Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

Sign Language Recognition

Shaik Khadar Sharif [1]
[1]VNR VJIET,
Department of Electronics and Communication Engineering,
Associate Professor
Hyderabad-500090, Telangana, India

Chava Sri Varshini [2], Guguloth Sreekanth [2], Gurram Hruday[2] , Mysakshi Chandu [2]
[2]VNR VJIET,
Department of Electronics and Communication Engineering, B. Tech Student
Hyderabad-500090, Telangana, India

***Abstract* - Human-Computer Interaction pushes ahead in the field of signal-based communication interpretation. Signal (Gesture) based correspondence (SL) Interpretation structure is an OK strategy to help the conference hindered people to connect with conventional people with the help of PC. When contrasted with other gesture-based communications, SL translation has got more consideration by the scientist. Some verifiable foundation, need, degree, and worry of SL are given. A vision-based hand signal affirmation system has been discussed as the hand plays a vital correspondence mode. Motion acknowledgment confronted numerous difficulties over conventional equipment-based methodology by effective usage of PC vision and example acknowledgment, it is possible to manage such a system which will be normal and recognized, by and large. In this task, the significant significance is given to hand motion development acknowledgment. These days, Interaction with the consultation disabled individuals becomes simpler when the PCs are associated with the procedure of communication and the calculations dependent on neural system models, for example, Convolutional Neural Networks (CNN) and the objective of the Convolution Operation is to evacuate the noteworthy level features, edges, from the data picture. CNN models were created for picture gathering, in which the model recognizes two-dimensional data addressing an image's pixels and concealing channels, in a strategy called highlight learning. This equal method can be applied to one-dimensional courses of action of data. The model concentrates incorporates groupings data and maps within the highlight of the plan. An exceptionally new field that was attempting to conquer the obsolete issues is HUMAN-COMPUTER INTERACTIONS which are executed by AI and profound learning**

Keywords- Convolutional Neural Networks, Deep Learning, Image Processing, Sign Language Recognition.

INTRODUCTION

Communication skills are most important for every human to share their thoughts and ideas. The word communication is a Latin word derived from SCIO means to share Communication includes the sharing of thoughts, knowledge, messages, or any kind of information. Communication is the only tool for the exchange of information through oral, writing, visual signs, or behavior. Communication is completed when the receiver receives the message and recognizes the message of the sender. Thoughts of ordinary people can be communicated to other people through speech, but for hearing-impaired people, communication becomes a major problem. The means of communication for the hearing-impaired people is done by the use of sign language. In daily communication, around 500,000 to 2,000,000 hearing-impaired people express their thought through Sign Language. Many Sign Languages have been invented for the hearing-impaired people and among those sign languages, American Sign Language (ASL) is the most used or the most referred sign language.

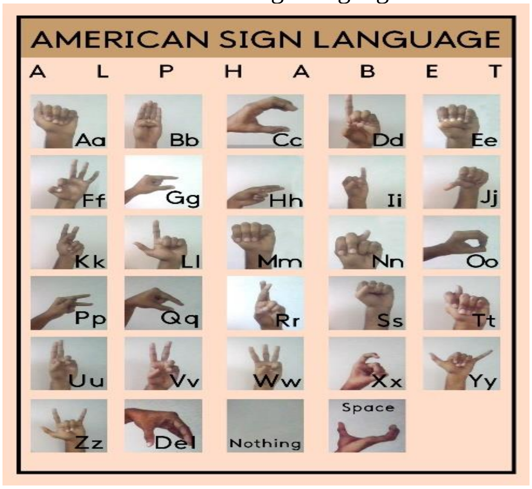


Fig: American Sign Language Gestures

II. CONVOLUTION NEURAL NETWORKS

CNN's are amazing picture handling, man-made consciousness (AI) that utilizes profound figuring out how to perform both generative and descriptive tasks, frequently utilizing machine vison that incorporates picture and video recognition, alongside recommender frameworks and Natural Language Processing (NLP). A neural system is an arrangement of equipment or potential programming designed after the activity of neurons in the human mind. Conventional neural systems are not perfect for image processing and should be given a decreased in the resolution of the image. CNN has their "neurons" arrangement similar to the frontal lobe, the region liable for preparing visual boosts in people and different creatures. The layers of neurons are orchestrated so as to cover the whole visual field maintaining a strategic distance from the piecemeal picture handling issue of conventional neural systems. CNN utilizes a framework much like a multilayer perceptron that has been intended for diminished processing necessities. The layers of a CNN comprise of an input layer, an output layer, and a hidden layer that incorporates different convolutional layers, pooling layers, fully connected layers, and normalization layers. The expulsion of constraints and increments in proficiency for image processing brings about a framework that is unquestionably progressively compelling, less complex to trains restricted for image processing, and natural language processing [1].

