

MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORKS (R-CNN) FOR SINHALA SIGN LANGUAGE TO TEXT CONVERSION

R. D. Rusiru Sewwantha¹ and T. N. D. S. Ginige²

¹Universal College Lanka, University of Central Lancashire,
Colombo, Sri Lanka

²Universal College Lanka, Colombo, Sri Lanka

ABSTRACT

Sign Language is the use of various gestures and symbols for communication. It is mainly used by disabled people with communication difficulties due to their speech or hearing impediments. Due to the lack of knowledge on sign language, natural language speakers like us, are not able to communicate with such people. As a result, a communication gap is created between sign language users and natural language speakers. It should also be noted that sign language differs from country to country. With American sign language being the most commonly used, in Sri Lanka, we use Sri Lankan/Sinhala sign language. In this research, the authors propose a mobile solution using a Region Based Convolutional Neural Network for object detection to reduce the communication gap between the sign users and language speakers by identifying and interpreting Sinhala sign language to Sinhala text using Natural Language Processing (NLP). The system is able to identify and interpret still gesture signs in real-time using the trained model. The proposed solution uses object detection for the identification of the signs.

KEYWORDS

Object detection, TensorFlow, Mask R-CNN, Sinhala, Sign Language, VideoCapture.

1. INTRODUCTION

Essentially, the use of gestures and symbols for communication is known as sign language. World health organization has revealed that over 5% of the world population (430 million people) require rehabilitation to address their ‘disabling’ hearing loss, it was also mentioned that by the year 2050, over 700 million people will have a disabling hearing loss [1]. In 2020, it was stated by the World Federation of the Deaf that there are around 72 million people worldwide who uses sign language for communication [2]. According to the World health organization, approximately 9% of the Sri Lankan population has loss of hearing [3] and from a study conducted by the Sri Lanka Federation of the Deaf, more than 300,000 people are deaf. Unfortunately, there are not many sign language interpreters in Sri Lanka. It was mentioned that there were only 6 sign interpreters in Sri Lanka [4]. Due to this, there is a huge demand for sign interpreters in our country. Even though there have been many proposed systems to tackle this problem, a proper system has not been deployed yet. Therefore, the goal of this project is to develop a mobile Sinhala sign language interpreter which can be easily used by language speakers or disabled people to communicate.

The objective and the core functionality of the system is that it should be able to successfully interpret inputted Sinhala signs into Sinhala text. To cater this requirement, the following features are implemented in the prototype.

- Real-time sign interpretation using video capture and video upload from gallery.
- Sign language dictionary.
- Trained mask rcnn inception resnet model for object detection.

The purpose of the sign dictionary is to make sure that the language speakers are able to learn the basic signs of Sinhala sign language and to inform users on what signs the system can interpret in the current version.

2. LITERATURE REVIEW

Sign language interpretation can be identified as gesture recognition. Since gesture recognition has been an area of interest for many years, several methods have been employed [5]. These methods can be categorized into two as data glove method and vision-based method [5].



Figure 1. Data glove method & Vision based method [6].

2.1. Data Glove Methods

Data glove method employs mechanical or optical sensors attached to a glove that transforms finger flexions into electrical signals to determine the hand posture [6]. This approach is not a mobile solution.

Using the data glove method, a Chinese sign language recognition system was built based on ARM9 [7]. It also uses combined flex sensors with 9-axis IMU sensor. The sensor modules can measure both the bending degree of the fingers and the angle of the palm. These readings are sent to the ARM9 processor which will analyse and process the data in real-time. Finally, using a voice broadcast module and the text display module, the interpreted text and voice are displayed and broadcasted.

A data glove with gyro-sensor was introduced for Japanese sign language [8]. The glove was developed to have 5 sensors for the fingers and on the back of the palm, an accelerometer with gyro-sensor, was installed. These data are sent through an analog-to-digital converter and then to the data control unit. From this the data is sent to a computer via a USB cable. This also consists of a palm turning motion algorithm.

2.2. Vision-Based Methods

Comparatively, vision-based approach uses cameras to input images [6]. The gestures used to create the gesture database should be selected with their relevant meaning and each gesture may contain multiple samples for increasing the accuracy of the system.

