Static Hand Gesture Recognition using Deep Learning

Chitransh Lodha
3rd Year Undergraduate
Department Of Electrical and Electronics
National Institute of Technology Karnataka, Surathkal
Mangaluru, Karnataka, 575025, India
*Contact: chitransh1998@gmail.com, phone +91-9611589859

***Abstract*— The rapid development of science and technology has made human-computer interaction much more frequently around us. In order to offer new possibilities to interact with machine and to design more natural and more intuitive interactions with computing machines, this research aims at automatic interpretation of hand gestures based on deep learning primarily convolution neural networks. The three main steps are 1) Image pre-processing 2) Feature detection and 3) Hand shape recognition. Image pre-processing is implemented using OpenCV and feature detection is carried out using Convolution layers. Finally, Dense or Fully Connected Layers are used for recognition and classification. The results show that the model can detect gestures in real time and accuracy as high as 96.46% was achieved on previously unseen data.**

Keywords – hand gesture recognition, convolution neural networks, deep learning, computer vision

1. Introduction

The matter of the human computer interface has risen since the rise of computers. It is a discipline concerned with the means and tools implemented so that humans can control and communicate with a computer system. In recent decades, there has been rapid development in this field and it finds applications in multiple domains like home automation, automotive sector, defence sector ,sign language interpretation, gaming sector and also recently in unlocking smart phones.

Hand gestures are powerful human to human non-verbal form of communication conveying major part of information transfer in our everyday life. Hand gesture recognition is the process of understanding and classifying meaningful movements by human hands which can then be used to perform different tasks.

The domain of hand gesture recognition is generally divided into two categories i.e. contact based and vision-based approaches. The first is complex and requires hardware devices to implement.

The aim is to recognize four static hand gestures while maintaining accuracy and speed of the system. Earlier hardware based hand gesture recognition was more prevalent. The user had to wear gloves helmet and similar other heavy apparatus which made the process difficult in real time environment. In contrast vision based models require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra hardware devices. The main problem with image classification is feature extraction for any model. A standard three channel RGB image has so many features which make it a very complex problem. The manual handcraft of creating features from images like shapes, edges, regions is not an easy task even with sophisticated tools. However, a convolution neural network along with a model for classification is able to create and select automatically useful features. The goal of gesture recognition is to build a powerful classification filter. The dataset for this project was collected manually from a webcam attached to the PC.

2.METHODOLOGY

The several steps involved in completing this project are explained in detail in the following sections:

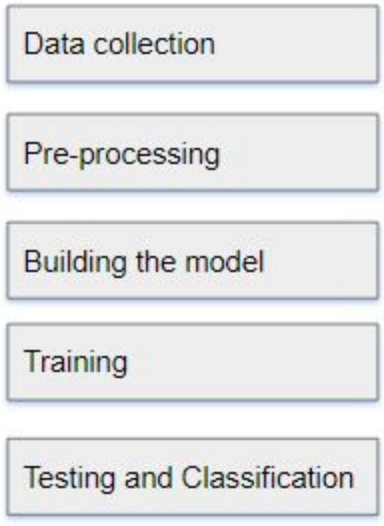


Fig1 System Flow Chart

2.1 Data Collection

The dataset for the project was collected manually using a webcam in the author's personal computer.A code was written to automate and speed up the process where in multiple images were taken and stored in the respective directories..The dataset consists of 2320 images each 200*200 pixels and three channel wide,divided into 2000 training and 320 test images.Further,each class has 500 training and 80 test images and the classes of gesture which are as follows:

- Stop
- Thumbs-up
- Punch
- Peace

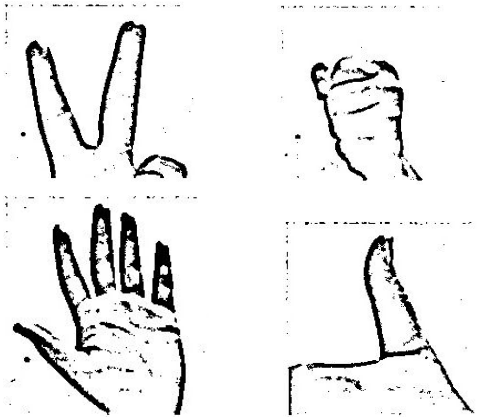


Fig 2 Visualization of dataset (From top left peace,punch,stop,thumbs-up)

Each image was first shot using a webcam and a binary mask was then applied to obtaining a binarized and thresholded image.Binary mask has been discussed in the preprocessing section in detail.

