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

———————— * ————————

GRADUATION THESIS PROPOSAL

USING MACHINE LEARNING METHODS IN TRANSLATING SIGN LANGUAGE INTO VIETNAMESE

Council: Software Engineering
Instructor: Assoc. Prof. Quan Thanh Tho

—o0o—

Student: Võ Tuấn Khanh (1810220)
Nguyễn Trí Nhân (1810390)

Ho Chi Minh City, December 2021

Declaration Of Authenticity

We claim that this research is our work, conducted under the supervision and guidance of Associate Professor Quan Thanh Tho. The result of our research is authentic and has never been published in any way before. All the materials I utilized in this research were gathered from various sources and are referenced correctly in the references section.

Furthermore, within this research, I also used the results of several other authors and organizations. All of them have been correctly cited. In any case of plagiarism, I 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 this thesis outline, I would like to express my deep gratitude to Associate Professor Quan Thanh Tho 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 luggage to enter life confidently.

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

Abstract

Presently, more than two million people with deaf and hard of hearing in Vietnam cannot talk freely with those who do not have those symptoms. There are a few ways for the deaf and hard of 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 know about sign language, making it hard to understand.

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 having 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/`. Based on the videos that instruct vocabularies, 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 communicate and enrich their lives, but it also increases the volume of the workforce and reduces the economic burden on the national budget since then.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

"Each deaf person is a separate world, and they feel more self-deprecating and alone when they do 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.

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 dependent 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.

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, there still exist inequalities in living standards and social participation for people with disabilities [6]. Many organizations are founded to support, help, and create better living conditions for people with disabilities to develop. However, this work still has many difficulties and inadequacies as there is no formal school or class. Moreover, there is no specific profession for this group of people, and the num-

ber of translators who know sign language is insufficient, 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 have an excellent 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 resolve to problems of the deaf and hard of hearing.

1.2 Goals of this thesis

This thesis aims to research, understand, and implement solutions to convert from sign language to 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 down into smaller ones listed below. With each issue, we will give our solution and architect a system that can solve the whole problem of this 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 this 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 that it has been trained with

1.4 Structure of this thesis proposal

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

Chapter	Content
1	A brief introduction about plan and objectives of thesis
2	Related works that had been done and how they help us in doing this thesis
3	Introduction of theoretical background as foundation knowledge that are applied in the thesis
4	Solution and design approach for problem statement of this thesis
5	Plan to finish this thesis in the upcoming time
6	Proposed chapters of the thesis
7	Summary of this thesis proposal

Table 1: Structure of this thesis proposal

Chapter 2

Related Work

Nowadays, Many scholars from all around the world have submitted research projects relating to the topic of turning sign language into text, using a variety of methodologies and perspectives. In which, two main approaches can be mentioned as follows:

- Glove based approaches: With this approach, it requires deaf and mute people to wearing a sensor glove. When user has any different action or gesture, these sensor will be recorded. After that, data from sensor will analyze by analyzer component and return the output for user.
- Vision based approaches: With this approach, image processing algorithms will be applied to be able to determine hand position, gestures and movements of the hand. The user will not have to wear necessary equipment like glove based approaches, which is convenient for user. However, with using library or algorithms of image processing, we need to deal with worst quality output, which is greatly affected by this algorithms.

Early work, with vision based approach, they used several types of image processing to build a feature vectors base on a single RGB image of hand. In this paper "Real-time sign language recognition using a consumer depth camera", with using multi-layered random forest (MLRF), not only it allows they recognize hand signs correctly, but also minimize training time and effort. Or in this paper "Sign Language Translation in Urdu/Hindi Through Microsoft Kinect", sign language can be recognize by an auxiliary equipment: Microsoft Kinect, which captures the signs of the deaf person, after that, through the computer system, they can detect what does deaf people say.

Beside vision based approach, glove based approach has many relevant research. This paper “The Language of Glove: Wireless gesture decoder with low-power and stretchable hybrid electronics” introduces the way to convert American Sign Language (ASL) alphabet into text and display it on a computer or smartphone. With sensor glove, they can detect which hand gesture

is performed and sent the result to smartphone via bluetooth. The sign language interprets into some text and displays on the digital display screen. Both this approach is useful in the real world for the deaf and dumb who are unable to converse with normal people.

With both approaches above, there is has some problems, that is, they can only recognize a very small number of words, like an alphabet, number or some word with easily hand shape and no motion . However, in sign language, it not easy, there will be many words that use the same hand shape but will differ in many characteristics, such as position and direction. To our knowledge, there is currently no model that can handle the conversion of sign language flexibly and conveniently for the deaf-mute, helping them to communicate effectively and naturally. Therefore, by applying appropriate technologies, the authors carry out this graduation thesis with the goal of breaking down the barriers between deaf-mute people and normal people, helping them to become self-sufficient. more confident in daily communication.

Chapter 3

Theoretical Background

3.1 Convolution Neural Network - CNN

Convolution Neural Networks are a special class of Neural Networks. 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 consist 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 certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. Some of the main uses of CNN can be mentioned as: image classification, object detection, semantic segmentation, face recognition, etc.

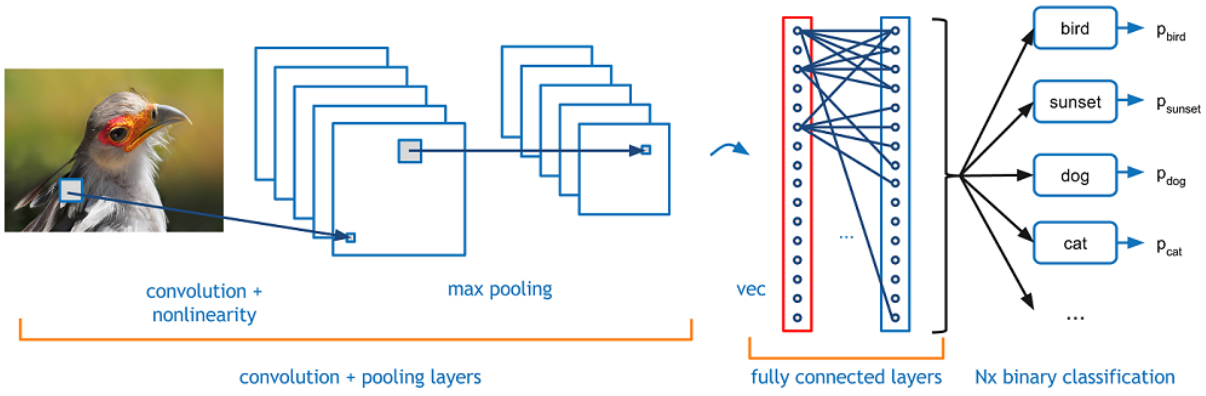


Figure 1: Convolution Neural Network

The figure 1 above shows an example of convolution neural network, which is taking an image as input and then extracting features from it through various layers and then finally pre-

dicting the class of the object in the given image.

3.1.1 Architecture

Convolution Neural Networks have a different architecture than regular Neural Networks, we can see this different in figure 2 below. Regular Neural Networks transform an input by putting it through a series of hidden layers. Every layer is made up of a set of neurons, where each layer is fully connected to all neurons in the layer before. Finally, there is a last fully-connected layer (the output layer) that represent the predictions. With CNN architecture. First of all, the layers are organized in 3 dimensions: width, height and depth. Further, the neurons in one layer do not connect to all the neurons in the next layer but only to a small region of it. Lastly, the final output will be reduced to a single vector of probability scores, organized along the depth dimension.

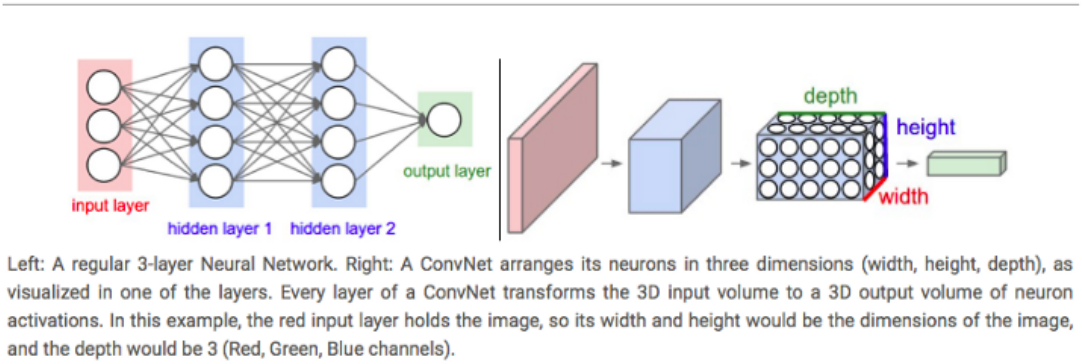


Figure 2: Different between Normal Neural Network and Convolution Neural Network

As we can see in figure 3. CNN can be divided into two parts:

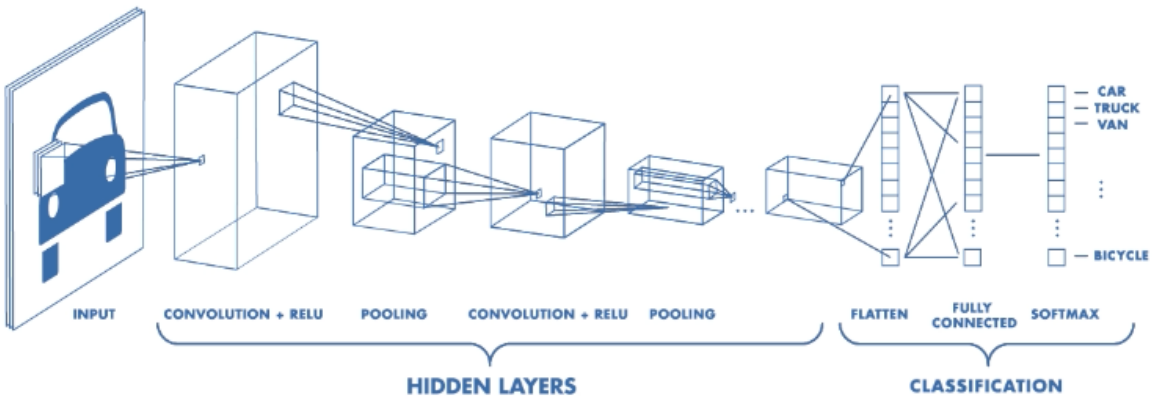


Figure 3: Convolution Neural Network Architecture

1. The hidden layers/ Feature extraction part
    - In this part, the network will perform a series of convolutions and pooling operations during which the features are detected. If you had a picture of a zebra, this is the part where the network would recognise its stripes, two ears, and four legs.
2. The Classification part
    - Here, the fully connected layers will serve as a classifier on top of these extracted features. They will assign a probability for the object on the image being what the algorithm predicts it is.