Nath & Arun, 2017 proposes a sign language interpreter implemented in ARM CORTEX A8 processor board using convex hull algorithm and template matching algorithm. A webcam is used to feed images into the system and these images are converted into text. Numbers are recognized by the convex hull method and alphabets are recognized by the template matching method. This system consists of several physical components (camera, display device, processor, etc.).

An interpreter for American sign language developed by Ahmed et. al. (2016) using Microsoft's Kinect V2's Continuous Gesture Builder for gesture recognition and training has two modules, speech to sign and sign to speech. Speech to sign is done using an external library for speech to text conversion by taking a sentence or a word as input. Keywords are identified from the input and compared against keywords in the database to identify the gesture against the keyword. These gestures are mapped with the animations to a 3d model using a data structure in Unity3D. For the sign to speech module, the gestures are inputted frame by frame using the Kinect sensor and matched it with the pre-stored gestures in the database. By using these keywords, a sentence is constructed and is converted into speech by text to speech libraries provided by .Net.

A Sinhala sign language framework is proposed by Madushanka et. al. (2016) using a wearable armband. This research is an extension of Surface Electromyography (sEMG) - used to gather gestural data and Inertial Measurement units (IMU) - (Accelerometer gyroscope and Magnetometer) are used for spatial data. For this system, a device called "Myo Gesture Recognition Armband" (developed by Thalmic Labs Inc.) has been used. This device is Bluetooth compatible and uses sEMG as the main gesture recognition technology. It also has a combination of sEMG, accelerometer, gyroscope, and magnetometer sensors. For gestural references, sEMG data has been used and for spatial references accelerometer, gyroscope and magnetometer has been used. For gesture recognition, a supervised machine learning technique has been used.

Dissanayake et. al. (2020) proposed a mobile app for Sinhala sign language interpretation using image processing and machine learning called "Utalk". This solution converts signs from videos to Sinhala text. This also can interpret both static and dynamic signs. A signing video is taken as the input to the system and then the frame segments are extracted from it so that the using image processing techniques, the background of these frames can be removed. Then these pre-processed images are fed into two separate machine learning models as static sign classifier and dynamic sign classifier by classifying them as static and dynamic signs. These are fed into the language model which is used to generate text based on the input video.

The above mentioned are the previous work conducted on sign language interpretation. Most of them have been proposed for American sign language and almost all solutions contain hardware components. Having hardware components are cumbersome, resulting in inability to use the products in the day-to-day life. However, the proposed solution by the authors of this research, is a mobile application. Therefore, it can be easily used by anyone with a smartphone. It is also based on Sinhala sign language which is the sign language used in Sri Lanka. Currently in Sri Lanka, a proper usable solution has not been implemented.

3. METHODOLOGY

3.1. Data Gathering

The image dataset prepared is the most important requirement in a data science project. No proper datasets were found for the Sinhala sign language to download from internet. Therefore, images were collected from several people posing the signs required to train the model. When

capturing the images, they were advised to take each sign from several angles. This is to make sure that the model is trained to identify signs from different angles. Following tables display which signs were used to train the model.

Table 1. Number Signs

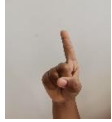






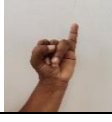

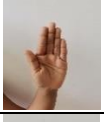

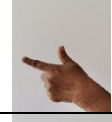



Value	Sign	Value	Sign	Value	Sign
1		4		7	
2		5		8	
3		6		9	

Table 2. Alphabet Signs

Value	Sign	Value	Sign	Value	Sign
අ		ඉ		උ	
ආ		එ		ඌ	

3.2. Technology Selection

3.2.1. Development Framework

From the choice between Django and Flask frameworks, Flask was selected to develop the backend Rest API for the sign interpretation system. Reasons for the selection of Flask framework instead of Django was due to its lightweight property compared to Django. Furthermore, using Flask to create HTTP services was easier. Also, for a Python beginner, Flask framework is easier to learn than Django which is rich in features.