The training data was used to train the network weights and the test data to validate the training

2.2 Preprocessing

Data input to the trained model is pre-processed before being sent for classification by the neural network.There are two modes of processing based on the amount of noise in the background viz .Binary Mode processing and Skin Mask mode processing

2.2. A Binary Mode Processing

In this mode the image is first converted into a grayscale using OpenCV command RGB2GRAY.Following which Gaussian Blurring is applied with a kernel of 5*5 to blur the noise in the image.This is followed by a adaptive gaussian thresholding.The adaptive feature allows the model to select a threshold as per the image and thus making the code more robust to noise.Finally,the thresholded image is input to the model for classification.This mode finds it use when you have an empty background without any other objects like a white wall or a whiteboard.

Fig 3 Steps in Binary Mode processing

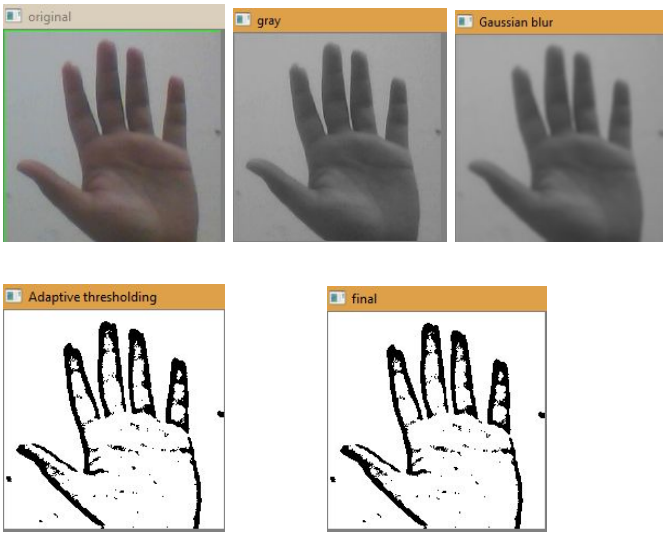


Fig 4 Visualization of various steps in Binary Mode Processing(From Top left original image, Grayscale Image,Gaussian blurring,Adaptive thresholding,final output)

2.2.B Skin Mode Processing

In this mode the source image first converted from RGB to YCrCb space using the BGR2YCR_CB function in OpenCV and saved with a different name.YCbCr represents a family of color spaces used as part of the color image pipeline in video and digital photography systems.Here,Y represents the luminance component and Cb and Cr represent the

blue-difference and red-difference chroma components.Luma component is the brightness of the color. That means the light intensity of the color. The human eye is more sensitive to this component.

