Real Time Sign Language Detection and Subsequent Text-to-Speech Generation

Arnisha Khondaker (ID: 20166060)  
*Department of Computer Science and Engineering   
BRAC University*66 Mohakhali, Bangladesh  
[arnisha.khondaker@bracu.ac.bd](mailto:arnisha.khondaker@bracu.ac.bd)Salman Sayeed Khan (ID: 20366016)  
*Department of Computer Science and Engineering   
BRAC University*66 Mohakhali, Bangladesh  
[salman.sayeed@bracu.ac.bd](mailto:ipshita.upoma@bracu.ac.bd)Dr. Md. Ashraful Alam  
*Assistant Professor, Department of Computer Science and Engineering   
BRAC University*66 Mohakhali, Bangladesh  
 [ashraful.alam@bracu.ac.bd](mailto:%20ashraful.alam@bracu.ac.bd)

*Abstract*— Automatic sign language recognition addresses the longstanding issue of communication barrier between people belonging to the deaf and dumb community and the ones with no such disability. Sign language is not common to all and an exceedingly small portion of the hearing majority knows about it while that is the only way of communication for the deaf and dumb community. Our work proposes a recognizer that can detect English alphabets and numbers from hand gestures with the help of OpenCV and CNN. The generated texts are then passed to a text-to-speech synthesizer to convert text into speech. The architecture gives an overall accuracy of 89.9%. This mechanism can be applied to a wide array of other gestures and signs used as a medium of communication across different parts of the world. It can also lead to automatic text editors where people can write using only hand gestures.

Keywords—*Sign Language Recognition, CNN, OpenCV, Text-to-Speech.*

# Introduction

Sign language, despite being incomprehensible to a vast majority of the hearing community, is a medium communication for 70 million people with hearing disability across the world; more than 300 sign languages are being used for day-to-day communication among the concerned communities globally [1]. In sign language, a person with hearing disability uses different parts of his/her body such as head, hand, fingers, expression, and others to convey a certain message to another person [2]; visual motion is the medium of communication here. Even though sign language is being used by 70 million people, majority of the communication tools and technologies currently in the market does not support this visual mode of communication; these technologies facilitate only spoken or written forms of communication, thus excluding the deaf and dumb community’s concern.

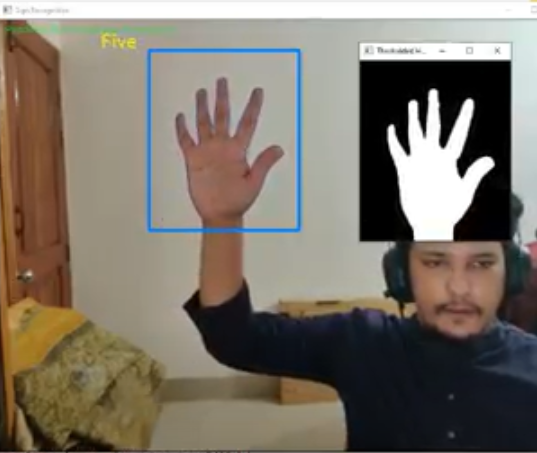
Recognition of the intended meaning of different visual signs in any sign language is a complex area of analysis in computer vision. This complexity mainly arises from the different meanings that the signs convey with even the slightest variation in the gestures made; there can be thousands of combinations based on the variation of motion profile, hand position, facial expression, body parts movement and shape of the hand. To be more precise, based on the use of the five parameters namely location, movement, expression, hand shape, and palm orientation, a person can convey a myriad of messages. Understanding these parameters can therefore aid in recognizing the messages.

