b(b, t − 1) · mat(c, t);
            else
                P_nb(b′, t) += P_tot(b, t − 1) · mat(c, t);
            end
            beams = beams ∪ b′;
        end
    end
end
return bestBeams(beams, 1);
```

Figure 21: CTC beam search

3.5 Technology

Overall, the product app has been developed with technologies such as Java for Android apps and Firebase system authentication. Furthermore, to keep the application as light as possible, we use only native components and no external UI libraries.

3.5.1 Java

Figure 22: Java Logo

James Gosling developed Java at Sun Microsystems, Inc. in 1995, later acquired by Oracle Corporation. It's a straightforward programming language. Java makes programming easy to write, compile, and debug. It aids in the development of reusable code and modular programs.

Java is an object-oriented programming language based on classes that are designed to have as few implementation dependencies as feasible. Furthermore, Java is a general-purpose programming language that allows developers to write code once and run it on any device that supports Java. Java programs are compiled into byte code that may be run on any Java Virtual Machine. Besides, Java has a syntax that is similar to C and C++.

3.5.2 Firebase

Figure 23: FireBase Logo

Firebase is a web and mobile app development platform with simple and powerful APIs that don't require a server or backend. Firebase is a cloud-based platform. Also present is Google's server system. Its main goal is to make database operations simpler for users so they can program apps more easily. The real-time database service allows users to store and synchronize data. This is a completely cloud-based service. If the device is offline, it will use up its memory before automatically syncing with the server once it is online.

This feature largely comprises of backend services that help developers construct and manage their apps more effectively. The following services are included in this feature:

- **Realtime database:** The Firebase Realtime Database is a cloud-based NoSQL database that processes data at millisecond speed. In the simplest sense, it can be thought of as a large JSON file.
- **Cloud firestore:** The Cloud Firestore is a NoSQL document database that allows users to store, sync, and query data from anywhere in the world using the app. It stores information in the form of documents, which are objects. It stores any data type, including text, binary data, and even JSON trees, using a key-value pair.
- **Authentication:** Using UI libraries and SDKs, the Firebase Authentication service makes it simple to authenticate users in the app. It reduces the time and effort required to develop and maintain the user authentication service. It even handles processes like account mergers, which can take a long time if done manually.
- **Remote configuration:** The remote configuration service speeds up the distribution of updates to users. Changes could range from updating UI components to altering application functionality. These are frequently used to deliver limited-time offers and content to a mobile application.
- **Hosting:** Firebase provides fast and secure application hosting. It can host both static and dynamic websites, as well as microservices. With a single command, it can host an application.

Chapter 4

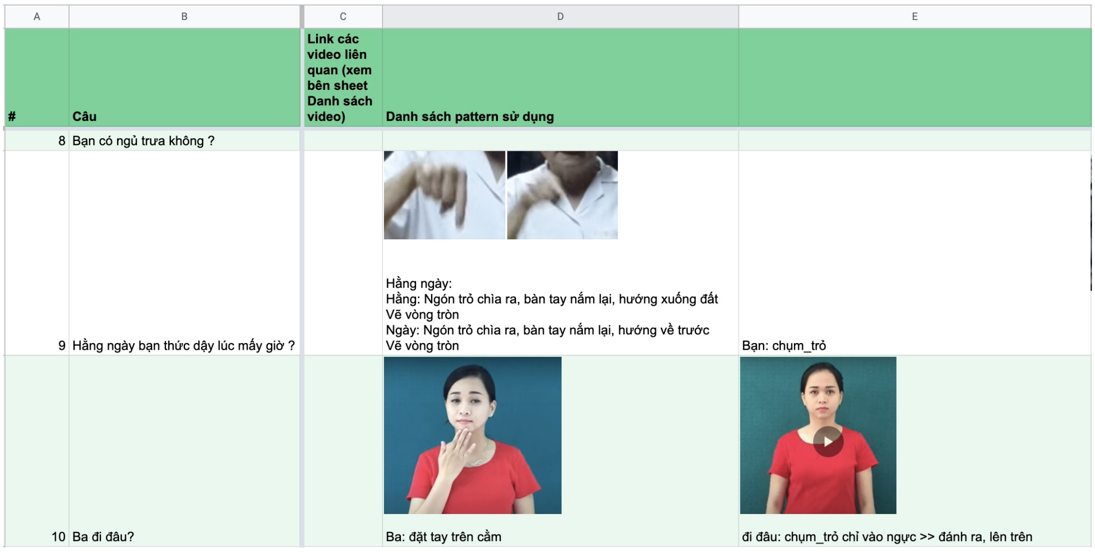
Design and Solution

4.1 Gathering Data

Before designing a system that can translate sign language, we must know what makes a word in sign language and where to collect the data. After a moment of observing the sign language on the website `https://tudienngonngukyhieu.com/`, we found an interesting point that some of the words tend to have the same pattern. Moreover, the sign's meaning depends on the direction and location that the sign has. Furthermore, some of the terms need movements of the hands or fingers to represent the meaning. Hence, we can somehow convert a word from sign language into Vietnamese with those four factors.

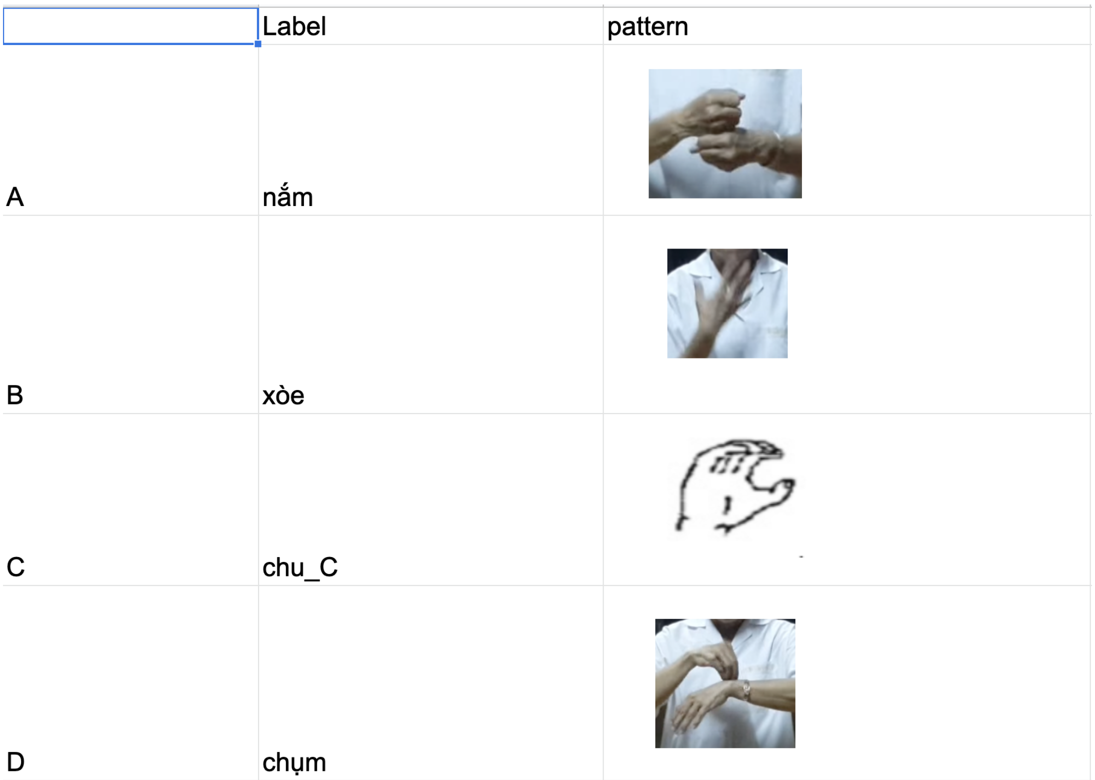
Nevertheless, first, we need to collect the sign language data for the model training phase in the thesis. Fortunately, the site https://tudienngonngukyhieu.com/ is considered the library with enough words in sign language that we need. Moreover, we learned some of the words from videos on youtube, taught by Mrs. Le Thi Thu Xuong, and channel CDS, a center for the Deaf in central Vietnam. Hence, we can train and test our system on our own.

To prepare data, we collect many words from the website and youtube channel. Then, we label and separate it into many elements, which we will discuss later in the section about hand state (4.3.4). We made the google sheet for the data we gathered. In this file, we have prepared many words that were labeled (see figure 24). Besides, we have a sheet for many handshapes (see in figure 25), which help us to classify hand patterns.



#	Câu	Link các video liên quan (xem bên sheet Danh sách video)	Danh sách pattern sử dụng	
8	Bạn có ngủ trưa không ?			
9	Hằng ngày bạn thức dậy lúc mấy giờ ?		Hằng ngày: Hằng: Ngón trỏ chìa ra, bàn tay nắm lại, hướng xuống đất Vẽ vòng tròn Ngày: Ngón trỏ chìa ra, bàn tay nắm lại, hướng về trước Vẽ vòng tròn	Bạn: chụm_trỏ
10	Ba đi đâu?		Ba: đặt tay trên cằm	đi đâu: chụm_trỏ chỉ vào ngực >> đánh ra, lên trên

Figure 24: Google sheet about words labeled



	Label	pattern
A	nắm	
B	xòe	
C	chu_C	
D	chụm	

Figure 25: Google sheet about Hand Shape can be recognized

4.2 System Structure

Overall, the system includes three parts of hardware modules: a camera module, the user's smartphone, and the server. Among those modules, the crucial one that handles the most complicated work is the server, which we will focus on in the thesis.

