Original Article

Sign Language Recognition Using Deep Learning

Anushka Ray[1*], Shahbaz Syed[2*], S. Poornima[3], M. Pushpalatha[4]

[1*,2*,3,4]Department of Computing Technologies, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Chengalpattu, Tamil Nadu, India.

[1*]ar5884@srmist.edu.in, [2*]ss2273@srmist.edu.in

Abstract

Sign language is the primary language of the people with speech and hearing impairment. Hearing impaired people use sign language to express themselves, participate in conversations, learn, and live as normal a life as possible. When deaf and dumb persons try to converse in sign language with those who aren't familiar with it, a problem occurs. This is where modern technology can step in. Although many existing projects have proposed methods to alleviate the problem but most of these have used external sensor or certain algorithms that do not work well under certain conditions like variation in skin color, inclusion of facial data and similarity between few signs and gestures. In our project we have used MediaPipe to solve the problem of variation in skin color as it is a very accurate hand tracking framework by Google. We have also created 3 sets of customs datasets to train 3 different deep learning models — 2 of these models are used specifically used for predicting particular groups of letters which are similar to each other. This fixes the issue arising when similar signs are encountered. The propose prototype can be used to help educate the elementary school children using which they can learn to fingerspell the American sign language alphabets and string them into basic words and learn them in a picturized manner. At the end our prototype should be able to detect the sign alphabets from the live video, print them on screen and display the image of the words formed using the sign alphabets.

Keywords: Artificial Neural Network, Computer Vision, Deep Learning, MediaPipe, Sign Language.

DOI: 10.47750/pnr.2022.13.S03.070

INTRODUCTION

SIGN language is a type of visual communication process that incorporates hand gestures, body language and facial expressions. It is the main medium of communication for people who are not only deaf and hard-of-hearing people, but also other groups of people. Individuals with disabilities including mental imbalance, apraxia of discourse, cerebral paralysis, and down condition may likewise find communication via sign language advantageous for conveying.

There is no single sign language that is used all over the world. Like spoken languages, sign languages too have been adapted naturally as people in different region used their own signs to communicate with one another, leading to a wide range of variations. There are somewhere close to 138 and 300 distinct kinds of sign language utilized all throughout the planet today.

We have decided to focus on the American Sign Language (ASL). ASL is a complete, natural language that has similar etymological properties as communicated in dialects. It is the predominant language of many deaf and hard-of-hearing persons in North America.

The fingerspelling of manual alphabets is the core of sign language. It is used to spell out proper nouns for which there are no signs. It can also be used to clarify a sign that is unfamiliar to the person who is trying to interpret it or to spell out words for signs that are unfamiliar to the person trying to communicate it. Fingerspelling signs are frequently used in conjunction with other ASL signs. ASL fingerspelling, unlike most sign languages, uses only one hand to sign alphabets. Our project classifies the 26 alphabets and delete, space and enter. These letters and gestures are recognized to form words and display the image of each word formed.

The aim of this study is to make a system using deep learning to implement a real time sign language detection system. It should be able to read the frames of our video capture in real time, display the predictions and string them together to form words and later display the images for the words formed.

LITERATURE SURVEY

This section elaborates the literature and related works for sign language recognition. The sign language recognition is a complex problem that has not completely been solved. There have been many studies to address this problem. These studies can be broadly classified into two types — vision-based techniques and sensor-based techniques.

A. Sensor-based techniques

This is one of the popular ways of hand gesture recognition i.e., via using sensors like a Leap Motion Controller (LMC)

or Kinect sensor. A system was proposed using SVM and LMC to recognize ASL alphabets [6]. The LMC sensor provided the input to the models for various hand signs. The DNN model performed much better than SVM. They arrived at the conclusion that the relative distance between tips of two adjacent fingers is a significant element in sign recognition. A low prediction accuracy due to the high similarity between certain letters and digits is one of the drawbacks. In another study Microsoft Kinect Sensor was used as it senses depth and spatial information [7]. Skeletonization algorithm was also used to processes the spatial data to find the key nodes of the hand. For similar gestures it removed the angular difference. They ran into the limitations of the Kinect as it could only display a vertical angle of 54°. As with other models they ran into a similar problem with signs that look similar. The algorithm gave a similar hand skeleton structure. This reduced the accuracy of the model. Leap motion controller and bidirectional RNN was used to accurately calculate the changes in angle while rotating wrist and palm speed between two adjacent frames of dynamic hand gestures [8]. In spite of having high accuracy and an external sensor to capture gestures it failed to predict many of them. These were the signs with similar hand contours and motion trajectories in opposite directions. Also, LMC hardware is expensive and hence difficult to access.