3.1.2 Feature extraction part

Convolutional Layer

Convolution Layer is the core building block of a Convolutional Network that does most the computational heavy lifting. A convolution is executed by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map. This process of extracting features from image happens throughout the CNN's convolutional layers. This process is illustrated in figure 4

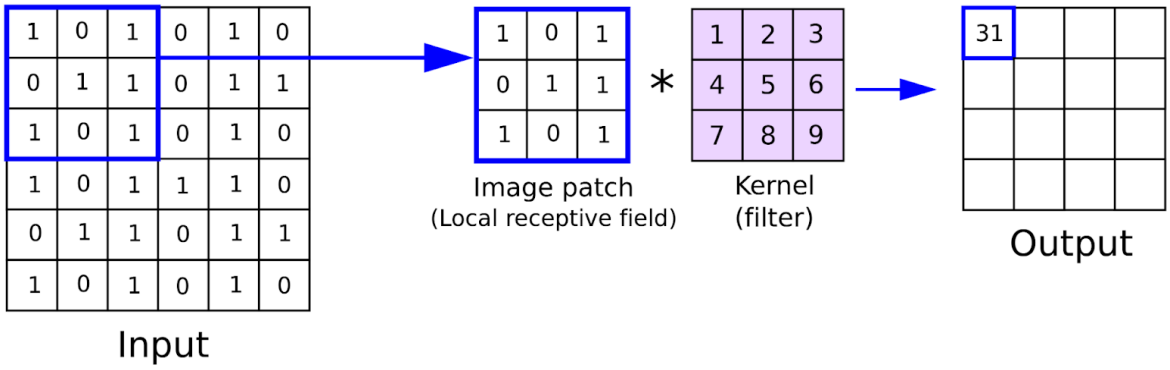


Figure 4: Convolution Neural Network Layer

When the feature map is made, we can pass each value in the feature map through a non-linearity function, such as ReLU, sigmoid, etc. 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 (5). 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 and makes sure the kernel and stride size will fit in the input.

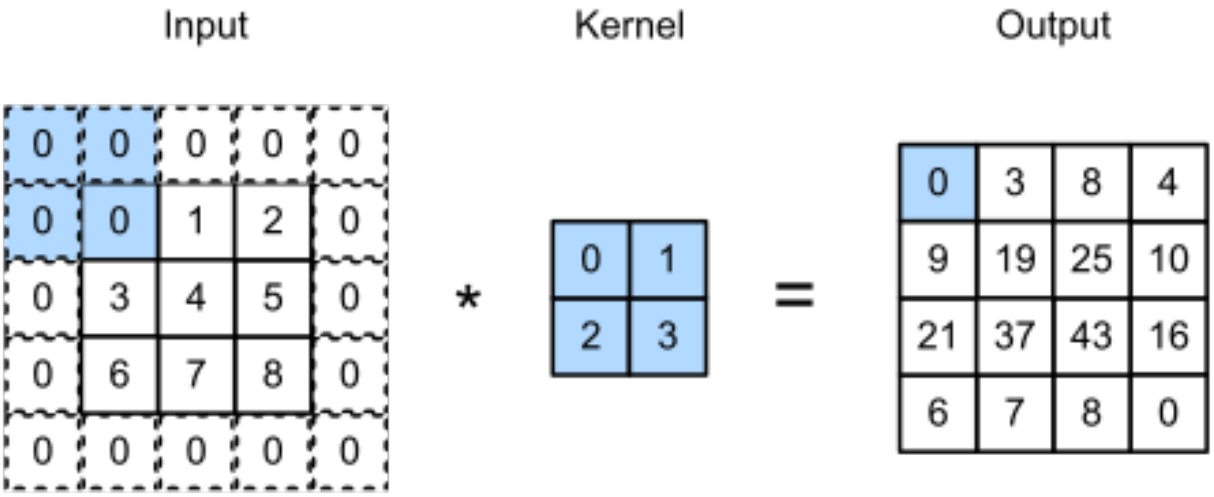


Figure 5: Using padding for strike one in Convolution Layer

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 shortens the training time and controls overfitting.

There are mainly two types of Pooling Layers in a CNN: Max Pooling and Average Pooling. The functionality of these two types of layers are demonstrated in figure 6 . Max Pooling restores the maximum value from the segment of the picture covered by the Kernel. Whereas, Average Pooling restores the average of the multitude of values from the bit of the picture covered by the Kernel.

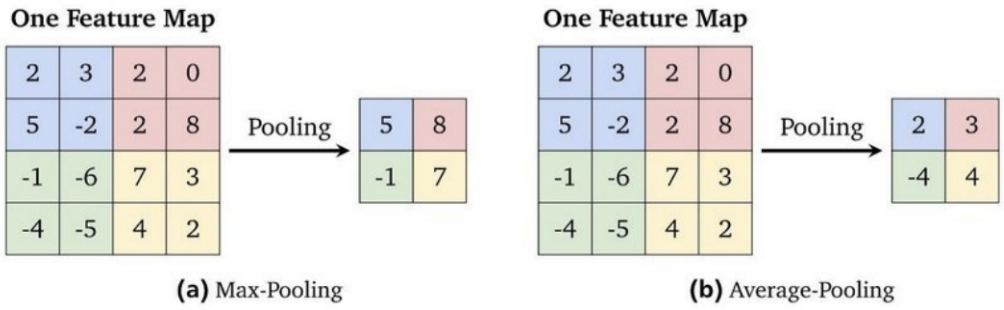


Figure 6: Max Pooling and Average Pooling

Activation Layers

Neural networks in general and CNNs in particular rely on a non-linear "trigger" function to signal distinct identification of likely features on each hidden layer. CNNs may use a variety of specific functions (figure 7), such as rectified linear units (ReLUs) and continuous trigger (non-linear) functions—to efficiently implement this non-linear triggering.

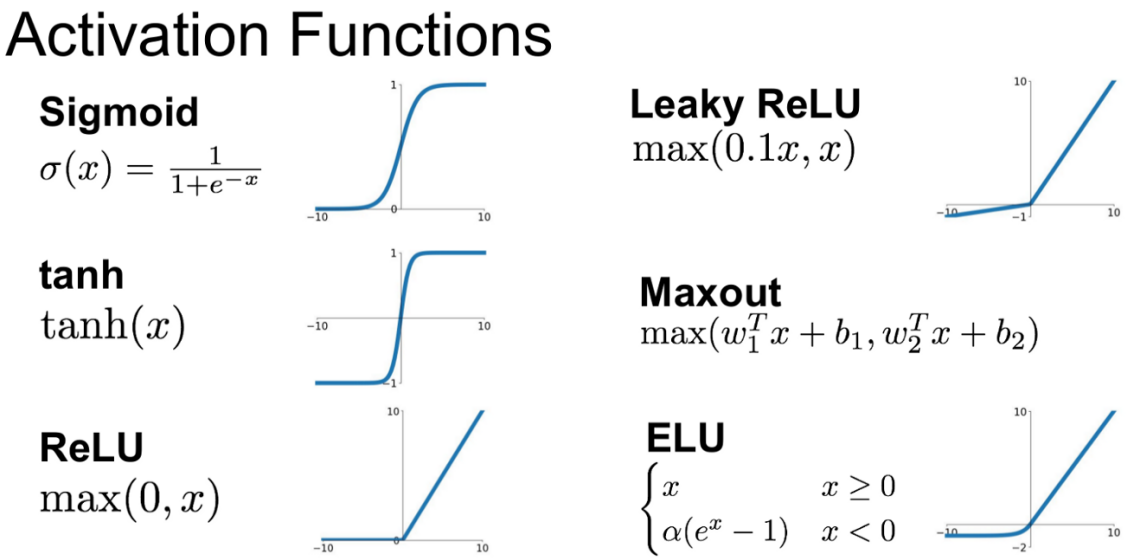


Figure 7: Some Active Function common used in CNN

3.1.3 Classification part

Fully connected layers

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

Figure 8 illustrates the way of input value stream into the fully connected layer. Because these fully connected layer can only accept one dimensional data. So, we need convert our 3D data to 1D data. After pass through some FC, we will get the result is the data classification.

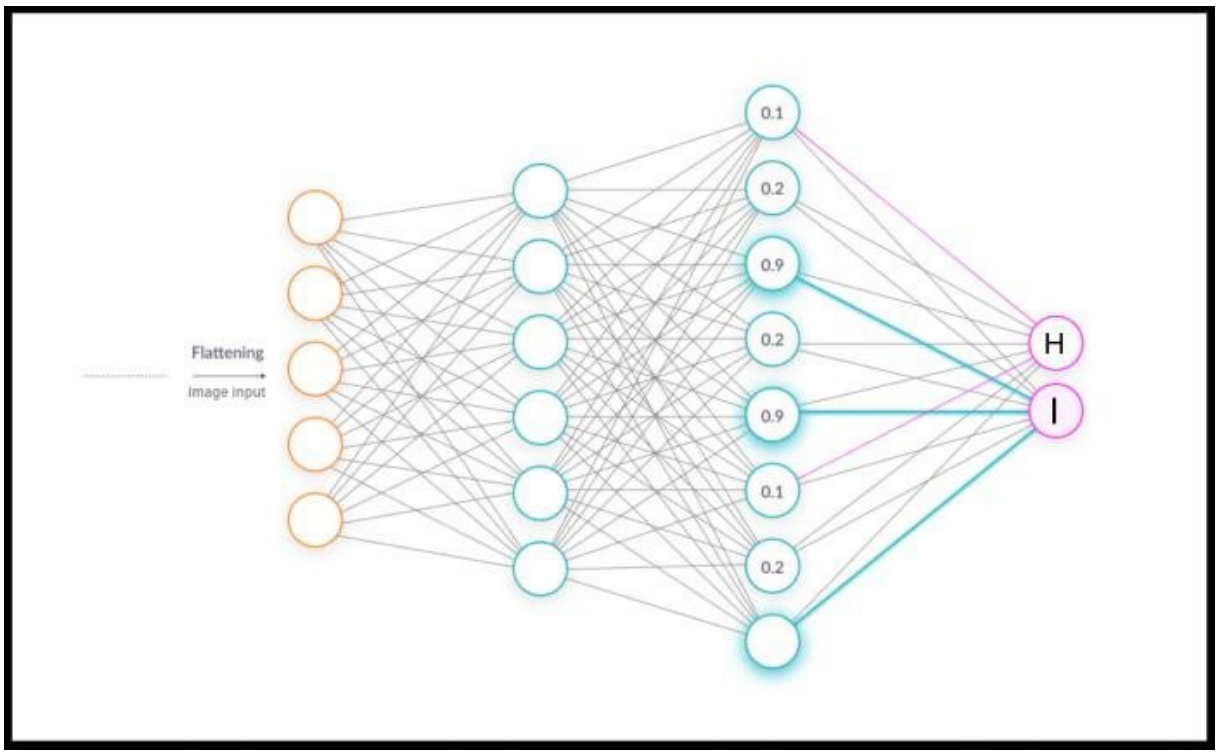


Figure 8: Fully connected Layer

3.2 Media Pipe

3.2.1 Introduction to Media Pipe Hands

MediaPipe Hands (9) is a high-resolution tracking system for hands and fingers. 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.

