ANG ZI YING

B.Sc. (Hons) Statistical Computing and Operations Research

2024

LEVERAGING 3D SKELETON VIDEO EXTRACTION AND DEEP LEARNING FOR REAL-TIME SIGN LANGUAGE RECOGNITION MODEL

ANG ZI YING

BACHELOR OF SCIENCE (HONOURS) STATISTICAL COMPUTING AND OPERATIONS RESEARCH

FACULTY OF SCIENCE

UNIVERSITY TUNKU ABDUL RAHMAN

JUNE 2024

LEVERAGING 3D SKELETON VIDEO EXTRACTION AND DEEP LEARNING FOR REAL-TIME SIGN LANGUAGE RECOGNITION MODEL

By

ANG ZI YING

A project report submitted to the

Department of Physical and Mathematical Science

Faculty of Science

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements for the degree of

Bachelor of Science (Honours)

Statistical Computing and Operations Research

June 2024

ABSTRACT

LEVERAGING 3D SKELETON VIDEO EXTRACTION AND DEEP LEARNING FOR REAL-TIME SIGN LANGUAGE RECOGNITION MODEL

ANG ZI YING

Sign language recognition is recognized as key research for reducing communication barriers between deaf and hearing people. Over the past two decades, researchers have shown great interest in sign language recognition due to technological advances. Researchers have conducted extensive studies on sign language recognition, but developing a highly accurate real-time model is still difficult due to the time-consuming nature of sign language video recognition. Due to the lack of a Malaysian Sign Language dataset, a video-based Malaysian Sign Language dataset (MSL10) was created and will further validate the results with the Argentinean Sign Language dataset (LSA64). This study aims to propose a combination that maintains high accuracy and reduces computational time, which consists of key points of important features, and a deep learning recurrent neural network model, a high-accuracy and low-computational model suitable for real-time sign language recognition. MediaPipe's 3D skeleton video helps in removing unnecessary information while reducing computation time. Compared to whole-body feature analysis, this study shows that hand features can effectively reduce computation time and improve accuracy. In the study, it was also found that the two-layer BiLSTM

model has the best performance in terms of accuracy and computation time as compared to the LSTM and three-layer BiLSTM models.

ACKNOWLEDGEMENTS

I am pleased to undertake the final year project entitled “Leveraging 3D Skeleton Video Extraction and Deep Learning for Real-Time Sign Language Recognition Model”. I would like to take this opportunity to thank every individual involved in this project. High appreciation is given to my supervisor, Ms. Chin Fung Yuen, a lecturer at the Faculty of Science at Universiti Tunku Abdul Rahman, for her strong and timely support in developing my project and writing this report. I would like to express my deepest gratitude to my external consultant, Dr. Lim Foo Weng, for providing me with his invaluable guidance throughout this project. Most importantly, I am grateful to Universiti Tunku Abdul Rahman, for allowing me to study for the program Bachelor of Science (Hons) in Statistical Computing and Operations Research. Lastly, special thanks to my family members and friends who encourage me to complete this project.

DECLARATION

I hereby declare that the project report is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

ANG ZI YING

APPROVAL SHEET

This project report entitled "**LEVERAGING 3D SKELETON VIDEO EXTRACTION AND DEEP LEARNING FOR REAL-TIME SIGN LANGUAGE RECOGNITION MODEL**" was prepared by ANG ZI YING and submitted as partial fulfilment of the requirements for the degree of Bachelor of Science (Hons) Statistical Computing and Operations Research at Universiti Tunku Abdul Rahman.

Approved by:

(Ms. Chin Fung Yuen) Date: **14 Apr 2024**

Supervisor

Department of Physical and Mathematical Science

Faculty of Science

Universiti Tunku Abdul Rahman

FACULTY OF SCIENCE

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 26/02/2024

PERMISSION SHEET

It is hereby certified that **ANG ZI YING** (ID No: **19ADB01751**) has completed this final year project entitled "**LEVERAGING 3D SKELETON VIDEO EXTRACTION AND DEEP LEARNING FOR REAL-TIME SIGN LANGUAGE RECOGNITION MODEL**" under the supervision of Ms. Chin Fung Yuen (Supervisor) from the Department of Physical and Mathematical Science, Faculty of Science.

I hereby give permission to the University to upload the softcopy of my final year project in pdf format into the UTAR Institutional Repository, which may be made accessible to the UTAR community and public.

Yours truly,

(ANG ZI YING)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

LSTM	Long Short-Term Memory
SVM	Support Vector Machine
RFE	Recursive Feature Elimination
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MSL	Malaysian Sign Language
ASL	American Sign Language
KNNOR	K Nearest Neighbor Oversampling
LogitBoost	Logit Boosting
CSLR	Continuous Sign Language Model
ISLR	Isolated Sign Language Recognition Model
MFD	Malaysian Federation of the Deaf
SLR	Sign Language Recognition
AUTSL	Ankara University Turkish
GRU	Gated Recurrent Unit
ResNet	Residual Network

VGG	Visual Geometry Group
PCA	Principal Component Analysis
FLD	Fisher’s Linear Discriminant
k-NN	K-nearest neighbor
HHMs	hidden Markov models
RGB	Red, Green and Blue
OBB	Oriented Bounding Box
3D	Three-dimensional space
1D	One-dimensional space
DNN	Deep Neural Network
VGP	Vanishing Gradient Problem
OpenCV	Open-Source Computer Vision Library
ANNs	Artificial Neural Networks
Adam	Adaptive Moment Estimation
BIM	Bahasa Isyarat Malaysia

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Deaf people use sign language to express their emotions and needs, and these sign languages include body language, facial expressions, and gestures. Communication barriers still exist between the deaf community and the public due to the limited understanding of sign language. Therefore, there is a need for a technology that facilitates communication between hearing and deaf people. Real-time gesture detection in sign language video streams requires a model with fast and accurate gesture recognition techniques. Computer vision technology has come a long way in the last 20 years, improving human pose estimation, vision-based sign language, and gesture recognition.

The Continuous Sign Language Recognition Model (CSLR) and the Isolated Sign Language Recognition Model (ISLR) are two models used for recognizing sign language (Sharma, Gupta, and Kumar, 2021). The difference between ISLR and CSLR is that the first uses a single image to represent specific hand shapes and poses, while the second uses a sequence of images to represent a moving gesture (Aloysius and Geetha, 2020). CSLR is also known as dynamic gesture recognition (Abdalla and Hemayed, 2013). There are several methods for solving CSLR recognition problems, and most of the methods fall into two stages (Liao et al., 2019). First, explore algorithms for computing hand and gesture movement trajectories. Second, the implementation of each sign language picture sequence.

The deaf population in Malaysia primarily uses Malaysian Sign Language (MSL), which is also known as Bahasa Isyarat Malaysia (BIM), as their main form of communication. The establishment of the Malaysian Federation of the Deaf (MFD) in 1998 marked the starting point of Malaysian Sign Language (MSL), which has gained significant popularity within deaf organizations (Hafit et al., 2019). Malaysian Sign Language (MSL) originates from American Sign Language (ASL) (Qodri, Rini Akmeliawati, and Mohammed, 2012). As MSL originated from ASL, numerous local signs from ASL have been added to the MSL, and MSL currently has roughly 75% similarity with ASL (SIL International, 2022). There is no global sign language; hence, different sign languages are used in different countries or areas.

1.2 Problem Statement

Most deep learning models that use images or video data take a longer time to process. On top of that, the noise of the background will lower the accuracy of sign language recognition. The goal of all sign language recognition researchers is to develop a real-time system that accurately translates sign language into words, enabling the deaf-mute community to connect with the hearing community. The real-time sign language recognition model needs to respond quickly, so computation time as well as the accuracy of the real-time sign language recognition model are important.

Features extracted from the video can be a drawback. Extracting unwanted features during the training process decreases accuracy and increases computation time (Rahman et al., 2020). Key points built from the MediaPipe holistic model will include face, pose, and hand key points (Indriani, Harris, and Agoes, 2021). MediaPipe holistic model can create a complete landmark for the human body and enables analysis tasks to cover full-body gestures, poses, and actions. It is crucial to consider how the face and pose features in the dataset impact the computation time and accuracy.

Currently, Long Short-Term Memory (LSTM) is widely used in sign language. Unfortunately, LSTM networks struggle to capture complex temporal dependencies inherent in sign language gestures, resulting in poor performance, especially when capturing bidirectional contexts (Telmo Adão et al., 2023). The LSTM model should be replaced.

Deep learning models have been formed by many layers of neurons; the layers of neurons have also been known as neural networks. The number of layers is selective; a high number of layers will increase the model complexity and enable the ability to learn more information from the dataset, but at the same time, the computation time will increase. Increasing the number of layers in a deep learning model can make the neural network more complex and difficult to train (Dumitru Erhan et al., 2009). A too-high number of layers will lead the deep learning model to be overfitted; if overfitting happens, the model accuracy will drop, and the computation time will increase as much unwanted information has been learned by the model in the training stage. Therefore, it is crucial to determine the number of layers that can maintain high accuracy and low computation time.

It was found that there is a lack of video-based Malaysian Sign Language (MSL) datasets in Malaysia. The exploitation of real-time continuous Malaysian Sign Language (MSL) recognition models is also crucial for the development of Malaysian Sign Language (MSL). Although MSL real-time recognition systems are available on the market, the algorithms for real-time recognition systems are not open source. For real-time recognition models, high accuracy, and low computation time for detecting the correct sign language are very important. Therefore, it is crucial to develop an open-source MSL real-time recognition model that can be the basis for researchers in MSL real-time recognition systems.

1.3 Objectives of Research

The main research objective for this project is:

- To create a fast and high-accuracy real-time Malaysian sign language recognition model.

The subsequent research objective for this project is:

- To convert the video-based dataset to 3D skeleton key points and exclude background information.
- To compare the performance of LSTM networks and Bidirectional Long Short-Term Memory (BiLSTM).
- To compare the performance of hands, posture, and facial key points with solely hand key points.
- To evaluate the effectiveness of deep learning models with varying numbers of layers.
- To create a dataset of Malaysian sign language videos.

1.4. Significance of Study

This study is crucial because it will enhance the understanding of sign language recognition while also improving accessibility and communication for the deaf community and promoting assistive technology research. This study is significant in various ways:

1. Bridging the communication gap. By developing a real-time recognition model of Malaysian Sign Language (MSL), this study contributes to bridging the communication gap between the deaf and normal communities in Malaysia. Accessible communication tools are essential to promote understanding between different linguistic groups.

2. Create the MSL10 dataset. The MSL10 dataset addresses the lack of an open-source MSL video dataset. This dataset can provide a foundation for developers and researchers working on real-time MSL recognition.

3. Cross-language transferability. The presence of the LSA64 dataset in this study proves that the proposed model applies not only to MSL but also to Argentinian Sign Language, which helps the cross-lingual transferability of the sign language recognition model. The real-time recognition model enhances its applicability to other national sign languages and shows its potential for multilingual understanding.

4. Deep learning model evaluation. By evaluating the performance of various deep learning designs such as LSTM and BiLSTM models, this study can provide insights into how well the models match the dataset so that the optimal models can be selected for forming real-time recognition algorithms for the MSL10 dataset.

5. Optimizing the number of layers in the model. The performance of the model can be optimized by comparing the number of layers in the BiLSTM model. Finding the right balance between computation time and accuracy is crucial for building real-time recognition models.

6. Focus on hand key points only. Selecting only hand key points highlights the importance of hand key points, whereas pose and face key points the excessive information, which will increase computation time and decrease accuracy. The selection of hand key points highlights the importance of feature selection in sign language recognition.

CHAPTER 2

LITERATURE REVIEW

2.1 Evolution of Sign Language Recognition (SLR)

Sign language recognition can be implemented with one or both hands, and the method can be classified into two main groups: feature-based systems and deep learning systems (Alsharif et al., 2023).

Tsakanikas and Dagiuklas (2018) define feature-based systems as those that extract specific features from input data, such as images or videos. In feature-based systems, local features are implemented into a multilayered model that incorporates face identification, preprocessing, feature extraction, dimension reduction, and classification, as shown in Figure 2.1.1 (Nguyen, 2014).