B. Vision-based approach

This approach uses only a camera to track the hand and later deep learning techniques for classification. An ASL recognition system was built by using no specific hardware except a camera [1]. CNN model was trained for recognizing spatial features while LSTM model was used to train on the temporal features. Combination of CNN and LSTM proved to be very accurate for detecting not just static images but also gestures. They observed that there was a loss in accuracy due to variation of face and variation in clothing. The lighting and the difference in skin tone can also reduce the accuracy of the model greatly. Another study also based on deep learning methods for static signs used the CNN algorithm [2]. They developed and tested various models using different optimizers. They came to the conclusion that more real data was needed for fine tuning the recognition system. Ref. [3] proposed a system for recognizing ASL gestures videos using a hybrid RCNN model which could learn multi-modal features as well as capture the temporal information. Despite being highly accurate when it came to detecting gestures, the system showed poor performance for signs that included facial information [3]. Another study using machine learning to detect and recognize the Bhutanese alphanumeric signs was proposed [4]. Different augmentation techniques were applied to create the first BSL hand shaped alphanumeric dataset. Different machine learning models were trained and tested with the dataset. The CNN based architecture using six convoluted layers with the batch normalization and different dropout ratios obtained better results compared to SVM, KNN and logistic regression. A prototype using modified LSTM model with skeletonization algorithm was also proposed in a study [5]. Ref. [9] presented a study which evaluated the immediate applicability of pose estimation frameworks available for sign language recognition, it was found that MediaPipe outperformed OpenPose. Using MediaPipe, the user was able to detect hand and extract keypoints even when there was hand-face interaction. In another study MediaPipe was used to extract joint coordinates from the palm and results were saved in a CSV file [10]. The SVM model was trained using the CSV file and was later used for predicting sign alphabets. Although this obtained an accuracy of over 88% for static images, there were few misclassifications when it came to detecting signs that were similar to each other.

Based on the studies done previously, our proposed system aims to resolve the many of the issues present when it comes to sign language recognition in cost-efficient way so that it is available to everyone. We have used MediaPipe to track the hand as it open-source and easy to access. With the help of MediaPipe the problem of inability to recognize sign language in real time which also includes facial data is overcome. We have used created separate deep learning models which are fine-tuned with attributes that specifically distinguish between those specific sets of letters that are similar to each other.

System Overview

The aim of the project is to build an American sign language fingerspelling detector using deep learning-based artificial neural network after extracting key points using MediaPipe. For that we trained a custom ANN model based on the relative distance between the key points extracted from the dataset. After training and testing the model, live video was fed to the ANN model to predict ASL alphabets and later these letters were strung together to form words and sentences and images of the words for is displayed.

A. Dataset

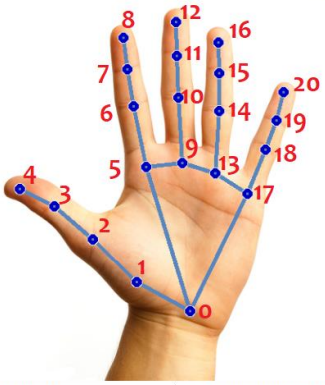
We have used 3 datasets for our project. The main dataset consists images arranged by folder according to the what they denote. Each folder denotes 26 letters in the English alphabet. In addition to the 26 folders, three additional folders are included. One folder denoting ‘space’ to later put a space between letter or words. Another folder labeled ‘enter’ consists of images of a hand. This sign is used to enter the word formed for further use. The other folder is the ‘delete’ folder. For the letters ‘J’ and ‘Z’, the images in these folders are varied to try to capture the action needed to gesture the letters.