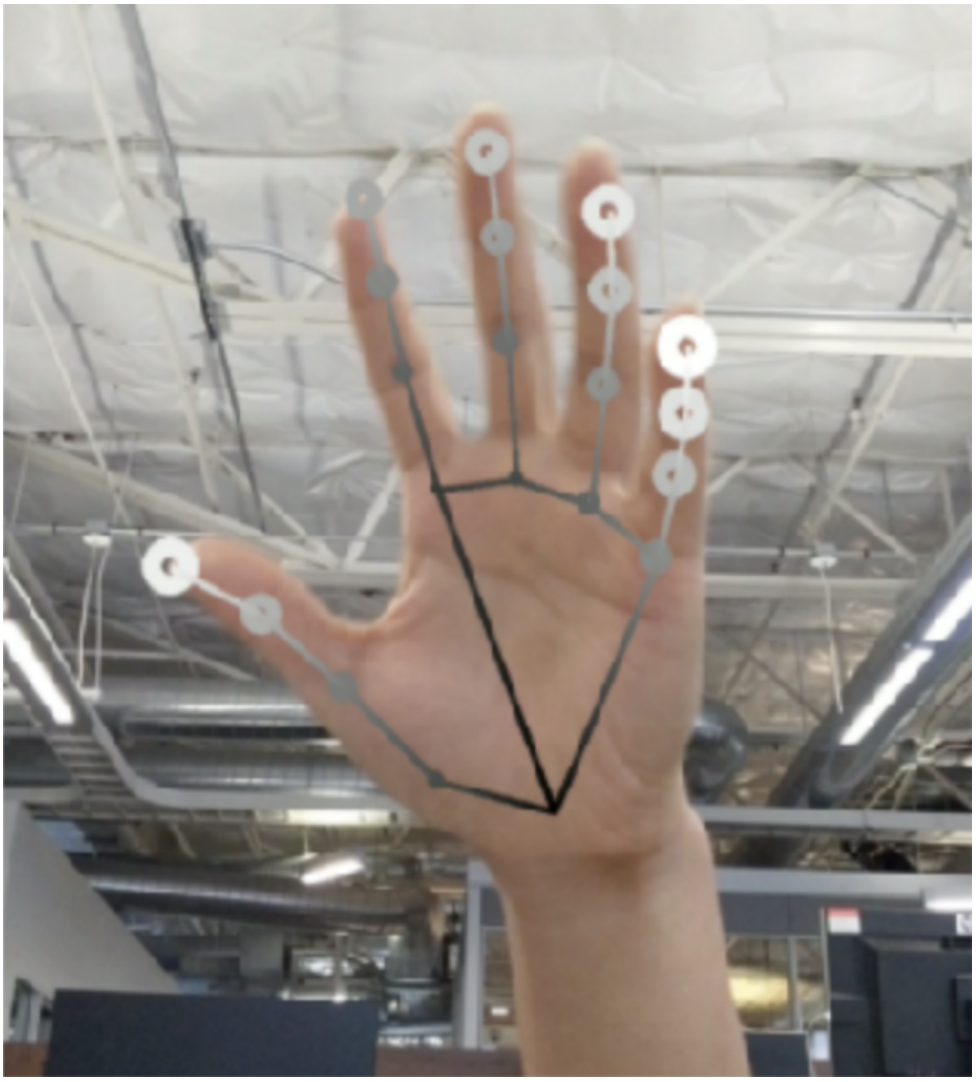


Figure 9: Media Pipe real time tracking 3D hand landmarks

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

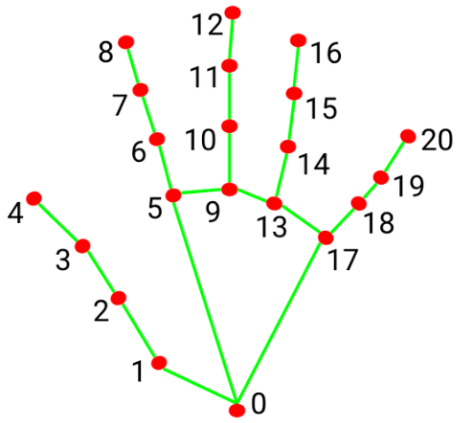
The Media Pipe team provides the palm detection model to detect initial hand locations and distinguish whether the hand recognized is left or right, which is very useful as each sign goes along with a different side will result in different meanings. They created a single-shot detector model, comparable to the face detection model in MediaPipe Face Mesh, 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 statement, the methods they use to address the above challenges vary in many strategies. First, instead of training a hand detector, they train a palm detector because estimating bounding boxes of inflexible objects like palms and fists is 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 10) 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.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 10: 21 Hand Landmarks

3.3 Distance Matrix

A distance matrix is a table that shows the distance between pairs of objects. For example, in the figure 11., we can see the distance of A and B is 16, B and C is 37 and so on. In the diagonal of table is the distance of 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 11: Distance Matrix

3.3.1 Create Distance Matrix

A distance matrix is computed from a raw data table. In the example below (12), we can use high school math (Pythagoras) to work out that distance between A and B.

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

Figure 12: Calculating distance between A and B

We can use same formula with more than two variables, and this is known as the Euclidean distance. In result, we have the distance matrix represented like figure 13

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 13: 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) is a type of Neural Network output helpful in tackling sequence problems like handwriting (figure 14) 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.

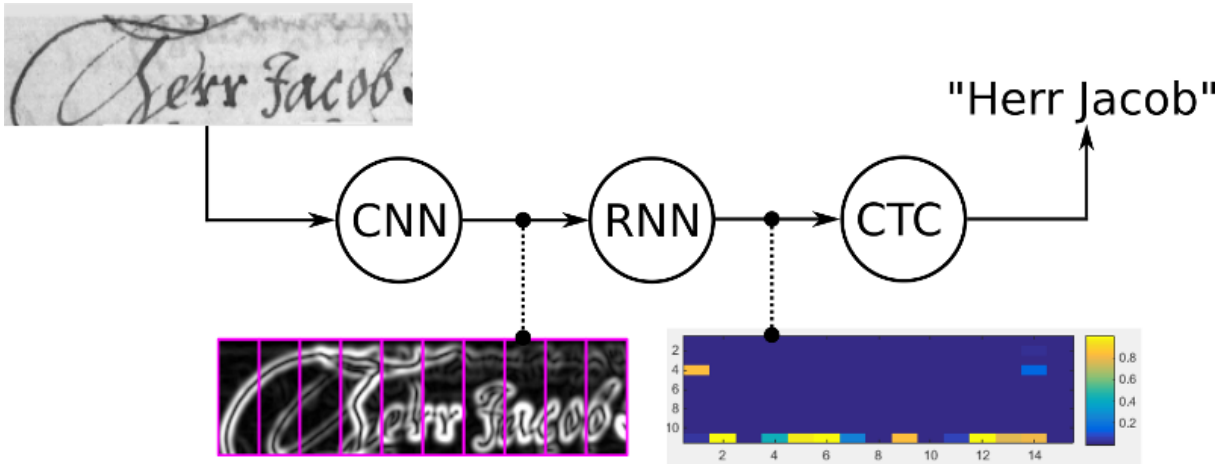


Figure 14: Overview of a Neural Network for handwriting recognition

3.4.2 Why we want to use CTC

In context of hand written recognition, we could create a data-set with images of text-lines, and then specify for each horizontal position of the image the corresponding character as shown in figure 15 Then, we could train a model to output a character-score for each horizontal position. However, there are two problem with this solution.

- It takes a lot of time, annotating dataset at the character level is a boring task.
- 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 Fig..... We have to remove all duplicate character like "t" and "o".

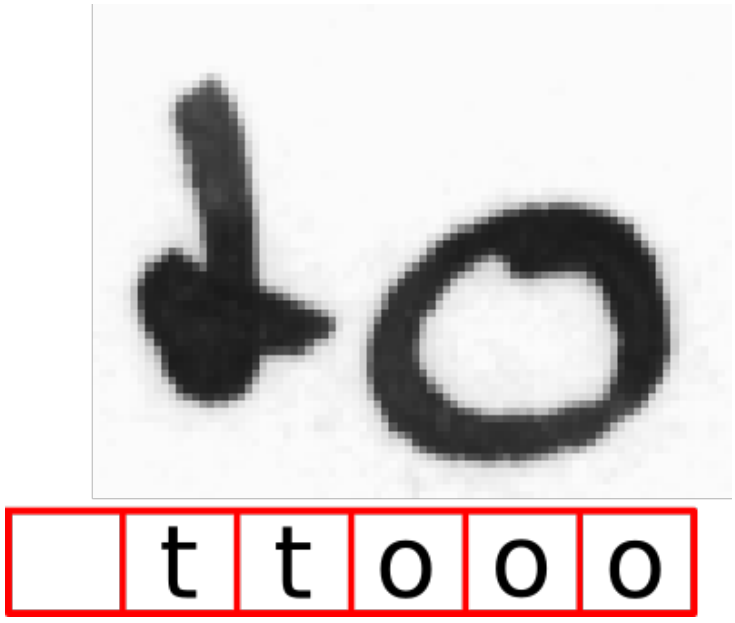


Figure 15: Annotation for each horizontal position of the image

CTC can solves both problem 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 image.
- Using decode techniques, we can directly get the result of the network and no further post-processing of the recongnized text is needed.

