THAI SIGN LANGUAGE RECOGNITION FOR EMERGENCY REQUESTS USING DEEP LEARNING

Nut Samaksaman

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Information Technology
Graduate Studies
Thai-Nichi Institute of Technology
Academic Year 2025

Thesis Topic	Thai Sign Language Recognition for Emergency Requests Using Deep Learning
By	Nut Samaksaman
Field of Study	Information Technology
Thesis Advisor	Dr. Pramuk Boonsieng

The Graduate Studies of Thai-Nichi Institute of Technology has been approved and accepted as partial fulfillment of the requirement for the Master's Degree

……Vice President for Academic Affairs
(Assoc. Prof. Dr. Warakorn Srichavengsup)
Month…… Date…… Year……..……

Thesis Committees

……..……Chairman
(Dr. Pikul Vejjanugraha)

……..……Committee
(Dr. Saprangsit Mruetusatorn)

……..……Committee
(Dr. Napaphat Vichaidis)

……..……Advisor
(Dr. Pramuk Boonsieng)

NUT SAMAKSAMAN : THAI SIGN LANGUAGE RECOGNITION FOR EMERGENCY REQUESTS USING DEEP LEARNING. ADVISOR : DR. PRAMUK BOONSIENG, 50 PP.

Sign Language (SL) Sign Language (SL) is essential for communication within the Deaf and hard-of-hearing community. Thai Sign Language (TSL) possesses a unique structure, distinct from Chinese and American Sign Languages. It operates under its own grammatical rules, frequently utilizing simplified sentence structures for effective communication. Recent research in TSL recognition has explored various methodologies, including the interpretation of fingerspelling and the translation of sequential signs into text. This study aims to interpret continuous TSL into simple sentences to bridge the communication gap between Deaf and hearing individuals. Given the vast vocabulary of the Thai language and limited research time, this study introduces a TSL recognition system designed specifically for emergency scenarios, with data sampled from the most common emergency requests. Real-time sign language recognition (SLR) systems typically use computer vision to translate sequences of hand movements and body postures into text or speech instantaneously. The proposed system utilizes MediaPipe Holistic for hand tracking, integrated with Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), and Gated Recurrent Unit (GRU) architectures to facilitate robust sequence recognition. Developed using OpenCV, the system translates individual signs into text and displays them as simple sentences following Thai grammatical rules (verb + noun or noun + verb).

Keyword: Sign Language, Open-cv, long short-term memory, BiLSTM, MediaPipe Holistic, GRU, RNN, Deep Learning

Graduate Studies Student's Signature……...
Field of Study Information Technology Advisor's signature……...
Academic Year 2025

Acknowledgement

I would like to express my sincere gratitude to all those who supported me. In particular. I am deeply grateful to Dr.Pramuk Boonsieng for his invaluable guidance throughout my year of research; his advice has had a significant impact on this study. And deeply grateful to the members of my defense committee for their time and expertise. I would especially like to thank Dr.Pikul Vejjanugraha Dr.Saprangsit Mruetusatorn and Dr.Napaphat Vichaidis for their encouragement and research guidance. Their feedback was instrumental in helping me navigate the complexities of this study.

Nut Samaksaman

Table of Contents

Page

Acknowledgement iv

Table of Contents v

Table of Tables vii

Table of Figures viii

Chapter

1 Introduction 1

Background and Rationale 1

Research Objectives 2

Scope of Research 2

2 Related Theories and Research 3

Neural Network 3

AI Algorithms for Sign Language Recognition 8

Sign Language 9

Thai Sign Language 10

Object Detection 13

Feature Extraction 13

MediaPipe Holistic 13

Model Evaluation 20

Literature Review 20

Table of Contents (continued)

Chapter		Page
3	Methodology	24
	Research Plan	24
	Research Strategy	24
	Data Collection	27
	Collecting Video Data for Sequential Thai Sign Language	30
	Feature Extraction	32
	Model Architect	34
	Improve Model Accuracy	35
	Model Training	36
4	Experiment and Results	37
	Experimental Setup	37
	Experimental Results	37
5	Conclusion	45
References		47
Bio Data		50

Table of Tables

Tables Page

2.1 Thai Sign Language Fingerspelling 11

2.2 Coordinate Indices of 33 Pose Landmarks 18

2.3 Index of 21 Coordinates Derived from the Hand Model 19

2.4 Summary of research papers on Real Time Sign Language Recognition using Mediapipe Holistic for Feature Extraction. 22

3.1 Research Plan 25

3.2 17 vocabulary words used as sample data, including their English meanings. 28

4.1 A table summarizing software and hardware requirements 37

4.2 Experiment Results from adjusting the number of LSTM and dense and layers 41

4.3 Results of experiment models 41

4.4 F1- score and recall from an experiment 42

4.5 Raw trainning data set combined from individual vocacabularies following thai grammar rule 44

Table of Figures

Figures Page

2.1 Activation Function in Neural 4
2.2 Graph plot from ReLU activation function 5
2.3 Graph of Sigmoid activation function 6
2.4 Graph of Tanh activation function 7
2.5 American Sign Language (ASL) Fingerspelling Alphabet 10
2.6 Visualization of 468 3D landmark coordinates for face tracking. 15
2.7 Visualization of 33 3D landmark coordinates for pose tracking 17
2.8 Distribution of 21 landmark points on the hand model 19
3.1 Simplified Block Diagram of Research Methodology 26
3.2 Distribution of Participants by Gender 29
3.3 Number of Participants by Impairment Status 29
3.4 Search Results for the word "รถดับเพลิง" th-sl.com 30
3.5 An Example from extracted frames collected from a failure data set, identified that similar sign language hand shapes or movements occur across several sequential frame ranges within a video. 32
3.6 An example of a video collected using OpenCV consisted of 9 frames for one video. 32
3.7 Example of input shape detected landmarks from one image frame. The coordinates comprise of 21*(x, y, z) points for left-hand and 21*(x, y, z) points for right-hand 33
3.8 Neural Network Architecture composed of an LSTM layer and two additional hidden layers 34
4.1 Training and Validation Loss Trained model with 100 epochs 39
4.2 Training and Validation Loss Trained model with 300 epochs 39
4.3 Training and Validation Loss Trained model with 600 epochs 40
4.4 Confusion matrix of predictions on the validation set generated by the LSTM model 42

Table of Figures (continued)

Figures Page

4.5 Confusion matrix of predictions on the validation set generated by the RNN model 43

4.6 Confusion matrix of predictions on the validation set generated by the GRU model 43

4.7 The proposed model detects individual hand signs capturing two actions and translates them into a single, simple sentence following Thai grammar structures 44

Chapter 1
Introduction

1.1 Background and Rationale

Each country has its own sign language (SL), which differs from others. Sign language is represented by varying hand gestures, body movements, and facial expressions. Due to the vastness of Thai Sign Language (TSL), this research defines specific words and sentences that Thai native hearing-impaired individuals need for emergency help, translating them into text to facilitate communication with people who do not know sign language. There have been enormous advancements in technology and research related to sign language in the past century. Recognition has been widely researched and implemented in other languages, such as American Sign Language (ASL) or Chinese Sign Language (CSL), using machine learning algorithms, convolutional neural networks (CNNs), deep learning, and various other techniques. People with disabilities in Thailand are estimated to account for roughly 3% to 6% of the total population [1]. According to 2024 data, there are approximately 423,483 individuals with hearing impairments. While Thai Sign Language (TSL) is recognized as the national language for the Deaf community, some individuals may struggle with it due to a lack of exposure or delayed access to education.

Some public resources and Thai Sign Language data are available at www.th-sl.com, where approximately 3,000 words were gathered by the National Association of the Deaf in Thailand [2]. However, Thai Sign Language (TSL) databases are significantly smaller than those for American or Chinese Sign Languages. This disparity can lead to accessibility issues and a lack of educational resources for the Deaf community in Thailand. As is well known, the total number of words in a language can exceed one million. The Oxford English Dictionary contains over 600,000 words, while the Royal Institute Dictionary the official prescriptive authority for the Thai language contains approximately 40,000 entries. Due to the vastness of the Thai lexicon, this research focused on sampling data from words used specifically in emergencies to help the unimpaired understand them more effectively.

Key challenges in sign language recognition (SLR) systems include maintaining real-time performance, particularly with continuous sign language. A primary difficulty lies in accurately identifying hand movements from video data amidst variations in signers, lighting conditions, background clutter, and spatial complexity. Furthermore, developing continuous SLR systems presents a significant computational burden, requiring high processing power to manage temporal data. While most recent studies have focused on static gestures, where Convolutional Neural Networks (CNNs) have excelled over the past decadethe current frontier requires systems to interpret real-time sequences. Consequently, modern dynamic SLR systems increasingly utilize deep learning models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)networks, which are specifically designed to process sequential video data.

1.2 Research Objectives

This system focused on high-accuracy recognition of Thai Sign Language for emergency requests. It employed LSTM, RNN, and GRU models to detect individual hand signs in real time, translated them into text, and labeled individual vocabulary items as simple sentences following Thai grammar rules.

1.3 Scope of Research

1.3.1 Thai Sign Language (TSL) comprises a vast vocabulary. Due to the time constraints of this research, the dataset sampling focuses specifically on TSL signs used by the hearing impaired to request assistance during emergencies.

1.3.2 Sign language recognition systems typically require high-performance computing and significant time for model training. This research focuses on improving model accuracy and ensuring the system interprets signs into text that adheres to Thai Sign Language grammar and structure.

Chapter 2
Related Theories and Research

2.1 Neural Network

Neural network design involves creating a computational architecture analogous to the human brain to learn from data and perform complex tasks. This process includes defining the network's structure (layers and neurons), selecting activation functions (such as ReLU or sigmoid), initializing weights, training via backpropagation, and validating performance before deployment. As an iterative process, it allows the network to recognize intricate patterns and generalize to unseen data. Essentially, a neural network architecture is the formal design of an Artificial Neural Network (ANN), a mathematical model that defines how artificial neurons (nodes) are organized, connected, and how information flows through the system [3].

2.1.1 Essential Components of Neural Network Architecture

1) Input layer: This layer receives raw input data, such as image pixel values or numerical features from a dataset. Each neuron in this layer corresponds to a single feature [4].

2) Hidden layers: These consist of one or more layers between the input and output, where the network performs its computations. Each neuron processes data from the previous layer using a weighted sum, a bias term, and an activation function. Adding more hidden layers allows the network to learn increasingly complex patterns.

3) Weights and Biases: Weights represent the strength of connections between neurons and are adjusted during training to minimize prediction errors. Biases are constant values added to the weighted sum, allowing the model to fit data more accurately [4].

4) Activation Functions: An activation function is a mathematical tool that determines a neuron's output. Its primary purpose is to introduce non-linearity, enabling the network to learn complex patterns. Without it, even a deep neural network would function like a simple linear regression model, unable to handle tasks like image classification. Operationally, each neuron calculates a weighted sum of its inputs and adds a bias; this result is then passed through the activation function to

determine whether the neuron "fires" and what information is transmitted to the next layer. Standard Neural Network activation functions are listed below.

