MAKATON SIGN LANGUAGE RECOGNITION (MSLR)

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

NIKHIL R PRINCE (39110697)
ROHAN OUSEPHACHEN THAYIL (39110853)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Nikhil R Prince (39110697)** & **Rohan Ousephachen Thayil (39110853)** who carried out the Project Phase-2 entitled **"MAKATON SIGN LANGUAGE RECOGNITION (MSLR)"** under my supervision from January 2023 to April 2023.

[illegible]

Internal Guide
Dr. M. D. Anto Praveena, M.E., Ph.D.,

[illegible]

Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva Voce Examination held on 20.04.2023

[illegible] [illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **Nikhil R Prince (39110697) ,** hereby declare that the Project Phase-2 Report entitled **"MAKATON SIGN LANGUAGE RECOGNITION (MSLR)"** done by me under the guidance of **Dr. M. D. Anto Praveena, M.E., Ph.D.,** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

Nikhil R prince

DATE: 24-04-2023

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. M. D. Anto Praveena, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved the way for the successful completion of our phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Makaton is a sign language system that is widely used by individuals with communication difficulties, such as those with Autism, Down syndrome, and other developmental disorders. However, despite its popularity, there is a significant lack of technology for accurately recognizing and understanding Makaton signs. This makes it difficult for individuals with hearing impairments to communicate effectively with technology. In recent years, there have been efforts to develop systems that can recognize sign language, but the task remains challenging due to the complexity and variability of signs. Therefore, the proposed model aims to address this challenge by combining MediaPipe Holistic with a neural network to recognize Makaton sign language.

The proposed model combines MediaPipe Holistic, a framework for recognizing human body parts and their movements in real-time video, with a neural network, such as SimpleRNN, LSTM, or GRU, to recognize Makaton sign language. The model is trained on a custom dataset of Makaton sign language videos using backpropagation and the Adam optimizer. The proposed network achieved high accuracy on the test set, demonstrating its effectiveness in recognizing Makaton sign language.

This project can help to improve the way technology interacts with individuals with communication difficulties, allowing them to communicate more effectively with technology. The proposed system can be further extended to recognize other sign language systems, which can have significant implications for individuals with hearing impairments.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
AR/VR	Augmented Reality/Virtual Reality
ASIC	Application-Specific Integrated Unit
ASL	American Sign Language
BPTT	Backpropagation through time
BSL	British Sign Language
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
DTW	Dynamic Time Warping
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradient
ISL	Indian Sign Language
LSTM	Long Short-Term Memory
ML	Machine Learning
MSL	Makaton Sign Language
MSLR	Makaton Sign Language Recognition
NLP	Natural Language Processing

PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROI	Regions of Interest
TPU	Tensor Processing Unit

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Effective communication is a fundamental human need that allows individuals to interact, express their thoughts, and build relationships with others. For individuals with communication difficulties, such as those with autism, learning disabilities, or hearing loss, communicating can be a significant challenge. To overcome this challenge, various assistive technologies have been developed, one of which is Makaton.

Makaton is a system of communication that uses a combination of speech, signs, and symbols to help people communicate more effectively. It is designed to support the communication needs of individuals with a range of cognitive and communication difficulties. The system is widely used across the world, with over 100,000 children and adults using it in the UK alone. Makaton is not a language in its own right but a collection of signs and symbols that can be used alongside spoken language to enhance communication.

Despite the widespread use of Makaton, there is a lack of technology for accurately recognizing and understanding Makaton signs. The ability to recognize Makaton signs automatically would enable individuals with communication difficulties to interact more effectively with the technology, and thus facilitate communication. Therefore, the development of a system that can recognize Makaton signs accurately is crucial in improving the quality of life for individuals with communication difficulties.

Recognizing Makaton signs presents a significant challenge due to variations in gesture execution and the intricacy of the MSL language. Makaton signs are not standardised, and there can be variations in the way that different individuals produce the same sign. Moreover, Makaton signs can be highly intricate, involving subtle differences in hand shape, orientation, and movement. Therefore, developing a system that can recognize Makaton signs accurately requires the ability to handle such variations and complexities.

To address this challenge, various machine learning techniques have been applied, including deep learning methods. Among deep learning methods, recurrent neural networks (RNNs) have been employed as a solution to handle sequential data. RNNs are a type of artificial neural network that can model time series data by taking into account the sequential dependencies in the data. LSTM (Long Short-Term Memory) andGRU (Gated Recurrent Unit) are specialised types of RNNs that are specifically designed to address long-term dependencies in sequential data. These networks have been widely adopted across various domains, such as speech recognition, gesture recognition, and natural language processing (NLP), due to their remarkable success in these applications.

In this paper, we propose a model that combines MediaPipe Holistic with a neural network, such as SimpleRNN, LSTM, or GRU, to recognize Makaton sign language (MSL). The model is trained on a custom dataset of Makaton sign language videos using backpropagation and the Adam optimizer. Our proposed network achieved high accuracy on the test set, demonstrating its effectiveness in recognizing Makaton sign language. Our findings provide insights into the potential of using different RNN architectures for sign language recognition and pave the way for the development of more accurate and efficient MSL recognition systems.

1.1 MOTIVATION

Communication is a fundamental aspect of human life. It enables us to connect with others, express our thoughts and emotions, and build relationships. For individuals with communication difficulties, however, communication can be a significant challenge. Such individuals include those with autism, learning disabilities, or hearing loss. Assistive technologies, such as Makaton, have been developed to address this challenge. Makaton is a system of communication that uses a combination of speech, signs, and symbols to help people communicate more effectively.

Despite the many benefits of Makaton, there are still challenges associated with using this system of communication. One such challenge is the difficulty in recognizing and interpreting MSL gestures. This difficulty arises due to the intricate nature of the MSL language, which can involve subtle variations in gesture execution that can be challenging to detect and interpret. Furthermore, the execution of Makaton signs can differ significantly from person to person, which further complicates the task of recognizing MSL gestures.

To address these challenges, researchers have explored the use of machine learning techniques for sign language recognition. In particular, recurrent neural networks (RNNs) have emerged as a promising solution for handling sequential data, such as the movements of the hands and fingers in MSL gestures. RNNs are specialised types of neural networks that can process and interpret sequential data by maintaining a memory of past inputs. This makes them well-suited for applications such as speech recognition, gesture recognition, and NLP.

Within the domain of sign language recognition, two specialised types of RNNs have emerged as particularly effective: Long Short-Term Memory (LSTM) networks andGated Recurrent Unit (GRU) networks. These networks are specifically designed to address the problem of long-term dependencies in sequential data, which can be a significant challenge for standard RNN architectures. LSTM networks use a combinationof "memory cells" and "gates" to selectively retain or discard information from past inputs, while GRU networks use a simpler gating mechanism to control the flow of information.

In this paper, we present a comparison of the performance of three different types of RNNs - SimpleRNN, LSTM, and GRU - for recognizing Makaton signs. We propose a method that involves preprocessing video data to extract features such as the position and movement of the hands and fingers. These features are then used to train each of the three RNNs for recognition purposes. Our findings provide insights into the potential of using different RNN architectures for sign language recognition, and the effectiveness of our proposed method in improving the recognition of MSL gestures. We hope that our work will contribute to the development of more advanced technologies for improving communication among individuals with disabilities, and ultimately, to a more inclusive and accessible society.

CHAPTER 2

LITERATURE SURVEY

For many years, researchers in machine learning and computer vision have been intrigued by the subject of sign language recognition. Early attempts to recognize sign language involved the use of sensor gloves for capturing the motion of hand and finger gestures and recognizing them using traditional machine learning algorithms [2]. But these methods were often expensive and cumbersome. More recent approaches have used computer vision techniques to extract features from video recordings of sign language gestures.

Initial attempts in this field concentrated on utilising Hidden Markov Models (HMMs) to identify individual signs. Grobel and Assan [3] introduced an HMM-based system for recognizing isolated signs, which showed encouraging outcomes. However, this approach has some limitations, such as its sensitivity to noise and variations in the signing styles.

Subsequently, researchers started investigating other methods for recognizing sign language, including Dynamic Time Warping (DTW) and Histogram of Oriented Gradient (HOG) features. Jangyodsuk and colleagues [4] introduced a DTW and HOG-based system for recognizing sign language that attained high precision. However, this technique is computationally demanding and may not be appropriate for real-time use. Another prevalent approach is employing Conditional Random Fields (CRFs) to spot sign language. Yang and colleagues [5] introduced a CRF-based threshold model for spotting sign language that achieved 87% accuracy.

Conventional approaches to sign language recognition such as Dynamic Time Warping (DTW) and Hidden Markov Models (HMMs) algorithms have shown success in recognizing isolated signs but have limited performance when recognizing continuous sign language. Fortunately, with the recent advancements in computer vision and deep learning, there have been significant improvements in the accuracy of sign language recognition.

One approach that has shown promising results is the use of CNNs to recognize sign language gestures. Pigou and colleagues [6] introduced a CNN-based method for recognizing sign language that attained high precision rates on benchmark datasets. The architecture of their model included two separate CNNs, one responsible for extracting hand features and the other for extracting upper body features. Both CNNs were designed with three layers, and achieved a recognition accuracy of 95.68% on theCLAP14 dataset.

Another prevalent method is employing recurrent neural networks (RNNs) to recognize temporal patterns in sign language gestures. Bantupalli and Xie [7] used a combination of computer vision and deep learning techniques to identify ASL gestures. They proposed a model that uses a combination of a CNN and an RNN network to recognize 150 ASL gestures. Their model achieved an accuracy of 90% on the American Sign Language Dataset.

Compared to the CNN-RNN model, the proposed model that combines MediaPipe Holistic with a neural network (SimpleRNN, LSTM or GRU) offers several advantages. Firstly, it requires less data to produce a highly accurate model. Additionally, the proposed model has a simpler neural network architecture, with approximately half a million parameters, which results in faster training times. Furthermore, as the neural network is simpler, it enables real-time detection with faster processing speeds. Unlike the CNN-RNN model, which has around 30 to 40 million parameters, the proposed model offers a more efficient solution for sign language recognition, making it a promising tool for practical applications.

Overall, deep learning techniques have shown great promise in enhancing the precision of systems that recognize sign language. However, there is still ongoing research in this area, with new models and datasets being proposed regularly.

2.1 INFERENCES FROM LITERATURE SURVEY

The literature survey conducted for this project reveals that sign language recognition has been a subject of extensive research in recent years. Various approaches have been proposed to address the challenges of recognizing sign language, including the use of computer vision and machine learning techniques. The majority of the studies have focused on American Sign Language (ASL), while other sign languages such as Indian Sign Language (ISL) and British Sign Language (BSL) have received less attention.

Moreover, the literature survey highlights the challenges in sign language recognition, including the high variability and complexity of sign language, which requires sophisticated algorithms for accurate recognition. Additionally, the limited availability of data and the lack of standardised datasets pose significant challenges for training and evaluating sign language recognition systems.

Furthermore, the survey reveals that the use of deep learning techniques, particularly recurrent neural networks (RNNs), has shown promising results in sign language recognition. In particular, the use of Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) has been widely investigated for this task. Convolutional neural networks (CNNs) have also been utilised for feature extraction in some studies.

Overall, the literature survey suggests that the development of accurate and practical sign language recognition systems is crucial for improving communication and accessibility for individuals with communication difficulties. The use of deep learning techniques, particularly RNNs and CNNs, appears to be a promising approach for addressing the challenges of recognizing sign language.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

While the existing systems for sign language recognition have shown promising results, there are still several open problems that need to be addressed. One of the primary issues is the limited availability of annotated sign language datasets, which can hinder the development and evaluation of new models. The lack of standardised datasets also makes it challenging to compare different approaches and assess their performance.

Another open problem is the development of models that can handle the variability in sign language gestures due to differences in signing styles, regional variations, and individual differences. There is a need for models that can generalise well to unseen data and can adapt to different sign languages and signing styles.

Another issue is the real-time recognition of sign language gestures. While some existing systems can recognize sign language gestures in real-time, there is still a need for models that can operate in real-world environments with varying lighting conditions, camera angles, and background noise.

Finally, there is a need to make sign language recognition systems more accessible and affordable. Currently, most systems require specialised hardware or expensive software, which can limit their adoption and accessibility for individuals with communication difficulties. There is a need for low-cost, portable systems that can be easily integrated into everyday devices such as smartphones, tablets, and smart home systems.

Overall, addressing these open problems can help improve the accessibility and effectiveness of sign language recognition systems and support the communication needs of individuals with communication difficulties.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES

Before undertaking any project, it is essential to assess its feasibility to determine if it is practical and viable. In the case of this project, we conducted a feasibility study to evaluate the project's viability and whether it could be executed successfully. The feasibility study primarily focused on technical, economic, and operational aspects.

3.1.1 TECHNICAL FEASIBILITY

The proposed model that combines MediaPipe Holistic with a neural network for sign language recognition is technically feasible. The technology required for this project is readily available, including the MediaPipe library and deep learning frameworks like TensorFlow and Keras. Additionally, the MediaPipe Holistic model provides us with the capability to track various body parts accurately, including the face, hands, and pose. The implementation of the RNNs (SimpleRNN, LSTM, and GRU) for recognizing Makaton signs has been extensively researched and widely adopted across various domains. Therefore, the project's technical feasibility is confirmed.

3.1.2 ECONOMICAL FEASIBILITY

The proposed project has moderate economic feasibility. The project will require the purchase of equipment such as computers with webcam. However, the costs of the required technology are within a reasonable budget. Additionally, the project's long-term economic benefits may outweigh the initial costs, such as facilitating communication for individuals with disabilities and improving their quality of life.

3.1.3 OPERATIONAL FEASIBILITY

The project's operational feasibility is high, as the implementation of the proposed model will be relatively simple. The MediaPipe Holistic model will be used to extract features from video data, and the three RNNs will be used for recognition purposes. The model will be trained on a dataset of Makaton signs, and the trained model will be tested for accuracy. If the results are satisfactory, the model can be deployed on variousdevices and platforms, allowing individuals with disabilities to communicate effectively.