3.4.3 Beam Search with CTC decoder

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

Beam search

In computer science, beam search 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 which orders all partial solutions (states) according to some heuristic. But in beam search, only a predetermined number of best partial solutions are kept as candidates. Pseudo-code for basic version of beam-search is shown is figure 16

```
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 16: Basic version of Beam Search

Beam search algorithm will be implemented through the following steps, with two parameter will be included: output matrix and beam width (BW) which specifies the number of beams to keep . First of all, the list of beam and corresponding score is initialized (line 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). Each of beam, we calculate the score and get result (line 8), we will cover this step in more detail 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 is returned as a result (line 16).

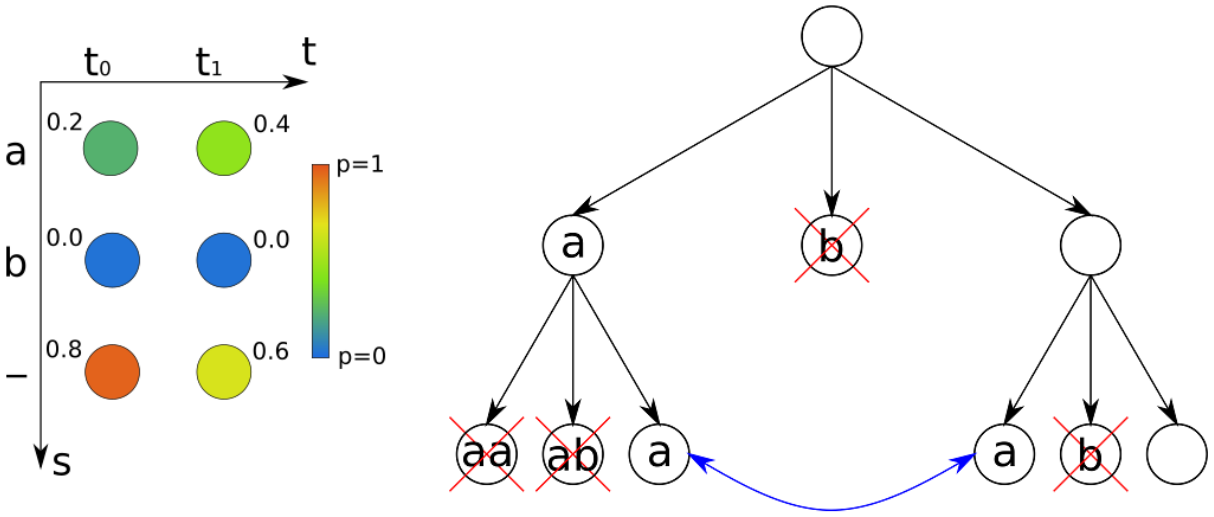


Figure 17: NN output and tree of beams with alphabet = "a", "b" and BW = 2

As we can see, in figure 17, both output matrix to be decoded and the tree of beams are 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 we just discuss above, in this part, we will talk about how to scoring 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 Pb(b,t) and by Pnb(b,t) for the non-blank case.
- The probability Ptot(b,t) of a beam b at time-step t is simply the sum of Pb and Pnb, for example: Ptot(b,t) = Pb(b,t) + Pnb(b,t)

blank: 'aa-' +
- '-' = 'aa--' → "a" (copy)
- 'a' = 'aa-a' → "aa" (extend)
- 'b' = 'aa-b' → "ab" (extend)

non-blank: 'aaa' +
- '-' = 'aaa-' → "a" (copy)
- 'a' = 'aaaa' → "a" (copy)
- 'b' = 'aaab' → "ab" (extend)

Figure 18: The effect of appending a character to paths ending with blank and non-blank

In figure 18, we will see what happens when we extend a path. Three main case we can mention is:

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

And when we collapse the extended paths, two result we will get and some case 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: $P_b(n,t) += P_{tot}(b,t-1) * mat(blank,t)$
  - Beside, with 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 t care about Pb(b+c,t) because we added a non-blank character

Putting it all together

Figure 19 depicts the CTC beam search algorithm. It is similar to the basic version previously displayed. However, it includes the code to score the beams: copied beams (lines 7-10) and extended beams(line 15-19). Finally, when we looking for the best scoring beams, the programs ranks them according to Ptot (line 4) and then take the BW best ones.

Data: NN output matrix mat, BW and LM
Result: decoded text

$beams = \{\varnothing\}$;
$P_b(\varnothing, 0) = 1$;
for $t = 1...T$ **do**
  $bestBeams = bestBeams(beams, BW)$;
  $beams$={};
  for $b \in bestBeams$ **do**
    if b != $\varnothing$ **then**
      $P_{nb}(b,t)$ += $P_{nb}(b, t-1) \cdot mat(b(-1), t)$;
    end
    $P_b(b,t)$ += $P_{tot}(b, t-1) \cdot mat(blank, t)$;
    $beams = beams \cup b$;
    for $c \in alphabet$ **do**
      $b' = b + c$;
      $P_{txt}(b') = applyLM(LM, b, c)$;