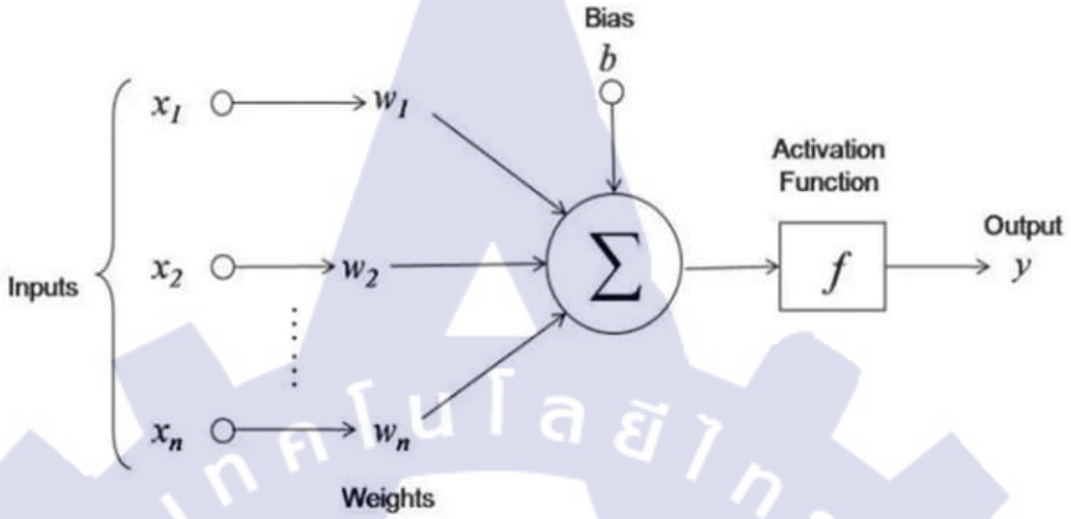


Figure 2.1 Activation Function in Neural

(1) ReLU (Rectified Linear Unit)

The ReLU activation function is widely used in neural networks and is defined by the formula:

$$f(x) = \max(0, x) \quad (2.1)$$

Where:

x is the input to the neuron

The function returns x if x is greater than 0

If x is less than or equal to 0, the function returns 0

It outputs the input directly if it is positive; otherwise, it outputs zero. ReLU is favored for its computational efficiency, its ability to introduce non-linearity, and its role in mitigating the vanishing gradient problem by providing a constant gradient for all positive inputs. Positive Input: If the input value (x) is greater than zero, ReLU outputs the input value unchanged. Non-Positive Input: If the input value

(x) is zero or negative, ReLU outputs zero. The most widely used activation function for hidden layers, especially in convolutional neural networks (CNNs), due to its computational efficiency and its ability to combat the vanishing gradient problem. The simple conditional operation makes ReLU very fast to compute on GPUs and CPUs, speeding up both training and inference [4].

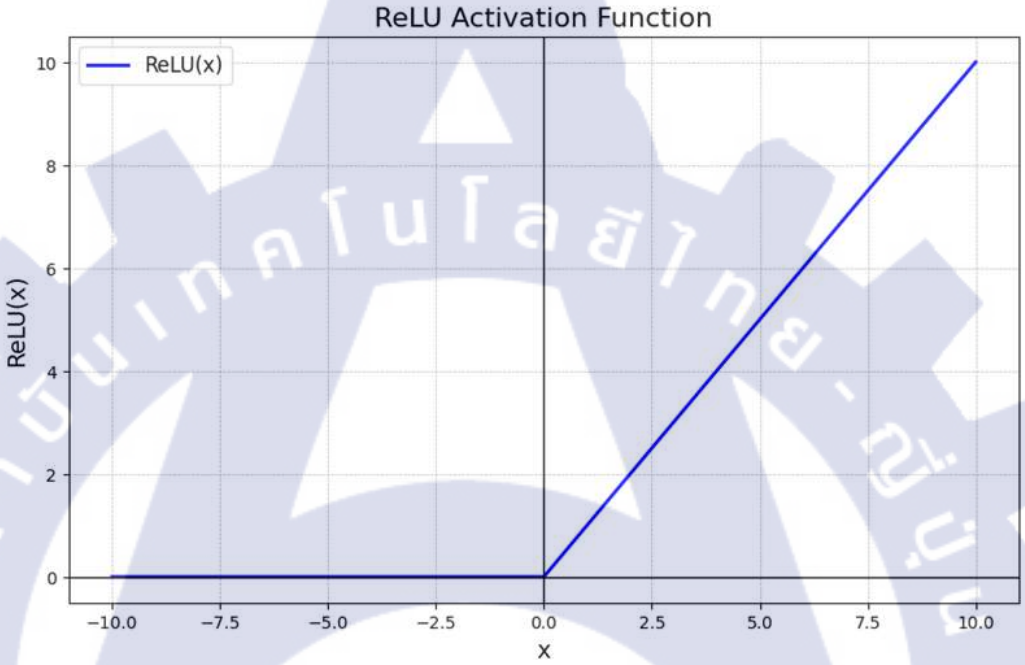


Figure 2.2 Graph plot from ReLU activation function

(2) Sigmoid

A sigmoid function is a mathematical function producing an 'S'-shaped curve—most commonly the logistic function—defined by the formula σ(x) = 1 / (1 + e^(-x)). It squashes any real-valued input into an output range of 0 to 1, making it useful for converting outputs into probabilities. It is frequently used as an activation function in neural networks for binary classification and in logistic regression to model nonlinear relationships in data.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Where:

x: is the input variable.

e: is Euler's number, approximately 2.718.

The function takes any real number as an input and outputs a value that is always in the range (0, 1). The sigmoid is a mathematical function that maps any real-valued number into a value between 0 and 1. It is particularly useful in scenarios where we need to convert outputs into probabilities. The sigmoid function is commonly used as an activation function in machine learning and neural networks for modeling binary classification problems, smoothing outputs, and introducing non-linearity into models.

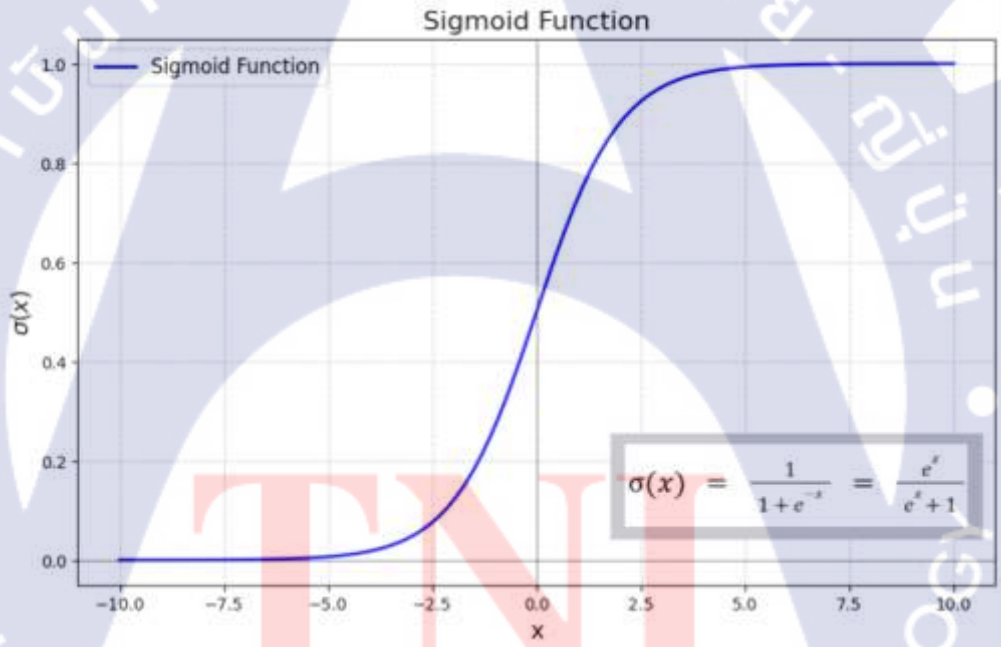


Figure 2.3 Graph of Sigmoid activation function

(3) Tanh (Hyperbolic Tangent)

Tanh (hyperbolic tangent) is a mathematical function commonly used as an activation function in neural networks. It maps input values to a range between -1 and 1, producing an S-shaped curve. Unlike the sigmoid function, it is zero-centered, which often facilitates faster convergence in deep learning.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

Tanh is often used in hidden layers of recurrent neural networks (RNNs), as its zero-centered output helps with gradient flow. Its ability to output values between -1 and 1 helps to center the data, which can improve the learning process for subsequent layers in a network. It is particularly effective when data has been normalized to have a mean of zero.

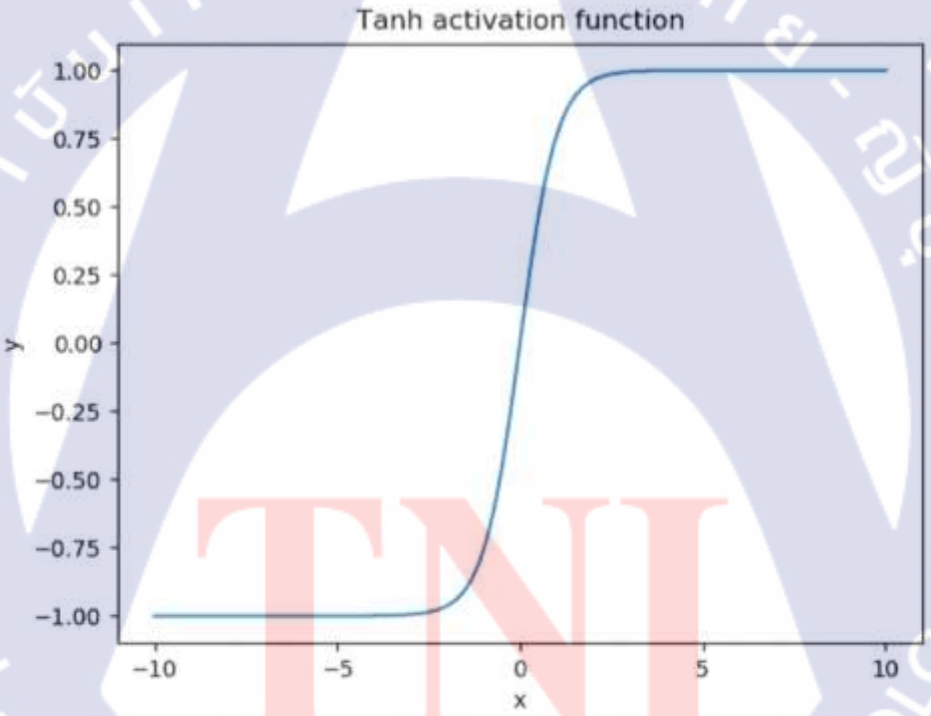


Figure 2.4 Graph of Tanh activation function

(4) The SoftMax function

The SoftMax function is a mathematical operation that transforms a vector of raw scores (logits) into a probability distribution, where each value represents the probability of a specific class. Used primarily in the output layer of multi-class classification neural networks, it ensures all outputs are between 0 and 1 and sum to 1.

2.2 AI Algorithms for Sign Language Recognition

2.2.1 Convolutional Neural Networks (CNNs): A specialized class of deep learning models designed to process data with a grid-like topology, such as images, videos, and time-series data.

1) Recurrent Neural Networks (RNNs) are deep learning models designed to process sequential data, such as text or time series; they incorporate 'memory' through feedback loops, allowing information from previous steps to influence current processing.

2) Support Vector Machines (SVMs) are supervised learning algorithms that classify data effectively for classification and regression using "support vectors.

3) Hidden Markov Models (HMMs) are a popular choice for implementation in Python, specifically via the hmmlearn library. It provides classes for various HMM types—such as CategoricalHMM, GaussianHMM, and GMMHMM—which can be used to model the temporal evolution of sign language gestures. These models are particularly effective for recognizing continuous sign language.

4) YOLO, which stands for "You Only Look Once," is a prominent real-time object detection algorithm in machine learning and computer vision. It revolutionized object detection by simplifying the process and achieving a balance between speed and accuracy.

5) Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN)designed to learn long-term dependencies in sequential data by using gated units to control information flow. These networks address the vanishing gradient problem, which hinders standard RNNs from retaining information over long sequences. LSTMs utilize an internal memory mechanism that allows them to

remember critical information across extended data sequences, making them ideal for tasks like time series forecasting, natural language processing (NLP) and speech recognition.