Since the frontend is a mobile application, to make the code reusable, Ionic Framework was decided to be used as it is a cross-platform framework. Main reason for selecting this was that, using the same implementation, the application can be easily deployed on both Android and iOS or as a web application. Another reason to select this was that the authors had prior knowledge on Ionic framework.

3.2.2. Machine Learning Library

As the machine learning library, Tensorflow 2 was selected. This was used to implement the object detection of the system, which is the core functionality of the sign interpretation system, which is the sign identification. It is also possible to use the Keras library with the Tensorflow library.

3.3. Pre-Processing

After the images were collected for the dataset, identical images were removed to reduce the overfitting of the model when training. Then all the images were resized to the dimensions 900x1024. Afterwards, in the ratio 0.2 to 0.8, the dataset was divided into two as test and train, respectively. These divided images were annotated to train the mask R-CNN model, which will also increase the accuracy of the training process, outlining the signs. Each sign image was annotated and labelled as its sign text value using the annotation tool LabelMe. Figure 8 shows how this was done.

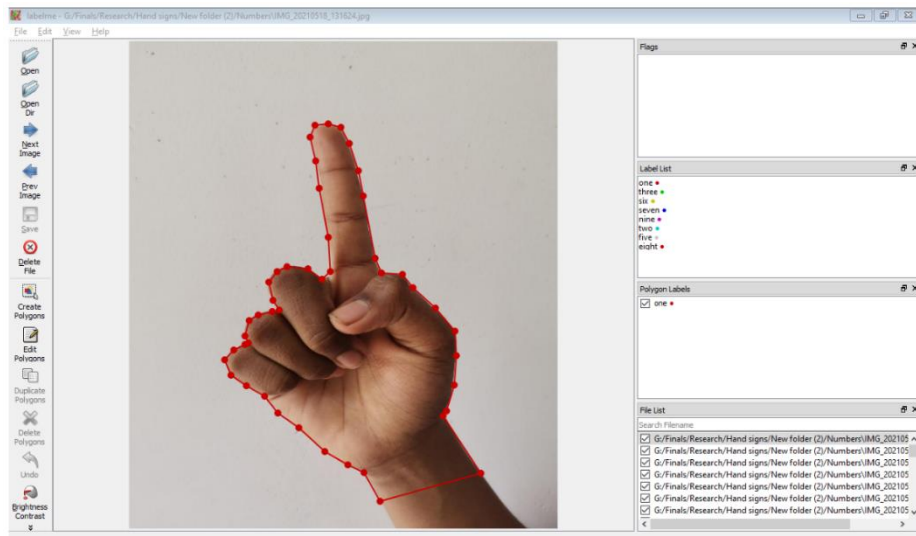


Figure 2. Label Me annotation tool

After saving the annotated image as a json file, all the images were converted to Common Object in Context (COCO) format. Finally, this dataset is converted to TFRecords, Tensorflow's binary storage format, which increases the performance and training speed of the model.

3.4. Implementing the Model

To train the custom model for the sign identification, a pretrained model which is best for this project was selected from the Tensorflow model zoo. This is because a pre-trained model is previously trained using a large-scale image classification task. To serve the purpose of Sinhala sign detection, transfer learning was used which is used to customize these pretrained models. The pretrained model selected for this purpose was `mask_rcnn_inception_resnet_v2` model, with the highest COCO mean average precision.

The configuration file, `mask_rcnn_inception_resnet_v2_1024x1024_coco17_gpu-8`, for training the model was available in the Tensorflow 2 GitHub repository in the 'configs' directory. This file was updated to meet the requirements needed to train the model for this project. Finally, Google Colaboratory was used to train the model with a GPU accelerated compute engine. Figure 7 shows the architecture of a mask R-CNN model. This trained model is loaded into the backend (Python API) of the system since it requires a considerable amount of processing power.