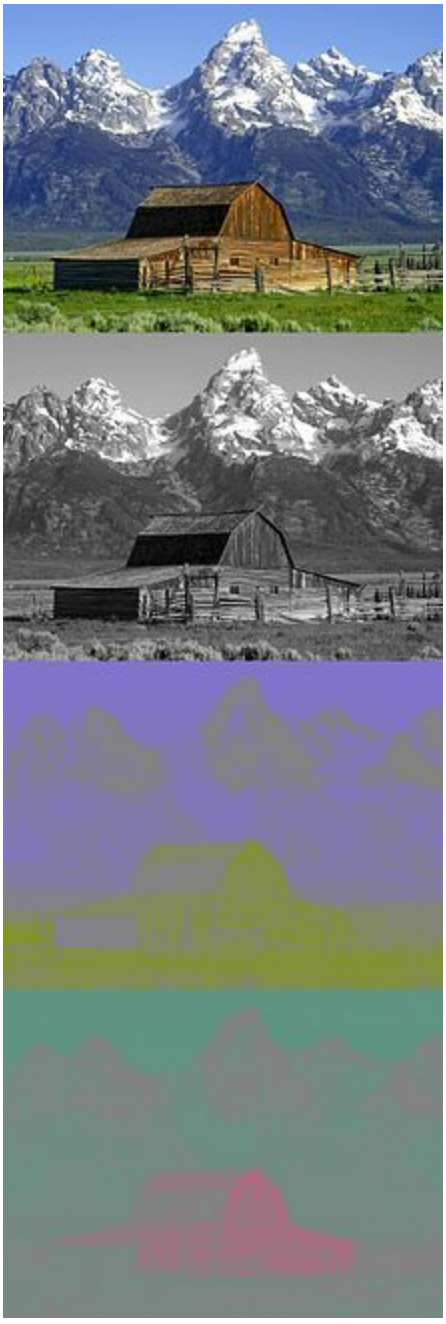


Fig 5 A Color image and its Y,Cb and Cr components.The Y image is essentially a grayscale copy of the main RGB image.
(Source:Wikipedia)

After converting to YCrCb color space the regions with the skin tone are easily identified by using a suitable range of values for minimum and maximum parameters in the cv2.inRange() function.Then contours are detected in the regions with the skin tone and finally the detected contours are drawn on the original source image.It is also made sure that only those contours are drawn whose areas is greater than 1000 pixel square to avoid small patches and noises.The original source image is then converted from RGB to GRAYSCALE and then binarized using cv2.threshold function.Finally the binarized image is passed to the neural network for classification

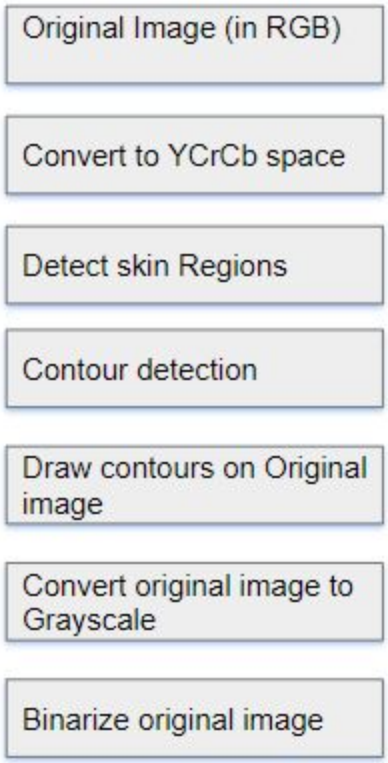
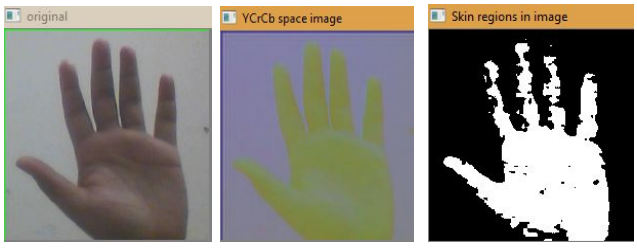


Fig 6. Steps in Skin Mode Processing



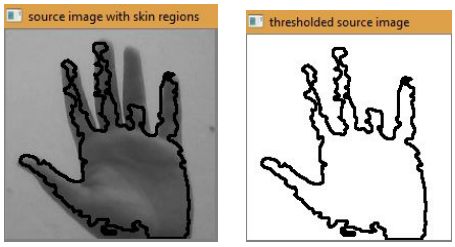


Fig 7 Visualization of Various steps in Skin Mode Processing (From Top Left :Original Image,YCrCb image,Skin Regions in image,Source image with contours,Binarized final image with skin boundaries)

2.3 Building the model

Keras is used for building,training and testing the model.

