Sign Language Detection and Translation: Text and Speech using CNN, NLP

Abstract — People who struggle with speech impairment often rely on sign language for communication since they can't use hearing and speech effectively. However, conversing with those who don't understand sign language can be challenging for them. This underscores the necessity for sign language interpreters to bridge this communication gap, both in informal and formal settings. Recent advancements in deep learning have led to promising developments in gesture and motion recognition technology. A proposed solution aims to translate hand gestures into text in real-time, making communication more accessible for non-signers. Unlike previous research that primarily focused on translating individual letters or numbers, this system utilizes Convolutional Neural Networks (CNN) for hand gesture classification. By implementing such a system, the disparity between signers and non-signers can be reduced, easing communication for individuals with speech impairments. Research into Artificial Neural Networks provided the groundwork for this project, with previous models achieving an accuracy of around 86% in character detection. While Linear Discriminant Analysis (LDA) was considered, its limitation in handling complex data led to its exclusion. The project also delved into hardware implementation, recognizing the associated costs and maintenance requirements. Efforts were made to mitigate these factors in the developed system. Achieving a high accuracy rate of 96.5%, the model also offers word suggestions and sentence formation features, setting it apart from previous research efforts.

Keywords — *CNN, ROI, OpenCV, NLP, ADA GRAD, N-Grams.*

I. Introduction

American Sign Language stands out the primary mode of communication for individuals with hearing and speech impairments. Because their disabilities mainly affect communication, those who are deaf and mute rely on sign language exclusively for conveying their thoughts and understanding others. Communication encompasses the exchange of ideas through various means, including speech, signals, behavior, and visual cues. Deaf and mute individuals, referred to as D&M people, utilize their hands to articulate different gestures, allowing them to communicate effectively with others. These gestures serve as nonverbal messages, comprehensible through visual perception. This form of nonverbal communication is known as sign language, a visual language comprising three major components:

WORD LEVEL VOCABULARY	FINGER SPELLING	NON-MANUAL FEATURES
Used for most of the communication.	Used to spell words letter by letter	Facial expressions and body position and tongue, mouth.

Our project primarily centers on developing a model capable of recognizing manual alphabet hand gestures and combining them to form the words. The gestures we intend to train the model on are depicted in Figure 1.

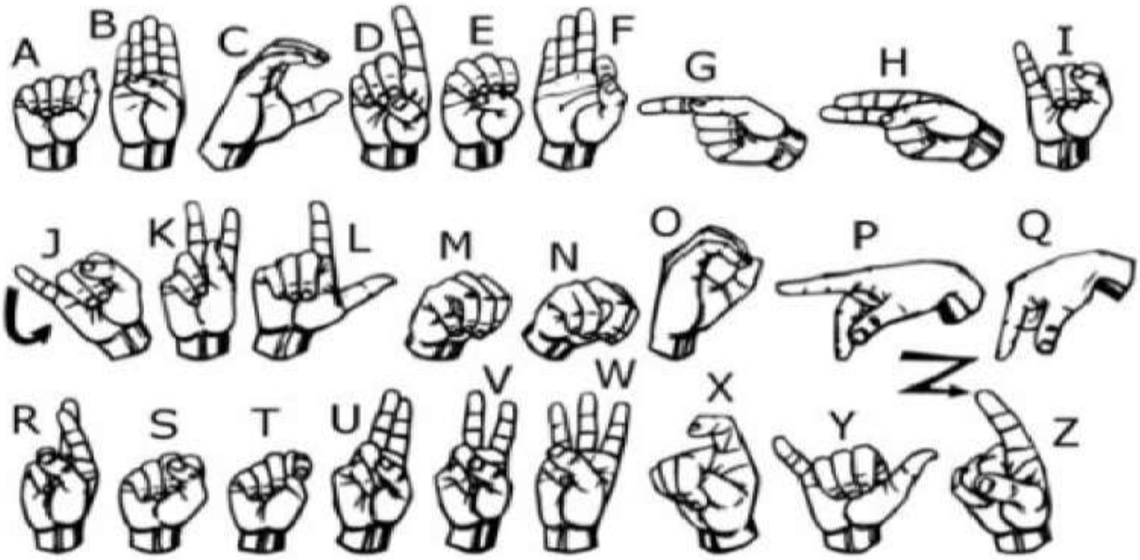


Figure 1. Signs for all 26 English alphabets A-Z.