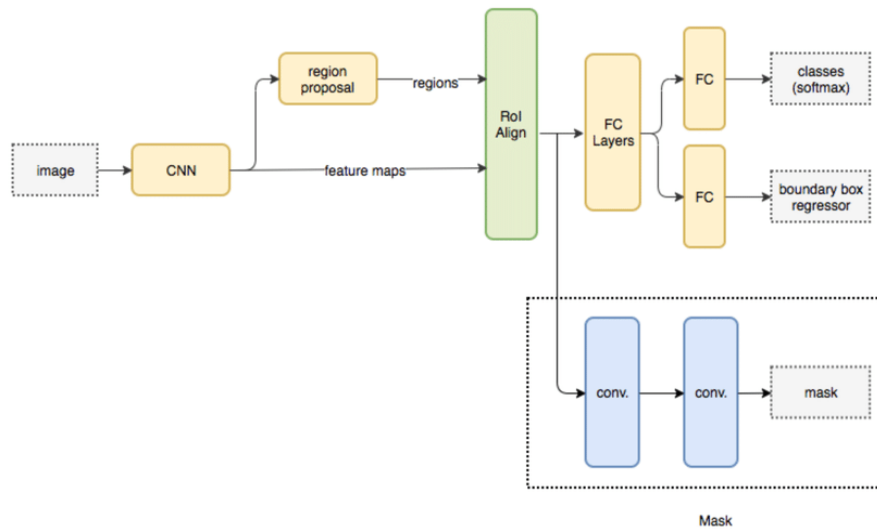


Figure 3. Architecture of the mask R-CNN framework [4]

3.5. Implementation of the Interpretation

The frontend of the system, mobile application, is used to feed data into the system. Interpretation occurs on the backend API of the system.

OpenCV VideoCapture was used to capture the video feed from the user. Since the API and database was not hosted on a Cloud Server, this was assigned to use the camera from the machine that the API is running since the server should have high GPU performance for the functionality of this feature.

Once the video is captured into the system, it will be separated into frames and processed to detect the sign in each frame by sending the frames through the loaded model. Later, the text value of the detected sign is retrieved from the database using the label class of the detected sign. Figure 4 shows the flow diagram for the interpretation process in the system.

Another feature of the system is the sign dictionary. For this a GET request is sent from the frontend to the API. After receiving the request, the API will execute the query to return all the data available in the database. This will display a list of signs and their values.

