Web Based Recognition and Translation of American Sign Language with CNN and RNN

https://doi.org/10.3991/ijoe.v17i01.18585

Dhanashree Bendarkar, Pratiksha Somase, Preety Rebari (✉),
Renuka Paturkar, Arjumand Khan
Government College of Engineering, Aurangabad, Maharashtra
rebaripreety@gmail.com

Abstract—Individuals with hearing hindrance utilize gesture-based communication to exchange their thoughts. Generally, hand movements are used by them to communicate among themselves. But there are certain limitations when they communicate with other people who cannot understand these hand movements. There is a need to have a mechanism that can act as a translator between these people to communicate. It would be easier for these people to interact if there exists direct infrastructure that is able to convert signs to text and voice messages. As of late, numerous such frameworks for gesture-based communication acknowledgment have been developed. But most of them are made either for static gesture recognition or dynamic gesture recognition. As sentences are generated using combinations of static and dynamic gestures, it would be simpler for hearing debilitated individuals if such computerized frameworks can detect both the static and dynamic motions together. We have proposed a design and architecture of American Sign Language (ASL) recognition with convolutional neural networks (CNN). This paper utilizes a pretrained VGG-16 architecture for static gesture recognition and for dynamic gesture recognition, spatiotemporal features were learnt with the complex architecture, called deep learning. It contains a bidirectional convolutional Long Short Term Memory network (ConvLSTM) and 3D convolutional neural network (3DCNN) and this architecture is responsible to extract 2D spatio temporal features.

Keywords—ASL, CNN, VGG-16, 3DCNN, ConvLSTM

1 Introduction

Gestures are of utmost importance in daily life of humans as a form of nonverbal language. They are also important in sign language recognition, virtual reality, Human Robot Interaction (HCI) and Human Computer Interaction (HCI) from an industrial point of view. In this paper, we aim to recognize hand gestures specifically for hearing impaired people as when they talk with normal people, they usually use hand gestures to communicate which normal people most of the time can't understand. It's the need of the hour to have an aid for the hearing impaired to interact with normal people.

Background research states that there has been a lot of ground work to design the architecture of wearable devices which helps to recognize sign language. For example, hand gloves with a flex sensor or accelerometer sensors [1]. Further development in this direction is the use of webcam and a Kinect as an alternative to the wearable devices [2]. However, all the above-mentioned techniques are high in price and can't be used by all people. The solution to this problem is a cost-effective solution which can be accessible to anyone and will help to reduce the gap between mute-deaf and other normal people. We have developed a system which recognizes and understands static and dynamic gestures of American sign language (ASL) in this paper.

We have implemented an ASL recognition system which is a real time system capable of translating the real time video into audio and text. This enables dynamic communication. We had three objectives in mind as follows:

1. Obtain a video of the user with actions which will serve as an input.
2. Converting the frame in the video to a specific letter for static gesture recognition and classifying a couple of continuous frames to a word for dynamic gesture recognition based on classification score of neural network in both cases.
3. Creating and displaying the entire sentence.

This problem is a major challenge as there are certain assumption from computer vision perspective:

- Environmental issues (position of camera, lighting sensitivity of background)
- Detecting the boundary of a sign (the end of a sign is the beginning of the next sign)
- Coarticulation (when preceding or succeeding sign affects the current sign).

Although ASL letters recognition has been done by training Neural Networks in the past, many of them need a 3D capture feature that requires motion tracking gloves or a Microsoft Kinect. Such solutions are limited to scalability and are not so feasible because of the extra hardware requirements.

Our system contains a video pipeline in which users sign a gesture for word/number/letter via a web application. For static gesture recognition, we extract individual video frames and generate probabilities of letters (letters a through y, except j and z as these gestures need to move hand) / numbers (numbers 0 through 9) for each frame with trained CNN. The letter/number with highest probability is given as prediction only if that letter/number has highest probability for at least 5 seconds. For dynamic gesture recognition, we capture 36 continuous frames and the frames serve as input to the trained model for prediction.

2 Related Work

Over the past few decades, lots of research is going on in gesture recognition as it can be used in various application domains like smart home applications, Human Computer Interaction (HCI), gaming, medical systems, etc. Solutions proposed by different researchers are of two types: solutions based on Hardware and solutions based on Software.

Solutions based on hardware include gesture recognition using gloves, wrist bands, etc. These hardware solutions contain sensors as they are necessary to track hand movements. Google has developed wristbands which are able to recognize gestures by tracking hand movements and user is able to hear the recognized word/sentence through a mobile device as the mobile device is connected to the wristband [3]. Glove based solutions are also developed in recent years. CyberGlove was unable to detect all fingers associated with ASL gestures because of the limited number of sensors. Because of the number of sensors, CyberGlove was not able to differentiate between some gestures in which wrist positions are almost similar e.g., R and U, G and H, etc. [4]. In another proposed method, data captured by gloves is sent to the neural network and is processed for classification [5]. InerTouchHand System is proposed for Human Machine Interaction (HMI) and uses distributed inertial sensors, vibro-tactile simulators [14]. Glove based systems may give wrong results as time goes on depending on sensor quality.