II. Related Work

The literature survey delves into recent advancements in sign language recognition, highlighting key findings from five distinct research papers. Barbhuiya et al. (2021) proposed a CNN+SVM approach for static sign language recognition, achieving an impressive accuracy of 99.82% in recognizing alphabets and numbers in American Sign Language (ASL). Their focus on isolated gestures underscores the significance of accurate classification for individual signs. Aly et al. (2020) employed a DeepLabv3+Bi-LSTM technique, achieving an accuracy of 89.59% in recognizing ASL alphabets. Their method, while static and isolated, showcases the efficacy of combining deep learning architectures for improved sign language understanding.

In a dynamic setting, Lee et al. (2020) presented a LSTM+KNN model capable of recognizing alphabets in Arabic Sign

Language (SL). Their hybrid approach, accommodating both static and dynamic signs, reflects the complexity of sign language communication. Xiao et al. (2020) explored the use of GAN+LSTM+3DCNN for static sign language recognition, focusing on word-level classification in ASL with an accuracy of 99.44%. Their work emphasizes the importance of capturing temporal dependencies in sign language gestures to enhance recognition accuracy.

Lastly, Elakkiya et al. (2021) introduced a CNN + Bi-LSTM with attention mechanism for recognizing words and sentences in continuous sign languages like Chinese SL (CSL) and German SL (GSL). Their model achieved accuracies of 81.22% and 76.12% for CSL and GSL, respectively. By addressing the challenges of continuous sign language recognition, their research contributes to the development of more inclusive communication technologies. When taken as a whole, these studies highlight the variety of methods and strategies used in sign language identification and highlight the continuous attempts to increase inclusion and accessibility for the deaf of hearing community.

III. Methodology And Proposed Work

To examine the effectiveness of several deep learning models in identifying multiple types of lung cancer using CT-Scan pictures, the following methodology was used:

A. Dataset Acquisition: Initially, we searched for pre-existing datasets as part of our inquiry, but we were unable to find any in the raw picture format that satisfied our needs. Instead, we discovered datasets with RGB values shown. Because of this, we made the following decision to create our own dataset:

- We generated our dataset using the OpenCv package. We initially took around 800 photos of each symbol for training, and then we took about 75 photos of each symbol for testing.
- Our process We start the procedure by taking a picture of every frame from our machine's camera. As shown in Figure 2, we draw a region of interest (ROI) inside each chassis, which is symbolized by a Blue bordered square.

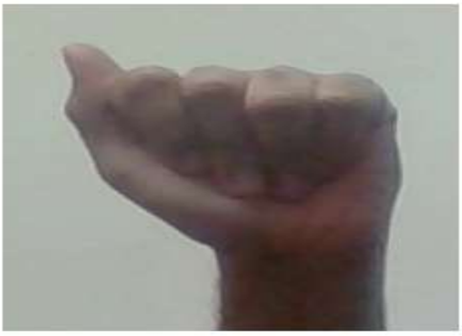


Figure 2. RGB image of alphabet "A"

- As seen in Figure 3, we convert this full image to a grayscale format by extracting our ROI, which is in RGB format.

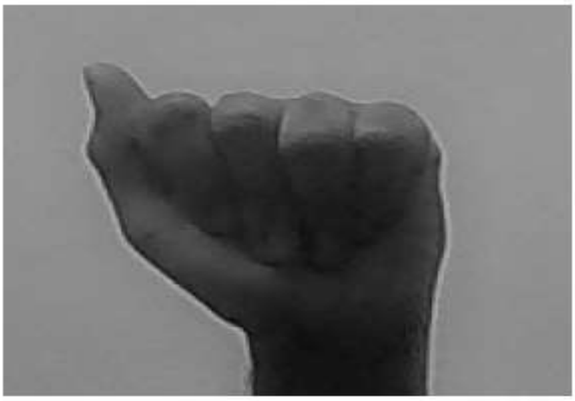


Figure 3. Grayscale image of alphabet "A"

- Finally, to help with the feature extraction process, we applied a Gaussian blur filter to our image. After activating the Gaussian blur, the final image resembles to Figure 4.