The second dataset contains folders having images for letters ‘R’, ‘U’ and ‘V’ while the third dataset contains folders for letters ‘S’ and ‘T’. Both of these datasets are created by us by capturing these signs for these letters in real-time using OpenCV. These custom datasets are later used for training ANN model to specifically predict these groups of letters.

B. MediaPipe and Keypoint Extraction

MediaPipe is a cross-platform library developed by Google that provides ready-to-use ML solutions for computer vision tasks using nothing more than the regular webcam. MediaPipe Hands is used to detect the hand and extract the 21-3D hand-knuckle coordinates detected inside the detected hand region. The distance between the key points is calculated using the following formula and saved in a CSV file for each of the images in the 26 folders.

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$



0 - WRIST	11 - MIDDLE_FINGER_DIP
1 - THUMB_CMC	12 - MIDDLE_FINGER_TIP
2 - THUMB_MCP	13 - RING_FINGER_MCP
3 - THUMB_IP	14 - RING_FINGER_PIP
4 - THUMB_TIP	15 - RING_FINGER_DIP
5 - INDEX_FINGER_MCP	16 - RING_FINGER_TIP
6 - INDEX_FINGER_PIP	17 - PINKY_MCP
7 - INDEX_FINGER_DIP	18 - PINKY_PIP
8 - INDEX_FINGER_TIP	19 - PINKY_DIP
9 - MIDDLE_FINGER_MCP	20 - PINKY_TIP
10 - MIDDLE_FINGER_PIP	

Fig. 1. 21 3D Hand Landmarks

We have created 3 CSV files containing the distances between two coordinates calculated using the distance formula. The first CSV named ALL.csv contains the distance values for all the letters and also special classes ‘space’, ‘delete’ and ‘enter’. The other two CSV files, RUV.csv and ST.csv, contain the distance value for the following groups of letters respectively —

1) *R, U and V*

For this group of letters, the relative distance between coordinates marked by the points of thumb (in light yellow), index finger (in purple) and middle finger (in yellow) is calculated as the main difference is between the tips of the fingers.

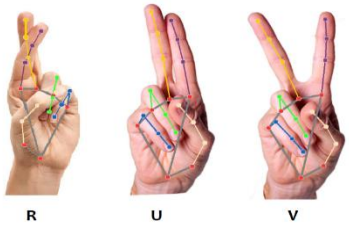


Fig. 2. Hand Landmarks for R, U and V

2) *S and T*

For this group of letters, the placement of thumb is the key factor. Thus, the relative distance is calculated by keeping the position of thumb in mind.

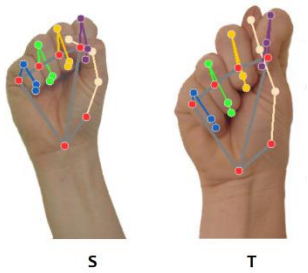


Fig. 3. Hand Landmarks for S and T

C. ANN

An artificial neural network is a computational network that is based on biological neural networks that build the structure of the human brain. Artificial neural networks, like the human brain, have neurons that are linked to each other in various layers of the network. These neurons are referred to as nodes. These links or connections are not all the same: each one may have different weights which encapsulate a network's knowledge.

ANNs can be of different types depending on their topology and learning algorithms. For our project we have used a feed forward neural network which primarily consists of three node layers: input, hidden and output layers. Except for the input nodes, each node is a neuron with a non-linear activation function. Data enters the network at the input layer and moves through each layer until it reaches the final output layer.

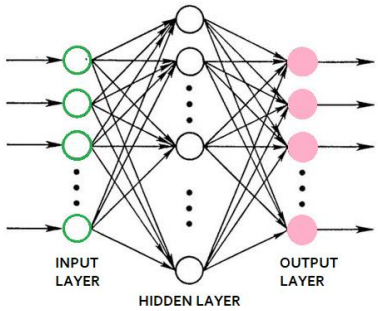


Fig. 4. ANN Architecture

This network's operation can be divided into two stages —

Learning Stage - A feed-forward ANN employs a supervised learning algorithm, which means that the neural net must know which category the pattern belongs apart from the input pattern. During the learning process, a pattern is established at the input layer. As the pattern advances through the layers of the network it transforms until it reaches the final layer. The output nodes in the final layer of the network are all

classified differently. The outputs predicted by the network are compared to what the actual classification is. The nodes in the output layer with the correct class have the highest score, with the other incorrect nodes have very low score. After comparing these scores, all weights of the connections are adjusted a little so that the next time the same sequence is presented to the nodes in the input layer, the resulting value in the output nodes correlating to the true class is higher than it was, while the scores of all other inaccurate output nodes are comparatively lower.