Software based solutions include gesture recognition using Support Vector Machines (SVM), Neural Networks (NN), Hidden Markov Models (HMMs), etc. Software based solutions require image processing before classifying gesture images. Amazon Alexa also is able to respond to sign language gestures [5]. But in this system, you have to capture yourself repeatedly performing each sign every time you launch the site in the browser and this is a very tedious task. Also, these systems are not affordable by all people. Histogram of Gradients (HOG) and Scale Invariant Feature Transform (SIFT) features are drawn out from the images of hand gestures and are fed to Support Vector Machines (SVM) for training which is then used to classify new hand gesture images [7]. They used a dataset containing images of different orientations for accurate classification. HOG along with Local Binary Pattern (LBP) features are used together to classify hand gestures and this system attained an accuracy of 92% [8]. However, this system takes greater execution and detection time. HOG features along with Principal Component Analysis (PCA) is used to propose a solution to detect continuous Indian Sign Language [9]. This system extracts key frames from real-time continuous streaming and classifies these key frames so as to reduce classifier request rate. HOG features are not robust against different lightning conditions and SIFT features are more useful in identification tasks rather than classification tasks. Thus, to deal with classification, research is done for hand gesture recognition using Convolutional Neural Networks (CNN). Faster-RCNN having five layers of neural network is proposed to classify hand gestures with an accuracy of 99.2% [9]. 3DCNN is used to classify continuous dynamic gestures and this system achieved 83.8% [11]. This system collected data using color, depth and stereo-IR sensors. Hidden Markov Models (HMMs) are useful to capture temporal-patterns. HMM with 3-D gloves that tracks hand gestures, are used by Starner and Pentland [12]. This HMM model is using time-series data to identify hand gestures and classifies them by recognizing hand position in recent frames. They achieved 99.2% accuracy on the test set. Vision-based solution using Raspberry Pi embedded platform is used to detect hand gestures of elder people [15]. This system is trained only on dynamic gestures instead of static gestures as some elderly people might not be able to keep their hands steady because of their physical problems (numbness or shaking hands). This system is able to classify only 6 hand gestures. A system for recognizing Arabic Sign Language has also been developed [16]. But this system is developed to

recognize static gestures for Arabic alphabets only. This is because there are variations in dynamic hand gestures in different Arabic-speaking Countries.

3 Method

In this section, we depict the training and architecture of our algorithm for static and dynamic gesture recognition.

3.1 Architecture of VGG16 model

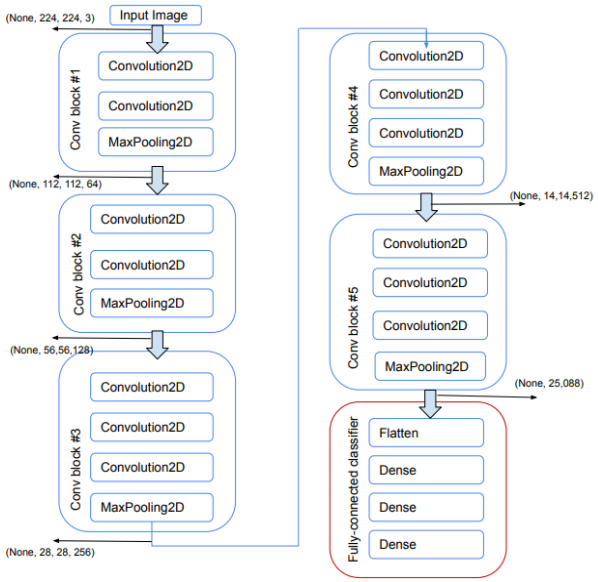


Fig. 1. VGG16 Architecture

VGG16 is made up of thirteen convolutional layers, five MaxPooling layers, two fully connected layers and one softmax layer for output. Visual Geometry Group (VGG) used VGG16 at Oxford University in the 2014 ILSVRC (ImageNet) competition. VGG16 architecture is shown in Figure 1.