6) A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that enhances the speed performance of LSTM networks by simplifying the structure with only two gates: the update gate and the reset gate.

7) Bidirectional Long Short-Term Memory (BiLSTM) is an extension of LSTM that processes input sequences in both forward and backward directions. By running two separate LSTMs on the same data, a BiLSTM captures a richer context by leveraging both past and future information.

2.3 Sign Language

Sign language is a visual language used primarily by individuals who are deaf or hard of hearing. It utilizes hand gestures, facial expressions, and body movements to communicate. To build a Sign Language Recognition (SLR) system, datasets are typically categorized into two main types based on the nature of the signs they capture.

2.3.1 Static Sign in Sign Language

Static sign language consists of fixed hand poses and body gestures without movement. Researchers study static signs to translate these visual inputs into text or speech. This involves identifying specific hand shapes, numbers, or words through precise handshapes and positions, such as the gestures used for letters. For example, the American Sign Language (ASL) Alphabet is a system of 26 hand signs corresponding to the letters of the English alphabet. It is most often used to fingerspell proper nouns, such as names of people and places, or to spell out English words that do not have their own unique sign, as shown in Figure 2.5.

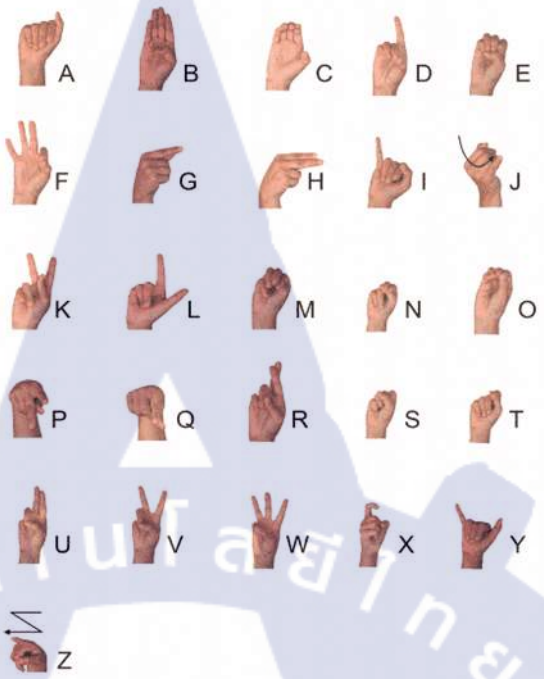


Figure 2.5 American Sign Language (ASL) Fingerspelling Alphabet

2.3.2 Dynamic Signs in Sign Language

Dynamic sign language (DSL) refers to the active motion of the hands, face, and body that forms a sign, rather than a static pose. Examples of dynamic signs are represented as processes rather than single points; for instance, the sign for 'thinking' involves the index finger moving from the temple outwards, while the sign for 'hello' involves the hand moving away from the forehead. These contrast with static signs, which consist of fixed handshapes without movement. The WLASL dataset is one of the largest publicly available video collections for word-level American Sign Language recognition.

2.4 Thai Sign Language

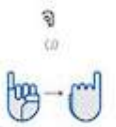
Historically, because the hearing impaired had to practice communicating by guessing based on body language or lip-reading and responding vocally, Thai Sign Language (TSL) developed a unique grammatical structure that differs from spoken Thai. Additionally, the fingerspelling used in TSL is derived from American Sign Language (ASL). When creating TSL fingerspelling, designers mapped ASL

handshapes to Thai letters based on phonetic similarities as shown in the examples below [5]:

Table 2.1 Thai Sign Language Fingerspelling

ก (Letter)	phonetically similar to K (Letter)	
ต (Letter)	phonetically similar to T (Letter)	
ส (Letter)	phonetically similar to S (Letter)	
พ (Letter)	phonetically similar to P (Letter)	
ห (Letter)	phonetically similar to H (Letter)	
บ (Letter)	phonetically similar to B (Letter)	
ร (Letter)	phonetically similar to R (Letter)	
ว (Letter)	phonetically similar to W (Letter)	
ด (Letter)	phonetically similar to D (Letter)	

Table 2.1 Thai Sign Language Fingerspelling (continued)

ฟ (Letter)	phonetically similar to F (Letter)	
ล (Letter)	phonetically similar to L (Letter)	
จ (Letter)	phonetically similar to J (Letter)	
ย (Letter)	phonetically similar to Y (Letter)	
ม (Letter)	phonetically similar to M (Letter)	
น (Letter)	phonetically similar to N (Letter)	
อ (Letter)	similar phonetic with A (Letter)	
ง (Letter)	similar phonetic with ng (Letter)	

Thai fingerspelling is based on the English (ASL) alphabet, but their structures differ. Thai has unique signs for vowels and tones, giving Thai Sign Language a linguistic structure similar to the spoken Thai language.

2.5 Object Detection

Object detection in AI refers to identifying and localizing specific objects within an image or video, typically by drawing bounding boxes around them and labeling them with their respective classes. In sign language recognition systems, object detection plays the critical role of identifying and locating hands and other relevant body parts within a frame, allowing the system to understand visual cues.

For example, systems often use deep learning models to classify hand gestures and body movements that represent specific signs in real-time video feeds. A common model for this task is YOLO (You Only Look Once), which is known for its speed and accuracy. When an image of a gesture is sent to a trained YOLO model, its convolutional neural network (CNN) layers extract features to accurately identify and classify the objects present.

2.6 Feature Extraction

In Sign Language Recognition (SLR), feature extraction identifies and quantifies relevant information from raw visual data (images or videos). This process represents signs for classification, with methods varying based on whether signs are static or dynamic and which specific features are prioritized.

2.7 MediaPipe Holistic

MediaPipe offers various machine learning models, such as face detection, multi-hand tracking, and object detection and tracking. It primarily acts as a mediator for handling model implementation across different platforms, which allows developers to focus on experimenting with models rather than managing system complexities.

The MediaPipe Holistic Landmark task coordinates pose, face, and hand landmarks to create a comprehensive map of the human body. This allows it to recognize human faces, body movements, and actions. MediaPipe Holistic is a specific Google AI solution that combines individual models for pose, face, and hand perception into a unified pipeline, enabling simultaneous detection and tracking of all three body parts. When an image or video frame is fed into a MediaPipe model, the detected landmarks are stored in a data structure (such as a list of objects) containing x,

y, and z coordinates. These represent the 3D coordinates in meters for each landmark point. While most models output these three values, Pose landmarks also include a visibility score. For simultaneous full-body tracking, the system identifies 543 landmarks in total: 33 for the pose, 468 for the face, and 21 for each hand. MediaPipe Holistic integrates separate models for pose, hand, and face tracking. While the pose model utilizes a fixed lower-resolution input (256x256) to maintain efficiency, this resolution is often insufficient for the hand and face models, which can lead to reduced accuracy in those regions [6].

2.7.1 Face Landmarks

Face landmark detection is a computer vision task used to identify and track key points (or landmarks) on a human face, such as the corners of the eyes, the tip of the nose, and the edges of the mouth. This process leverages machine learning models to map facial structure and track movements in real-time, allowing for the analysis of facial expressions and expressions [7], as shown in Figure 2.6, which visualizes the 468 3D landmark coordinates for face tracking using MediaPipe Holistic.

1) Face detection involves identifying the coordinates and dimensions of human faces within digital images or video frames. This is typically achieved through object detection frameworks, ranging from traditional Haar cascades to modern deep learning architectures like Convolutional Neural Networks (CNNs).

2) Landmark localization involves using specialized machine learning models to detect key facial points. Historically, methods ranged from holistic approaches that model the entire face to Constrained Local Models (CLMs) focused on local features. Currently, deep learning regression methods like CNNs are the industry standard.

3) The system generates a set of facial landmarks within the detected bounding box. These landmarks are represented in either 2D (x, y) coordinates. Advanced frameworks, such as Google's MediaPipe, also predict 'blendshape scores' to infer detailed facial expressions alongside the 3D mesh.

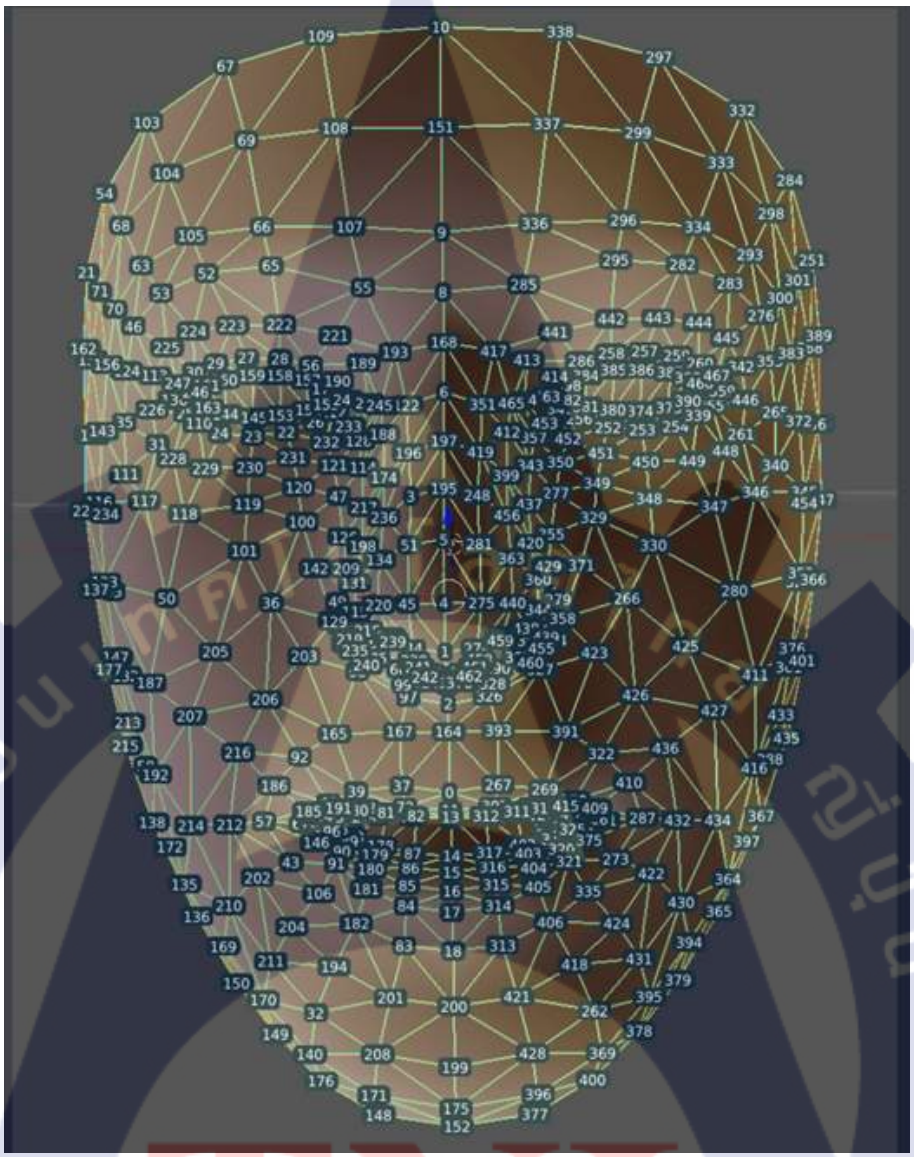


Figure 2.6 Visualization of 468 3D landmark coordinates for face tracking.

2.7.2 Pose Landmarks

MediaPipe Holistic is a comprehensive solution that integrates pose, face, and hand landmark detection into a single model, offering full-body tracking from continuous image streams. The pose landmarks, in particular, serve as the core component of this system. The pipeline operates in the following sequential steps to ensure high-resolution accuracy across the entire body [7], as shown in Figure 2.7, which visualizes the 33 3D landmark coordinates for pose tracking using MediaPipe Holistic.