The model consists of two broad divisions namely:1.Feature Extraction 2. Classification

2.3.A Feature Extraction:

Feature extraction is a dimensionality reduction process,where an initial set of raw variables is reduced to more manageable groups or features for processing,while still accurately and completely describing the original dataset.When the input data to an algorithm is too large to be processed and it is suspected to be redundant (eg the repetitiveness of images presented as pixels),then it can be transformed into a reduced set of features.The selected features are expected to contain the relevant information from the input data,so that the desired task can be performed using this reduced representation of the complete initial data

In image processing,algorithms are used to isolate various desired portions or shapes(features) of a digitized image or a video stream.Some low level features include edges,corners,blobs,ridges etc.Some higher level features could be shapes,templates,lines,etc.Images contain a huge amount of parameters and hence to reduce the computation effort,we use convolutional networks.Convolutional layers were inspired by biological processes in that the connectivity between neurons resembles the organization of the animal visual cortex.Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable.The structure of the feature extraction part is displayed in Fig 8

The network consists of two convolutional layers with a ReLU activation function..Max Pooling in two dimensions and dropout layers have also been added to reduce overfitting to the training fata.

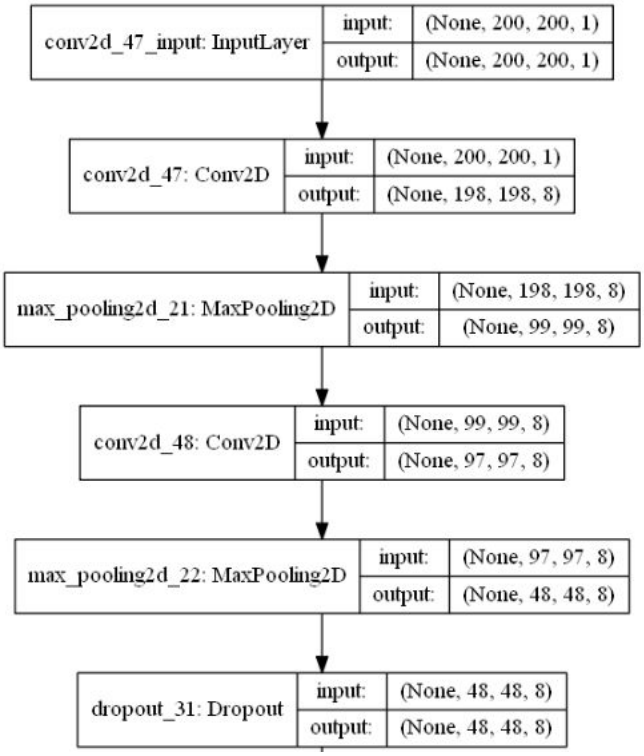


Fig 8 Structure of the Feature Extraction part

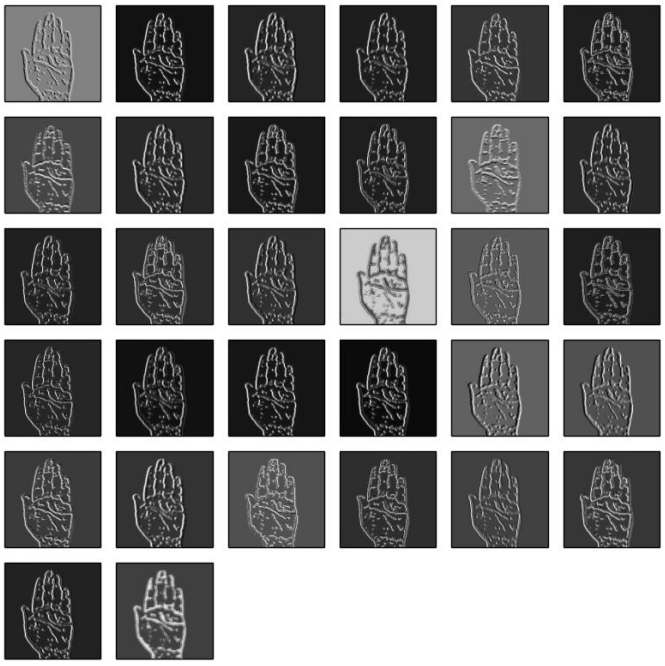


Fig 9 Visualization of the Stop Gesture

2.3.B Classification

Image Classification is assigning pixels in image to categories or classes of interest.In order to achieve this,the relationship between the data and classes into which they are classified must be well understood.To achieve this,we feed the features extracted using the convolution layer into a fully connected or dense layer.