In recent times, with the application of dep learning, CNN has shown remarkable accuracy in terms of feature understanding and recognition related to computer vision tasks. Numerous research efforts have been documented in the field of sign language recognition [3,4]. These recognition models improve their results significantly by employing deep learning approaches and learning deep features instead of hand-crafted features. However, challenges remain. The most crucial challenge is the development of such a mechanism that can translate the visual signs or gestures, as a whole, into texts, words, or sentences so that the latter can be recognized by the hearing community, thus facilitating an uninterrupted flow of communication between the deaf and dumb and the rest of the world. The problem with current developments in sign recognition is that they can work with dataset related to signs or images of a single character, thus losing the whole intended meaning of a conversation. In this project, not only will real time texts be generated from gestures using a more comprehensive custom dataset to include multiple characters, but generated texts will also be converted to speech using pyttsx3 python text-to-speech conversion library.

The remaining part of the paper is laid out as follows: Section II does a review of relevant literature of Deep Learning algorithms in the field of sign language recognition and speech conversion mechanisms. Section III describes the methodology used for the project. The results are described in section IV. Finally, limitations and future work are discussed in section V.

# Literature Review

When it comes to sign recognition, it can be done in two ways: recognition based on vision and sensor [5]. Approaches based on sensors require the use of wires, gloves, and other tools. However, these technologies demand for one to wear them continuously which can serve as a hindrance. Therefore, research focus was shifted in numerous studies to image-based approaches; in this regard, in the last few years, studies were done on hand gesture and sign recognition. According to those studies, HMMs (Hidden Markov models) [6], ANNs (Artificial neural networks) [7], Eigen value based models [8], and perceptual color based models [9] are some of the approaches that proved to be promising in gesture recognition from images for sign language analysis. Moreover, ANNs and variants of ANNs showed significant performance in feature extraction compared to others.

A CNN model was proposed by Roel et al. where recognition of a set of 50 non-identical signs could be performed using Microsoft Kinect. The signs were from Flemish Sign language; the margin of error was only 2.5%. However, a limitation of this model was that only a single individual in a given environment was considered [10]. In [11], a CNN was employed on a preprocessed Italian sign gesture dataset. The model had 6 layers for training and Rectified linear Units (ReLUs) as activation functions. The authors achieved a remarkably high accuracy, having only 8% error rate. Due to its excellent performance, CNN was adopted for this project. Although generative networks such as FlowTron [12] and transformer networks [13] give a higher quality of text to speech synthesis; for the conversion of text to speech, a rather simpler approach is utilized in this work to lower down the computational complexity and get a better runtime. Python’s pyttsx3 library was used for the text to speech conversion.

# Methodology

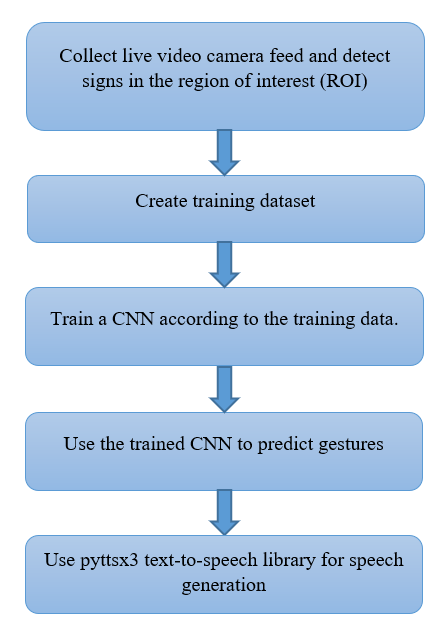
The overall architecture of the work consists of four parts: dataset creation, training provision for a CNN, prediction of data, and generation of speech. The basic workflow chart of the proposed model is depicted in figure 1.

Figure 1. Workflow diagram of proposed architecture

* 1. *Dataset*

Instead of using a readily available dataset from the internet, a custom dataset has been utilized to increase the versatility of the project and to ensure future researchers can employ the system with their own dataset at ease.

To begin, a live feed is collected from the video cam. Afterwards, every frame is tracked to detect a hand in the region of interest (ROI). Then, a gesture directory is used to save those specific frames where a hand was detected. The gesture directory has a train and a test folder; each of these two folders has 10 sub-folders that contain the captured images.

