A Real-Time Approach for Finger Spelling Interpretation Based on American Sign Language Using Neural Networks

Muhammad Saif Ur Rehman, Muhammad Rehman Shahid, Iram Shahzadi and Majid Hussain

August 20, 2023

A Real-Time Approach for Finger Spelling Interpretation Based on American Sign Language using Neural Networks

Muhammad Saif ur Rehman[1], Muhammad Rehman Shahid[1], Iram Shahzadi[2], Majid Hussain[1]

[1] Department of Computer Science, The University of Faisalabad, 38000 Pakistan
[2] Faculty of CS and IT, Superior University, Lahore 54000, Pakistan
Ms.rehman790@gmail.com, mrehman0892@gmail.com, iram5433@gmail.com, majidhussain1976@gmail.com

Abstract. Since old times, sign language has been one of the oldest and most natural forms of communication; nevertheless, because most people do not know sign language, interpreters are in high demand, Based on American Sign Language, we developed a real-time approach for finger spelling that employs neural networks. We will assess the needs and shortcoming of the created sign language system and create a model according to the needs to improve the model accuracy and usage. People who are deaf or dumb rely on sign language interpreters to communicate. However, finding experienced and qualified interpreters for their day-to-day affairs throughout their lifetime is a very difficult task and unaffordable. Data pre- processing and features extraction with gesture recognition has already applied in the past research. However, there is no research on how to create sentences using dynamic gestures and no already created model has the ability to be upgraded to optimize sentences using grammatical structure. The hand is first sent through a filter in our technique, and then it is passed through a classifier, which predicts the class of the hand motions. Using Gaussian method and classification to perfect the image quality to process the image and produce quality results with high accuracy.

Keywords: Finger Spellings, Sign Language, Feature Extraction, Artificial Neural Networks

1 INTRODUCTION

American Sign Language is the most widely used sign language in the world since the only limitation that dumb and deaf people have is a communication problem, and they cannot use spoken languages to speak correctly, thus sign language becomes the only means for them to communicate. Communication is defined as the process of transferring thoughts and messages using speech, visuals, behavior, and signals helping people understand each other without any misunderstandings. Deaf and Dumb (D&M) people utilize their hands and facial expressions to communicate to each other, express their intentions, ideas through gestures, and hand movements. More than 100 million people all around the world are unable to hear, requiring visual communication to talk with each other. ASL is the most popular sign language that most of the D&M people use and millions of people use it worldwide. Gestures are hand movements, facial expressions, other non-verbal activities used to express ideas, exchange messages and communicate with each other, andthese gestures can be easily interpreted using vision. This kind of non-verbal interaction between deaf and dumb people to communicate with each other known as sign language.

1.1 Problem Statement

Sign language is a common way among D&M people to communicate with each other and is termed as visual language. The sign language is composed of three major components as shown in Table 1.

Table 1 Sign Language Components

Fingerspelling	**Word Level Sign Vocabulary**	**Non_Manual Feature**
Used to spell words letter by letter.	Used for the majority of communication.	Facial expression, Tongue, Mouth & Body Position.

The problem is to ensure that the program can accurately understand the distinctions between these three major components. As American Sign Language uses these components for communication, the created program should be able to comprehend the non-manual features with finger spelling and word sign vocabulary. Else, it should be able to at least process the fingerspelling with word sign vocabulary and be able to work on non-manual features if future work needs to be done. The main problem we would be facing is creating a model that can accurately tell the fingerspelling and word-level vocabulary while being capable of handling non-manual features for future work.

1.2 Purpose of Project

Our study focuses on creating and training a model that can identify Fingerspelling-based hand movements that can be used to combine the texts to produce a whole sentence or word. It would also be able to differentiate between sign language vocabularies provided by ASL while being capable of being further enhanced into capable of handling non- manual features. We will use machine learning, convolutional neural network (CNN)[1], GPU acceleration, and Long Short-Term Memory (LSTM), which is a type of recurrent neural network (RNN). By using ASL characters as a data set, we will create several models using the above- mentioned techniques to get the best-desired results. Our project would help to accommodate communication between signers and non-signers.