3.2 RISK ANALYSIS

In any project, there are potential risks that must be identified and mitigated. In the case of this project, the primary risks are technical, operational, and ethical risks.

3.2.1 TECHNICAL RISKS

The project's technical risks primarily relate to the accuracy and reliability of the proposed model. There is a risk that the model may not accurately recognize Makaton signs, resulting in communication errors. Additionally, there is a risk of overfitting the model, where it performs well on the training data but poorly on unseen data. To mitigate these risks, we will ensure that the dataset used for training the model is diverse and that the model is thoroughly tested to ensure its accuracy and reliability.

3.2.2 OPERATIONAL RISKS

The operational risks relate to the practical implementation of the proposed model. The model may not work correctly in real-world scenarios, or there may be hardware or software issues that could affect the model's performance. To mitigate these risks, we will test the model extensively in real-world scenarios, identify potential issues, and resolve them promptly.

3.2.3 ETHICAL RISKS

The project's ethical risks relate to the potential misuse of the technology and the privacy of the individuals using it. There is a risk of unauthorised access to the video data, which could result in the violation of privacy rights. Additionally, there is a risk of individuals misusing the technology for fraudulent or malicious purposes. To mitigate these risks, we will implement appropriate measures to prevent the misuse of the technology.

In conclusion, the feasibility study and risk analysis show that the proposed project is technically feasible, economically viable, and operationally feasible, with the identified risks mitigated. The proposed model that combines MediaPipe Holistic with a neural network has the potential to improve the lives of individuals with disabilities and facilitate more effective communication.

3.3 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

- Jupyter notebook : The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

- Python : Python is a high-level, general-purpose programming language. Its design philosophy emphasises code readability with the use of significant indentation.

- Matplotlib : Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

- NumPy : NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- OpenCV : OpenCV is a library of programming functions mainly aimed at real-time computer vision.

- MediaPipe : MediaPipe offers cross-platform, customizable ML solutions for live and streaming media. End-to-End acceleration: Built-in fast ML inference and processing accelerated even on common hardware.

- TensorFlow : TensorFlow is a free and open-source software library for machine learning and artificial intelligence.

- scikit-learn : Scikit-learn is a free software machine learning library for the Python programming language.

3.4 SYSTEM USE CASE

The use case for our system is primarily aimed at individuals with communication difficulties, such as those with Autism, Down syndrome, and other developmental disorders who use sign language as their primary mode of communication. Our system is designed to recognize sign language gestures in real-time and convert them into text or speech output, enabling better communication between these individuals and others who may not understand sign language.

In a typical use case scenario, the user would interact with the system by performing a sign language gesture in front of a camera, which would be processed by the MediaPipe Holistic algorithm to extract landmark points. These landmark points would then be fed into the RNN model to recognize the gesture and convert it into text or speech output. The output can be displayed on a screen or communicated through a speaker or other audio output device.

The system can be used in various settings, such as at home, in school, or in public places. It can be particularly useful in situations where a sign language interpreter is not available, such as in remote or rural areas. Moreover, the system can be integrated with assistive technologies, such as smart home systems and communication tools, to enable a more seamless and accessible user experience.

Overall, the use case for our system is centred on improving communication and accessibility for individuals with communication difficulties, particularly those who rely on sign language as their primary mode of communication. Our system aims to provide a more efficient, effective, and practical solution for recognizing and interpreting sign language gestures in real-time, with the potential to improve the lives of millions of people around the world.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

The proposed system for Makaton Sign Language Recognition (MSLR) using SimpleRNN, LSTM, and GRU networks involves several important steps. The first step is data collection, which is crucial for building a robust and accurate recognition system. To recognize Makaton signs, we first collected a dataset by recording videos of individuals signing Makaton signs. The dataset included a range of signs and variations in signing style, which ensures that the system is capable of recognizing signs from different individuals with varying styles. These videos were labelled with thecorresponding text or speech, so that the networks could learn to associate different signs with different words or phrases.

The second step is data pre-processing, where the collected videos were pre-processed to extract the necessary information for the analysis. This included cropping the video to focus on the signing hand and converting the video to a sequence of images. This step is crucial in extracting relevant information from the video, while reducing the noise and background information that can hinder the recognition process.

The third step is feature extraction, which involves obtaining a collection of features from the pre-processed data to represent the signing gestures. These features encompass the hand's position, movement, and shape. The use of hand movement and shape information in feature extraction will provide more robustness against variations in signing style, which can be helpful in building a system that can accurately recognize signs from different individuals.

The fourth step is network training, where SimpleRNN, LSTM, and GRU networks are separately trained on the extracted features using backpropagation and the Adam optimizer. Each network consists of several layers that culminate in a fully connected layer. The networks are trained to classify the signing gestures into different classes, such as different Makaton signs. The use of SimpleRNN, LSTM, and GRU networks is important in handling sequential data, which is a characteristic of sign language recognition. The use of these networks can also handle variations in signing style, which can increase the accuracy of the system.

The fifth step is evaluation, where the trained networks are evaluated on a separate test dataset to measure their performance in recognizing and classifying Makaton signs. The performance is measured using metrics such as accuracy score. This step is important in determining the accuracy and robustness of the system, which is crucial in building a system that can accurately recognize signs in real-world scenarios.

The final step is sign language recognition, where the trained networks can be employed for real-time recognition of Makaton Sign Language by processing the video input and recognizing the signs. This step is crucial in building a system that can be used in real-world scenarios, such as in communication with individuals who use Makaton Sign Language.

Overall, the proposed method for Makaton Sign Language Recognition (MSLR) using SimpleRNN, LSTM, and GRU networks is a promising approach for building a robust and accurate sign language recognition system. The use of hand movement and shape information in feature extraction, along with the use of SimpleRNN, LSTM, and GRU networks in network training, can increase the accuracy and robustness of the system. This approach can have important applications in communication with individuals who use Makaton Sign Language, and can greatly enhance their ability to communicate with others.

4.1 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

The block diagram of the proposed system has been shown below in Fig 4.1. The OpenCV allows a straightforward interface to capture livestream with the webcam. It converts video into grayscale and displays it. Then the MediaPipe Holistic integrates separate models for pose, face and hand components, each of which are optimised for their particular domain.Then we train and test the model which is a systematic process of collecting or extracting key points and using that information to improve our training. We use the Long Short-Term Memory (LSTM), Simple Recurrent Neural Network (RNN)and Gated Recurrent Unit (GRU) networks for making the predictions and use a confusion matrix for evaluating the model. The output of the model is captured through

Real-Time Testing.

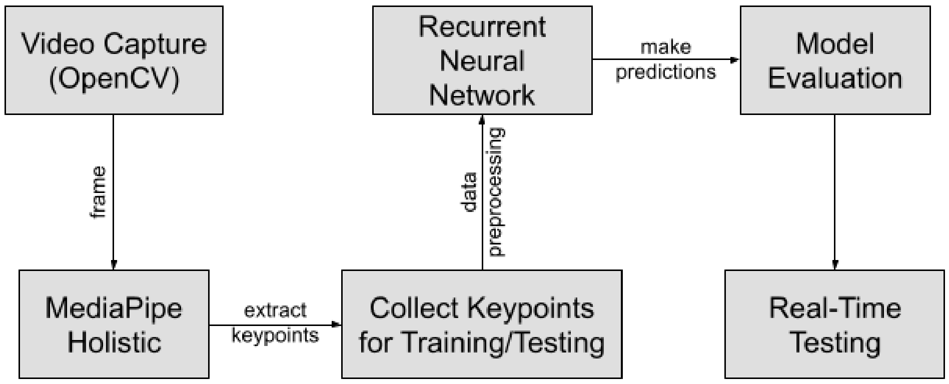


Fig 4.1: System Architecture for Makaton Sign Language Recognition

4.2 HAND GESTURE RECOGNITION

Gesture recognition is an essential topic in computer science and builds technology that aims to interpret human gestures where anyone can use simple gestures to interact with the device without touching them directly. The entire procedure of tracking gestures to their representation and converting them to some purposeful command is known as gesture recognition. Identify from explicit hand gestures as input then process these gestures representation for devices through mapping as output is the aim in hand gestures recognition. Recognition of the hand gesture in kinds of literature based on extracted features is divided into three groups as follows:

- **High-Level Features-Based Approaches**: Aim to figure out the position of the palm and joint angles such as the fingertips, joint location, or anchor points of the palm. Whereas, effect collisions or occlusions on the image are difficult to detect after features are extracted, and sensitivity segmentation performance on 2D hand image are the problem that occurred frequently.The gestures are defined from the results with a set of rules and conditions from the vectors and joints of the hands.

- **Low-Level Feature-Based Approaches**: Utilised these features could be extracted quickly for robust noise. Zhou discovered recognition of the hand shape as a cluster-based signature using a novel distance metric called Finger Earth’s Distance.Stanner determines the bounding region of the hand elliptically to implement hand recognition based on principal axes. Yang did research using the optical flow of the hand region as a low level feature. Low-Level Feature-Based is not efficient when a cluttered background.

- **3D Reconstruction-Based Approaches**: Use the 3D model of features for achieving the construction of the hand completely. Research showed that successfully segmenting the hand in skin colour needs similarity and high contrast of the background related to the hand through structured light to bring in 3D of depth data. Another one uses a stereo camera to track numerous interest points of the superficies of the hand which results in difficulty for handling robust 3Dreconstruction, despite data containing 3D has valuable information that can help dispose of vagueness. See for more 3D reconstruction-based approach.

From kinds of literature, there are three Hand gesture recognition methods, as follow:

- **Machine Learning Approaches**: The resulting output came from the stochastic process and approach based on statistical modelling for dynamic gestures such as PCA, HMM , advanced particle filtering and condensation algorithm.

- **Algorithm Approaches**: Collection of encoded conditions and restraints manually for defining as gestures in dynamic gestures. Galveia applied a 3rd-degree polynomial equation to determine the dynamic component of the hand gestures (create a 3rd-degree polynomial equation, recognition, reduced complexity of equations, and comparison handling in the gestures library).

- **Rule-based Approaches**: Suitable for dynamic gestures or static gestures which contain a set of pre-encoded rules and features inputs. The features of input gestures are extracted and compared to the encoding rules that flow the recognized gestures. Matching between gestures with rule and input which is output approved as known gestures.

4.3 MEDIAPIPE FRAMEWORK

Today, there are many frameworks or libraries of machine learning for hand gesture recognition. One of them is MediaPipe. The MediaPipe is a framework designed to implement production-ready machine learning that must build pipelines to perform inference over arbitrary sensory data, has published code accompanying research work, and build technology prototypes. In MediaPipe, graph modular components come from a perception pipeline along with the function of inference model function, media processing model, and data transformations. Graphs of operations are used in other machine learning such as Tensor flow, MXNet, PyTorch, CNTK, OpenCV 4.0.

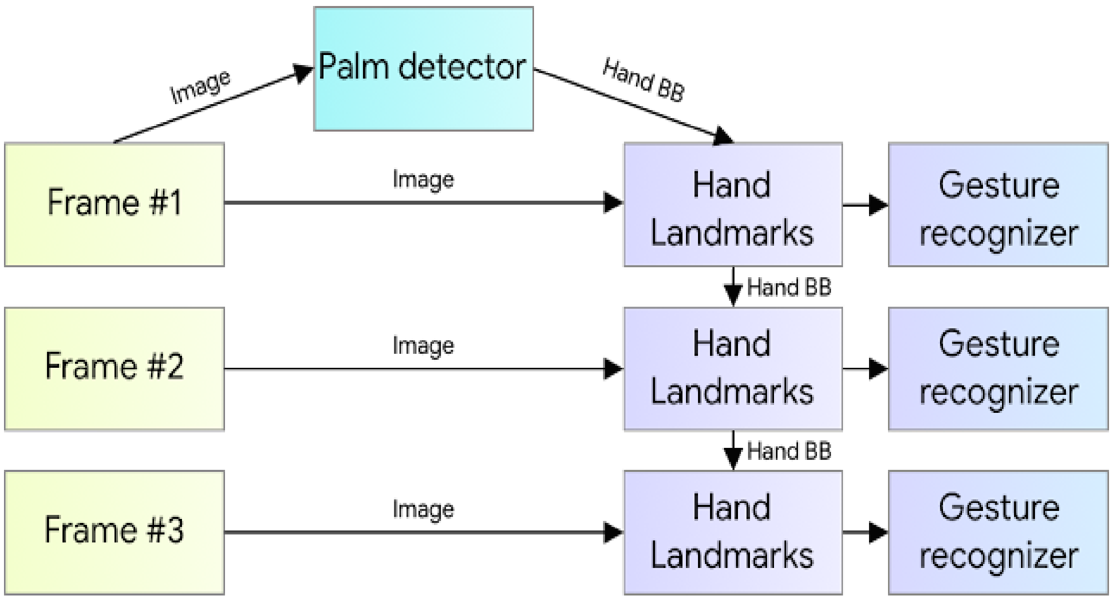


Fig 4.2: Hand Perception Pipeline

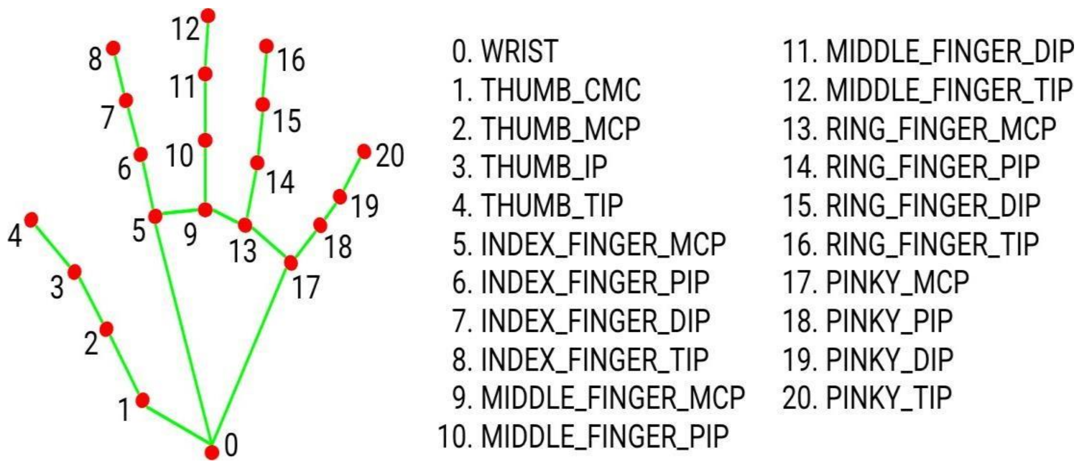
Using MediaPipe for hand gesture recognition has been researched by Zhang before, using a single RGB camera for AR/VR application in a real-time system that predicts a hand skeleton of the human. We can develop a combined MediaPipe using other devices. The MediaPipe implements the pipeline in Fig 4.2. consists of two models for hand gesture recognition as follows :

1. A palm detector model processes the captured image and turns the image with an oriented bounding box of the hand.
2. A hand landmark model processes a cropped bounding box image and returns 3D hand key points on hand.
3. A gesture recognizer that classifies 3D hand key points then configures them into a discrete set of gestures.