Our sign language translating artificial intelligence system includes six main modules: hand pattern recognition, direction determination, location detection, action detection, word decoder, and text to speech (figure 26). Firstly, the system continuously captures the hand's motion, processes it with the hand landmark model, and then puts it into those modules. Each of them has a unique role. After combining the first four modules' results (hand pattern, direction, location, and action detection), the word decoder module will take the output data and produce the corresponding outcome. Then, the result will show up on the main screen; meanwhile, the phone will speak out that word.

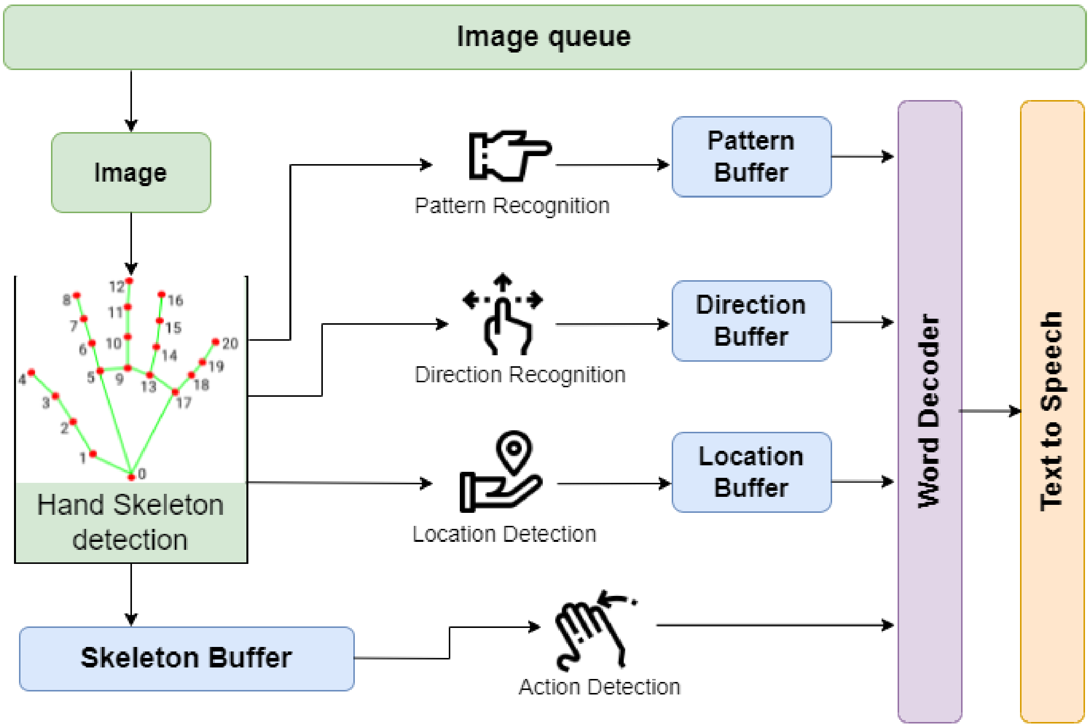


Figure 26: Overview of the old system structure

Those six main modules mentioned above are the ones we planned up at the beginning of the thesis. However, we found it hard to build the action detection module during the implementing period, despite going through most of the modules. It is problematic due to its demands on the smartphone and server.

There are words in sign language that contain many continuously moving patterns. There are a few solutions to detect which action the hands are doing; the first way is to send the whole video the camera captured to the server to process. This way, however, requires a strong connection between the camera module, smartphone, and the server and puts stress on the physical devices (the camera and smartphone); as a result, those devices will get hot quickly and can be damaged somehow. Another way is to increase the frame rate to get the action, but this one can also stress those devices; Moreover, we must have an algorithm continuously processing and detecting the movements, which, we admit, is hard to achieve.

Therefore, we had to deprecate that module and change our method to get the correct Vietnamese word to resolve that problem. Instead of combining the four modules, including the action detection module, it now only has three left: pattern, direction, and location. Furthermore, in the word decoder module, we apply a heuristic search algorithm known as beam search, which uses the result of the three modules to look up the word in the database and return it to us. We will discuss each module's role and how it works in section 4.3.

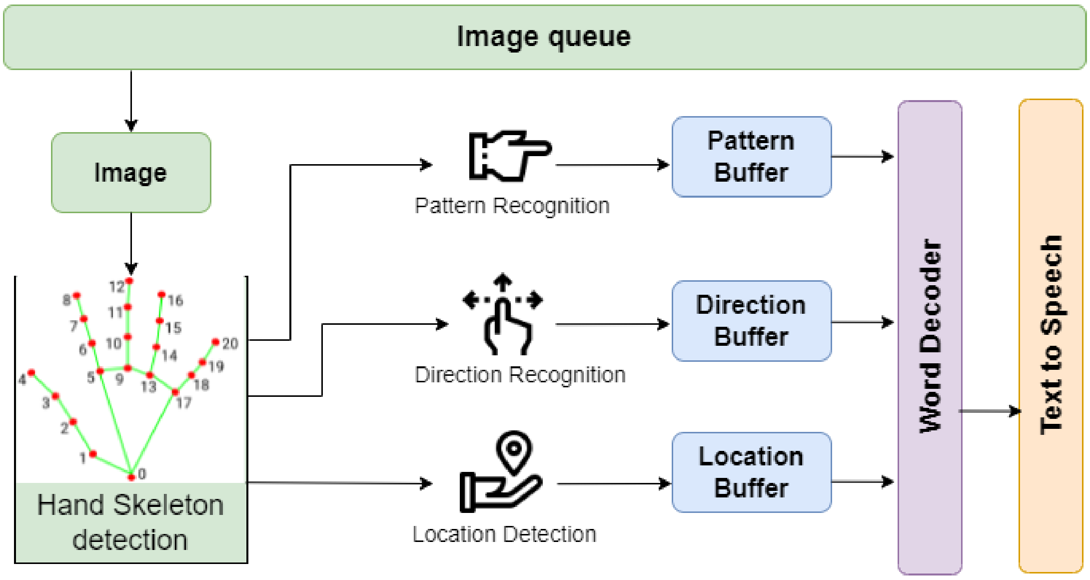


Figure 27: Overview of the new system structure

4.3 Detail Implementation

4.3.1 Hand pattern recognition

Hand pattern recognition is the first and basic module of this system. While a person with disabilities does signs of sign language, his hands perform a series of different movements, where their hand may be spread out, clenched, or his fingers pointing out at something. Therefore, the role of this module is to recognize the pattern of the hands. Then combining the outcome with other modules, the system can give the final result.

This module uses the output of the hand landmark model, which is a matrix size of 21. After calculating all the values in that matrix, we get a new matrix representing the distance between those 21 coordinates. Using the distance matrix as the input of CNN with the designed structure (see figure 28), as seen in figure 27, will tell us the pattern of the hand at the moment it is captured.

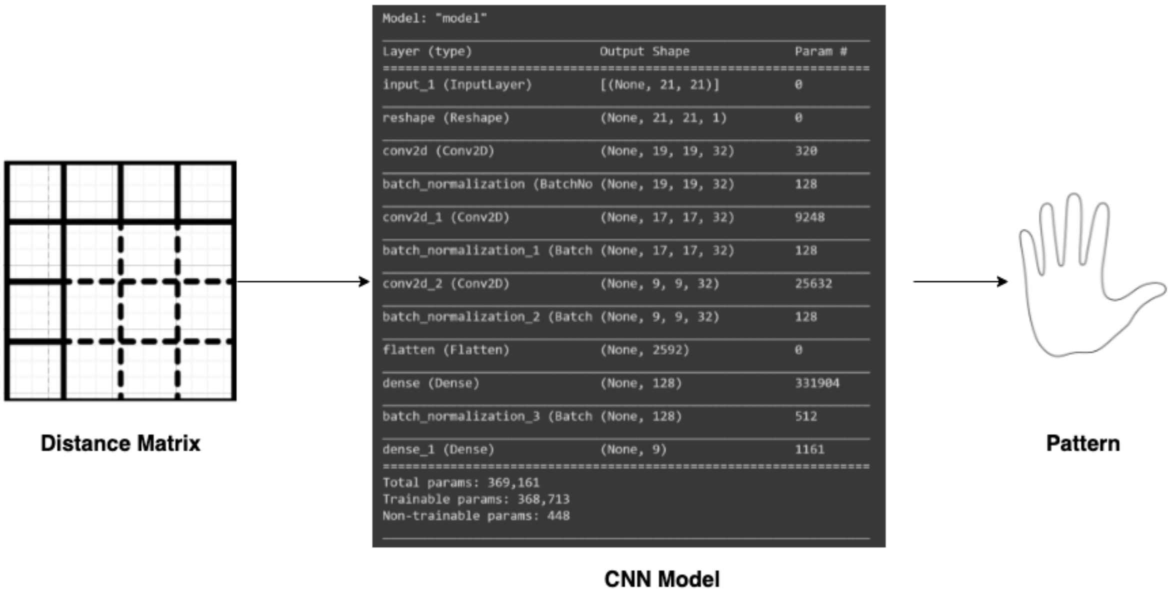


Figure 28: Hand pattern recognition pipe line

4.3.2 Direction determination

The directions of the hand include four directions, i.e., right, left, up, down, front, and back. Each hand's pattern, combined with different directions, leads to a different meaning. For example, the pattern that points at someone means the word "you"; on the other hand, when we point at ourselves, it means the word I (see figure 29 and figure 30).

Figure 29: Word "You" (bạn) in sign language

Figure 30: Word "I" (tôi) in sign language

To determine the hand's direction, we use the hand landmark model provided in MediaPipe (see section 3.2). The inception here is that we calculate the distance between the tip of the index finger and the wrist, which can be called **vector(0, 8)**, then project it to the axis Ox, Oy, Oz, respectively. After that, we take those coordinates and compare them with the others. Finally, the one with the immense value will tell which axis the hand is on; besides, with the direction from the wrist to the tip of the index finger projected on that corresponding axis, we will know which direction the hand is.