For the first step of the workflow (creation of dataset), OpenCV is used to get cam feed and create a region of interest. Here, ROI is that specific section of the frames where hand (for gesturing) is detected. As illustrated in figure 2, the box in blue is the region of interest whereas the window is for collecting live webcam feed.

Figure 2. Extracting region of interest from live video feed

To differentiate the background, the background’s accumulated weighted average is first found out. Subsequently, the result is deducted from those frames that has something in front of the background; this layer or object needs to be separated as the foreground. This process is accomplished by finding out the total weight for 60 frames. Once the background’s accumulated average is ready, it is deducted from every frame that was read after 60 frames. It is done to identify the objects occluding the background.

The next step is to find out the threshold value for each frame, and then identify the contours. For this purpose, cv2.findContours has been utilized which return the maximum contours with the help of the function segment. With these contours, the existence of any foreground object in the ROI can be detected. In simpler terms, it can be found out whether there is a hand in the region of interest.

Once a hand is detected in an ROI, the image saving process starts. It is saved in the train and test dataset respectively. 3010 images per character or number were saved for the train data set whereas 1010 images were saved for the test set.



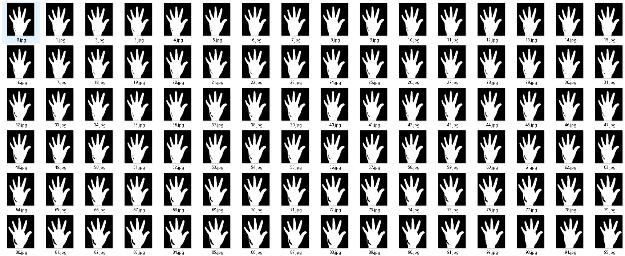
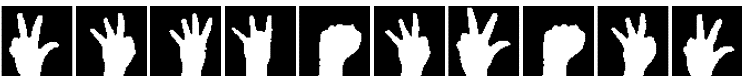


Figure 3. Sample training dataset after pre-processing

## Training CNN

Data is loaded with the help of ImageDataGenerator where flow\_from\_directory function is used to load both the datasets. All the names of the numbered folders has been used here as image class names. There is an image function with the purpose of image plotting for the loaded datasets.

Figure 4. Sample images loaded from the dataset

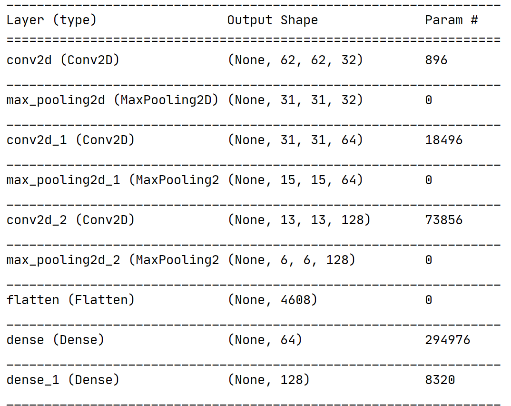
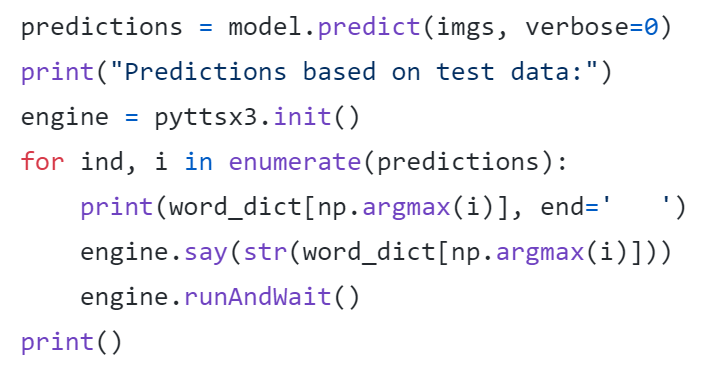
The CNN is then designed as show in Figure 5. After generating the model, it is fitted to the data.

