Real time prediction of American Sign Language using Convolutional Neural Networks

Shobhit Sinha[1][0000−0001−7759−890X],
Siddhartha Singh[1][0000−0003−2434−2120],
Sumanu Rawat[1][0000−0003−1812−8785], and
Aman Chopra ✉[1][0000−0001−6650−4445]

[1]Manipal Institute of Technology, Manipal, Karnataka, India - 576104
{shobhit.sinha19,singh.siddhartha23,sumanurawat12,amanchopra64}@gmail.com

Abstract. The American Sign Language (ASL) was developed in the early 19th century in the American School for Deaf, United States of America. It is a natural language inspired by the French sign language and is used by around half a million people around the world with a majority in North America. The Deaf Culture views deafness as a difference in human experience rather than a disability, and ASL plays an important role in this experience. In this project, we have used Convolutional Neural Networks to create a robust model that understands 29 ASL characters (26 alphabets and 3 special characters). We further host our model locally over a real-time video interface which provides the predictions in real-time and displays the corresponding English characters on the screen like subtitles. We look at the application as a one-way translator from ASL to English for the alphabet. We conceptualize this whole procedure in our paper and explore some useful applications that can be implemented.

Keywords: American Sign Language · Convolution Neural Network · Image processing · Video processing.

1 Introduction

The National Institute on Deafness and Other Communication Disorders is an organization that conducts biomedical research processes of hearing, balance, smell, taste, voice, speech, and language. Their statistics dictate that within the USA, 3 in every 1000 people are born with a certain degree of impaired hearing capacity in one or both ears. Around 30 million people above the age of 12 years have hearing loss in both ears. A number of solutions have been developed to ease communication without being able to hear clearly.

The American Sign Language provides a wide array of symbols, actions, and movements which enables communication without sound. It has a vast scope, hence for the purpose of conceptualization, we have concentrated our research on the ASL Alphabet dataset only. In this paper, we develop a Convolutional Neural Network model using the Keras library. After making sure that the CNN

model is trained with a high accuracy metric, we host our model to enable real-time prediction. We provide a video input through the webcam. Each frame is then decolorized and augmented for efficient processing. Using optimal multi-processing for hosting the model on multiple CPU cores simultaneously, we are able to make a real-time prediction on live video stream input with negligible latency. This method can play a great role in developing state of the art real time sign language detection software for people who have difficulty in speaking and hearing. This could bridge the communication gap which persists between impaired people.

A brief about the objective of this paper has been mentioned in Section 2. A discussion about previous works and research done in this field has been mentioned under Section 3. Section 4 talks about the methodology which has been used to extract the most influential features. Section V explains the real-time prediction of the video and finally, the paper ends with Section 6 and Section 7 presenting the results and conclusion respectively.

2 Objective

In this paper, we have proposed a Machine Learning Software Application that is capable of acting as a one-way translator for sign languages. Our application makes real-time predictions on the 26 ASL alphabet and 3 special characters enabling anyone to understand ASL if they know the English alphabet. Due to computational limitations, we have performed our experiments on a small section of ASL. This concept can be extended to the complete ASL and even more sign languages if appropriate datasets are available. Our objective is to provide such real-time translators for sign languages and weaken the communication barrier caused due to the variety and complexity of sign languages.

3 Related work

A number of studies have been performed in the fields of gesture recognition, sign language translation, and image recognition and varied approaches have been taken to solve similar problems. In this section, we will take a look at a few of such researches. Classification of large image datasets using Convolutional neural networks is proposed in [8] . The authors have documented their submission to the image net challenge where they use 1.3 million images to train a Convolutional Neural Network and the resulting model consists of 60 million parameters. This research was a milestone in works related to the use of CNN for image recognition.

In work [14], an approach for Indian sign language recognition has been proposed. The authors have processed the input image and each entry is compared with a large database of existing images to give out the most likely output. This comparison has been sped up with the use of multiprocessing and GPU computing. The authors go on to write about how they converted their output into an audio format to facilitate communication further.