Specifications of the layers from which VGG16 network is made are as follows:

- **Convolution box-1:** It has two convolutional layers with 64 filters each. Result shape: 224x224x64
- **Max pool-1:** This layer is known as Max-pooling layer which yields:112x112x 64
- **Convolution box-2:** It has two convolutional layers and has 128 filters each. Result shape is 112 x 112 x 128
- **Max pool-2:** This is Max-pooling layer which yields: 64x64x128

- **Convolution box-3**: It has three convolutional layers and has 256 filters each. Result shape is 56 x 56 x 256
- **Max pool-3**: This is also Max-pooling layer which yields: 28x28x256
- **Convolution box-4**: It has three convolutional layers with 512 filters each. Result shape: 28x28x512
- **Max pool-4**: Max-pooling layer which yields: 14x14x512
- **Convolution box-5**: It has three convolutional layers with 512 filters each. Result shape: 14x14x512
- **Max pool-5**: Max-pooling layer which yields: 7x7x512
- **Fully Connected Layer 1** (FC1): Result shape is 1x1x4096
- **Fully Connected Layer 2** (FC2): output shape is 1x1x4096
- **Output** (predictions): output shape is 1x1x1000 (For ImageNet)

3.2 Training VGG16 model for static gesture recognition

In our approach, Convolutional Neural Networks (CNNs) are used for classification of ASL letters from A to Z (except J and Z as they are dynamic gestures) and digits from 0 to 9.

For training a model for number recognition, we have used transfer-learning. Using transfer-learning, we can use a pretrained model and train it on more specific dataset to give specific results. We can do this by adjusting some of the weights of the pre-trained model and altering or reinitializing weights at bottom layers by training the model on a new dataset. By using this technique, we can train models in less time and also require less amount of data. However, disadvantage in transfer learning is due to the contrasts between the data that is originally trained and the new data which is being classified. When there are bigger differences in original data on which VGG16 model is trained and the new data which we want to classify, they often require to re-initialize or expand learning rates for more profound layers in the network.

In Keras, each layer has a parameter called “trainable”. We can set this parameter to False to freeze the weights, indicating that this layer should not be trained. For training the new model on hand gestures for digits, we have frozen the weights of the first 3 layers and added a new layer with 10 nodes as the last output layer for prediction.

Transfer learning is useful for classification tasks where less amount of data is present for training a model. Since adequate data of ASL gesture images was available, we have not used transfer-learning for training model on ASL gestures. Instead, we have trained the VGG16 model without freezing weights of all layers in the pre-trained model.

3.3 Mathematical equation for transfer learning

CNNs can learn features along with weights corresponding to each feature and thus, transfer learning is useful to work with. CNNs use loss functions to optimize parameter values. Here, softmax-based loss function is used:

$$\text{Loss}=\frac{1}{N}\sum_{i=1}^{N}-\log\left(\frac{e^{f_{i,yi}}}{\sum_{k=1}^{C}e^{z}k}\right) \quad (1)$$

$$f_i(z) = \frac{e^{z}j}{\sum_{k=1}^{C}e^{z}k} \quad (2)$$

C = total number of classes

N = total number of training examples

Equation (2) is the softmax function.

Input: Vector of features z for given training example.

Output: Flattens values of vector z to a vector of [0,1] which ultimately adds to 1.

Equation (1) calculates full softmax loss using mean loss for every training example x_i. Utilizing a softmax-based classification head permits us to yield values similar to probabilities for every ASL letter.

This varies from another well-known decision: the SVM loss. Using a SVM classification, head would bring about scores for every ASL letter that would not straightforwardly guide to probabilities. These probabilities are provided by the softmax loss and thus we can use those probabilities to classify more accurately through trained models.

3.4 3DCNN-LSTM model

3DCNN-LSTM network: 3D ConvNets are a conspicuous decision for video classification since they intrinsically apply convolutions (and max poolings) in the 3D space, where the third measurement for our situation is time. Long Short-Term Memory systems – normally called "LSTMs" – are an extraordinary sort of RNN, fit for learning long-term dependencies. LSTMs are unequivocally intended to maintain a strategic distance from the long-term dependency issue. Recalling data for extensive stretches of time is realistically their default conduct.

Short-term spatiotemporal features are learnt with the help of 3DCNN while long-term spatiotemporal features are learnt with the help of bidirectional convolutional LSTM one after the other. Afterwards, depending on the learnt 2D long-term spatiotemporal feature maps, higher level spatiotemporal features are learnt using 2DCNN for the final gesture recognition.