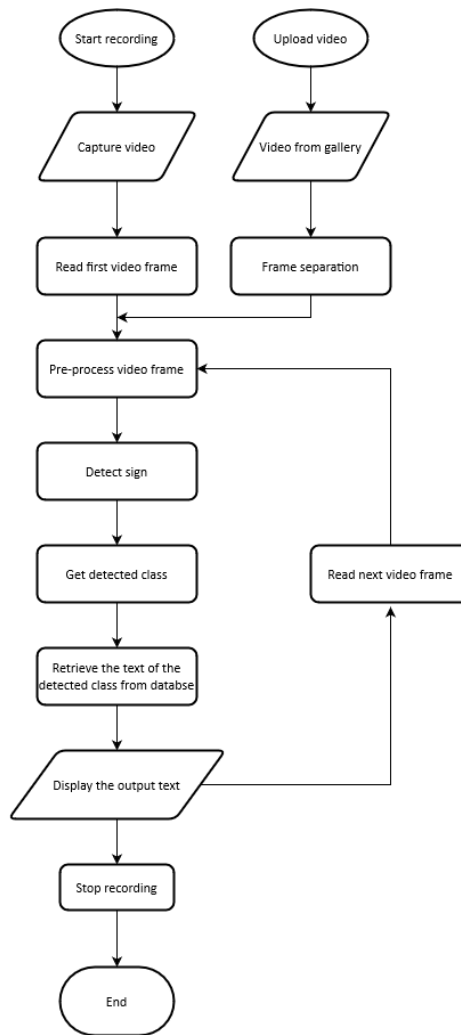


Figure 4. Process flow of the interpretation

3.6. Final Prototype

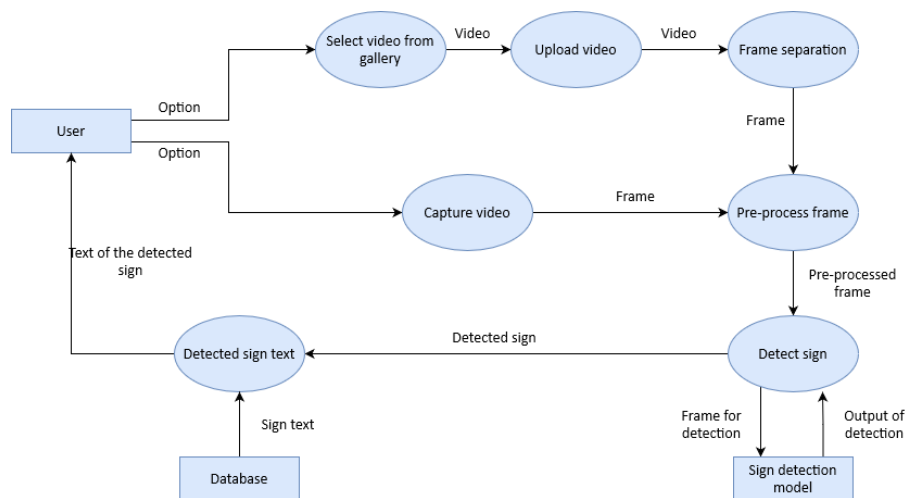


Figure 5. 2-Level Data Flow Diagram

Figure 5 shows the 2-Level Data flow diagram of the system that explains the flow of the core functionality of the system, which is real-time interpretation of Sinhala sign language, by showing both the core and sub functionalities.

In this diagram, the user is considered as an external entity. User is given two options, to upload video from the gallery or capture real-time video and upload to the system, for the interpretation to happen. If a video is uploaded from the gallery, it is separated into frames and pre-processed. If the user selects to capture real-time video, since the video is captured as frames, the frames are sent straight to the pre-processing stage. Once the frames are processed, they are sent to the detection model for detection. Detected sign value is then retrieved and using the database, the text value of the detected sign is retrieved and displayed to the user.

3.6.1. User Interface

User interface was designed to be simple and easy to use. Below are the screenshots of the UI of the final prototype.



Figure 6. Real-time interpretation UI

Figure 6 shows the real-time interpretation UI after performing an interpretation.

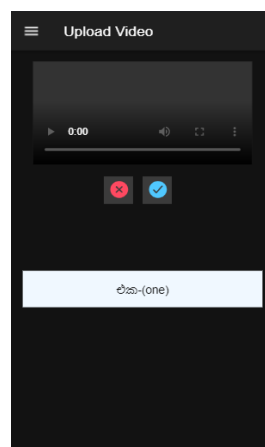


Figure 7. Video interpretation UI

Figure 7 shows the UI and an example output of the video interpretation.



Figure 8. Sign dictionary UI

Figure 8 shows the list of signs with its UI in the sign dictionary.

4. EVALUATION

4.1. Evaluation Data and Process

4.1.1. Accuracy

The accuracy was tested for each sign that the model was trained. This was done for both the real-time and video interpretation features. For the real-time interpretation, each sign was tested with 10 different users where each user performed the same sign for 5 times. For the video interpretation, the same 10 users were used to record 10 small video clips with each person signing. Accuracy values were taken as the average calculated from the results generated from the mentioned process. Below table shows the calculated average.

Table 3. Accuracy readings for real-time sign interpretation

Sign used	Percentage accuracy
1	71%
2	75%
3	73%
4	75%
5	82%
6	79%
7	83%
8	76%
9	88%
ඒ	70%
ඈ	71%
ඉ	84%
එ	79%
උ	76%
ඌ	85%

Overall accuracy of the trained model for real-time interpretation – 77.8%

4.1.2. Usability

To test the usability and responsiveness of the mobile application, it was installed and tested in several android mobile devices. The user interface was displayed without any issues and was responsive irrespective of the screen resolution. Same process was done by running the application as a web application. This was also successful without any problems. The flow of the application was working without any dead ends. Since this application will be mostly used by language speakers, for testing random people were selected who are natural language speakers. Below is the feedback received from the users on usability.

“The application that was developed is surprisingly easy to learn and use.” – Mr. Thathsara Nandun.

“The user interface of the system can be understood easily and without any prior knowledge of the system, it could be easily used.” – Mrs. Chandra Jayanthi.