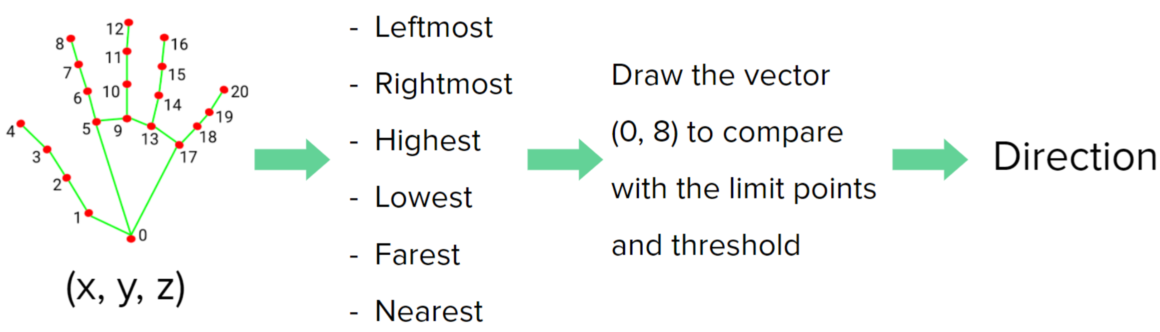


Figure 31: Steps to detect the direction of the hand

For instance, a hand is pointing in the left direction. The value of the distance, when projected on the axis Ox, will be the biggest one among the three projected values. Then, calculate the vector drawn from the wrist to the tip of the index finger; we will know the direction of the hand itself.

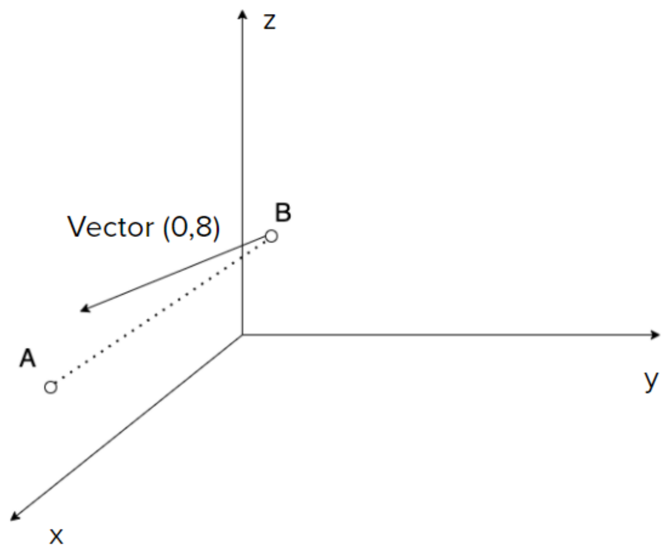


Figure 32: Vector(0, 8) represent the hand pointing toward the left

4.3.3 Location detection

Locations of hand vary, is the hand put at the forehead, mouth or the chest level, and so on. Every hand pattern that goes with every location will result in different words. Nevertheless, it is hard for the AI to know the hand's coordinates with only one camera, and its view is from above (see figure 33). However, we came up with some solutions to this issue.

The zooming method is the first approach we use to detect the hand location. In this solution, we will take images of hands and calculate the hands' size in every frame to know whether those hands are getting larger or smaller. Hence, if those hands' sizes are smaller than before, they are getting far from the camera, and their locations are somewhere at the chest level or the stomach level. Otherwise, the hand's location is nearer to the camera, at the mouth, nose, or forehead level.

Figure 33: View from the camera module

Nonetheless, the above solution still has an issue: every man's hand has a different size, and the system does not know the correct position of the hand. Therefore, another solution is to use a wide-angle camera and set it away from the forehead. With this solution, the camera can have a much broader view. Nevertheless, since we only have a normal-angle camera, we could not try out this solution and confirm its suitability.

Another solution to detect the hand's location is using an ultrasonic sensor. In short, this sensor is an instrument that measures the distance to an object using ultrasonic sound waves (see figure 34). It works by emitting a sound wave with a frequency above the human hearing range. The sensor's transducer functions as a microphone, receiving and transmitting ultrasonic sound. The sensor measures the time between sending and receiving an ultrasonic pulse to determine the distance to a target.

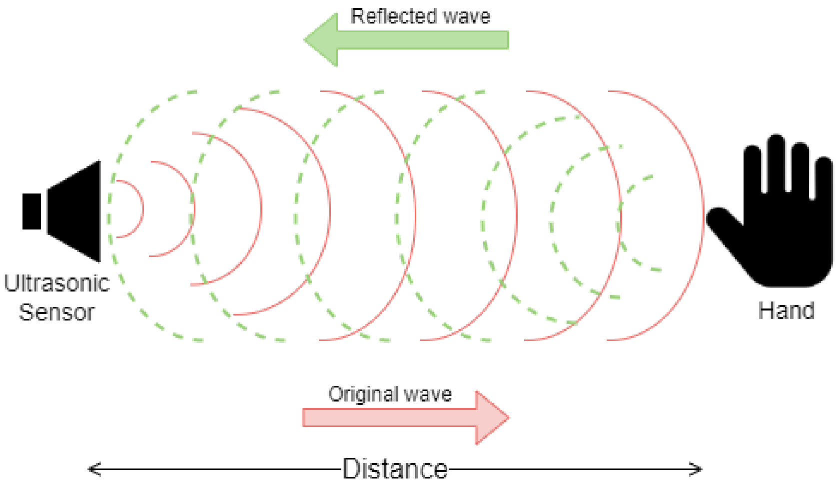


Figure 34: Illustration of how the ultrasonic sensor works

In the thesis, the one we use for this module is the ultrasonic sensor HY-SRF05 (figure 35), which is relatively cheap and meets our demand in measuring the distance between the camera and the hand. According to the retailer, the wide-angle this sensor can scan is up to 15 degrees. Moreover, its scanning range is between 2 cm and 450 cm, with the relative error fluctuating around 0.3 cm. Besides, the most accurate measurement distance is under 100 cm, which is more than enough to measure from the forehead to the user's waist.

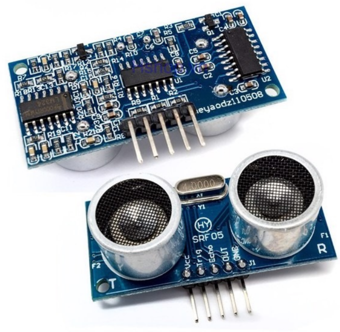


Figure 35: The ultrasonic sensor HY-SRF05

However, the sensor method has not yet achieved the desired effect. The reason is that the sensor can only detect one hand at a time, but what if sign language uses two hands simultaneously? In this case, with only one sensor, we can not bring possible results.

From there, we offer another solution for measuring this distance as follows. Replace that distance method using a location sensor with a distance measurement method based on object size. It is almost similar to the zooming method, but this method is better in that it will not depend on the size of the hand, so it is perfectly suitable for determining the distance for the problem we are trying to solve.

This method is as follows. We will use the following formula:

$$distance\_predict = A * distance\_input + B * distance\_input + C$$

In which *distance_input* will be the distance between two points (5,17) on the hand model taken from the MediaPipe model, and *distance_predict* will be the distance from the camera to the hand.

The trio of coefficients A, B, and C will be determined by interpolating the above polynomial with a pre-prepared data set. After having the above three coefficients, combined with calculating the distance between 2 points 5 and 17, we will quickly deduce the hand's distance.

Figure 36 shows the result we get

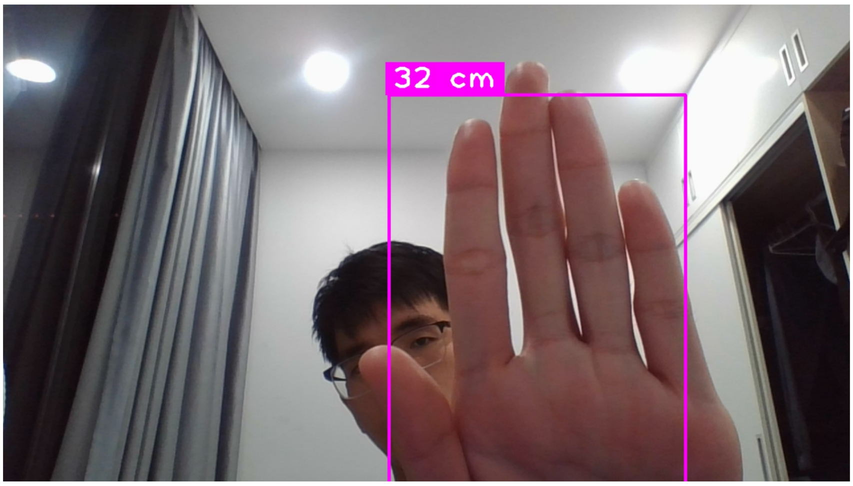


Figure 36: Result of location module

Accordingly, after getting the result from this location module, we put it within a hand state. Likewise, in the following section, we will discuss how that hand state will help us translate sign language into Vietnamese.

4.3.4 Word decoder

There are considerable technical difficulties in implementing the action detection module, as discussed above. We did some research and proposed a new model to resolve these problems. As a result, this change affects the word decoder module, which needs some adjustments.

The previous model decodes a word into four factors: pattern, location, direction, and action. After getting the outputs from the four modules, it will search the database to find the corresponding word. figure 37 illustrates how an input containing four factors is mapped to the correct word in the database. Applying a basic searching algorithm, we have the system find the most appropriate word. If it can not find any, it will replace or deprecate some parts of the input and try again to find another word. After decoding and finding the suitable word, the application will display that word on the screen.

