International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

A Vision Based Hand Gesture Recognition System using Convolutional Neural Networks

Simran Shah[1], Ami Kotia[2], Kausha Nisar[3], Aneri Udeshi[4], Prof. Pramila. M. Chawan[5]

[1,2,3,4]U.G. Students, Department of Computer Engineering, VJTI College, Mumbai, Maharashtra, India
[5]Associate Professor, Department of Computer Engineering, VJTI College, Mumbai, Maharashtra, India

---***---

Abstract - *In more recent years, human computer interaction is becoming an important part of most state-of-the-art emergent technologies. The traditional mode of interaction via keyboards, mouse and joystick cannot meet the demands of this fast growing technology, and hence, in this paper, hand gesture recognition is explained and implemented, to enable further development of natural communication between humans and computers. Some methods and algorithms used in the process are further delved into along with diagrams explaining the entire flow. Finally, a technique is proposed to show its increased efficiency in processing images, reduced training time and accurate hand gesture recognition. The experiment results are also tabulated.*

Key Words: Hand Gesture Recognition, Human Computer Interaction(HCI), Convolutional Neural Networks(CNN)

1. INTRODUCTION

Traditionally, users needed to tie themselves up with the help of electronic wires in order to connect or interface with the computer system. In the previously used wired technology, a user was unable to freely move as they are connected with the computer system with the wire and movement is limited to the length of wire. Instrumented gloves, which are also called electronics gloves or data gloves are an example of wired technology. These data gloves provide good results but they are extremely expensive to utilise in wide range of common application. Recently, some advanced vision based techniques have been introduced that require processing of image features like texture and colour.

The purpose of this project is to implement natural interaction between humans and computers so that the recognised hand gestures can be used to convey meaningful information. We humans communicate not just with our words, but also with our gestures. With the recent development in computer vision and human computer interaction, we can create a system that is capable of identifying hand gestures and then performing suitable actions like managing certain display settings, allowing play/pause of video players, volume moderation and forward/rewind of videos as well. We can define different positions or specified sequences of our hand movements as the hand gesture that our computer should recognise. Gestures may be static - requiring less computational complexity, or dynamic, which are more complex and also more feasible for real time systems. To exploit the use of gestures in HCI, it is important to provide the means by which they can be interpreted by the computers.

There are usually two main characteristics that should be deemed when designing an HCI system, and they are: Functionality and Usability. System functionality refers to the set of functions or services that the system equips the user to perform, and system usability refers to the level and scope under which the system can perform specific user purposes efficiently and more accurately.

1.1 Significance of Hand Gesture Recognition

Although the world is moving very fast with various voice recognition techniques and applications, a part very essential of human interaction, gestures, are still in the process of being developed to their full potential. We are trying to implement these gestures for a fully functional gesture based video player, allowing the system to understand human actions and perform action sequences. This could be the prototype for many more evolving systems that make the interface interaction easier for humans and computers.

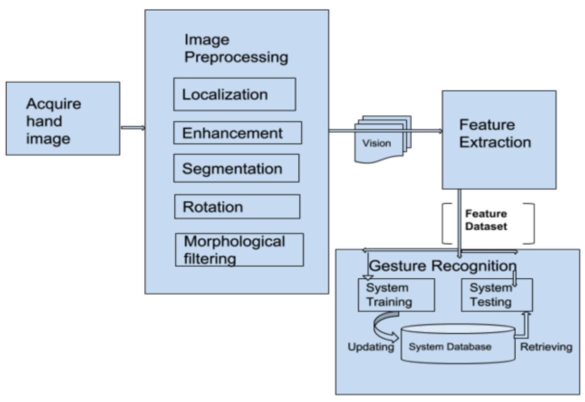

Fig 1- Architecture for Proposed System

2. LITERATURE REVIEW

In gesture recognition, there are certain image processing features, which are followed by neural networks to help classify the identified gesture.
Some of the techniques used are:

1) Image Pre processing- In our model, we have two modes of preprocessing the captured images. Binary Mode is used to convert the image to grayscale, whereas SkinMask Mode is used to convert the images to HSV format where the value range depends on the skin colour. In each of these, further noise removal techniques like gaussian blur, erosion and dilation are applied.