1) Initial Pose Estimation: MediaPipe Holistic first utilizes a pose detection model to identify the presence of human bodies in the input image or video frame and estimates initial pose keypoints.

2) Region of Interest (ROI) Derivation: Based on these initial pose keypoints, it derives regions of interest for the face, hands, and the overall pose.

3) Refined Pose Landmark Detection: While the face and hands are then processed by dedicated models for more detailed landmark detection, the pose component focuses on tracking and identifying 33 key body landmark locations. These landmarks correspond to various body parts, such as:

(1) Nose, eyes, ears, shoulders, elbows, wrists.

(2) Hips, knees, ankles, heels, big toes, pinky toes.

4) Output: The result is a set of 3D coordinates for each detected pose landmark, representing their position in image coordinates and in 3-dimensional world coordinates. These landmarks can then be used for various applications, including posture analysis, movement categorization and gesture recognition.

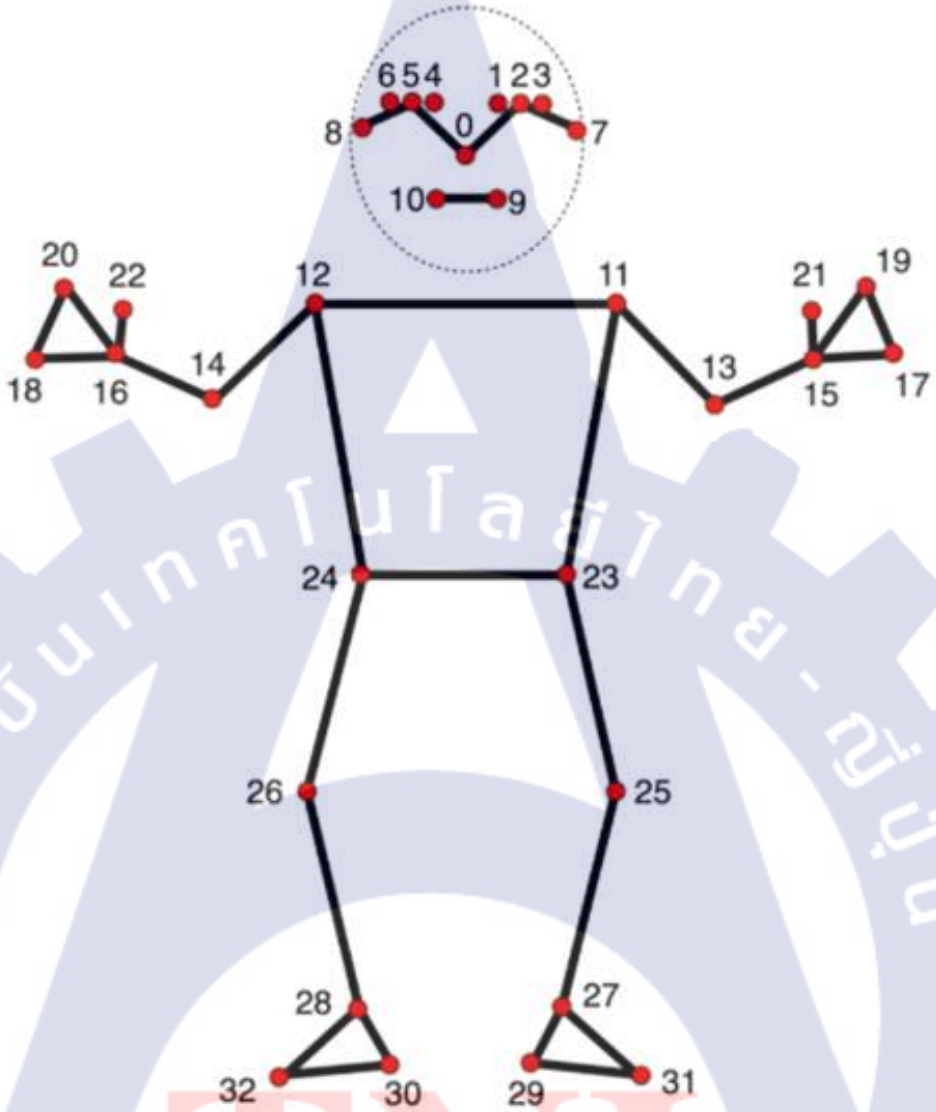


Figure 2.7 Visualization of 33 3D landmark coordinates for pose tracking

Table 2.2 Coordinate Indices of 33 Pose Landmarks

0 – Nose	8 - right ear	16 - right wrist	25 - left knee
1 - Left eye (inner)	9 - mouth (left)	17 - left-pinky	26 - right knee
2 - Left eye	10 - mouth (right)	18 - right pinky	27 - left ankle
3 – Left eye (outer)	11 - left shoulder	19 - left index	28 - right ankle
4 - right eye (inner)	12 - right shoulder	20 - right index	29 - left heel
5 - right eye	13 - left elbow	21 - left thumb	30 - right heel
6 - right eye (outer)	14 - right elbow	22 - right thumb	31 - left foot index
7 - left ear	15 - left wrist	23 - left hip	32 - right foot index
8 - right ear	16 - right wrist	24 - right hip	

2.7.3 Hand Landmarks

Hand landmarks are a set of 21 key points on the hand, identified using computer vision models to provide a precise, three-dimensional representation of hand and finger poses. These landmarks are used in applications like gesture recognition, augmented reality, and robotics. In MediaPipe Holistic, each hand is represented by 21 3D landmarks, which provide a detailed understanding of its geometry and pose. The Holistic model processes the hands, face, and body simultaneously, combining their respective landmark models to provide a holistic view of human motion. The "21 landmarks of the hand" refer to the 21 key points identified by computer vision models like Google's MediaPipe Hand Landmarker [8], which track their precise location as shown Figure 2.8.

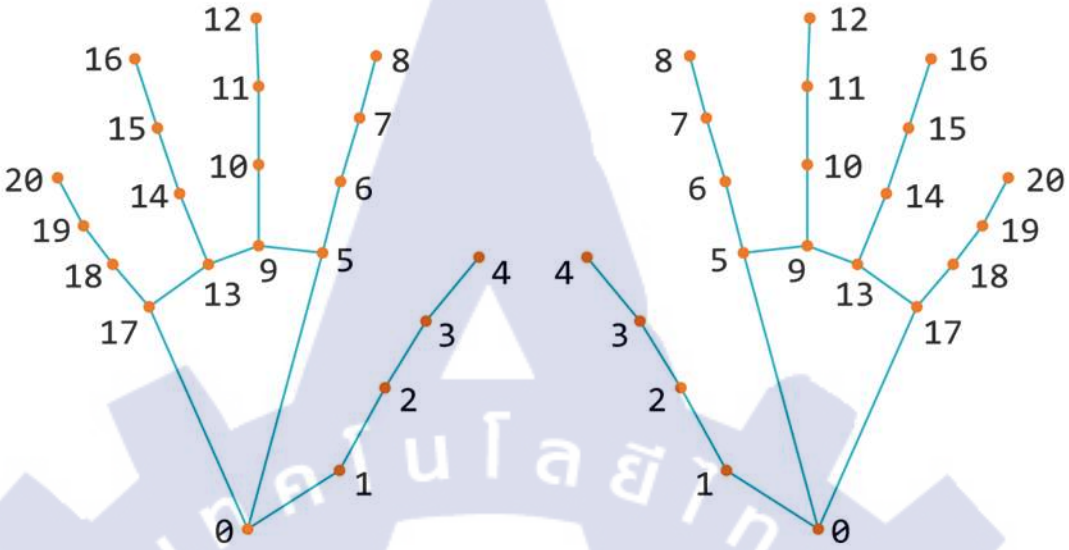


Figure 2.8 Distribution of 21 landmark points on the hand model

Table 2.3 Index of 21 Coordinates Derived from the Hand Model

0 - WRIST	1 - THUMB_CMC	11 - MIDDLE_FINGER_ DIP
	2 - THUMB_MCP	12 - MIDDLE_FINGER_ TIP
	3 - THUMB_IP	13 - RING_FINGER_ MCP
	4 - THUMB_TIP	14 - RING_FINGER_ PIP
	5 - INDEX_FINGER_MCP	15 - RING_FINGER_ DIP
	6 - INDEX_FINGER_PIP	16 - RING_FINGER_ TIP
	7 - INDEX_FINGER_DIP	17 - PINKY_ FINGER_MCP
	8 - INDEX_FINGER_TIP	18 - PINKY_ FINGER_PIP
	9 - MIDDLE_FINGER_ MCP	19 - PINKY_ FINGER_ DIP
	10 - MIDDLE_FINGER_ PIP	20 - MIDDLE_FINGER_ TIP

2.8 Model Evaluation

Model evaluation is the process of using specific metrics to analyze a model's performance. This involves assessing the model's accuracy, reliability, and ability to generalize to new or unseen data. To evaluate a multiclass classification model, a combination of metrics and visualizations is commonly used, such as the Confusion Matrix, Accuracy, Precision, Recall, and F1-Score. These metrics are often calculated on a class-by-class basis or as an average (macro/micro) to provide a comprehensive measure of overall performance.

2.8.1 The accuracy score of a Long Short-Term Memory (LSTM) model quantifies its ability to correctly predict outcomes in classification tasks. It is defined as the ratio of correct predictions to the total number of observations.

2.8.2 Precision is a performance metric used to evaluate the reliability of a model's positive predictions. The specific value for a given LSTM model varies according to the task and dataset.

2.9 Literature Review

Research on Artificial Intelligence (AI)-based sign language recognition has utilized various machine learning and deep learning algorithms, achieving accuracy rates nearing 100%. The performance and recognition capabilities of Thai Sign Language (TSL) systems depend heavily on the type and quality of training data. Generally, these data types are categorized into sensor-based and vision-based modalities. According to [9], the MNIST dataset was used to train a sign language recognition model using deep learning and computer vision. This database provided 60,000 training and 10,000 testing images of the English alphabet; using a Convolutional Neural Network (CNN), the model achieved 99% accuracy. Recent research in static sign language recognition (SLR) focuses on the high-accuracy identification of the American Sign Language (ASL) alphabet, primarily using deep learning models such as YOLO and CNNs [10]. In 'American Sign Language Detection using YOLOv5 and YOLOv8,' the authors compare these two models for ASL detection using image datasets. Real-Time Arabic Sign Language Recognition Using a Hybrid Deep Learning Model, the authors present a system for the real-time translation of Arabic Sign Language into text or speech [11]. The proposed hybrid

model combines a CNN to extract spatial features from sign language data with an LSTM to capture temporal characteristics for handling sequential data. The study presents a convolutional neural network (CNN) architecture developed using the TensorFlow framework to accurately recognize individual letters of American Sign Language (ASL), where the model achieved a mean validation accuracy of 95.48% and a test accuracy of 99.77% in identifying ASL characters [12]. Our research focused on implementing sequence sign language recognition, which incorporates temporal and spatial analysis. In[13], the authors propose the ST-GAT-SL method for continuous sign language recognition using skeleton data to mitigate background interference, where the OpenPose method is employed for extracting the skeleton data. OpenPose is an open-source, real-time computer vision library for multi-person 2D and 3D pose estimation. It uses deep learning to detect and track keypoints for the body (e.g., shoulders, elbows, hips), face, hands, and feet from images or video. Recent research utilized MediaPipe Holistic to track hand signs in videos. The research papers summarized below describe methods for real-time sign language recognition that use the MediaPipe Holistic framework for feature extraction, most commonly in combination with Long Short-Term Memory (LSTM) networks for sequence sign language classification.