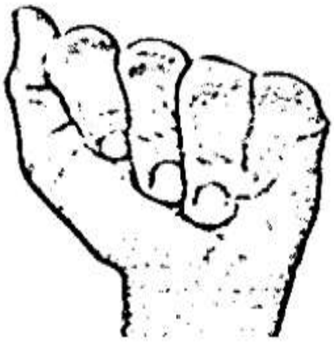


Figure 4. Image of alphabet “A” after applying Gaussian filter.

B. Gesture Classification: We use two levels of algorithms to estimate the user's final sign.

- Algorithm Layer:
    1. Take OpenCV collected frames and apply Gaussian blur filter and threshold on them. This method extracts characteristics from a picture.
    2. After analysis, the picture is sent into the CNN model for forecasting. A letter is printed and utilized to construct the word if it can be recognized in more than 60 frames. The blank sign is also used to denote the gaps between words.
- CNN Model:
    1. 32 filter weights (3x3 pixels) are applied to the picture, which is having a resolution of 128x128 pixel, in the first convolution layer. This generates a picture of 126 by 126 pixels for every filter weight.
    2. To downsample pictures while preserving the maximum value in each 2x2 square, the first pooling layer employs max pooling with a 2x2 window. The image is downsampled to 63x63 pixels therefore.
    3. Second Convolution Layer: 32 filter weights (3x3 pixels each) are applied to the 63x63 of the first pooling layer output as it passes through Second Convolution layer. This generates an image of 61x61 pixels. The second pooling layer reduces pictures to 30x30 pixels by using max pooling with a 2x2 frame.
    4. First Densely Connected Layer: 128 neurons in this fully connected layer receive images.
        - The array of 28800 values (30x30x32) is the result of the second convolutional layer's rearrangement of output. We employ a dropout layer with a value of 0.5 to avoid overfitting.
        - Fully connected layer of 96 neurons receives the output of the first densely coupled layer.
    5. Final Layer: Neurons in the final layer, which has an identical number of classes, are fed by the output of the second densely connected layer. The integrated design of the CNN that was previously mentioned is shown in Figure 5.

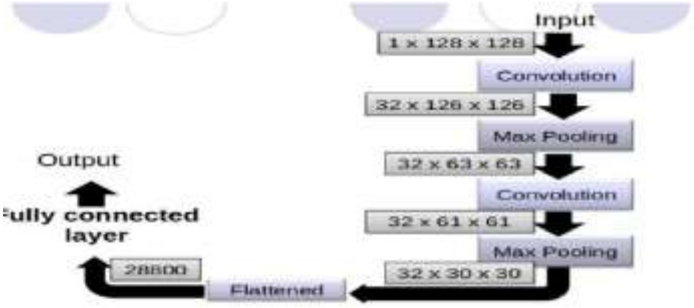


Figure 5. CNN Model

- Activation Function:

Whether the layers are fully linked or convolutional, we have incorporated ReLU in each one. ReLU improves the learning of complex features by computing max (x, 0) for each input pixel, adding nonlinearity to the calculation. It solves the vanishing gradient issue and shortens computation times to accelerate training.

- Pooling Layer:

With a pool size of (2, 2), we apply Max pooling to the input picture and combine it with the ReLU activation function. In doing so, overfitting is decreased, computational costs are decreased, and the number of parameters is minimized.

- Dropout Layers:

We utilize dropout layers, as seen in Figure 6, to prevent overfitting, in which the network's weights become unduly specialized to the training instances, leading to worse performance on new examples. By setting them to zero, these layers arbitrarily disable a portion of their activations. The network should still be able to give relevant outputs or classifications for samples even after some activations have been eliminated.