2) Segmentation- Color based skin detection is most preferable for realistic applications. We use skin segmentation to reject as much of 'non-skin' background as possible. Since people with different complexion have different likelihood, an adaptive thresholding process is required to achieve the optimal threshold value. The output will be a grayscale image whose gray values represent the likelihood of the pixel belonging to skin.

3) Enhancement- Image enhancement is done in order to improve illumination and remove blurring caused during image acquisition. Image features stand out more clearly with the use of this concept. Gaussian blur is used to smoothen out the noise.

4) Morphological filtering- Morphological filtering is necessary to be applied on segmented images to get a better smooth, closed and contour of a gesture. This is achieved using a sequence of dilation and erosion operations over the rotation invariant segmented gesture image.

In our project we have used convolutional neural networks. It is a class of deep neural networks which is most accurate and efficiently applied for analyzing visual imagery.

Convolutional Neural Networks use a slight variation of multilayer perceptrons. This is designed so that it required minimal preprocessing, resulting in better experiment accuracy.[1] They are also called shift invariant or space invariant artificial neural networks (SIANN), due to their translation invariance characteristics and shared-weights architecture.

These neural networks draw their main principle from the manner in which the animal visual cortex is organised. There are many connectivity patterns between neurons, and the neural networks are inspired by this.

There is a restricted region of the visual field which is known as the receptive field. The individual cortical neurons respond to stimuli only in this field. These receptive fields of different neurons partially overlap in a manner such that they cover the entire visual field.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

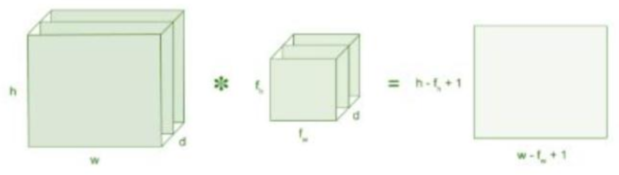

Fig 2- Convolutional Layer of CNN

Some of the important parts of a convolutional neural network are-

1) Convolutional Layer- This is the first layer to extract features from an input image. Convolution helps to preserve the relationship between various pixels by learning image features using small squares of input data. It is a mathematical operation which usually takes two inputs. They are the image matrix and a filter or a kernel.

2) Stride- This is the number of pixels shifts that we perform over the input matrix. For example, when the stride is 1 then we move the filters to 1 pixel at a time. Similarly, when the stride is 2 then we move the filters to 2 pixels at a time and so on.

3) Padding- It is observed that at times the filter does not fit perfectly fit the input image. We have two main options in this case: a) Pad the picture with zeros (zero-padding) so that it fits or b) drop the part of the image where the filter did not fit. The second method is called valid padding which keeps only valid part of the image.

4) Non Linearity- ReLU means Rectified Linear Unit for a non-linear operation. The output is f(x) = max(0,x). ReLU is quite important in CNNs: the main purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn what could be non-negative linear values.

5) Pooling Layer- This section would reduce the number of parameters when the images are too large. Spatial pooling also called downsampling or

subsampling because it reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types-

a) sum pooling
b) max pooling
c) average pooling

6) Fully Connected Layer- In this final layer, we flatten our matrix into a vector and feed it to a fully connected layer like the proposed neural network.