      if $b(t) == c$ **then**
        $P_{nb}(b',t)$ += $P_b(b, t-1) \cdot mat(c, t)$;
      else
        $P_{nb}(b',t)$ += $P_{tot}(b, t-1) \cdot mat(c, t)$;
      end
      $beams = beams \cup b'$;
    end
  end
end
return $bestBeams(beams, 1)$;

Figure 19: CTC beam search

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. And 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.

But first, we need to collect the sign language data for the model training phase in this 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 preparing data, we collect many word from the website and youtube channel. Then, we label and seperate it into many element which we will discuss later in section about hand state (4.3.4). We made the google sheet for the data we gathered. In this file, we have prepare many word were labeled (see figure 20). Beside, we have a sheet for many hand shape (see in figure 21) which help us to classifying hand pattern.

	A	B	C	D	E	F
	#	Câu	Link các video liên quan (xem bên sheet Danh sách video)	Danh sách pattern sử dụng		
1		Tôi đi học ?		Tôi: chụm_trỏ		
2		Xin chào	3	Vẫy tay (1 hay 2 đều được)		
3		Bạn tên gì ?	3	Bạn: chụm_trỏ	Tên: Gõ ngón trỏ này lên ngón trỏ kia 2 lần	What: 2 tay để OK -> vẫy ra
4		Tôi tên N-H-Â-N	4	Mình: chụm_trỏ (inward)	Tên: Gõ ngón trỏ này lên ngón trỏ kia 2 lần	N

Figure 20: Google sheet about word labeled

	Label	pattern
A	nắm	
B	xòe	
C	chu_C	
D	chụm	

Figure 21: 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 this 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 22). 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, and after combining the first four modules' results (hand pattern, direction, location, and action detection), the word decoder module will take the output data and bring out the corresponding outcome. Then, the result will show up on the main screen; meanwhile, the phone will speak out that word.

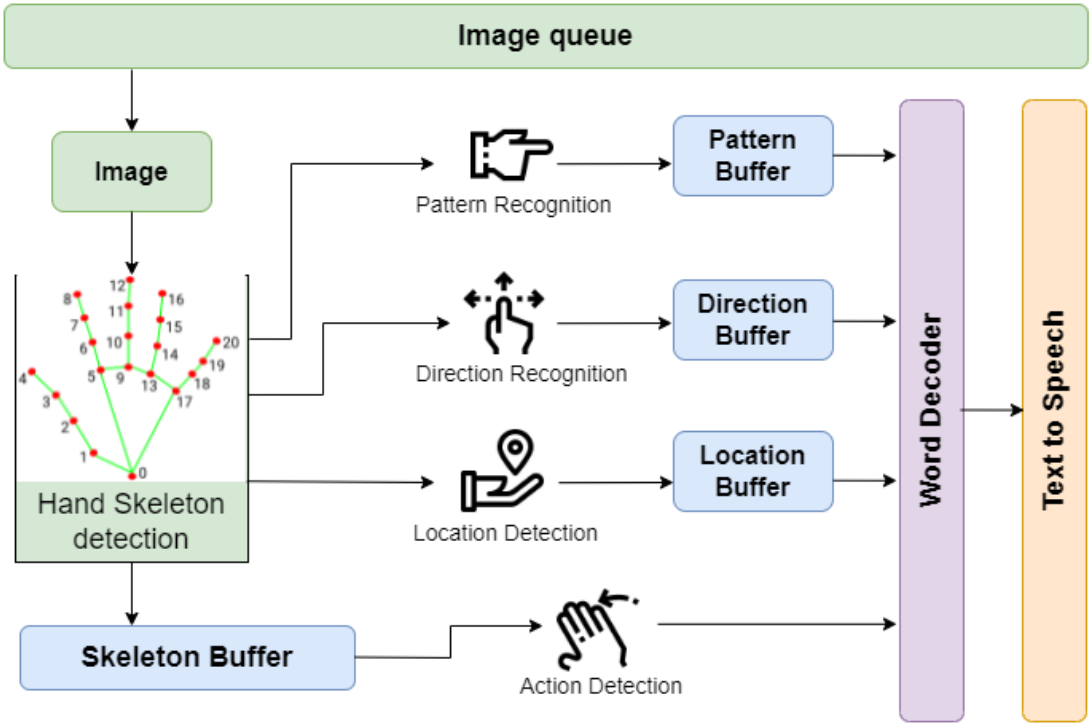


Figure 22: Overview of the old system structure

Those six main modules mentioned above are the ones we planned up at the beginning of this thesis. However, during the implementing period, we found it hard to build the action detection module, 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 sending 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 Viet-

namese word to resolve that problem. Instead of using a combination of 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.

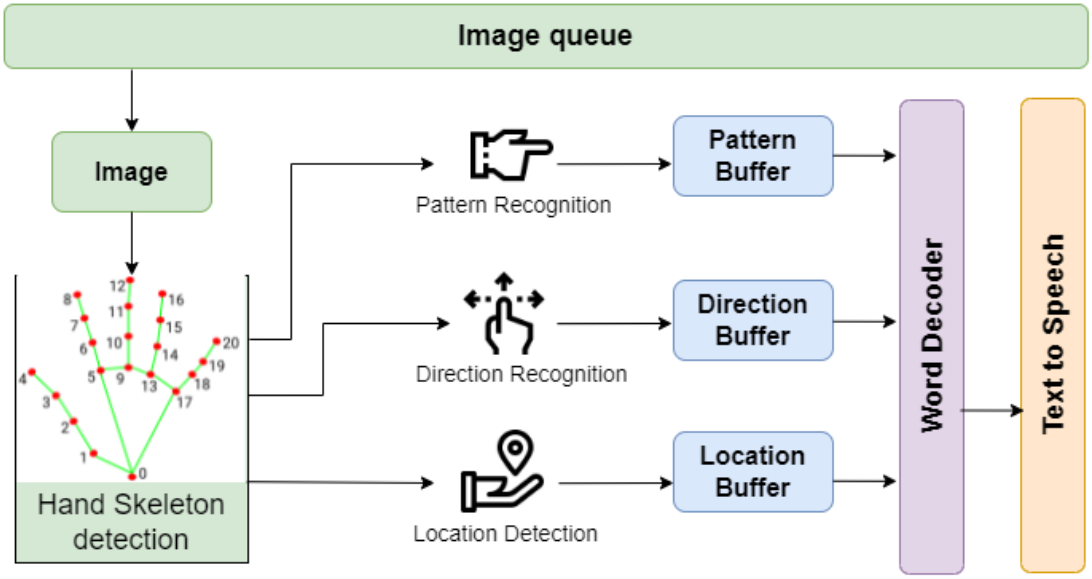


Figure 23: Overview of the new system structure

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 out 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 24), as seen in Figure 23, will tell us the pattern of the hand at the moment it is captured.

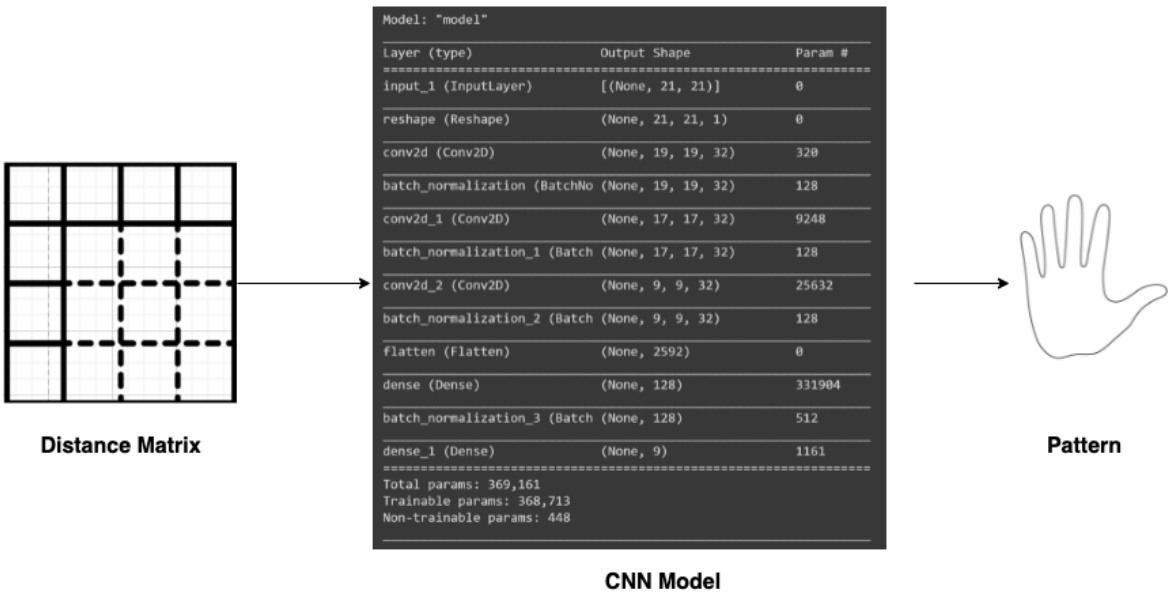


Figure 24: Hand Pattern Recognition Pipe Line

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 25 and Figure 26).

Figure 25: Word "You" (bạn) in sign language

Figure 26: 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 each of 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.

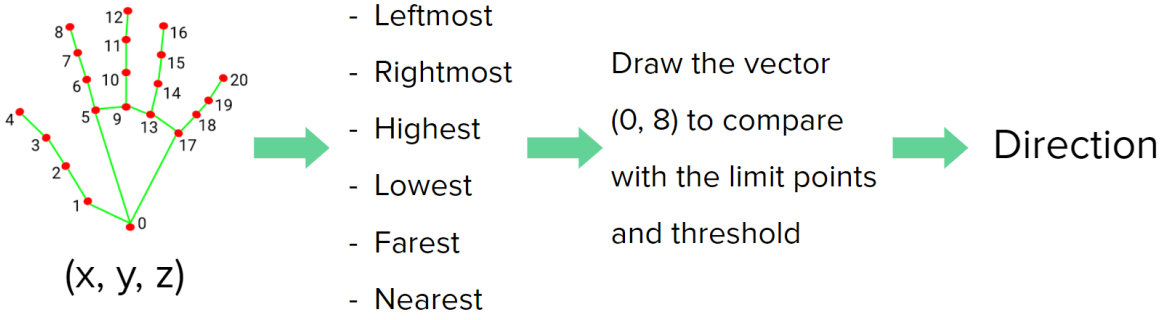


Figure 27: Steps to detect the direction of the hand

For instance, a hand is known to be pointing toward 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.

