Invention The Program of Hands Motion Detector for Translating Disabilities People's Sign Language

Santi Pattanavichai* and Nopparuj Phongtulee*

Received: July 3, 2024
Revised: August 22, 2024
Accepted: August 27, 2024

* Corresponding Author: Santi Pattanavichai, E-mail: pattanavichai@gmail.com

DOI: 10.14416/j.it/2024.v2.005

Abstract

The number of people with disabilities is currently increasing, and in Thai society, the proportion of individuals with disabilities is notably high, approximately 3% of the population (Krungthep Turakij, 2022). Providing support and facilitating accessibility for people with disabilities is of great importance. As part of these efforts, we developed a motion detection program aimed at translating sign language for individuals with disabilities. This program utilizes object detection techniques, such as MediaPipe, in combination with machine learning. It accesses the camera to capture sign language gestures and compares them with models trained on a sign language gesture dataset. The program then displays the corresponding meaning on the screen, with the goal of aiding communication for people with disabilities. The results indicate that the model performs well when tested with a benchmark dataset of 75,000 sign language gestures. However, challenges arise when gestures are unclear, incorrect, not included in the dataset, or closely resemble other gestures, which may lead to misclassification. Additionally, limitations due to insufficiently powerful training equipment have caused a delay in processing and displaying gesture meanings, with a lag time of approximately 5-10 seconds. Despite these challenges, the model achieves an accuracy of 76.40%, which is considered satisfactory. The program is also capable of translating the detected gestures into the Thai language.

Keywords: Hands Motion Detector, Sign Language, Machine Learning, Media Pipe, Datasets, Pillow library, Disabilities People.

1. Introduction

In today's Thai society has the number of people with disabilities is increasing. For example, in Thai society, the ratio of people with disabilities, it is approximately 3 percent [1] from the all population ratio. This makes facilitating people with disabilities is an important matter and should be developed for communication. Because of the disabled person, it deserves quality and convenience in living and no different from normal people Both in terms of living and understanding of communication.

Disabled people who are hearing impaired and the speech impaired considered a type of person Disabled people. They still have communication barriers because of the disabled person unable to communicate and can understand speech like normal people in society who due to hearing or speaking problems. There has been an invention Sign Language for the disabled or Sign Language for use in communication among the disabled people Sign language can be seen on television programs. or news reporting that must be communicated to people with disabilities to understand. But for normal people in society May not have knowledge understanding of sign language communication for people with disabilities people who is considered a barrier to communication between people with disabilities and other people in society.

Sign language for the disable, it is considered nonverbal. Which is the use of moving the body with the fingers, hands and arms are symbols to convey meaning into various meanings that need to be communicated instead of spoken words. It may have meaning. or must use spelling methods to convey meaning. [2], [3].

* *Faculty of Science and Technology, Rajamangala University of Technology Thanyaburi, RMUTT*

The organizing team therefore conducted a study and research on sign language for the disabled using techniques for Object Detection such as Media Pipe combined with machine learning or Machine Learning and developed into a system to translate sign language for the disabled people. To reduce communication barriers between people with disabilities and other people in society.

This paper is organized as follows. Section I introduces the topic and elaborates of a study and research on sign language for the disabled people. Section II describes previous studies related to sign language and machine learning. Section III discusses the proposed the program for using techniques for Object Detection such as MediaPipe combined with machine. Section IV describes the performance of the program for using techniques for Object Detection such as MediaPipe combined with machine. Section and this program can translate to Thai language. Finally, Section V, we discuss the conclusions and proposed future work.

2. Related Work

2.1 Sign language

The system that the group developed each one focuses on recognizing a single sign language gesture parameter [4].

These parameters are the position of the hands. Shape, movement and orientation and non-manual gestures in the form of facial expressions [5], [6].

It's part of a full animation show of animated human characters. Sign language, so humans viewing the animation may be able to identify something that is signed based on the words spoken or the movement of the surrounding arms. In this case, it is the difference between the quality of the hand shape perceptions in the first evaluation study appear less dramatic. When hand shapes are part of the full animation of sign language [7], [8].

2.2 Machine Learning

Data is collected in the form of images to be used for machine learning. This program used object detection technology such as Media Pipe together with machine learning in order to remember and consider the posture of movement. Using OpenCV image processing technology. In order to do Machine learning using KERAS and receiving image values from the camera in order to process and display them in the form character. Popular Machine Learning Algorithms, it is based on biological neurons, which are the foundation of artificial neural networks [9], [10].

2.2.1 Media Pipe It is object detection technology. This program used the translate sign language python language with Media Pipe which it is owned by Google and it is an open source AI platform that can be used as a pipeline for detecting and recognizing complex faces, hands, and gestures. Using acceleration in identification and processing So it came out to be an accurate and fast solution. Now Google AI has taken it to the next level. and is ready to introduce everyone to "Media Pipe Holistic," a solution that will allow devices that can detect multiple parts of the body at the same time to actually be developed [10] - [12].

MediaPipe Holistic uses a trade-off between the three sensing points, and its efficiency depends on the speed and quality of the data exchange. Combining the three sensing points results in a single, cohesive topology that captures 540 + motion keypoints (33 gesture points, 21 points on each hand, and 468 points on the face) at an unprecedented level, and can be processed in near real-time for mobile display.

But that's just the tip of the iceberg. MediaPipe also makes use of OpenCV, a powerful open-source library for computer vision. OpenCV has lots of tools and algorithms for working with images and videos. By using OpenCV, MediaPipe can easily add features like video capture, processing, and rendering to its pipelines. MediaPipe also teams up with TensorFlow, Google's machine learning tool, to make adding pre-trained or custom models easy. This makes tasks like recognizing faces or understanding speech easy. MediaPipe can also support popular languages like C++, Java, and Python, so it's simple to add to your projects.