4.3.1 PALM DETECTOR MODEL

The MediaPipe framework has built an initial palm detector called BlazePalm. Detecting the hand is a complex task. Step one is to train the palm instead of the
hand detector, then using the non-maximum suppression algorithm on the palm, where it is modelled using square bounding boxes to avoid other aspect ratios and reducing the number of anchors by a factor of 3-5. Next, encoder-decoder of feature extraction that is used for bigger scene context-awareness even small objects, lastly, minimise thefocal loss during training with support a large number of anchors resulting from the high scale variance.

4.3.2 HAND LANDMARK



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

Fig 4.3: Hand Landmark in Mediapipe

Achieves precise key point localization of 21 key points with a 3D hand-knuckle coordinate which is conducted inside the detected hand regions through regression which will produce the coordinate prediction directly which is a model of the hand landmark in MediaPipe, as shown in the Fig 4.3.

4.3.3 MEDIAPIPE HOLISTIC

MediaPipe Holistic provides a unified topology for a groundbreaking 540+ keypoints (33 pose, 21 per-hand and 468 facial landmarks) and achieves near real-time performance on mobile devices. MediaPipe Holistic is being released as part of MediaPipe and is available on-device for mobile (Android, iOS) and desktop. We are also introducing MediaPipe's new ready-to-use APIs for research (Python) and web (JavaScript) to ease access to the technology.

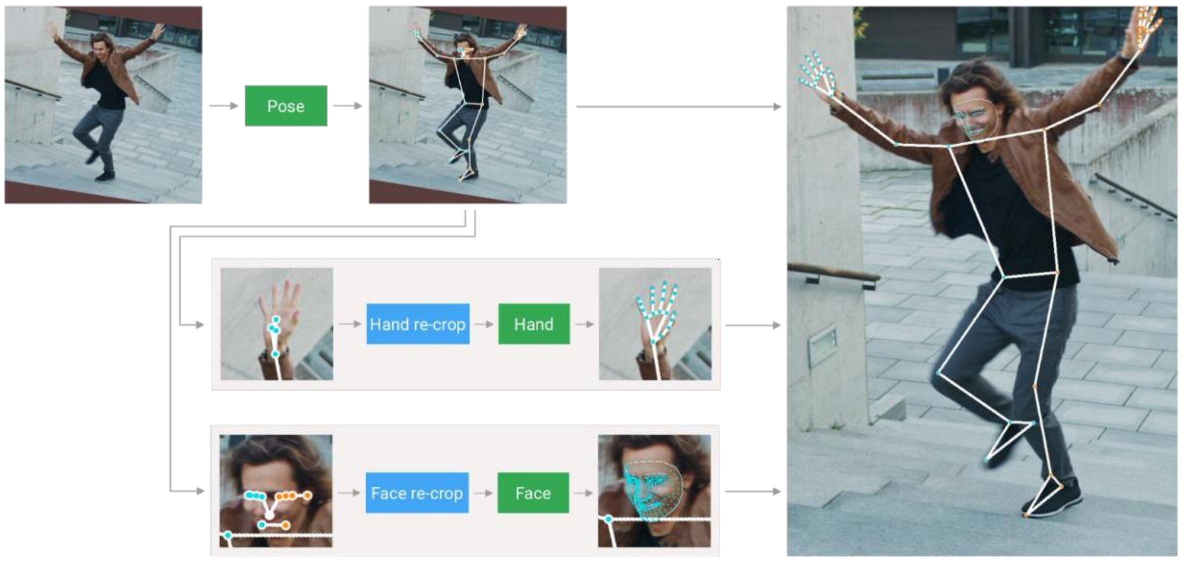


Fig 4.4: MediaPipe Holistic pipeline overview

The MediaPipe Holistic pipeline integrates separate models for pose, face and hand components, each of which are optimised for their particular domain. However, because of their different specialisations, the input to one component is not well-suited for the others. The pose estimation model, for example, takes a lower, fixed resolution video frame (256x256) as input. But if one were to crop the hand and face regions from that image to pass to their respective models, the image resolution would be too low for accurate articulation.

First, MediaPipe Holistic estimates the human pose with BlazePose's pose detector and subsequent keypoint model. Then, using the inferred pose key points, it derives three regions of interest (ROI) crops for each hand (2x) and the face, and employs a re-crop model to improve the ROI as shown in Fig 4.4. The pipeline then crops the full-resolution input frame to these ROIs and applies task-specific face and hand modelsto estimate their corresponding keypoints. Finally, all key points are merged with those of the pose model to yield the full 540+ key points which is shown in Fig 4.5.

MediaPipe Holistic requires coordination between up to 8 models per frame — 1 pose detector, 1 pose landmark model, 3 re-crop models and 3 keypoint models for hands and face. While building this solution, we optimised not only machine learning models, but also pre- and post-processing algorithms which take significant time on most devices due to pipeline complexity. In this case, moving all the pre-processing computations to GPU resulted in ~1.5 times overall pipeline speedup depending on the device. As a result, MediaPipe Holistic runs in near real-time performance even on mid-tier devices and in the browser.

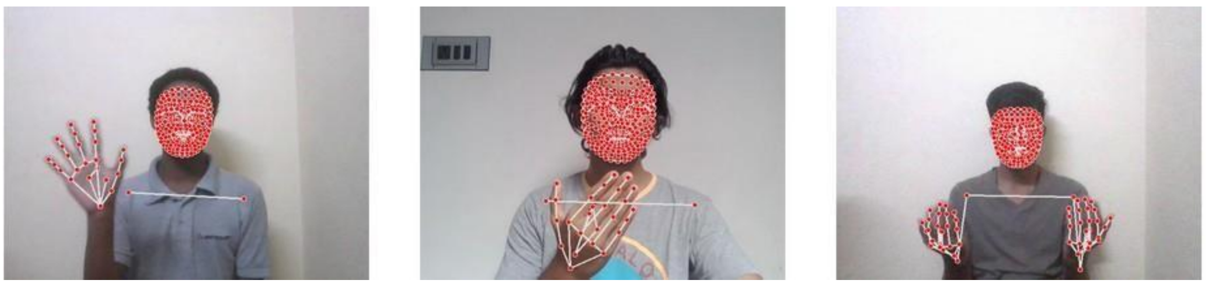


Fig 4.5: MediaPipe Holistic Key Points

4.4 TENSORFLOW

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs).TensorFlow uses a unified dataflow graph that is shown in the below Fig 4.6 to represent both the computation in an algorithm and the state on which the algorithm operates. We draw inspiration from the high-level programming models of dataflow systems and the low-level efficiency of parameter servers. Unlike traditional dataflow systems, in which graph vertices represent functional computation on immutable data, TensorFlow allows vertices to represent computations that own or update mutable state.Edges carry tensors (multi-dimensional arrays) between nodes, and TensorFlow transparently inserts the appropriate communication between distributed subcomputations.By unifying the computation and state management in a single programming model, TensorFlow allows programmers to experiment with different parallelization schemes that, for example, offload computation onto the servers that hold the shared state to reduce the amount of network traffic. We have also built various coordination protocols, and achieved encouraging results with synchronous replication, echoing recent results [10, 18] that contradict the commonly held belief that asynchronous replication is required forscalable learning.

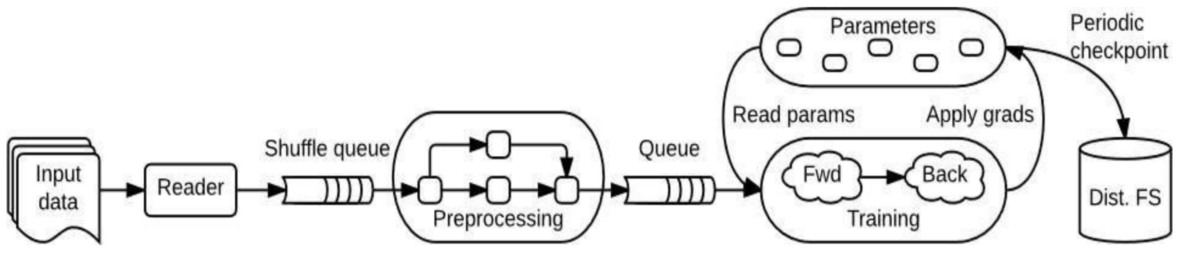


Fig 4.6: TensorFlow Data Flow Graph

4.5 SIMPLE RECURRENT NEURAL NETWORK

SimpleRNN is a type of Recurrent Neural Network (RNN) that is used for modelling

sequential data. RNNs are neural networks that can process sequences of inputs by maintaining a "memory" or "hidden state" of previous inputs, which is then used to inform the processing of subsequent inputs. SimpleRNN is a type of RNN in which the output at each time step is fed back into the network as input for the next time step, creating a fully-connected RNN architecture.
In a SimpleRNN, the input at each time step is fed into a layer of neurons, which produces an output and updates the hidden state. The output at each time step is then passed back into the network as input for the next time step, along with the currentinput. This feedback loop allows the network to maintain a memory of previous inputs and use that memory to inform the processing of subsequent inputs.

The SimpleRNN is called "simple" because it has a single layer of neurons that is fully connected to the input and output layers. However, despite its simplicity, SimpleRNNs can be powerful tools for modelling sequential data, including natural language, speech, and time series data.

One of the main advantages of SimpleRNNs is their ability to handle variable-length sequences of input data. Because the network maintains a hidden state that is updated at each time step, it can handle inputs of different lengths and adapt its processing to the length of the input sequence. This makes SimpleRNNs well-suited for tasks such as language modelling, where the length of the input sequence can vary widely.

However, one of the limitations of SimpleRNNs is their difficulty in learning long-term dependencies in the data. Because the hidden state is updated at each time step, information from earlier time steps can quickly "decay" or be overwritten by information from later time steps. This can make it challenging for the network to maintain a memory of earlier inputs over long sequences.

To address this issue, more advanced RNN architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been developed. These architectures use additional mechanisms such as memory cells and gates to better capture long-term dependencies in the data.

In summary, SimpleRNN is a powerful tool for modelling sequential data due to its ability to handle variable-length sequences and maintain a memory of previous inputs. However, it has limitations in learning long-term dependencies, which can be addressed by more advanced RNN architectures such as LSTM and GRU.

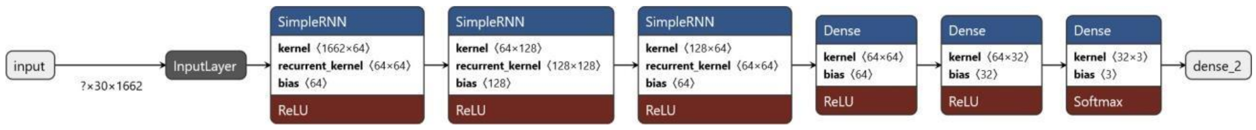


Fig 4.7: Architecture for SimpleRNN

4.6 LSTM NEURAL NETWORK

LSTM, short for Long Short-Term Memory, is a type of recurrent neural network (RNN) that is designed to address the problem of vanishing gradients in traditional RNNs.

In traditional RNNs, the output of the network at each time step is fed back as input to the next time step. However, as the network tries to learn long-term dependencies in thedata, the gradients can become very small or vanish, making it difficult for the network to learn. This can result in poor performance or even cause the network to stop learningaltogether.

LSTMs were designed to overcome this issue by introducing a memory cell, which allows the network to selectively remember or forget information from previous time steps. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate.

The input gate determines which information from the current time step should be added to the memory cell. The forget gate determines which information from the previous time step should be forgotten. And the output gate determines which information from the memory cell should be used to produce the output at the current time step.

By selectively remembering or forgetting information from previous time steps, LSTMs are able to handle long-term dependencies in the data and avoid the vanishing gradient problem. This makes them well-suited for a wide range of applications, including speech recognition, natural language processing, and image captioning.

One potential downside of LSTMs is that they have a larger number of parameters than traditional RNNs, which can make them slower to train and require more data. However, there are now many optimizations and variations of LSTMs that can help address these issues.

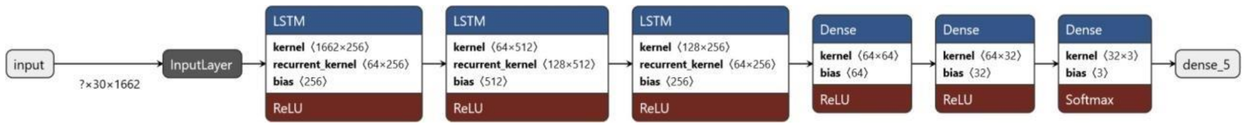


Fig 4.8: Architecture for LSTM

4.7 GATED RECURRENT UNIT

Gated Recurrent Unit (GRU) is a type of neural network architecture that is similar to the Long Short-Term Memory (LSTM) network. It is particularly useful for processing sequential data, such as time-series data or natural language processing tasks.

The GRU architecture was proposed as a simpler alternative to LSTM, which has a more complex structure. Like LSTM, GRU is also designed to avoid the vanishing gradient problem that can occur when training traditional recurrent neural networks.

One of the main differences between GRU and LSTM is the number of gates that they use to control the flow of information. While LSTM has three gates (input, output, and forget), GRU has only two gates (reset and update). The reset gate determines how much of the past information to forget, while the update gate controls how much of the current input to remember.