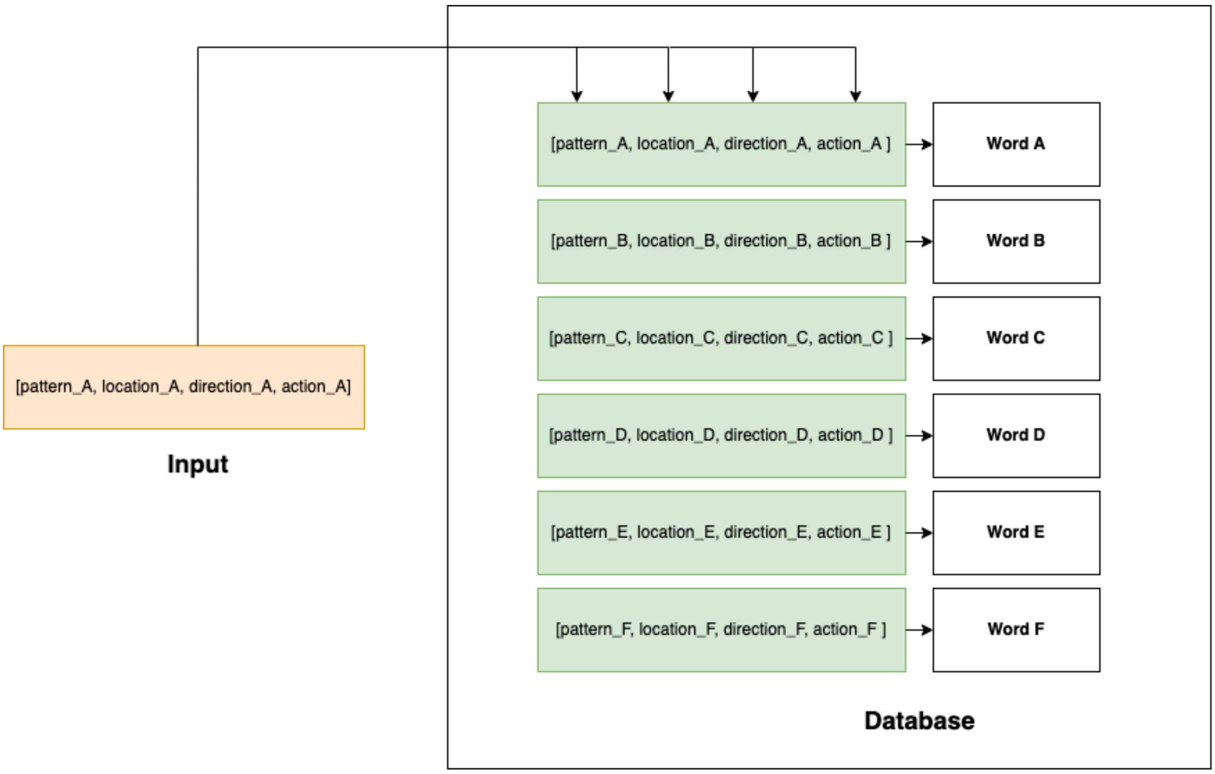


Figure 37: Map one to one data from four component with word in database and get result

Introduction to handstate

Right after the deprecation of the action detection module, the question that comes up is how we can find the correct word without that module. Therefore, we propose a different model for a word that is not decoded into four factors like the previous model. It only contains three elements left: pattern, direction, and location. Consequently, each set of those three elements is called a hand state (38), and a word is decoded into many different hand states.

This concept of hand state comes from the research of natural language processing, in which a word is composed of many characters. Accordingly, a word is concatenated from many hand states in the thesis. Then, we will get the desired word when going through the processing steps that we will discuss later in this proposal.

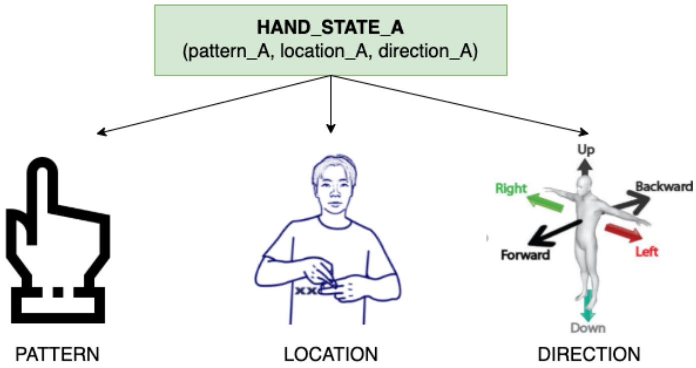


Figure 38: Hand State which construct from pattern, location and direction

Using beam search and CTC decode to map word

After we have grasped the concept of hand state, we will come to the essential part of the model: converting the received hand states into words.

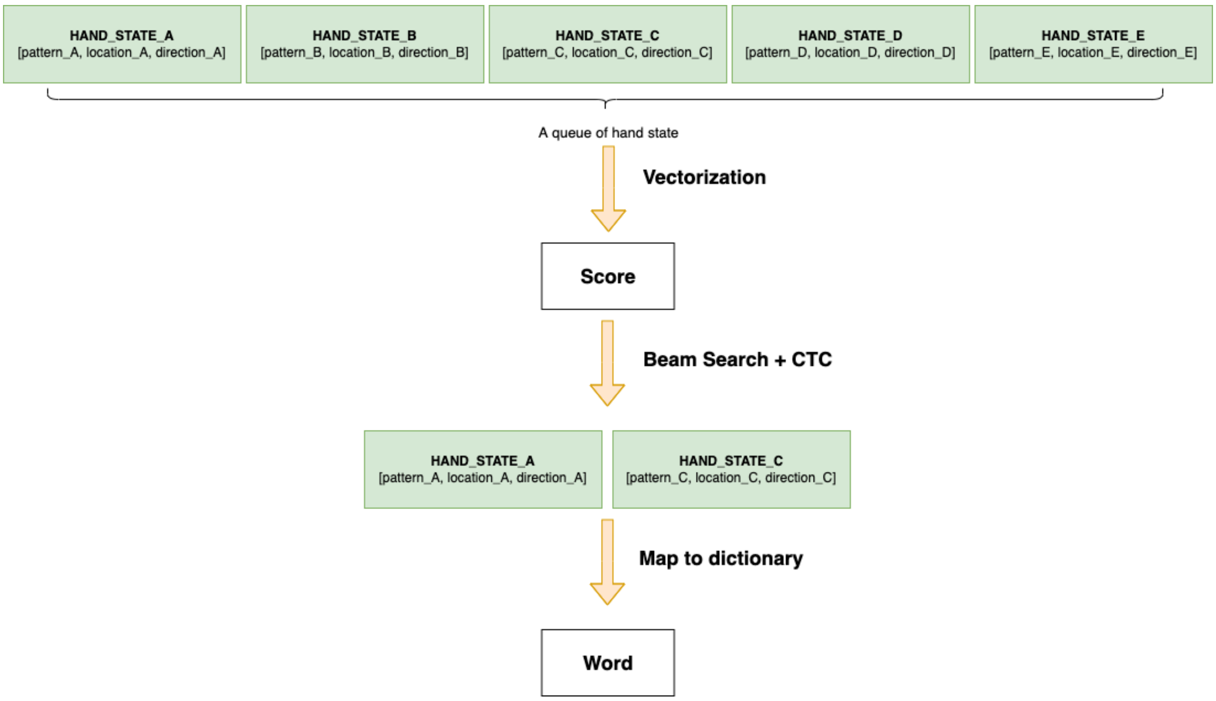


Figure 39: Architecture

Figure 39 is the model proposed by the authors for this section. The input will be a queue of hand states taken from the previous three components. Here, in our conventions, the queue length is set to 5, but it is not the final number, as we need more calculations and experimentation to find the right queue length. This model consists of three steps:

1. **Vectorization:** This step converts a queue of many hand states 40 into a matrix as input for

beam search.

2. **Beam search:** In this step, we will perform a beam search algorithm to choose which hand states are suitable for the input from the database. Besides, we propose using the CTC decode model to eliminate the wrong hand states or previous duplicated hand states, increasing the model's efficiency.
3. **Map to the dictionary:** And finally, after going through the above two steps, from the initial queue, we will get the most likely hand states. Our job is to map these hand states to the database and find the correct word.

Vectorization

When we get to this step, we get a queue of hand states. Because before entering the beam search module, we need a matrix representing the correlation between the outputs received from the components and the data in the database.

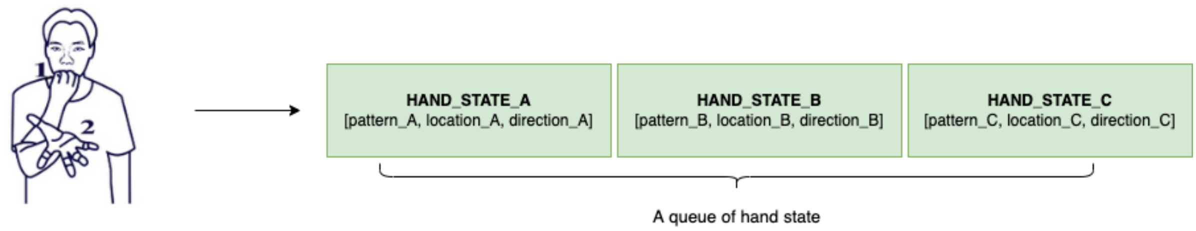


Figure 40: A queue of hand state which get from three component in section ...