2 LITERATURE REVIEW

A lot of study has been done on monitoring hand gestures in sign language in recent years introduced a Monte Carlo approach for hand tracking in picture hierarchy [2]. They captured hand postures and finger articulations using divide and conquer approach for their convenience. The sampling approach is used in the tracing algorithm. The production of samples at time (t+1) from samples at time (t) is an important aspect of sequential Monte Carlo. [3] also presented a unique method for predicting the hand's movement path and orientation. They provided a method for calculating movement parameters based on the basic notion of control point expulsion. For hand tracking in complicated surroundings employing rich data, introduced mean shift analysis and the Kalman filter approach[4]. They suggested a reliable approach of efficient tracing based on 3D depth maps, as rich data eliminates the problem of hands and face duplication. Various video topologies and hand forms are put to the test. The photographs are taken using a bumblebee camera model with 25 frames per second and a resolution of 240X320 pixels. The suggested technique is robust for online tracing when the mean shift iterations are performed to both hands. Marker-less hand gesture identification for American Sign Language was presented by [5].

Table 2 Competitive Analysis

Year	**Work**	**Method**	**Technology/Algorithm**	**Accuracy**
2017	Natural Sign Language Translation	RNN, CNN	Neural Network Translation	87 **%**
2018	Hierarchical LSTM for Sign Language Translation	HMM, CTC and LSTM	Hierarchal LSTM Approach	75 **%**
2019	Sign Language Translation using Deep CNN	RNN, Deep Learning	Deep Convolutional Neural Network	89 **%**
2020	Sign Language Translation with Transforms	Transformers, CSLRK, NMT	Transformation Technology	81 **%**
2020	Joint End-to-End Sign Language	CSLR, NMT, SLT, Transforme	Transformation Technology	96 **%**

	Recognition	rs		

They began by providing an overview of the language and comparing vision-based and glove-based methodologies. They studied the processes of identification of desired items (head and hands). The second step involves segmentation and tracing utilizing several color information such as RGB, YCbCr, and HASV. The tracing methodology used is a mean shift. The unique theory states locating extremes in an image's histogram to do picture segmentation. Finally, there is contemplation and gesture modelling. A dataset with a restricted number of motions and one unknown gesture rating is employed. Technique is a particle filtering-based improvised predictive Eigen tracer. They observed that their suggested work is a strong appearance based visual tracker in their trials using demonstrative gesture set, which included four basic hand shapes, 64 motions for training, and 16 extra gestures. Cam shift is used by the compiler because it utilizes the fewest CPU cycles when only a single hue is assumed in the color space model [6]. The histograms are normalized as an input, a synthetic movie with concealment and orientation is used. The results reveal that if graded histograms are not utilized, the camshaft-based tracer fails, however if they are employed, noise like sleigh and snow colors may be removed from the target histogram. The vision-based hand gesture identification system was suggested. Pseudo two-dimensional hidden Markov models were used [7]. The Kalman Filter Algorithm and color space model are utilized for tracing and detection, respectively. There are 36 American Sign Language motions in the study.
With the aid of a literature research, we determined the fundamental stages of hand gesture recognition:

- Data Collection
- Data preprocessing
- Feature extraction
- Gesture categorization

3 DEVELOPMENT METHODOLOGY

Before we describe the architecture of our project, we must understand what we are using and how we are implying these techniques.

3.1 Attribute Selection and Presentation

The presentation of a picture as a 3D matrix with picture dimensions of width and height and pixel merits of depth (1 for Greyscale and 3 for colored or RGB). Furthermore, by utilizing pixel values we were able to extract important features with CNN.

3.2 Data Set Generation

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar. We tried to discover datasets from online platform to be used in project, but we couldn't get any in the form of raw photos that met our specifications. We could only discover datasets in the form of RGB values. As such, we decided to develop datasets by ourselves.

3.3 The processes that were utilized

For creating the dataset, the system utilizes the (OpenCV) package[8]. At start, we took around 45 photographs of each ASL symbol as training data and 15 pictures for testing them Firstly, each frame is captured using the webcam. In each frame, a region of interest (ROI), that is defined and marked by red and white key points as seen below in Figure 1.