Another difference between GRU and LSTM is that GRU does not have a separate memory unit like the cell state in LSTM. Instead, it uses a hidden state that stores information about the past inputs.

In terms of training, GRU can be trained using backpropagation through time (BPTT), a method that computes the gradient of the loss function with respect to the network weights. This allows the network to learn the optimal weights for making predictions based on the input sequence.

Overall, GRU is a powerful neural network architecture that is widely used in a variety of applications, including speech recognition, machine translation, and sentiment analysis. Its simplicity and efficiency make it a popular choice for many researchers and developers in the field of deep learning.

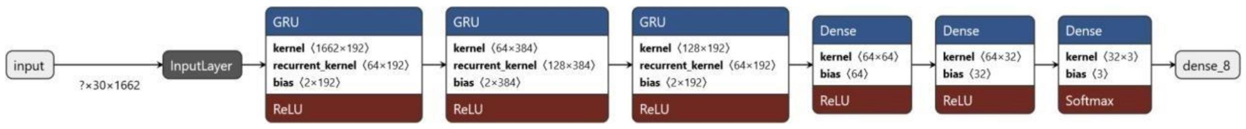


Fig 4.9: Architecture for GRU

CHAPTER 5

IMPLEMENTATION DETAILS

To implement our proposed Makaton Sign Language Recognition (MSLR) system, we conducted several experiments using a custom dataset consisting of 180 videos of three signs: Hello, Thank You, and Welcome. The dataset was created with the help of three volunteers, each sign having 60 videos of 30 frames each. We used MediaPipe Holistic to produce a total of 543 landmarks, with 33 landmarks for pose estimation, 468landmarks for facial surface, and 21 landmarks for each hand. It is illustrated in figure 4.5.

To preprocess the data, we collected the keypoint values (landmarks) from the series of frames and created NumPy arrays consisting of 1662 landmarks (334+4683+213+213) for each frame in each video. The dataset was divided into two sets: a training set and a testing set. We used an 80:20 ratio for the split, with 36 videos reserved for the testing set. The input shape for the first layer of the sequential model was (30, 1662).

We experimented with three different types of recurrent neural networks (RNNs): SimpleRNN, LSTM, and GRU. For each type of RNN, we used a three-layer architecture with 64 hidden units in the first and third layers and 128 hidden units in the second layer. The architectures were followed by two dense ReLU layers, the first one having 64 hidden layers and the second one with 32 hidden layers. We used a softmax layer atop the RNNs for predicting the sign label. The architectures of the three models are shown in figure 4.7, 4.8 and 4.9 respectively. The Adam optimizer was used to train the networks for 2000 epochs, with a batch size of 64.

To evaluate the performance of our proposed method, we varied several parameters, such as the number of RNN layers, the number of neurons in each RNN layer, and the batch size.

We tested our proposed method using real-time video of individuals signing Makaton signs to verify its ability to recognize Makaton signs in real-time. The results were promising, with the system recognizing signs accurately and quickly.

In summary, we implemented our MSLR system using a custom dataset and experimented with three different types of RNNs. Our system has the potential to improve the accessibility and communication for individuals who use Makaton Sign Language. Future work could explore the performance of the system with a larger and more diverse dataset and investigate ways to optimise the system for real-time applications.

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The development and deployment setup for our MSL recognition system consisted of several components, including hardware, software, and data.

Hardware: For development and testing, we used a desktop computer with an NVIDIA GeForce GTX 1650 graphics card, which provided the necessary computational power to train our deep learning models.

Software: We used several software tools and libraries to develop and deploy our MSL recognition system. The system was implemented using Python programming language and the TensorFlow deep learning framework. We also used several other Python libraries, including OpenCV, MediaPipe, and NumPy, for image processing and data manipulation.

Data: The development of our MSL recognition system required a large amount of data, including videos of sign language gestures, and corresponding label data. We used MediaPipe Holistic to extract the 2D landmarks from the videos, and then manually annotated the landmarks with labels corresponding to each sign. To create a diverse dataset, we collected data from multiple users with different backgrounds and signed in different environments.

In summary, the development and deployment setup for our MSL recognition system involved a combination of hardware, software, and data. We leveraged the computational power of a desktop computer for model training and model deployment. We also used several software tools and libraries, including Python, TensorFlow, OpenCV, and MediaPipe, to develop and deploy the system. Finally, we collected and annotated a diverse dataset to train the deep learning models.

5.2 ALGORITHMS

In our study, we employed Recurrent Neural Networks (RNNs) for recognizing MSL gestures. Specifically, we experimented with three different types of RNNs: SimpleRNN, LSTM, and GRU.

SimpleRNN is the simplest type of RNN, and it is well-suited for simple sequential data. It has a single recurrent layer, and its architecture is relatively straightforward. However, it suffers from the "vanishing gradient" problem, which can make it difficult to train on long sequences.

LSTM (Long Short-Term Memory) is a more advanced type of RNN that addresses the vanishing gradient problem. It has a more complex architecture with three gates (input, output, and forget) that control the flow of information. The memory cells of an LSTM network can retain information over long periods, making it well-suited for recognizing complex temporal patterns.

GRU (Gated Recurrent Unit) is a variant of the LSTM that has fewer parameters and is therefore faster to train. It has two gates (reset and update) that control the flow of information, and it has been shown to achieve comparable performance to LSTM on many tasks.

For landmark extraction, we utilised the MediaPipe Holistic model, which is a pre-trained neural network for recognizing human poses and facial landmarks. MediaPipe Holistic is based on the concept of a "holistic" representation of the human body, which includes both 2D and 3D landmarks. This model is trained on large datasets and is capable of accurately detecting facial expressions, hand gestures, and body movements.

Overall, our study utilised existing algorithms and models for MSL recognition, but we also experimented with different types of RNNs to improve accuracy and efficiency. Our findings suggest that the use of GRU-based models can lead to improved accuracy in MSL recognition, but SimpleRNN can be a better choice for real-time applications due to its simpler architecture.

5.3 TESTING

Testing is a crucial aspect of any software development project, and our MSL recognition system is no exception. The goal of testing is to ensure that the system functions correctly and meets the specified requirements. In our project, we conducted several types of testing to evaluate the accuracy and performance of our system.

Firstly, we performed unit testing, which involved testing each component of our system individually to ensure that it functioned correctly. We tested our MediaPipe Holistic pipeline for landmark extraction, our RNN models for MSL recognition, and the integration of these components into a complete system. We also performed integrationtesting to ensure that the different components worked together as expected.

Secondly, we conducted functional testing to evaluate the system’s ability to correctly recognize MSL gestures. We used our custom dataset to test the system's accuracy and evaluated its performance using accuracy metric. We also performed real-time testing to evaluate the system's performance under real-world conditions.

Finally, we conducted usability testing to evaluate the system's effectiveness of its feedback mechanisms. This involved working with a group of users and collecting feedback on the system's reliability, and effectiveness.

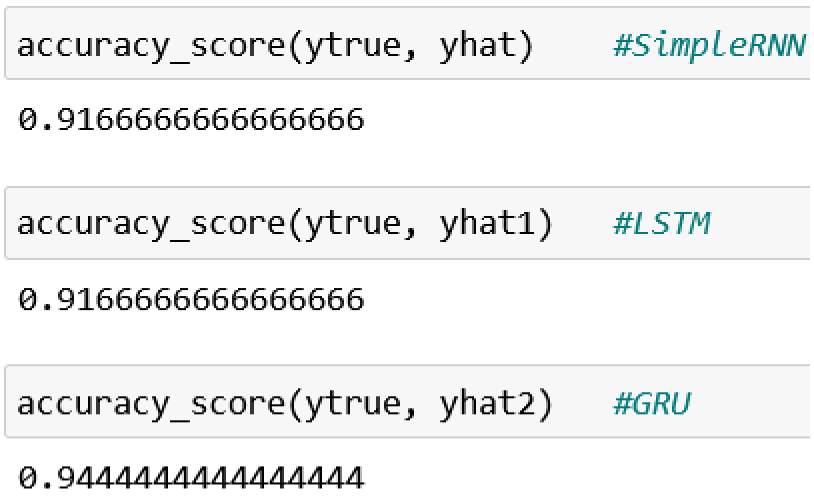
Overall, testing was an important aspect of our project, and it allowed us to identify and address issues early in the development process. By conducting thorough testing, we were able to ensure that our system was reliable, accurate, and effective in recognizing MSL gestures, thereby providing a valuable communication tool for individuals with communication difficulties.

CHAPTER 6

RESULTS AND DISCUSSION

The study compared the performance of three different types of recurrent neural networks (RNNs) in recognizing Makaton sign language. Makaton is a form of communication used by individuals with speech and language difficulties, and the study aimed to explore whether RNNs could be effective in recognizing it.

The study found that the GRU model had the highest accuracy of 94.4%, followed by SimpleRNN and LSTM, which both achieved an accuracy of 91.6% as shown in figure 6.1. This suggests that all three types of RNNs are effective in recognizing Makatonsign language. The findings of this study have significant implications for the development of practical applications that could benefit individuals with speech and language difficulties.



```
accuracy_score(ytrue, yhat)     #SimpleRNN
0.9166666666666666

accuracy_score(ytrue, yhat1)    #LSTM
0.9166666666666666

accuracy_score(ytrue, yhat2)    #GRU
0.9444444444444444
```

Fig 6.1: Accuracy score obtained by various models

However, the study also found that SimpleRNN was the most efficient model for real-time recognition due to its significantly smaller number of parameters compared to LSTM and GRU. This highlights the importance of considering not only accuracy but also efficiency in selecting the appropriate model for real-time recognition scenarios.

The use of RNNs in various fields such as speech recognition and natural language processing has been increasing due to their ability to handle sequential data. The study's findings suggest that RNNs can also be effective in recognizing sign language, specifically Makaton sign language, which has the potential to improve the communication abilities of individuals with speech and language difficulties.

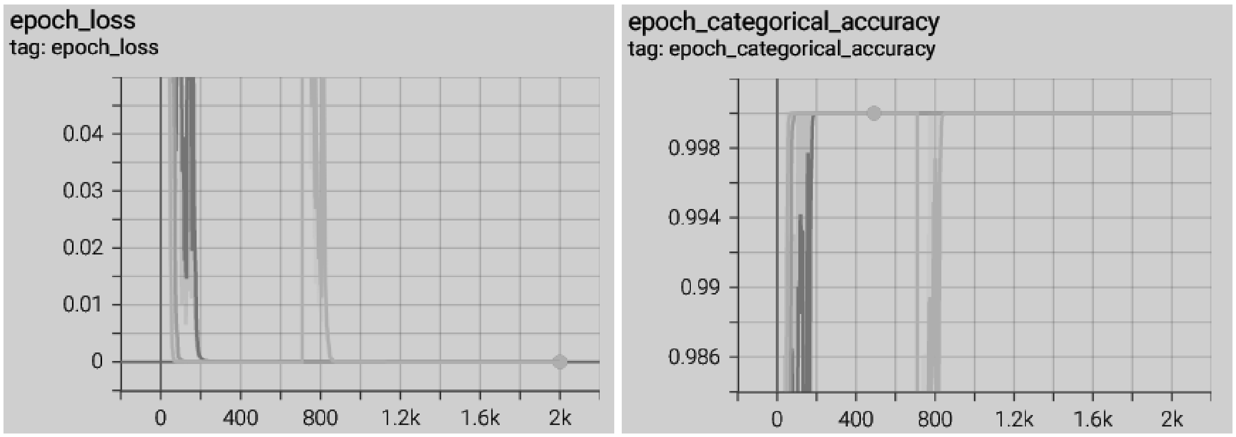


Fig 6.2: Categorical Accuracy & Loss of the three models

In conclusion, the study's findings suggest that all three types of RNNs, including GRU, LSTM, and SimpleRNN, can be effective in recognizing Makaton sign language. SimpleRNN is the most efficient for real-time recognition, which could be an advantage in scenarios where speed is critical. The results of this study have significant implications for the development of practical applications that could benefit individuals with speech and language difficulties.

CHAPTER 7

CONCLUSION

This project describes a proposed model for recognizing Makaton Sign Language (MSL) using a combination of MediaPipe Holistic and a recurrent neural network (RNN). The goal of the model is to accurately recognize three common signs in MSL: Hello, Thank You, and Welcome. MediaPipe Holistic is a framework developed by Google that combines several machine learning models to enable the detection of various elements in a video stream, such as body pose, hand gestures, and facial expressions. By using this framework, the proposed model can extract information about the signer's body pose and hand gestures to recognize the signs.

In addition to MediaPipe Holistic, the proposed model also employs a recurrent neural network (RNN). RNNs are a type of neural network that are well-suited for processing sequential data, making them a natural fit for recognizing sign language, which is a form of sequential communication. The RNN component of the proposed model is trained on the custom dataset consisting of 180 videos of the three signs, allowing it to learn the patterns and nuances of MSL. Overall, the proposed model combines MediaPipe Holistic and an RNN to recognize three common signs in MSL. The use of MediaPipe Holistic enables the extraction of important visual features from the videos, while the RNN component processes this information and recognizes the signs.

The project also describes the findings of a study that proposed a model for recognizing Makaton Sign Language (MSL) using MediaPipe Holistic and recurrent neural networks (RNNs). The study used a custom dataset consisting of 180 videos of three signs, namely Hello, Thank You, and Welcome. The results of the study showed that the GRU-based model outperformed SimpleRNN and LSTM with an accuracy score of 94.4%. However, the study also found that SimpleRNN performed better in real-time scenarios due to its simpler architecture, which had around hundred thousand parameters compared to LSTM and GRU, which had around half a million parameters. The study's contributions included the creation of a custom dataset for MSL recognition,the use of MediaPipe Holistic for landmark extraction, and the evaluation of different types of RNNs for MSL recognition. The study's findings suggest that the use of GRU-based models can lead to improved accuracy in MSL recognition, indicating their effectiveness in this domain. Moreover, the study highlights the importance of considering real-time scenarios and choosing simpler architectures for such applications. The use of SimpleRNN in such scenarios could lead to faster computation times and more efficient use of computational resources.