4.2. Evaluation of Machine Learning Model

The Jupyter notebook created was loaded from the Google Colaboratory. The required libraries and protocol buffers were installed after cloning the Tensorflow GitHub repository to the google drive and mounting the drive to the Google Colaboratory. The training of the model was stopped when the graph curve started to flatten, making small changes in the loss value at 10460 steps with each step being batch size of 1 feeding the model with an image per step and until the loss reading was less than zero. This was done to prevent the model from overfitting. Finally, the model was exported for inference. Figure 9 shows the output generated during the model training.

```

!cd /content/drive/MyDrive/Level6/Project/tensorflow/models/research/object_detection/
!python model_main_tf2.py --pipeline_config_path=resnet/mask_rcnn_inception_resnet_v2_1024x1024_coco17_gpu-8.config --model_dir=resnet --alsologtostderr

INFO:tensorflow:Step 8900 per-step time 0.519s loss=0.660
10329 10:54:09.539093 139751235426176 model_lib_v2.py:651] Step 8900 per-step time 0.519s loss=0.660
INFO:tensorflow:Step 9000 per-step time 0.491s loss=1.294
10329 10:55:04.157033 139751235426176 model_lib_v2.py:651] Step 9000 per-step time 0.491s loss=1.294
INFO:tensorflow:Step 9100 per-step time 0.520s loss=0.321
10329 10:56:03.157981 139751235426176 model_lib_v2.py:651] Step 9100 per-step time 0.520s loss=0.321
INFO:tensorflow:Step 9200 per-step time 0.519s loss=0.593
10329 10:56:58.766529 139751235426176 model_lib_v2.py:651] Step 9200 per-step time 0.519s loss=0.593
INFO:tensorflow:Step 9300 per-step time 0.521s loss=0.522
10329 10:57:54.738133 139751235426176 model_lib_v2.py:651] Step 9300 per-step time 0.521s loss=0.522
INFO:tensorflow:Step 9400 per-step time 0.488s loss=0.311
10329 10:58:59.664499 139751235426176 model_lib_v2.py:651] Step 9400 per-step time 0.488s loss=0.311
INFO:tensorflow:Step 9500 per-step time 0.547s loss=0.685
10329 10:59:45.347732 139751235426176 model_lib_v2.py:651] Step 9500 per-step time 0.547s loss=0.685
INFO:tensorflow:Step 9600 per-step time 0.672s loss=0.658
10329 11:00:40.723896 139751235426176 model_lib_v2.py:651] Step 9600 per-step time 0.672s loss=0.658
INFO:tensorflow:Step 9700 per-step time 0.651s loss=0.629
10329 11:01:35.772361 139751235426176 model_lib_v2.py:651] Step 9700 per-step time 0.651s loss=0.629
INFO:tensorflow:Step 9800 per-step time 0.504s loss=1.406
10329 11:02:30.688259 139751235426176 model_lib_v2.py:651] Step 9800 per-step time 0.504s loss=1.406
INFO:tensorflow:Step 9900 per-step time 0.522s loss=1.463
10329 11:03:26.065765 139751235426176 model_lib_v2.py:651] Step 9900 per-step time 0.522s loss=1.463
INFO:tensorflow:Step 10000 per-step time 0.599s loss=0.496
10329 11:04:22.162692 139751235426176 model_lib_v2.py:651] Step 10000 per-step time 0.599s loss=0.496
INFO:tensorflow:Step 10100 per-step time 0.618s loss=0.520
10329 11:05:18.732120 139751235426176 model_lib_v2.py:651] Step 10100 per-step time 0.618s loss=0.520
INFO:tensorflow:Step 10200 per-step time 0.632s loss=0.506
10329 11:06:16.123389 139751235426176 model_lib_v2.py:651] Step 10200 per-step time 0.632s loss=0.506
INFO:tensorflow:Step 10300 per-step time 0.567s loss=0.985
10329 11:07:11.151652 139751235426176 model_lib_v2.py:651] Step 10300 per-step time 0.567s loss=0.985
INFO:tensorflow:Step 10400 per-step time 0.627s loss=0.428
10329 11:08:06.408119 139751235426176 model_lib_v2.py:651] Step 10400 per-step time 0.627s loss=0.428

```

Figure 9. Model training output

4.3. Evaluation of the Final System

The research aim was to implement a Sinhala Sign Language interpreter by designing, developing, and evaluating Artificial Intelligence models and finally use them in building a mobile which will be able to successfully identify and interpret Sinhala signs in real-time.