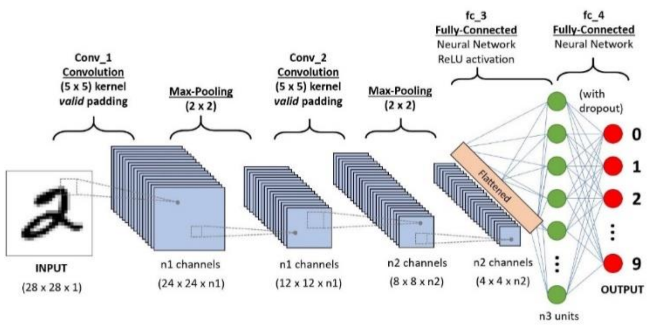


Fig: A CNN sequence to classify handwritten digits

III. LITERATURE REVIEW

This survey describes the existing and established theory and research in sign language from the American Sign language. There are thirty-six gestures of ASL which are used as a communication for the recognition and we have different algorithms to recognize them. Communication plays a vital role in human-to-human interaction which allows the people to express themselves. Different methodologies are used to recognize American gestures using different sign languages.

Zafer Ahmed ansari has done the research in the major field of Indian sign language categorization. By including finger-spelling numbers, common phrases as well as alphabets zafer classified 150 classes. A Kenect sensor is used for the dataset and along with the depth data RGB resolution of 640x480 are captured. In depth data file each pixel of the image has a value and the depth values are identified and stored in the depth data of the image. As the datasets that is the images which are collected consists of their outstretched hands. Through this outstretched hand, Zafer identified the hand with the least depth and the least depth hand is identified by masking pixels which has the depth value more than a certain threshold value. However, there were some problems during the identification of hand gestures, Zafer was unable to train the data using neural networks but he carried out the process using unsupervised learning and also by using k-algorithm. To detect various parts of the body such as torso, hand points of local maxima are used to initialize k algorithm. SIFT algorithm was used to train the data set and also to extract features.

Divya Deora has also recognized the sign symbols by using principal component analysis (PCA). The paper proposed by divya was with neural networks. They collected the data with a 3-megapixel camera and due to the poor quality of camera the result was not satisfactory and the other reason for their unsatisfactory result is the collection of a smaller number of datasets. They have collected 15 images for each sign and stored as a dataset. After the collection of datasets, segmentation of images is performed. They performed a simple boundary pixel analysis by separating the RGB into components. They got an accuracy of 60%. From this we can say that even though by combining the PCA with fingertip analysis has got the result, this result can be more accurate if they have used neural networks.

In order to recognize the sign language Lionel et al has carried out the process in two steps. They are extraction of features and action classification. This process was also carried out by using neural networks and by using Microsoft kinnect. Lionel has collected 20 different Italian gestures which were signed by 27 people of their surroundings. Microsoft kinnect is used to record the videos and he used a total of 6600 images in which 4000 images were sent to training and remaining images are sent to testing or for validating. For the learning of the model, only one side is used and for extracting the features max pooling was used. The model contains of two outputs and this are combined and given to the ANN. For the lengthening of the data CPU was used and the model was trained using GPU. The disadvantage of this process is output of the model which is trained is with less accuracy of almost 72.5%.

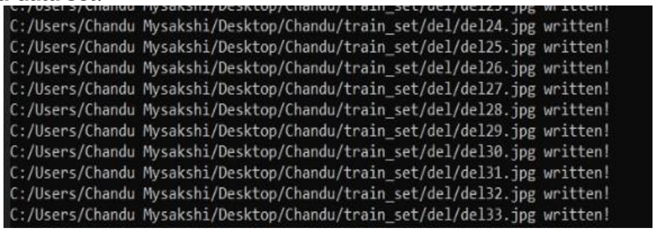
Emil M.P. et.al proposed that for image description a Haar like features are used and for the classification Adaboost is used. But, the usage of this process has many disadvantages such as it requires large number of features because of the usage of AdaBoost classifier and the other problem is detection of skin colour. Rather than this disadvantages it has the only advantage of fast computation. So this process was not preferred.