From this queue, we will cycle through each hand state, compare it with the available hand state database, and evaluate the score for it based on the following principles 41:

1. The score will be increased if the hand state matches the word in the database.
2. Otherwise, the score will be decreased if that hand state does not match any.

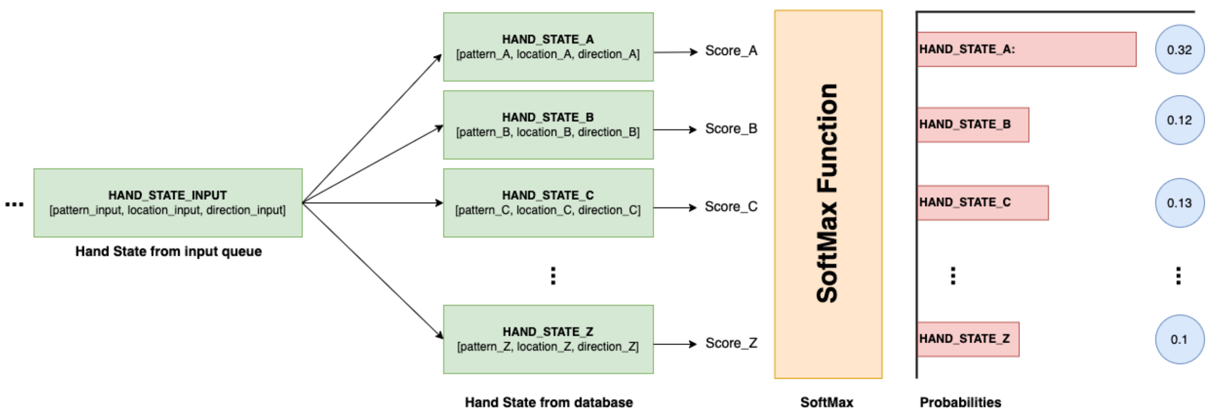


Figure 41: Vectorization

In the first principle, The more similarities the hand state retrieved from the queue has compared to that in the database, the higher it is scored. For example, in the database, we have a hand state as follows: $\begin{bmatrix} pattern\_A & location\_A & direction\_A \end{bmatrix}$, and the hand state we get from the input is $\begin{bmatrix} pattern\_A & location\_A & direction\_A \end{bmatrix}$ then this hand state will be rated higher than the hand state $\begin{bmatrix} pattern\_A & location\_A & direction\_B \end{bmatrix}$. And so on, we will, in turn, score the hand states taken from the queue.

On the second point, the minus point is evaluated based on its matching pattern with the hand states in the database. When the system recognizes patterns from the hand pattern recognition module (using the vision approach), it is likely to be wrong detected or mistaken. To resolve this problem and maximize the accuracy of the result, we put out a rule. With those patterns that are usually hard to detect, the minus point will be lower than simple ones. In short, the more complex the pattern to be recognized, the smaller the minus point is going to be.

After completing the above evaluation and scoring step, we will use a function to normalize the data (here, the authors use the softmax function [12]) and return us a set of probabilities of the hand states in the row. Wait. We will use this set of probabilities as input for the beam search step.

Using beamsearch with CTC decode

After passing the vectorization step, the hand states in our queue have been converted to an MxN matrix, where M is the length of the hand state's database, and N is the length of the queue.

By using beam search (42), we will get the most likely k hand state from the database. We can imagine what happened later during the beam search from the image below.

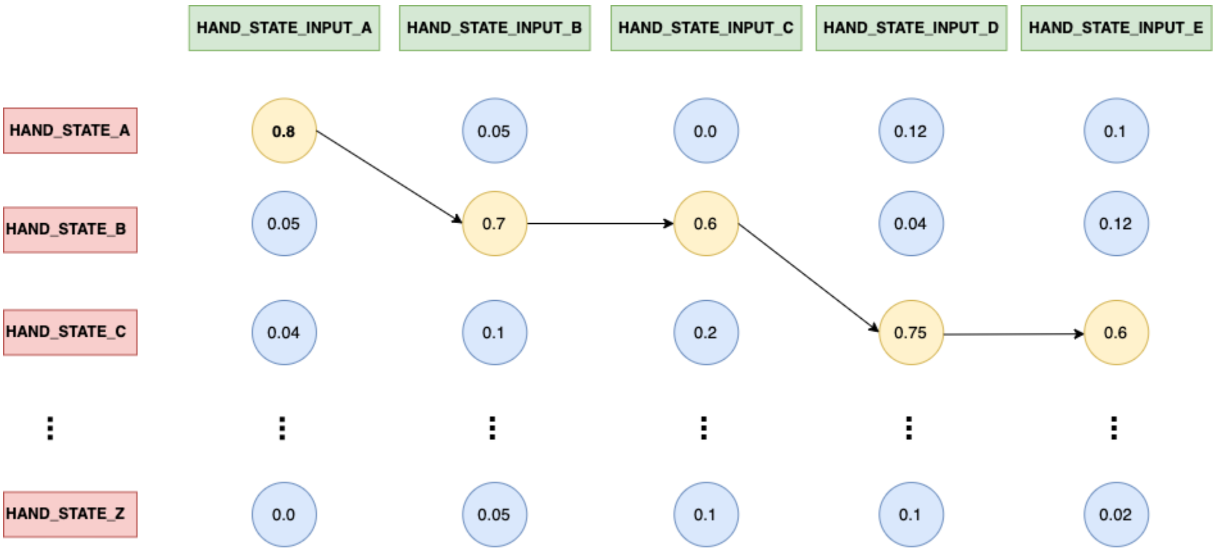


Figure 42: Beam Search

However, as we can see, after performing the beam search step, we will get a sequence of hand states whose length corresponds to the length of the input queue, these hand states can include duplicate hand states, or they can be wrong hand states. Therefore, we need to apply the CTC algorithm to remove the hand states from infection. In the Vectorization step, we will set a threshold to discard these hand states and see it as a blank character for the wrong hand states. Moreover, after beam search CTC decode, we will get the desired result.

Map to dictionary

Figure 43: Map to dictionary and get word

After getting a set of most likely hand states, all that remains is for us to map to the database, as we did in the section above, but instead of the map with a 4-component set, we will map with input retrieved from this previous step. Finally, we will get the word (43) without using the action detect module.

4.3.5 Text-to-speech

In addition, sometimes, people do not always read the result from the phone's screen, so to make it easier for them to know the answer after the translation process, the application can speak it out loud. One way to do this is to build a database of many sound files mapped with the corresponding word in Vietnamese. This approach, however, is not efficient as it requires a massive effort to create the database. We must record every single word and map them all together.

Instead of using that approach, we use a well-known speech service from Google called Text-to-speech [3]. According to Google, Text-to-Speech converts text input into natural human speech audio data. Furthermore, this service supports many languages, including the one that we need, Vietnamese. With the provided API from that speech service, our application can speak up the result without an extensive sound database in the user's phone.

This API is a freemium service. Text-to-Speech costs are determined by the number of characters transmitted to the service each month to be synthesized into audio. Google states on their website, "The first 1 million characters for WaveNet voices are free each month. For Standard (non-WaveNet) voices, the first 4 million characters are free each month. After the free tier has been reached, Text-to-Speech is priced per 1 million text characters processed." the thesis only needs the standard (non-WaveNet) plan, which provides us 4 million characters free each month and only costs USD 4.00 per following 1 million characters. In the upcoming phases of building up the application, the number we will use is negligible compared to 4 million free characters. Therefore, we decided to apply this Text-to-speech module using the Google service.

4.3.6 The camera module

After having discussed mainly the solutions and implementations of the soft modules in the system, we must move on to the main one that is considered to be the eyes of the thesis, which is the camera module. This section will discuss what parts are in a camera module and represent some images of a real one we built.

We can easily find the camera modules' parts from any retailer selling electrical components, robots, and Arduino kits. Additionally, in the current era of e-commerce, it is easier for us to find and compare those components that we need online. The parts required to build a camera module are listed below.

First, we need a camera part, and ESP32-CAM is perfect for this role. It is inexpensive and easy to use, making it ideal for our thesis, which requires complex functions like image tracking and recognition. Furthermore, it integrates Wi-Fi, and traditional Bluetooth, which help us send the images to the user's smartphone for the next steps in translating sign language.

Secondly, we need a converter adapter to help us sideload the program into the camera module. The third part that we need is the ultrasonic sensor mentioned in the TK section. It plays a role in location detection, which will tell the system the distance between hands and the camera module. Last but not least, this camera module needs a battery to power the whole module, and we reckon that the volume of about 100 mAh is fine.

Furthermore, there must be a box to store all the above parts. With the help of current 3D printing technology, we design that package on Tinkercad, an online 3D modeling program that runs on the web browser. After getting all the necessary components, we tried to put them together and get the result below.