Our results demonstrated that the system has the potential to improve communication for individuals with communication difficulties, such as those with Autism, Down syndrome, and other developmental disorders. Our findings also suggest that the proposed method could serve as a foundation for developing more advanced sign language recognition systems that can recognize multiple signs in a sentence or continuous conversation.

7.1 FUTURE WORK

There are several avenues for future research to enhance the effectiveness and usefulness of our model for MSL recognition. Firstly, one promising direction is to investigate the use of more complex neural network architectures, such as transformer-based models. Transformer-based models have shown promising results in natural language processing tasks and may be well-suited for the complex language of MSL. The use of transformers would help capture the complex temporal dependencies in sign language gestures, and could potentially improve the accuracy of our model.

Additionally, we plan to explore the integration of other modalities, such as audio recognition, to enhance the model's effectiveness in real-world settings. By combining visual and auditory information, we can create a more comprehensive and robust MSL recognition system. This could enable our model to better recognize and differentiate between similar signs or gestures, as well as improve the overall accuracy of the system.

Furthermore, we plan to expand our dataset to include a larger number of signs and more diverse users to improve the generalizability of our model. This would enable our model to recognize a wider range of signs and gestures, and better accommodate the needs of individuals with diverse communication difficulties. Additionally, we aim to investigate the transferability of our model to different sign languages, including ASL, ISL and BSL, to provide a more comprehensive and versatile solution to sign language recognition.

We also aim to explore the potential for integrating our technology into assistive devices such as smart home systems and communication tools. This could enable individuals with communication difficulties to interact with their environment in a more intuitive and efficient manner, and ultimately improve their quality of life.

Finally, we aim to investigate the development of more advanced algorithms for handling sequential data, such as sequence-to-sequence models. This would enable the recognition of multiple signs in a sentence or continuous conversation, which could have significant implications for improving communication support for individuals with communication difficulties.

In conclusion, there are several promising directions for future research to improve the effectiveness and practicality of MSL recognition systems. These efforts could lead to the development of more advanced and practical sign language recognition systemsthat can improve communication support for individuals with communication difficulties, and ultimately enhance their quality of life.

7.2 RESEARCH ISSUES

There are multiple research issues that occurred in implementing the project, such as:

- Limited availability of data: There is a limited amount of data available for MSL recognition, which makes it difficult to train and evaluate deep learning models effectively.

- Variability in signing styles: Signers may have different styles of signing, which can make it challenging to develop a model that can accurately recognize MSL gestures across different signers.

- Temporal variability: MSL gestures may have different durations and speeds, and the speed of signing may vary between signers. This variability can make it challenging to develop a model that can accurately recognize MSL gestures across different signing speeds.

- Handling multiple signs: MSL sentences may consist of multiple signs that are performed in a sequence. Recognizing multiple signs in a sequence is a challenging task that requires the development of more advanced algorithms.

- Robustness in real-world settings: MSL recognition models need to be robust to noise, lighting conditions, and other environmental factors that can affect the quality of the video stream.

- Integration with other modalities: MSL recognition models may need to be integrated with other modalities, such as audio recognition, to improve the accuracy of recognition in real-world settings.

Addressing these research issues requires the development of new approaches, algorithms, and datasets that can help improve the accuracy and applicability of MSL recognition models.

7.3 IMPLEMENTATION ISSUES

During the implementation of our MSL recognition system, we encountered several technical and logistical issues that required careful consideration and resolution. In this section, we will discuss some of the main challenges we faced and the solutions we implemented to address them.

One of the primary challenges we faced was the collection and annotation of the MSL dataset. The process of recording and annotating the videos was time-consuming and required significant effort from the volunteers. Additionally, we had to ensure that the lighting and background conditions were consistent across all videos to minimise the impact of environmental factors on the recognition performance. To address these challenges, we provided detailed instructions to the volunteers regarding the recording and annotation process and carefully monitored the quality of the collected data.

Another issue we encountered was the extraction of landmarks from the video frames. MediaPipe Holistic provided a reliable way to extract the landmarks, but we had to ensure that the landmarks were accurate and consistent across all videos. We found that slight variations in hand positioning and orientation could lead to differences in the landmark positions, which in turn could affect the recognition performance. To address this issue, we carefully reviewed and corrected the landmark positions manually and implemented a quality control process to ensure that the extracted landmarks were consistent and accurate.

Another challenge we faced was the training of the RNN models. The training process required a significant amount of computing resources, and we had to carefully optimise the hyperparameters to achieve the best performance while avoiding overfitting. We also had to ensure that the models were trained for a sufficient number of epochs to converge to a stable solution. To address these issues, we utilised cloud computing resources to train the models and performed extensive hyperparameter tuning to achieve the best performance.

Finally, we encountered challenges related to real-time recognition performance. We found that the RNN models were computationally intensive and required significant processing power to achieve real-time performance. To address this issue, we optimised the model architectures and explored the use of hardware accelerators, such as GPUs, to improve the processing speed.

In conclusion, the implementation of our MSL recognition system involved several technical and logistical challenges that required careful consideration and resolution. Through careful planning and optimization, we were able to develop a robust and effective system that has the potential to improve communication support for individuals with communication difficulties.

REFERENCES:-

[1] Grishchenko I, Bazarevsky V, MediaPipe Holistic - Simultaneous Face, Hand and Pose Prediction on Device, Google Research, USA, 2020,Access 2021.

[2] Mehdi, Syed Atif, and Yasir Niaz Khan. "Sign language recognition using sensor gloves." In Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02., vol. 5, pp. 2204-2206. IEEE, 2002.

[3] Grobel, Kirsti, and Marcell Assan. "Isolated sign language recognition using hidden Markov models." In 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 1, pp. 162-167. IEEE, 1997.

[3] Abadi M, Barham P, Chen J et.al, Tensorflow: A System for Large-Scale Machine Learning, In 12th USENIX Symposium on Operating System Design and Implementation (OSDI), USA, 2016.

[4] Jangyodsuk, Pat, Christopher Conly, and Vassilis Athitsos. "Sign language recognition using dynamic time warping and hand shape distance based on histogram of oriented gradient features." In Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments, pp. 1-6. 2014.

[5] Yang, Hee-Deok, Stan Sclaroff, and Seong-Whan Lee. "Sign language spotting with a threshold model based on conditional random fields." IEEE transactions on pattern analysis and machine intelligence 31, no. 7 (2008): 1264-1277.

[6] Pigou, Lionel, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. "Sign language recognition using convolutional neural networks." In Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13, pp. 572-578. Springer International Publishing, 2015.

[7] Bantupalli, Kshitij, and Ying Xie. "American sign language recognition using deep learning and computer vision." In 2018 IEEE International Conference on Big Data (Big

Data), pp. 4896-4899. IEEE, 2018.

[8] Liu, Tao, Wengang Zhou, and Houqiang Li. "Sign language recognition with long short-term memory." In 2016 IEEE international conference on image processing(ICIP), pp. 2871-2875. IEEE, 2016.

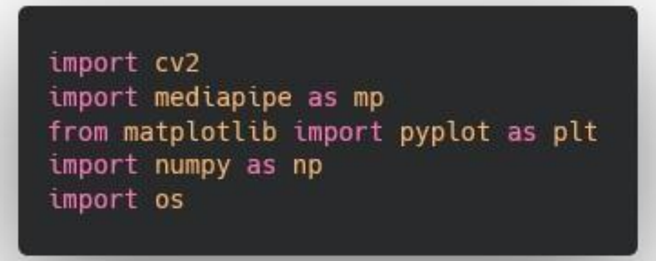
[9] M. Harris and A.S. Agoes, "Applying Hand Gesture Recognition for User Guide Application Using MediaPipe", 2nd International Seminar of Science and Applied Technology (ISSAT 2021), pp. 101-108, 2021, November.

[10] Chandandeep Kaur, Nivit Gill, An Automated System for Indian Sign Language Recognition, International Journal of Advanced Research in Computer Science and Software Engineering.

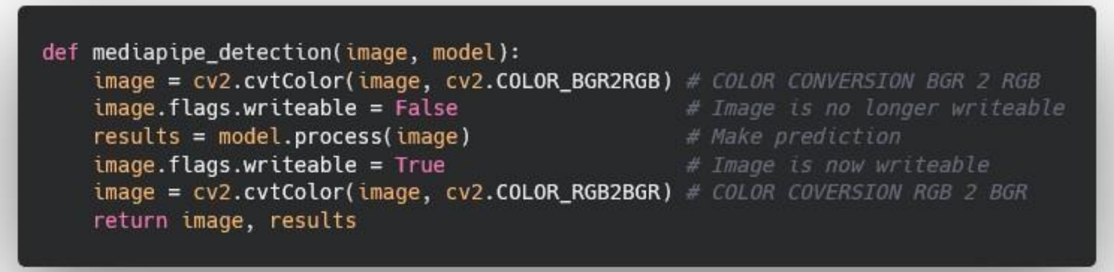
[11] S. Shirbhate1, Mr. Vedant D. Shinde2, Ms. Sanam A. Metkari3, Ms. Pooja U. Borkar4, Ms. Mayuri A. Khandge/Sign-Language- Recognition-System.2020 IRJET Vol3 March,2020.

APPENDIX:-

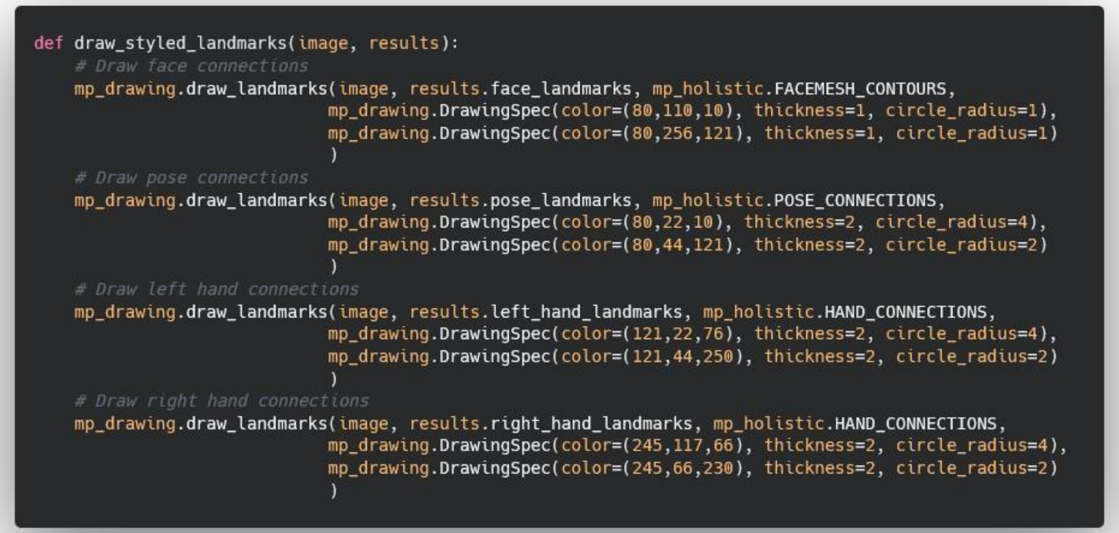
A. SOURCE CODE



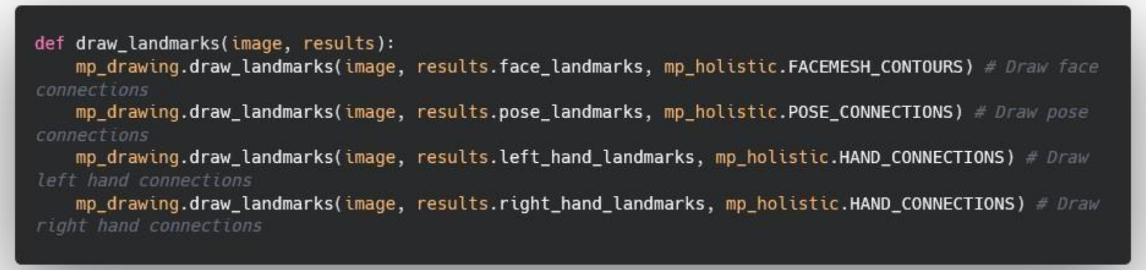
```python
import cv2
import mediapipe as mp
from matplotlib import pyplot as plt
import numpy as np
import os
```



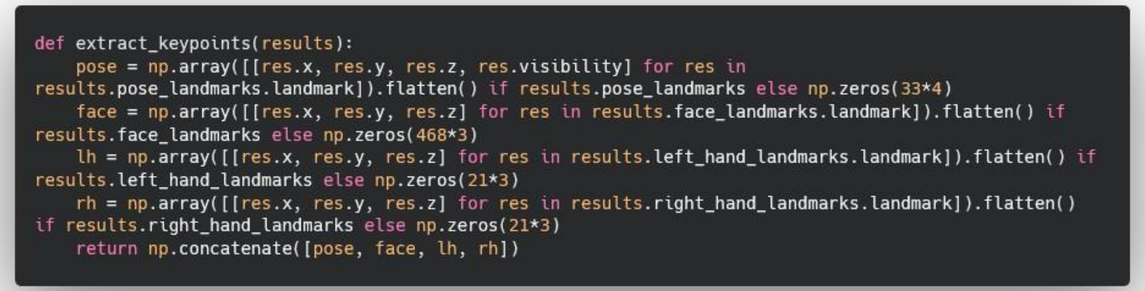
```python
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    image.flags.writeable = False                  # Image is no longer writeable
    results = model.process(image)                 # Make prediction
    image.flags.writeable = True                   # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR COVERSION RGB 2 BGR
    return image, results
```