Here,supervised classification is employed as here the classifier or the model has the advantage of an analyst or domain knowledge using which the classifier can be guided to learn the relationship between the data and the classes.Supervised classification generally performs better than unsupervised classification if good quality training data is available.

The dense layer gets a horizontal input from the feature extraction part.There are three dense layers with ReLu activation functions.Each layer if followed by a dropout layer to reduce overfitting.The output layer consists of four outputs one for each gesture and has sigmoid activation function to give probabilistic interpretation of the outputs.

The structure of the dense layers employed is displayed below.

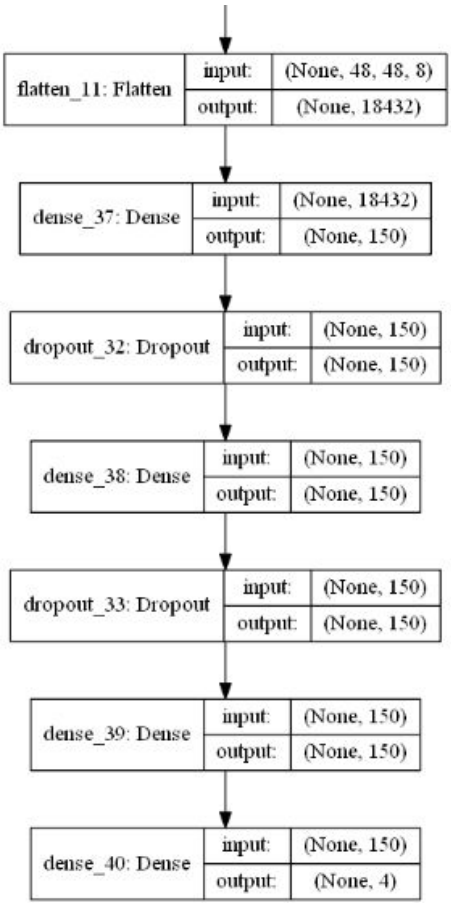


Figure 10. Structure of the Classification part.

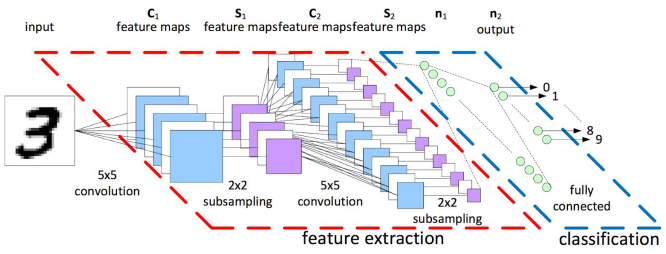


Fig 11. Visualization of the roles of Feature extraction and classification in a neural network

2.4 TRAINING

After building the model,it is compiled where the backend automatically chooses the best way to represent the network for training and making predictions to run on hardware.It uses efficient numerical libraries like Theano or tensorflow for the same.Training means finding the best set of weights and biases to make a prediction for the problem.

The categorical cross entropy function is used to accuracy and evaluating the model.

Adam optimizer is used to search for different weights to make prediction for this problem.

Adam optimization has been found empirically to be computationally efficient and also reduces oscillation in the descent towards the minima effectively,thus working better for problems with large number of data and parameters.

Adam realizes the benefits of both Gradient descent with Momentum and RMSProp.

Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.The different parameters for Adam configuration are along with their values used:

Alpha. Also referred to as the learning rate or step size. The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training. Alpha=0.001

Beta1. The exponential decay rate for the first moment estimates (e.g. 0.9). Beta1 = 0.9

Beta2. The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems). Beta2 = 0.999

Epsilon. Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8). Epsilon = 10e-8

Number of epochs =10,Batch Size = 32

The adam optimization is explained in the below figure:

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Fig 12. Adam optimization algorithm

Three models were trained using the above specified criteria and making small changes.The model with the highest accuracy and minimum loss was taken.

```
195/195 [==============================] - 371s 2s/step - loss: 2.6651 - acc: 0.8815 - val_loss: 1.5971 - val_acc: 0.9125
Epoch 2/10
195/195 [==============================] - 337s 2s/step - loss: 1.1230 - acc: 0.9928 - val_loss: 1.2933 - val_acc: 0.9062
Epoch 3/10
195/195 [==============================] - 336s 2s/step - loss: 0.6845 - acc: 0.9933 - val_loss: 1.0357 - val_acc: 0.8781
Epoch 4/10
195/195 [==============================] - 376s 2s/step - loss: 0.5290 - acc: 0.9918 - val_loss: 0.5682 - val_acc: 0.9219
Epoch 5/10
195/195 [==============================] - 414s 2s/step - loss: 0.3935 - acc: 0.9938 - val_loss: 0.5575 - val_acc: 0.9219
Epoch 6/10
195/195 [==============================] - 331s 2s/step - loss: 0.3233 - acc: 0.9954 - val_loss: 0.5116 - val_acc: 0.9156
Epoch 7/10
195/195 [==============================] - 333s 2s/step - loss: 0.2093 - acc: 0.9938 - val_loss: 0.4860 - val_acc: 0.9156
Epoch 8/10
195/195 [==============================] - 321s 2s/step - loss: 0.2617 - acc: 0.9903 - val_loss: 0.4152 - val_acc: 0.9125
Epoch 9/10
195/195 [==============================] - 313s 2s/step - loss: 0.2308 - acc: 0.9918 - val_loss: 0.4382 - val_acc: 0.9187
Epoch 10/10
195/195 [==============================] - 335s 2s/step - loss: 0.1762 - acc: 0.9949 - val_loss: 0.7636 - val_acc: 0.8687
Model has been saved.
```

Fig 13. Training the model

Graphics may be full color. All colors will be retained in the online proceedings but will be gray scale in the printed proceedings. Graphics must not use stipple fill patterns because they may not be reproduced properly. Please use only *SOLID FILL*colors which contrast well both on screen and on a black-and-white hardcopy, as shown in Fig. 1.

3. RESULTS AND DISCUSSIONS

The training results of the proposed model were found to beas :

Training Loss: 0.1762
Training Accuracy: 99.49%
Test Loss: 0.4382
Test Accuracy: 91.87%

A high training accuracy was observed due to the high amount of data available and well-tuned structure of the neural network.The model was then tested on live images using a webcam and the results were found to be satisfactory.