2.2.2 OpenCV is the image processing technology. It is a Library of Programming functions, mainly aimed at

real-time computer vision. It was originally developed by Intel, but later became supported by Willow Garage followed by Itseez (later acquired by Intel). OpenCV is a cross-platform library. (Cross-Platform) and free to use under the Open-Source BSD License [13] - [15]. OpenCV is a library of programming functions used primarily for images processing.

Opencv is a huge open-source library for computer vision, machine learning, and image processing. Now, it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human.

When it is integrated with various libraries, such as NumPy, python is capable of processing the Opencv array structure for analysis. To Identify an image pattern and its various features we use vector space and perform mathematical operations on these features.

2.2.3 KERAS is the tool for Machine Learning or Deep Learning, we need to choose. Library (Library) to suit our work. In order to make the performance of our model good. KERAS is an open-source neural network written in Python. KERAS is mainly used for doing Deep Learning where the Input (data import) is an image. Because it is ready-made, but it will be a little customizable and it is a finished product. Highly adaptable, KERAS gives every programmer freedom through the integration of low-level machine learning languages such as Tensor Flow or Theano, which means that anything written in the source language may be executed by KERAS [16], [17].

Keras is an open-source deep-learning framework that gained attention due to its user-friendly interface. Keras offers ease of use, ùexibility, and the ability to run seamlessly on top of TensorFlow. In this article, we are going to provide a comprehensive overview of Keras.

Keras is a high-level, user-friendly API used for building and training neural networks. It is designed to be user-friendly, modular, and easy to extend. Keras allows you to build, train, and deploy deep learning models with minimal code. It provides a high-level API that is intuitive and easy to use, making it ideal for beginners and experts alike.

To develop a model that can help with prediction, we can use KERAS or TensorFlow, which are deep-learning libraries. KERAS is a high-level deep learning API developed by Google for implementing neural networks. It is written in Python and is used to simplify the implementation of neural networks, and supports multiple neural network computation backbends, which use a high level of abstraction in the Python frontend. This makes KERAS slower than other deep learning frameworks.

2.2.4 Pillow Library It is the library for translating to Thai Language. The developer encountered a problem with the library that was selected for use in the Image Processing section, OpenCV, which was unable to display the language in the system like UTF-8, causing various display sections to be unable to display Thai language. The developer therefore chose to use the Pillow Library, which is used in the Image Processing section, to work with OpenCV, which is to use OpenCV to access working with the camera and use Pillow to do all the text display work. Therefore, we allowing the system to display Thai language [18], [19].

2.3 Comparisons with Previous Research

In this study, we compare our work with the paper titled Development of Thai Sign Language Detection and Conversion System into Thai with Deep Learning. That paper utilized the MediaPipe framework and Bidirectional Long Short-Term Memory (BiLSTM) to compare the performance of LSTM and BiLSTM models. While it demonstrated that BiLSTM provided better results than LSTM, it did not present an implementation in a programmatic format as our study does.

Our research integrates a broader set of tools, including MediaPipe, OpenCV, LSTM, KERAS, and the Pillow library, to develop a program specifically designed for individuals with disabilities [20]. The primary objective of our research is to create a practical tool that can be used by people with disabilities in real-world scenarios. The combination of these tools enabled us to successfully develop a program that meets this objective.

3. The Working Structure of Program Hands Motion Detector for Translating Disabilities People's Sign Language

3.1 Flow Chart shows the flow of the program

From the flow chart, it shows the sequence of the program's operations when starting to use. The program will receive values from the device's camera through OpenCV and then use the Media Pipe to capture points as defined, including right and left fingers, right and left hands, shoulders, and faces, by capturing the data in Array format. After that, it will use the AI that has been performed. Practice it. to compare with the values currently captured.

After that, if the detected gesture Is it closest to which word? The word will be displayed on the screen and the whole system will be repeated again. If there is still movement. If the user is not moving anymore The program will stop working. and if the user presses to shut down the system The system will shut down immediately shown in Figure 1.

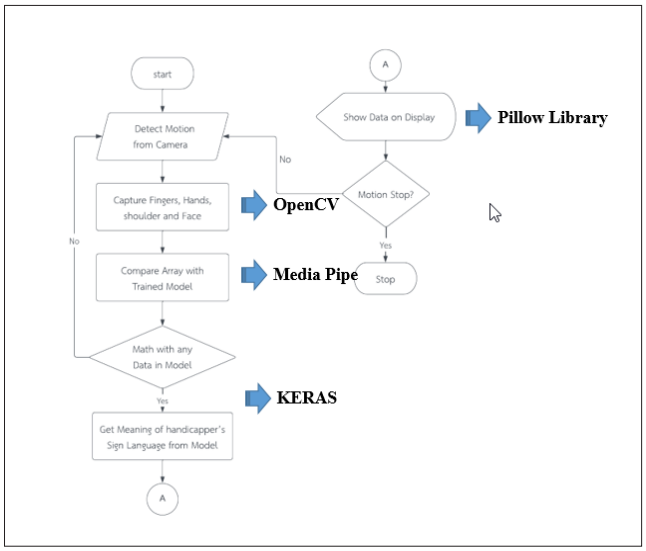

Figure 1. *Figure for Flow Chart shows the flow of the program.*

From the flow chart, it shows how Machine Learning works. It uses the dataset that the developer has collected to train the AI. The model being built uses LSTM layers to handle sequence data [21], [22]. It is characterized as a "time series" where the model's input is a sequence of data with 30 time steps (number of frames) and 1662 features (number of landmarks at various points to be used in learning) in each time step. The resulting Dense transforms the nature of the data to produce output of the correct class. The activation function used is ReLU (Rectified Linear Unit) [23].

After that, the data set will be divided into 2 sets, the ratio is 80:20, with 80 sets used for learning and 20 used for testing. Then use the Adam algorithm to divide and improve the weights in the train, then train a total of 1,000 times, then export to get the Action.h5 file to use in the next program shown in Figure 2.