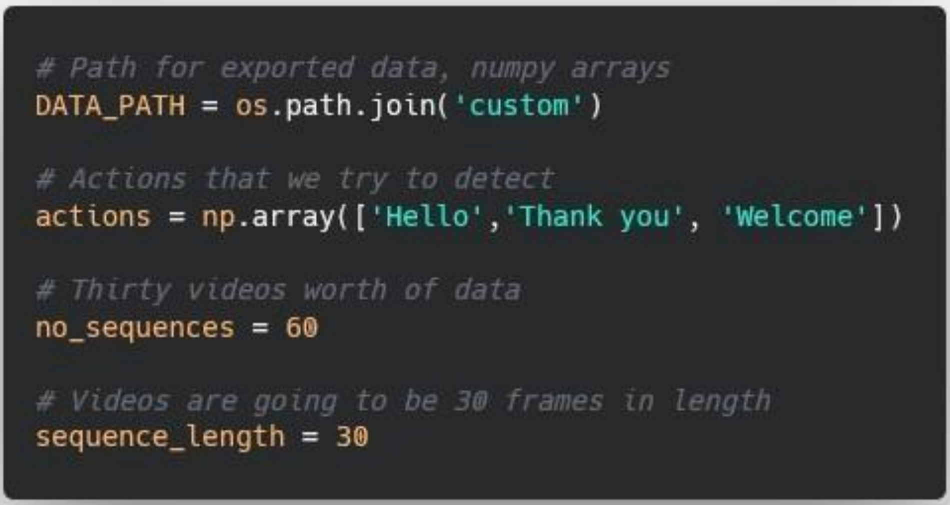
```python
def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS,
                             mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                             mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                             )
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                             mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                             mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                             )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                             mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                             mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                             )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                             mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                             mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                             )
```



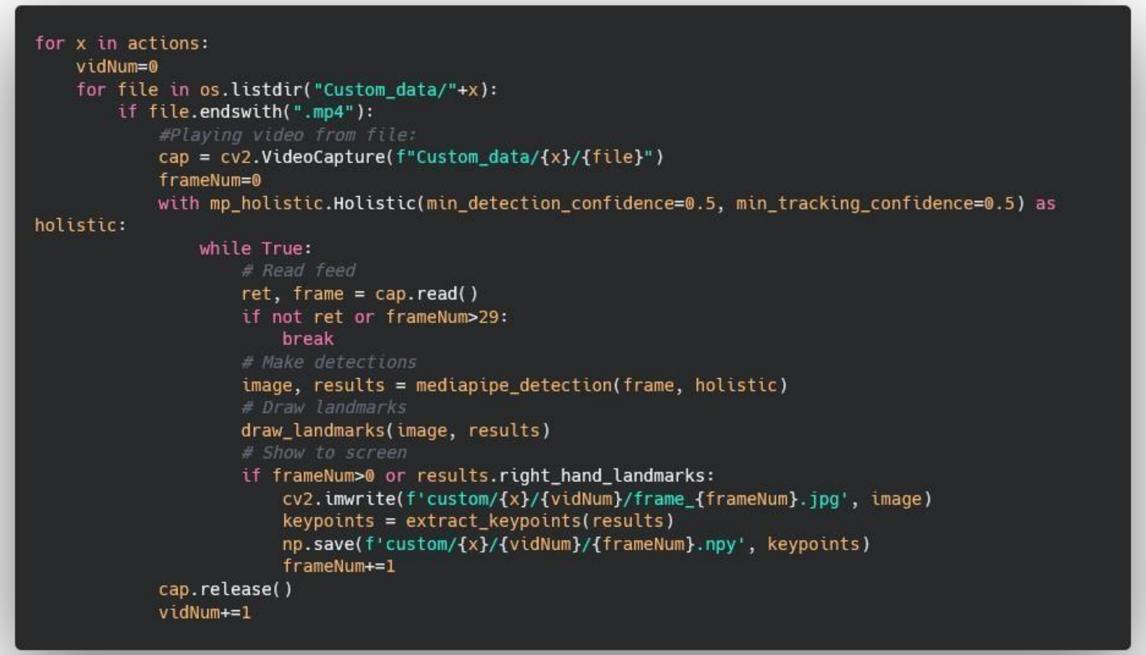
```
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS) # Draw face
connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS) # Draw pose
connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw
left hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw
right hand connections
```



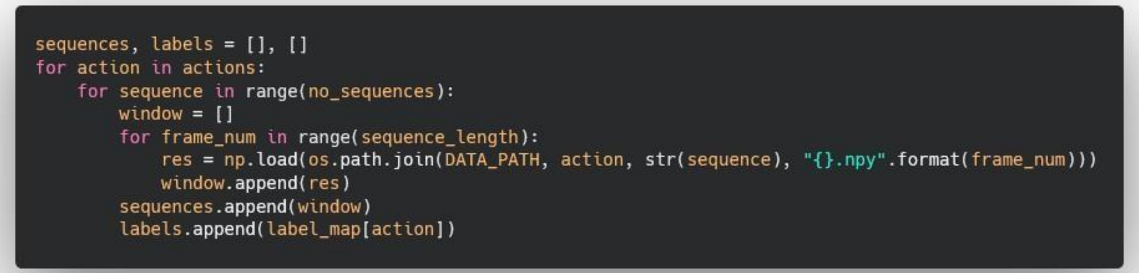
```
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in
results.pose_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros(33*4)
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if
results.face_landmarks else np.zeros(468*3)
    lh = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() if
results.left_hand_landmarks else np.zeros(21*3)
    rh = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten()
if results.right_hand_landmarks else np.zeros(21*3)
    return np.concatenate([pose, face, lh, rh])
```



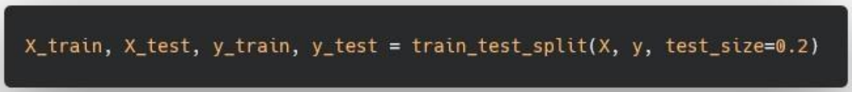
```
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('custom')

# Actions that we try to detect
actions = np.array(['Hello','Thank you', 'Welcome'])

# Thirty videos worth of data
no_sequences = 60

# Videos are going to be 30 frames in length
sequence_length = 30
```



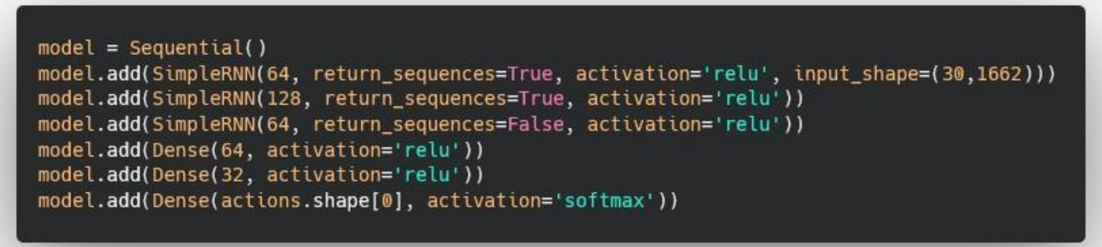
```
for x in actions:
    vidNum=0
    for file in os.listdir("Custom_data/"+x):
        if file.endswith(".mp4"):
            #Playing video from file:
            cap = cv2.VideoCapture(f"Custom_data/{x}/{file}")
            frameNum=0
            with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as 
holistic:
                while True:
                    # Read feed
                    ret, frame = cap.read()
                    if not ret or frameNum>29:
                        break
                    # Make detections
                    image, results = mediapipe_detection(frame, holistic)
                    # Draw landmarks
                    draw_landmarks(image, results)
                    # Show to screen
                    if frameNum>0 or results.right_hand_landmarks:
                        cv2.imwrite(f'custom/{x}/{vidNum}/frame_{frameNum}.jpg', image)
                        keypoints = extract_keypoints(results)
                        np.save(f'custom/{x}/{vidNum}/{frameNum}.npy', keypoints)
                        frameNum+=1
            cap.release()
            vidNum+=1
```



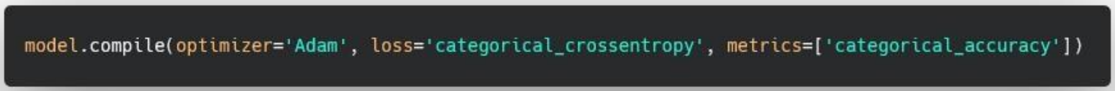
```
sequences, labels = [], []
for action in actions:
    for sequence in range(no_sequences):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```



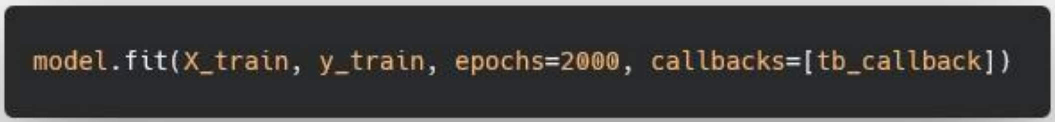
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```



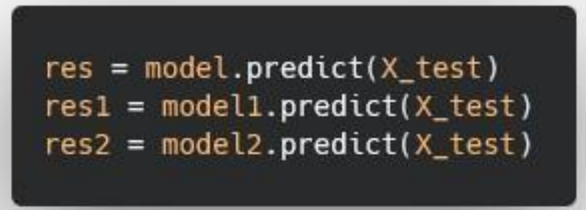
```
model = Sequential()
model.add(SimpleRNN(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(SimpleRNN(128, return_sequences=True, activation='relu'))
model.add(SimpleRNN(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```



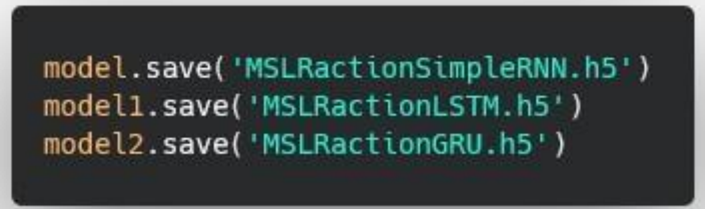
```python
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
```



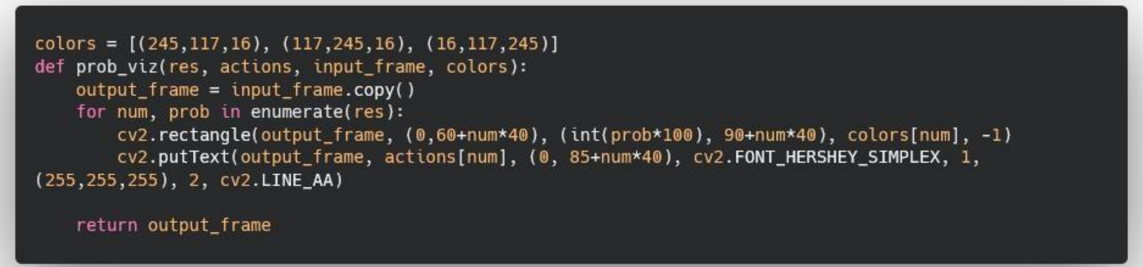
```python
model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])
```



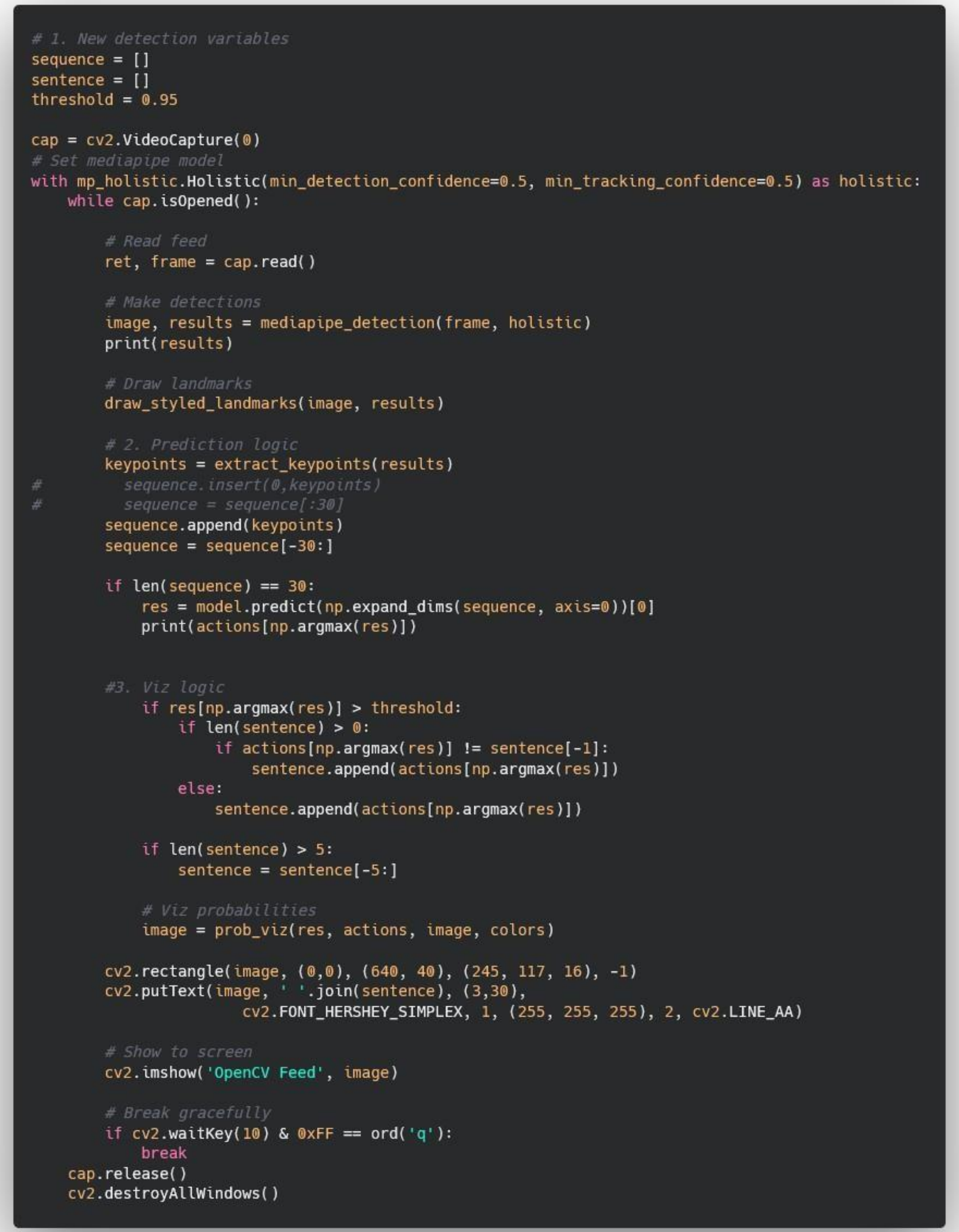
```python
res = model.predict(X_test)
res1 = model1.predict(X_test)
res2 = model2.predict(X_test)
```



```python
model.save('MSLRactionSimpleRNN.h5')
model1.save('MSLRactionLSTM.h5')
model2.save('MSLRactionGRU.h5')
```