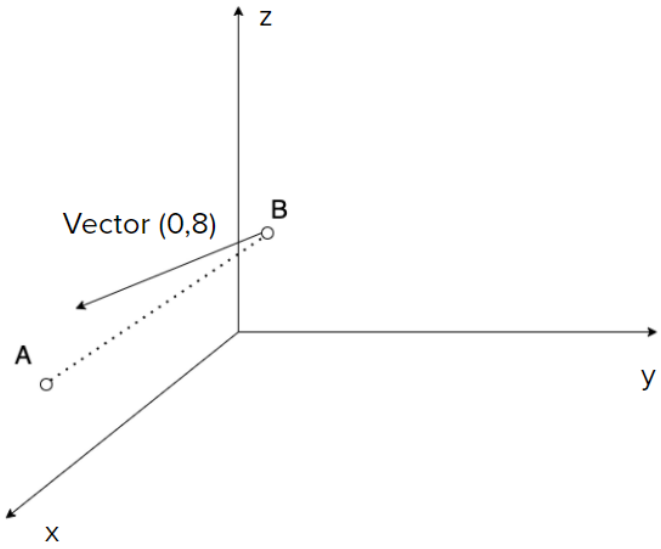


Figure 28: Vector(0, 8) represent the hand pointing toward the left

4.3.3 Location detection

Locations of hand vary, is the hand put at 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 29). 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 the hand and calculate the size of the hand in every frame in order to know whether that hand is getting larger or smaller. Hence, if that hand is smaller than before, it means the hand is getting far away from the camera, and its location is 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 29: 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. Yet, 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 30). 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.

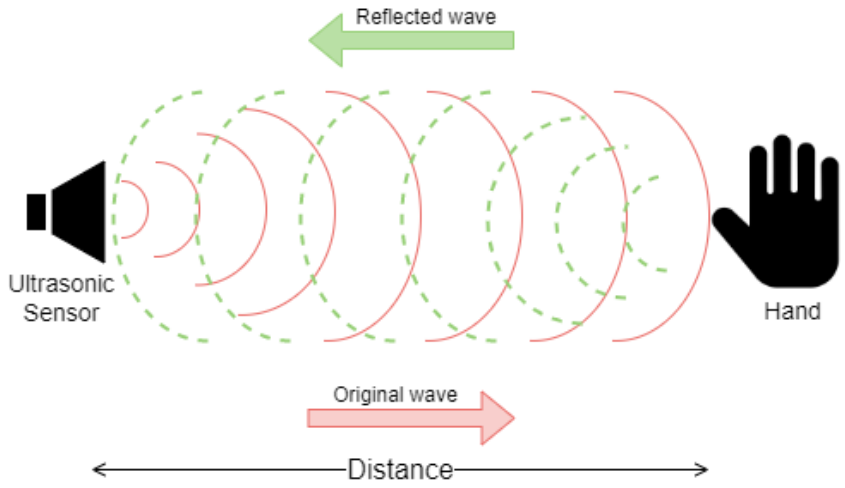


Figure 30: Illustration of how the ultrasonic sensor works

In the thesis, the one we use for this module is the ultrasonic sensor HY-SRF05 (Figure 31), 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 when it comes to measuring from the forehead to the user's waist.

Figure 31: The ultrasonic sensor HY-SRF05

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

4.3.4 Word decoder

As discussed above, there are considerable technical difficulties in implementing the action detection module. 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 32 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.

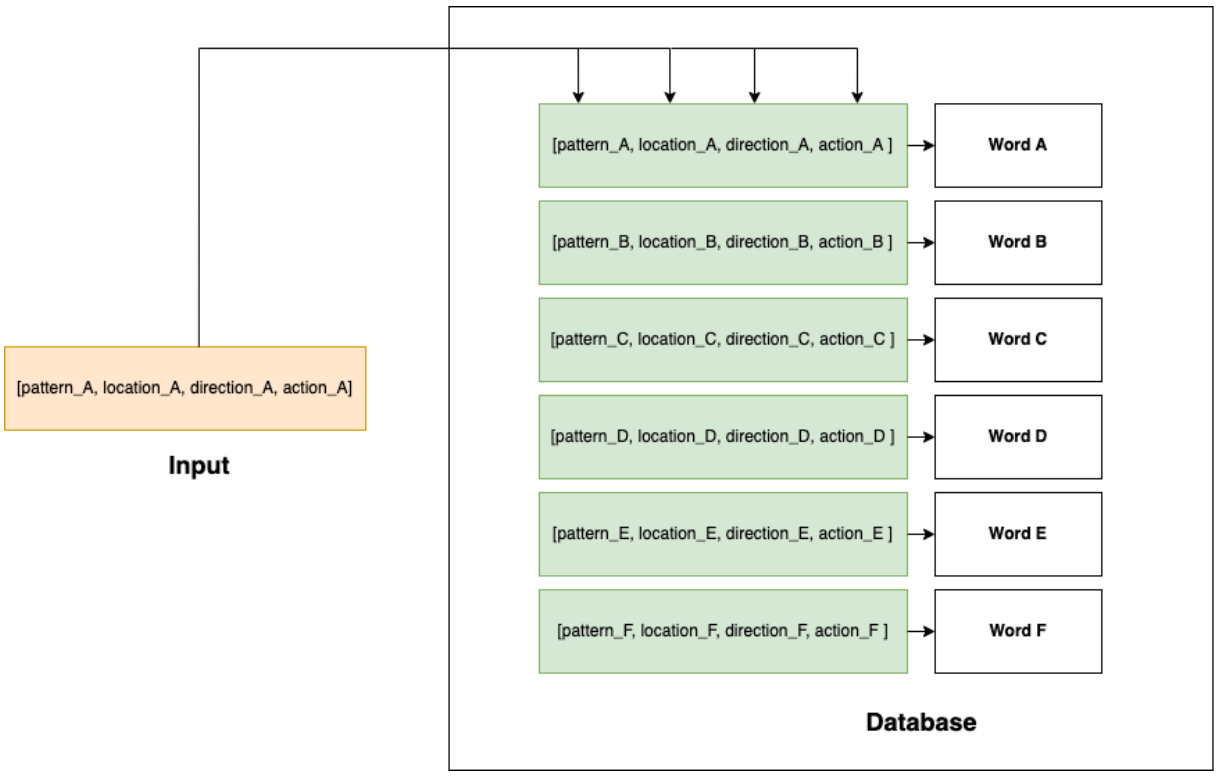


Figure 32: Map one to one data from four component with word in database and get result

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 (33), 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 this thesis. Then, we will get the desired word when going through the processing steps that we will discuss later in this proposal.

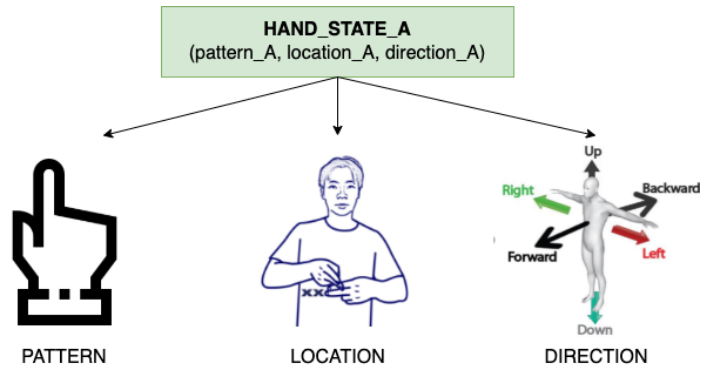


Figure 33: Hand State which construct from pattern, location and direction

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.

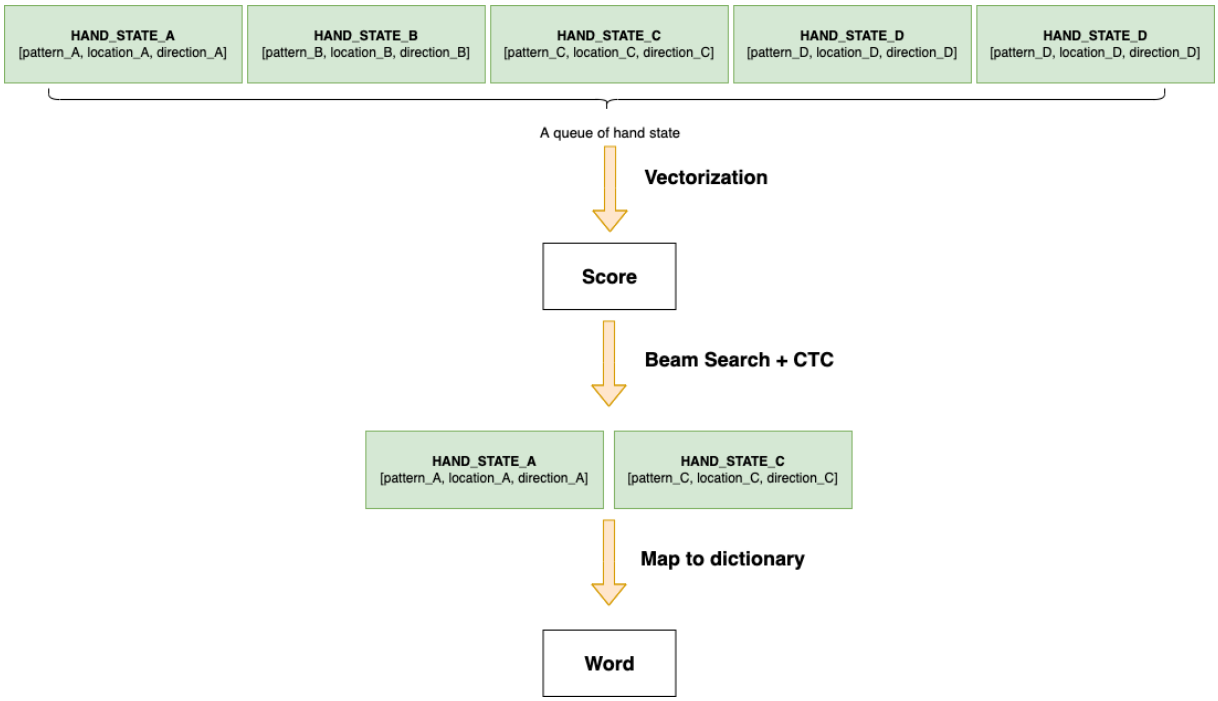


Figure 34: Architechture

Figure 34 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 35 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.

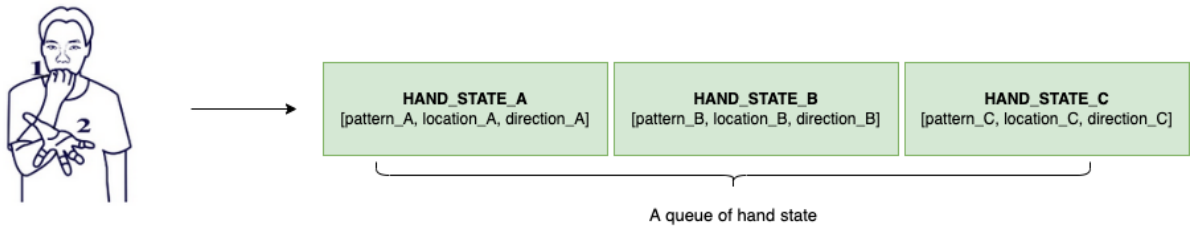


Figure 35: A queue of hand state which get from three component in section ...

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 36:

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.

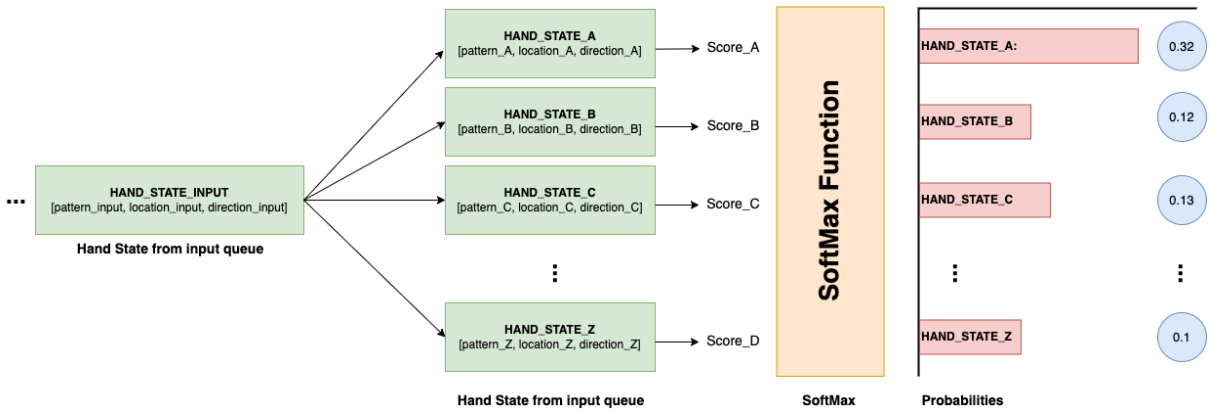


Figure 36: Vectorization

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) 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 has been converted to a MxN matrix, where M is the length of the hand state's database and N is the length of queue.

By using beam search (37), we will get the most likely k hand state from the database. From the image below, we can imagine what happened later during beam search.

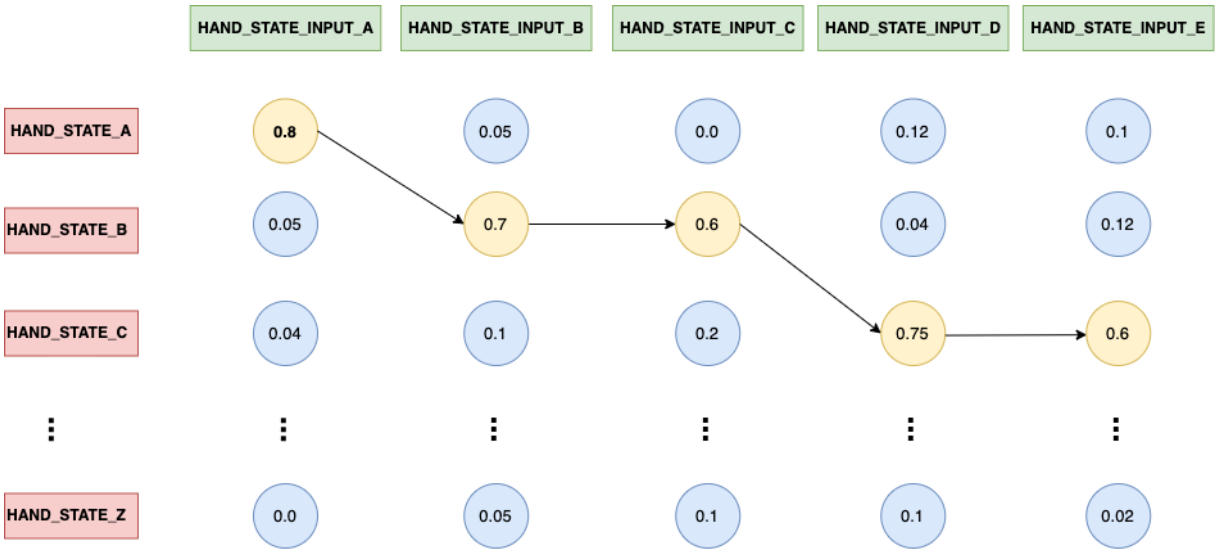


Figure 37: Beam Search

However, as we can see that, 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 CTC algorithm to remove the hand states from infection. For the wrong hand states, in Vectorization step, we will set a threshold to discard these hand states and see it as a blank character. And after all step, after beam search CTC decode, we will get the desired result.