FACS, that is a facial action coding system was developed by Ekman and Friensen to code the facial expression movements and this are described using AU's which are also known as action units. Some kind of muscular basis is given to each AU and this coding process was done using a set of prescribed rules. The main disadvantage of this process is it is very time consuming. The other researcher named mase continued this process and to recognize the facial techniques he used an optical flow. But by using this process accuracy was very less.

From this survey we have decided to identify the sign gestures using neural networks. Firstly, the collection of datasets is done and then skin filtering is completed. After the process of skin filtering, hand cropping is done and after the cropping of hand the image is converted into binary image. From the binary image, features of the images are extracted and then classified into different datasets. This whole part comes under training of datasets and training of datasets is done using Recurrent neural networks (RNN) and then testing of datasets is completed.

IV. SIGN LANGUAGE DATA SET

We created a data set of the American Sign Language of all the alphabets and extra three gestures. The extra gestures are Space, Nothing, and Delete. We developed a code to capture the images of the data set when the gesture is placed in different light conditions and a different distance from the camera. We used OpenCV for capturing the images and to save them in a prescribed path. We captured the images using our laptop camera with the dimensions 200 x 200. Width and height of the image are 200, 200 pixels respectively, horizontal resolution and vertical resolution with 96 dpi and 96 dpi respectively and with Bit depth 24. For each gesture, we captured 3,000 images. In total, we captured 87,000 images of a data set.



```
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del24.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del25.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del26.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del27.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del28.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del29.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del30.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del31.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del32.jpg written!
C:/Users/Chandu Mysakshi/Desktop/Chandu/train_set/del/del33.jpg written!
```

Fig: Screenshot of creation of data set and saving them to a specified folder

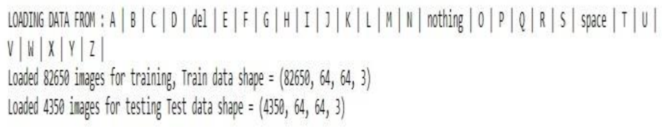
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

V. IMPLEMENTATION

Presently it is an ideal opportunity to explain the examination work with thoughts accumulated in the above ideas by receiving any of beneath appropriate methodologies: Let us execute two separate programs for the RGB data set and threshold data set.

A. Loading of the Data set

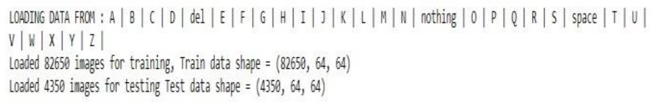
The previously created data set is loaded into the system by resizing them to 64 x 64 pixels. In one program code is written to convert the resized data set from BRG to RGB data set and split the data set into training data and testing data.



```
LOADING DATA FROM : A | B | C | D | del | E | F | G | H | I | J | K | L | M | N | nothing | O | P | Q | R | S | space | T | U |
V | W | X | Y | Z |
Loaded 82650 images for training, Train data shape = (82650, 64, 64, 3)
Loaded 4350 images for testing Test data shape = (4350, 64, 64, 3)
```

Fig: Report for splitting the converted RGB data set into training and testing set.

In another program, a code is written for converting from RGB to thresholding.



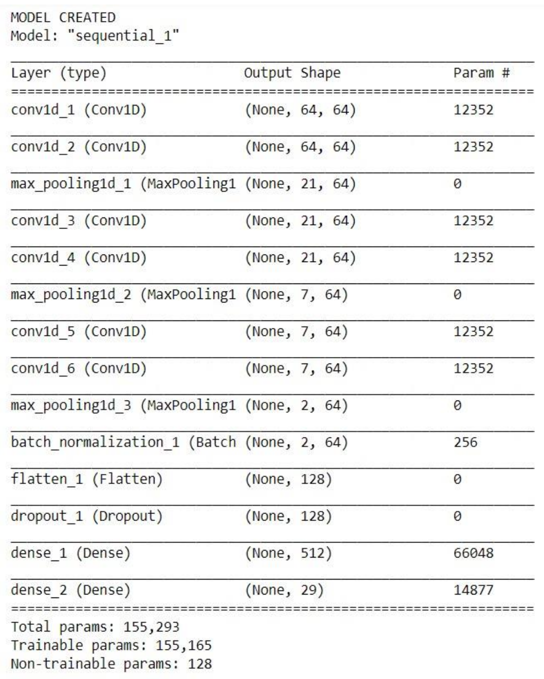
```
LOADING DATA FROM : A | B | C | D | del | E | F | G | H | I | J | K | L | M | N | nothing | O | P | Q | R | S | space | T | U |
V | W | X | Y | Z |
Loaded 82650 images for training, Train data shape = (82650, 64, 64)
Loaded 4350 images for testing Test data shape = (4350, 64, 64)
```

Fig: Report for splitting the converted threshold data set into training and testing set.