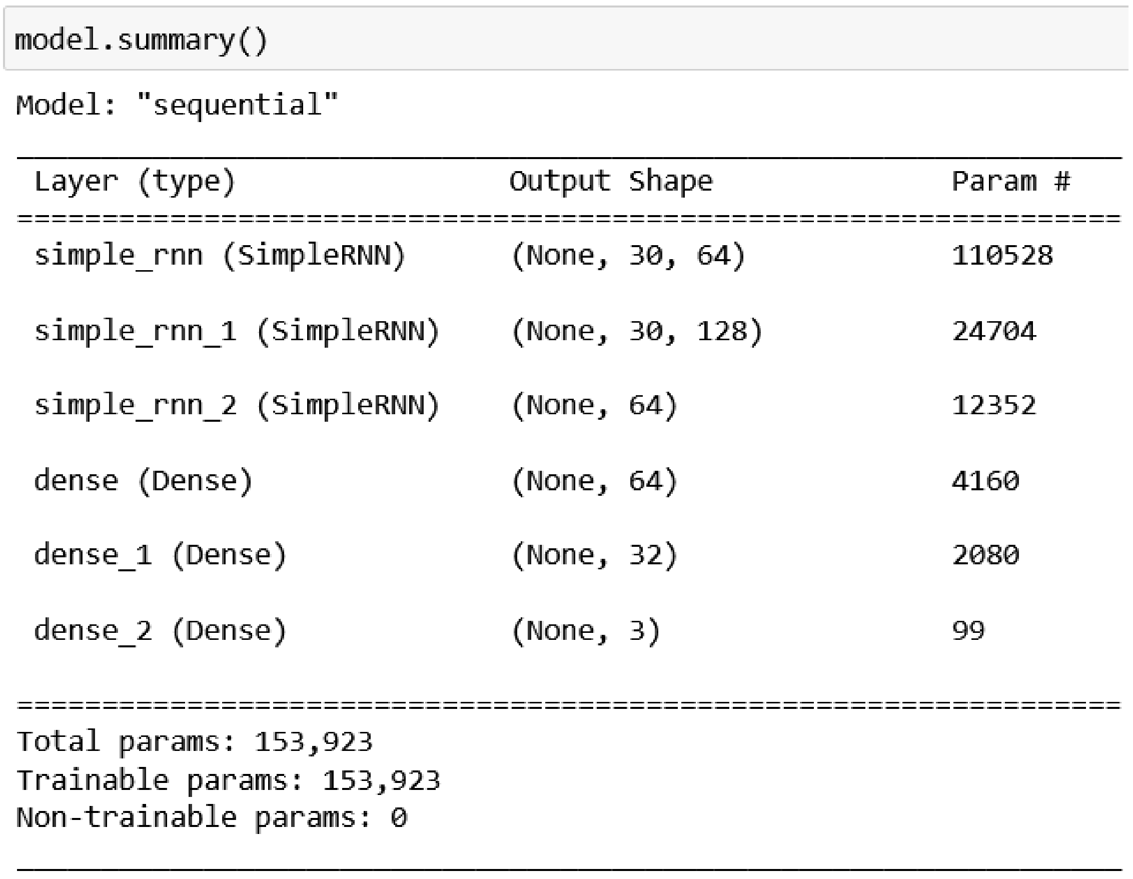


```python
colors = [(245,117,16), (117,245,16), (16,117,245)]
def prob_viz(res, actions, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40), colors[num], -1)
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255,255,255), 2, cv2.LINE_AA)

    return output_frame
```



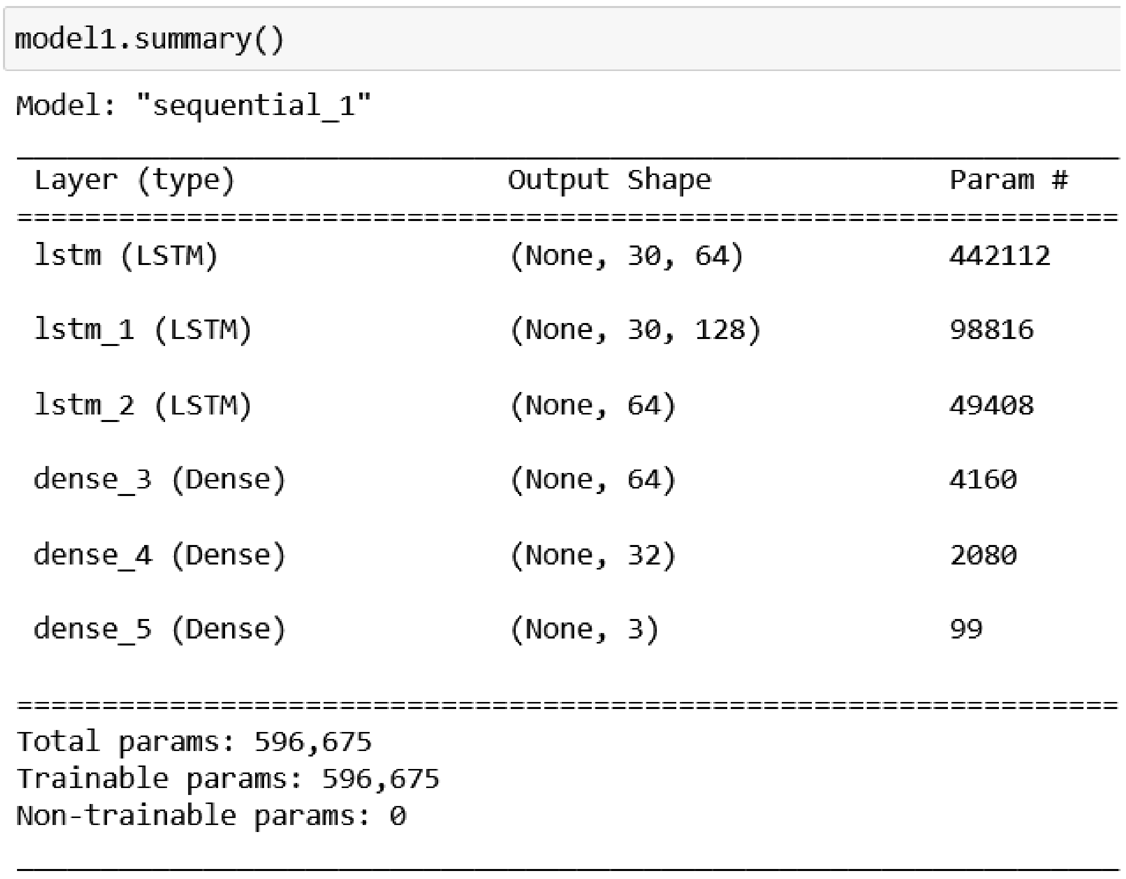
```python
# 1. New detection variables
sequence = []
sentence = []
threshold = 0.95

cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        draw_styled_landmarks(image, results)

        # 2. Prediction logic
        keypoints = extract_keypoints(results)
#         sequence.insert(0,keypoints)
#         sequence = sequence[:30]
        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)])


        #3. Viz logic
            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:]

            # Viz probabilities
            image = prob_viz(res, actions, image, colors)

        cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
        cv2.putText(image, ' '.join(sentence), (3,30),
                       cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

B. SCREENSHOTS



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 30, 64)            110528

 simple_rnn_1 (SimpleRNN)    (None, 30, 128)           24704

 simple_rnn_2 (SimpleRNN)    (None, 64)                12352

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 3)                 99

=================================================================
Total params: 153,923
Trainable params: 153,923
Non-trainable params: 0
_________________________________________________________________
```

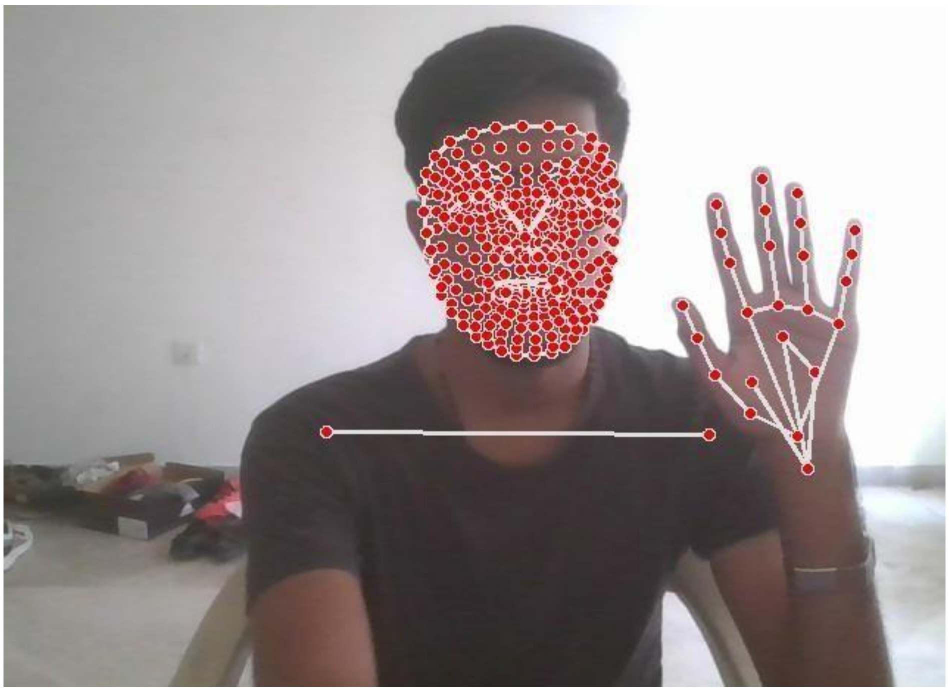


```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 64)            442112

 lstm_1 (LSTM)               (None, 30, 128)           98816

 lstm_2 (LSTM)               (None, 64)                49408

 dense_3 (Dense)             (None, 64)                4160

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 3)                 99

=================================================================
Total params: 596,675
Trainable params: 596,675
Non-trainable params: 0
_________________________________________________________________
```

```
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 30, 64)            331776

 gru_1 (GRU)                 (None, 30, 128)           74496

 gru_2 (GRU)                 (None, 64)                37248

 dense_6 (Dense)             (None, 64)                4160

 dense_7 (Dense)             (None, 32)                2080

 dense_8 (Dense)             (None, 3)                 99

=================================================================
Total params: 449,859
Trainable params: 449,859
Non-trainable params: 0
_________________________________________________________________
```



C. RESEARCH PAPER

Makaton Sign Language Recognition

ROHAN OUSEPHACHEN THAYIL
STUDENT
DEPARTMENT OF CSE
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
CHENNAI
rohanousep@gmail.com

NIKHIL R PRINCE
STUDENT
DEPARTMENT OF CSE
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
CHENNAI
np31121@gmail.com

DR.M.D.ANTO PRAVEENA
ASSISTANT PROFESSOR
DEPARTMENT OF CSE
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
CHENNAI
antopraveena.cse@sathyabama.ac.in

***Abstract*--- Makaton is a widely used sign language system that is primarily used by individuals with communication difficulties, such as those with Autism, Down syndrome, and other developmental disorders. However, despite its widespread usage, there is a lack of technology for accurately recognizing and understanding Makaton signs. This paper proposes a model that combines MediaPipe Holistic with a neural network, such as SimpleRNN, LSTM, or GRU, to recognize Makaton sign language (MSL). The model is trained on a custom dataset of Makaton sign language videos using backpropagation and the Adam optimizer. The proposed network achieved a high accuracy on the test set, demonstrating its effectiveness in recognizing Makaton sign language. This study introduces an innovative method for advancing the development of systems that recognize sign language by utilizing MediaPipe Holistic in conjunction with a neural network. This can improvethe way technology interacts with people who have hearing impairments.**

Keywords*-- *Makaton Sign Language, Recurrent neural networks (RNNs), MediaPipe Holistic, Long Short-Term Memory (LSTM), Sign Language Recognition, Gated Recurrent Unit (GRU).

I. INTRODUCTION

Makaton is a system of communication that uses a combination of speech, signs and symbols to help people to communicate. It is designed for use by people with communication difficulties, including those with autism, learning disabilities, and hearing loss. Advancements in technology that can interpret MSL gestures can enhance the lives of individuals with disabilities by facilitating more effective communication. Nevertheless, recognizing MSL gestures presents a significant challenge due to variations in gesture execution and the intricacy of the MSL language. To address this challenge, recurrent neural networks (RNNs) have been employed as a solution to handle sequential data. LSTM and GRU networks are specialized types of RNNs that are specifically designed to address long-term dependencies in sequential data. These networks have been widely adopted across various domains, such as speech recognition, gesture recognition, and NLP, due to their remarkable success in these applications.

In this paper, we present a comparison of the performance of three different types of RNNs - SimpleRNN, LSTM, and GRU (Gated Recurrent Unit) - for recognizing Makaton signs. We propose a method that involves preprocessing video data toextract features such as the position and movement of the hands and fingers. These features are then used to train each of the three RNNs for recognition purposes. Our findings provide insights into the potential of using different RNN architectures for sign language recognition.

II. LITERATURE SURVEY

For many years, researchers in machine learning and computer vision have been intrigued by the subject of sign language recognition. Early attempts to recognize sign language involved the use of sensor gloves for capturing the motion of hand and finger gestures and recognizing them using traditional machine learning algorithms [2]. But these methods were often expensive and cumbersome. More recent approaches have used computer vision techniques to extract features from video recordings of sign language gestures.

Initial attempts in this field concentrated on utilizing Hidden Markov Models (HMMs) to identify individual signs. Grobeland Assan [3] introduced an HMM-based system for recognizing isolated signs, which showed encouraging outcomes. However, this approach has some limitations, such as its sensitivity to noise and variations in the signing styles.