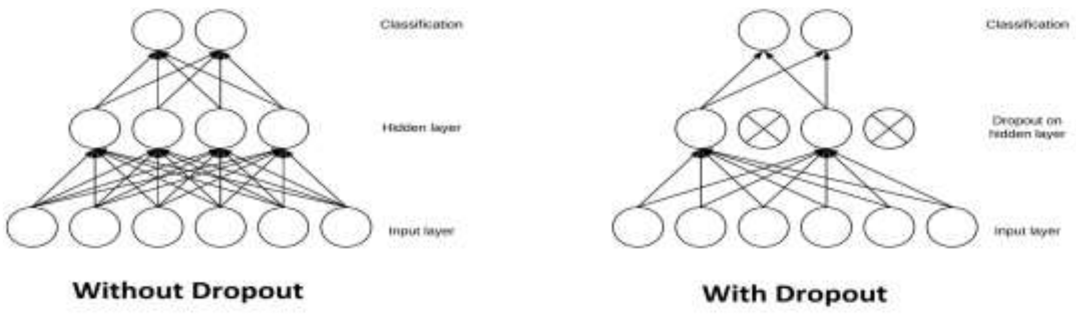


Figure 6. Dropout Layers

- Optimizer:

  Based on the loss function's result, we updated the model using the Adam optimizer. The ADA GRAD and RMSProp stochastic gradient descent techniques' benefits are combined in the Adam optimizer.

  Implementation:

  1. We output the ascertain letter and append it to the current string, if it surpasses a threshold value and no other letters are nearby.
  2. If not, we clear the current dictionary with the number of ascertains of the symbol to avoid incorrect letter prediction.
  3. No gaps are detected if there are more blank (plain background) identifications than a predetermined threshold and the buffer is empty.
  4. Otherwise, It adds the current buffer to the sentence and prints a space to denote the end of a word.

- Autocorrect Feature:

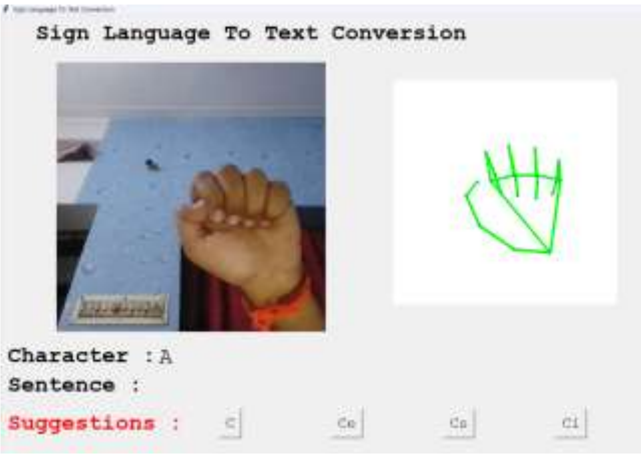
  We use the Python package Hunspell_suggest suggesting appropriate alternatives for each invalid input word. Users are presented with a list of words that match the current term, from which they can select one to add to the current statement. This function helps reduce spelling errors and forecast complicated words.

C. Training and Testing:

1. We first convert the RGB input photos to grayscale, then utilize adaptive thresholding to separate the hands from the backdrop and use Gaussian blur to eliminate extraneous noise. A 128 by 128 image resizing is used.
2. Following the aforementioned procedures, we send the input photos to our model for training and testing after preprocessing.
3. The prediction layer calculates the probability that a picture will belong to a particular class. The SoftMax function is used to standardize outputs between 0 and 1, guaranteeing that the total of the values in each class equals 1.
4. We use labeled data to train the network to increase prediction accuracy. A performance metric used in classification is called cross-entropy. It is minimized to optimize network weights, accomplished through the Adam Optimizer, which is a variant of gradient descent. TensorFlow provides an inbuilt function for calculating cross-entropy.