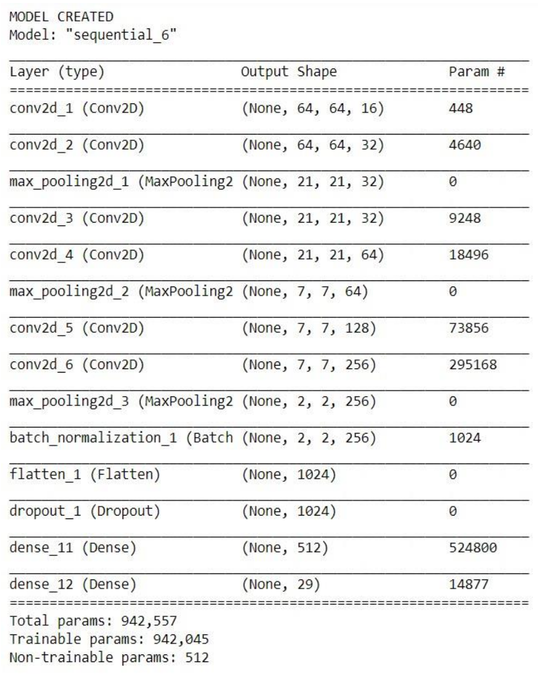
B. Creation of Model

From the above two, we got the dimension of training data as *(82650, 64, 64, 3)* and testing dataset as *(4350, 64, 64, 3)* for RGB converted images. And for dimensions of training data as *(82650, 64, 64)* and testing dataset as *(4350, 64, 64)* for thresholding converted images.

We have different dimensions so we will be using conv2d [2] to create a model for RGB converted images and conv1d [3] to create a model for thresholding converted data set.



```
MODEL CREATED
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 64, 64)            12352
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 64, 64)            12352
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 21, 64)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 21, 64)            12352
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 21, 64)            12352
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 7, 64)             0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 7, 64)             12352
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 7, 64)             12352
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 2, 64)             0
_________________________________________________________________
batch_normalization_1 (Batch (None, 2, 64)             256
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               66048
_________________________________________________________________
dense_2 (Dense)              (None, 29)                14877
=================================================================
Total params: 155,293
Trainable params: 155,165
Non-trainable params: 128
```



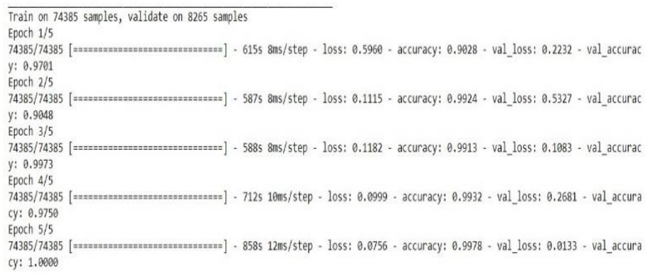
```
MODEL CREATED
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 16)        448
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 21, 21, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 32)        9248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 2, 2, 256)         1024
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_11 (Dense)             (None, 512)               524800
_________________________________________________________________
dense_12 (Dense)             (None, 29)                14877
=================================================================
Total params: 942,557
Trainable params: 942,045
Non-trainable params: 512
```

Fig: Summary for the model created using conv1d and conv2d i.e. thresholding data set and RGB data set.

C. Training and validation of created models

Now we have a training data set of 82650 images in both the cases. Among these we use 0.1% for validation and rest for training purpose i.e. we have 74385 images for training and 8265 samples for validation in both cases. We will be using the batch size 64 and 5 epochs.