3.4 Data Set Pre-Processing

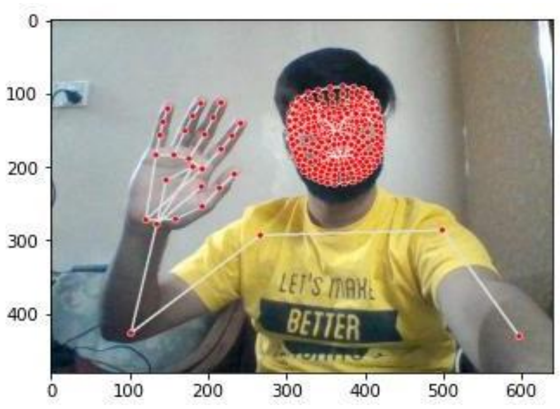


Figure 1 Raw Image ROI

Because the system that will be constructed is entirely new and has never been in any form before, and we do not have fixed needs, all requirements cannot be gathered at the time of creation. We will begin with the core needs and functionality and then implement additional requirements as they become available during development. The main needs of this product are what its consumers would anticipate from it, we have opted to apply the prototyping processing paradigm. In this model, we will collect fundamental needs or expected functionality from its consumers and then, after assessing them, we will begin working on those criteria that are obvious to us[9]. We extract our ROI, which is RGB, from the entire picture and convert to a greyscale image, as shown in Figure 2.

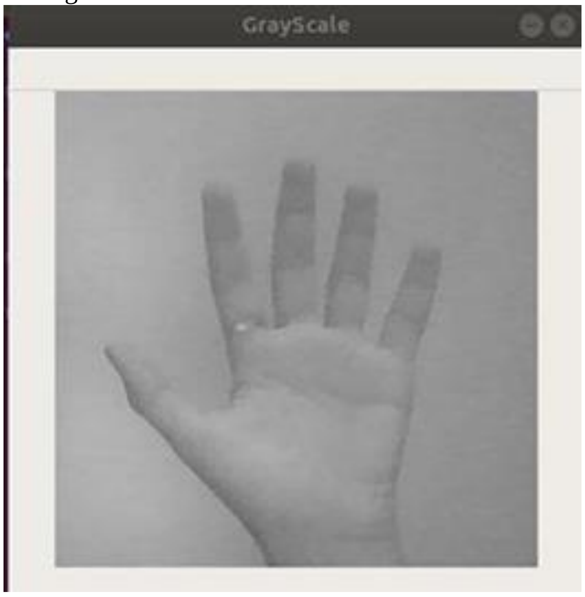


Figure 2 Gray Scale Image

In the end, we applied Gaussian blur filter corresponding to each picture, which helps us extract various features of our image. The image after applying Gaussian blur looks like below in Figure 3.

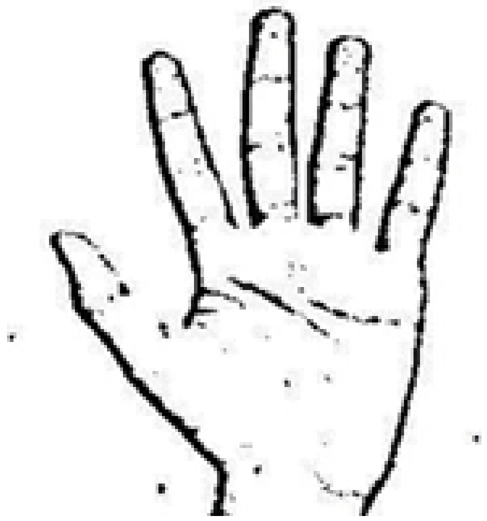


Figure 3 Gaussian Blur Effect

3.5 Gesture Classification

Our model predicts the user's final symbol using two layers of algorithms.

Algorithm Layer 1:

- To obtain the processed picture after feature extraction, apply the Gaussian blur filter and feed it to the OpenCV frame.
- This processed picture is transferred to the CNN model for recognition, and 50 frames of letter are predicted then it is counted as a word.
- The blank symbol is used to represent the space between the words.

Algorithm Layer 2:

- We recognize several sets of symbols that provide comparable consequences when detected.
- We then use classifiers designed specifically for those sets to differentiate between them.