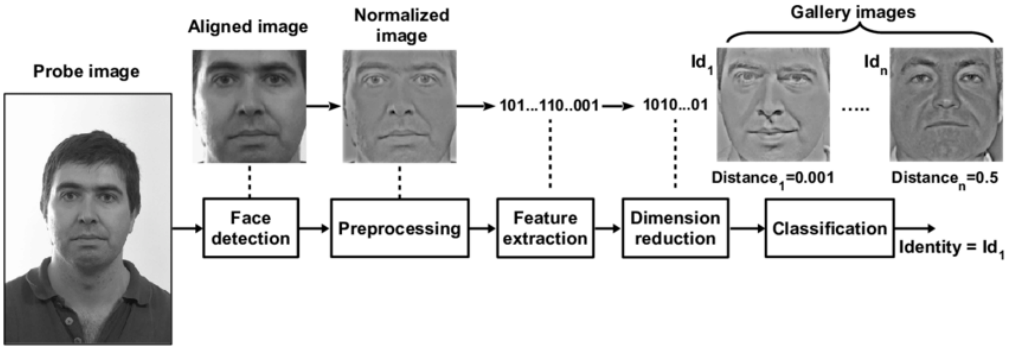


Figure 2.1.1: Architecture for recognizing faces based on local features (Nguyen, 2014)

Nguyen (2014) demonstrated the entire feature-based system process using a face detection framework as a starting point. The image is initially taken by a camera, and the algorithm will have to recognize the presence or absence of a face. Then, the image is cropped according to the eye coordinates to find the face to reduce the unwanted information and obtain a good frontal face image. The next stage is image normalization, where the cropped images will be processed using a preprocessing technique to eliminate light intensity. The feature extraction method is applied to the images to extract the most important features for classification. The dimension reduction task is carried out by Principal Component Analysis (PCA) and Fisher's Linear Discriminant (FLD). The final stage is the classification process, which uses Support Vector Machines (SVM) and K-nearest neighbor (k-NN).

Other than feature-based systems that focus on face detection, hand-detection feature-based systems have also been highly used in sign language recognition. Chen, Fu, and Huang (2003) implemented a hand gesture recognition system to recognize continuous gestures. As mentioned by Chen, Fu, and Huang (2003), the first stage in building a gesture recognition system is to extract the hand region and track the moving hand. Then spatial and temporal features are characterized and combined into feature vectors. Lastly, hidden Markov models (HHMs) are applied to recognize the feature vectors based on different scores.

The assessment of hand-shaped features serves as the foundation for certain related research. Kim et al. (2017) used deep neural networks to solve the Arabic finger spelling recognition problem based on hand form characteristics. Nevertheless, as handshape rather than hand movements are considered for deep neural networks, the system is still restricted to simple action gestures. Furthermore, there is always a calculation delay when photos or videos are used as model inputs.

Ge et al. first proposed the Hand PointNet method for feature extraction by transforming data types in 2018. The method estimates hand pose by directly analyzing 3D point cloud data representing the visible surface of the hand. Including the fingertip refinement network, the method outperforms existing methods. Ge et al. (2018) illustrated how the method can accurately predict the 3D shape and position of the entire hand based on a single RGB image in Figure 2.1.2.

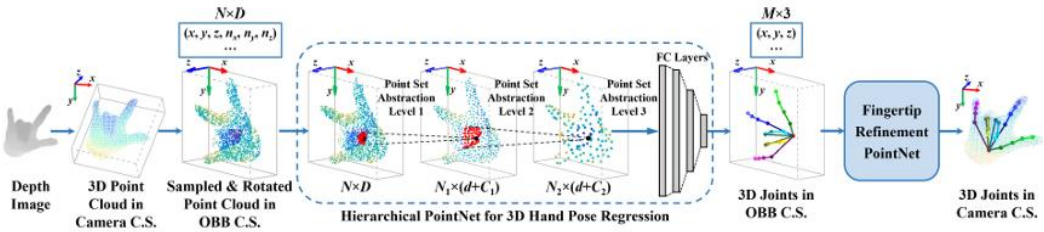


Figure 2.1.2: Overview of the Hand PointNet-based technique for estimating 3D hand poses in single-depth pictures (Ge et al., 2018)

MediaPipe is a library invented by Google. It is a tool that creates 3D skeleton videos by extracting critical points in 3 dimensions from the face, pose, and hand. The difference between PointNet and MediaPipe is that PointNet is designed especially for processing point cloud data (A. Garcia-Garcia et al., 2016), whereas MediaPipe is a tool to handle 3D data in a large framework of computer vision tasks (Bora et al., 2023). Researchers have changed their focus to employing MediaPipe since MediaPipe was introduced.

Other than MediaPipe Holistic, other libraries have been used for sign language recognition. Kavana et al. (2022) adopted the hand landmark model and a palm detector model from the MediaPipe hands library for sign language recognition. Although there are other libraries for sign language recognition, MediaPipe is still the most popular library for sign language recognition.

There are a few studies that employ MediaPipe Holistic for feature extraction. Marais et al. (2022) compared a few feature extraction methods and concluded that the model that uses MediaPipe Holistic to extract hand key points has higher computational efficiency. Other than that, Selvaraj et al. (2021) used the 75 key points in MediaPipe holistic pose extraction for pose estimation, further proving that MediaPipe is highly used for different functions.

A study was conducted to compare MediaPipe holistic models with and without facial key points (Samaan et al., 2022). Without facial key points, the model can achieve higher accuracy in dynamic sign language recognition.

Other than PointNet, MediaPipe has also frequently been compared because they are famous for computer vision for human posture estimation. OpenPose is famous for multi-person human pose estimation. Figure 2.1.3 shows the OpenPose 25 key points, and the multi-person pose estimation with OpenPose.

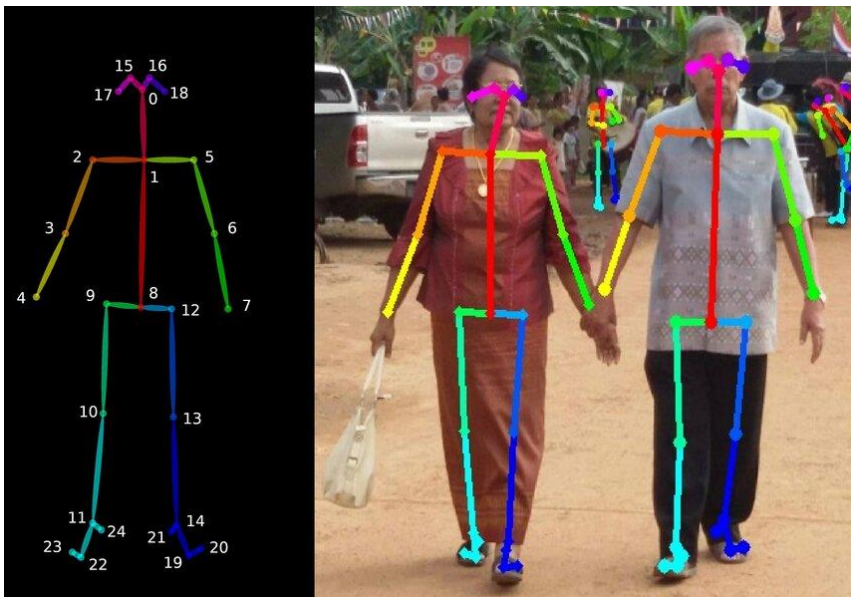


Figure 2.1.3: The left picture shows the OpenPose 25 key points, and the right picture shows multi-person pose estimation with OpenPose (Namburi & Hengsanankun, 2022).

There are a few comparison studies conducted between MediaPipe and OpenPose, including one comparing the MediaPipe overall pose estimation framework to the OpenPose single network whole body posture estimation by Amit Moryossef et al. (2021). Other than that, Necati Cihan Camgoz et al. (2020) presented the SLR transformer in the OpenPose model and the MediaPipe system with a two-layer BiLSTM system. The results of the study show that the accuracy attained by combining MediaPipe and BiLSTM is higher than that of OpenPose and SLR. According to Lin, Jiao, and Zhao (2023), OpenPose fails to recognize 3D human posture data. Its robustness is poor, and the demands for computer graphics card hardware are costly as well. Therefore, in this study, MediaPipe is used for the key point construction. From many MediaPipe libraries, MediaPipe holistic is selected because of the high number of key points that can be collected from the model.

2.2 Deep Learning Models

The early stage of deep learning began with McCulloch and Pitts (1943), who studied the logic underpinning neural activity and the complex interactions between brain events and logical propositions.

2.2.1 Deep Neural Network (DNN) and Convolutional Neural Network (CNN)

A Deep Neural Network (DNN) uses feature learning to map input features and outputs, and the mapping process occurs within multiple connected layers. Each layer contains multiple interconnected neutrons to learn the relationship between inputs and outputs. Figure 2.2.1.1 shows the DNN with a single computational neuron.

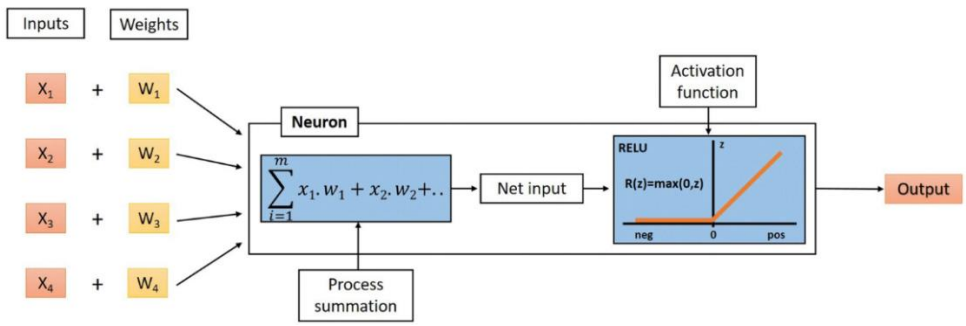


Figure 2.2.1.1: Deep Neural Network (DNN) with a single computational neuron (Georgevici and Terblanche, 2019).

After the development of DNN, the Convolutional Neural Network (CNN) was created as an obvious choice for picture recognition. It is a multi-layer network trained with gradient descent to learn complicated, high-dimensional, non-linear mappings from vast collections of samples (Lecun et al., 1998). CNNs have demonstrated exceptional performance in picture segmentation, classification, detection, and retrieval-related tasks, making them one of the best learning algorithms for comprehending image content (Cireşan et al., 2012; Liu, Deng, and Yang, 2018). Figure 2.2.1.2 shows the structure of the CNN proposed by Muhammad Mizanur Rahaman et al. (2019).

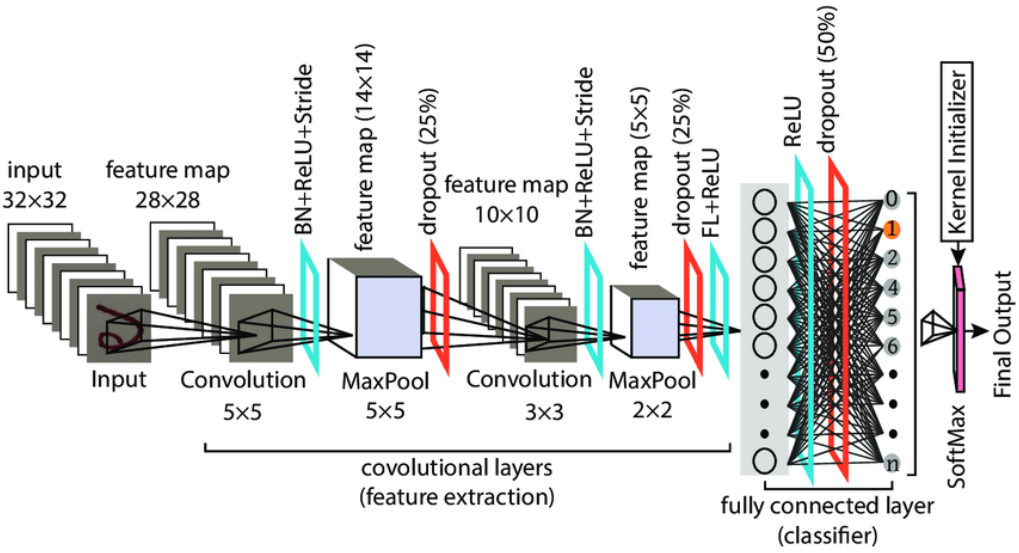


Figure 2.2.1.2: Convolution Neural Network (Muhammad Mizanur Rahaman et al., 2019)

2.2.2 Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (Bi-LSTM)

Standard neural networks are limited because they assume that training and test examples are independent of one another, and the examples are vectors with a fixed length (Lipton, Berkowitz, and Elkan, 2015). To handle sequential data, Recurrent Neural Networks (RNN) are built upon recent and historical data, and in this way, RNN can handle sequential data hierarchically (Avraam Tsantekidis, Nikolaos Passalis, and Anastasios Tefas, 2022). In long-term dependencies, RNNs are vulnerable to the vanishing gradient problem (VGP), which stops the network from learning and results in low prediction accuracy (Eshraghian et al., 2023). However, this weakness poses no issues for RNN as it can be easily solved using gated recurrent units (GRU) and long short-term memory (LSTM) (Safwan Mahmood Al-Selwi et al., 2023).

After Hochreiter and Schmidhuber (1997) showed how recurrent network algorithms learning superiority over Long Short-Term Memory (LSTM) was invented, RNNs' efficacy grew considerably. Extensions to LSTM-style algorithms include bidirectional LSTM, hierarchical LSTM, and hierarchical attention GRU (Huang et al., 2018). Many sequence modeling tasks, including language translation, time-series forecasting, and speech recognition, have seen the successful application of LSTMs. Because LSTM networks can learn long-term dependencies, they have been explored and applied to the classification of sign language data. Research was conducted to translate Indian Sign Language's static and dynamic signals into speech (Abraham, Nayak and Iqbal, 2019). LSTM models require less computational time compared to CNN models.