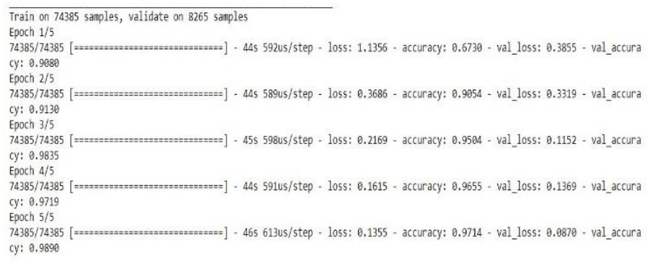
In the case of RGB converted images, the total time taken for completion of the process is 56 minutes. On average each epoch took 11.2 minutes to complete.



```
Train on 74385 samples, validate on 8265 samples
Epoch 1/5
74385/74385 [==============================] - 615s 8ms/step - loss: 0.5960 - accuracy: 0.9028 - val_loss: 0.2232 - val_accurac
y: 0.9701
Epoch 2/5
74385/74385 [==============================] - 587s 8ms/step - loss: 0.1115 - accuracy: 0.9924 - val_loss: 0.5327 - val_accurac
y: 0.9048
Epoch 3/5
74385/74385 [==============================] - 588s 8ms/step - loss: 0.1182 - accuracy: 0.9913 - val_loss: 0.1083 - val_accurac
y: 0.9973
Epoch 4/5
74385/74385 [==============================] - 712s 10ms/step - loss: 0.0999 - accuracy: 0.9932 - val_loss: 0.2681 - val_accura
cy: 0.9750
Epoch 5/5
74385/74385 [==============================] - 858s 12ms/step - loss: 0.0756 - accuracy: 0.9978 - val_loss: 0.0133 - val_accura
cy: 1.0000
```

Fig: Training and Validation report for RGB Converted Images

In the case of thresholding converted images, the total time taken for completion of the process is 3.7 minutes. On average each epoch took 44.6 micro minutes to complete.

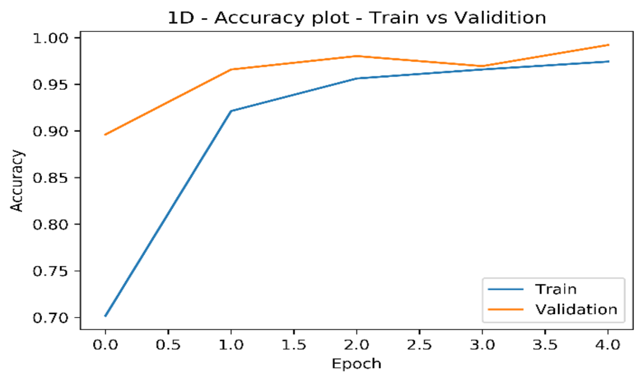


```
Train on 74385 samples, validate on 8265 samples
Epoch 1/5
74385/74385 [==============================] - 44s 592us/step - loss: 1.1356 - accuracy: 0.6730 - val_loss: 0.3855 - val_accura
cy: 0.9080
Epoch 2/5
74385/74385 [==============================] - 44s 589us/step - loss: 0.3686 - accuracy: 0.9054 - val_loss: 0.3319 - val_accura
cy: 0.9130
Epoch 3/5
74385/74385 [==============================] - 45s 598us/step - loss: 0.2169 - accuracy: 0.9504 - val_loss: 0.1152 - val_accura
cy: 0.9835
Epoch 4/5
74385/74385 [==============================] - 44s 591us/step - loss: 0.1615 - accuracy: 0.9655 - val_loss: 0.1369 - val_accura
cy: 0.9719
Epoch 5/5
74385/74385 [==============================] - 46s 613us/step - loss: 0.1355 - accuracy: 0.9714 - val_loss: 0.0870 - val_accura
cy: 0.9890
```

Fig: Training and Validation report for Thresholding Converted Images

D. Plots for the obtained metrics

The Accuracy and Loss plots for the in-Training phase and Validation phase are obtained for both the cases.