3.6 In Layer 1: - CNN Model

First CNN Layer: The image size of the input picture is 128x128 pixels. The first CNN was processed utilizing 32 filter merits/weights (3x3 pixels each), which will result in a 126X126-pixel picture for each of the Filter-merits as shown in figure 4.
First Pooling Layer: In this layer, the pictures are down sampled utilizing max pooling of 2x2, which means we maintain the maximum value in the 2d array. As a result, our image is sampled at 63x63 pixels.
Second CNN Layer: As an input for this layer, the 63 x 63-pixel images from previous layer is used. The 32 filter weights with (3 x 3 pixels) are utilized to proves the image which produces a 60 × 60-pixel picture.
Second Pooling Layer: The generated pictures are down sampled once more using a highest pool of 2x2 and downsizing it to 30 x 30 image dimension.
First Densely Connected Layer: Then, the pictures go through fully connected layer of neurons (128), and the image is transformed into an array of 28800 values. Then, the layer gets an array of 28800 values as input and the output is transferred into the second densely layer. A dropout layer of 0.5 is used to get rid of overfitting.
Second Densely Connected Layer: As input, the first Densely Connected Layer's output is now used as an input and converting it to a connected layer with 96 neurons.
Final layer: In final layer, the input of this layer is the output of last layer, having same number of neurons as classes that are being categorized.

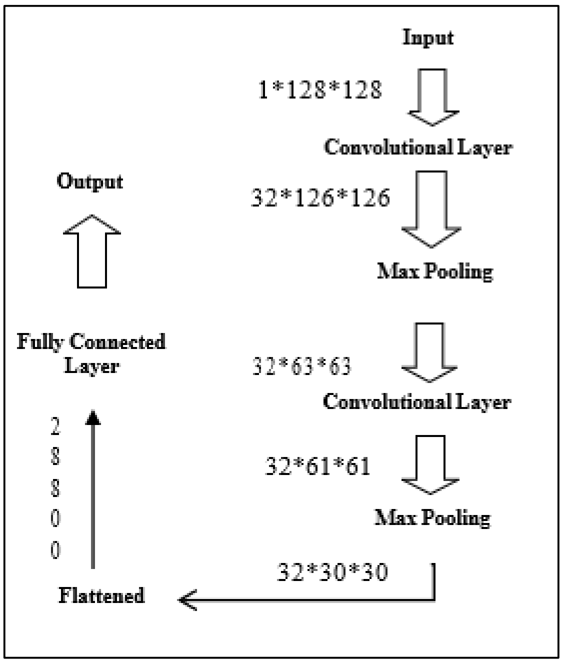


Figure 4 CNN Model Layers

Activation Function
In each layer, we utilized ReLu (Rectified Linear Unit) for each input to compute values which adds nonlinearity to the equations and aids in learning more complex characteristics. It aids in the removal of the vanishing gradient problem and speeds up training by lowering calculation duration.
Pooling Layer:
Using the RELU activation function, the max pooling is used with the input picture with a pool size of (2, 2). This results in minimizing of parameters, decreasing the computing cost and decreasing overfitting.
Dropout Layers:
Overfitting is a problem in which the network's weights get so tuned to the training examples after training that the network fails to perform properly when given new instances. This layer "drops out" a random collection of activations from that layer by setting them to zero. Even if some activations are unsuccessful, the network should be able to categorize or output a single sample correctly.
Optimizer:
The Adam optimizer was used to update the model in response to the loss function output. Adam combines the advantages of two stochastic gradient descent extensions: AdaGard and root-mean-square propagation (RMSProp).

3.7 In Layer 2:

To come as close to identifying the symbol supplied as feasible, we use two levels of algorithms to validate and forecast symbols that are more similar to each other. We observed that the following symbols were not showing correctly and were instead displaying alternative symbols throughout our testing:

- For D: R and U
- For U: D and R
- For I: T, D, K and I
- For S: M and N

To tackle the aforementioned circumstances, we created three alternative classifiers for categorizing these sets:

- {D, R, U}
- {T, K, D, I}
- {S, M, N}

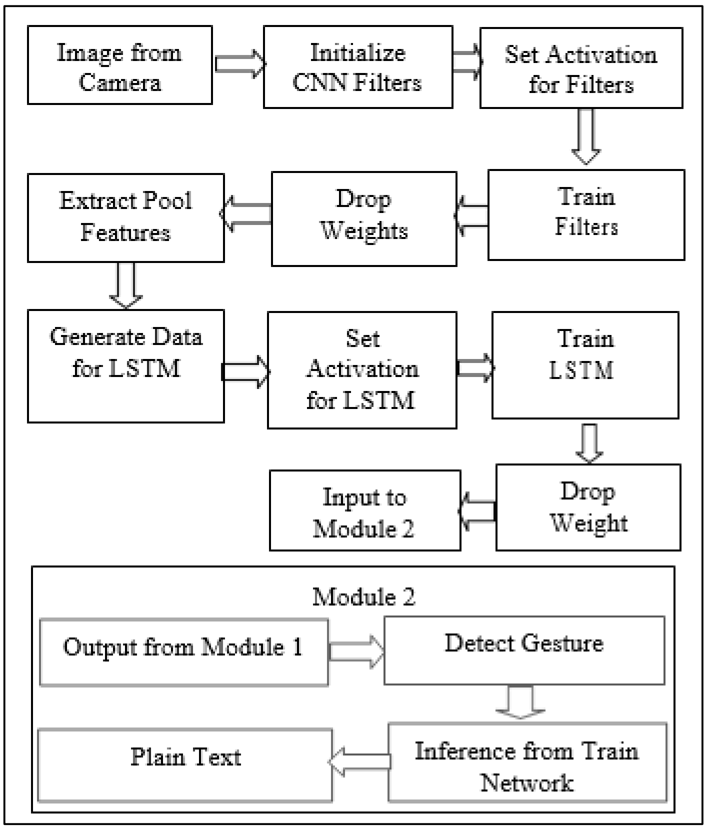


Figure 5 CNN Model Architecture

3.8 Architecture

Architecture design of our CNN model which shown in figure 5. This diagram shows that our CNN model used different following step to gives our desired results.

- **Image in RGB:** the image captured is in RGB format.
- **CNN:** A collection of learnable filters (or kernels) with a narrow receptive field but extending over the whole depth of the input volume Each filter is convoluted over the width and height of the input volume during the forward pass, computing the dot product between the filter's entries and the input and providing a 2-dimensional activation map of that filter.
- **Activation:** The activation function converts a linear action with inputs into a non-linear process.
- **Pooling:** The maximum value of a given grid is used in max pooling
- **LSTM:** It is in charge of remembering values across arbitrary time intervals (Hierarchical LSTMfor Sign Language Translation).

3.9 Activity Analysis

In the activity diagram, the model is expressed as various activities in step format shown in Figure 6. Firstly, video is captured and split into frames. Then the image is passed onto CNN for processing. If these are training images, then CNN filters are trained else CNN outputs feature vector. The feature vector is then used for generating RNN data if it belongs to train class else for prediction. These ensembles of classifiers give out the converted text.

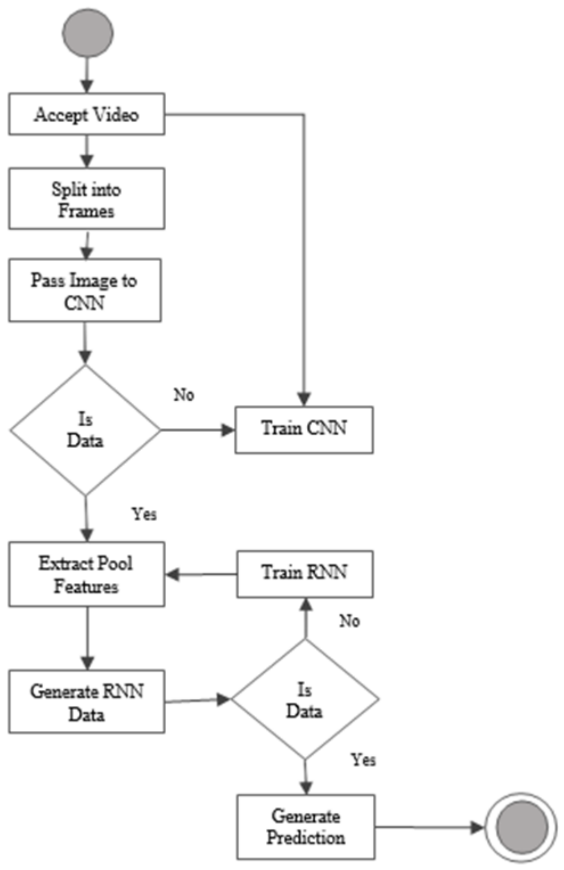


Figure 6 Activity Analysis

4 RESULTS

We convert our RGB input photographs to greyscale and use Gaussian blur to remove unnecessary noise. We utilize an adaptive threshold and scale our photographs to 128 128 to isolate our hand from the background. Similarly, we feed the preprocessed input photographs to our model for training and testing after executing all of the operations outlined above. The probability that the image will fall into one of the classes is calculated by the prediction layer. As a consequence,