Table 2.4 Summary of research papers on Real Time Sign Language Recognition using Mediapipe Holistic for Feature Extraction.

Paper Title	Key Methodology	Core Findings & Accuracy
MediaPipe-LSTM-Enhanced Framework for Real-Time Dynamic Sign Language Recognition in Inclusive Communication Systems [14]	Focuses on using MediaPipe as a feature extraction method with LSTMs to address challenges in continuous sign language recognition (CSLR), such as processing intricate sign sequences	This study aimed to increase recognition accuracy for continuous sequences, demonstrating (or hav ing demonstrated) the effectiveness of the hybrid approach for dynamic signs
MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition [15]	This paper addresses the challenges of dynamic sign language recognition by leveraging MediaPipe to extract key points from the hands, body, and face across a sequence of frames	This study used RNN models such as GRU, LSTM, and Bidirectional LSTM to handle the frame dependency of sign movement and achieved high accuracy on a custom dataset (DSL10-Dataset)
Dynamic Sign Language Recognition Using Mediapipe Library and Modified LSTM Method [16]	This research presents a framework integrated with the MediaPipe Holistic pipeline for feature extraction	Classification accuracy for individual signs reached approximately 85%, while sentences approached 80%. The study utilized a modified model focused on hand, body pose, and facial landmarks as primary features

Table 2.4 Summary of research papers on Real Time Sign Language Recognition using MediaPipe Holistic for Feature Extraction. (continued)

Paper Title	Key Methodology	Core Findings & Accuracy
Development of Thai Sign Language Detection and Conversion System into Thai with Deep Learning [17]	This paper presents a system leveraging the MediaPipe framework to extract hand, face, and gesture key points from video data. The authors developed a custom dataset consisting of seven classes, with three images per class	An LSTM-based architecture was designed to facilitate posture analysis and Thai word conversion. Following an evaluation that yielded accuracies of 0.83 and 0.91 for the LSTM and Bi-LSTM models respectively, the superior model was integrated into a mobile application
Thai Sign Language Recognition: an Application of Deep Neural Network [18]	This research utilized MediaPipe for the tracking of hand gestures	Various Recurrent Neural Network (RNN) models, including GRU, LSTM, and Bi-directional LSTM, were applied to recognize dynamic signs. Utilizing a custom-built dataset (DSL10-Dataset), the researchers achieved 99% accuracy

Chapter 3
Methodology

3.1 Research Plan

This research aims to develop a Thai Sign Language Recognition system capable of interpreting sequences of hand signs, body movements, and facial expressions. While current technology primarily focuses on isolated signs, such methods are inadequate for real-world conversational use; continuous recognition is essential for authentic, fluid communication, creating a significant communication barrier for the Deaf and hard-of-hearing in settings such as hospitals, schools, and offices. A real-time CSLR system can act as a direct bridge, reducing dependency on human interpreters. This article focuses on implementing a CSLR system within a limited timeframe; the sampling dataset consists of Thai sign language signs specifically used in emergency situations, The research workflow is illustrated in the diagram in Figure 3.1.

3.2 Research Strategy

Thai Sign Language (TSL) is a distinct, fully developed language that is linguistically separate from spoken Thai and unique to the Thai Deaf community. It possesses its own unique grammar and vocabulary. This article begins by defining a research problem in dynamic or sequential Thai Sign Language recognition. Our strategy was to conduct a literature review by examining systematic reviews and recent articles to identify gaps in research methodology and specific datasets. After conducting this review, we identified research gaps that fall into three parts, as outlined below:

3.2.1 Insufficient large-scale datasets are a primary obstacle for Thai Sign Language (TSL) recognition and translation.

3.2.2 Recent continuous Thai Sign Language models perform well in controlled environments (e.g., dark backgrounds and long sleeves) but struggle in 'unconstrained' real-world settings with varied lighting and dynamic backgrounds. Comparing recent developments in Thai Sign Language recognition, datasets and

vocabulary sizes are still small compared to the vocabulary of spoken Thai. Gathering more samples, implementing deep learning, and achieving real-time detection remain significant challenges.

3.2.3 Because Thai Sign Language has its own grammatical structure and the Thai Deaf community often utilizes simple sentence constructions, most recent research focuses on translating individual signs rather than full sentences.

Table 3.1 Research Plan

Research plan	
Year 2 Semester 1	Weeks 1–4: Scope, literature, and plan - Define contributions and research questions. - Draff dissertation/project outline. - Design preliminary experiments and establish data access.
	Weeks 5-7: Setup Tool and dependencies. - Set up the primary tool and all necessary dependencies. - Configure settings and establish a specialized baseline.
	Weeks 8–13: Data Preparation - Prepare the dataset and set up a reproducible environment. - Run baseline experiments and collect initial results.
	Weeks 14–15: Finalize and submit for research proposal - Obtain scope, required data and technique research proposal. - Prepare a concise presentation that summarizes research scope, methodology, and key finding.

Table 3.1 Research Plan (continued)

Research plan	
Year 2 Semester 2	Weeks 1–3: Find appropriate analytical techniques - Research and select techniques for hand detection and temporal analysis, then apply them to the sample dataset.
	Weeks 4-13: Training model - Cleaning data. - Labeling sequence data. - Selecting algorithms. - Training the model.
	Weeks 13–16: Evaluate performance - Evaluate model performance and visualize data results. - Implement real-time detection and interpret signs into text. - Train the RNN model. - Detect sign language as simple sentences. - Summarize research findings. - Submit the final paper.

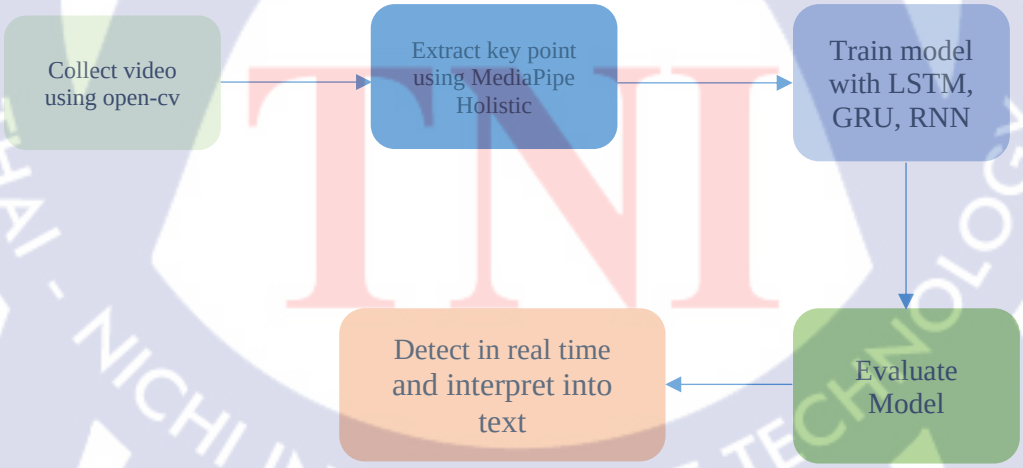


Figure 3.1 Simplified Block Diagram of Research Methodology

3.3 Data Collection

Estimating the exact number of words in the Thai language is challenging, but the vocabulary is undoubtedly vast. Standard large-scale dictionaries typically list between 400,000 and 600,000 words. In contrast, electronic databases like Lexitron focus on common usage, containing roughly 53,000 Thai words [19].

3.3.1 A Survey of Key Phrases for Emergency Help

To identify the most common Thai vocabulary for emergency assistance, a survey was conducted among 100 participants, including both hearing and hearing-impaired individuals. The study focused on key words and phrases related to emergency help of the 65 vocabulary words identified, only 18 were found in the TH-SL Thai Sign Language Database www.th-sl.com, the primary digital repository maintained by the National Association of the Deaf in Thailand. These 18 words consisting of ten verbs, seven nouns and “I” as a pronoun, were used as sample data for this study, with signs referenced directly from the TH-SL website as shown in Table 3.2, The survey indicates that 100 individuals provided a gender response: 42 women, 21 men and 37 transgender individuals as illustrated in Figure 3.2. Additionally the survey included 94 unimpaired and 6 impaired participants as illustrated in Figure 3.3.

Table 3.2 17 vocabulary words used as sample data, including their English meanings.

Verb	Noun
ยืม (Borrow)	หมอ (Doctor)
เป็นลม (Faint)	โรงพยาบาล (Hospital)
ไฟไหม้ (Fire)	เงิน (Money)
ไป (Go)	โทรศัพท์ (Phone)
หิว (Hungry)	ห้องน้ำ (Toilet)
หาย (Lose)	ตำรวจ (Police)
ป่วย (Sick)	
ขอโทษ (Sorry)	
เขียน (Write)	
ช่วย (Help)	
ปวดหัว (Headache)	