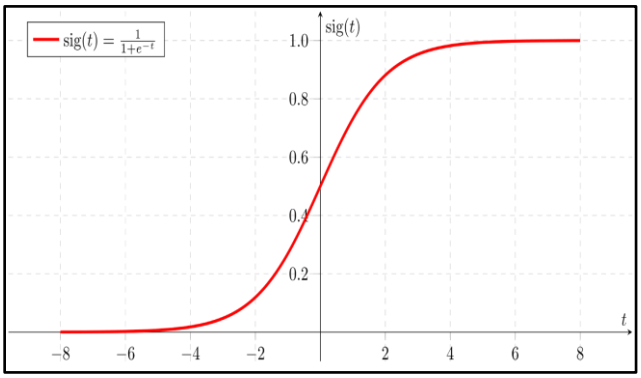
We proposed to initially learn transient spatiotemporal features utilizing a shallow 3DCNN, and afterward long-term spatiotemporal features are learnt further utilizing bidirectional convolutional LSTM, ultimately recognize gestures utilizing 2DCNN on the basis of learnt 2D spatio temporal feature maps.

LSTM: Long Short-Term Memory (LSTM) systems are a sort of recurrent neural network (RNN) and are able to learn long-term dependencies. They were presented by Hochreiter and Schmidhuber (1997), and were refined and publicized by a number of individuals in the accompanying work.

LSTM mainly consists of three gates as shown in Figure 2:

1. Input Gate
2. Forget Gate
3. Output Gate

Above mentioned gates are nothing but sigmoid activation functions. Thus, their output is between 0 or 1 and in the vast majority of the cases output value is either 0 or 1.



Graph 1. Sigmoid activation function

Sigmoid functions are used for gates since we need a gate to give just positive values and this function should tell us clearly whether we have to keep a particular feature or we have to dispose that feature.

"0" signifies the gates are blocking everything.

"1" signifies gates are permitting everything to go through it.

The equations for the gates in LSTM are:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{3}$$

$$f_t = \sigma\left(w_f[h_{t-1}, x_t] + b_f\right) \tag{4}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{5}$$

where,

σ = sigmoid function

h_{t-1}= output of the previous lstm block (at timestamp t-1)

w_x= weight for the respective gate(x) neurons

x_t= input at current timestamp

b_x= biases for the respective gates(x)

o_t= output gate

f_t= forget gate

i_t= input gate

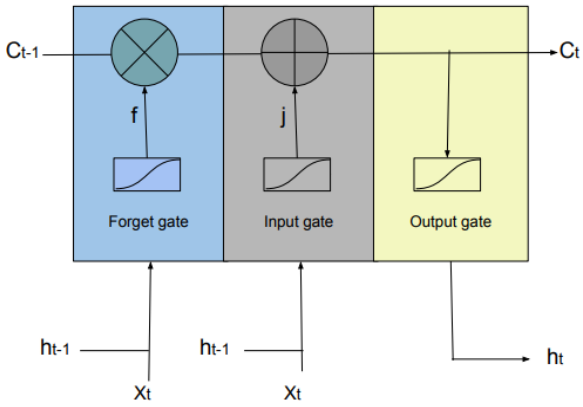


Fig. 2. LSTM memory cell

Equation (3): This equation is present at the input gate and it tells us about new data that we will be storing in the cell state.

Equation (4): This equation is present at the forget gate and it tells us what data to discard from the cell state.

Equation (5): Final output activation of the LSTM block at timestamp 't' is provided by this equation. This equation is for the output gate.

4 Dataset

ASL consists of a set of 26 signs for letters from A to Z as shown in Figure 3. We have implemented ASL recognition and number recognition (numbers from 0 to 9) using a Web application.

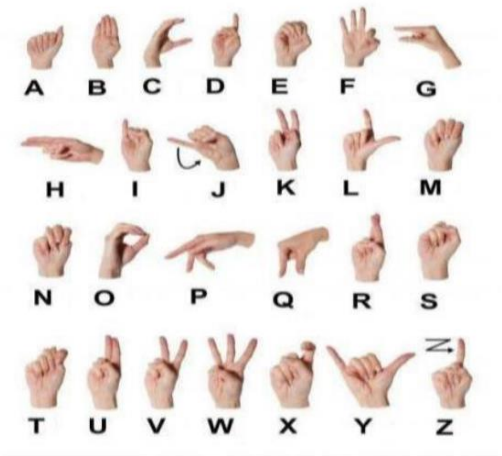


Fig. 3. American Sign Language (ASL) as appeared in Bahan, Benjamin [13]

For static gesture recognition, a dataset contains 26 classes and these classes include gestures for letters A to Y (excluding J and Z) in American Sign Language along with gestures for "space" and "del". This dataset is downloaded from Kaggle. This dataset contains total of 78000 images where each class has 3000 images each. Also, the dataset for gestures of numbers from 0 to 9 consists of total 2050 images, where each class consists of 205 images. Dataset for number gestures is also downloaded from Kaggle.

For dynamic gesture recognition we have created 240 videos of each gesture and each video is saved in the form of frames. Each video consists of 36 frames. Dataset consists of gestures for my, name, what, your, yes, no and nice to meet you. Dataset contains gestures performed by 4 different persons in different background conditions.