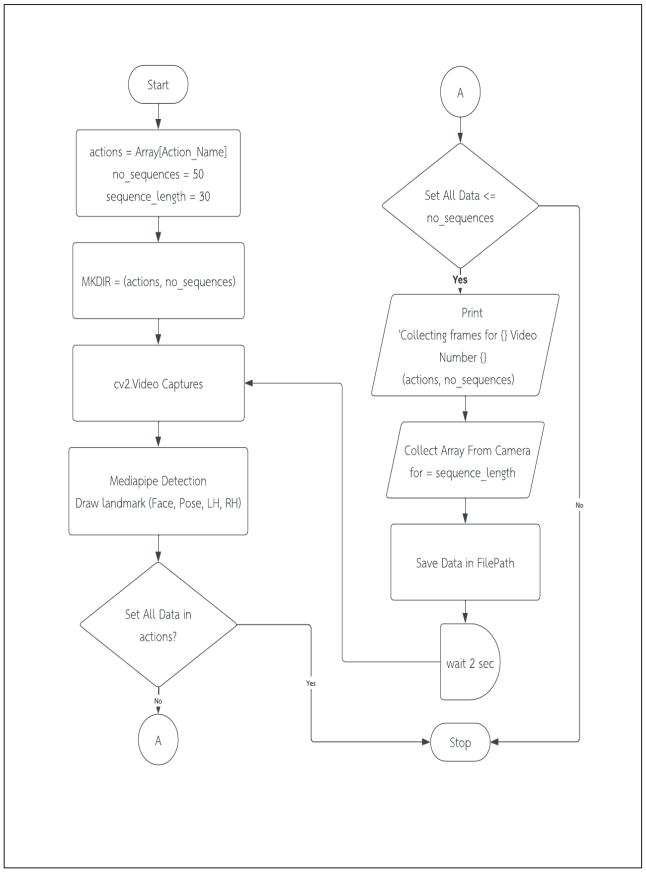

Figure 2. *Flow Chart shows the machine learning sequence based on KERAS.*

Based on the flow diagram, the developer designed the Dataset storage system, starting with creating variables to store gesture meanings in Array format and creating variables to define the number of cycles. (no_sequences) to a value of 50, meaning that 50 cycles will be stored, 30 frames per cycle (sequence_length). After that, create an actions folder to store the Dataset separated by poses and the number of cycles. The machine learning based the approach typically trains a sentiment classifier using features by KERAS and LSTM [24].

Then use OpenCV to activate the camera and use Media Pipe to detect points on the body, edit the face, shoulder, left hand, and right hand, then create 1,662 points and create an array in the npy extension format.

Then begin collecting the Dataset, sorting them according to the folder of actions, repeating 50 rounds, 30 frames each, and then saving the Array file (.npy) in the order of the frames shown in Figure 3.

From the Context Diagram, it shows the basic data flow that occurs within the system. There is a lot of information from people with disabilities sign language gestures. Then the system will move to search from the dataset then make a comparison. Before using the information shown results through information to users and to communicate with people with disabilities shown in Figure 4.

From the Data Flow Diagram, it shows the flow of data. and various subsystems that occur within the program shown in Figure 5.