The following paper [10] delves into the field of Human-Computer Interaction on the basis of hand gesture recognition. The authors of the paper have used Kinect sensors to record the depth and color information of the subject to come to more robust and accurate conclusions about hand shape and structure. A successful experiment is recorded in the paper wherein the application is used to play a game of Rock Paper Scissors. Apart from the above-mentioned works, there has been a lot of time, effort and resources put into the research directing to understanding gestures of the human body, with the goal of making meaningful conclusions without having to type your instructions into a computer. The paper [13]used Hidden Markov Models to develop a single view camera system that recognizes hand gestures. The authors in [4] proposed a 2 level approach to solve the real-time hand gesture recognition. They used two levels of approach using Haar-like features and AdaBoost. The system presented in the paper [7] uses skin detection and hand contour comparison algorithm to detect and track a bare hand in a cluttered background via bag of features and SVM (Support Vector Machines). The paper [9] presents a software which aims to present a real time of hand gesture on basis of detection of some shape based features like orientation, Centre of mass centroid, fingers status, thumb in positions of raised or folded fingers of hand. In work [11] authors have extracted features such as Eigen vectors and Eigen values for recognition and have used principal component analysis for gesture recognition. In work [12] the authors have performed segmentation on skin pixels and have also used depth information in parallel to get better results. They have used SVM for the final classification. The paper [5] discusses sign language recognition using linguistic sub units. They have used Markov models to encode the temporal changes between sub-units and sequential pattern boosting to apply discriminative feature selection. [2] segments the hand and then uses Hilbert space-filling curve to extract a feature vector invariant to translation, scaling, and stretching. Paper [1] showcases a method for automatic recognition of finger-spelling in Indian sign language using the back-propagation algorithm through artificial neural networks.

4 Methodology

The methodology of the system has been shown in Figure-1.

4.1 Data set description

The ASL dataset [3] which is downloaded from Kaggle consists of all 26 alphabets of English language and 3 special characters namely delete, nothing and space. The dataset is divided into different directories for different alphabets and special characters. Each directory contains approx 3000 images out of which 80% is used for training and 20% is used for validation. The image size of the dataset is 200 X 200 pixels and the image resolution is 72 pixels/inches. The size of the entire dataset is 1.11 GB.