5 Proposed System

The system's **frontend** consists solely of HTML and JavaScript and a flask app written in python works as a server which is running in the background. Thus, the system's **backend** consists of a python script. Flask app containing a model for static gesture recognition is running continuously in the background at port 5000. Trained VGG16 models on ASL letters as well as hand gestures of numbers are loaded inside the flask app which is running continuously at port 5000 and when an image is sent from webapp to the server, the label of gesture with highest probability is predicted to the user.

Flask app containing a model for dynamic gesture recognition is also running continuously at port 8000 and a neural network consisting of 3DCNN and LSTM is loaded inside this flask app and when 36 continuous frames are sent from the webapp, the label of gesture with highest probability is predicted to the user. Flask app contains Keras, TensorFlow, Matplotlib, OpenCV, Numpy libraries and some sub packages of these libraries.

Following are the steps for sign language recognition:

1. Capture hand gesture of user and preprocess image
2. Feature Extraction and classification using trained model

5.1 Capture hand gesture of user and preprocess image

Webcam is used for capturing hand gestures. Webcam took color pictures, which were then converted into grayscale format. The main reason for sticking to grayscale was the extra amount of processing required to deal with color images. External systems like depth sensors are used by most gesture capturing softwares to detect the hand motion. However, these frameworks are more slow once in a while and space utilization can be more. To conquer this, we have made a rectangular space where the client needs to put his/her hand and perform signals.

Webapp by default opens in static mode. If we want the prediction for the gesture of number, we have to switch to that mode by clicking on a button named number. Starting the web app enables the webcam to capture images in the intervals of 1 second in static mode. Image is sent to the server running at port 5000 for prediction and resized to

224x224. Image is converted to a numpy array before passing it to the trained VGG16 model.

If we want the output of a dynamic gesture, then we will have to switch to dynamic mode in the webapp and click on the Start button to start the real time streaming. When 36 continuous frames are captured, they are sent to the web server running at port 8000 and converted to a numpy array. These frames are resized and fed to the joint 3DCNN and LSTM model.

5.2 Feature extraction and classification using trained model

a) **Static gesture recognition:** In this phase, the big problem was image capture rate. We overcame this by balancing network computation speeds with the speed with which images are sent to the network. Captured frames are sent to VGG16 fine-tuned model and the label with highest probability is temporarily set as an output. If the predicted label of the gesture is the same for 5 continuous frames then only the label is sent to the user as a prediction. The predicted label text is then converted into the system's voice. All single labels are cached into a temporary word variable continuously. When the user does the gesture for "space" then, the cached word is given to the user as an output and word variable is reinitialized to empty string for next word. The system flow for static gesture recognition is shown in Figure 4.

In case of hand gesture recognition for digits also, the gesture label is predicted only if that label occurs with the highest probability for 5 continuous frames.

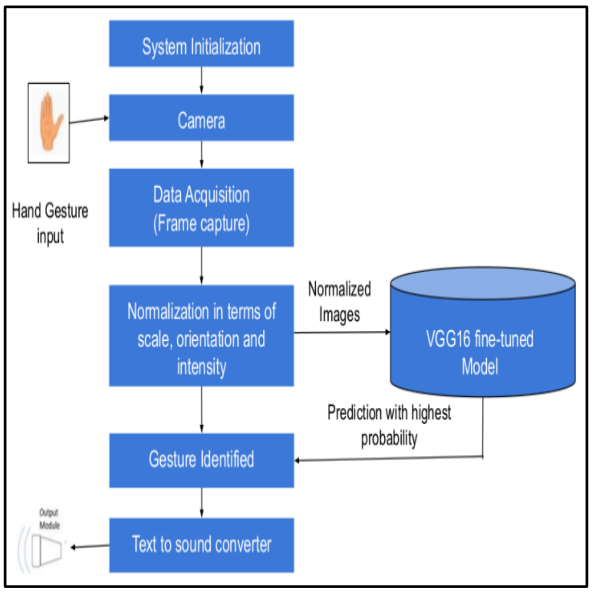


Fig. 4. Block diagram for static gesture recognition

b) **Dynamic gesture recognition:** 36 continuous frames are resized and fed to the joint 3DCNN and LSTM model. The label of gesture with highest prediction probability is sent to the user and converted to the system's voice. Predicted labels are written inside of a text area on a webapp. The system flow for dynamic gesture recognition is shown in Figure 5.