IV. Results:

An application interface has been successfully developed, capable of interpreting American Sign Language (ASL) in real-time and converting it to text, aiding individuals who are deaf and mute in effective communication without the need for a translator. Figure 7 illustrates a character prediction by converting hand gestures using the Media pipe library.


Sign Language To Text Conversion

Character : A

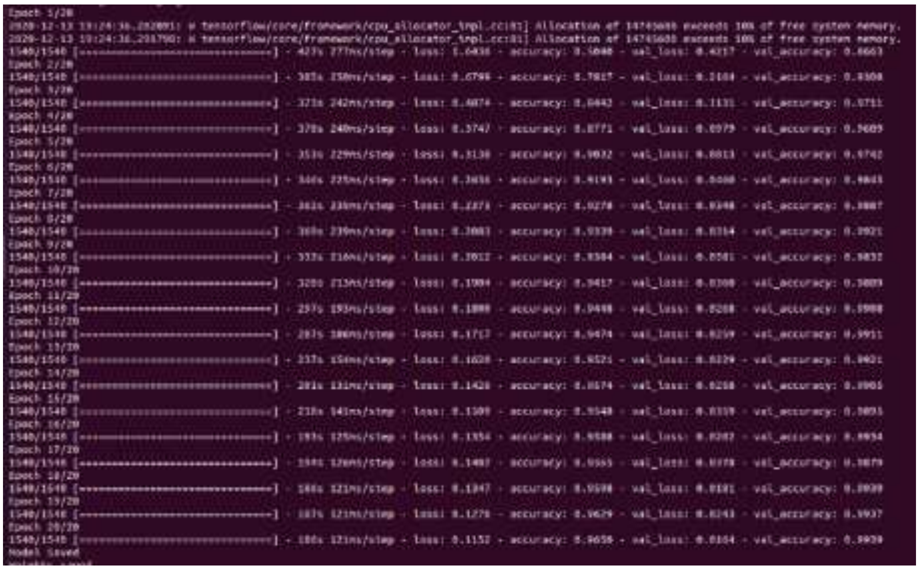
Sentence :

Suggestions :

Figure 7. Result obtained for a single character “A”.

The developed model outperformed several of the models investigated, achieving an astounding accuracy of 96.5%. The performance of the model is shown by the loss and accuracies of 20 epochs in Figure 8.