Map to dictionary

Figure 38: 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 section above, but instaed of map with 4-component set, in here, we will map with input retrieved from this previous step. And finally, we will get the word (38) 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 translating process, the application can speak it out loud. Basically, 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 in creating 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. According to Google, Text-to-Speech converts text input into natural human speech audio data. And 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.

In fact, 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 that, "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."

This 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 that we will use is negligible compared to 4 million free characters. Therefore, we decided to apply this Text-to-speech module using Google service in this thesis.

Chapter 5

Upcoming plan

5.1 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 this 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 the perfect one for this role. It is inexpensive and easy to use, making it ideal for our thesis that requires complex functions like image tracking and recognition. Furthermore, it integrates Wi-Fi, traditional Bluetooth, which help us in sending 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. Besides, 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 all together and get the result below.

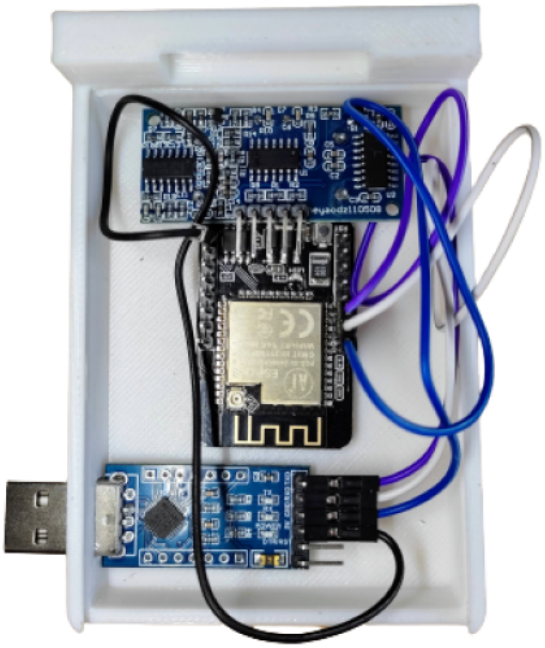


Figure 39: The components inside the camera module prototype

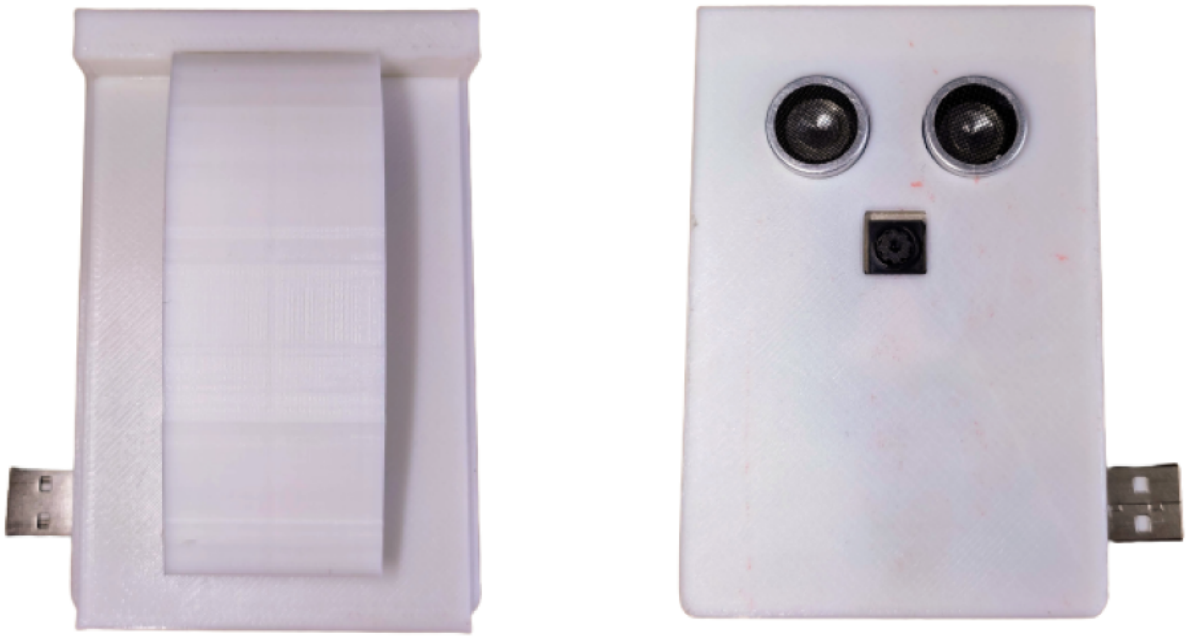


Figure 40: Views of the camera module prototype from the above and under

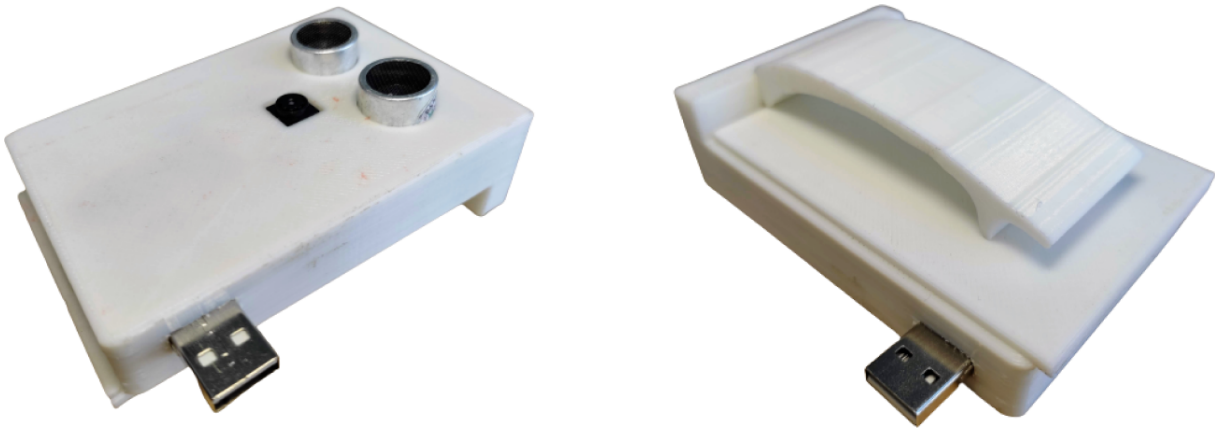


Figure 41: Views of the camera module prototype from the sides

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

5.2 App design

The application must satisfy user experience, and user interface demands. And during the research phase of this thesis, we did design a prototype for the application, including two more features besides the main one. Those features are a sign language dictionary and a learning system. We will talk more about those features in the later sections.

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.

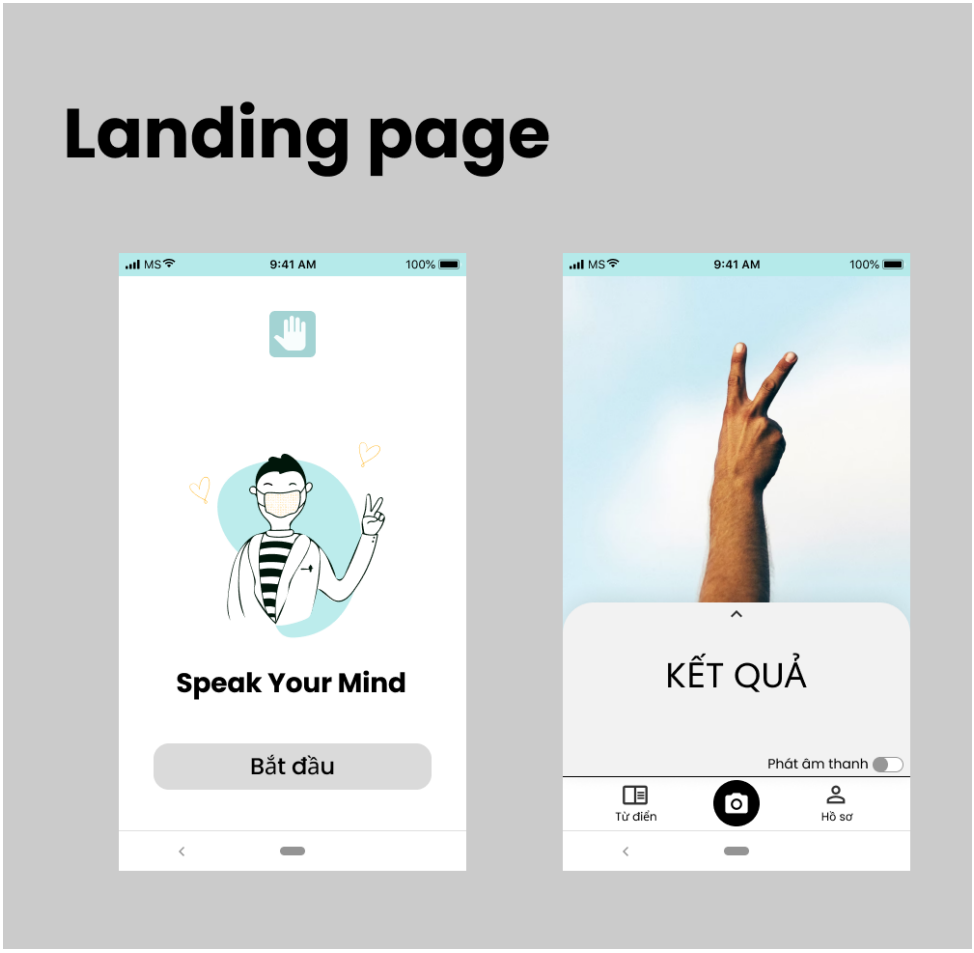


Figure 42: The landing page of the application

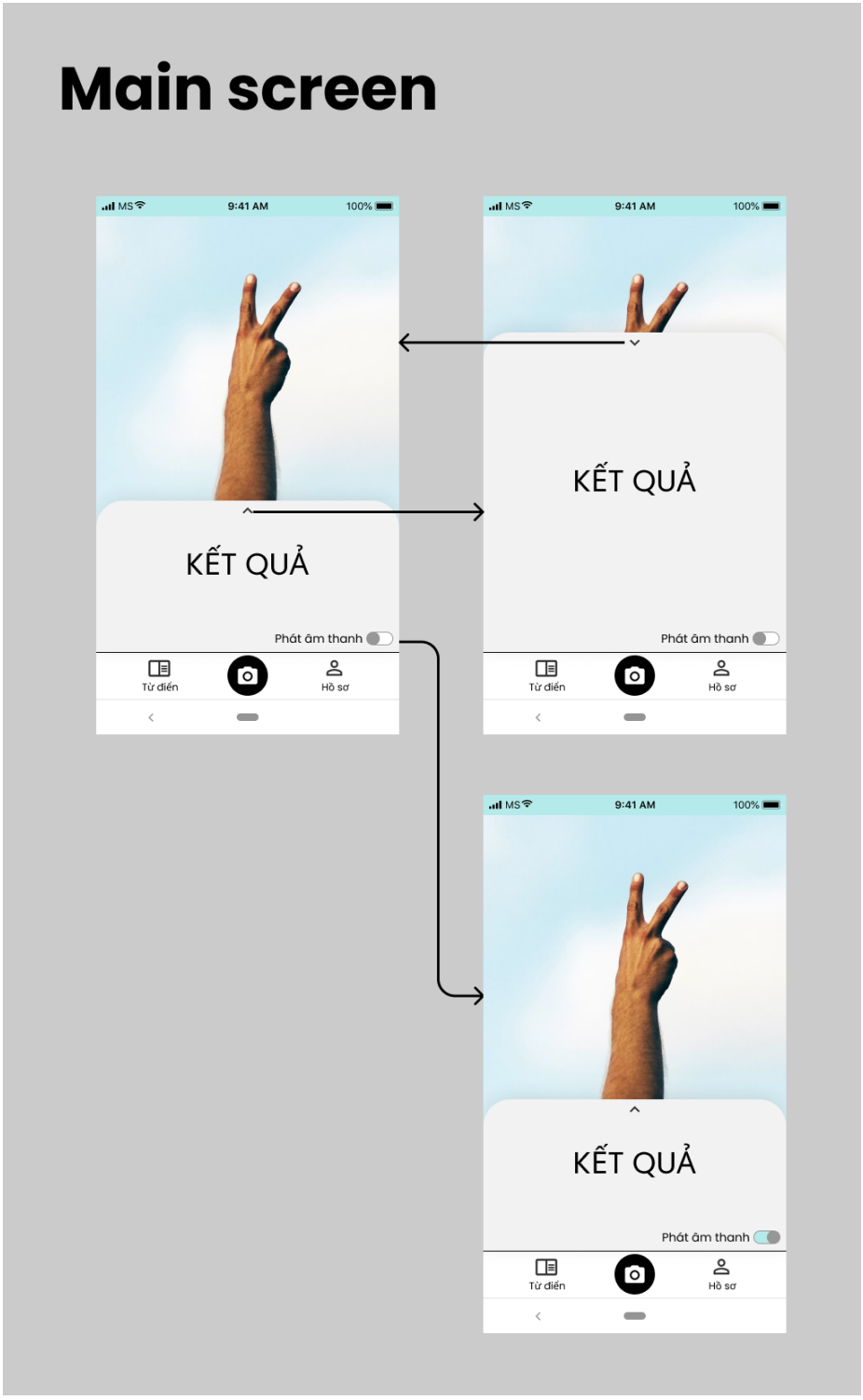


Figure 43: The main screen of the application

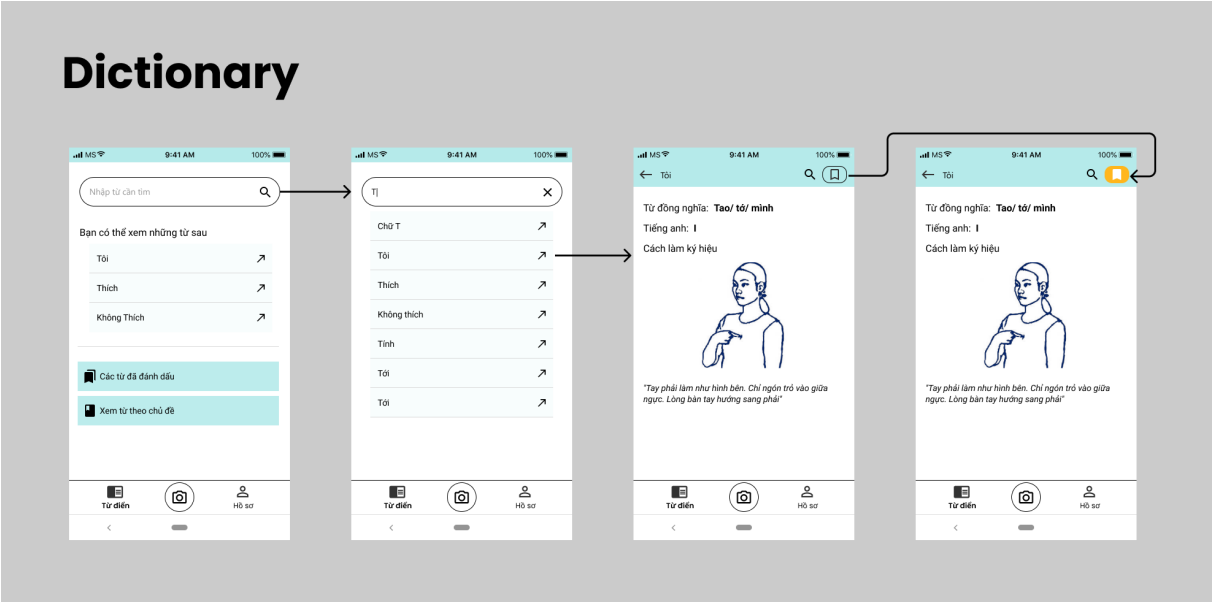


Figure 44: The main screen of the application

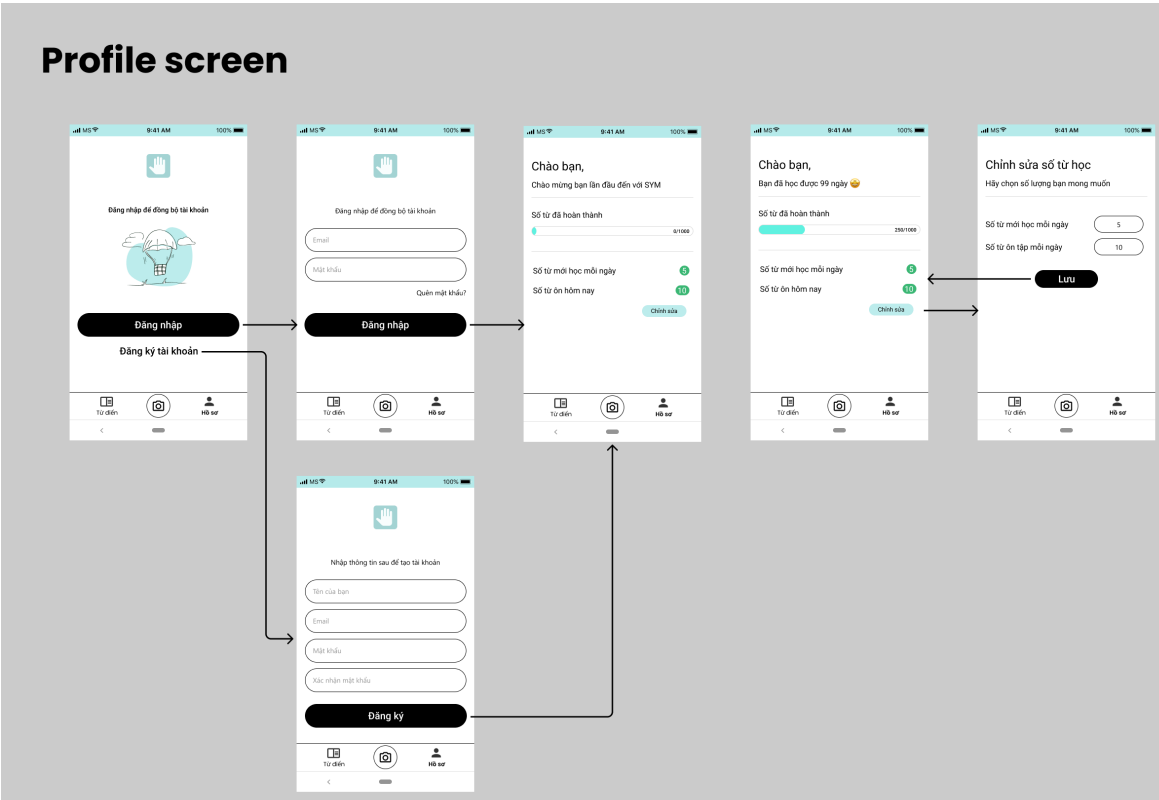


Figure 45: The main screen of the application

5.3 New features and plan for thesis

To expand the group of people using this application, we will implement additional features into the main application to help ordinary people learn and know more about sign language. Those features are a sign language dictionary and a learning system that help people learn sign language more efficiently.