The axis-independent architecture of the LSTM model is recognized as AI-LSTM. According to Al Amin Hosain et al. (2019), training AI-LSTM took an average of 25 minutes, and training spatial AI-LSTM took an average of 30 minutes, whereas training 3DCNN and Max3DCNN models required over 20 hours per model. Although LSTM is faster than CNN models, the computation time of LSTM models is also considered to be too slow for large amounts of data. The great gain in recognition accuracy provided by LSTM comes at the cost of increased computational complexity when the size of the model increases (Gers and Schmidhuber, 2000).

The BiLSTM network not only uses LSTM to cope with long-term dependencies but also utilizes future information by architecturally incorporating two forward and backward LSTM layers (Peng et al., 2021). Abduljabbar, Dia, and Tsai (2021) also mentioned that BiLSTM trains the input data in both forward and backward directions. Inspired by the "coarse-to-fine" technique and predictive coding theory, Ling, Zhong, and Li (2022) suggest a multi-scale prediction model for video prediction. By fusing top-down and bottom-up information flows, the model seeks to increase prediction ability while reducing reliance on input data. The BiLSTM design comprises an encoder-decoder network. Smaller BiLSTM hidden states are used to avoid computational overhead and prediction difficulty. Adversarial training is a component of the training strategy that addresses the instability of long-term predictions and sharpens the generated images.

Researchers have extensively used 2-layer BiLSTM models and 3-layer BiLSTM models in sign language recognition. A two-layer Bidirectional LSTM (BiLSTM) network enhances pre-trained networks by tracking temporal dependencies, thus improving the accuracy of the system used (Jella Sandhya and KANCHARLA ANITHASHEELA, 2024). Li et al. (2020) propose a three-layer BiLSTM in their study to search for key actions of their sign language recognition. Researchers often discuss the number of layers appropriate for deep learning models. Increasing the number of hidden layers can enhance training set accuracy; however, additional layers are expected to cause overfitting problems (Li et al., 2020).

In summary, previous studies have involved experiments to reduce facial expressions in the overall MediaPipe key points but have not involved experiments to reduce facial and postural key points in the overall MediaPipe landmarks. The results of the previous study suggest that facial key points are not necessary for sign language recognition. In addition to this, no study provides a step-by-step approach to model a real-time recognition model for Malaysian sign language. Based on what has not been covered in previous studies, this study will experiment with using only hands key points to reduce unnecessary features and computation time. Due to the lack of video based MSL datasets and to create a real-time recognition model, the MSL dataset must be created first. MediaPipe will be used as a tool to convert the video dataset to 3D skeletal coordinates to reduce background noise and retain only 3D key points. The performance of the LSTM model can be further improved by the BiLSTM

model. Since the BiLSTM model has 2 and 3 layers, the performance of both models will be evaluated with a different number of layers. Finally, the best model will be selected based on the results of this study to form a real-time recognition model.

CHAPTER 3

METHODOLOGY

3.1 Dataset Introduction

Due to the lack of a Malaysian Sign Language (MSL) video dataset, the MSL10 dataset was recorded to study Malaysian Sign Language (MSL). The dataset focused on ten words selected from the Malaysian Sign Language dictionary. These words include familiar ones such as "Beli" (to buy), "Benar" (correct), "Kenyang" (full), "Maaf" (sorry), "Makan" (to eat), "Mana" (where), "Minum" (to drink), "Salah" (wrong), "Tandas" (bathroom), and "Tidur" (to sleep).

Each word chosen from the MSL is associated with a specific action. These signs are learned from instructional videos on YouTube, as the MSL books are not cost-free. The dataset consists of 1000 videos in total, with each word repeated 100 times. In the 100 times recording for each word, 50 signs will be recorded using the left hand, while 50 times will be recorded using the right hand to make sure that when changing hands, the sign can still be detected. Throughout the whole recording, only one signer is included in this dataset. A few screenshots illustrating how MSL will be presented are shown in Figures 3.1.1, 3.1.2, 3.1.3, and 3.1.4.

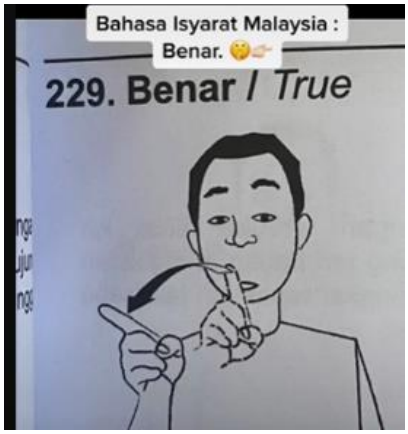


Figure 3.1.1: MSL10 dataset “Benar”

Figure 3.1.2: MSL10 dataset “Maaf”

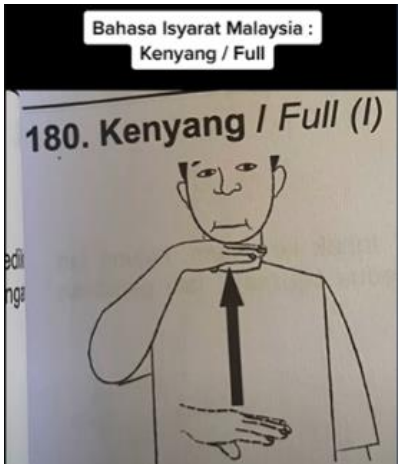


Figure 3.1.3: MSL10 dataset “Kenyang”

Figure 3.1.4: MSL10 dataset “Makan”

The MSL10 dataset is a valuable resource for researchers performing sign language recognition in deep learning training by providing a set of MSL video datasets. In addition, the creation of the MSL10 dataset pushes the pace of research to allow researchers to improve their findings on Malaysian sign language interpreting technology and ultimately create a complete real-time sign language recognition model that will bridge the communication gap between the deaf and normal communities in Malaysia.

Since the MSL10 dataset is self-created data and is small, the LSA64 dataset will be used to represent the large volume of data with multiple signers. The LSA64 dataset contains a wider range of Argentine sign languages which consists of 64 different words. The dataset is open-source data that contains a total of 3200 videos. The LSA64 dataset can be downloaded from https://facundoq.github.io/datasets/lsa64/.

The LSA64 dataset is unique in inviting non-experts in sign language. Specifically, 10 non-experts participated in the creation of the dataset, each repeating 64 different sign language words five times. Each video in the LSA64 dataset captures a specific gesture to facilitate the training of deep learning models. Compared with the MSL10 dataset, this dataset provides a higher number of types of sign language, enabling the study to handle a larger amount of data and study more complex information in the dataset. Figure 3.1.5 shows the cropped image from LSA64 videos.

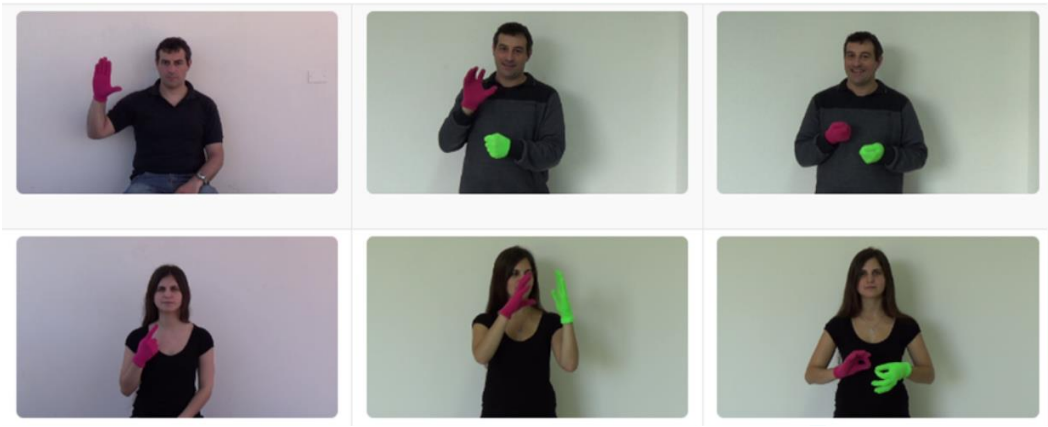


Figure 3.1.5: The cropped images from LSA64 videos

3.2 3D Skeleton Video Creation with MediaPipe

MediaPipe holistic is a model that consists of hand landmarks, pose landmarks, and face landmarks that give estimations of posture, hand movements, and facial expressions (Naz et al., 2023). Figure 3.2.1 shows an overview of MediaPipe holistic, Figure 3.2.2 shows the 21 crucial points plotted with MediaPipe for every hand, Figure 3.2.3 shows the 33 crucial points plotted with MediaPipe for pose estimation, and Figure 3.2.4 shows the 468 crucial points plotted with MediaPipe for facial regions.

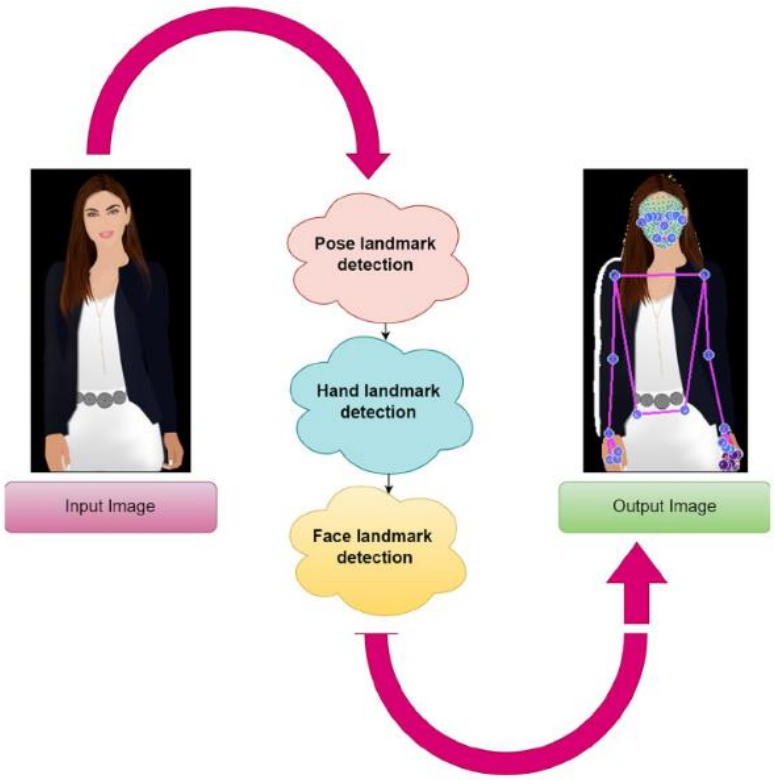


Figure 3.2.1: Overview of MediaPipe Holistic (Subramanian et al., 2022).

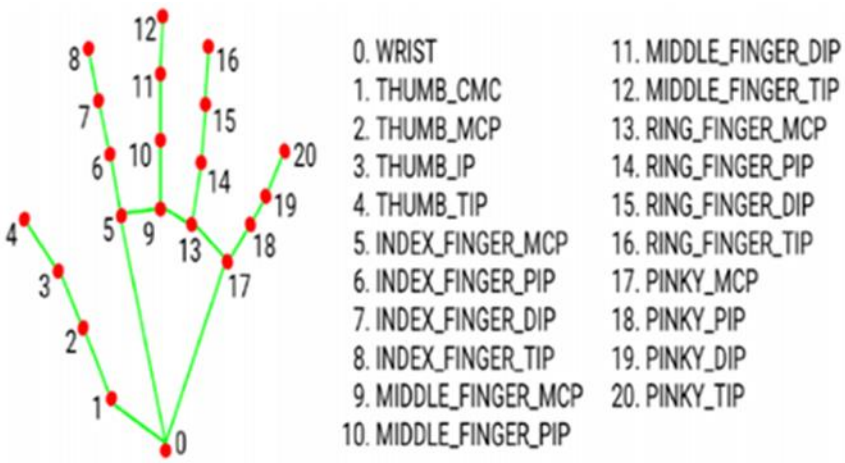


Figure 3.2.2： MediaPipe hands key points label (Zhang et al., 2020)

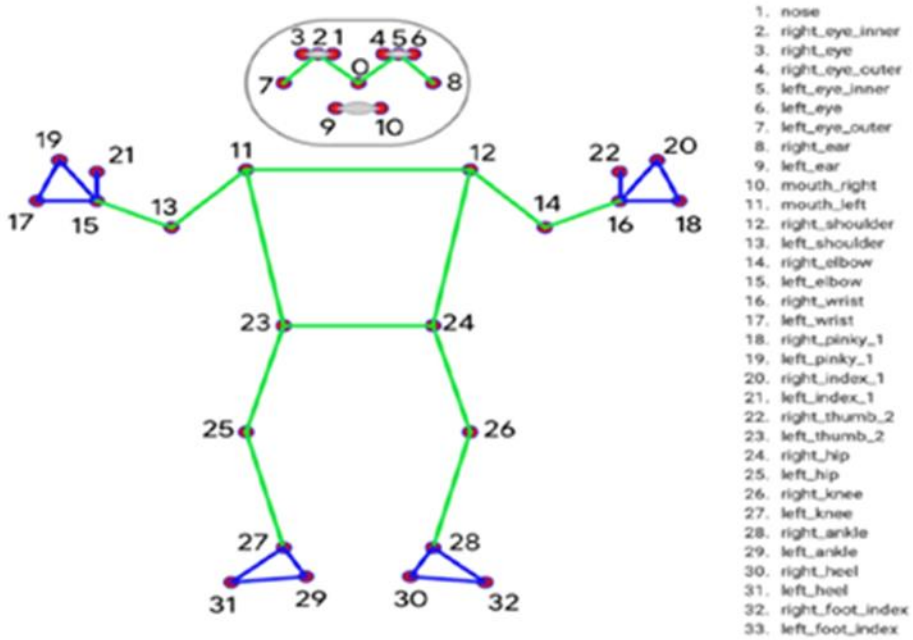


Figure 3.2.3: MediaPipe poses key points label (Bazarevsky et al., 2020)

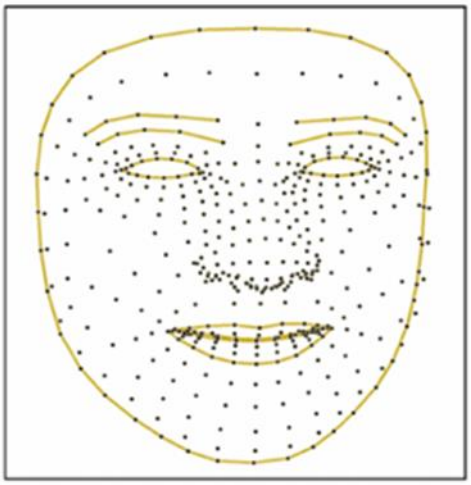


Figure 3.2.4: MediaPipe facial key points label (Yury Kartynnik et al., 2019).