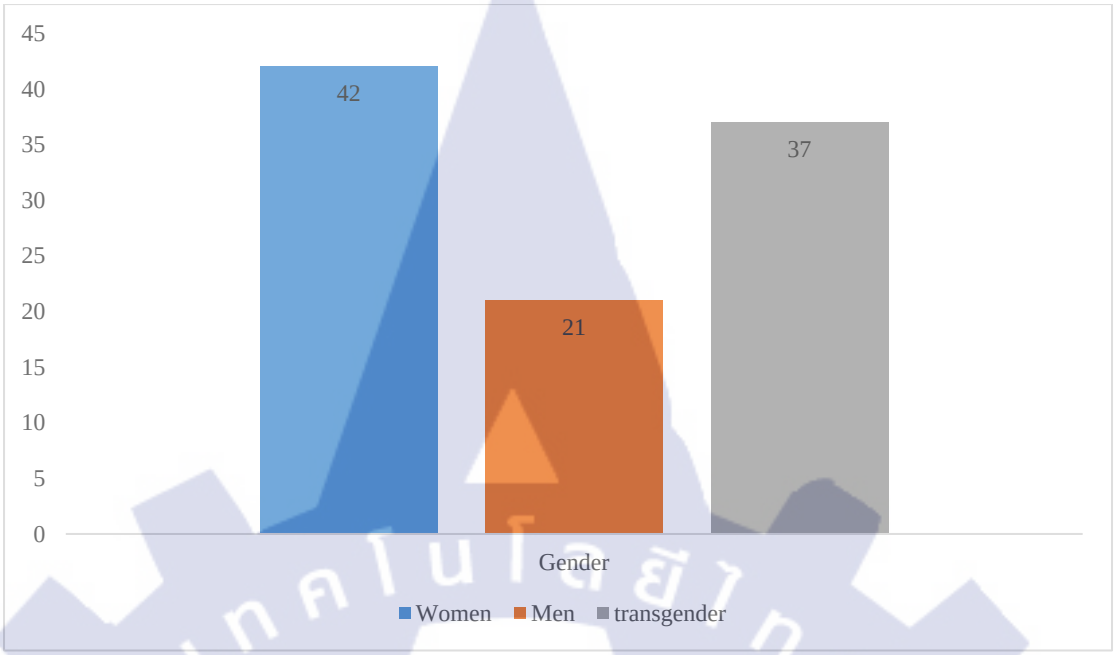


Figure 3.2 Distribution of Participants by Gender

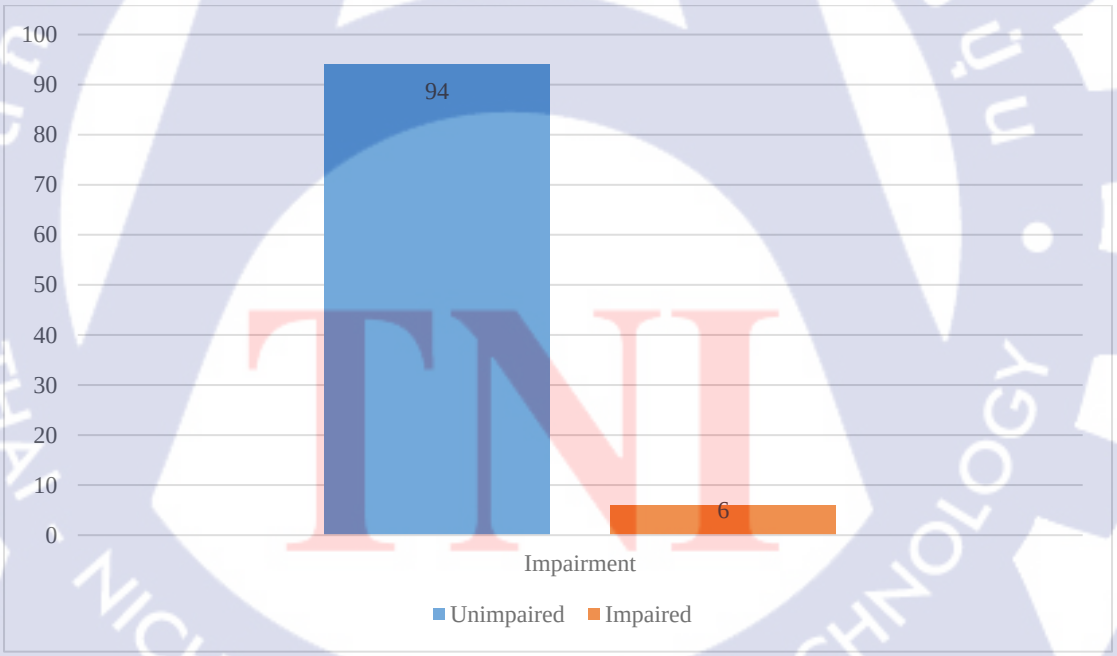


Figure 3.3 Number of Participants by Impairment Status

The Thai Sign Language (TSL) Database is a digital resource established as a pilot project by the National Association of the Deaf in Thailand (NADT). Developed as a digital platform, its objective is to provide a comprehensive database of TSL elements and everyday vocabulary. As of August 1, 2025, the site features 3,754 words and is hosted at th-sl.com [2], as shown in Figure 3.4, search results for the word “รถดับเพลิง” on www.th-sl.com.

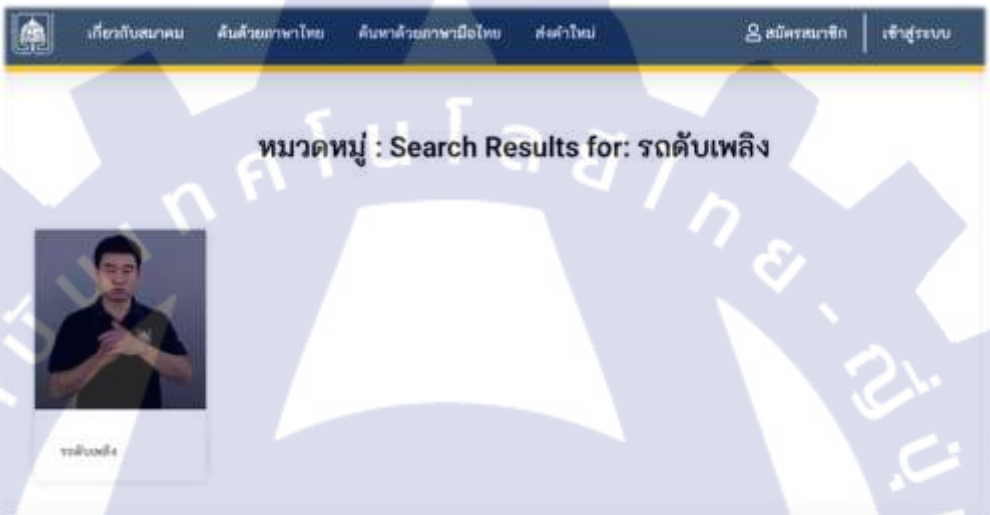


Figure 3.4 Search Results for the word "รถดับเพลิง" th-sl.com

3.4 Collecting Video Data for Sequential Thai Sign Language

Setting up an environment to collect sign language videos for research using OpenCV requires configuring both your hardware (such as a webcam) and software with the necessary OpenCV libraries. Collecting with diversity was the goal to achieve high accuracy. Videos of each word were collected using an OpenCV function: 50 videos per word, in an uncontrolled environment. In this experiment, we captured individual image frames of clear hand signs using OpenCV rather than recording video to save time during keyframe selection, collecting three frames per individual video. During the initial phase of this experiment, we encountered two primary challenges in our data collection process. The first occurred when utilizing online datasets for Thai Sign Language; the poor quality, unreliability, and lack of diversity in these datasets ultimately led to project failure. The second challenge arose

during the processing of video data. We initially collected videos with a duration of 3 seconds at a frame rate of 15 fps. However, as illustrated in Figure 3.5, frame extraction revealed that similar hand shapes and movements often recurred across multiple sequential frame ranges within a single video. As shown in Figure 3.5, relevant hand signs are absent from the initial and final frames, as well as several intermediate frames during transitions. To minimize the time required for keyframe searching, we gathered video data by selecting specific frames with clear hand signs. Our results indicate that model performance was influenced by the number of sequence frames per sample. The data were adjusted from 45 frame sequences to 15 and 9 frames, respectively. As shown in Figure 3.6, a sequence length of 9 yielded the best results. Detecting the start and end of continuous signs in continuous sign language (CSL) videos remains a significant challenge in computer vision and machine learning, especially concerning segments of a video where no action is being performed.

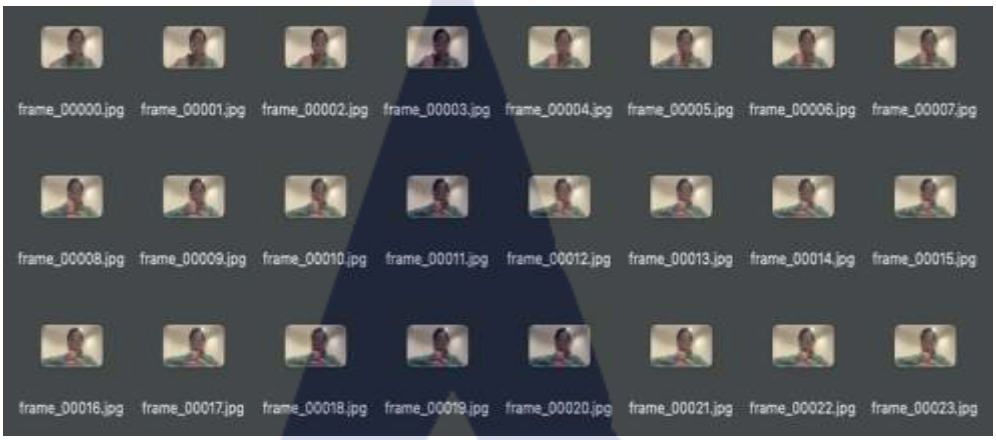


Figure 3.5 An Example from extracted frames collected from a failure data set, identified that similar sign language hand shapes or movements occur across several sequential frame ranges within a video.

Frame 0 Frame 2 ..... Frame 8

Figure 3.6 An example of a video collected using OpenCV consisted of 9 frames for one video.

3.5 Feature Extraction

The MediaPipe Holistic is a machine learning (ML) solution by Google that enables the real-time, simultaneous perception of human pose, face landmarks, and hand tracking on a single device. The task outputs a total of 543 landmarks (33 pose landmarks, 468 face landmarks, and 21 hand landmarks per hand) in real time [6]. This article utilizes skeleton-based feature extraction via MediaPipe Holistic by extracting detailed landmark data from the hands and body, the face landmark models were entirely removed; only hand and pose landmarks are being used for features in

this experiment. When processing an image or video frame using a MediaPipe Holistic task, the result object contains several landmark lists [8]:

- Face_landmarks: Face landmarks in normalized image coordinates (x, y, z).
- Pose_landmarks: Pose landmarks in normalized image coordinates (x, y, z, visibility).
- Left_hand_landmarks: Left-hand landmarks in normalized image coordinates (x, y, z).
- Right_hand_landmarks: Right-hand landmarks in normalized image coordinates (x, y, z).

As shown in Figure 3.5, some video frames contain invisible landmarks. Using a specific default coordinate such as (0, 0, 0) for hand landmark is a common and practical method for handling missing data points in datasets. The feature set for this experiment consists of 21 right-hand landmarks and 21 left-hand landmarks in 3D coordinates per frame.

Figure 3.7 Example of input shape detected landmarks from one image frame.
The coordinates comprise of 21*(x, y, z) points for left-hand and 21*(x, y, z) points for right-hand

3.6 Model Architect

Our approach involves building models to recognize hand signs from frame sequences. These models are designed to learn temporal dynamics (movement over time) by utilizing LSTMs, RNNs, and GRUs within deep learning frameworks. As shown in Figure 3.8, the finalized model is based on an LSTM architecture with two additional hidden layers following the input. In our experiments, each sample consisted of nine time steps derived from video captures. As illustrated in Figure 3.7, the input for each time step corresponding to a single image frame consisted of 126 values extracted from detected hand landmarks.

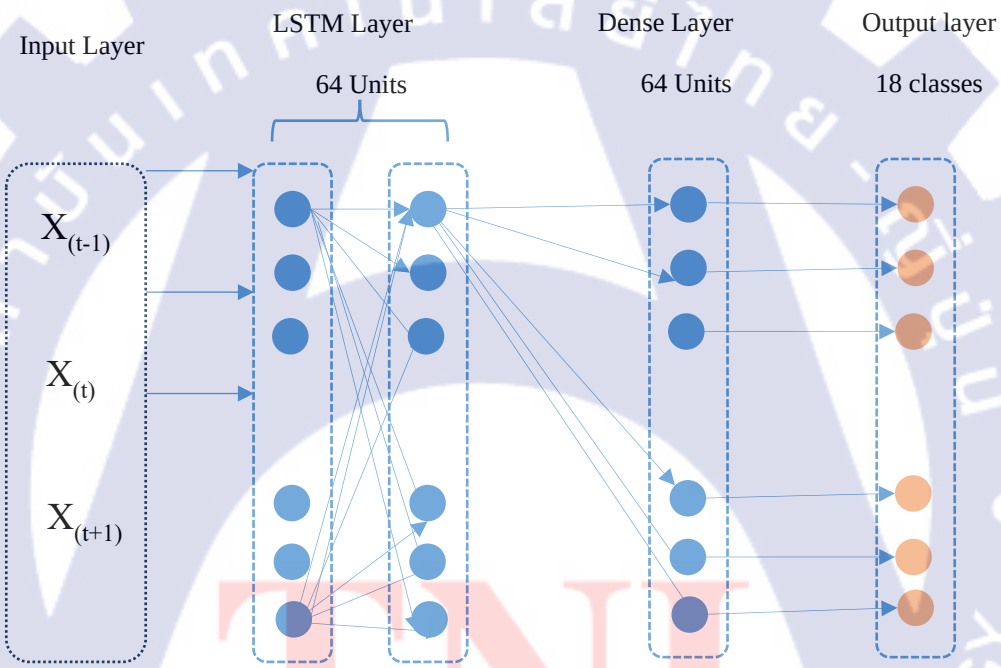


Figure 3.8 Neural Network Architecture composed of an LSTM layer and two additional hidden layers

3.7 Improve Model Accuracy

3.7.1 Improve Feature Extraction Pipeline

1) Normalize landmark data: The raw coordinates from MediaPipe Holistic are sensitive to the signer's position and distance from the camera. Normalize these landmarks by:

(1) Translating to a Reference Point: Use a central point (e.g., the center of the torso) to make data independent of the signer's location.