The following features were implemented for this prototype of the system.

- Real-time sign interpretation using video capture and video upload from gallery.
- Sign language dictionary.
- Trained mask rcnn inception resnet model for object detection.

In the implementation of the prototype, few limitations were identified. They are,

- The accuracy of the sign identification depends on the light conditions of the captured/uploaded video.
- The video capture feature is working as a web application, and this feature was not deployed on a mobile device.
- Due to the lack of resources such as a GPU, the speed of the interpretations was longer than expected.
- Sign identification only works for still gesture signs.

After identifying the limitations of the system, the following future enhancements are discussed.

- Improve the accuracy of the system based on the light factor of the input video.
- Implement and deploy a mobile version for the video capturing feature.
- Increase the speed of the interpretations by hosting the API on a cloud server.
- Implement and train the model for identifying dynamic signs.

This was how the research was done, a prototype was implemented and tested, its limitations were identified, and future enhancements were discussed.

5. CONCLUSION

The research aim was to implement a Sinhala Sign Language interpreter by designing, developing, and evaluating Artificial Intelligence models and finally use them in building a mobile or web application which will be able to successfully identify and interpret Sinhala signs in real-time. This was successfully achieved by implementing the system.

Implementing this research project has also benefitted the authors by helping to learn new tools and technologies and improve existing skills. Knowledge needed was gained by referring the documentations of each technology and testing the knowledge gained by implementing simple functionalities. Usage of Tensorflow library was studied deeply for the purpose of this research.

During the process, one of the challenges faced was the lack of datasets to train the model. As a result, a custom dataset was developed by the authors. Another problem was the lack of resources to train the model. To overcome one such challenge, Google Colaboratory was used to train the model without any issues.

The authors hope this will be a good solution to identify and interpret signs from Sinhala sign language and by doing so, helping to reduce the communication gap between sign users and language speakers.

ACKNOWLEDGEMENTS

We would like to extend our gratitude towards our parent university, University of Central Lancashire for providing us with the opportunity and resources to complete this research successfully. Our appreciation goes to the staff of Universal College Lanka for supporting and advising us whenever needed.

We would also like to thank our colleagues for supporting and encouraging us at times required and the candidates that volunteered to collect the data needed and support when testing the system. Last but not least, we would like to thank our parents for their continuous support and the motivation given to us throughout the project.