Figure 5. The design of the CNN model

Callbacks of reduced LR on plateau and earlystopping – both are used in training; these two rely on the validation dataset loss.

At the end of each epoch, the validation dataset is used to find out the loss and accuracy; if the loss is not going down, the LR is decreased using Reduce LR; doing so stops the model from crossing the loss minima. Additionally, earlystopping is used to stop the training in case the validation accuracy keeps falling.

For better accuracy, separate algorithms are used for optimization - SGD (stochastic gradient descent) and Adam which is a combination of RMSProp and Adagrad.

Once all compilation is done, the model is fit to the train batches for 10 epochs in the way discussed above.

## Prediction Model

For ROI detection and measurement accumulated average is calculated in the dataset creation phase. A bounding box is also generated. The creation is done by pinpointing a foreground object.

Next, the max contour is pinpointed; the detection of the contour means that a hand was identified and thus the ROI threshold is considered as a test image.

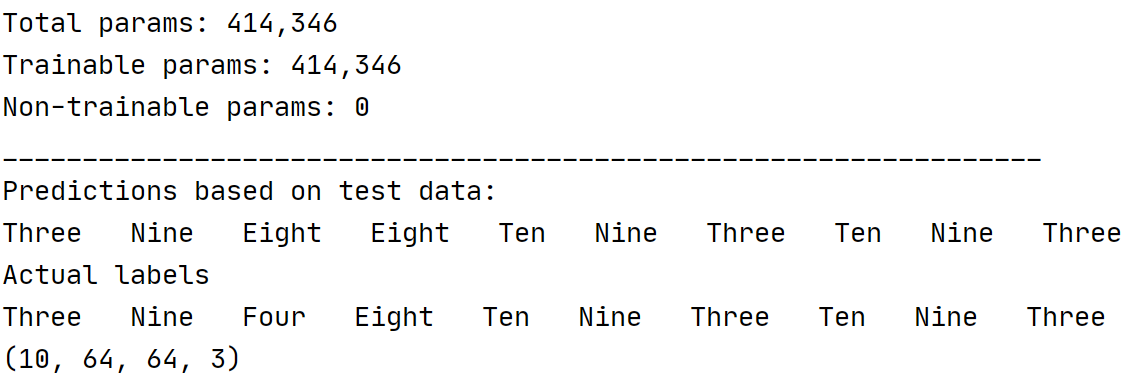
The earlier saved model is loaded with the help of keras.models.load\_model; the ROI image is fed as a model input for prediction purposes as depicted in figure 6.

Figure 6. Prediction of the model on sample test data

## Text-to-Speech

We use python’s library pyttsx3 to convert the text to speech. The simple approach is undertaken to reduce the computational complexity and reduce the runtime of the overall model. The code block for text-to-speech conversion is shown in figure 7.

Figure 7. Implementation of text-to-speech conversion

# Experimental Analysis and Results

The primary tools used for this project are python, keras with a tensorflow backend and a webcam of resolution of 640x480. Training the model has been done on a Nvidia GPU (GeForce GTX 1080 Ti).

## Results

89.9% training accuracy and validation accuracy of about 81% was found out as the results. The loss of this model converges to 0.0017 at 100th epoch.

Figure 6. Accuracy and Loss over training epochs of CNN

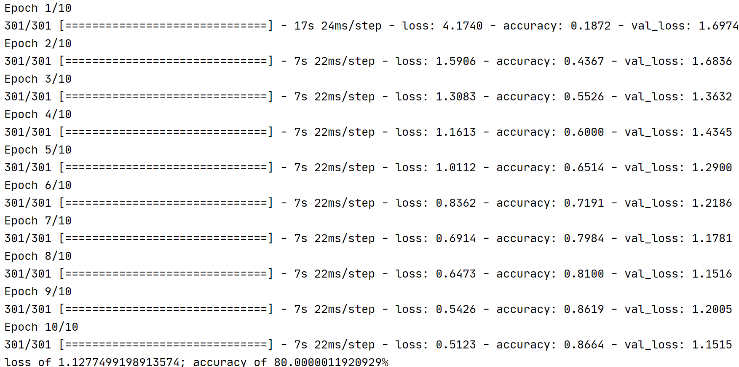
 The accuracy on the test set is 80% as shown in figure 7. It has been run for a 10 epochs.