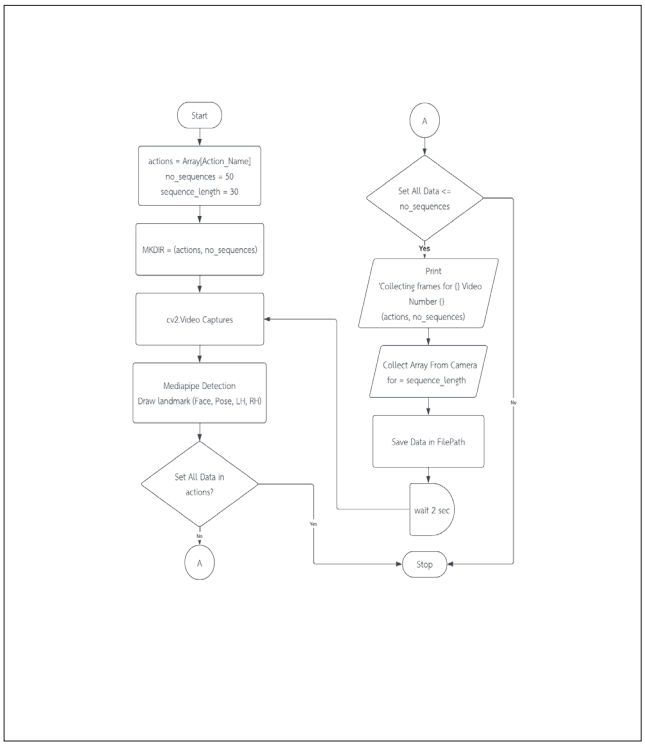

Figure 3. *Flow Chart Dataset Storage System.*

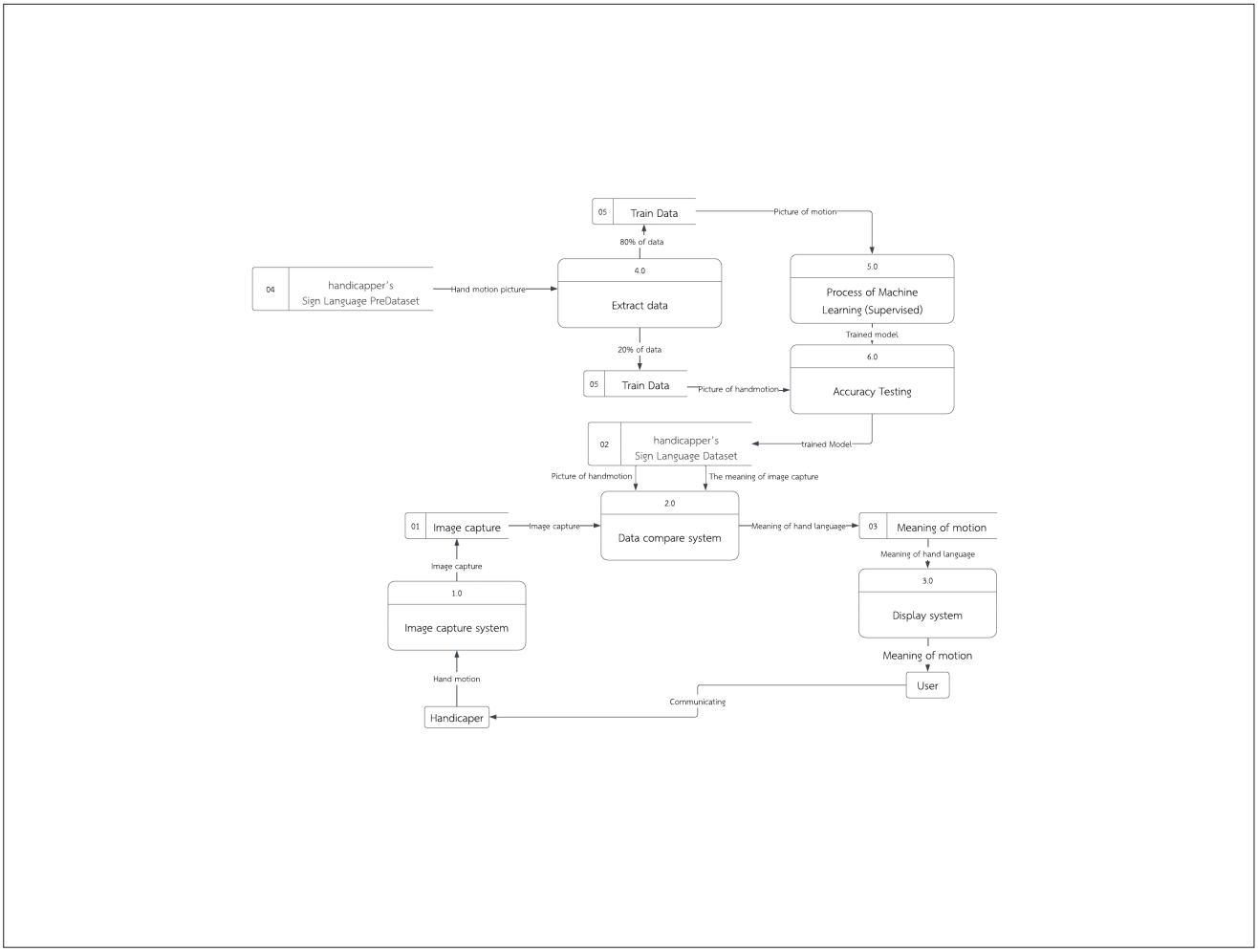

Figure 4. *Data Flow Diagram shows the flow of data in the system.*

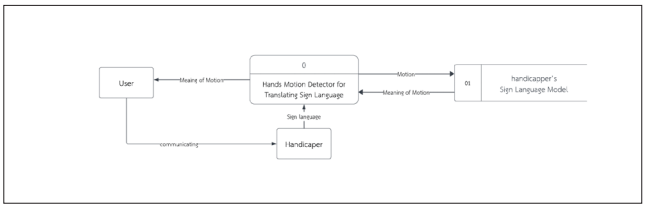

Figure 5. *Context Diagram.*

3.2 Implementation of Datasets

Begin developing the system by collecting a dataset to track hand and body gestures according to the available dataset and using machine learning technology. This program used Datasets Benchmark from MNIST on website https://www.kaggle.com/datasets/datamunge/sign-language-mnist [25]. By giving examples to teach, remember and be able to separate different gestures from each other. In this step, the developer has stored a dataset of 50 sign Thai language words shown in Table 1 that can be used in everyday life, including:

Table 1. *Words stored as a Datasets.*

สวัสดี	ขอบคุณ	ขอโทษ	ใช่	ไม่
สบายดี	ไม่เป็นไร	ชื่อ	นามสกุล	ภาษามือ
ผู้พิการทางการได้ยิน	ผู้มีการได้ยินปกติ	ไม่สบาย	น้ำมูก	หิวน้ำ
เจ็บคอ	อุณหภูมิ	ฉันพบกับคุณ	คุณพลกัน	ไว้พบกันใหม่
โชคดี	วัน	วันจันทร์	วันอังคาร	วันพุธ
วันพฤหัสบดี	วันศุกร์	วันเสาร์	วันอาทิตย์	เดือน
วันนี้	พรุ่งนี้	เมื่อวาน	สัปดาห์	ปี
เวลา	คน	ผู้ใหญ่	ผู้ชาย	ผู้หญิง
เด็ก	ฉัน	คุณ	พวกเรา	อะไร
ทำไม	ที่ไหน	เมื่อไหร่	อย่างไร	หิว

3.3 The source code of the Jupyter Notebook Program

This program is implemented in Python Language on the Jupyter Notebook Program for the functionality of OpenCV, to train dataset on machine learning and the results display section in this program shown in Figure 6, 7, 8.

```
font_path = 'Kanit-Medium.ttf'
font_size = 20
font = ImageFont.truetype(font_path, font_size)
mp_holistic = mp.solutions.holistic
holistic = mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
round_interval_seconds = 2
for action in actions:
draw_styled_landmarks(image, results)
for sequence in range(no_sequences):
pil_image = Image.new('RGB', (640, 480), color=(255, 255, 255))
draw = ImageDraw.Draw(pil_image)
draw.text((120, 200), 'เริ่มการบันทึกสำหรับ {} วิดีโอที่ {}'.format(action,sequence),
fill=(0, 255, 0), font=font)
initial_message_image = cv2.cvtColor(np.array(pil_image),cv2.COLOR_RGB2BGR)
cv2.imshow('Processed Image', initial_message_image)
cv2.waitKey(2000)
for frame_num in range(sequence_length):
ret, frame = cap.read()
image, results = mediapipe_detection(frame, holistic)
draw_styled_landmarks(image, results)
pil_image = Image.fromarray(cv2.cvtColor(image,cv2.COLOR_BGR2RGB))
draw = ImageDraw.Draw(pil_image)
draw.text((15, 12), 'กำลังบันทึกเฟรมสำหรับ {} วิดีโอที่'.format(action, sequence),
fill=(0, 0, 255), font=font)
keypoints = extract_keypoints(results)
npy_path = os.path.join(DATA_PATH, action, str(sequence),str(frame_num))
np.save(npy_path, keypoints)
result_image = cv2.cvtColor(np.array(pil_image),cv2.COLOR_RGB2BGR)
cv2.imshow('Processed Image', result_image)
if cv2.waitKey(10) & 0xFF == ord('q'):
break
time.sleep(round_interval_seconds)
cap.release()
cv2.destroyAllWindows()
```