(2) Scaling by Body Dimensions: Scale coordinates relative to a fixed dimension, such as the distance between the shoulders, to account for camera depth. This ensures the model is robust to scale variations.

2) Add landmark features: MediaPipe Holistic provides landmarks for the face, hands, and pose. Model accuracy can be improved by feeding it a richer feature set that includes not only raw (x, y, z) coordinates but also derived features.

3) Handle missing landmarks: MediaPipe may fail to detect landmarks in some frames due to occlusion or poor lighting. Implement a robust strategy for handling missing data, such as:

(1) Interpolating missing values from surrounding frames.

(2) Replacing them with default values or zeros.

3.7.2 Augment and Enhance the Dataset

A larger, more diverse dataset is crucial for improving model generalization and reducing overfitting. To create a robust dataset, capture data from various sources that reflect real-world conditions, including:

1) Varying lighting and backgrounds.

2) Multiple signers with different signing styles.

3) Various repetitions of each sign.

3.7.3 Optimize the LSTM Model Architecture

1) Stacking LSTM layers: Increasing the number of LSTM layers allows the model to learn more complex temporal patterns and higher-level abstractions.

2) Use bidirectional LSTMs (BiLSTM): Bidirectional LSTMs process the sequence data in both forward and backward directions. This is particularly

effective for sign language, where context from both before and after a key gesture is important for accurate interpretation.

3) Introduce an attention mechanism: For longer, more complex sign sequences, an attention layer helps the LSTM focus on the most relevant frames and landmark features. This prevents the loss of critical early information and enhances overall model performance.

3.8 Model Training

The model is trained using a labeled dataset of sign language videos. During this process, it adjusts its internal weights and biases via backpropagation. Data augmentation techniques for Long Short-Term Memory (LSTM) models particularly in the context of time series data aim to increase the size and diversity of the training set to improve generalization and reduce overfitting. Beyond jittering (noise injection), common techniques include flipping and rotation. These involve inverting or rotating the time series along its axis, provided the transformation is applicable to the data type, such as in certain sensor data or image sequences where the specific temporal order may be less critical.

Chapter 4
Experiment and Results

4.1 Experimental Setup

This section describes the experimental evaluation of a Sign Sequence Recognition System using Deep Learning on a custom dataset of Thai Sign Language. For video sequence processing, models must capture both spatial information within frames and temporal information across frames. This study utilizes two methods for spatial feature extraction, combined with Recurrent Neural Networks (RNNs) or Transformers for temporal analysis. Each model is trained, evaluated, and compared to determine the optimal configuration. The training environment was hosted on a macOS virtual machine (VM) using Anaconda to manage the Python-based Deep Learning environment. The specific experimental setup and hardware configurations are summarized in Table 4.1.

Table 4.1 A table summarizing software and hardware requirements

Environment	Anaconda for Apple Silicon (arm64), Python 3.13
Host	Jupyter Notebook
CPU	Apple Silicon Chip M2 Pro CPU with 12-core design
GPU	Apple Silicon Chip M2 Pro GPU with 16-core design
RAM	16 GB
Data Storage	512 GB
Operating System	Mac OS Tahoe Beta 26.0.1
Required Dependencies	TensorFlow, MediaPipe, Sk-lern, Keras

4.2 Experimental Results

This research focuses on tuning the architectural and training parameters of an LSTM model, specifically the number of units and layers, to optimize performance and mitigate overfitting. Furthermore, we incorporate various algorithmic enhancements to refine the learning process.

4.2.1 Algorithms

1) Long Short-Term Memory (LSTM), a specialized type of Recurrent Neural Network (RNN) is used to process temporal sequences of frames. The LSTM's ability to retain information over long sequences is critical for interpreting motion-based cues in dynamic signs. The primary components of an LSTM network are the sequence input layer and the LSTM layer. The sequence input layer feeds time-series data into the network, while the LSTM layer learns long-term dependencies between time steps [20].

2) Gated Recurrent Units (GRU), the GRU utilizes a simpler architecture compared to LSTM networks, making it computationally efficient and faster to train. Despite its simplicity, it provides strong performance on tasks involving time-series and sequential data [21].

3) Recurrent Neural Networks (RNN), standard RNNs address sequential data by incorporating loops that allow information from previous steps to be fed back into the network, maintaining a "memory" of prior inputs [4].

4.2.2 Data Splitting

In this research, we focus on optimizing LSTM model performance and preventing overfitting by tuning both architectural and training configurations, such as the number of units and layers. Furthermore, we enhance the learning process through various algorithmic approaches. The dataset was partitioned into training, validation, and test sets using an optimal 60/20/20 ratio. The model was trained across1 different epoch values to determine the best fit. Based on the training and validation loss graphs (Figures 4.1–4.3), we analyzed the learning curves to identify the optimal stopping point, with 600 epochs yielding the best performance. As illustrated in Figure 4.3, both training and validation losses decrease, indicating ideal learning. The model demonstrates effective learning from the training data and generalizes well to unseen data, with both curves converging and stabilizing at a low loss value.

4.2.3 Training Parameters

The model was trained with the following parameters:

1) Activation Function: This model utilizes the SoftMax activation function to handle multiclass classification.

2) Epoch: the model's performance was monitored via a learning curve, which plots training and validation loss (e.g., categorical cross-entropy) over the course of the training epochs to diagnose model fit.

3) The number of units in the LSTM and dense layers significantly affects model performance by controlling capacity, complexity, and computational cost. Optimizing these units is a critical component of hyperparameter tuning, as an inappropriate configuration can lead to underfitting or overfitting.

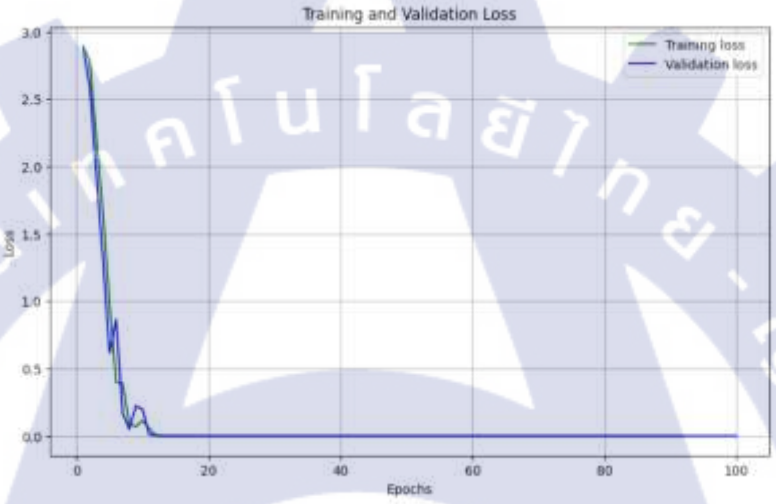


Figure 4.1 Training and Validation Loss Trained model with 100 epochs

Figure 4.2 Training and Validation Loss Trained model with 300 epochs

Figure 4.3 Training and Validation Loss Trained model with 600 epochs

4.2.4 Experimental Results

We applied various numbers of epochs for model training. By visualizing the training and validation loss graphs, we learned from the curves to find the good number of epochs to perform the model in our experiment. Specifically. The result indicated that 600 epochs is a good number to perform the model in our experiment. A training and validation loss graph for an LSTM model, also known as a learning curve, is a crucial tool for diagnosing the performance of a deep learning model. This graph typically plots the loss values (e.g., mean squared error for regression, categorical cross-entropy for classification) of both the training and validation datasets over the course of training epochs. In Fig. 6, shows the training and validation loss results from the trained model. Our experiment results, shown in Table 4.2,Our experiment results, shown in Table III, compare the performance of LSTM, RNN and GRU using 30 and 50 videos per sample.

Table 4.2 Experimental results comparing the performance of LSTM, RNN, and GRU using 30 and 50 videos per sample

Model	30 samples		50 samples	
	Computation time (sec)	**Accuracy**	**Computation time (sec)**	**Accuracy**
LSTM	54.17	0.82	82.18	0.85
RNN	25.00	0.70	35.89	0.72
GRU	53.04	0.79	81.55	0.84

In our approach, the classification accuracies using 30 videos per class for LSTM, RNN, and GRU were 0.82, 0.70 and 0.79 respectively. Increasing the sample size to 50 videos per class improved the results to 0.85, 0.72 and 0.84 as shown in Table III. To optimize the architecture, we systematically adjusted the number of layers for LSTM, RNN and GRU models, alongside the dense layers, using a dataset of 50 videos per class, these findings are presented in Table IV. Implementation of a 2-layer LSTM combined with a 1-layer dense network yielded the highest accuracy, achieving 0.90.

Table 4.3 Results of Experimental Models

Model Architect	**Training time**	**Total Parameter**	**Accuracy**
LSTM 2Layer & Dense Layer 1 Layer	60.48	450,104	0.90
GRU 2Layer & Dense Layer 1 Layer	26.61	117,752	0.73
RNN 2Layer & Dense Layer 1 Layer	60.70	362,360	0.83
LSTM 2Layer & Dense Layer 2 Layer	63.36	475,928	0.83
GRU 2Layer & Dense Layer 2 Layer	27.57	139,064	0.74
RNN 2Layer & Dense Layer 2 Layer	60.35	362,360	0.80
LSTM 3Layer & Dense Layer 1 Layer	83.36	607,352	0.86
GRU 3Layer & Dense Layer 1 Layer	35.38	151,352	0.74
RNN 3Layer & Dense Layer 1 Layer	90.27	461,816	0.79

4.2.4 Model Evaluate

We evaluate the classification performance of our model using F1-score and recall, The confusion matrices for the implemented LSTM, RNN, and GRU models are displayed in Figures 4.4-4.6.

Table 4.4 F1- score and recall from an experiment

Algorithms	F1-Score	Recall
LSTM 2 Layer & Dense Layer 1 Layer	0.89	0.90
RNN 2 Layer & Dense Layer 1 Layer	0.74	0.73
GRU 2 Layer & Dense Layer 1 Layer	0.83	0.83

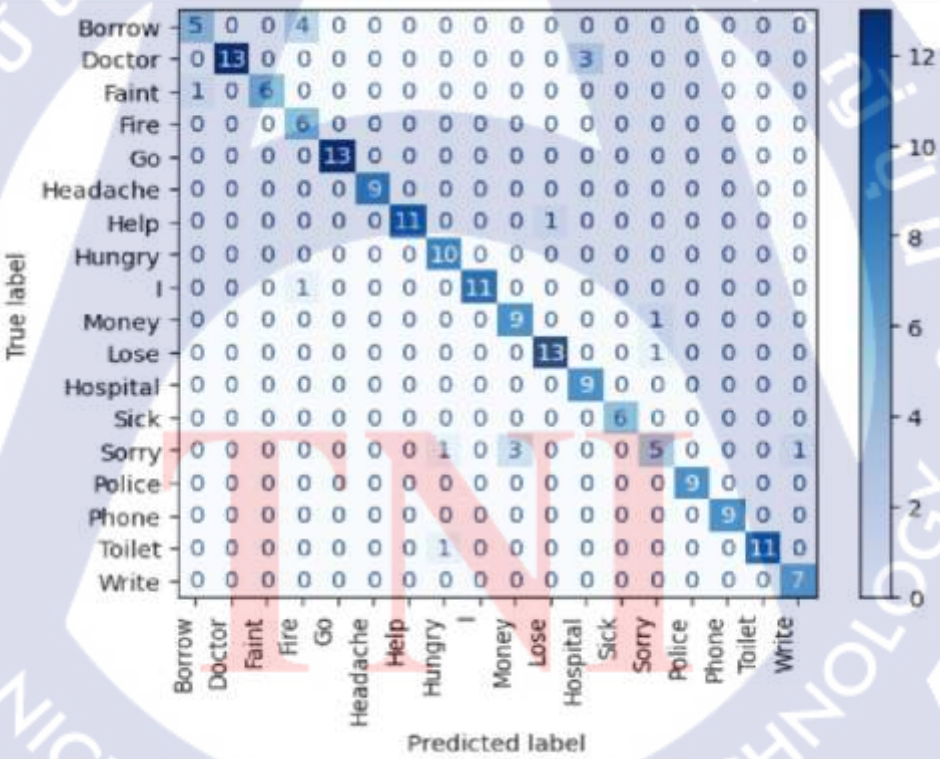


Figure 4.4 Confusion matrix of predictions on the validation set generated by the LSTM model