Figure 7. Accuracy and Loss over testing epochs of CNN

# Limitations and future work

This project shows that for accurate recognition of different signs in a sign language, convolutional neural networks can be relied upon. Such generalization capacity of CNNs might add to further research on automatic sign language recognition. In the future, a more complicated CNN architecture can be employed to increase the accuracy. A more complex text to speech synthesizer can also be adhered. Additionally, the system can be integrated with a text editor to make texting easy for the deaf community.

##### References

1. Murray, J. (2018). World Federation of the deaf. Rome, Italy. Retrieved from http://wfdeaf.org/our-work/. (Accessed 30 January 2020). MXNET (2020). MXNET. Available online: Accessed date: Jun, 2020.
2. M. Cheok, Z. Omar and M. Jaward, "A review of hand gesture and sign language recognition techniques", *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131-153, 2017. Available: 10.1007/s13042-017-0705-5 [Accessed 26 May 2021].
3. A. Wadhawan and P. Kumar, "Deep learning-based sign language recognition system for static signs", *Neural Computing and Applications*, vol. 32, no. 12, pp. 7957-7968, 2020. Available: 10.1007/s00521-019-04691-y [Accessed 26 May 2021].
4. S. Bambach, S. Lee, D. Crandall and C. Yu, "Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions", *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. Available: 10.1109/iccv.2015.226 [Accessed 26 May 2021].
5. A. Chonbodeechalermroong and T. Chalidabhongse, "Dynamic contour matching for hand gesture recognition from monocular image", *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2015. Available: 10.1109/jcsse.2015.7219768 [Accessed 26 May 2021].
6. S. Lang, M. Block and R. Rojas, "Sign Language Recognition Using Kinect", *Artificial Intelligence and Soft Computing*, pp. 394-402, 2012. Available: 10.1007/978-3-642-29347-4\_46 [Accessed 26 May 2021].
7. S. Saengsri, V. Niennattrakul and C. Ratanamahatana, "TFRS: Thai finger-spelling sign language recognition system", *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, 2012. Available: 10.1109/dictap.2012.6215407 [Accessed 26 May 2021].
8. J. Singha and K. Das, "Indian Sign Language Recognition Using Eigen Value Weighted Euclidean Distance Based Classification Technique", *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, 2013. Available: 10.14569/ijacsa.2013.040228 [Accessed 26 May 2021].
9. M. M. and I. Jackin, "Wireless Vision Based Mobile Robot Control Using Hand Gesture Recognition Through Perceptual Color Space", *2010 International Conference on Advances in Computer Engineering*, 2010. Available: 10.1109/ace.2010.69 [Accessed 26 May 2021].
10. Verschaeren, R. “Automatic gesture recognition with the Microsoft Kinect translation”, *Neural Computing and Applications*, vol. 32, no. 12, pp. 7957-7968, 2012.
11. L. Pigou, S. Dieleman, P. Kindermans and B. Schrauwen, "Sign Language Recognition Using Convolutional Neural Networks", *Computer Vision - ECCV 2014 Workshops*, pp. 572-578, 2015. Available: 10.1007/978-3-319-16178-5\_40 [Accessed 26 May 2021].
12. R. Valle, K. Shih, R. Prenger and B. Catanzaro, "Flowtron: an Autoregressive Flow-based Generative Network for Text-to-Speech Synthesis", in *International Conference on Learning Representations*, 2021.
13. N. Li, S. Liu and Y. Liu, "Neural Speech Synthesis with Transformer Network", in *AAAI Conference on Artificial Intelligence*, 2021.