3. ARCHITECTURE FOR PROPOSED CNN MODEL

1) We have used sequential API to create our model layer-by-layer.
2) Our ConvNet for hand gesture recognition has the architecture [INPUT - CONV - RELU - CONV - RELU - MAXPOOL - DROPOUT - FLATTEN - DENSE - RELU - DROPOUT - FC - SOFTMAX]
3) INPUT [200x200x1] will hold the raw pixel values of the image, in this case an image of width 200, height 200, and with 1 color channel. (gray scale)
4) CONV layer will calculate dot product between their weights and a small region they are connected to in the input volume.
5) The RELU layer will apply an elementwise activation function, such as the activation function of max(0,x) thresholding at zero.
6) The POOL layer will perform a subsampling operation along the spatial dimensions (width, height).
7) FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1xnb_classes], where each of the nb_classes no. of numbers correspond to a class score. As with other conventional Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.
8) The objective function that the model tries to minimise is categorical cross-entropy.
9) The model uses 'adadelta' optimizer.

```
_____________________________________________
Layer (type)              Output Shape          Param #
=============================================
conv2d_1 (Conv2D)         (None, 32, 198, 198)     320
_____________________________________________
activation_1 (Activation) (None, 32, 198, 198)       0
_____________________________________________
conv2d_2 (Conv2D)         (None, 32, 196, 196)    9248
_____________________________________________
activation_2 (Activation) (None, 32, 196, 196)       0
_____________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 98, 98)      0
_____________________________________________
dropout_1 (Dropout)       (None, 32, 98, 98)         0
_____________________________________________
flatten_1 (Flatten)       (None, 307328)             0
_____________________________________________
dense_1 (Dense)           (None, 128)         39338112
_____________________________________________
activation_3 (Activation) (None, 128)                0
_____________________________________________
dropout_2 (Dropout)       (None, 128)                0
_____________________________________________
dense_2 (Dense)           (None, 5)                645
_____________________________________________
activation_4 (Activation) (None, 5)                  0
=============================================

Total params: 39,348,325.0
Trainable params: 39,348,325.0
```

4. USAGE AND FEATURES

This model can be used on Windows and Macintosh, using Tensorflow and Theano as backend respectively. We use these for the KERAS backend.

We have trained 7 gestures in our model, which are:

1) OK
2) PEACE
3) STOP
4) PUNCH
5) THUMBS UP
6) THUMBS DOWN
7) NOTHING

This application provides following functionalities:
1) Prediction: This feature allows the app to guess the user's gesture against pretrained gestures. This app can dump the prediction data to the console terminal or to a json file directly which can be used to plot real time prediction bar chart.
2) New Training : This feature allows the user to retrain the Neural Network model. Any user can change the model architecture or add/remove new gestures. This app has inbuilt options to allow the user to create new image samples and folders of user defined gestures if required.
3) Visualization : This feature allows the user to see feature maps of different Neural Network layers for a given input gesture image.

5. PROPOSED TECHNIQUE

a)Pre-Processing of Gesture Images

We are using OpenCV for capturing the user's hand gestures.
We have provided two modes of processing on captured images:

1) Binary Mode processing
2) SkinMask Mode processing

1)Binary Mode processing:

ALGORITHM:

Step 1: Convert the input image to grayscale.
Step 2: Apply a gaussian blur effect with adaptive threshold filter. This mode is quite useful when you have an empty background like a whiteboard, wall etc.

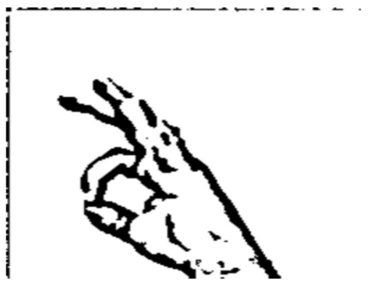

Fig 3- Binary mode processing on captured image

2)SkinMask Mode processing:

ALGORITHM:

Step 1: Convert the input image to HSV.
Step 2: Put range on the H,S,V values based on skin color range.
Step 3: Apply erosion followed by dilation.
Step 4: Apply gaussian blur to smoothen out the noises.
Step 5: Using this output as a mask on original input, mask out everything other than skin colored things.
Step 6: Convert from color to grayscale.

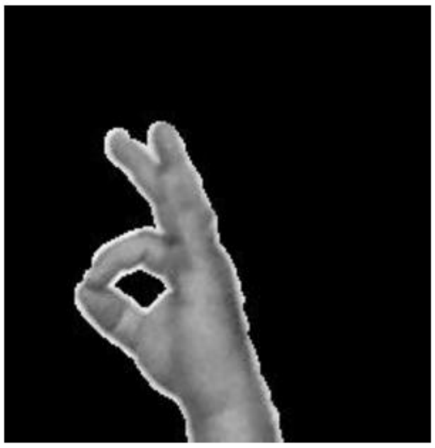

Fig 4- Skin Mask processing on captured image

b) Dataset Creation

The model provides a method to add new gestures and train them accordingly. As mentioned above, we have 2 modes of pre-processing the images, and we use these to create our dataset for every gesture.