Figure 44: The components inside the camera module prototype

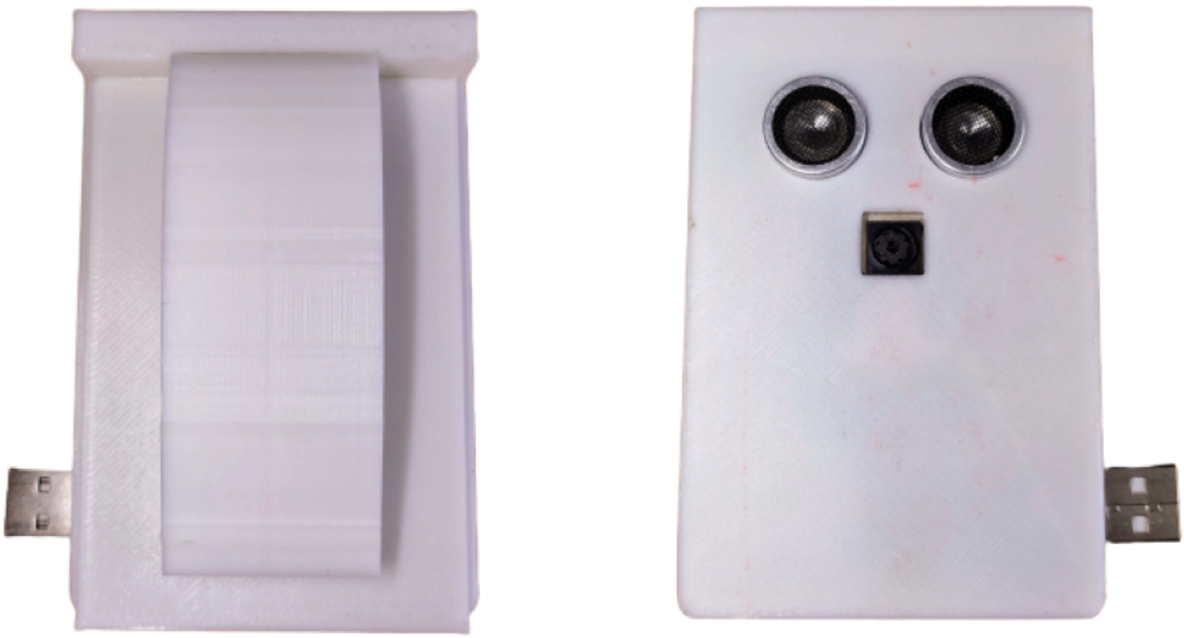


Figure 45: Views of the camera module prototype from the above and under

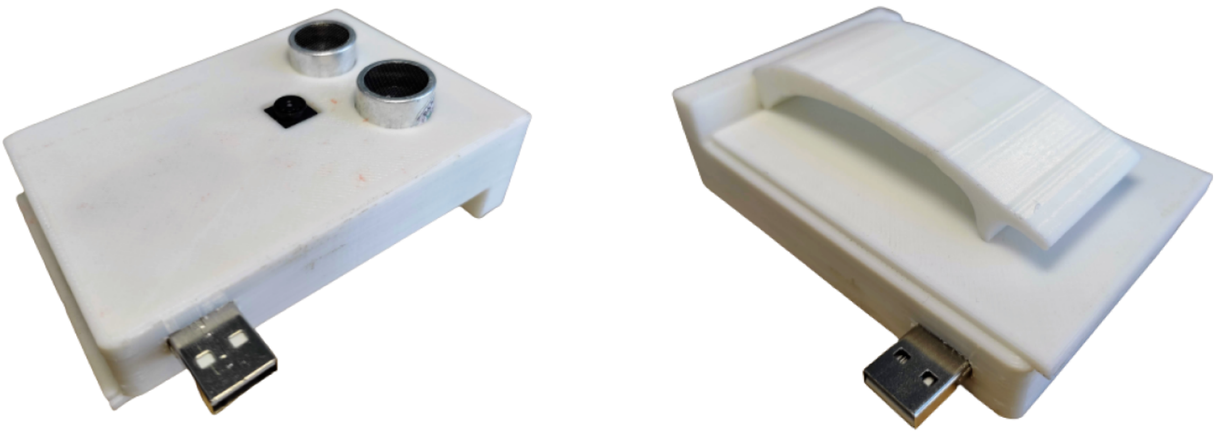


Figure 46: Views of the camera module prototype from the sides

Nevertheless, due to the smallest number that a 3D printer can print, the box's cover is a bit hard to put in. The hanger that helped hang the box on the hat is not as flexible as we thought, so it needs a redesign.

4.3.7 App design

The application must satisfy user experience, and user interface demands. And during the research phase of the thesis, we did design a prototype for the application, including one more features besides the main one. That side feature is a sign language dictionary. We illustrate that feature in Table 7.

Before going through the design of this application, we must state that they do not cover all the screens needed for the application yet. And they are not the final design that we have. However, we have some conventions when designing this prototype, such as the corner is rounded and the colors are pale, not too bright, to make the users feel calm somehow and comfortable when using the application.

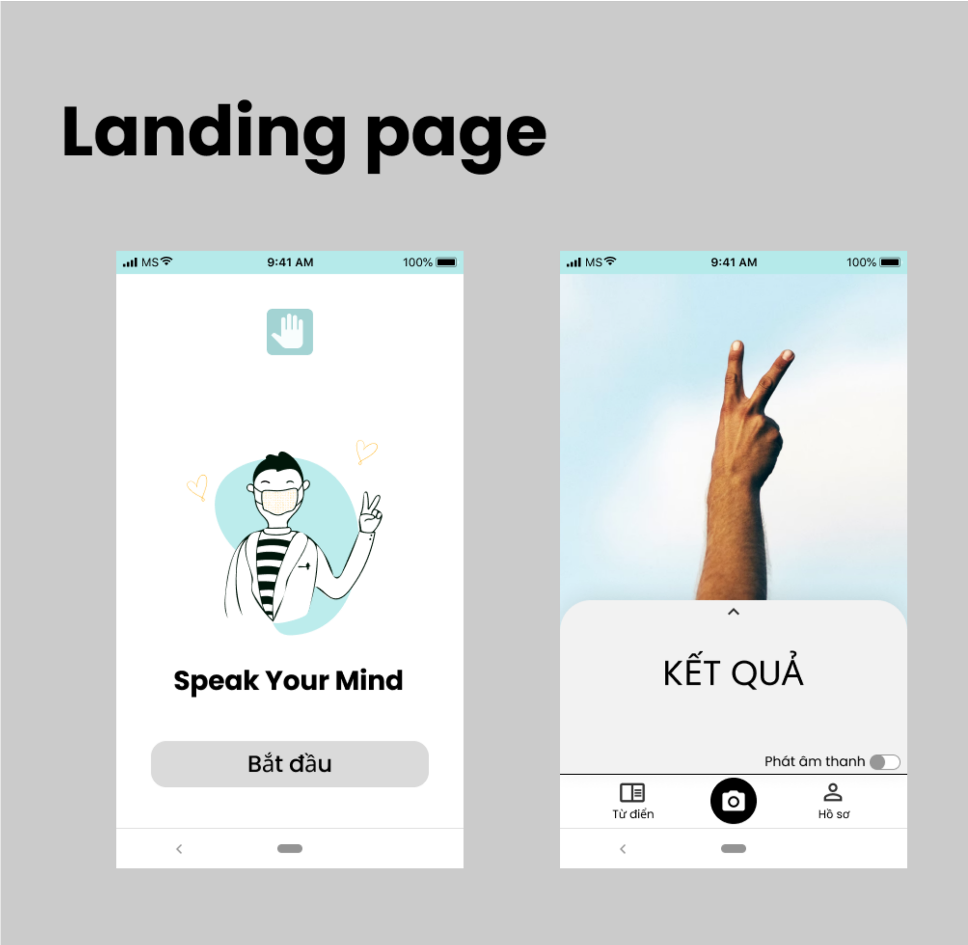


Figure 47: The landing page of the application