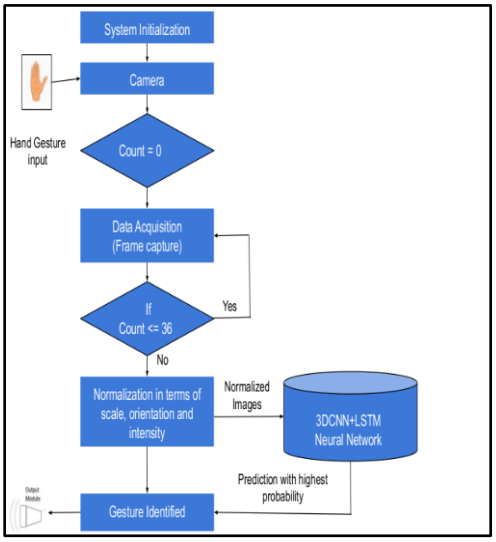
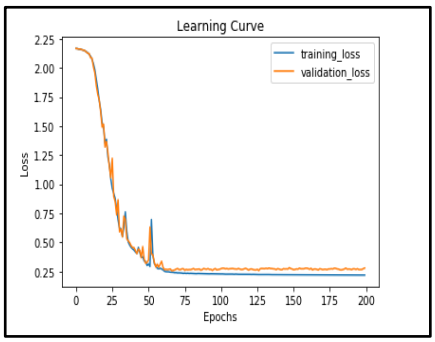


Fig. 5. Block diagram for dynamic gesture recognition

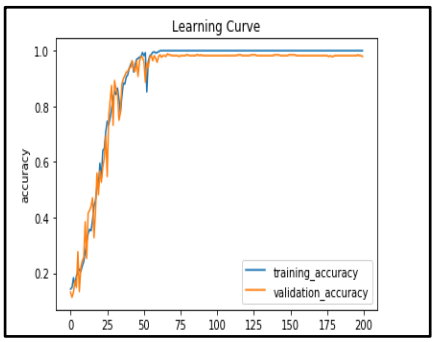
6 Results

6.1 Performance of the 3DCNN-LSTM model

Loss during training and accuracy of the model are as follows:



Graph 2. Loss curve



Graph 3. Accuracy curve

This model gives an accuracy of 98.81% on validation data.

Confusion Matrix of 3DCNN-LSTM model trained on 7 dynamic gestures is shown in Figure 6.

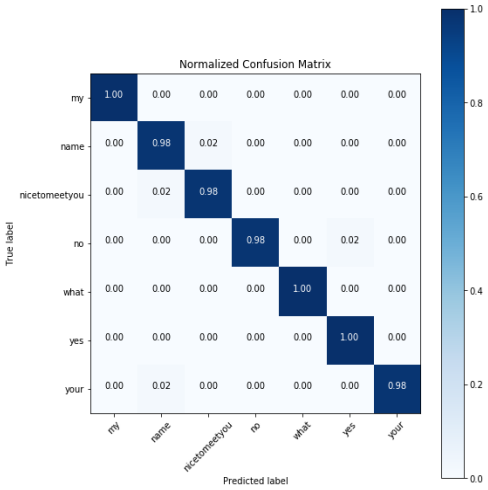


Fig. 6. Confusion matrix of 3DCNN-LSTM model trained on 7 dynamic gestures

6.2 Performance of VGG16 model trained on ASL letters from A to Y (except J and Z) and gestures for "delete" and "space"

Confusion matrix for this model is shown in Figure 7.

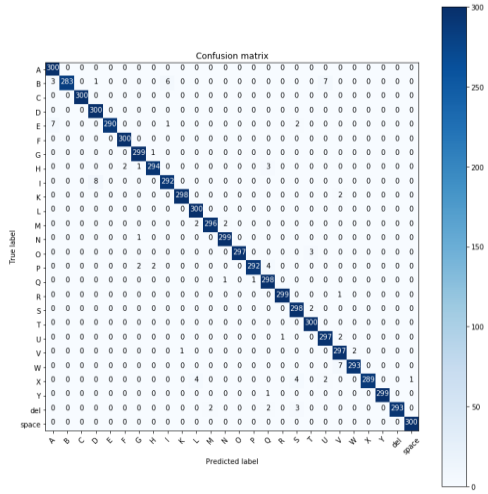


Fig. 7. Confusion matrix of VGG16 model trained on ASL gestures

The VGG16 model trained on ASL letters attains accuracy of 98.75% on test samples.

6.3 Performance of VGG16 model trained on digits from 0 to 9

Confusion matrix for this model is shown in Figure 8.