ALGORITHM:

Step 1: Run the pretrained model for gesture recognition.
Step 2: Press ‘n’ to add a new gesture folder name.
Step 3: Enter the new gesture folder name in the prompt.
Step 4: Place hand in the green box with the required gesture. 301 sample pictures will be captured for every new folder.
Step 5: You may also apply any of the masks to store your data images in a different format.
Step 6: 803 samples of every new gesture are added to the main image folder.

c) Training Algorithm

ALGORITHM:

Step 1: Store the images in set X and their labels i.e. gesture indices in set Y.
Step 2: Split X and Y into training and testing sets X_train, Y_train and X_test, Y_test respectively.
Step 3: Define batch size and number of epochs for training.
Step 4: Define validation split to further split the training dataset X_train, Y_train into training set and validation set.
Step 5: Feed X_train and Y_train in the loaded CNN model, specify the parameters batch size, no. of epochs and validation split and start training.
Step 6: Visualise the accuracy by plotting both training and validation accuracy against number of epochs.
Step 7: Visualise the losses by plotting both training and validation loss against number of epochs.
Step 8: Store the trained weights in the weights file.

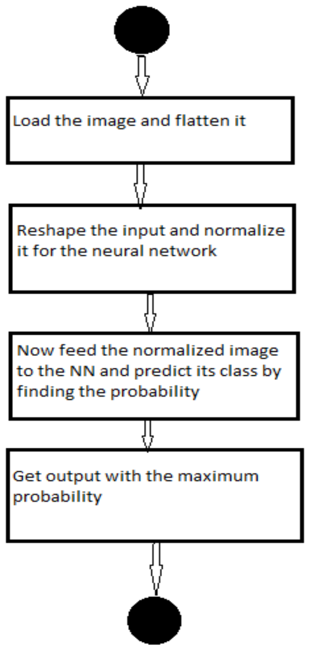
Load the image and flatten it
Reshape the input and normalize it for the neural network
Now feed the normalized image to the NN and predict its class by finding the probability
Get output with the maximum probability

Fig 5- Flow chart for CNN training process

d) Guess Gesture

Guess Gesture does the guessing work based on the input images

ALGORITHM:

Step 1: Load the image and flatten it.
Step 2: Reshape the input image and normalize it for NN.
Step 3: Now feed the normalized image to the NN, to fetch the predictions by predicting the classes and finding the probability.
Step 4: Get the output with the maximum probability

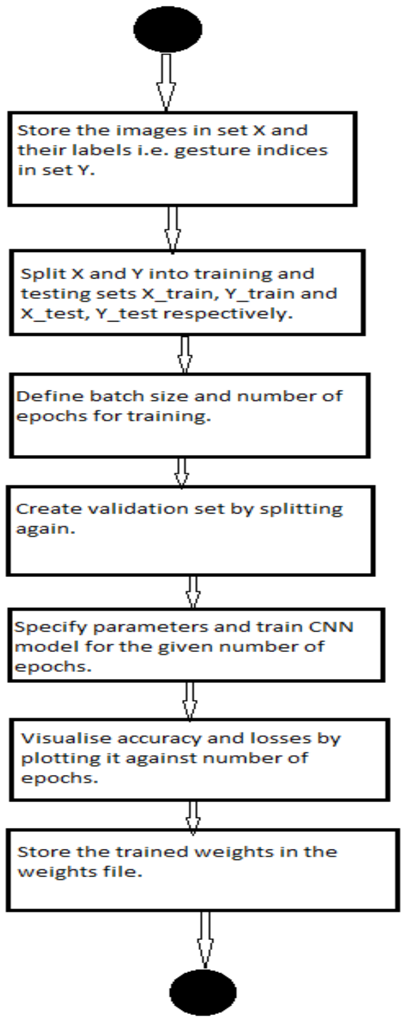
Store the images in set X and their labels i.e. gesture indices in set Y.
Split X and Y into training and testing sets X_train, Y_train and X_test, Y_test respectively.
Define batch size and number of epochs for training.
Create validation set by splitting again.
Specify parameters and train CNN model for the given number of epochs.
Visualise accuracy and losses by plotting it against number of epochs.
Store the trained weights in the weights file.