Classification Stage - The network's weights are fixed during the classification phase. The pattern is transformed as it passes through each layer until it reaches the output layer, where the main classification occurs by identifying the class that corresponds to the output node with the highest output value.

3) ANN Layers

A layer is a collection of neurons, also known as, nodes. An ANN has 3 main layers — input, hidden/dense and output layers. The neurons in each layer of the network serves the same task of adding the bias to the weighted sum, later performing an activation function on the output.

a) Input layer

This layer contains the nodes in charge of receiving inputs, performing the calculations using its neurons, and then transmitting the output to the subsequent layers.

b) Output layer

This layer contains nodes that are in charge of producing the final result. It receives input from the preceding hidden layer, performs calculations using its neurons, and then computes the output.

c) Hidden layer

This layer(s) sits between the input and output layers. The hidden layer's job is to compute the weighted sum of inputs by adding weights and bias and then pass the results through an activation function as output.

4) Activation Function

The activation function is defined as a mathematical computation that is used to generate a normalized output. These functions are activated when the computed result reaches the specified threshold which is pre-defined numerical value in the activation function that we are using.

There are various types of activation functions, below are the ones that we have used for our system:

a) ReLU

The rectified linear activation function or ReLU is a linear function that takes the input and outputs it directly if it is positive or gives the output as 0. Zero is the threshold for the activation. It is primarily used as the default activation function for several kinds of neural networks; it is easier to train and the model often achieves better performance. ReLU can be represented by the following:

$$f(x) = y^{+} = \max(0, y)$$

$$\Rightarrow f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$

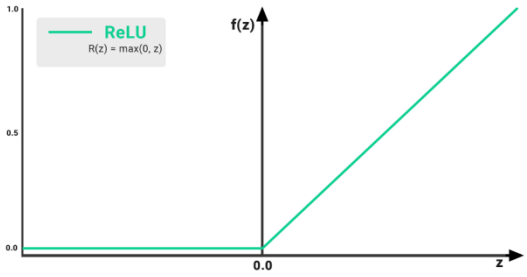


Fig. 5. ReLU Graphical Represntation

b) Softmax

The softmax activation function is used to normalize the input values into output values that are in the range [0,1]. This allows us to avoid the problem of binary classification while fitting in as many classes as possible into our neural network model.

$$\sigma(\vec{z})_i = \frac{e_i^z}{\sum_K^{j=1} e_j^z}$$

where $\vec{z_i}$ is the input vector to the softmax function while z_i are the elements of the input vector that takes integer values. Each element of the input vector is subjected to the standard exponential function e_i^z. In the normalization term $\sum_K^{j=1} e_j^z$, K denotes the number of classes. This summation function is used to ensure that the output values always sum up to 1 with each being in the range of (0, 1).

The softmax layer is useful because it transforms the real-valued scores into a normalised probability distribution. As a result, it is normally added to the final layer of a neural network.

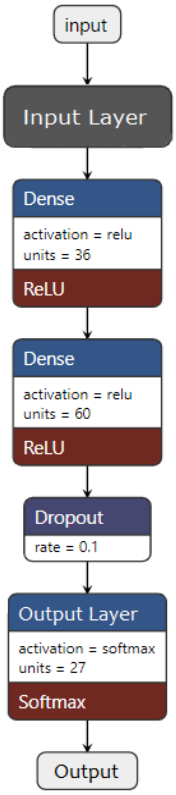


Fig. 6. ANN Model

We have used ANN as the deep learning method for the purpose of our project. This deep learning algorithm suits best for our study as we using the relative distances between the 21 keypoints of the hand after extracting it from the image data set and storing it in CSV files. Also ANN is a lightweight solution for classification problem. Our ANN models have an input layer followed by 2 hidden layers and finally an output layer. The activation functions used by the two hidden layers and the output layer are - ReLU and Softmax respectively. Also, we used an Adam optimizer and Cross Entropy Loss function to keep a track of our loss while training. We chose an optimal learning rate of 0.001 as a large LR may cause the model's performance to fluctuate over training epochs, whereas a small LR may never converge or become stuck on a subpar solution.