- The sign required to remain steady in front of the camera for at least 60 frames to guarantee proper detection.
- Different frame capturing values (40 and 80 frames) were evaluated for prediction.
- However, a frame value of 60 proved to be the optimal choice, balancing accuracy and speed of detection.



```
Epoch 1/20
2020-12-13 19:24:36.282091: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 14745600 exceeds 10% of free system memory.
2020-12-13 19:24:36.293790: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 14745600 exceeds 10% of free system memory.
1540/1540 [==============================] - 427s 277ms/step - loss: 1.0436 - accuracy: 0.5040 - val_loss: 0.4217 - val_accuracy: 0.8663
Epoch 2/20
1540/1540 [==============================] - 385s 250ms/step - loss: 0.6796 - accuracy: 0.7817 - val_loss: 0.2104 - val_accuracy: 0.9308
Epoch 3/20
1540/1540 [==============================] - 373s 242ms/step - loss: 0.4874 - accuracy: 0.8442 - val_loss: 0.1131 - val_accuracy: 0.9711
Epoch 4/20
1540/1540 [==============================] - 370s 240ms/step - loss: 0.3747 - accuracy: 0.8771 - val_loss: 0.0979 - val_accuracy: 0.9689
Epoch 5/20
1540/1540 [==============================] - 353s 229ms/step - loss: 0.3130 - accuracy: 0.9032 - val_loss: 0.0813 - val_accuracy: 0.9742
Epoch 6/20
1540/1540 [==============================] - 346s 225ms/step - loss: 0.2656 - accuracy: 0.9193 - val_loss: 0.0400 - val_accuracy: 0.9843
Epoch 7/20
1540/1540 [==============================] - 361s 234ms/step - loss: 0.2273 - accuracy: 0.9278 - val_loss: 0.0348 - val_accuracy: 0.9887
Epoch 8/20
1540/1540 [==============================] - 369s 239ms/step - loss: 0.2003 - accuracy: 0.9339 - val_loss: 0.0314 - val_accuracy: 0.9921
Epoch 9/20
1540/1540 [==============================] - 333s 216ms/step - loss: 0.2012 - accuracy: 0.9384 - val_loss: 0.0501 - val_accuracy: 0.9832
Epoch 10/20
1540/1540 [==============================] - 328s 213ms/step - loss: 0.1904 - accuracy: 0.9417 - val_loss: 0.0300 - val_accuracy: 0.9889
Epoch 11/20
1540/1540 [==============================] - 297s 193ms/step - loss: 0.1800 - accuracy: 0.9448 - val_loss: 0.0208 - val_accuracy: 0.9908
Epoch 12/20
1540/1540 [==============================] - 287s 186ms/step - loss: 0.1717 - accuracy: 0.9474 - val_loss: 0.0259 - val_accuracy: 0.9911
Epoch 13/20
1540/1540 [==============================] - 237s 154ms/step - loss: 0.1628 - accuracy: 0.9521 - val_loss: 0.0229 - val_accuracy: 0.9921
Epoch 14/20
1540/1540 [==============================] - 201s 131ms/step - loss: 0.1428 - accuracy: 0.9574 - val_loss: 0.0258 - val_accuracy: 0.9905
Epoch 15/20
1540/1540 [==============================] - 218s 141ms/step - loss: 0.1509 - accuracy: 0.9548 - val_loss: 0.0319 - val_accuracy: 0.9895
Epoch 16/20
1540/1540 [==============================] - 193s 125ms/step - loss: 0.1354 - accuracy: 0.9588 - val_loss: 0.0202 - val_accuracy: 0.9934
Epoch 17/20
1540/1540 [==============================] - 194s 126ms/step - loss: 0.1407 - accuracy: 0.9565 - val_loss: 0.0378 - val_accuracy: 0.9879
Epoch 18/20
1540/1540 [==============================] - 186s 121ms/step - loss: 0.1347 - accuracy: 0.9598 - val_loss: 0.0181 - val_accuracy: 0.9939
Epoch 19/20
1540/1540 [==============================] - 187s 121ms/step - loss: 0.1278 - accuracy: 0.9629 - val_loss: 0.0243 - val_accuracy: 0.9937
Epoch 20/20
1540/1540 [==============================] - 186s 121ms/step - loss: 0.1152 - accuracy: 0.9658 - val_loss: 0.0104 - val_accuracy: 0.9939
Model Saved
```

Figure 8. Epochs for CNN model trained.

Compared to hardware-based systems that use flex sensors, this solution is more economical and efficient since it uses a deep learning model implemented in Python and just needs a laptop webcam and a computer system for deployment. The only equipment required for maintenance is a working camera for sign detection. Using a US dictionary, the algorithm makes recommendations depending on the term that is now being translated.

When a person with speech or hearing impairments wants to sign "ELECTRON" to a normal person, for example, the system suggests letters that correspond to the first alphabets typed, such "ELECTOR," "ELECTRON," "ELECTRA," or "ELECTROCUTE." The Hunspell library's recommendations assist in identifying and fixing spelling errors that may result from improper alphabet recognition or a lack of word spelling expertise. One novel characteristic not seen in previous studied models is the ability to mix several characters to make words and so construct whole sentences.

V. Conclusions And Future Work:

Sign language recognition using Convolutional Neural Networks (CNNs) is a major development in artificial intelligence, especially in the field of computer vision. In this work, we have put forth a unique approach that uses CNNs in conjunction with pattern recognition approaches to identify fingerspelling in American Sign Language (ASL). After undergoing comprehensive training on a dataset consisting of 27 symbols—26 English alphabets and one "blank" sign for spacing—our CNN classifier demonstrated an astounding 96.5% accuracy rate. This high degree of accuracy not only demonstrates the effectiveness of our method but also highlights how it might lessen the need for human translators, improving accessibility for those who have hearing loss.