Fig 6- Flow chart for gesture recognition process

6. RESULTS

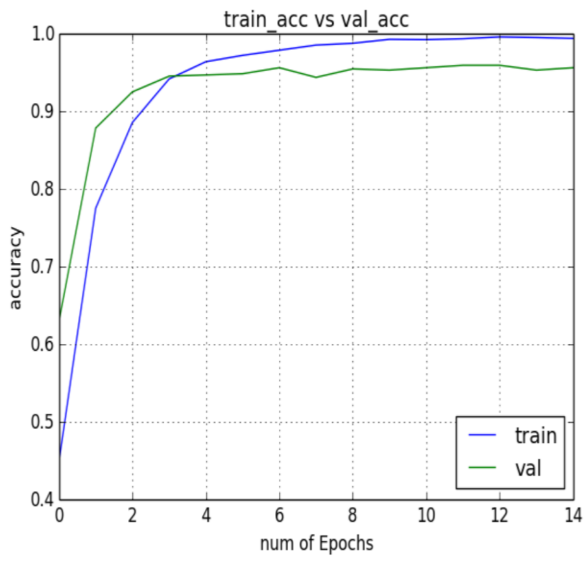

Fig 7- Graph of accuracy against number of epochs

Training accuracy – 98.98%
Validation accuracy - 96.03%

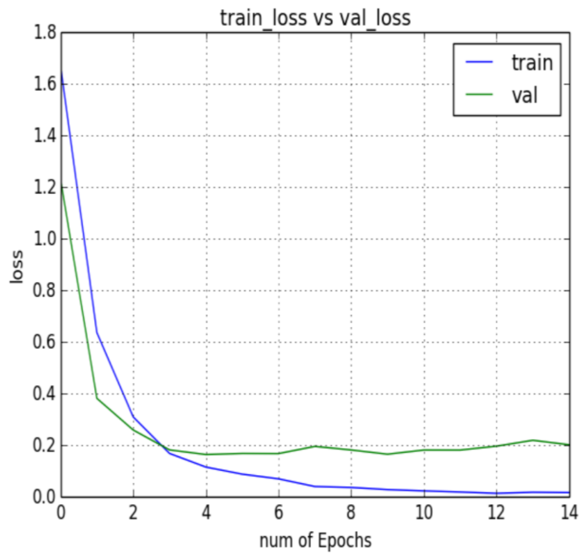

Fig 8- Graph of loss against number of epochs

Training loss - 0.1048
Validation loss - 0.2001

7.CONCLUSION

In the implemented system, thus, designed allows seamless interaction between humans and computers in the YouTube application.

We have implemented 5 gestures, namely, fist, stop, thumbs up, thumbs down, point and peace, which help us control different functionalities like brightness, volume, start/stop in the application.

One of the major challenges we faced in this system was in determining the interval of obtaining images to detect the gestures, so as to achieve maximum accuracy. There is more scope in expanding this system in other applications like Acrobat Reader and Microsoft PowerPoint

8.REFERENCES

[1] Hamid A. Jalab, Herman. K. Omer, "Human Computer interface using Hand Gesture Recognition based on neural network", IEEE 06 August 2015 , Electronic ISBN: 978-1-4799-7626-3, CD-ROM ISBN: 978-1-4799-7625-6

[2] Sagar P.More, Prof. Abdul Sattar, "Hand Gesture Recognition System For Dumb People", A R DIGITECH, International Journal Of Engineering, Education And Technology (ARDIJEET), ISSN 2320-883X, Volume 3 Issue 2, 2015

[3] LeCun, Yann. "LeNet-5, convolutional neural networks". Retrieved 16 November 2013.

[4] E. Hunter, J. Schlenzig, and R. Jain. Posture Estimation in Reduced-Model Gesture Input Systems. Proc. International Workshop Automatic Face and Gesture Recognition, pp. 296-301, 1995.

[5] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," Computer Vision Image Understanding, volume 108, Issue 1–2, pages 52–73, October-November 2007.

[6] Harpreet Kauri and Jyoti Rani, "A Review: Study of Various Techniques of Hand Gesture Recognition", IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), 2016.

[7] S. Mitra, T. Acharya, "Gesture Recognition: A Survey", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, pages 311-324, 2007.

[8] Shweta. K. Yewale and Pankaj. K. Bharne, "Hand Gesture Recognition Using Different Algorithms Based on Artificial Neural Network", IEEE International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pages 671-675.