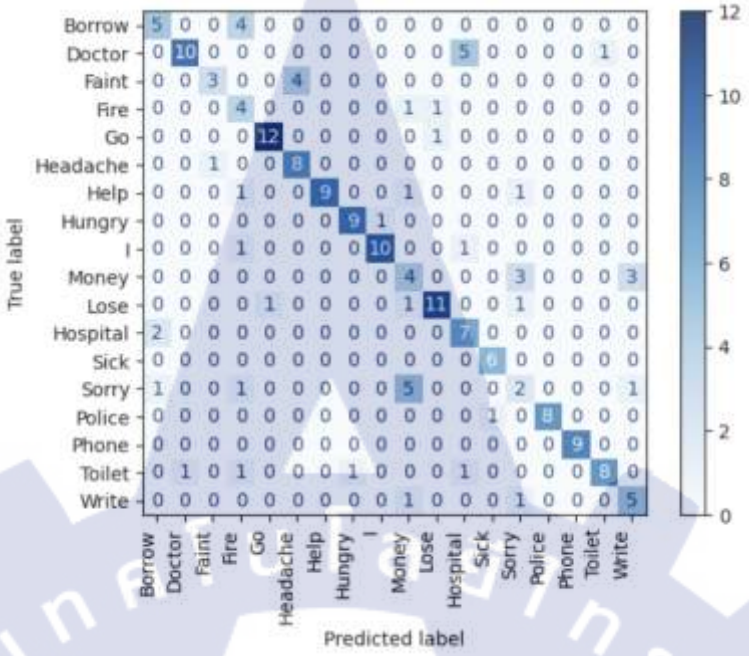


Figure 4.5 Confusion matrix of predictions on the validation set generated by the RNN model

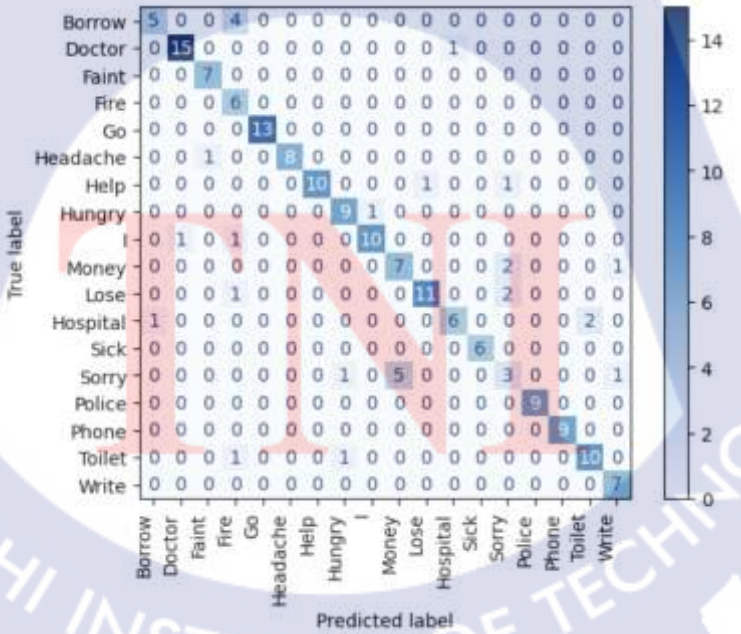


Figure 4.6 Confusion matrix of predictions on the validation set generated by the GRU model

After real-time hand sign detection, this article structures individual vocabulary into sentences following Thai grammatical rules (Verb + Noun or Noun + Verb) using an RNN model. The vocabulary includes terms such as “ไป”, “โรงเรียน”, “โรงพยาบาล”, “ยืม”, “โทรศัพท์”, “เงิน”, and “หาย”. The raw dataset, summarized in Table 4.5, was used to train the RNN. In real-time detection, individual hand signs were interpreted as simple sentences, achieving an 81% accuracy rate in sentence classification. As shown in Figure 4.7, the proposed model detects individual hand signs capturing two actions within ten seconds and translates them into a single, simple sentence following Thai grammar structures.

Table 4.5 Raw Training data set combined from individual vocabularies following Thai grammar rule

ไปโรงพยาบาล	ยืมเงิน	เงินหาย
ไปโรงเรียน	ยืมโทรศัพท์	โทรศัพท์หาย

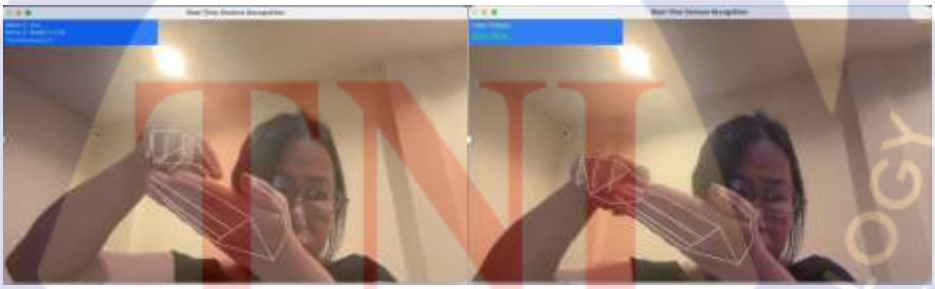


Figure 4.7 The proposed model detects individual hand signs capturing two actions and translates them into a single, simple sentence following Thai grammar structures

Chapter 5
Conclusion

Initial failures in this research were attributed to the limitations of a public dataset, where a sparse number of examples and frames negatively affected model accuracy. To address this, a custom dataset for Thai Sign Language, focusing on emergency scenarios, was collected. The proposed system utilizes MediaPipe Holistic for landmark extraction and employs deep learning architecture specifically LSTM, RNNs and GRU for training. Finally the model integrates RNNs to detect on-screen text, translating individual vocabulary into coherent sentences that align with Thai grammar structures. The implementation achieved a high accuracy of 90% using LSTM for detecting individual hand signs and 81% for classifying complete sentences.

.

References

References

[1] สิทธิเดช อุดมนา, *"สถานการณ์ด้านคนพิการ,"* Department of empowerment of persons with disabilities [Online]. Available: https://www.dep.go.th/th/law-academic /knowledge-base/disabled-person-situation. [Accessed : April 30, 2025].

[2] National Assosiation of the Deaf in Thailand, *"ฐานข้อมูลภาษามือไทย,"* [Online]. Available:https://www.th-sl.com/en/search-by-ac. [Accessed : April 30, 2025].

[3] B.V. Elsevier,*"Neural Network Model,"* ScienceDirect [Online]. Available: https://www.sciencedirect.com/topics/computer-science/neural-network-model. [Accessed: May 24, 2026].

[4] Cole Stryker, *"What are Recurrent Neural Networks?,"* IBM Think [Online]. Available: https://www.ibm.com/think/topics/recurrent-neural-networks. [Accessed: Febuary 14, 2026].

[5] Ratchasuda College, *Thai Sign Language Book 2*, Bangkok, Thailand: Ratchasuda College, Mahidol University [Online]. Available: http://www. braille-cet.in.th/ Braille-new/files/ThaiSignLanguageBooks/ ThaiSignLanguage Book_2.pdf. [Accessed: Febuary 13, 2026].

[6] Google AI, *"Google AI launches MediaPipe Holistic: Simultaneous Face, Hand and Pose Prediction,"* Google Research Blog [Online]. Available: sertiscorp.com. [Accessed: Febuary 13, 2026].

[7] Google AI Edge Team, *"Face Landmarker Detection Guide for Python,"* AI Edge MediaPipe [Online]. Available:https://ai.google.dev/edge/mediapipe/ solutions/vision/face_landmarker/python/. [Accessed: May 24, 2026].

[8] Google, *"Hand Landmarks Detection Guide for Python,"* AI Edge MediaPipe [Online]. Available:https://ai.google.dev/edge/mediapipe/solutions/ vision/ hand_landmarker/python/. [Accessed: March 1, 2025].

[9] R.S. Sabeenian et al., "Sign language recognition using deep learning and computer vision," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 12, no. 5, pp. 964–968, May 2020.

[10] S. Tyagi et al., *"*American sign language detection using YOLOv5 and YOLOv8*," Research Square*, July 2023, DOI: 10.21203/rs.3.rs-2831835/v1.

[11] T.H. Noor et al., *"*Real-Time Arabic sign language recognition using a hybrid deep learning model,*" Sensors (MDPI),* vol. 24, no. 12, pp. 3855, June 2024.

[12] A. Abdullah et al., "American sign language character recognition using convolutional neural networks," *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE),* Regina, SK, Canada, September 24 – 27, 2023, pp. 165–169.

[13] Q. Guo et al., "Continuous sign language recognition based on spatial-temporal graph attention network," *Computer Modeling in Engineering & Sciences*, vol. 134, no. 1, pp. 297–311, September 2022.

[14] N. Anithadevi et al., "MediaPipe-LSTM-enhanced framework for real-time dynamic sign language recognition in inclusive communication systems," *Engineering Reports,* vol. 7, no. 7, pp. e12937, July 2025.

[15] G. H. Samaan et al., "MediaPipe's landmarks with RNN for dynamic sign language recognition, " *Electronics*, vol. 11, no. 19, pp. 3228, October 2022.

[16] Ridwang et al., "Dynamic sign language recognition using mediapipe library and modified LSTM method," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 13, no. 6, pp. 2223–2229, December 2023.

[17] C. Damrongekarun et al., "Development of Thai sign language detection and conversion system into Thai with deep learning," *KKU Science Journal*, vol. 51, no. 5, pp. 216-225, September 2023.

[18] A. Chaikaew et al., "Thai sign language recognition: An application of deep neural network," *Joint Int. Conf. Digital Arts, Media and Technology with ECTI Northern Section Conf. Electrical, Electronics, Computer and Telecommunication Engineering (ECTI DAMT & NCON)* , Chiang Rai Rajabhat University, Chiang Rai, March 3-6, 2021, pp. 128-131.

[19] NECTEC, "*Open Data Portal,*" LEXITRON 2.0[Online]. Available: https://opend-portal.nectec.or.th/dataset/lexitron-2-0. [Accessed: February13, 2026].

[20] Joshua Noble, "*What is Long Short-term Memory (LSTM)?*," IBM Think [Online]. Available: https://www.ibm.com/think/topics/lstm. [Accessed: February 14, 2026].

[21] F. M. Salem, *Recurrent Neural Networks,* Cham, Switzerland: Springer International Publishing, 2022.

Bio Data

Name - Surname Ms.Nut Samaksaman

Date of Birth 1st September 1986

Address 65/16 S-gate premium Kantana
Bangmaung Bangyai Nonthaburi
11140 Thailand
Phone number: 098-902-0155
e-mail: sm.nut_st@tni.ac.th

Education Background

2025	Master of Science in Information Technology.
	Thai-Nichi Institute of Technology
2010	Bachelor of Science in Computer Engineering
	Suranaree University of Technology

Work Experience

2014 – 2018	Hamani Thai Co., Ltd.,
	M.D. Assistant
2018 – Present	Digital Marketing
	Freelance