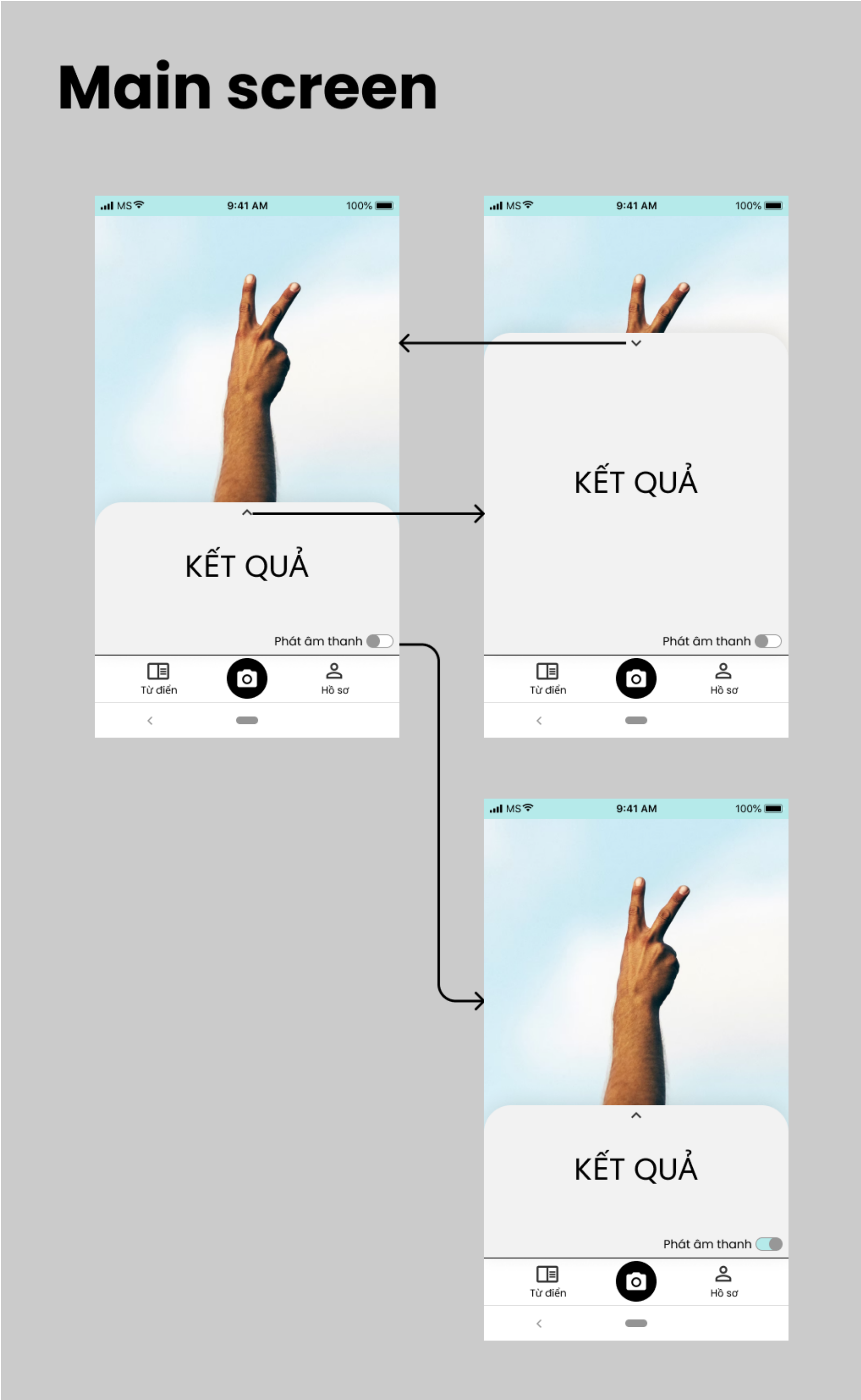


Figure 48: The main screen of the application

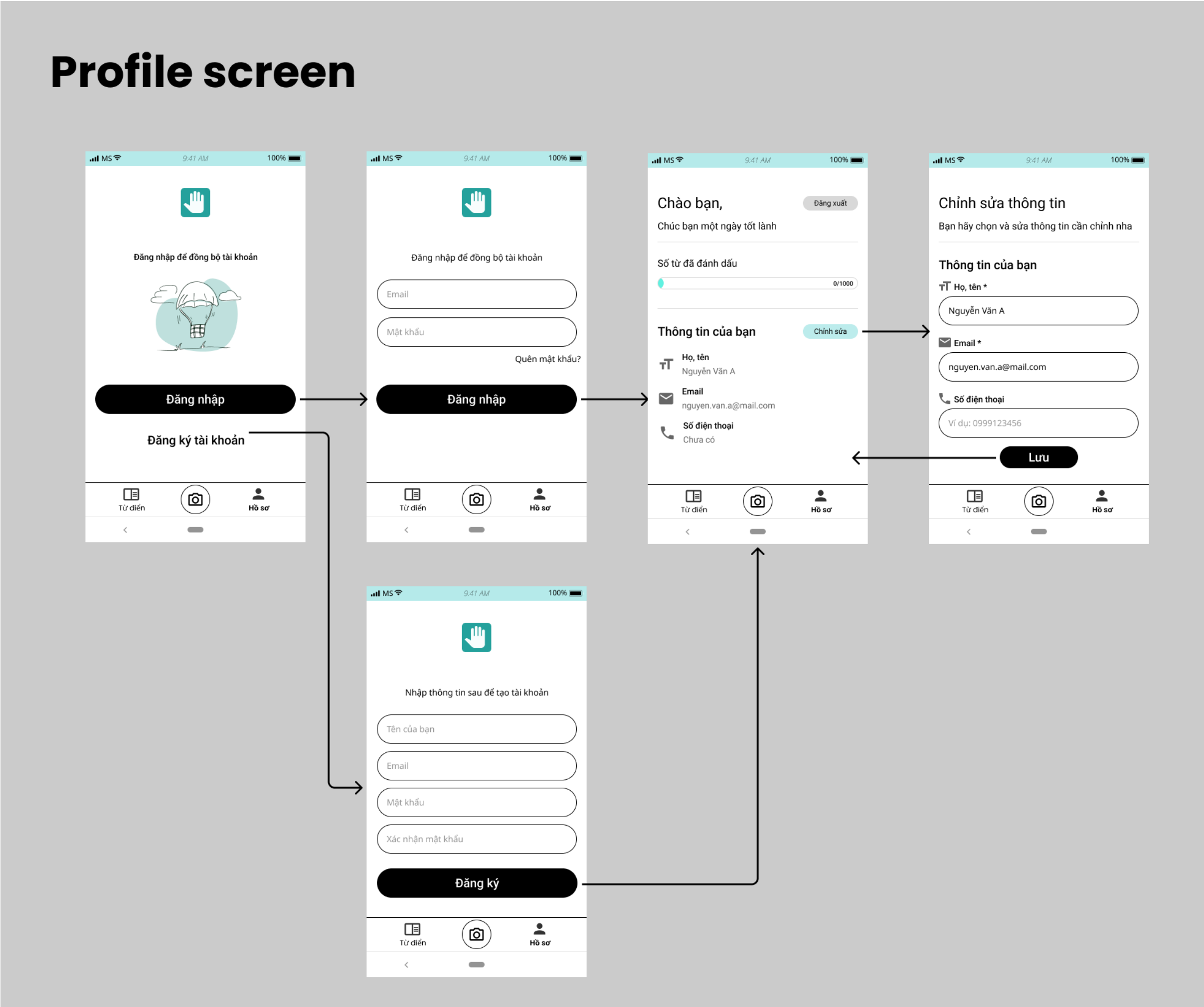


Figure 49: The profile screen

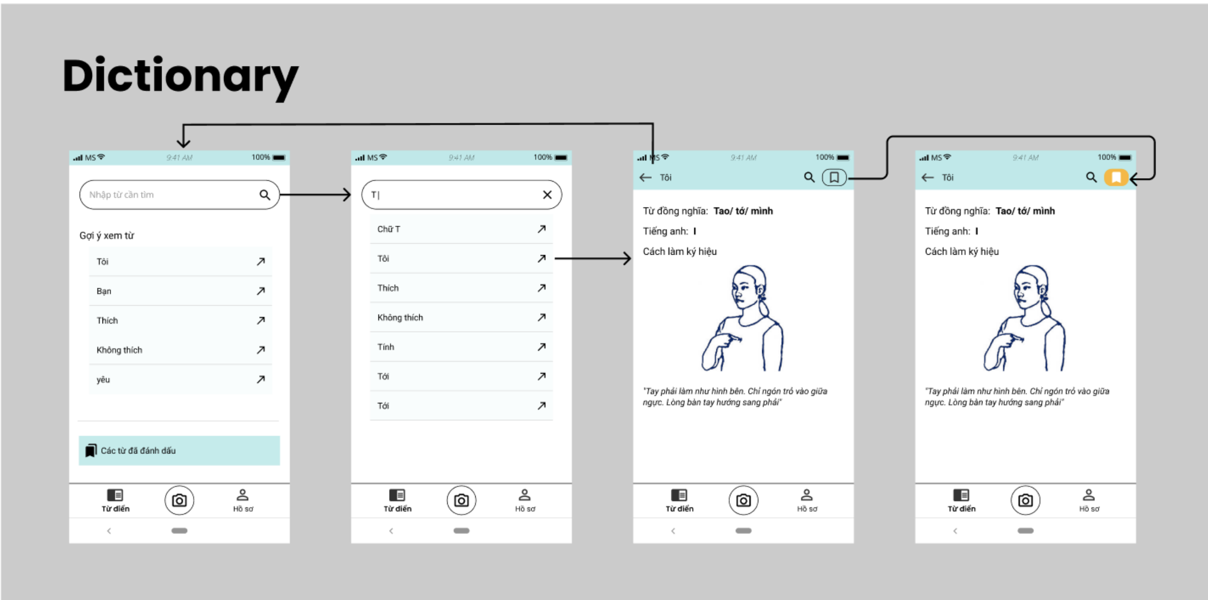


Figure 50: The dictionary screen

4.3.8 Use case

On the whole, Figure 51 illustrates the overall use-case of the project.

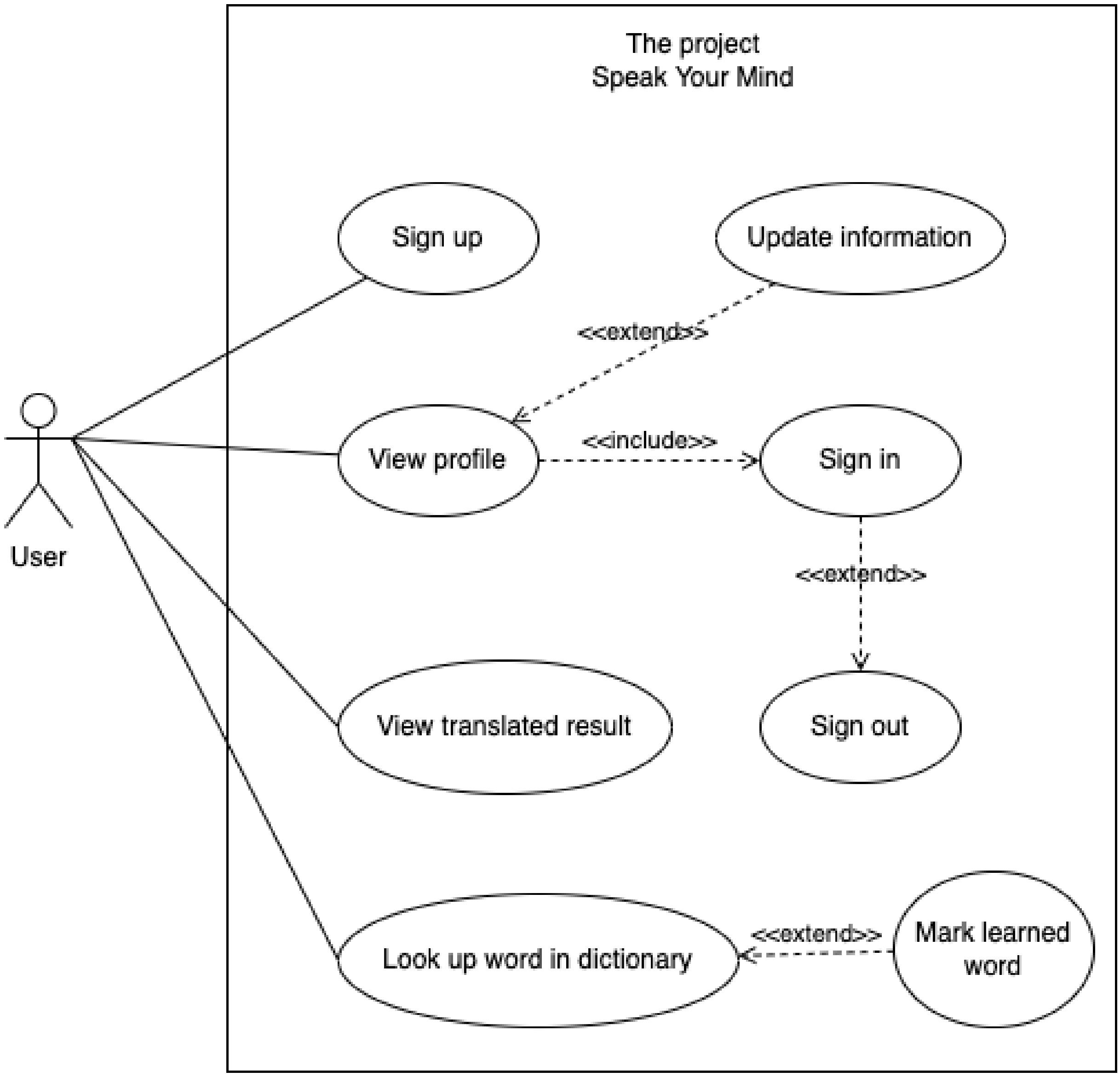


Figure 51: Use case diagram

4.3.9 Use case description

View result after predict

Use case ID	1
Use case name	View result after predict
Description	The user receives information about the hand gesture he has just made
Actor	User
Post-condition(s)	Detects successful sign language and returns results to the user
Normal flow	1. User visit main page 2. The user performs the operation describing the sign language 3. Application that records actions and makes predictions 4. The application displays the results after the prediction
Exception flow	Exception1: 4a. The application cannot predict the sign language 5a. The application shows no more information and ends

Table 2: Use case view result after predict

View profile

Use case ID	2
Use case name	View profile
Description	User can view his personal information and perform some operations to edit the number of learned words and the number of new words learned in the day.