MediaPipe can comprehend gestures and tracking actions like hand movements from the hand's key points. To precisely locate and orient important anatomical landmarks on the human body, pose estimation is a necessity. At the same time, facial key points will also be identified and tracked by MediaPipe, allowing for the analysis of facial expressions and the recognition of emotions.

Before the recording of the MSL10 dataset, MediaPipe settings were done to plot the key points of the MSL10 dataset while presenting the 1000 sign languages. MediaPipe can recognize and monitor 21 different points on each hand, 33 critical points for posture estimation, and 468 facial key points (Lugaresi et al., 2019).

One of the goals is to compare the performance of the hand, pose, and face key points versus the hand key points alone. MediaPipe will create the first set of key points for the MSL10 dataset, consisting of a combination of hand, pose, and face key points. Following the creation of the first set of data, hand-key points will be extracted solely from the first set of key points, creating the second set of data for the MSL10 dataset.

The calculation of the total number of key points per frame for all features: (*Hand key points × Three dimensions × No. of hands) + (pose key points × Three dimensions + Visibility) + (Face key points × Three dimensions)*

= (21×3×2) + (33 × (3+1)) + (468 ×3)

= 126 + 132 + 1404

= 1662 key points.

The calculation of the total number of key points per frame for hands features:

(Hand key points, three dimensions, number of hands)

= (21×3×2)

= 126 key points.

OpenCV (Open-Source Computer Vision Library) is useful for a variety of image processing and applications related to computer vision (Mohamad et al., 2015). OpenCV was designed for computational efficiency, with a focus on real-time applications (Kwon and Kim, 2022).

It has been discovered that MediaPipe and OpenCV can work together. According to Almufti and Adnan Mohsin Abdulazeez (2024), OpenCV is used to process live video data for gesture recognition, whereas MediaPipe can be used for precise hand tracking from live video streams, followed by feature extraction and data serialization.

OpenCV is used together with MediaPipe to capture the key points of the 3D skeleton videos, and the MSL10 dataset was captured in an indoor setting with controlled lighting. One second was split into thirty frames for each 3D skeleton movie that was recorded. Every segment of sign language in every video had the same number of frames. For the MSL10 dataset, two sets of data have been formed with the 3D shape of (1000, 30, 1662) and (1000, 30, 126) as shown in Figure 3.2.5.

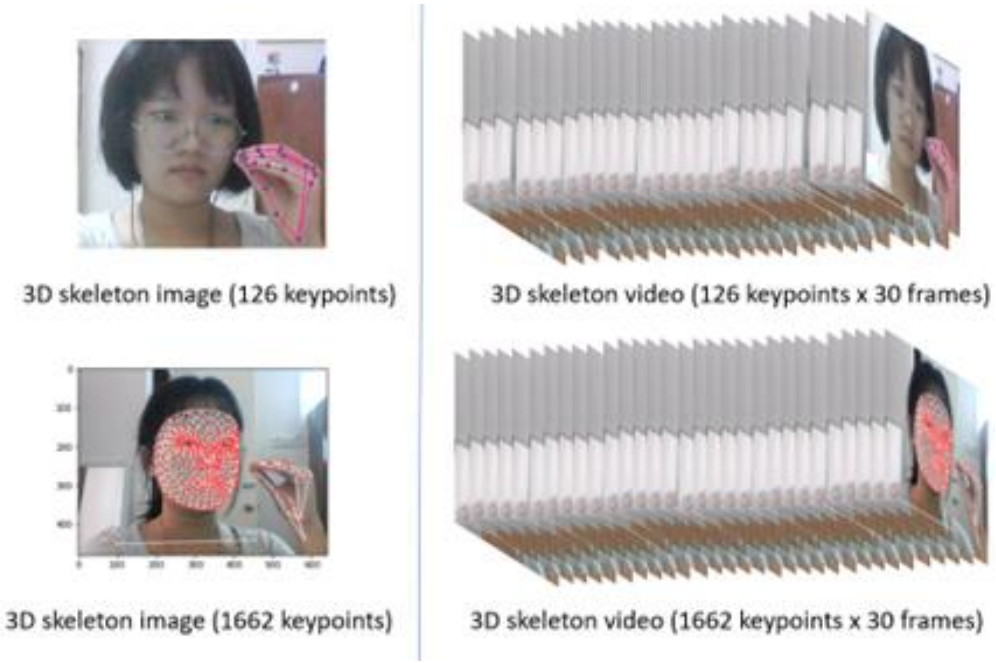


Figure 3.2.5: Shape of MSL10 dataset

Since the LSA64 dataset consists of pre-recorded videos, the method used by MSL10 does not apply to the LSA64 dataset. Therefore, a loop is created to open each video in the LSA64 dataset and then uses MediaPipe to extract 30 frames of key points from each video and arrange them into appropriate shapes. The key points plotted in the LSA64 dataset are complete features that include a combination of hand, pose, and face key points. Same with the MSL10 dataset, another set of only hand key points will be extracted from the LSA64 full-featured data to form two sets of data with the shapes (3200, 30, 1662) and (3200, 30, 126), respectively, as shown in Figure 3.2.6.

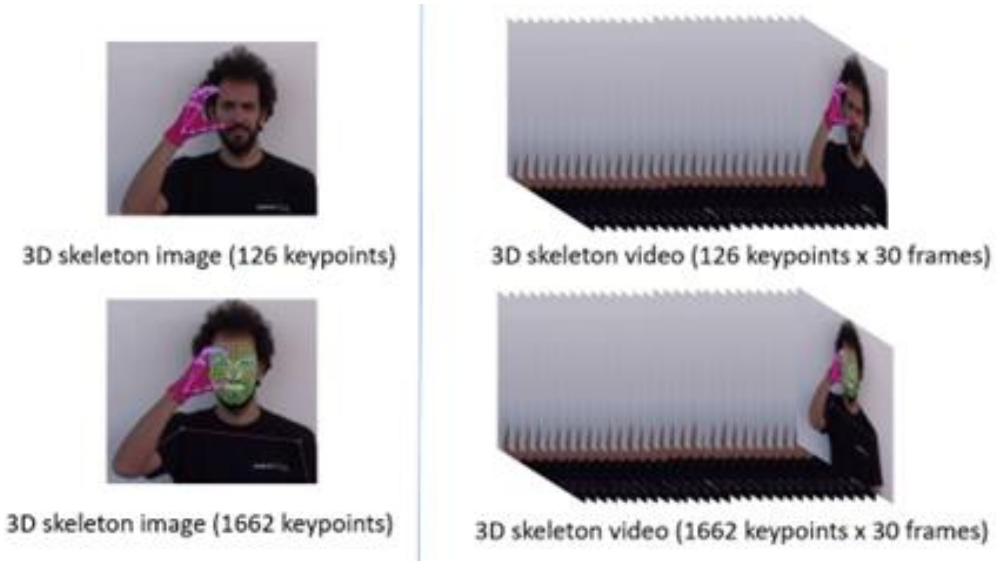


Figure 3.2.6: Shape of LSA64 dataset

3.3 Data Preprocessing

After forming two different sets of data with different characteristics from each of the two datasets, the next step is to categorize all the data using words. In the previous step, the formation of the dataset followed the sequence of each word while collecting the data. For example, if the first word in the MSL10 dataset is "Beli," followed by "Benar," then all the 3D skeleton datasets of "Benar" will be sequenced after the entire "Beli" dataset. This arrangement further simplifies the labeling of the datasets. For supervised training, each piece of data will be labeled with the corresponding sign language word. The MSL10 dataset has ten different labels, whereas the LSA64 dataset has 64.

Data splitting will be carried out for each dataset after labeling. In each dataset, 75% of the data will be split into a training set and 25% of the data will be split into a testing set.

3.4 RNN models

Once the data has been split, the next step is to train the training set with a deep-learning Recurrent Neural Network (RNN) model. RNNs are a form of Artificial Neural Network (ANN) that are created for handling sequential and time-series data. RNNs are well-known for having feedback linkages in the networks (Chung et al., 2015).

RNNs outperform non-recurrent models in continuous gesture recognition because they can learn the hierarchy of actions and foresee the start and end of actions (Pigou et al., 2016). Figure 3.4.1 displays the construction of the normal Recurrent Neural Network (RNN) and the unfolded RNN presented by Feng et al. (2017).

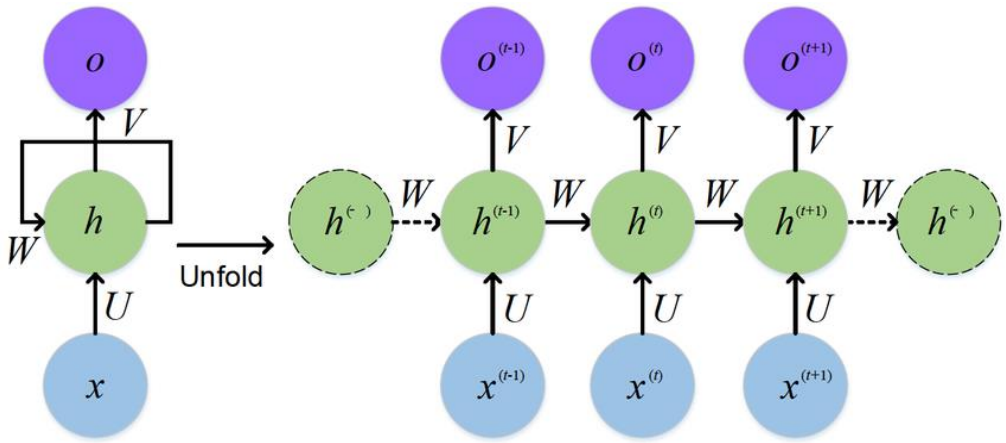


Figure 3.4.1: Standard Recurrent Neural Network (RNN) and unfolded RNN (Feng et al., 2017)

Because RNN has to foresee the next word of a sentence by remembering the previous word, it has to consider the output of the previous step as input to the current step.

TensorFlow was published by Google in November 2015. It acts as an open-source deep learning software library that allows developers to define, train, and deploy machine learning models (Goldsborough, 2016). MediaPipe can also work with TensorFlow. According to Sundar and Bagyammal (2022), MediaPipe collects hand key points while TensorFlow trains and detects the machine learning algorithm. Therefore, TensorFlow is used in this study to build the three RNN models.

In this study, three RNN-related models are used to deal with the sign language recognition task, namely the Long Short-Term Memory (LSTM) model, the 2-layer Bidirectional Long Short-Term Memory (BiLSTM) model, and the 3-layer BiLSTM model. These RNN models usually capture the temporal dependencies in the data. The LSTM model can solve the gradient vanishing problem, which is a possible problem when training traditional RNNs. Figure 3.4.2 shows the structure of the LSTM model.