Furthermore, we created an intuitive graphical user interface (GUI) program to make testing and practical implementation of our classifier easier. With the use of this program, users may easily construct ASL characters, phrases, and sentences to suit their communication needs. Furthermore, the program makes many predictions for the matching word that is now produced, which facilitates more accurate and seamless communication. Our work represents a major advancement in improving accessibility and inclusion for the deaf and hard of hearing population by successfully removing the need for human translators through our CNN-based technique and offering a useful tool for ASL communication.

This paper describes the development of a real-time vision based American Sign Language (ASL) recognition system tailored for the deaf and mute (D&M) population. Our technique achieves a final accuracy of 95% on our dataset by utilizing two layers of algorithms to increase prediction accuracy, particularly for related symbols. Our system can recognize ASL symbols and show the corresponding text on the screen when hand movements are proper, background noise is minimal, and lighting is enough. Although our primary focus was on gesture detection, we recognize the importance of facial expressions in sign language, and these elements may be added in future system upgrades.

Our technology can be used in a multitude of ways. It may be used, for instance, to fill out online forms without the assistance of an interpreter or to visit government websites that do not offer sign language movies. With 70 million people worldwide estimated to have speech and hearing impairments, our initiative aims to bridge the communication gap between the public and the D&M community. In the future, the system might be operated on low-cost hardware like Raspberry Pis, and image processing capabilities could be enhanced to enable two-way communication—that is, the ability to translate between spoken and sign language.

Our future objectives include the recognition of motion-based signals and the conversion of gesture sequences into words and sentences, with the possibility of adding speech synthesis for aural output. Even with such high accuracy, there is still space for improvement and extension because our dataset lacks word-level sign language data. Furthermore, real-time efficiency is just as important for practical implementation as algorithmic accuracy.

VI REFERENCES:

[1] M.M.Gharasuie, H.Seyedarabi, “Real-time Dynamic Hand Gesture Recognition using Hidden Markov Models”, 8th Iranian Conference on Machine Vision and Image Processing (MVIP), 2013.

[2] Pradumn Kumar, Upasana Dugal, “Tensorflow Based Image Classification using Advanced Convolutional Neural Network”, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-6, March 2020, March, 2020.

[3] Xin Jia, “Image Recognition Method Based on Deep Learning”, CCDC, DOI: 978-1-5090-4657-7/17, 2017.

[4] Jiudong Yang, Jianping Li, “Application Of Deep Convolution Neural Network”, IEEE, DOI: 978-1-5386-1010-7/17, 2017.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 15 (2014) 1929-1958, 2014.

[6] Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, Dr. Dayananda P, “Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network”, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, NCAIT - 2020 Conference Proceedings.

[7] Nobuhiko MUKAI, Naoto HARADA, Youngha CHANG, “Japanese Fingerspelling Recognition based on Classification Tree and Machine Learning”, NICOGRAPH International, 2017.

[8] Qi Wang, Zequn Qin, Feiping Nie, Yuan Yuan, “Convolutional 2D LDA for Nonlinear Dimensionality Reduction”, Proceedings of the TwentySixth International Joint Conference on Artificial Intelligence (IJCAI-17).

[9] Aarthi M, Vijayalakshmi P, “Sign Language To Speech Conversion”, Fifth International Conference On Recent Trends In Information Technology, 2016.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, ‘‘Going deeper with convolutions,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1–9.

[11] Tang Z., Chen K., Pan M., Wang M. and Song Z. 2019 An augmentation strategy for medical image processing based on Statistical Shape Model and 3D Thin Plate Spline for deep learning IEEE Access 7 1-1.

[12] "Rethinking the Inception Architecture for Computer Vision" by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna.

[13] Y. Song, Y. Zhang, Y. Chen, and Y. Wu, "Lung nodule detection using 3D deep convolutional neural networks with multiple GPUs," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 10-18.

[14] K. He, X. Zhang, and S. Ren, ‘‘Deep residual learning for image recognition,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. 770–778.