Subsequently, researchers started investigating other methodsfor recognizing sign language, including Dynamic Time Warping (DTW) and Histogram of Oriented Gradient (HOG) features. Jangyodsuk and colleagues [4] introduced a DTW and HOG-based system for recognizing sign language that attained high precision. However, this technique is computationally demanding and may not be appropriate for real-time use. Another prevalent approach is employing Conditional Random Fields (CRFs) to spot sign language. Yang and colleagues [5] introduced a CRF-based threshold model for spotting sign language that achieved 87% accuracy.

Conventional approaches to sign language recognition such as Dynamic Time Warping (DTW) and Hidden Markov Models (HMMs) algorithms have shown success in recognizing isolated signs but have limited performance when recognizing continuous sign language. Fortunately, with the recent advancements in computer vision and deep learning, there have been significant improvements in the accuracy of sign language recognition.

One approach that has shown promising results is the use of CNNs to recognize sign language gestures. Pigou and colleagues [6] introduced a CNN-based method for recognizing sign language that attained high precision rates on benchmark datasets. The architecture of their model included two separate CNNs, one responsible for extracting hand features and the other for extracting upper body features. Both CNNs were designed with three layers, and achieved a recognition accuracy of 95.68% on the CLAP14 dataset.

Another prevalent method is employing recurrent neural networks (RNNs) to recognize temporal patterns in sign language gestures. Bantupalli and Xie [7] used a combination ofcomputer vision and deep learning techniques to identify ASL gestures. They proposed a model that uses a combination of a CNN and an RNN network to recognize 150 ASL gestures. Their model achieved an accuracy of 90% on the American Sign Language Dataset.

Compared to the CNN-RNN model, the proposed model that combines MediaPipe Holistic with a neural network (SimpleRNN, LSTM or GRU) offers several advantages. Firstly, it requires less data to produce a highly accurate model. Additionally, the proposed model has a simpler neural network architecture, with approximately half a million parameters, which results in faster training times. Furthermore, as the neural network is simpler, it enables real-time detection with faster processing speeds. Unlike the CNN-RNN model, which has around 30 to 40 million parameters, the proposed model offers a more efficient solution for sign language recognition, making it a promising tool for practical applications.

Overall, deep learning techniques have shown great promise in enhancing the precision of systems that recognize sign language. However, there is still ongoing research in this area, with new models and datasets being proposed regularly.

III. SYSTEM ARCHITECTURE

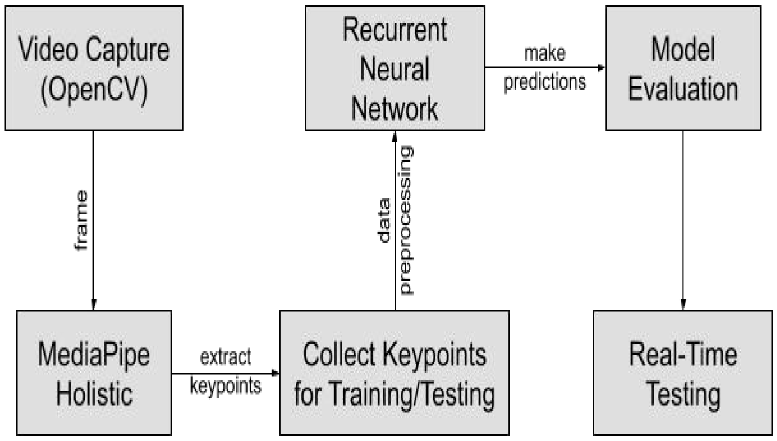


Figure 1: System Architecture for Makaton Sign Language Recognition

IV. METHODOLOGY

The proposed method for Makaton Sign Language Recognition (MSLR) using SimpleRNN, LSTM, and GRU networks involves the following steps:

1. Data collection: To recognize Makaton signs, we first collected a dataset by recording videos of individuals signing Makaton signs. The dataset included a range of signs and variations in signing style. These videos were labeled with the corresponding text or speech, so that the networks could learn to associate different signs with different words or phrases.

2. Data pre-processing: The collected videos were pre-processed to extract the necessary information for the analysis. This included cropping the video to focus on the signing hand and converting the video to a sequence of images.

3. Feature extraction: A collection of features was obtained from the pre-processed data to represent the signing gestures. These features encompassed the hand's position, movement, and shape.

4. Network training: SimpleRNN, LSTM, and GRU networks were separately trained on the extracted features using backpropagation and the Adamoptimizer. Each network consisted of several layersthat culminated in a fully connected layer. The

networks were trained to classify the signing gestures into different classes, such as different Makaton signs.

5. Evaluation: The trained networks were evaluated on a separate test dataset to measure their performance in recognizing and classifying Makaton signs. The performance was measured using metrics such as accuracy score.

6. Sign language recognition: Once the networks are trained and evaluated, they can be employed for real-time recognition of Makaton Sign Language by processing the video input and recognizing the signs.

This approach aims to improve the accuracy and robustness of MSLR systems by leveraging the ability of SimpleRNN, LSTM, and GRU to handle sequential data and handle variations in signing style. Additionally, the use of hand movement and shape information in feature extraction will provide more robustness against variations in signing style.

V. EXPERIMENTS

To assess the effectiveness of our suggested models for Makaton Sign Language (MSL) recognition, we conducted several experiments using a custom dataset consisting of 180 videos of 3 signs (Hello, Thank You, Welcome). The datasetwas created with the help of 3 volunteers, each sign having 60 videos of 30 frames each. Figure 2 illustrates that MediaPipe Holistic was utilized to produce a total of 543 landmarks, with
33 landmarks for pose estimation, 468 landmarks for facial surface, and 21 landmarks for each hand.

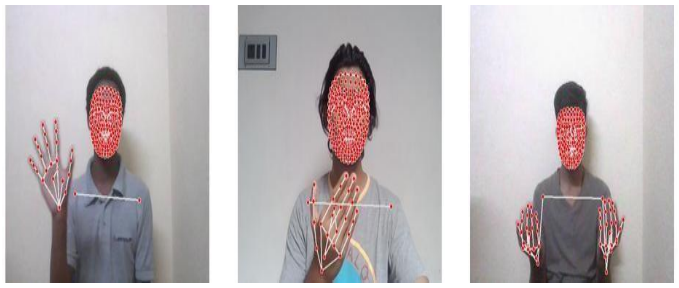


Figure 2: MediaPipe Holistic in action

We collected the keypoint values (landmarks) from the series of frames and created NumPy arrays consisting of 1662 landmarks (33*4+468*3+21*3+21*3) for each frame in each video. The dataset used in this study was divided into two sets: a training set and a testing set. The ratio used for the split was 80:20, with 36 videos reserved for the testing set. The input shape for the first layer of the sequential model was (30, 1662) as shown in figure 3.

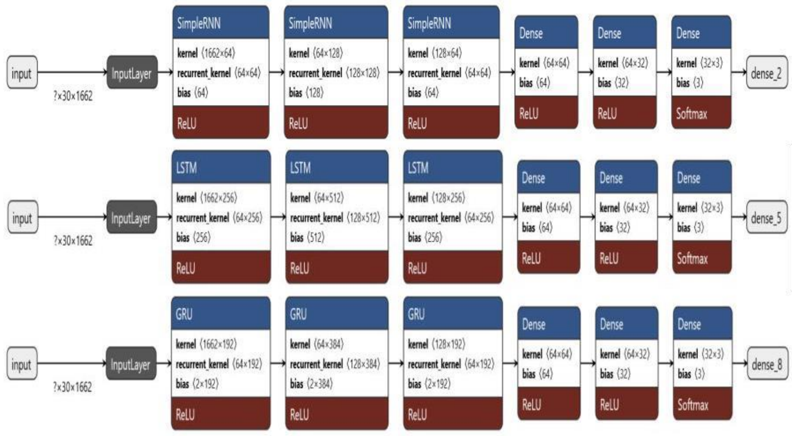


Figure 3: Architectures of SimpleRNN, LSTM & GRU

We experimented with three different types of recurrent neural networks (RNNs): SimpleRNN, LSTM, and GRU. For eachtype of RNN, we used a three-layer architecture with 64 hidden units in the first and third layers and 128 hidden units in the second layer. The architectures were followed by 2 dense ReLU layers, the first one having 64 hidden layers and the second one with 32 hidden layers. Figure 3 depicts the utilization of a softmax layer atop the RNNs for predicting the sign label. The Adam optimizer was used to train the networks for 2000 epochs,with a batch size of 64. Our proposed method's performance was evaluated by varying several parameters, such as the number of RNN layers, the number of neurons in each RNN layer, and the batch size. Finally, the proposed method was tested using real- time video of individuals signing Makaton signs to verify its ability to recognize Makaton signs in real-time.

VI. RESULTS

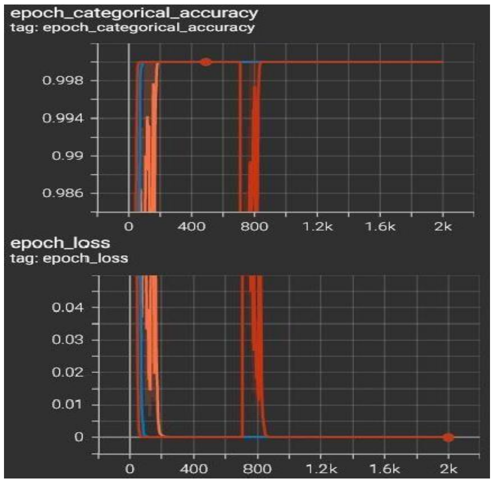


Figure 4: Categorical Accuracy and Loss of the three models

Our results showed that the GRU model achieved the highest accuracy of 94.4%, followed by SimpleRNN and LSTM, which both achieved an accuracy of 91.6%. However, SimpleRNN was found to be the most efficient for real-time recognition as it has a significantly smaller number of parameters (around 150,000) compared to LSTM & GRU (around half a million parameters). Despite this, all three types of RNNs demonstrated high accuracy in recognizing Makaton sign language, indicating their effectiveness in this domain. These results suggest that the proposed methods have great potential for practical applications, especially in real-time scenarios where SimpleRNN could be an optimal choice.

VII. CONCLUSION

In our study, we introduced a new method for recognizing MSL that combines MediaPipe Holistic and an RNN. To assess the performance of our proposed approach, we conducted a series of experiments using a custom dataset containing 180 videos of three signs (Hello, Thank You, and Welcome).

Our experiments showed that the GRU-based model performed better than SimpleRNN and LSTM, achieving 94.4% accuracy. However, we also found that SimpleRNN performed better in real-time scenarios because of its simpler architecture (having around hundred thousand parameters as compared to LSTM and GRU, which had around half a million parameters).

Our contributions include the creation of a custom dataset for MSL recognition, the use of MediaPipe Holistic for landmark extraction, and the evaluation of different types of RNNs for MSL recognition. Our findings suggest that the use of GRU- based models can lead to improved accuracy in MSL recognition, but SimpleRNN can be a better choice for real-timeapplications due to its simpler architecture.

Our results demonstrated that the system has the potential to improve communication for individuals with communication difficulties, such as those with Autism, Down syndrome, and other developmental disorders.Our research indicates that the approach we proposed has the potential to be a basis for creating more sophisticated SLR systems capable of recognizing multiple signs in a sentence or during a continuous conversation.

VIII. FUTURE WORK

In order to enhance the effectiveness and usefulness of ourmodel for MSL recognition, there are several avenues for future research that could be pursued. One potential direction is to investigate the use of more complex neural network architectures, such as transformer-based models, to capture the complex temporal dependencies in sign language gestures. Transformer-based models have shown promising results in natural language processing tasks and may be well-suited for thecomplex language of MSL. Additionally, we plan to explore the integration of other modalities, such as audio recognition, to enhance the model's effectiveness in real-world settings.

Furthermore, we plan to expand our dataset to include a larger number of signs and more diverse users to improve the generalizability of our model. We also aim to investigate the transferability of our model to different sign languages, including ASL, ISL and BSL, and explore the potential forintegrating our technology into assistive devices such as smart home systems and communication tools.

Finally, we aim to investigate the development of more advanced algorithms for handling sequential data, such as sequence-to-sequence models, to enable the recognition of multiple signs in a sentence or continuous conversation. These efforts could lead to the development of more advanced and practical sign language recognition systems that can improve communication support for individuals with communication difficulties.

REFERENCES

[1] Grishchenko I, Bazarevsky V, MediaPipe Holistic – Simultaneous Face, Hand and Pose Prediction on Device, Google Research, USA, 2020,Access 2021.

[2] Mehdi, Syed Atif, and Yasir Niaz Khan. "Sign language recognition using sensor gloves." In Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02., vol. 5, pp. 2204-2206. IEEE, 2002.

[3] Grobel, Kirsti, and Marcell Assan. "Isolated sign language recognition using hidden Markov models." In 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 1, pp. 162-167. IEEE, 1997.

[4] Jangyodsuk, Pat, Christopher Conly, and Vassilis Athitsos. "Sign language recognition using dynamic time warping and hand shape distance based on histogram of oriented gradient features." In Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments, pp. 1-6.2014.

[5] Yang, Hee-Deok, Stan Sclaroff, and Seong-Whan Lee. "Sign language spotting with a threshold model based on conditional random fields." IEEE transactions on pattern analysis and machine intelligence 31, no. 7 (2008): 1264-1277.

[6] Pigou, Lionel, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. "Sign language recognition using

convolutional neural networks." In Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13, pp. 572-578. Springer International Publishing, 2015.

[7] Bantupalli, Kshitij, and Ying Xie. "American sign language recognition using deep learning and computer vision." In 2018 IEEE International Conference on Big Data (Big Data), pp. 4896- 4899. IEEE, 2018.

[8] Liu, Tao, Wengang Zhou, and Houqiang Li. "Sign language recognition with long short-term memory." In 2016 IEEE international conference on image processing (ICIP), pp. 2871-2875. IEEE, 2016.

[9] M. Harris and A.S. Agoes, "Applying Hand Gesture Recognition for User Guide Application Using MediaPipe", 2nd International Seminar of Science and Applied Technology (ISSAT 2021), pp. 101-108, 2021, November.