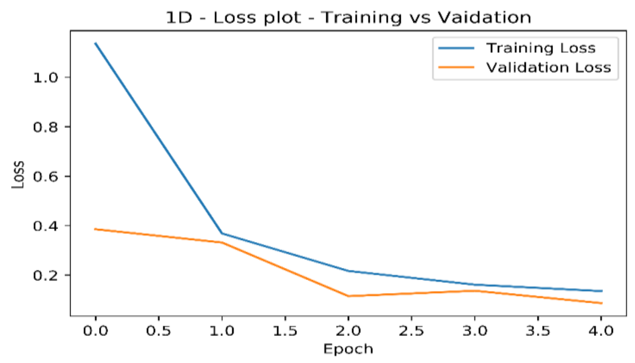
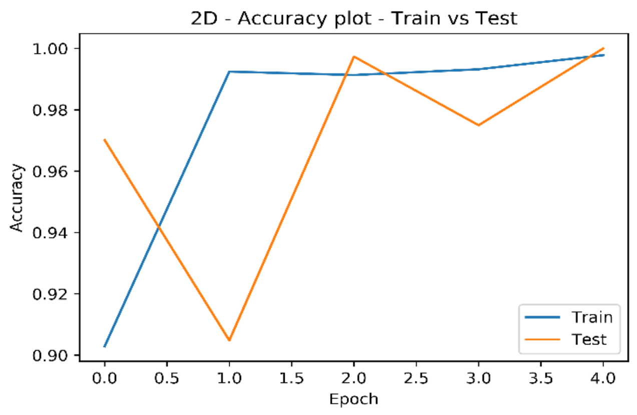


Fig: Accuracy plot and Loss plot for Thresholding converted dataset



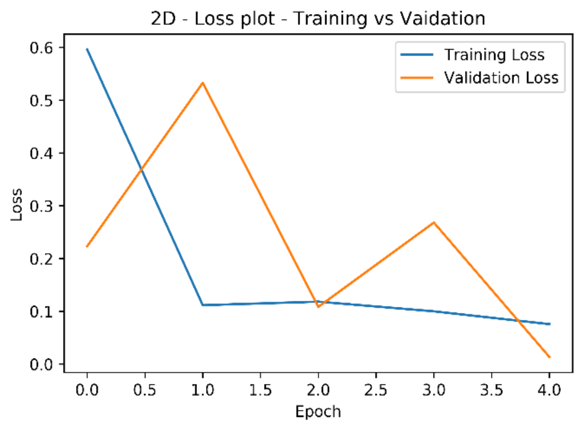


Fig: Accuracy plot and Loss plot for RGB converted dataset

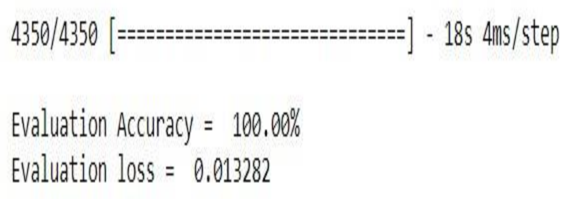
On comparing both the plots of RGB and thresholding converted datasets Accuracy plots we can say that we have stability in Thresholding plots because on average we can say that the accuracy of this plot is increasing linearly.

Whereas the accuracy of the RGB converted data set in accuracy plot is not linear but it has a zig-zag approach. That has a greater deflection of the values.

E. Giving Testing data set to the created model

We have 4350 testing set images. All these images are fed into the model to predict the output for the corresponding input gesture to model. The evaluation metrics for the testing the data set are Evaluation Accuracy and Evaluation Loss.

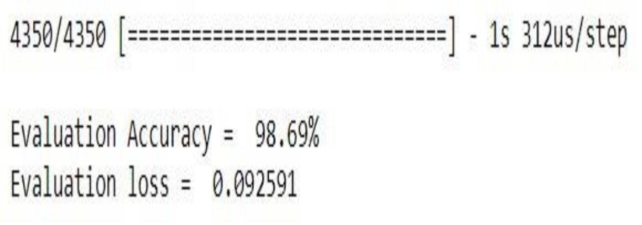
The greater the evaluation accuracy the model has a good prediction of the input given to it. Here are the evaluation metrics for both the RGB converted data set and the Thresholding converted data set.



```
4350/4350 [==============================] - 18s 4ms/step

Evaluation Accuracy =  100.00%
Evaluation loss =  0.013282
```

Fig: Testing Report for RGB converted data set.



```
4350/4350 [==============================] - 1s 312us/step

Evaluation Accuracy =  98.69%
Evaluation loss =  0.092591
```

Fig: Testing Report for Thresholding converted data set.

The time taken to complete the evaluation of the test data set is for Thresholding data set very much less when compared to RGB converted data sets. Here if we consider the evaluation Accuracy for RGB is 100% whereas in the accuracy plot we are having high fluctuations of accuracy. Whereas in the case of Thresholding converted data set we have evaluation accuracy as 98.69% and we do not have large fluctuations of accuracy in the plot.