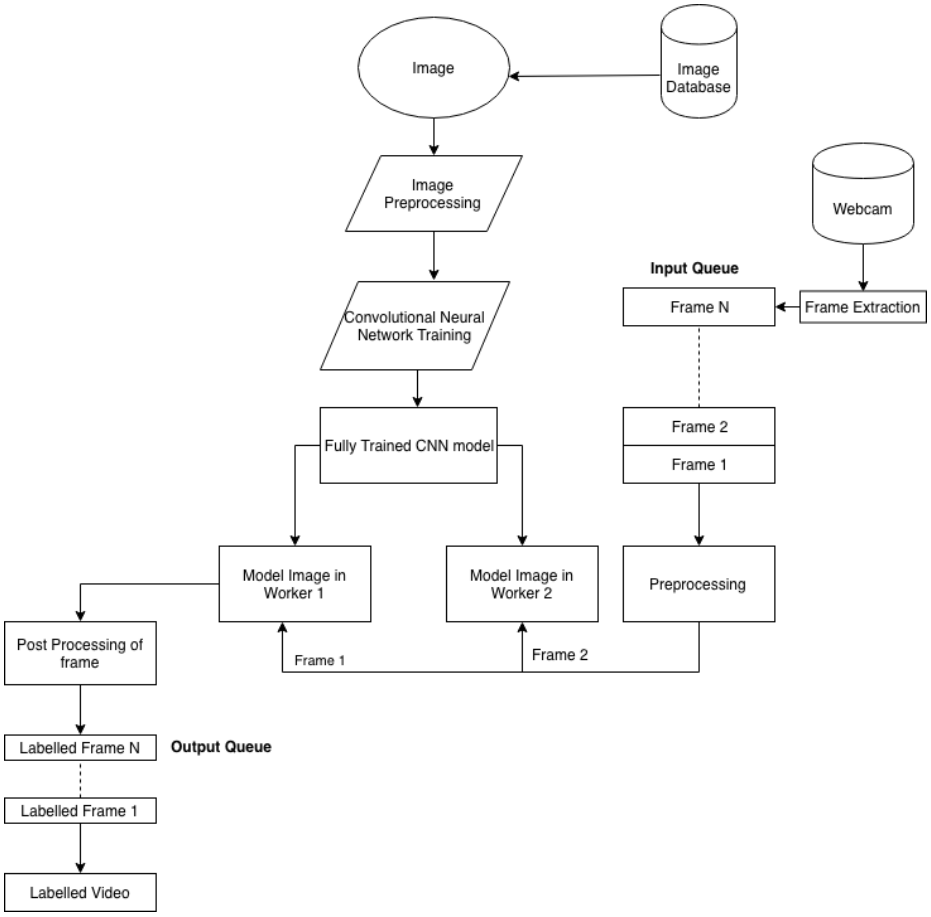

Fig. 1. Block diagram of methodology

4.2 Data pre processing

To build a good image classifier model we need a huge dataset. Having a small dataset, approx. 3000 per class in our case (87,000 images in total), can lead to poor accuracy of the model. Data augmentation is done to artificially increase the dataset. The ImageDataGenerator API of keras is used to generate real-time augmented data. Performing actions like changing the height and the width, rotating the image, zooming in and out, horizontal and vertical flip, changing sheer intensity and changing the brightness level will create artificial data for the model to improve the accuracy. Certain data preprocessed images have been shown in Figure-2. As a preprocessing step, we are reducing the dimensions of the image and scaling the image pixels to [0, 1] to make it trainable on the resources available to us.

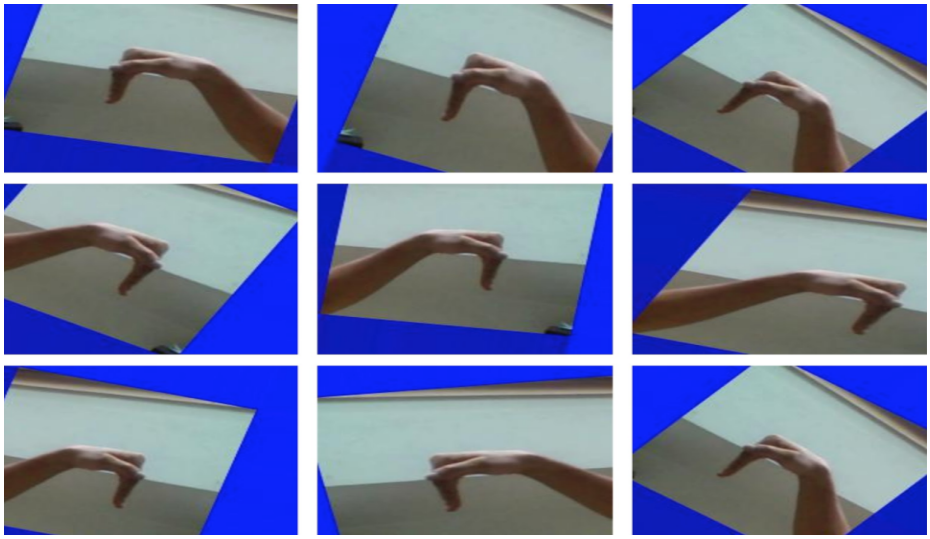

Fig. 2. Data Pre Processing steps

The ASL dataset, available at Kaggle, comes as a separate directory for every class. After extracting the class name from the paths a numpy array of labels are created which is the binarized for the CNN model. The LabelBinarizer enables us to convert the human readable string of class labels into one hot encoded labels to improve the prediction done by the model.

The data is randomly partitioned into two sections 80% for training and 20% for testing.

4.3 Model

The aim was to develop a model capable of predicting ASL alphabets and special characters with very high accuracy. Since the input of this model is an image, we decided to use the CNN architecture for this model. The model contains multiple convolutional layers, pooling layers, ReLU layers, and a fully connected layer.

Random shuffling was done while reading the data into memory to make sure that the training and testing images are not the same every time. The image dimensions are reduced and the pixels are scaled to [0,1]. Before training the dataset is randomly divided into two sections 80% for training and 20% for validation. The model consists of Conv2D layers having filters 32, 64 and 128. We purposely increased the filter size of Conv2D layers because the deeper we go in the network, the smaller the spatial dimensions of our volume, and the more filters we learn. The architectural design of the CNN model is showing in Figure-3 In this layer the padding used is same because we want the output layer should be of the same length as the input layer. The kernel size of these layers is (3,3).