D. Training and Testing

The CSV file made earlier is used to split the dataset into training data and validation dataset in the ratio 90:10. The x contains the relative distances while y contains the target names. The sklearn function splits them accordingly and stores them in xtest, xtrain, ytest, ytrain variables. While training each row in the CSV is inputted to the model. Based on the output the loss criterion is calculated which is used to update the weights in the backward pass. The training accuracy and loss is stored for later use in graphs. Similarly, validation accuracy and loss are extracted. After obtaining the predicted result from the model, the predicted output and the target values are used by the sklearn function to calculate the confusion matrix.

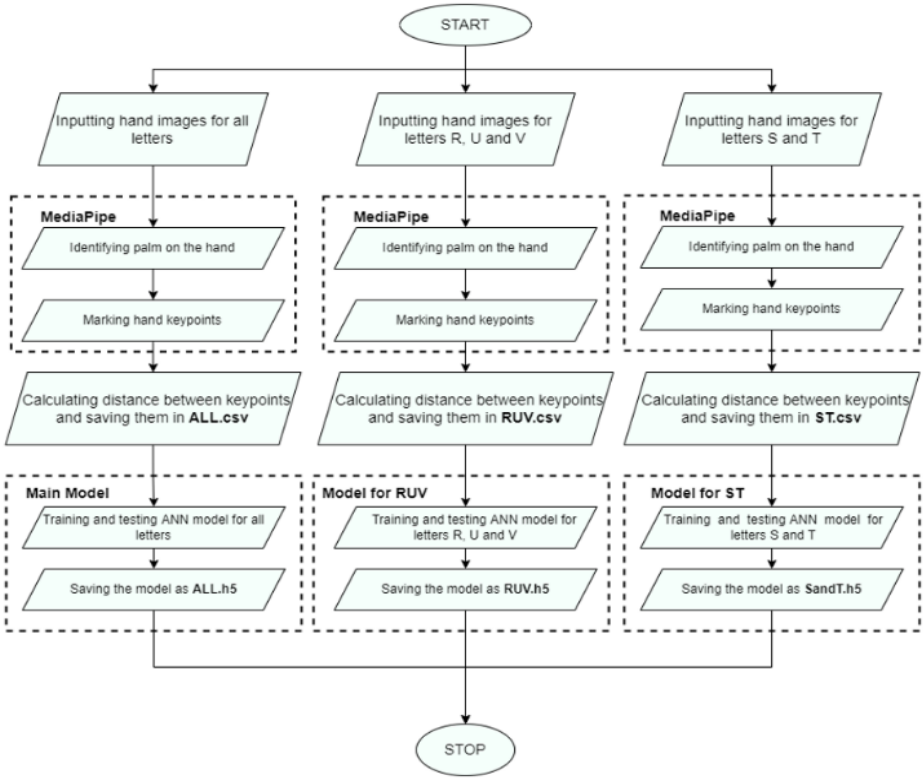


Fig. 7. Training and Testing ANN Models

We have built 3 ANN models – one for predicting all the classes and other 2 for predicting specific groups of letters i.e., "S and T" and "R, U and V". The latter 2 models have been built to train upon particular features that differentiate the letter groups amongst each other. This later helps in producing more accurate and stable results.

E. Live Detection

We have used OpenCV for the live detection of the ASL signs. The window takes continuous feed from the camera at a given framerate. The camera feed is then given as input to MediaPipe. It processes the video and detects the position of the landmarks. The landmarks are then drawn on the live video frame. The distances between the coordinates of the landmarks are calculated. These sets of distances are then fed into the model which gives the live prediction. This is then displayed on the same window.