the output is normalized between 0 and 1, with each class's total value equaling 1. The soft max function was used to achieve this. The output of the prediction layer will be far from the true value at first. To improve it, we used tagged data to train the networks. Cross-entropy is a classification performance statistic. It's a continuous function that's positive when the values differ from the designated values and zero when they're the same. As a consequence, we increased the cross-entropy as close to zero as feasible to maximize it. To do this, we change the weights of our neural networks at the network layer. A function in TensorFlow calculates cross- entropy. After establishing the cross-entropy function, we employed Gradient Descent to optimize it. Adam Optimizer is the most effective gradient descent optimizer. We attempted to treat the photo sequences as a time series using 3D Convolutional Networks, however they failed due to memory constraints. We tried batching the sequences to address these issues, but this did not work, and the memory constraints continued. We then proceeded to build LSTM networks and observed that a single layer LSTM network is sufficient and performs well.

We sought to retrain the Inception V3 model later since it already had pre-trained weights, and updating those weights would be significantly more efficient than starting from scratch. We noticed that the model was picking up on the motioned elements and was rather accurate (63 percent). The accuracy was roughly 81 percent after passing it through the LSTM network, and we were able to detect the motions rather well in real time. The current technique is the best since it extracts attributes and accurately detects gesture sequences while effectively recognizing gestures.

4.1 Accuracy and Loss of Trained Model

Our trained model epochs of accuracy and loss which shown in Figure 7 & Figure 8.

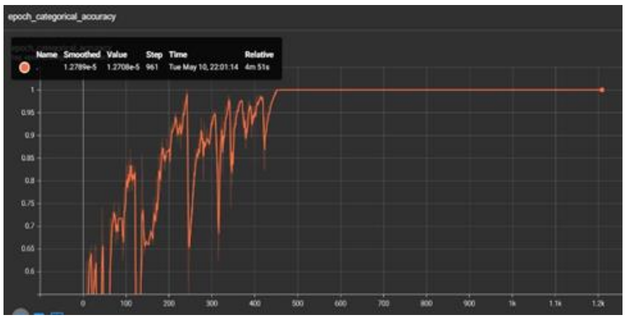


Figure 7 Accuracy of Trained Model

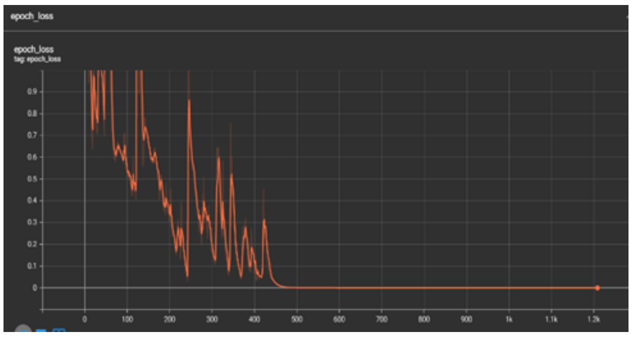


Figure 8 Loss of Trained Model

4.2 Accuracy and Loss of Test Model

Our test model epoch accuracy and loss which shown in Figure 9 & Figure 10.