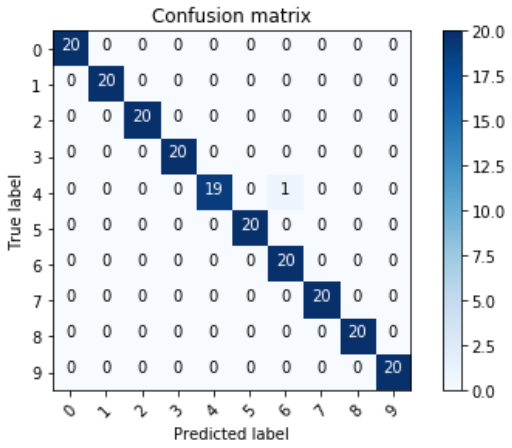


Fig. 8. Confusion matrix of VGG16 model trained on numbers gestures from 0 to 9

The VGG16 model trained on digits attains accuracy of 99.5% on test samples.

7 Output

7.1 Output of ASL letter gesture

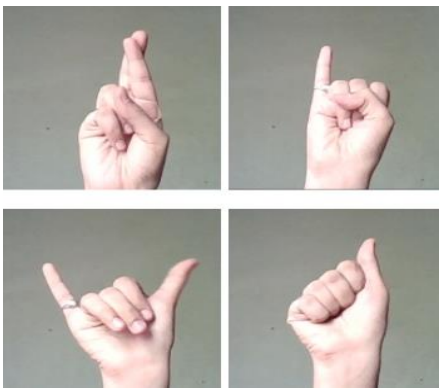


Fig. 9. Static gesture for RIYA

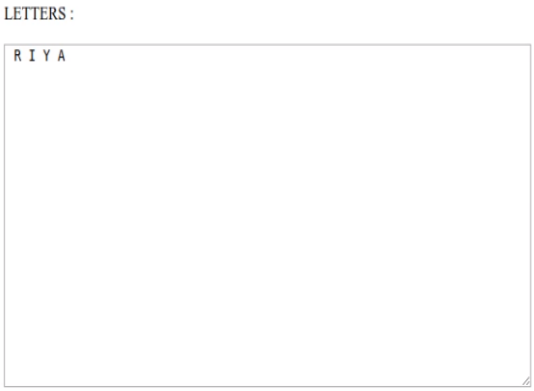


Fig. 10. Output for static gestures in figure 9

7.2 Output of number gestures

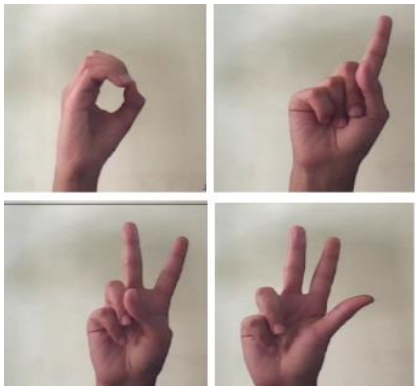


Fig. 11. Static gesture for Numbers - 0,1,2,3

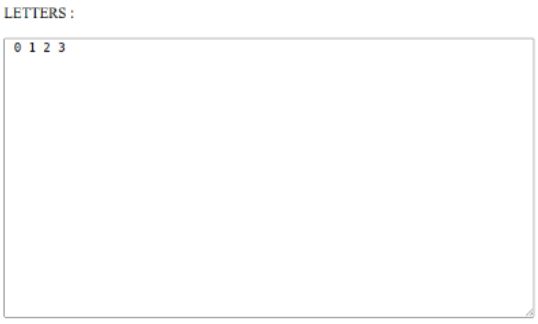


Fig. 12. Output for static gestures in figure 11

7.3 Output of dynamic gesture

Fig. 13. Dynamic gesture sequence for word "nice to meet you"

Fig. 14. Output of dynamic gesture sequence in figure 13

Output is given both in text and voice format.

8 Conclusion

For American Sign Language (ASL), we proposed a web-based sign language recognition system. It makes an interpretation of the gesture-based communication into text and predicts the labels using the pictures captured from the web camera. We proposed a novel 3DCNN-LSTM classifier for dynamic gesture recognition. Accuracy on the test dataset of ASL letters is 97.50% and 99.5% on test dataset of digits. On the validation dataset of dynamic gestures, the model attains accuracy of 98.81%. Accuracy tested against hand gestures for ASL letters captured by webcam in real time appears to be 90%. Precision of the framework can be improved by appropriate enlightenment. It differs with intricate backgrounds.

9 Acknowledgement

Authors would like to express their gratitude towards the Head of Computer Science dept. Dr. V.P. Kshirsagar for lending us motivation to work towards the project in an appropriate manner. And also, for their opportune guidance and precise counselling that helped us figure out systematic solutions.