We use a separate window to save the letters that have been correctly recognized. For this we have set a limited number of frames for which it should give a continuous prediction of the same letter. This can be experimented with and set according to the level of confidence of the model. Once a letter has been correctly recognized, it is saved and the rest of the letters are strung together similarly. This string of letters can be a word or a sentence that a user wants to search. This string is then used as a query for searching the web.

For searching we have used Google's official API. We received the API credentials by activating it in a developer project on Google's Cloud Platform. A Programmable Search Engine was created for our project. It focuses mainly on image search. Google's API also gives us the option to toggle safe search and other features such as specific image shapes/types/sizes. We also have the option to search and download images from the public domain so that we do not use copyrighted property.

Our code accesses the API and searches the query entered by the user. It then downloads the image and stores it locally. The image is then resized to fit our OpenCV window. It is then displayed at the bottom of the window.

The user can further delete the query and search for another word. The image is then replaced by the current search result.

F. Working

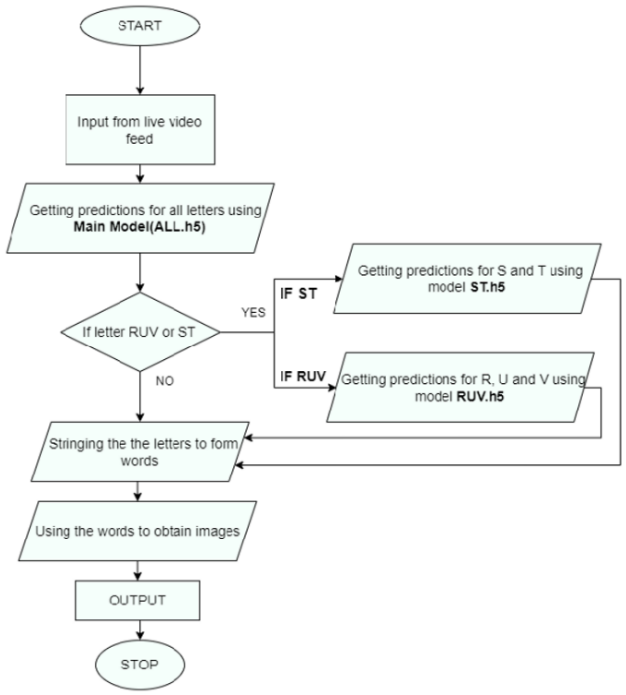


Fig. 8. System Architecture

The various components work in conjunction to give us a stable and accurate output. The live video captures the signs made by the user with the help of OpenCV. The frames are then fed into the MediaPipe framework. It recognizes the hand and outputs the coordinates of the landmarks. The distance is then calculated between particular landmarks. This data for a particular frame is inputted to the overall model. If the model predicts the sign to be 'S'/'T' or 'U'/'V'/'R' then the data containing the distances is fed into either of the two respective models. Since these models are fine-tuned with attributes that specifically distinguish between those specific sets of letters, we get a much more accurate result.

When a particular letter is predicted for more than a given number of frames, it is saved and strung together with other letters. This then forms the search query.

The query is then searched through customized Google search. The resulting image is downloaded and displayed on the window. The user can then delete the query with the delete sign. He can form another query and search again.

RESULTS

Our system can be divided into roughly 2 parts — ANN Model and Live Detection.

A. ANN Model

Our main model gives a training accuracy of 93.47% and testing accuracy of 94.07%. The accuracy of both training

and testing data is exceptional. As seen in the graphs, the validation loss is significantly lower than the train loss. This infers that the model has not been overfitted.