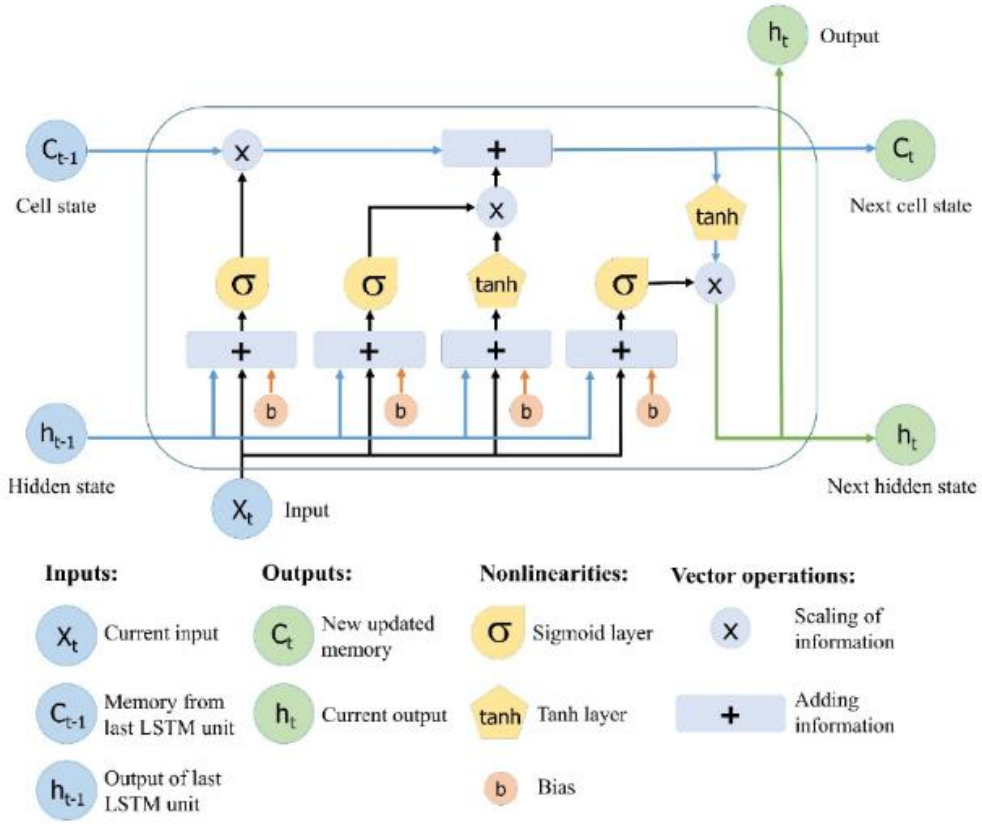


Figure 3.4.2: Long Short-Term Memory (Le et al., 2019)

An LSTM network consists of blocks of storage called cells, and cell states and hidden states are transferred to the next cell. Data can be added or removed from the cell state through sigmoid gates. A gate is analogous to a layer containing different weights.

The first step in practicing LSTM is to decide what information to forget from the previous memory cell. The input to the gate of forgetting (f_t) comes from the output of the previous LSTM cell (h_{t-1}) and the current input (X_t). The

forgetting gate (f_t) decides which parts of the old storage cell (C_{t-1}) should be forgotten and takes a value ranging from 0 to 1.

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f \quad (1)$$

In equation (1), σ represents the sigmoid function, W_f the weight matrix and b_f the bias of the forget gate.

The LSTM stores and refreshes the storage cell with new information as needed. The sigmoid layer determines whether the cell should be updated with the new information. The tanh layer assigns weights to new data to judge its relevance. The updated information is coupled with the old-stored cell (C_{t-1}) to calculate the new cell state (C_t).

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \quad (2)$$

$$N_t = tanh(W_n[h_{t-1}, X_t] + b_n) \quad (3)$$

$$C_t = C_{t-1}f_t + N_t i_t \quad (4)$$

C_t represents the cell state at time t, while C_{t-1} represents the cell state at time $t-1$. W is the weight matrix, while b is the cell state bias.

The LSTM creates an output value (h_t) based on the updated cell state (C_t), but filtered. The final output value (h_t) is calculated by multiplying the output gate (O_t) by the tanh layer value applied to the cell state.

$$O_t = \sigma(W_o[h_{t-1}, X_t] + b_o \quad (5)$$

$$h_t = O_t tanh(C_t) \quad (6)$$

W_o is the weight matrix and b_o is the bias for the output gate.

The BiLSTM model is formed by combining LSTM model and bi-directional RNN model (Ameur, Ben Khalifa, and Salim Bouhlel, 2020). The BiLSTM model is designed to handle both forward and backward information. Figure 3.4.5 demonstrates the structure of the two-layer BiLSTM.

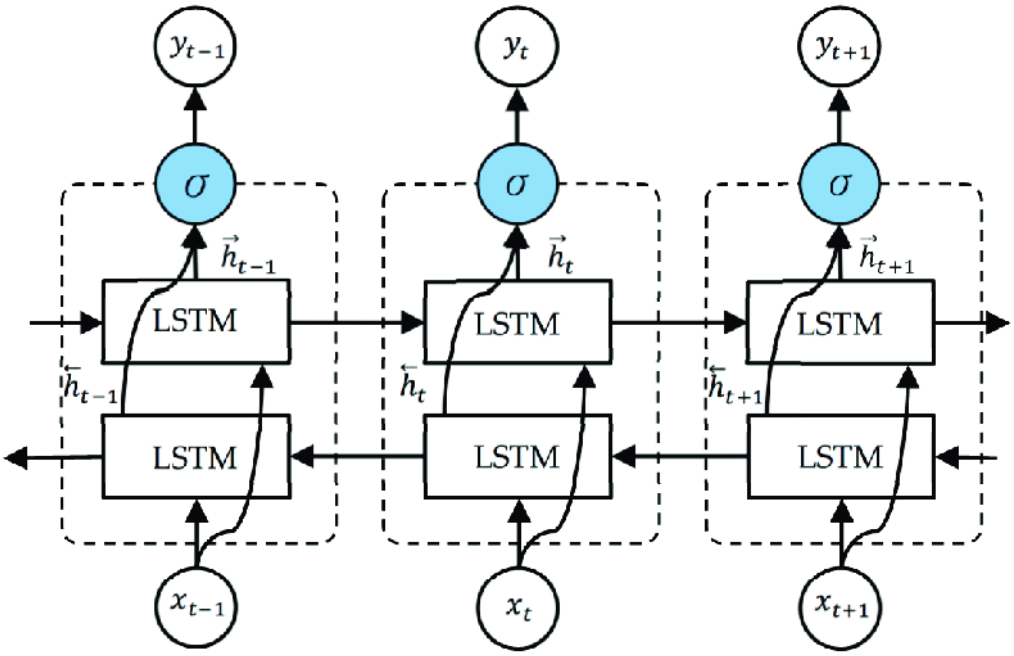


Figure 3.4.5: 2-layer Bidirectional Long Short-Term Memory (Li et al., 2020).

The input sequences of the LSTMs are processed from time steps $t = 1$ to $t = n$. In each time step, the LSTM layer generates a hidden state $\rightarrow h_t$. The backward LSTM layer output sequence ($\leftarrow h_t$) is produced by processing the input sequences in the reverse order. The combination of $\rightarrow h$ and $\leftarrow h$ is realized by a sigmoid function (σ) to form the final output vector y_t. y_t contains information from the forward and reverse contexts as the final output of the BiLSTM layer.

Tables 3.4.1, 3.4.2, and 3.4.3 list the structural models of the three RNN models.

Table 3.4.1： Structure of LSTM (3 layers)

Layer (type)	Output Shape
lstm (LSTM)	(None, 30, 64)
lstm_1 (LSTM)	(None, 30, 128)
lstm_2 (LSTM)	(None, 64)
dense (Dense)	(None, 64)
dense_1 (Dense)	(None, 32)
dense_2 (Dense)	(None, 10)

Table 3.4.2： Structure of BiLSTM (2 layers)

Layer (type)	Output Shape
bidirectional (Bidirectional)	(None, 30, 128)
bidirectional_1 (Bidirectional)	(None, 128)
dense (Dense)	(None, 32)
dense_1 (Dense)	(None, 10)

Table 3.4.3： Structure of BiLSTM (3 layers)

Layer (type)	Output Shape
bidirectional (Bidirectional)	(None, 30, 512)
bidirectional_1 (Bidirectional)	(None, 30, 1024)
bidirectional_2 (Bidirectional)	(None, 512)
dense (Dense)	(None, 64)
dense_1 (Dense)	(None, 32)
dense_2 (Dense)	(None, 10)

Each of the RNN models comprises a consistent number of dense layers in its structure. "Relu" is selected as the activation of the model in a neural network's hidden layers to introduce nonlinearity. Rectified Linear Unit (ReLU) is a non-linear activation function used in deep neural networks (Dureja and Pahwa, 2018). Nonlinearity is introduced through ReLU in the neural network's hidden layers (SCI, 2020).

To estimate the class of an input image, the model's activation in the final layer of a neural network is set to "Softmax". Softmax returns the output in probabilities (Rahman and Aris Rakhmadi, 2023) and it is frequently used in the final layer to determine the output class of the data (Agarap, 2019).

The Adaptive Moment Estimation can minimize the loss function during the model's training (Dhandapani, Vikranth Lokeshwar, and Jain, 2024). Adaptive Moment Estimation which is also known as Adam is chosen as the optimizer in this study.

3.5 Model Evaluation Metrics

The model's performance is evaluated with computation time and accuracy. Accuracy can be defined in terms of the number of correct classifications over the total number of classifications attempted while the computation time is the time required to complete the RNN model training and test set fitting process.

The formula for defining accuracy and computation time is given below:

$$Accuracy = \frac{number\ of\ correct\ classifications}{total\ number\ of\ classifications\ attempted} \quad (7)$$

$$Computational\ time\ = \text{RNN model fitting end time -RNN model fitting start time} \quad (8)$$

The performance and efficiency of deep learning models can be observed by evaluating their accuracy and computation time. The desired goal of this study is to obtain high accuracy with minimum computation time.

3.6 Flowchart of the proposed method

To give a thorough overview of the suggested technique and all its steps, a full flowchart has been constructed and is presented in Figure 3.6.1.

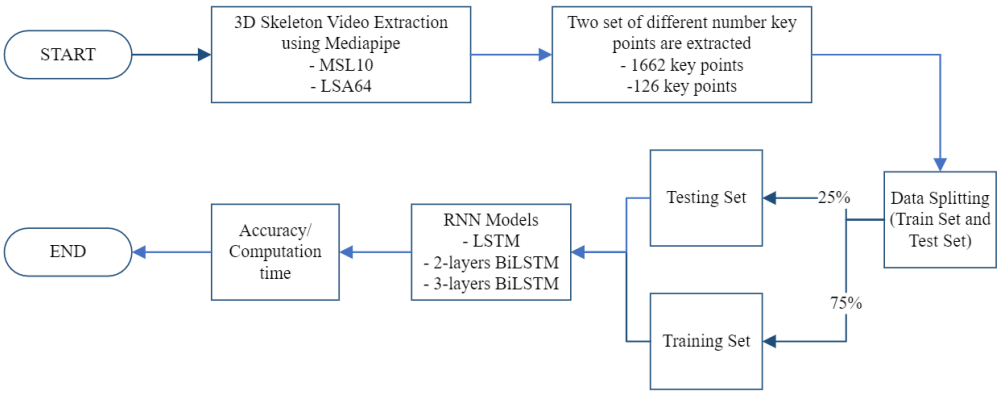


Figure 3.6.1: Flowchart of the proposed method

The first stage of the flowchart is 3D skeleton video formation using MediaPipe for the MSL10 dataset and LSA64 dataset. The potting of key points follows the holistic mode in MediaPipe, which contains face, pose, and hand key points (1662 key points). The second stage is hand-key points (126 key points) that will be formed from the holistic model as the second set of data. The third stage is the data splitting of the dataset. Each of the datasets is split into training and testing sets. The training set comprises 75% of the data, which is used to train the models, while the remaining 25% forms the testing set, which is utilized to evaluate model performance. In the fourth stage, the training data will be fitted into the RNN models (LSTM, 2-layer BiLSTM, and 3-layer BiLSTM). These

models are trained to learn patterns and relationships within the data. Lastly, after training the RNN models, their performance is evaluated in terms of accuracy and computational time.

This flowchart shows the complete process, from data formation to model evaluation, giving an easy understanding of the stages involved in the proposed technique.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results

Table 4.1.1 shows the accuracy and computation time performance of the 3D skeleton videos on the RNN models for the MSL10 and LSA64 datasets.

Table 4.1.1: Overall Accuracy and Computation Time Performance of the 3D Skeleton Video on RNN Models of the MSL10 dataset and LSA64 dataset

		MSL10			LSA64		
		LSTM	2-layer BiLSTM	3-layer BiLSTM	LSTM	2-layer BiLSTM	3-layer BiLSTM
1662 key points	Accuracy (%):	88.80	96.40	92.40	0.50	76.375	0.75
	Computation Time (s):	192.66	604.32	3945.66	1264.42	1224.65	11022.28
126 key points	Accuracy (%):	96.40	97.60	98.00	79.875	86.875	81.875
	Computation Time (s):	94.88	79.79	2948.88	317.95	255.23	9629.44

To improve the accuracy of the data, it is recommended to conduct five repeats for each combination to get a total of five sets of data for a single combination. The accuracy and computation time will vary due to the random splitting of the data. The repeats of 12 combinations in 12 tables are shown below.