- **Sign language dictionary:** like any other dictionary, but dictionary has another element, which is containing the videos to illustrate the sign of the words.
- **Learning system:** apply the spaced repetition technique, which help people learn much faster and more efficient.

Additionally, to finish this thesis, we break the whole process into three milestones; each will inherit the previous outcome and test them multiple times to ensure they actually work. The table 2 below contains those milestones and tasks needed to complete each milestone.

Milestone	Main tasks	Estimated days
Translator module and basic application	Collect data to train model	4 days
	Build pattern recognition module	7 days
	Train pattern model, test and fix if needed	5 days or more
	Build direction determination module	7 days
	Test and fix if needed	Not estimated
	Build location detection module	7 days
	Build word decoder module	14 days
	Test and fix word decoder	Not estimated
	Build basic app and camera module, test their connection	7 days
	Build server, connect it with the application, fix anything if needed	12 days
Sign language dictionary feature	Collect at least 1000 words in sign language	3 days
	Build basic UI for sign language dictionary	7 days
	Implement a search engine	3 days
	Add dictionary data to the server	1 days
	Build word screen, containing definition and instruction video	3 days
	Test feature and fix if needed	Not estimated
Learning system	Build basic UI for learning system	7 days
	Add learner class	3 days
	Implement algorithm that represent the spaced repetition learning technique	6 days

Table 2: Three milestones and tasks to complete the thesis

Chapter 6

Proposed Thesis Chapters

Chapter 1. Overview. In the chapter, we will present the reason for choosing this topic, **Using machine learning methods to translate sign language into Vietnamese**; Introducing the related works, raising the challenges when doing the research and building the system; then presenting the aim, scope, and structure of the thesis.

Chapter 2. Related works. This chapter contains the associated works that we have referenced when doing the thesis and how they help us build the system.

Chapter 3. Theoretical background. This chapter show all the foundation knowledge that are applied in the thesis, and their assistance in completing the thesis.

Chapter 4. Proposed system. We will analyze the main problem and break it down into pieces to develop the solution for the sub-problems; We also present the design of the system structure, the application, and how they are connected.

Chapter 5. Implementation. We will present the steps to build the system, including setting up the environment, programming the server, application, the physical module, and connecting them so that they can work precisely.

Chapter 6. Result and evaluation. This chapter contains how to use the system, convey the outcome when using it; And evaluate the accuracy rate to ensure it is ready to be used.

Chapter 7. Summary. In the last chapter, we will summarize the whole system, its implementation, result and evaluate them to emphasize the good and bad of the system; Moreover, we will suggest the plan to develop this system in the future.

Chapter 7

Summary

This 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.