Figure 6. *Part of the Dataset collection function.*

```
from keras.layers import Flatten, Dropout
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30, 1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Flatten())
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
from keras.callbacks import EarlyStopping, ModelCheckpoint
es = EarlyStopping(monitor = 'val_acc', min_delta = 0.01, patience = 4,verbose = 1)
mc = ModelCheckpoint("realth.keras", monitor = "val_acc", verbose = 1,save_best_only = True)
cb = [es, mc]
model.fit(X_train, y_train, epochs=100, callbacks=[cb], validation_split =0.3)
model.summary()
```

Figure 7. *Train Model section.*

```
sequence = []
sentence = []
predictions = []
thai_font = ImageFont.truetype(thai_font_path, font_size)
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:
while cap.isOpened():
image, results = mediapipe_detection(frame, holistic)
print(results)
draw_styled_landmarks(image, results)
keypoints = extract_keypoints(results)
sequence.append(keypoints)
sequence = sequence[-30:]
if len(sequence) == 30:
res = model.predict(np.expand_dims(sequence, axis=0))[0]
print(actions[np.argmax(res)])
predictions.append(np.argmax(res))
if np.unique(predictions[-10:])[0] == np.argmax(res):
if res[np.argmax(res)] > threshold:
if len(sentence) > 0:
if actions[np.argmax(res)] != sentence[-1]:
sentence.append(actions[np.argmax(res)])
else:
sentence.append(actions[np.argmax(res)])
if len(sentence) > 5:
sentence = sentence[-5:]
pillow_image = Image.fromarray(cv2.cvtColor(image,
cv2.COLOR_BGR2RGB))
draw = ImageDraw.Draw(pillow_image)
draw.rectangle([(0, 0), (640, 60)], fill=(245, 117, 16))
draw.text((3, 10), ' '.join(sentence), font=thai_font, fill=(255,
255, 255))# Convert Pillow image back to OpenCV format for display
image = cv2.cvtColor(np.array(pillow_image), cv2.COLOR_RGB2BGR)
cv2.imshow('Sign Language Translator', image)
key = cv2.waitKey(1)
if key & 0xFF == ord('q'):
break
cap.release()
cv2.destroyAllWindows(
```

Figure 8. *Results display section.*

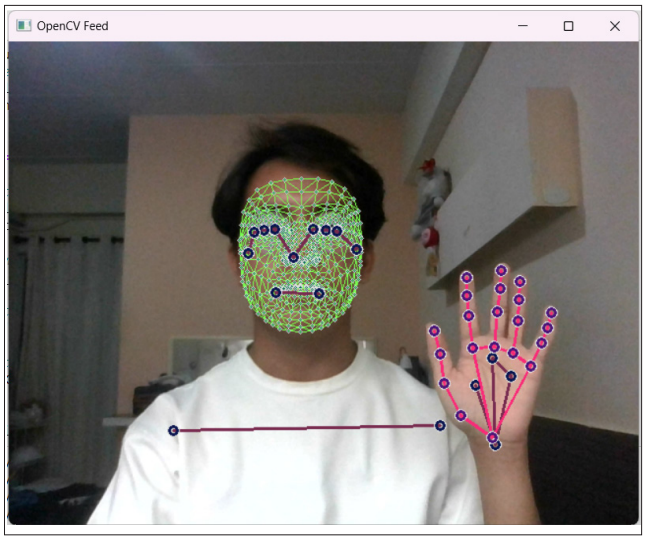

Figure 9. *Train Model section.*

The program made it possible to send images captured from the Video Capture program to check the arrangement of important points again. Through the operation of the Jupyter Notebook program as shown in Figure 10.

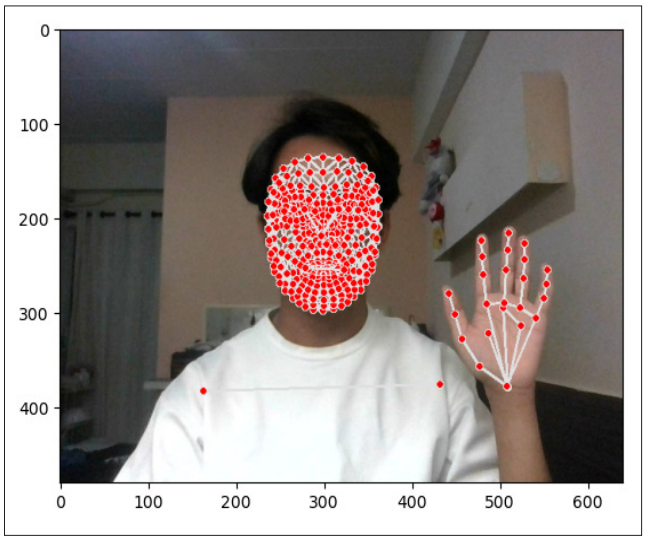

Figure 10. *Image obtained from the gesture detection program.*

4. The Performance of Program Hands Motion Detector for Translating Disabilities People's Sign Language

4.1 Sign language gesture detection

From the development of a sign language gesture detection system used OpenCV to enable access to the device's camera. Then created a program window to work with the Media pipe to detect points. Various features that have been specified then create colored dots for observation, it is called "Landmark" which the developer has used to detect, including Face or face, Pose or posture around the shoulder, Left Hand or left hand and Right Hand or right hand by setting the confidence in a program window value to 0.5, resulting shown in Figure 9.