F. Saving the model

The models created by both cases are saved in the JSON format. The weights are stored in the H5 file. The purpose of saving the model and weights are from next onwards no need of creating a model for giving the input. We can access the model directly and its weights and give input to the model to get the output. This saves the time for next use.

G. Accessing the model with live input

The saved model and weights are loaded back into systems. Now we use OpenCV for capturing the images. The Region of Interest (ROI) is cropped and the images are resized to 64 x 64 pixels and this is fed to the model by applying both the RGB method and Thresholding method and we get the output on the screen for the corresponding input gesture.

VI. ALGORITHM

Algorithm to create a Model and Saving the model:

STEP 1: Creation of Data set of the required Gestures and saving them.
STEP 2: Converting all the data sets form BRG into RGB.
STEP 3: Converting all the data sets from RGB to Gray Scale.
STEP 4: Converting all the data sets from the Gray Scale to Binary.
STEP 5: Creation of Images list, Labels list, and adding all the Binary images to the Images list.
STEP 6: Converting all the images list into arrays.
STEP 7: Now dividing the data into Training (95%) and Testing (5%) set.
STEP 8: Creation of Keras model using sequential by adding the layers.
STEP 9: By calling create_model() function model is created.
STEP 10: While training the model we need to have some data for validation of the model.
STEP 11: Here we are creating a validation data of 0.1% of total training data.
STEP 12: Plotting the Accuracy and Loss graphs between the Training and Validation.
STEP 12: Now getting the evaluation accuracy and loss.
STEP 13: Now Loading the Testing data.
STEP 14: Repeat the step from 2 to 6 on training data.
STEP 15: Now feeding the Test data to model for prediction of the output.
STEP 16: Save the model and weights created.
Thus, the model is created and saved.

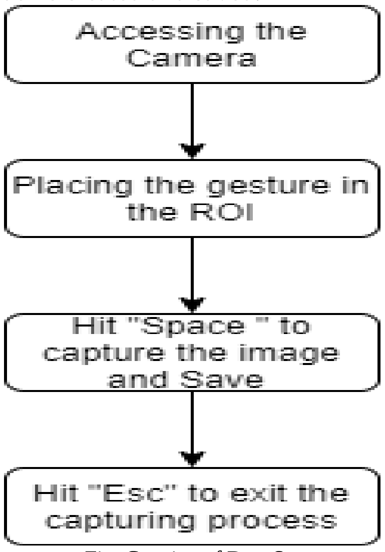

Accessing the Camera
Placing the gesture in the ROI
Hit "Space " to capture the image and Save
Hit "Esc" to exit the capturing process

Fig: Creation of Data Set

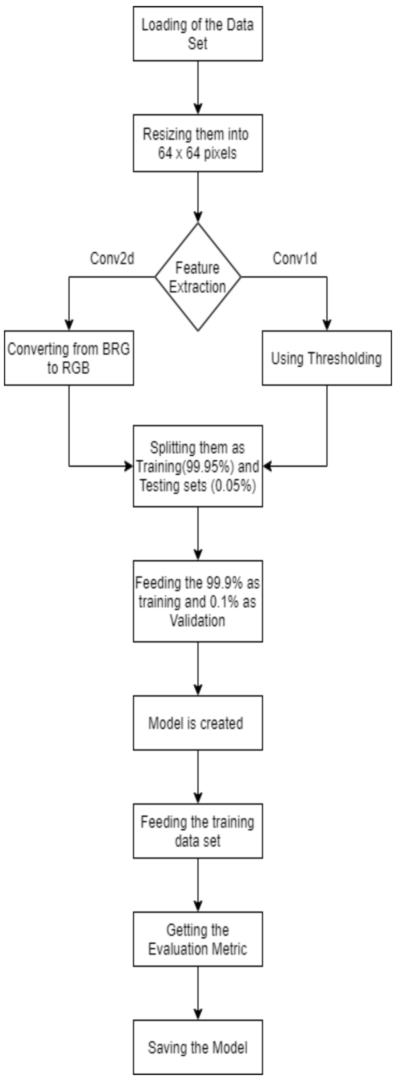

Loading of the Data Set
Resizing them into 64 x 64 pixels
Conv2d
Feature Extraction
Conv1d
Converting from BRG to RGB
Using Thresholding
Splitting them as Training(99.95%) and Testing sets (0.05%)
Feeding the 99.9% as training and 0.1% as Validation
Model is created
Feeding the training data set
Getting the Evaluation Metric
Saving the Model