Table 4.1.2: The Average of Accuracy and Computation Time Performance of the MSL10 dataset with 1662 key points and LSTM model

	MSL10, 1662 key points, LSTM	
No.	Accuracy (%)	Computation Time (s)
1.	88.80	192.66
2.	59.20	217.50
3.	78.00	211.54
4.	82.00	241.18
5.	94.40	253.94
Average:	80.48	223.364

Table 4.1.3: The Average Accuracy and Computation Time Performance of the MSL10 dataset with 1662 key points and 2-layer BiLSTM model

	MSL10, 1662 key points, 2-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	96.4	604.32
2.	97.6	269.33
3.	96.4	310.86
4.	96.8	384.07
5.	96.8	344.84
Average:	96.8	382.684

Table 4.1.4: The Average Accuracy and Computation Time Performance of the MSL10 dataset with 1662 key points and 3-layer BiLSTM model

	MSL10, 1662 key points, 3-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	92.4	3945.66
2.	87.2	3883.93
3.	91.2	3611.82
4.	88.4	4041.65
5.	84	4173.34
Average:	88.64	3931.28

Table 4.1.5: The Average Accuracy and Computation Time Performance of the MSL10 dataset with 126 key points and LSTM model

	MSL10, 126 key points, LSTM	
No.	Accuracy (%)	Computation Time (s)
1.	96.4	94.88
2.	96	104.48
3.	93.6	116.55
4.	96.4	140.32
5.	84.8	132.75
Average:	93.08	117.8

Table 4.1.6: The Average Accuracy and Computation Time Performance of the MSL10 dataset with 126 key points and 2-layer BiLSTM model

	MSL10, 126 key points, 2-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	97.6	79.79
2.	97.6	87.74
3.	97.2	84.91
4.	96.4	79.99
5.	97.2	84.09
Average:	97.2	83.3

Table 4.1.7: The Average Accuracy and Computation Time Performance of the MSL10 dataset with 126 key points and 3-layer BiLSTM model

	MSL10, 126 key points, 3-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	98	2948.88
2.	97.2	2790.65
3.	98	3498.97
4.	94.8	3470.63
5.	98	3611.51
Average:	97.2	3264.128

Table 4.1.8: The Average Accuracy and Computation Time Performance of the LSA64 dataset with 1662 key points and LSTM model

	LSA64, 1662 key points, LSTM	
No.	Accuracy (%)	Computation Time (s)
1.	0.50	1264.41
2.	52.5	667.06
3.	0.50	735.58
4.	0.75	792.79
5.	1.375	725.01
Average:	11.125	836.97

Table 4.1.9: The Average Accuracy and Computation Time Performance of the LSA64 dataset with 1662 key points and 2-layer BiLSTM model

	LSA64, 1662 key points, 2-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	76.375	1224.65
2.	70.625	957.02
3.	78.5	845.11
4.	75.875	1094.49
5.	76.875	1135.60
Average:	75.65	1051.374

Table 4.1.10: The Average Accuracy and Computation Time Performance of the LSA64 dataset with 1662 key points and 3-layer BiLSTM model

	LSA64, 1662 key points, 3-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	0.75	11022.28
2.	0.75	10451.67
3.	0.625	10426.82
4.	0.5	10427.46
5.	0.625	12703.33
Average:	0.65	11006.312

Table 4.1.11: The Average Accuracy and Computation Time Performance of the LSA64 dataset with 126 key points and LSTM model

	LSA64, 126 key points, LSTM	
No.	Accuracy (%)	Computation Time (s)
1.	79.875	317.95
2.	79.125	413.32
3.	0.75	341.44
4.	81.625	278.72
5.	78	353.38
Average:	63.875	340.962

Table 4.1.12: The Average Accuracy and Computation Time Performance of the LSA64 dataset with 126 key points and 2-layer BiLSTM model

	LSA64, 126 key points, 2-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	86.875	255.23
2.	85.625	229.07
3.	87.5	290.62
4.	84.875	286.45
5.	86.125	293.56
Average:	86.2	270.986

Table 4.1.13: The Average Accuracy and Computation Time Performance of the LSA64 dataset with 126 key points and 3-layer BiLSTM model

	LSA64, 126 key points, 3-layer BiLSTM	
No.	Accuracy (%)	Computation Time (s)
1.	81.875	9629.44
2.	81.5	8719.24
3.	83	9591.64
4.	79.625	10647.68
5.	83	9177.39
Average:	81.3	9481.078

The average accuracy and computational time for five iterations of the experiment for every model displayed in Tables 4.1.2 through 4.1.13 were substituted into Table 4.1.14 to create the final table.

Table 4.1.14: Overall Average Accuracy and Computation Time Performance of 3D Skeleton Videos on RNN models with 5 repetitions on MSL10 dataset and LSA64 dataset

		MSL10			LSA64		
		LSTM	2-layer BiLSTM	3-layer BiLSTM	LSTM	2-layer BiLSTM	3-layer BiLSTM
1662 key points	Accuracy (%):	80.48	96.8	88.64	11.125	75.65	0.65
	Computation Time (s):	223.364	382.684	3931.28	836.97	1051.374	11006.312
126 key points	Accuracy (%):	93.08	97.2	97.2	63.875	86.2	81.8
	Computation Time (s):	117.8	83.3	3264.128	340.962	270.986	9481.078

4.2 Discussion

From Table 4.1.14, by comparing the performance of LSTM and BiLSTM models, we can see that both 2-layer and 3-layer bi-directional LSTM (BiLSTM) models outperform the traditional LSTM model in terms of accuracy at different key points and on different datasets. Since BiLSTM can read data in both directions, BiLSTM models are better at understanding sequence relationships than LSTM models. It is worth noting that the accuracy of the LSTM model decreases when the dataset capacity increases from the MSL10 dataset to the LSA64 dataset. For the 1662 key points, the LSTM model tended to overfit when the capacity of the dataset was maximized in this experiment. From Table 4.1.8, out of the five replicated experiments, only one experiment did not show overfitting, and the accuracy remained low. The reason for the overfitting may be that the LSTM model is not capable of handling such a large amount of data. In terms of computation time, while the LSTM model performs better at 1662 key points, the 2-tier BiLSTM model performs better at 126 key points. In this case, since 126 key points result in higher accuracy than 1662 key points, it can still be concluded that the performance of the 2-layer BiLSTM outperforms the LSTM model in terms of computation time. The reason for this situation could be that BiLSTM processes input sequences in both directions and therefore converges faster during training.

From the results of Table 4.1.14, by comparing the performance of the 2-layer BiLSTM model and the 3-layer BiLSTM model, the 2-layer BiLSTM model has higher accuracy in the MSL10 and LSA64 datasets. This is because the 3-layer BiLSTM model suffers from an overfitting problem. The overfitting problem is

when the model captures a lot of noise or irrelevant information from the training data. The complex model formed from the training data cannot detect the test data well, resulting in low accuracy and long computation time. The computation time of the 2-layer BiLSTM model is also shorter compared to the 3-layer BiLSTM. This is because the 2-layer BiLSTM is simpler as compared to 3-layer BiLSTM, so it takes less time to train the model. In addition to this, the computation time increases when the model is overfitted.

Throughout the performance from Table 4.1.14, by comparing the 1662 key points to the 126 key points, accuracy improves as the number of key points decreases. This is because the reduced key points are not important or contain noise. Reducing the number of key points improves accuracy by eliminating sources that confuse the model. Reducing the number of key points from 1,662 to 126 will also reduce the computation time of the model as the sample size decreases. With fewer key points, the model may become simpler and less likely to overfit. Simpler datasets may be easier for the model to learn the relationships and patterns in the data, thus reducing computation time.

CHAPTER 5

CONCLUSION

5.1 Summary of Research

This research tackles the issue of the model's lengthy processing time while utilizing picture or video input, which arises from the deep learning model's excessive processing time. The MSL10 dataset was created due to the lack of open-source Malaysian sign language videos. To reduce the video dataset to a video featuring 3D skeleton key points and excluding background information, MediaPipe is used. The 3D skeletal video extracted with MediaPipe can precisely represent the semantic meaning of sentences and extract significant information, such as the key points of hands, poses, and faces. With the use of MediaPipe, the time used to convert the video can be reduced significantly.

Throughout the studies, the BiLSTM model outperformed the LSTM model in terms of accuracy and computational time. The 2-layer BiLSTM model also consistently outperforms the 3-layer BiLSTM model in terms of accuracy and computation time. When all feature key points were compared to only hand key points, the accuracy increased while the computation time decreased.

In conclusion, a 2-layer BiLSTM model is recommended together with the MediaPipe model, which contains only the hand key points. This study demonstrates the correlation between the number of layers in the BiLSTM model and the performance of the model in terms of computation time and accuracy. This study also shows that hand-key points alone are sufficient for deep-learning models to recognize sign language.

Using OpenCV and Python, a real-time recognition model has been developed. The MediaPipe library is used by the real-time recognition algorithm to identify hand landmarks on the webcam. The system constantly records webcam frames and uses MediaPipe's holistic model to process them to identify landmarks. After landmarks have been identified, key points are extracted and fed into a machine learning model that has already been trained for prediction. Predicting the sign being performed involves maintaining track of a series of important points over time and applying this sequence to prediction logic. The algorithm updates a sentence variable with the detected sign if the predicted sign exceeds a predetermined threshold, hence guaranteeing seamless transitions between signs in the recognized phrase. Lastly, the algorithm uses OpenCV to display the sentence and symbol that have been identified on the screen. With this configuration, real-time sign language recognition can be done straight from a camera stream, which opens a wide range of useful applications.

In conclusion, the 2-layer BiLSTM model, hand key point extraction, and 3D skeleton videos may all be used to achieve the fastest computation time without sacrificing the model's accuracy. The final approach combines the 2-layer BiLSTM deep learning model with MediaPipe 126 hands key points. Using the resulting technique, a real-time model for the MSL10 dataset has been developed.

5.2 Limitations and Recommendations

Since Malaysian Sign Language (MSL) is not an open-source sign language, it is a difficult task to gather enough high-quality data on MSL, thus limiting the dataset's representativeness and variety. It will have an impact on how broadly the findings can be applied. Sign languages can differ between communities and between regions, and peculiarities of culture can influence how signs are understood and recognized.

The technique of selecting important point features will be the focus of future studies in this study. The process of Recursive Feature Elimination (RFE) includes prioritizing each feature and determining the most crucial elements (Arif Mudi Priyatno and Triyanna Widiyaningtyas, 2024). Using RFE, key points can be ranked according to their importance in sign language recognition. RFE iteratively removes the least important features until the desired number of features is reached. To increase accuracy, researchers might conduct experiments to find the ideal number of important points to choose from. There will be a continuous process involved in choosing features based on high-level features.

REFERENCES

A. Garcia-Garcia, Gomez-Donoso, F., Garcia-Rodriguez, J., Orts-Escolano, S., Cazorla, M. and Azorin-Lopez, J. (2016). PointNet: A 3D Convolutional Neural Network for real-time object class recognition. doi: https://doi.org/10.1109/ijcnn.2016.7727386.

Abdalla, M.S. and Hemayed, E.E. (2013). Dynamic Hand Gesture Recognition of Arabic Sign Language using Hand Motion Trajectory Features. Global journal of computer science and technology, 13(5).

Abduljabbar, R.L., Dia, H. and Tsai, P.-W. (2021). Development and evaluation of bidirectional LSTM freeway traffic forecasting models using simulation data. Scientific Reports, 11(1). doi: https://doi.org/10.1038/s41598-021-03282-z.

Abraham, E., Nayak, A. and Iqbal, A. (2019). Real-Time Translation of Indian Sign Language using LSTM. 2019 Global Conference for Advancement in Technology (GCAT). doi: https://doi.org/10.1109/gcat47503.2019.8978343.

Agarap, A.F. (2019). An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification. arXiv:1712.03541 [cs, stat]. [online] Available at: https://arxiv.org/abs/1712.03541.

Al Amin Hosain, Panneer Selvam Santhalingam, Pathak, P.H., Kosecka, J. and Rangwala, H. (2019). Sign Language Recognition Analysis using Multimodal Data. doi: https://doi.org/10.1109/dsaa.2019.00035.

Almufti, S.M. and Adnan Mohsin Abdulazeez (2024). An Integrated Gesture Framework of Smart Entry Based on Arduino and Random Forest Classifier. *Indonesian Journal of Computer Science*, 13(1). doi: https://doi.org/10.33022/ijcs.v13i1.3735.

Aloysius, N. and Geetha, M. (2020). Understanding vision-based continuous sign language recognition. Multimedia Tools and Applications, 79(31-32), pp.22177–22209. doi: https://doi.org/10.1007/s11042-020-08961-z.

Alsharif, B., Altaher, A.S., Altaher, A., Ilyas, M. and Alalwany, E. (2023). Deep Learning Technology to Recognize American Sign Language Alphabet. Sensors, [online] 23(18), p.7970. doi: https://doi.org/10.3390/s23187970.