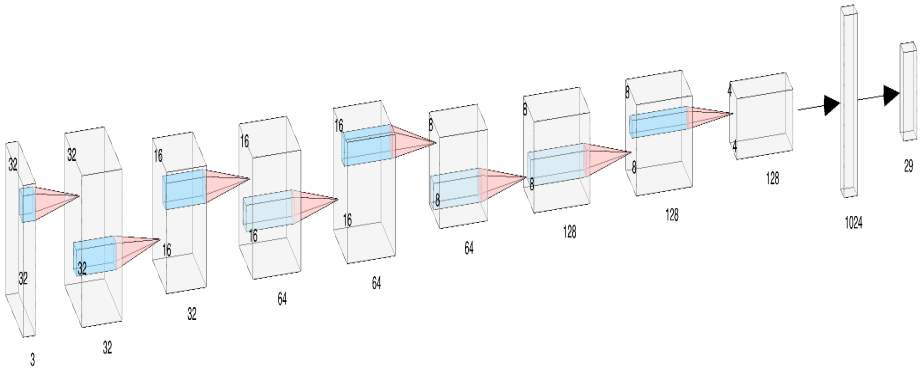

Fig. 3. CNN Architecture

Activation Layer ReLU or Rectified Linear Unit is used as the activation function in this model. The ReLU is nothing but R(x) = Max(0,x) i.e when x is less than 0 the value is 0 and for x > 0 the value is x where x is the input to the neuron. Some of the advantages of ReLU are Sparse activation, better gradient propagation,Efficient computation and Scale-invariant. We use ReLU activation layer only for the hidden layers,and for the output layer we are using SoftMax function because it gives the probability for classifying the image in different classes.

Pooling layer: The basic job of pooling layer is to provide downsampling operations. This plane is responsible for reducing the dimensionality of the feature maps thus decreasing the number of subsequent parameters to learn. Using the MaxPooling2D API we added the pooling layer in our model to reduce the dimensionality of the image by reducing the number of pixels. In our model, we are using a pool size of 3 x 3 for the initial layer and then 2 x 2 for subsequent layers. The pool size in Max-Pooling is decreased towards the end layers of the model to make sure that the spatial dimensions are not reduced too quickly.

Dropout: Dropout layer is important in a model to avoid overfitting. After each layer, were dropping some percentage of neurons to make sure that not one neuron is fully responsible for any feature. The dropout forces the layer to learn the same feature with a different set of neurons and thus avoiding the overfitting. In our model, the dropout is 25% in the intermediate layers and 50% in the fully connected layer.

Fully Connected Layer: The fully connected layer serves as a linear combinatorial function for the nonlinear activation maps produced in the last convolutional layer of the CNN. It is an easy way to learn to combine the different activations that would lead to a correct classification of the given image. Of course, due to a large number of parameters that are usually present in these fully connected layers, it is often very computationally expensive.

5 Real time prediction of video

This paper processes the video in real time. The following diagram gives an overall methodology of how the video is being processed. The model was trained using the Keras python library. After model training, the model was saved. Video processing was done using OpenCV [6] and Tensorflow [15]. For tensorflow to use the saved Keras model, the model had to be converted to tensorflow compatible model and the graph was thus extracted. The video is nothing but a sequence of a number of frames in time. Video processing can be divided into multiple image processing tasks. ASL sign in each frame was classified using the model that we developed. We made use of inbuilt webcam as the source of video for the classifier. Since the frame rate of the video was pretty high and a GPU environment wasnt available, we had to improvise and use multiprocessing and a queue mechanism to make the process as fast and smooth as possible. OpenCV was used to extract frames from the video source. Each frame required some preprocessing to be done which acted as a blocking operation for the next frame extraction operation. To solve this a queue was implemented to hold the next set of frames until the time preprocessing was being performed on the preceding frames. Since the system used had dual cores, a set of worker threads were spawned. Image of the model was loaded in both the set of workers so that they can act as individual classifiers and increase the speed of computation.

6 Results

The CNN model trained on 87,000 ASL images gave an accuracy of **96.03%**. Figure-4 depicts the Loss and accuracy vs Epoch number of the model. It can be seen that with the increasing number of epochs the accuracy increased substantially and the loss values decreased. The model was successfully used by a real time system, which could detect sign language with minimal latency. Example predictions of the system have been shown in Figure-5, 6 and 7.

7 Conclusion and Future work

In our paper, we have trained a CNN model from nothing but existing sign language images. It is easy to generate a database for projects like this and the reward is manifold. Our experiments were focused on a small part of the American Sign Language, for which we were able to produce strikingly accurate results. We produce the live output of symbols shown to a webcam, with a negligible time lag, almost as if viewing a video with subtitles. We strongly believe that our approach can prove to be useful in helping people to communicate with sign language users and understand what the person is trying to convey without having to be proficient in American Sign Language. CNN turned out to be perfect for this application.