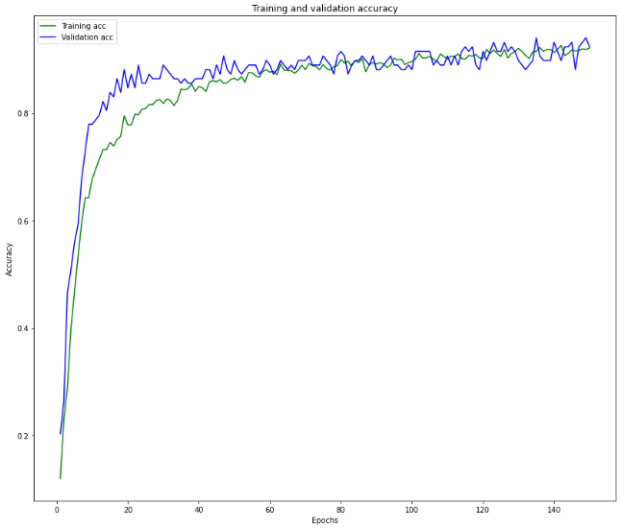


Fig. 9. Accuracy Plots

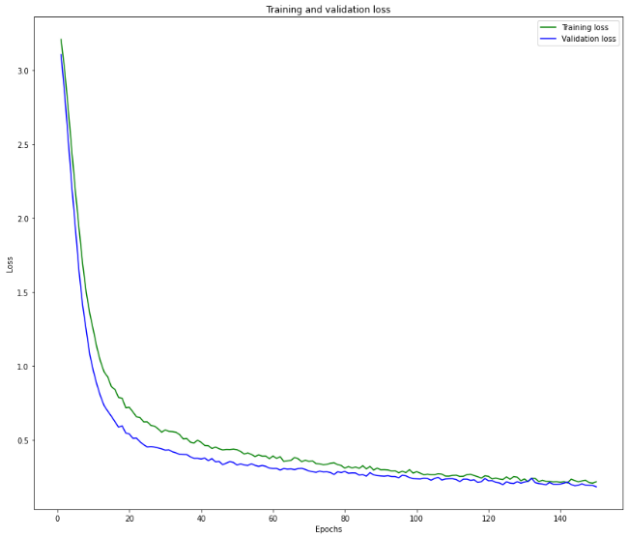


Fig. 10. Loss Plot

We have used a multi-class classification confusion matrix to evaluate the performance of our system. The matrix evaluates the actual target values against the predictions given by the model. This gives us a fuller understanding of how well our classification model is working. From our confusion matrix, we can see that there are not many misclassifications. But overall, we find that the model performs exceptionally well in classifying the majority of the sign language alphabets.

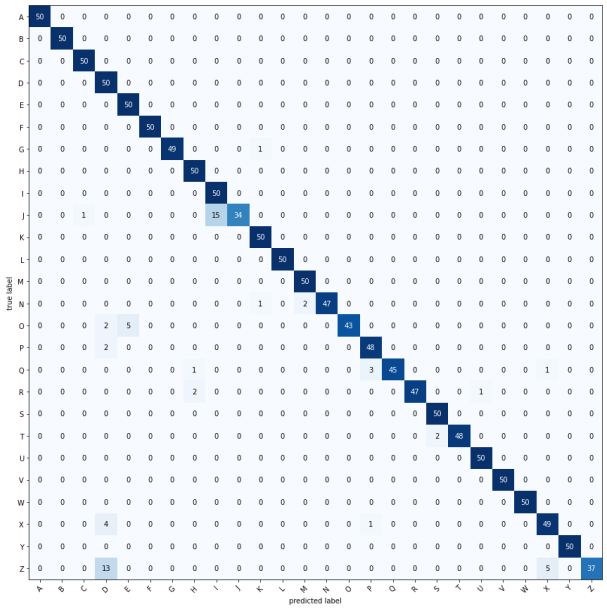


Fig. 11. Confusion Matrix

B. Live Detection

The model was able to predict the signs fed into it through live feed. The output was stable and accurate. The system can predict each sign accurately irrespective of hand size and skin color features. Our system works very well even if there is hand-face interaction. The accuracy does not drop due to inclusion of facial data. Proper lighting conditions is an important factor for getting better results.