Actor	User
Post-condition(s)	Returns the user's information
Normal flow	1. User visit profile page 2. The application accesses the system to get user information 3. Application that displays user information
Alternative flow	Null
Exception flow	Null

Table 3: Use case view profile

Sign up

Use case ID	3
Use case name	Sign up
Description	Create a personal account to be able to log into the system
Actor	User
Pre-condition(s)	Device containing the application with internet connection
Post-condition(s)	Successful account registration
Normal flow	1. User clicks on “Đăng ký tài khoản” button 2. The application displays the account registration view 3. The user enters the required system information 4. User presses "Đăng ký" button 5. The system returns to the login screen
Alternative flow	A. The user stopped creating an account and wants to go back to the login screen User presses "Đăng nhập” button and the application jumps to step 5
Exception flow	If there is no internet connection, the application displays the message "No internet connection! Please try again."

Table 4: Use case sign up

Sign in

Use case ID	4
Use case name	Sign in
Description	Allow user to log in to his account to use the application's services
Actor	User
Pre-condition(s)	Have internet connection
Post-condition(s)	Logged in successfully
Normal flow	1. The application displays the login information filling interface 2. User enters account name and password in 2 boxes Email, Password 3. User presses "Đăng nhập" button 4. The application shows "Logged in successfully"
Alternative flow	A. User entered wrong login email 4.1 The application shows "Error ! There is no user record corresponding to this identifier. The user may have been deleted." 4.2 Application goes back to step 2 B. User enters username that is not email 4.1 The application shows "Error ! The email address is badly formatted." 4.2 Application goes back to step 2 C.User entered wrong password 4.1 The application shows "Error ! The password is invalid." 4.2 Application goes back to step 2
Exception flow	If there is no internet connection, the application displays the message "No internet connection! Please try again."

Table 5: Use case sign in

Sign out

Use case ID	5
Use case name	Sign out
Description	Sign out
Actor	User
Pre-condition(s)	The device contains an application with internet connection Previously logged in
Post-condition(s)	Sign out successfully
Normal flow	1. The user presses the "Đăng xuất" button 2. The application returns to the login screen.
Alternative flow	No
Exception flow	If there is no internet connection, the application displays the message "No internet connection! Please try again."

Table 6: Use case sign out

Look up word in dictionary

Use case ID	6
Use case name	Look up word in dictionary
Description	User needs to look up a word in the sign language dictionary
Actor	User
Pre-condition(s)	For the better result, the device should have internet connection
Post-condition(s)	User successfully finds out how to sign the word
Normal flow	1. The user presses dictionary in the navbar 2. The application directs the user to screen dictionary. 3. The user inputs the word and presses it on screen. 4. The application show the corresponding information of the word, includes the instruction video.
Alternative flow	No
Exception flow	If there is no internet connection, the application only shows the local information, and the video is note included.

Table 7: Use case look up word in dictionary

Mark learned word

Use case ID	7
Use case name	Mark learned word
Description	User wants to mark a word as learned to find it easily in the future
Actor	User
Pre-condition(s)	Previously logged in
Post-condition(s)	User marked it and this will increase the learning bar in the profile page
Normal flow	1. The user presses dictionary in the navbar 2. The application directs the user to screen dictionary. 3. The user inputs the word and presses it on screen. 4. The application show the corresponding information of the word, includes the instruction video. 5. User taps on the bookmark button on the top right 6. That bookmark will change from outline state to filled state
Alternative flow	No
Exception flow	If the user hasn't logged in, the application will notify that "You have to login to perform this"

Table 8: Mark leaned word

Chapter 5

Result and evaluation

5.1 Result

To remove barriers for the deaf and hard of hearing. The team has created an artificial intelligence application to recognize gestures, manipulate words, and translate them directly into Vietnamese.

A demo of the system is presented here: `https://www.youtube.com/watch?v=Gn8mdojQnqY`

Besides, with the achieved results, the group has also carried out procedures to apply for intellectual property rights for the product and is waiting for approval.

However, the system still has certain disadvantages, such as long, complex terms and many operations the proposed system cannot recognize. Depending on its complexity, the time it takes for the system to recognize a sign language is still slow.

5.2 Evaluation

From the actual results, the group also has plans and orientations to make the system more complete as follows:

- Ultimately convert the application to use react-native to meet the needs of both Android and IOS platforms.
- Improve the system and fix the errors in the system.

Chapter 6

Summary

the thesis applies image processing and artificial intelligence, whose purposes are to research and build a system that can translate sign language into Vietnamese using only a camera module and a smartphone.

So far, a sign language translating system has been a massive challenge to many scientists and engineers because of the complexity of sign language and the diversity of the way people use it around the world. Moreover, when we researched and built the system, there were a few similar systems, but they only translated the sign language alphabet.

In addition, talking about human values, this system can resolve the lack of sign-language translators in Vietnam. It, indeed, means that people having disabilities will have the chance to live, work and communicate like those who do not. They can have a better education as the teacher can understand their thoughts and connect more efficiently. They can have better health as the health force has the chance to know more about how they are, what they feel, which means we can provide them a better treatment for their problem. Their life will be easier as the surrounding people can get them and talk to them more clearly.

The deaf and mute are also a part of this world, a part of us, not apart from us. Therefore, we firmly believe the deaf and mute deserve to have the chance to speak up, be heard, be seen, and be acknowledged. With this application, we people can know each other and communicate fluently regardless of our level of knowledge of sign language. Ultimately, our bonds will grow more vital and more profound, which will lead to a better world for the entire human race.

Those human values emphasize the importance of this project in our world. Besides, the promising solution we presented throughout this proposal means it is possible to translate sign language with the current technologies. In the upcoming time, we have more resources to dedicate our time to completing our algorithm, which results in the higher accuracy of the translation process and completing the thesis thoroughly.

Bibliography

[1] Tim Bock. What is a distance matrix? https://www.displayr.com/what-is-a-distance-matrix/, 2021. (Accessed on 5/12/2021).

[2] Ketan Doshi. Foundations of nlp explained visually: Beam search, how it works. https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24, 2021. (Accessed on 5/12/2021).

[3] Google. Text-to-speech. https://cloud.google.com/text-to-speech, 2021. (Accessed on 03/12/2021).

[4] Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. https://distill.pub/2017/ctc.

[5] Nguyễn Yên Kiều Tuyết. Thiếu phiên dịch viên, bệnh nhân đặc biệt thiệt thòi về cơ hội chăm sóc y tế. https://vovgiaothong.vn/thieu-phien-dich-vien-benh-nhan-dac-biet-thiet-thoi-ve-co-hoi-cham-soc-y-te, 2021. (Accessed on 03/08/2021).

[6] Alina Kuznetsova, Laura Leal-Taixé, and Bodo Rosenhahn. Real-time sign language recognition using a consumer depth camera. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 83–90, 2013.

[7] Sarfaraz Masood, Adhyan Srivastava, Harish Chandra Thuwal, and Musheer Ahmad. Real-time sign language gesture (word) recognition from video sequences using cnn and rnn. In *Intelligent Engineering Informatics*. Springer, 2018.

[8] MediaPipe. Mediapipe face mesh. https://google.github.io/mediapipe/solutions/face_mesh.html, 2021. (Accessed on 03/12/2021).

[9] Timothy F O'Connor, Matthew E Fach, Rachel Miller, Samuel E Root, Patrick P Mercier, and Darren J Lipomi. The language of glove: Wireless gesture decoder with low-power and stretchable hybrid electronics. *PloS one*, 12(7):e0179766, 2017.

[10] Harald Scheidl, Stefan Fiel, and Robert Sablatnig. Word beam search: A connectionist temporal classification decoding algorithm. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 253–258. IEEE, 2018.

[11] Hải Vân. Mở cánh cửa hy vọng cho người khiếm thính. https://nhandan.vn/baothoinay-xahoi/mo-canh-cua-hy-vong-cho-nguoi-khiem-thinh-313775/. (Accessed on 03/08/2021).

[12] Thomas Wood. What is the softmax function? `https://deepai.org/machine-learning-glossary-and-terms/softmax-layer`, 2021. (Accessed on 2/12/2021).

[13] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.