As a part of future work, we suggest using better hardware and more computation power to come up with a more generalized model that can cover the

Fig. 4. Loss/Accuracy vs Epoch graph

Fig. 5. Results for letter O, Space and C

Fig. 6. Results for letter F, Delete and V

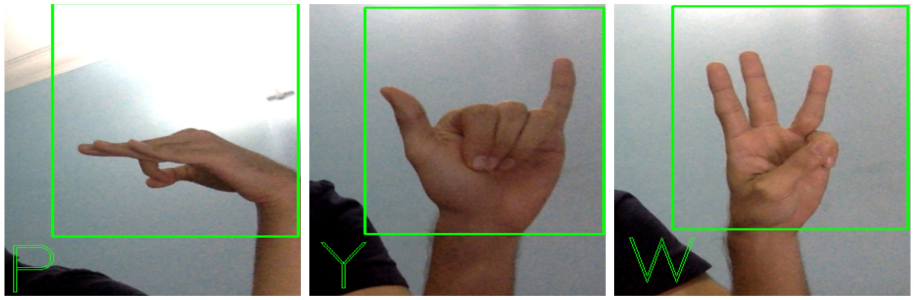

Fig. 7. Results for letter P, Y and W

complete ASL. Implementation of Active Learning in our live hosted model will enable the contribution of the everyday sign language users. It would inculcate our model with a sense of dialects and varied shapes and forms of the human hand. The authors of this paper plan to implement this approach on other sign languages and eventually host this application on public platforms such as Android and the web, free of cost. We sincerely hope that our research will help people with difficulty in speaking and listening and will improve the quality of their life.

References

1. V. Adithya, P. R. Vinod and U. Gopalakrishnan, "Artificial neural network based method for Indian sign language recognition," 2013 IEEE Conference on Information & Communication Technologies, Thuckalay, Tamil Nadu, India, 2013, pp. 1080-1085.
2. Ragab A., Ahmed M., Chau SC. (2013) Sign Language Recognition Using Hilbert Curve Features. In: Kamel M., Campilho A. (eds) Image Analysis and Recognition. ICIAR 2013. Lecture Notes in Computer Science, vol 7950. Springer, Berlin, Heidelberg
3. ASL Alphabet. Image dataset for alphabets in the american sign language. https://www.kaggle.com/grassknoted/asl-alphabet
4. Emil M. Petriu Qing Chen, Nicolas D. Georganas. Real-time vision-based hand gesture recognition using haar-like features, 2007.
5. Helen Cooper, Eng-Jon Ong, Nicolas Pugeault, and Richard Bowden. 2012. Sign language recognition using sub-units. J. Mach. Learn. Res. 13, 1 (July 2012), 2205-2231.
6. The OpenCV Library Dr. Dobbs Journal of Software Tools (2000) by G. Bradski
7. Nasser H. Dardas and Nicolas D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. IEEE transactions on instrumentation and measurment, 2011.
8. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet classification with deep convolutional neural networks In: Advances in Neural Information Processing Systems 25 Editor: F. Pereira and C. J. C. Burges and L. Bottou and K. Q. Weinberger, 1097–1105 (2012)

9. A. S. Nikam and A. G. Ambekar, "Sign language recognition using image based hand gesture recognition techniques," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-5.
10. Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. 2011. Robust hand gesture recognition with the kinect sensor. In Proceedings of the 19th ACM international conference on Multimedia (MM '11). ACM, New York, NY, USA, 759-760.
11. S. N. Sawant and M. S. Kumbhar, "Real time Sign Language Recognition using PCA," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, 2014, pp. 1412-1415.
12. Raheja, J.L., Mishra, A. & Chaudhary, A. Pattern Recognit. Image Anal. (2016) 26: 434. https://doi.org/10.1134/S1054661816020164
13. Thad Eugene Starner. Visual recognition of American sign language using hidden Markov models. Master's thesis, Massachusetts Institute of Technology, Cambridge MA, 1995.
14. Madhuri, Yellapu & g, Anitha & Mariamichael, Anburajan. (2013). Vision-based sign language translation device. 2013 International Conference on Information Communication and Embedded Systems, ICICES 2013. 565-568. 10.1109/ICICES.2013.6508395.
15. Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.