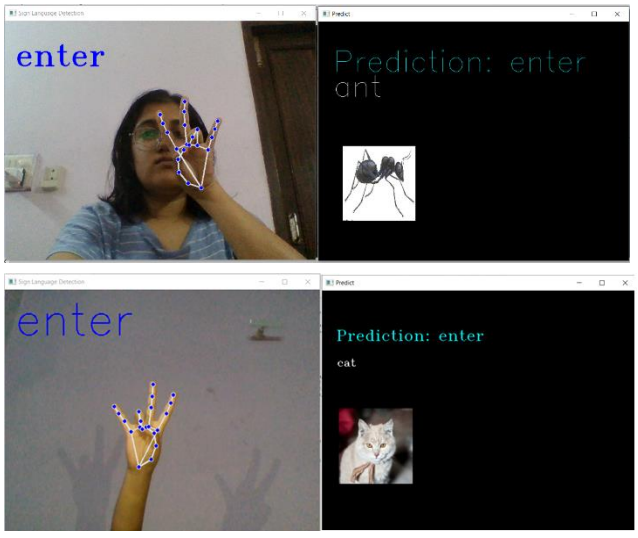


Fig. 12. Outputs

For most signs the prediction was stable. Due to the shape similarity of some alphabets (S, T) and (R, U, V) initially there was less recognition rate but this was fixed by training models specifically for the given letter sets.

If the same sign is recognized for more than 20 frames then the character is added to the string which could be further

manipulated by adding spaces or deleting to form words and sentences. Once the desired word is formed, it can be 'entered' to search and display image of that word/string. So, we have been successful at converting sign language into text form and displaying image for the word entered.

Conclusion

In this study, we have used images obtained from a web camera to recognize ASL alphabets. The coordinates of the hand joints from the images were generated using MediaPipe, and the distances between the coordinates were calculated. This data for a particular frame is then fed to the overall model for obtaining predictions. The issue of signs having similar shapes resolve by creating models that specifically differentiated those groups of signs. For static images the results obtained were highly accurate and also the prediction in the live video feed was also very accurate. The sign letters were strung together and can be manipulated by deleting letters or inserting space to form words or sentences.

In our system the word formed can be used as a search query. When the query is entered the resulting image is displayed. The query can be erased and a new query can be entered to display a new image. This feature of our system can be used as a learning tool for elementary school children where they can spell out words and see the related image and learn.

Future scope of this proposed method can be its deployment as a mobile application so that the hearing and speech disabled community can benefit from it. Also, dynamic gestures recognition to detect ASL words which use wrist movements. This project shows that it is possible to bridge the gap of communication with technology. We have been successful at converting sign language into text.

References

K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 4896–4899.

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.

Y. Ye, Y. Tian, M. Huenerfauth, and J. Liu, "Recognizing american sign language gestures from within continuous videos," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 2064–2073.

K. Wangchuk, P. Riyamongkol, and R. Waranusast, "Real-time bhutanese sign language digits recognition system using convolutional neural network," ICT Express, vol. 7, no. 2, pp. 215–220, 2021.

A. Mittal, P. Kumar, P. P. Roy, R. Balasubramanian, and B. B. Chaudhuri, "A modified LSTM model for continuous sign language recognition using leap motion," IEEE Sensors Journal, vol. 19, no. 16, pp. 7056–7063, 2019.

T. W. Chong and B. G. Lee, "American sign language recognition using leap motion controller with machine learning approach," Sensors, vol. 18, no. 10, p. 3554, 2018.

D. Jiang, G. Li, Y. Sun, J. Kong, and B. Tao, "Gesture recognition based on skeletonization algorithm and CNN with ASL database," Multimedia Tools and Applications, vol. 78, no. 21, pp. 29953–29970, 2019.

L. Yang, J. Chen, and W. Zhu, "Dynamic hand gesture recognition based on a leap motion controller and two-layer bidirectional recurrent neural network," Sensors, vol. 20, no. 7, p. 2106, 2020.

A. Moryossef et al. "Evaluating the immediate applicability of pose estimation for sign language recognition," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3434–3440.

A. Halder and A. Tayade, "Real-time vernacular sign language recognition using mediapipe and machine learning," Journal homepage: www. ijrpr. com ISSN, vol. 2582, p. 7421, 2021.

Ibrahim, S. (2022). DISCRETE LEAST SQUARE METHOD FOR SOLVING DIFFERENTIAL EQUATIONS. Advances and Applications in Discrete Mathematics, 30, 87-102.