Ameur, S., Ben Khalifa, A. and Salim bouhlel, M. (2020). A novel Hybrid Bidirectional Unidirectional LSTM Network for Dynamic Hand Gesture Recognition with Leap Motion. Entertainment Computing, p.100373. doi: https://doi.org/10.1016/j.entcom.2020.100373.

Amit Moryossef, Ioannis Tsochantaridis, Dinn, J., Necati Cihan Camgoz, Bowden, R., Jiang, T., Rios, A., Muller, M. and Ebling, S. (2021). Evaluating the Immediate Applicability of Pose Estimation for Sign Language Recognition. arXiv (Cornell University). doi: https://doi.org/10.1109/cvprw53098.2021.00382.

Arif Mudi Priyatno and Triyanna Widiyaningtyas (2024). A SYSTEMATIC LITERATURE REVIEW: RECURSIVE FEATURE ELIMINATION

ALGORITHMS. *JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer)*, 9(2), pp.196–207. doi: https://doi.org/10.33480/jitk.v9i2.5015.

Avraam Tsantekidis, Nikolaos Passalis and Anastasios Tefas (2022). Recurrent neural networks. Elsevier eBooks, pp.101–115. doi: https://doi.org/10.1016/b978-0-32-385787-1.00010-5.

Bazarevsky, V., Grishchenko, I., Karthik Raveendran, Zhu, T., Zhang, F. and Grundmann, M. (2020). BlazePose: On-device Real-time Body Pose tracking. arXiv (Cornell University). doi: https://doi.org/10.48550/arxiv.2006.10204.

Bora, J., Saine Dehingia, Abhijit Boruah, Anuraag Anuj Chetia and Dikhit Gogoi (2023). Real-time Assamese Sign Language Recognition using MediaPipe and Deep Learning. 218, pp.1384–1393. doi: https://doi.org/10.1016/j.procs.2023.01.117.

Chen, F.-S., Fu, C.-M. and Huang, C.-L. (2003). Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image and Vision Computing*, 21(8), pp.745–758. doi: https://doi.org/10.1016/s0262-8856(03)00070-2.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2015). Gated Feedback Recurrent Neural Networks. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1502.02367.

Cireşan, D., Meier, U., Masci, J. and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. Neural Networks, 32, pp.333–338. doi: https://doi.org/10.1016/j.neunet.2012.02.023.

Dhandapani, V.L. and Jain, S., n.d. Neural Networks for Portfolio-Level Risk Management: Portfolio Compression, Static Hedging, Counterparty Credit Risk Exposures and Impact on Capital Requirement. *arxiv.org*. [online] Available at: https://arxiv.org/html/2402.17941v1 [Accessed 18 May 2024].

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio and Vincent, P. (2009). The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. Pp.153–160.

Dureja, A. and Pahwa, P. (2018). Analysis of Non-linear Activation Functions for Classification Tasks using Convolutional Neural Networks. Recent Patents on Computer Science, 11. doi: https://doi.org/10.2174/2213275911666181025143029.

Eshraghian, J.K., Ward, M., Emre Neftci, Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Doo Seok Jeong and Lü, W. (2023). Training Spiking Neural Networks Using Lessons From Deep Learning. Proceedings of the IEEE, 111(9), pp.1016–1054. doi: https://doi.org/10.1109/jproc.2023.3308088.

Feng, W., Guan, N., Li, Y., Zhang, X. and Luo, Z. (2017). Audio visual speech recognition with multimodal recurrent neural networks. doi: https://doi.org/10.1109/ijcnn.2017.7965918.

Ge, L., Cai, Y.-J., Weng, J. and Yuan, J. (2018). Hand PointNet: 3D Hand Pose Estimation Using Point Sets. doi: https://doi.org/10.1109/cvpr.2018.00878.

Georgevici, A.I. and Terblanche, M. (2019). Neural networks and deep learning: a brief introduction. *Intensive Care Medicine*. doi: https://doi.org/10.1007/s00134-019-05537-w.

Gers, F.A. and Schmidhuber, J. (2000). Recurrent nets that time and count. [online] IEEE Xplore. doi: https://doi.org/10.1109/IJCNN.2000.861302.

Goldsborough, P. (2016). A Tour of TensorFlow. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1610.01178.

Hafit, H., Xiang, C.W., Yusof, M.M., Wahid, N. and Kassim, S. (2019). Malaysian Sign Language Mobile Learning Application: A recommendation app to communicate with hearing-impaired communities. International Journal of Electrical and Computer Engineering (IJECE), 9(6), p.5512. doi: https://doi.org/10.11591/ijece.v9i6.pp5512-5518.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780.

Huang, J., Zhou, W., Zhang, Q., Li, H. and Li, W. (2018). Video-Based Sign Language Recognition Without Temporal Segmentation. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). doi: https://doi.org/10.1609/aaai.v32i1.11903.

Indriani, Harris, Moh. and Agoes, A.S. (2021). Applying Hand Gesture Recognition for User Guide Application Using MediaPipe. *Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021)*. doi: https://doi.org/10.2991/aer.k.211106.017.

Jella Sandhya and KANCHARLA ANITHASHEELA (2024). Spatiotemporal Modeling for Dynamic Gesture Recognition in Video Streams. Research Square (Research Square). doi: https://doi.org/10.21203/rs.3.rs-4019650/v1.

Kavana, K.M. and Suma, N.R., 2022. Recognization of hand gestures using mediapipe hands. *International Research Journal of Modernization in Engineering Technology and Science*, *4*(06).

Kim, T.-H., Keane, J., Wang, W., Hall, M.B., Riggle, J., Shakhnarovich, G., Brentari, D. and Livescu, K. (2017). Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation. Computer Speech & Language, 46, pp.209–232. doi: https://doi.org/10.1016/j.csl.2017.05.009.

Kwon, Y. and Kim, D. (2022). Real-Time Workout Posture Correction using OpenCV and MediaPipe. *The Journal of Korean Institute of Information Technology*, 20(1), pp.199–208. doi: https://doi.org/10.14801/jkiit.2022.20.1.199.

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278–2324. doi: https://doi.org/10.1109/5.726791.

Li, H., Gao, L., Han, R., Wan, L. and Feng, W. (2020). Key Action and Joint CTC-Attention based Sign Language Recognition. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). doi: https://doi.org/10.1109/icassp40776.2020.9054316.

Liao, Y., Xiong, P., Min, W., Min, W. and Lu, J. (2019). Dynamic Sign Language Recognition Based on Video Sequence With BLSTM-3D Residual Networks. IEEE Access, 7, pp.38044–38054. doi: https://doi.org/10.1109/access.2019.2904749.

Lin, Y., Jiao, X. and Zhao, L. (2023). Detection of 3D Human Posture Based on Improved Mediapipe. *Journal of Computer and Communications*, [online] 11(2), pp.102–121. doi: https://doi.org/10.4236/jcc.2023.112008.

Ling, C., Zhong, J. and Li, W. (2022). Predictive Coding Based Multiscale Network with Encoder-Decoder LSTM for Video Prediction. arXiv (Cornell University). doi: https://doi.org/10.48550/arxiv.2212.11642.

Lipton, Z.C., Berkowitz, J. and Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. [online] arXiv.org. Available at: https://arxiv.org/abs/1506.00019.

Liu, X., Deng, Z. and Yang, Y. (2018). Recent progress in semantic image segmentation. Artificial Intelligence Review, [online] 52(2), pp.1089–1106. doi: https://doi.org/10.1007/s10462-018-9641-3.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M.G., Lee, J., Chang, W.-T., Hua, W., Georg, M. and Grundmann, M. (2019). MediaPipe: A Framework for Building Perception Pipelines. doi: https://doi.org/10.48550/arxiv.1906.08172.

Marais, M., Brown, D., Connan, J. and Boby, A. (2022). An Evaluation of Hand-Based Algorithms for Sign Language Recognition. [online] IEEE Xplore. doi: https://doi.org/10.1109/icABCD54961.2022.9856310.

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, [online] 5(4), pp.115–133. doi: https://doi.org/10.1007/bf02478259.

Mohamad, M., Saman, M.Y.M., Hitam, M.S. and Telipot, M., 2015. A Review on OpenCV. *Terengganu: Universitas Malaysia Terengganu*, *3*, p.1.

Muhammad Mizanur Rahaman, Mahin, M., Ali, H. and Md. Hasanuzzaman (2019). BHCDR: Real-Time Bangla Handwritten Characters and Digits Recognition using Adopted Convolutional Neural Network. doi: https://doi.org/10.1109/icasert.2019.8934476.

Namburi, A., & Hengsanankun, T. (2022). Combining SVM and human-pose for a vision-based fall detection. *ICIC Express Letters, Part B: Applications*, 13(11), 1177-1187.

Naz, N., Sajid, H., Ali, S., Hasan, O. and Ehsan, M.K. (2023). Signgraph: An Efficient and Accurate Pose-Based Graph Convolution Approach Toward Sign Language Recognition. *IEEE Access*, 11, pp.19135–19147. doi: https://doi.org/10.1109/access.2023.3247761.

Necati Cihan Camgoz, Koller, O., Hadfield, S. and Bowden, R. (2020). Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation. doi: https://doi.org/10.1109/cvpr42600.2020.01004.

Nguyen, H.-T. (2014). *Contributions to facial feature extraction for face recognition*. [online] hal.science. Available at: https://hal.science/tel-01138363 [Accessed 29 Mar. 2024].

Peng, T., Zhang, C., Zhou, J. and Nazir, M.S. (2021). An integrated framework of Bi-directional long-short term memory (BiLSTM) based on sine cosine algorithm for hourly solar radiation forecasting. Energy, 221, p.119887. doi: https://doi.org/10.1016/j.energy.2021.119887.

Pigou, L., Oord, A. van den, Dieleman, S., Van Herreweghe, M. and Dambre, J. (2016). Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1506.01911.

Qodri, A., Rini Akmeliawati and Mohammed, S. (2012). Malaysian Sign Language database for research. doi: https://doi.org/10.1109/iccce.2012.6271327.

Rahman, M.M., Manik, Md.M.H., Islam, Md.M., Mahmud, S. and Kim, J.-H. (2020). An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network. [online] IEEE Xplore. doi: https://doi.org/10.1109/IEMTRONICS51293.2020.9216386.

Rahman, R. and Aris Rakhmadi, S.T. (2023). *Implementation Of Hand Recognition Using Convolutional Neural Network For Sign Language*. [online] eprints.ums.ac.id. Available at: http://eprints.ums.ac.id/id/eprint/110235 [Accessed 29 Mar. 2024].

Safwan Mahmood Al-Selwi, Mohd Fadzil Hassan, Said Jadid Abdulkadir and Amgad Muneer (2023). LSTM Inefficiency in Long-Term Dependencies Regression Problems. Journal of Advanced Research in Applied Sciences and Engineering Technology, 30(3), pp.16–31. doi: https://doi.org/10.37934/araset.30.3.1631.

Samaan, G.H., Wadie, A.R., Attia, A.K., Asaad, A.M., Kamel, A.E., Slim, S.O., Abdallah, M.S. and Cho, Y.-I. (2022). MediaPipe's Landmarks with RNN for Dynamic Sign Language Recognition. *Electronics*, 11(19), p.3228. doi: https://doi.org/10.3390/electronics11193228.

SCI, J.H. (2020). Relu Deep Neural Networks and Linear Finite Elements. *Journal of Computational Mathematics*, 38(3), pp.502–527. doi: https://doi.org/10.4208/jcm.1901-m2018-0160.

Selvaraj, P., Gokul NC, Kumar, P. and Khapra, M.M. (2021). OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages. *arXiv (Cornell University)*. doi: https://doi.org/10.48550/arxiv.2110.05877.

Sharma, S., Gupta, R. and Kumar, A. (2021). Continuous sign language recognition using isolated signs data and deep transfer learning. Journal of Ambient Intelligence and Humanized Computing. doi: https://doi.org/10.1007/s12652-021-03418-z.

SIL International. (2022). *Malaysian Sign Language: A phonological statement*. [online] Available at: https://www.sil.org/resources/archives/57953 [Accessed 30 Mar. 2024].

Subramanian, B., Olimov, B., Naik, S.M., Kim, S., Park, K.-H. and Kim, J. (2022). An integrated mediapipe-optimized GRU model for Indian sign language recognition. *Scientific Reports*, [online] 12(1), p.11964. doi: https://doi.org/10.1038/s41598-022-15998-7.

Sundar, B. and Bagyammal, T. (2022). American Sign Language Recognition for Alphabets Using MediaPipe and LSTM. *Procedia Computer Science*, 215, pp.642–651. doi: https://doi.org/10.1016/j.procs.2022.12.066.

Telmo Adão, Oliveira, J., Somayeh Shahrabadi, Jesus, H., Fernandes, M., Costa, Ferreira, V., Martinho Fradeira Gonçalves, Guevara, M.A., Peres, E. and Luís Gonzaga Magalhães (2023). Empowering Deaf-Hearing Communication: Exploring Synergies between Predictive and Generative AI-Based Strategies towards (Portuguese) Sign Language Interpretation. Journal of Imaging, 9(11), pp.235–235. doi: https://doi.org/10.3390/jimaging9110235.