After that, a function will be created to extract keypoints from the results obtained from the gesture tracking and face capture systems Media pipe. This function worked by receiving all the results from the Media pipe system and then selecting the keypoints of interest in the form of an array that stores the X, Y, and Z values of various Keypoints and saves them to a file in Array format with the extension .npy. If the Keypoints were not detected from the capture, they will substitute the value 0 into the Array to prevent A NULL value occurred shown in Figure 11.

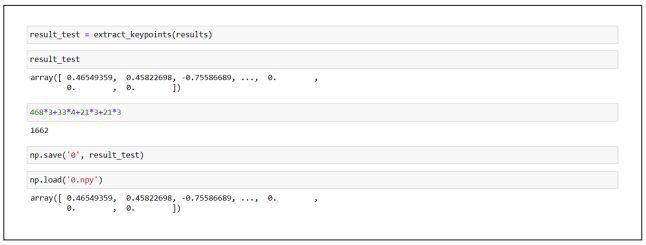

Figure 11. *Example of displaying values in an array stored from a keypoint.*

4.2 Datasets Benchmark collection system

The Datasets Benchmark collection system started by creating a folder to store the array obtained from collecting the Dataset through Python commands. It starts by creating an array according to the name of the pose you want to store. and create a folder to store the number of rounds and the frame that is stored in the system. You will get a folder as shown in Figure 12.

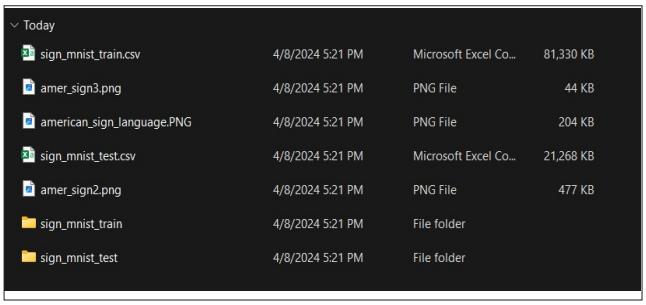

Figure 12. *Folder for storing datasets benchmark.*

The developer chosen to create the folder via command. Because it will allow the system to iterate over the collection of the Dataset according to the array and the values in the variables of the pose name, number of cycles, and number of frames. This making it is possible to collect Datasets continuously, easily and with greater accuracy. rather than creating folders manually.

4.3 Results of training for AI

During the training, a log file will be created to track the results of the training via Localhost:6006 of the Tensor Board. It will be created according to the path of the developed program shown in Figure 13.

The results of the trial found that the model that had been trained had an accuracy of 76.40%, resulting in a model file for use in further development, file name Model.h5.

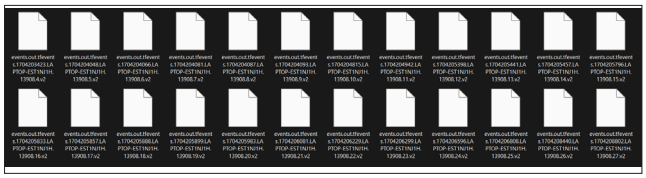

Figure 13. *Logs recorded during training.*

After that, the collected Dataset, which had 75,000 Array formats (50 words, 50 times each word, 30 frames at a time) will be used in the Long-Short Term Model format, RNN format. To make it possible to recognize patterns for a long time, it is effective for prediction problems. It is sequential because previous data can be collected and used in processing and then trained with KERAS using the Adam algorithm.

In the MNIST dataset benchmark, it is suggested that a training set of approximately 60,000 datasets samples is sufficient. However, in our experiments using a training set of 50 words, we obtained 75,000 datasets samples, achieving an accuracy of 76.40%, which is considered adequate. When the training set size was increased to 100 words, resulting in 150,000 datasets samples, the accuracy improved by an additional 5-10%. Despite this improvement, the increased computational requirements pose a challenge for implementation on smartphone devices, given the current performance limitations.

This will result in a trained model. To be used in programs that detect in real time shown in Figure 14.

Testing: Test the system. By testing sign language translation to be accurate. At not less than 75 percent of all poses, by checking the training results each time through TensorFlow that was accessed through Localhost:6006 which will be displayed through the form of a line graph. This is the accuracy value obtained from each training session shown in Figure 15.

From the work of the model that has been tested, the developer has introduced the model to use with the system in order to show the performance results, it was found that the model could work correctly and accurately, but there was a slowness in processing. This is caused by the limitations of the equipment used to train the model, including its ability to display results. This causes a delay of approximately 5-10 seconds after performing the corresponding gesture. The display format is as shown in Figure 16, 17, 18.