REFERENCES

- [1] World health organization. (2021, April 1). Deafness and hearing loss. <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>.
- [2] Ahmed, M., Idrees, M., ul Abideen, Z., Mumtaz, R., Khaliq, S. (2016). Deaf talk using 3D animated sign language: A sign language interpreter using Microsoft's kinect v2. IEEE. <http://dx.doi.org/10.1109/SAI.2016.7556002>.
- [3] Sinhala Sign Language the main communication mode for the Deaf in Sri Lanka. (2019, January 18). DailyFT. <http://www.ft.lk/opinion/Sinhala-Sign-Language-the-main-communication-mode-for-the-Deaf-in-Sri-Lanka/14-671078>.
- [4] Rupasinghe, H. (2018, March 27). Sri Lanka Terribly short of sign language interpreters. *Dailymirror*.
- [5] Nath, G. G., Arun, C. S. (2017). Real time sign language interpreter. IEEE. <http://dx.doi.org/10.1109/ICEICE.2017.8191869>.
- [6] Ibraheem, N. A., Khan, R. Z. (2021). Vision Based Gesture Recognition Using Neural Networks Approaches: A Review. <https://www.researchgate.net/profile/Noor-Ibraheem/publication/267991106>.
- [7] Lei, L. Dashun, Q. Design of data-glove and Chinese sign language recognition system based on ARM9. 2015 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), 2015, pp. 1130-1134, <http://dx.doi.org/10.1109/ICEMI.2015.7494440>.
- [8] Mori, Y. Toyonaga, M. Data-Glove for Japanese Sign Language Training System with Gyro-Sensor. 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), 2018, pp. 1354-1357, <http://dx.doi.org/10.1109/SCIS-ISIS.2018.00211>.
- [9] Nath, G. G., Arun, C. S. (2017). Real time sign language interpreter. IEEE. <http://dx.doi.org/10.1109/ICEICE.2017.8191869>.
- [10] Ahmed, M., Idrees, M., ul Abideen, Z., Mumtaz, R., Khaliq, S. (2016). *Deaf talk using 3D animated sign language: A sign language interpreter using Microsoft's kinect v2*. IEEE. <http://dx.doi.org/10.1109/SAI.2016.7556002>.
- [11] Madushanka, A. L. P., Senevirathne, R. G. D. C., Wijesekara, L. M. H., Arunatilake, S. M. K. D., Sandaruwan, K. D. (2016). *Framework for Sinhala Sign Language recognition and translation using a wearable armband*. IEEE. <http://dx.doi.org/10.1109/ICTER.2016.7829898>.
- [12] Dissanayake, I. S. M., Wickramanayake, P. J., Mudunkotuwa, M. A. S., Fernando, P. W. N. (2020). *Utalk: Sri Lankan Sign Language Converter Mobile App using Image Processing and Machine Learning*. IEEE. <http://dx.doi.org/10.1109/ICAC51239.2020.9357300>.

AUTHORS

Rusiru Sewwantha is currently a BSc (Hons) in Software Engineering 4th year Undergraduate at Universal College Lanka affiliated with University of Central Lancashire, UK. His passion towards programming and machine learning has made it possible for him to complete this research.



Thepul Ginige is currently reading for his Ph.D. (University of Colombo), has a Master of Information Technology (University of Colombo), Pg. Dip in Criminology & Criminal Justice (University of Sri Jayewardenepura), Postgraduate diploma national university of Singapore, Bachelor of Science (Hons) (Information Technology Sri Lanka Institute of Information Technology), Bachelor of Science (University of Sri Jayewardenepura). He is currently working as a Senior Lecturer / Programme Coordinator– Universal College Lanka. Former -Senior Lecturer / Manager Special Project -Saegis Campus, Senior Lecturer / Academic Coordinator (Scottish Qualifications Authority -SAQ) – Saegis Campus, Senior Lecturer / Programme Coordinator - Institute of Human Resource Advancement(IHRA) -the University of Colombo, Senior Lecturer/Head of Division- National Institute of Business Management, Senior Lecturer/Lecturer /Assistant Lecturer- University of Colombo School of Computing, Graduate Project Assistant University of Colombo School of Computing. Mr. Ginige is a Member of the Institute of Electrical and Electronics Engineers (MIEEE), a Member of the British Computer Society (MBCS), Member of the Sri Lankan Computer Society (MCSSL). Session Chair if International Conferences, 2021 International Conference On Computer Communication And Artificial Intelligence (Ccai 2021) Guangzhou, China - Session 3: Computer Vision and Image Application, ICIM 2021 the 7th International Conference on Information Management, 2021 |London, UK. Co-Sponsored By Patrons EAET 2021, 2nd European Advanced Educational Technology Conference- Session 4: Information Teaching and Management, ICBT Annual International Research Symposium (AIRS 2018)- Co Char for Information Technology Session. Conference Committee Member as a Reviewer in International Conference on - ICGDA(January 21-23, 2022) Paris, France, 2nd European Advanced Educational-Technology Conference (EAET 2021, Imperial College, London, UK), 5th International Conference on Information System and Data Mining (ICISDM2021, Silicon Valley, CA, USA), 7th International Conference on Computing and Data Engineering (ICCDE2021, Phuket, Thailand), International Conference on Computer Communication and Artificial Intelligence (CCAI 2021, Guangzhou, China), 4th International Conference on Computers in Management and Business (ICCMB2021, Singapore), ICBT Annual International Research Symposium (AIRS 2020, 2019, 2018, Colombo, Sri Lanka)