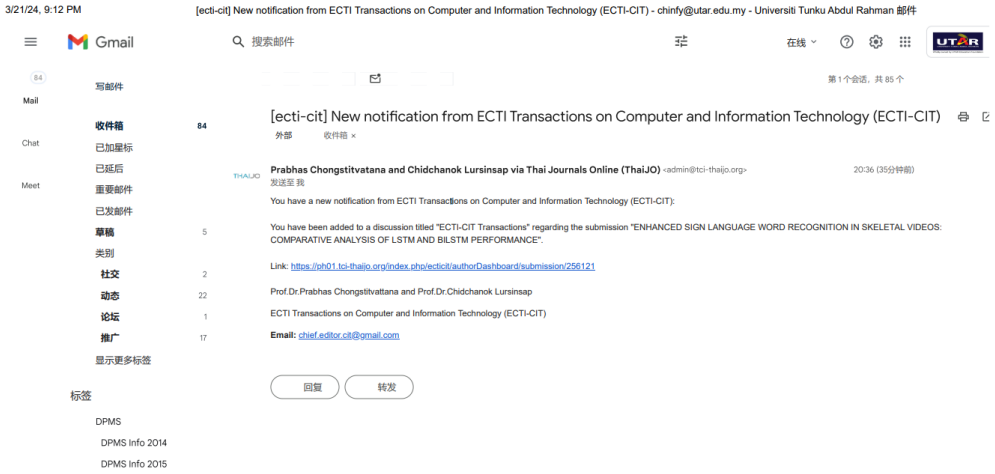
Tsakanikas, V. and Dagiuklas, T. (2018). Video surveillance systems-current status and future trends. *Computers & Electrical Engineering*, [online] 70, pp.736–753. doi: https://doi.org/10.1016/j.compeleceng.2017.11.011.

Yury Kartynnik, Artsiom Ablavatski, Grishchenko, I. and Grundmann, M. (2019). Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs. arXiv (Cornell University). doi: https://doi.org/10.48550/arxiv.1907.06724.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L. and Grundmann, M. (2020). MediaPipe Hands: On-device Real-time Hand Tracking. arXiv:2006.10214 [cs]. [online] Available at: https://arxiv.org/abs/2006.10214.

APPENDIX A

Gmail Notification of Paper Under-Review.



APPENDIX A

First Page of Journal Publication.

ENHANCED SIGN LANGUAGE WORD RECOGNITION IN SKELETAL VIDEOS: COMPARATIVE ANALYSIS OF LSTM AND BILSTM PERFORMANCE

ZI YING ANG*, FUNG YUEN CHIN and FOO WENG LIM

Department of Physical and Mathematical Science, Universiti Tunku Abdul Rahman, Jalan Universiti, Bandar Barat, 31900 Kampar, Perak, Malaysia

Abstract: Sign language recognition is widely recognised as the most important research to help eliminate communication barriers between deaf and ordinary people. With the development of technology, sign language recognition has received a lot of attention from researchers in the past two decades. However, video recognition of sign language is still too time-consuming for deep learning models. This paper aims to validate the performance of two deep learning models, Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM), in two national languages, Malaysian Sign Language (MSL) and Argentinean Sign Language (LSA64 dataset), by using a three-dimensional skeletal video feature extraction method, which can save time significantly. Bidirectional LSTM (BiLSTM) modelling further investigates the different performances of the 2-layer BiLSTM model and the 3-layer BiLSTM model. The MSL10 dataset contains 10 words in Malaysian Sign Language (MSL) repeated 100 times by a signer, while the LSA64 dataset is the Argentine Sign Language (ASL) dataset. Two distinct categories of 3D skeleton videos are obtained: one that captures facial, postural, and manual data, and another that only captures manual data. Twelve combinations were performed using the MSL10 dataset and the LSA64 dataset. Two types of features were collected from the 3D skeletal videos and trained using the LSTM model: the 2-layer BiLSTM model and the 3-layer BiLSTM model. The objective of this study is to identify the most efficient model while maintaining the model's accuracy. The experiments demonstrated that the most efficient model is the one that utilises 3D skeletal video and extracts only the hand key points. This model, trained with a 2-layer BiLSTM architecture, obtains an accuracy of 97.6% with a computation time of 79.79 seconds on the MSL10 dataset. On the LSA64 dataset, it achieves an accuracy of 86.875% with a computation time of 1224.65 seconds.

Keywords: sign language recognition; LSTM; BiLSTM; deep learning; 3D skeleton video; feature extraction; video processing

1. Introduction

Through sign language, the deaf-mute can express themselves using gestures, body movements, facial expressions, and emotions. There are still barriers to communication between the deaf-mute and the general public because the general public does not understand sign language. Therefore, a system that makes it easier for ordinary people and deaf people to communicate is necessary. The system requires a fast and highly accurate gesture recognition method that can detect gestures in a sign language video stream in real-time. Tremendous developments in computer vision technology over the past two decades have made advances in human pose estimation, leading to improvements in vision-based sign language and gesture recognition.

Sign language has been divided into two models for recognition: the sign language recognition isolated model (ISLR) and the continuous sign language continuous model (CSLR) [1]. The differences between ISLR and CSLR are that ISLR is a single image representing a particular shape and posture of the hand, while CSLR is a moving gesture represented by a series of images [2]. Our strategy is built around CSLR, which is also known as dynamic gesture recognition [3]. There are various methods to solve the problems in CSLR recognition. Most of the methods can be categorized into two groups [4]. First, algorithms for computing the motion trajectories of gestures and hand shapes. Secondly, methods for learning sequences of images of each sign language.

Most models that use image data or video data, on the other hand, take longer to process. The processing time can be cut down by using a 3D skeleton video made up of key points taken from each frame of the video [5]. The features extracted from the video may also become negative features. If unwanted features are extracted during training, the computation time increases and the accuracy decreases; however, if too few features are extracted, the accuracy will be low even though the computation is low [6]. The features extracted from

APPENDIX B

Turnitin Originality Report

Appendix D

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1

FACULTY OF SCIENCE

Full Name(s) of Candidate(s)	Ang Zi Ying
ID Number(s)	19ADB01751
Programme / Course	SC
Title of Final Year Project	Leveraging 3d Skeleton Video Extraction and Deep Learning For Real-Time Sign Language Recognition Model

Similarity	**Supervisor's Comments** **(Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 10% **Similarity by source** Internet Sources: _5_____ % Publications: _8_____ % Student Papers: _0_____ %	**The similarity is within the range.**
Number of individual sources listed of more than 3% similarity: 0	**No individual source more than 3% similarity.**
Parameters of originality required and limits approved by UTAR are as follows: **(i) Overall similarity index is 20% and below , and** **(ii) Matching of individual sources listed must be less than 3% each , and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

_____Emmy_____
Signature of Supervisor
Name: _Chin FungYuen_____
Date: 14 Apr 2024_____

_____
Signature of Co-Supervisor
Name: _____
Date: _____

Ang Zi Ying FYP Chapter 1 to 5.docx

ORIGINALITY REPORT

10%	5%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	1%
2	Le, Ho, Lee, Jung. "Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting", Water, 2019 Publication	<1%
3	Rashmi Gaikwad, Lalita Admuthe. "Chapter 23 Real-Time Sign Language Recognition of Words and Sentence Generation using MediaPipe and LSTM", Springer Science and Business Media LLC, 2024 Publication	<1%
4	Xuemei Hu, Junwen Yang. "G-LASSO/G-SCAD/G-MCP penalized trinomial logit dynamic models predict up trends, sideways trends and down trends for stock returns", Expert Systems with Applications, 2024 Publication	<1%
5	Kavishaalini Padmanand, Phei-Chin Lim. "Malaysian Sign Language Recognition Using 3D Hand Pose Estimation", 2022 International	<1%

APPENDIX C

Pseudocode for MSL10 data collection.

```
Import necessary libraries (OpenCV, numpy, os, matplotlib, t
ime, mediapipe)

Define a function mediapipe_detection(image, model):
    Convert image from BGR to RGB
    Make image unwriteable
    Process the image using the provided model
    Make image writeable again
    Convert image from RGB to BGR
    Return the modified image and results

Define a function draw_landmarks(image, results):
    Draw landmarks for left and right hand connections on th
e image

Define a function draw_styled_landmarks(image, results):
    Draw styled landmarks for face, left hand, and right han
d connections on the image

Initialize the webcam feed using cv2.VideoCapture(0)

Set up the mediapipe holistic model with minimum detection a
nd tracking confidence

Start a loop to capture frames from the webcam:
    Read a frame from the webcam feed
    Use mediapipe_detection function to process the frame an
d get results
    Draw styled landmarks on the image using draw_styled_lan
dmarks function
    Display the image with landmarks on the screen
    Check for the 'q' key press to exit the loop

Extract key points for pose, face, left hand, and right hand
 landmarks from the results

Define a function extract_keypoints(results):
    Extract pose, face, left hand, and right hand landmarks
from results
    Return the concatenated array of all landmarks

Save the extracted keypoints from results as a numpy array
```

APPENDIX D

Pseudocode for extracting key points from pre-recorded LSA64 dataset.

```
Import necessary libraries (OpenCV, numpy, os, mediapipe)

Initialize MediaPipe holistic model and drawing utilities

Define a function to perform MediaPipe detection on an image
 using the provided model
    Convert the image to RGB format
    Process the image with the model
    Convert the image back to BGR format
    Return the modified image and detection results

Define a function to draw styled landmarks on the image base
d on the detection results
    Draw face, pose, left hand, and right hand connections o
n the image with specified colors and thickness

Define a function to extract keypoints from the detection re
sults
    Extract pose, face, left hand, and right hand keypoints
from the results
    Return the concatenated array of keypoints

Set up MediaPipe holistic model with minimum detection and t
racking confidence

Define the path to the folder containing video files

List all video files in the folder

Initialize a label count

Loop over each video file in the folder:
    Get the full path of the video file
    Extract the video name
    Open the video file for capturing frames

    Initialize frame count and set the maximum number of fra
mes to capture

    While the video capture is open and the frame count is l
ess than the maximum frames:
        Read a frame from the video
        Perform MediaPipe detection on the frame
        Draw styled landmarks on the image
```

```
            Show the image with landmarks on the screen
            Export the extracted keypoints from the detection re
sults and save as a numpy array

            Increment the frame count

            Check for the 'q' key press to break out of the loop
 gracefully

        Release the video capture object and close any open wind
ows

        Increment the label count after processing each video
```

APPENDIX E

Pseudocode for MSL10 dataset with 126 key points in 3-layer BiLSTM model.

```
Import necessary libraries (OpenCV, numpy, os, matplotlib, t
ime, mediapipe, train_test_split, to_categorical from tensor
flow.keras.utils)

Define the data path for exported numpy arrays

Define the actions to be detected and the number of sequence
s

Define the sequence length for videos

Create directories for each action and sequence in the data
path

Create a label map for actions

Initialize lists to store sequences and labels

Loop over actions and sequences:
    For each action and sequence, load the saved numpy array
s for each frame
    Extract the last 126 keypoints from each frame
    Append the extracted keypoints to a window
    Append the window and corresponding label to sequences a
nd labels lists

Convert sequences and labels to numpy arrays

Split the data into training and testing sets using train_te
st_split

Import necessary libraries (tensorflow, Sequential, Bidirect
ional, LSTM, Dense from tensorflow.keras.models, TensorBoard
 from tensorflow.keras.callbacks)

Set up the model architecture with Bidirectional LSTM layers
 and Dense layers

Compile the model with optimizer, loss function, and metrics

Start recording time for model training
```

```
Train the model on the training data with a specified number
 of epochs and TensorBoard callback

End recording time for model training

Print the computational time for model training

Display the model summary

Make predictions on the test data using the trained model

Calculate accuracy using accuracy_score from sklearn.metrics

Print the accuracy score
```

APPENDIX F

Pseudocode for real-time recognition algorithm.

```
Initialize an empty list 'sequence' to store keypoints and a
n empty list 'sentence' to store detected actions
Set a confidence threshold for action prediction

Open a video capture object for webcam feed

Set up MediaPipe holistic model with minimum detection and t
racking confidence

Start a loop to continuously capture frames from the webcam
feed
    Read a frame from the webcam feed
    Perform MediaPipe detection on the frame
    Print the detection results

    Draw styled landmarks on the image based on the detectio
n results

    Extract keypoints from the detection results
    Append the keypoints to the 'sequence' list and maintain
 only the last 30 sequences

    Check if the 'sequence' list has 30 elements
        Make a prediction using the model on the 'sequence'
list
        Print the predicted action

        Update the 'sentence' list based on the predicted ac
tion and confidence threshold
            If the predicted action is different from the la
st action in 'sentence', add the predicted action to 'senten
ce'
            Limit the 'sentence' list to a maximum of 10 act
ions

        Visualize the action probabilities on the image

    Draw a rectangle and display the detected sentence on th
e image
    Show the image with the detected landmarks and sentence
on the screen

    Check for the 'q' key press to break out of the loop
```

```
Release the video capture object and close all windows
```

APPENDIX G

MSL10 Real-time Recognition Model Screenshots.