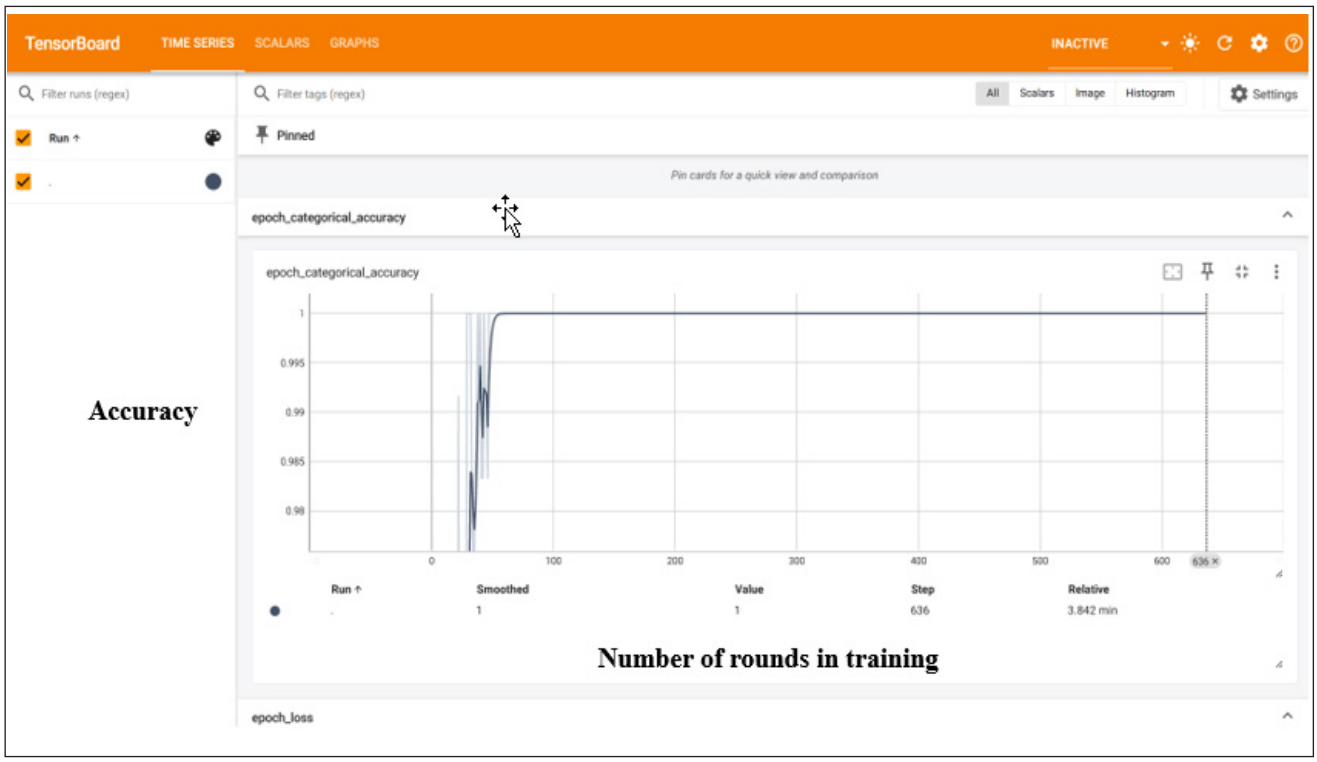

Figure 14. *Line graph showing the accuracy of training.*

Figure 15. *Line graph showing the accuracy of training.*

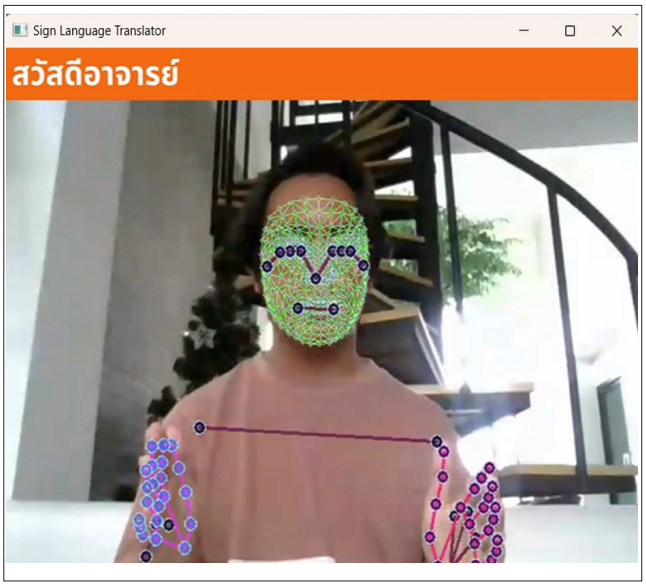

Figure 16. *Image of the display system hello teacher.*

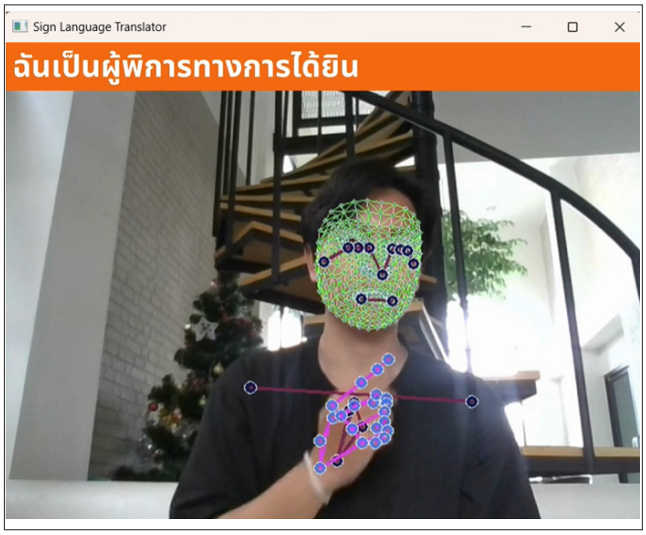

Figure 17. *Image of the display system I am hearing impaired.*

Figure 18. *Example of a dataset collection image (hello, thank you, sorry, and how are you).*

5. Conclusions

From the results of using the Model together with the display section, it was found that the Model can work well. Within the training dataset gesture experiment but problems may be encountered if gestures are unclear, gestures are wrong, gestures that are not in the dataset, and gestures that are similar. It may cause the system to classify the wrong gestures. Including limitations from insufficiently effective training depend on equipment causing the system to be delayed in comparison and display the meaning of the gesture for approximately 5-10 seconds, depending on the difficulty and the similarity of gestures.

As a result of training the model, the accuracy of the work was 76.40%, which is considered to be in the good range. and higher than the target value that has been set according to the scope of the project.

The operation of the display is clear. There is a display from the camera showing various points on the body that are important points of the model and can correctly display the meaning of the gesture as the model sends the values. It can be accessed. and can stop using the system via directly closing the program.

The developer encountered a problem with the library that was selected for use in the Image Processing section, OpenCV, which was unable to display the language in the system like UTF-8, causing various display sections to be unable to display Thai language. The developer therefore chose to use the Pillow Library, which is used in the Image Processing section, to work with OpenCV, which is to use OpenCV to access working with the camera and use Pillow to do all the text display work. Therefore, allowing the system to display Thai language.