10 References

[1] Munib, Q. "Android sign language (ASL) recognition based on Hough transform and neural networks", Expert Systems with Applications, 2000701. https://doi.org/10.1016/j.eswa.2005.11.018

[2] Davi Hirafuji Neiva, Cleber Zanchettin. "Gesture Recognition: a Review Focusing on Sign Language in a Mobile Context", Expert Systems with Application, 2018. https://doi.org/10.1016/j.eswa.2018.01.051

[3] http://gizmodo.com/electronic-wristbands-translatesign-language-into-smar-1594190782

[4] Farid Parvini, Dennis McLeod, Cyrus Shahabi, Bahareh Navai, Baharak Zali, Shahram Ghandeharizadeh,"An Approach to Glove-Based Gesture Recognition". https://doi.org/10.1007/978-3-642-02577-8_26

[5] Maria Eugenia Cabrera*, Juan Manuel Bogado, Leonardo Fermin, Raul Acuña, Dimitar Ralev, "Glove-Based Gesture Recognition System", Clawar 2012 – Proceedings of the Fifteenth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, Baltimore, MD, USA, 23 – 26 July 2012. https://doi.org/10.1142/9789814415958_0095

[6] https://blog.tensorflow.org/2018/08/getting-alexa-to-respond-to-sign-language-using-webcam-tensorflow-js.html

[7] Anita Jadhav, Rohit Asnani, Rolan Crasto, Omprasad Nilange, Anamol Ponkshe, "Gesture Recognition Using Support Vector Machine", International Journal of Electrical, Electronics and Data Communication, ISSN: 2320-2084, Volume-3, Issue-5, May-2015

[8] Houssem Lahiani, Mahmoud Neji, "Hand gesture recognition method based on HOG-LBP features for mobile devices", 22nd International Conference Engineering on Knowledge-Based Systems and Intelligent Information & Engineering System. https://doi.org/10.1016/j.procs.2018.07.259

[9] Ramesh M. Kagalkar, S.V Gumaste, "Gradient Based Key Frame Extraction for Continuous Indian Sign Language Gesture Recognition and Sentence Formation in Kannada Language: A Comparative Study of Classifiers", JCSE International Journal of Computer Sciences and Engineering.

[10] Xiaoguang Yu, Yafei Yuan, "Hand Gesture Recognition Based on Faster-RCNN Deep Learning," *Journal of Computers* vol. 14, no. 2, pp. 101-110, 2019. https://doi.org/10.17706/jcp.14.2.101-110

[11] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree and J. Kautz, "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 4207-4215, https://doi.org/10.1109/cvpr.2016.456

[12] R. Sharma et al. Recognition of Single-Handed Sign Language Gestures using Contour Tracing descriptor. Proceedings of the World Congress on Engineering 2013 Vol. II, WCE 2013, July 3 - 5, 2013, London, U.K.

[13] Bahan, Benjamin (1996). Non-Manual Realization of Agreement in American Sign Language. Boston University.

[14] Jorge Lobo, Pedro Trindade, "InerTouchHand System - iTH - Demonstration of a Glove Device with Distributed Inertial Sensors and Vibro-tactile Feedback", International Journal of Online and Biomedical Engineering (iJOE) – eISSN: 2626-8493, Vol 9 (2013). https://doi.org/10.3991/ijoe.v9is8.3385

[15] Thittaporn Ganokratanaa, Suree Pumrin, "Hand Gesture Recognition Algorithm for Smart Cities based on Wireless Sensor", International Journal of Online and Biomedical

Engineering (iJOE) – eISSN: 2626-8493, Vol 13, No 06 (2017). https://doi.org/10.3991/ijoe.v13i06.7022

[16] Yaser Saleh, Ghassan Farid Issa, "Arabic Sign Language Recognition through Deep Neural Networks Fine-Tuning", International Journal of Online and Biomedical Engineering (iJOE) – eISSN: 2626-8493, Vol 16, No 05 (2020). https://doi.org/10.3991/ijoe.v16i05.13087

11 Authors

Dhanashree Bendarkar is a student at the Department of Computer Science and Engineering, Government College of Engineering, Aurangabad, Maharashtra, India, dhanashreesb1998@gmail.com.

Pratiksha Somase is a student at the Department of Computer Science and Engineering, Government College of Engineering, Aurangabad, Maharashtra, India, pas.soma713@gmail.com.

Preety Rebari is a student at the Department of Computer Science and Engineering, Government College of Engineering, Aurangabad, Maharashtra, India, rebaripreety@gmail.com.

Renuka Paturkar is a student at the Department of Computer Science and Engineering, Government College of Engineering, Aurangabad, Maharashtra, India, renukapaturkar1998@gmail.com.

Arjumand Khan is an Assistant Professor at the Department of Computer Science and Engineering, Government College of Engineering, Aurangabad, Maharashtra, India, arjumand21.khan@gmail.com.

Article submitted 2020-09-15. Resubmitted 2020-10-16. Final acceptance 2020-10-22. Final version published as submitted by the authors.