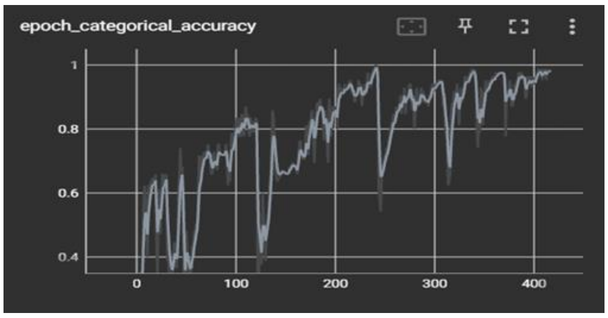


Figure 9 Accuracy of Test Model

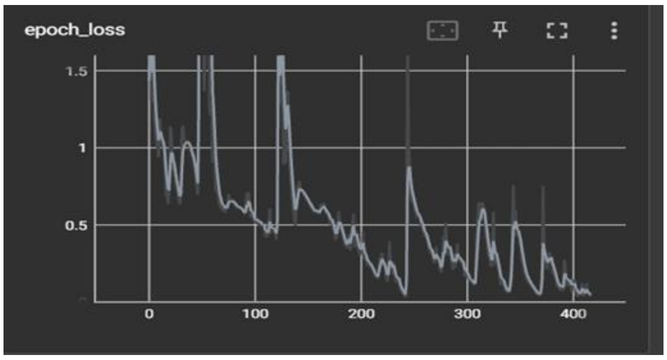
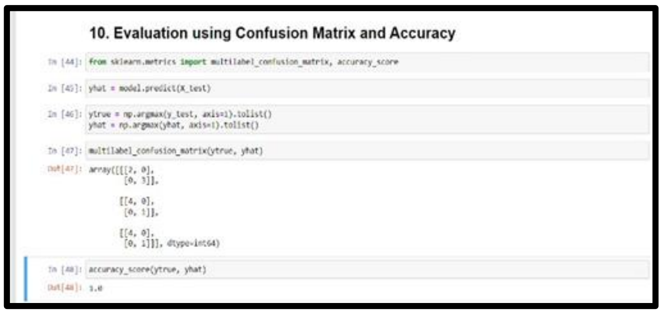


Figure 10 Loss of Test Model

4.3 Final Output of Model

Confusion matrix and output of our program which shown in Figure 11.



10. Evaluation using Confusion Matrix and Accuracy

```
In [44]: from sklearn.metrics import multilabel_confusion_matrix, accuracy_score

In [45]: yhat = model.predict(X_test)

In [46]: ytrue = np.argmax(y_test, axis=1).tolist()
         yhat = np.argmax(yhat, axis=1).tolist()

In [47]: multilabel_confusion_matrix(ytrue, yhat)
Out[47]: array([[[2, 0],
                 [0, 3]],

                [[4, 0],
                 [0, 1]],

                [[4, 0],
                 [0, 1]]], dtype=int64)

In [48]: accuracy_score(ytrue, yhat)
Out[48]: 1.0
```

Figure 11 Confusion Matrix

5 CONCLUSION

A functional real-time vision-based American Sign Language recognition for D&M individuals has been established in this paper for ASL alphabets. On our dataset, we attained a final accuracy of 98.0 percent. After constructing two layers of algorithms that check and forecast symbols that are increasingly similar to each other, we can enhance our prediction. We can recognize practically all of the symbols in this manner if they are adequately displayed, there is no noise in the background, and the lighting is acceptable.

6 REFERENCES

[1] R. H. Abiyev, M. Arslan, and J. B. Idoko, "Sign Language Translation Using Deep Convolutional Neural Networks," *KSII Transactions on Internet and Information Systems (TIIS),* vol. 14, no. 2, pp. 631–653, 2020, doi: 10.3837/TIIS.2020.02.009.

[2] A. Rezaei, M. Vafadoost, S. Rezaei, and Y. Shekofteh, "A feature based method for tracking the 3-D trajectory and the orientation of a signer's hand," *Proceedings of the International Conference on Computer and Communication Engineering 2008, ICCCE08: Global Links for Human Development*, pp. 346–351, 2008, doi: 10.1109/ICCCE.2008.4580626.

[3] M. Abdi, E. Abbasnejad, C. P. Lim, and S. Nahavandi, "3D hand pose estimation using simulation and partial-supervision with a shared latent space," *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[4] C. , W. A. , C. G. et al. Ma, "Hand joints-based gesture recognition for noisy dataset using nested interval unscented Kalman filter with LSTM network," *Vis Comput 34, 1053–1063* , 2018.

[5] R. Akmeliawati *et al.*, "Towards real-time sign language analysis via markerless gesture tracking," *2009 IEEE Intrumentation and Measurement Technology Conference, I2MTC 2009*, pp. 1200– 1204, 2009, doi: 10.1109/IMTC.2009.5168637.

[6] M. Elmezain, A. Al-Hamadi, O. Rashid, and B. Michaelis, "Posture and Gesture Recognition for Human-Computer Interaction," *Advanced Technologies*, Oct. 2009, doi: 10.5772/8221.

[7] "Hierarchical LSTM for Sign Language Translation | Proceedings of the AAAI Conference on Artificial Intelligence." https://ojs.aaai.org/index.php/AAAI/article/view/12 235 (accessed May 11, 2022).

[8] "Smoothing Images — OpenCV2.4.13.7 documentation." https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html (accessed May 11, 2022).

[9] M. M. Zaki and S. I. Shaheen, “Sign language recognition using a combination of new vision based features,” *Pattern Recognition Letters*, vol. 32, no. 4, pp. 572–577, Mar. 2011, doi: 10.1016/J.PATREC.2010.11.013.