The future work, we can use this program to apply on smartphones but it must depend on the efficiency of smartphones. We plan to expand the vocabulary from 50 words to 100 words and run the enhanced model on our server and through an API accessible on smartphones. This expansion aims to improve the model's accuracy. By using a larger dataset and more

efficient training equipment, we anticipate achieving higher accuracy and faster performance in the program.

Acknowledgements

Compliance with Ethical Standards Funding This research was financially supported by Rajamangala University of Technology Thanyaburi, RMUTT. Authors' Contributions

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

6. References

[1] Bangkokbiznew, *Thailand.* Available Online at https://www.bangkokbiznews.com/corporate-moves/lifestyle/judprakai/1041542, accessed on 27 April 2024.

[2] S. Naqvi, J. Ohene-Djan, and R. Spiegel. "Testing the effectiveness of digital representations of sign language content." *the Instructional Technology and Education of the Deaf Symposium Rochester,* New York, 2005.

[3] J. Whitehill. *Automatic Real-time Facial Expression Recognition for Signed Language Translation.* Master's thesis. University of the Western Cape. Computer Science, 2006.

[4] V. M. Segers. "The efficacy of the eigenvector approach to south african sign language identification." *In South African Telecommunication Networks and Applications Conference 2009*, pp. 363-366, 2009.

[5] P. Li. *Hand shape estimation for South African Sign Language.* Master's thesis University of the Western Cape Computer Science, 2010.

[6] C. Rajah. *Chereme-based recognition of isolated, dynamic gestures from South African Sign Language with Hidden Markov Models.* Master's thesis University of the Western Cape Computer Science, 2006.

[7] C.-S. Lee, Z. Bien, G. T. Park, W. Jang, J. S. Kim, and S. K. Kim. "Real-Time recognition system of Korean sign language based on elementary components." *In Proceedings of the 6th IEEE International Conference on Fuzzy Systems 1997,* pp. 1463-1468, 1997.

[8] Y. H. Lee and C. Y. Tssai. "Taiwan sign language (TSL) recognition based on 3D data and neural networks." *Expert Systems with Applications,* Vol. 36, No. 2, pp. 1123-1128, 2007.

[9] N. D. Sandhya and K. R. Charanjeet. "A review on Machine Learning Techniques." *International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC),* Vol. 4, No. 3, pp. 395-399, 2019.

[10] D. M. Kishore, S. Bindu, and N. K. Manjunath. "Estimation of yoga postures using machine learning techniques." *International Journal of Yoga,* Vol. 15, No. 2, pp. 137-143, May-August, 2022.

[11] Y. Quiñonez, C. Lizarraga, and R. Aguayo. "Machine Learning solutions with MediaPipe." *2022 11th International Conference On Software Process Improvement (CIMPS),* pp. 212-215, 2022.

[12] MediaPipe Pose. Available online: https://google.github.io/mediapipe/solutions/pose.html, accessed on 28 December 2023.

[13] M. Nour, M. Gardoni, J. Renaud, and S. Gauthier. "Real-time detection and motivation of eating activity in elderly people with dementia using pose estimation with TensorFlow and OpenCV." *Advances in Social Sciences Research Journal,* Vol. 8, No. 3, pp. 28-34, 2021.

[14] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Inc. 2008.

[15] OpenCV Open Source Computer Vision library. Available Online at http://opencv.willowgarage.com/wiki/, accessed on 27 April 2024.

[16] M. Ashraf, S. M. Ahmad, N. A. Ganai, R.A. Shah, M. Zaman, S. A. Khan, and A. Shah. "Prediction of cardiovascular disease through cutting-edge deep learning

technologies: an empirical study based on TENSOR FLOW, PYTORCH and KERAS." *In International Conference on Innovative Computing and Communications. Proceedings of ICICC Springer Singapore 2020,* Vol. 1, pp. 239-255, 2020.

[17] S. W. B. Tan, P. K. Naraharisetti, S. K. Chin, and L. Y. Lee. "Simple Visual-Aided Automated Titration Using the Python Programming Language." *Journal of Chemical Education,* Vol. 97, pp. 850-854, 2020.

[18] D. Ulus, V. Kraevo, and A. Sheffer. "D-Charts: Quasi-Developable Mesh Segmentation, Computer Graphics." *Eurographics 2005,* Vol. 24, No. 3, pp. 981-990, 2005.

[19] A. Sheffer, B. Levy, M. Mogilnitsk, and A. Bogomyakov. "ABF++: Fast and Robust Angle Based Flattening." *ACM Transactions on Graphics,* Vol. 24, No. 2, pp. 311-330, 2005.

[20] C. Damrongekarun, L. Pisitpipattana, S. Waijanya, and N. Promrit. "Development of Thai Sign Language Detection and Conversion System into Thai with Deep Learning." *KKU Science Journal,* Vol. 51, No. 3, pp. 216-225, 2023.

[21] X. Shi, Z. Chen, H. Wang, and D. Y. Yeung. "Convolutional LSTM network: a machine learning approach for precipitation nowcasting." *Advances in Neural Information Processing Systems 28 (NIPS 2015),* 2015.

[22] S. Kim, S. Hong, M. Joh, and S. Song. "Deep Rain: ConvLSTM network for precipitation prediction using multichannel radar data." *7th International Workshop On Climate Informatics,* pp. 20-22, September, 2017.

[23] N. K. Mai, D. Yang, I. Shin, H. Kim, and M. Hwang. "Dprelu: Dynamic parametric rectified linear unit." *The 9th International Conference on Smart Media and Applications,* pp. 121-125, 2020.

[24] A. Kumar and A. Abraham. "Opinion Mining to Assist User Acceptance Testing for Open-Beta Versions." *Journal of Information Assurance and Security,* Vol. 12, No. 4, pp. 146-153, 2017.

[25] Tecperson, *Sign Language MNIST.* Available Online at https://www.kaggle.com/datasets/datamunge/sign-language-mnist, accessed on 27 April 2024.