Fig: Creation of model

Algorithm to load a saved Model and get output to the given input:

STEP 17: Use cv2.VideoCapture() to get a video capture object for the camera.
STEP 18: Set up an infinite while loop and use the read () method to read the frames using the above-created object.
STEP 19: Use the cv2.imshow() method to show the frames in the video.
STEP 20: Breaks the loop when the user clicks an ‘ESC’ key.
STEP 21: Crop the image of the in-frame where the gesture is placed.
STEP 22: Resizing the cropped image into size 64*64.
STEP 23: Again, follow the steps from 2 to 6 to cropped image.
STEP 24: Feeding the image to the model and we get the output to the corresponding given input on the screen.

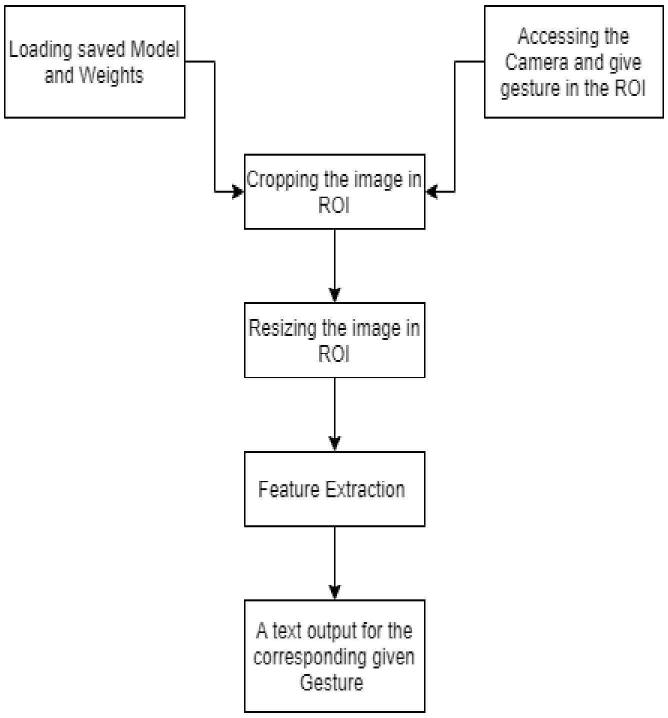

Loading saved Model and Weights
Accessing the Camera and give gesture in the ROI
Cropping the image in ROI
Resizing the image in ROI
Feature Extraction
A text output for the corresponding given Gesture

Fig: Giving Live input to the Model

VII. RESULTS

These are the output we got when we executed the program while giving live input to the model. In the case of the Thresholding converted data set we got exact output for the given input gestures. Whereas in RGB converted data set we did not get the accurate output.

Fig: These are the output for the few inputs given for the Thresholding data set.

VIII. CONCLUSION

Here we used two feature extraction methods. Among these the model that trained with Thresholding converted data got an accuracy of 98.69% and also, we got an accurate prediction for given gesture into ROI. But in the case of the RGB converted data sets we did not get the accurate results for the given input gesture. The thresholding feature extraction technique is used for getting an accurate output for gesture recognition.

ACKNOWLEDGEMENT

We might want to thank the open-source network on the Internet, specifically the guides at TensorFlow, whose help was essential to our learning procedure.

REFERENCES

[1] MARGERET ROUSE BASIC INFORMATION IN CONVOLUTION NEURAL NETWORKS [ONLINE]. ACCESSIBLE: https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network
[2] KERAS CONV2D: WORKING WITH 1D CONVOLUTION NEURAL NETWORKS [ONLINE]. AVAILABLE https://missinglink.ai/guides/keras/keras-conv2d-working-cnn-2d-convolutions-keras/
[3] KERAS CONV2D: WORKING WITH 1D CONVOLUTION NEURAL NETWORKS [ONLINE]. AVAILABLE https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/

AUTHORS

[1] **First Author** – Chava Sri Varshini, B. Tech Student, VNR VJIET,
[2] **Second Author** – Guguloth Sreekanth, B. Tech Student, VNR VJIET,
[3] **Third Author** – Gurram Hruday, B. Tech Student, VNR VJIET,
[4] **Fourth Author** – Mysakshi Chandu, B. Tech Student, VNR VJIET,
[5] Correspondence Author --Shaik Khadar Sharif, Associate Professor VNR VJIET,