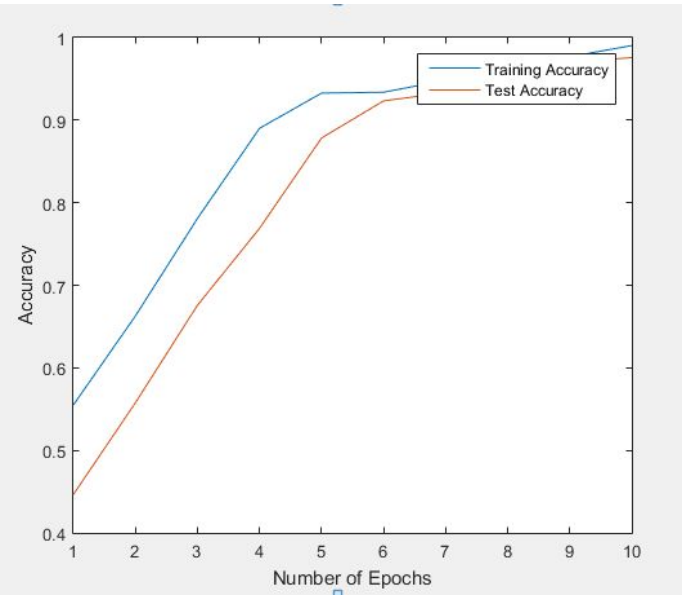


Fig 14 ,Accuracy

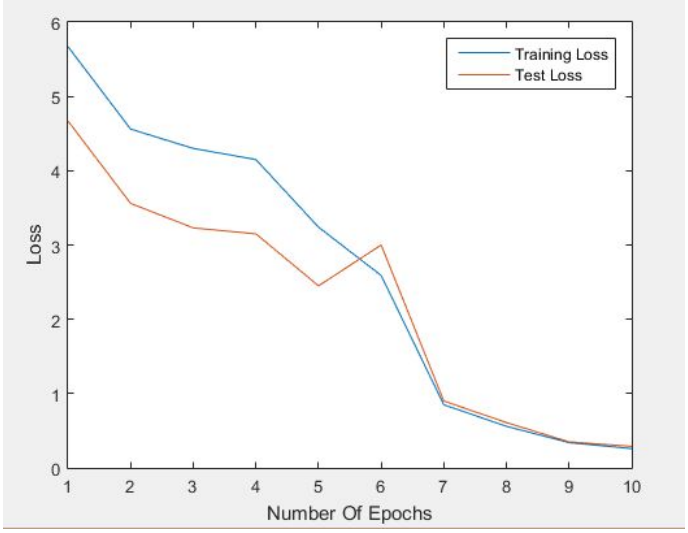


Fig 15. Loss

The training and the testing accuracy curves show a steady increase in accuracy as the number of epochs progresses.Similarly,there is consistent decrease the testing and training loss with the subsequent epoch.Hence,the model behaves well suited.

Previously,I was using simply images with clean background to recognize gestures.Now,with the previously discussed pre-processing techniques it was possible to recognize gestures even with not clear backgrounds.

The high training accuracy shows that the model might have overfitted to the training data.Hence,some alternate methods and possible changes to improve the model are as follows:

- Noise injection: Different techniques of noise injection to data during ANN optimization were discussed in a number of papers (Holmstrom and Koistinen, 1992; Grandvalet et al., 1997; Skurichina et al., 2000; Brown et al., 2003; Seghouane et al., 2004). In practical applications, the methodological variants of adding Gaussian noise to input data proposed by Holmstrom
  and Koistinen (1992)becomes the most popular. Noise injection was found to improve ANN generalization ability (Sietsma and Dow, 1991; An, 1996), especially in the case of classification problems for small data samples (Hua et al., 2006; Zur et al., 2009). The close similarities between noise injection and other methods designed to improve generalization properties of ANNs, including error regularization, were theoretically studied by Reed et al. (1995) and confirmed empirically by Zur et al. (2009). As a result, regularization methods are not considered in the present comparative study.
- Regularization: Regularization modifies the objective function that we minimize by adding additional terms that penalize large weights. In other words, we change the objective function so that it becomes Error+λf(θ), where f(θ) grows larger as the components of θ grow larger and λ is the regularization strength (a hyper-parameter for the learning algorithm).The most common type of regularization is L2 regularization. It can be implemented by augmenting the error function with the squared magnitude of all weights in the neural network. In other words, for every weight w in the neural network, we add 1/2 λw^2 to the error function. The L2 regularization has the intuitive interpretation of heavily penalizing "peaky" weight vectors and preferring diffuse weight vectors
- Early stopping : With early stopping, the choice of the validation set is also important. The validation set should be representative of all points in the training set.When you use Bayesian regularization, it is important to train the network until it reaches convergence. The sum-squared error, the sum-squared weights, and the effective number of parameters should reach constant values when the network has converged.
- Data Augmentation:There are some data augmentation techniques such as scaling, translation, rotation, flipping,resizing.Data augmentation can help reduce the manual interventation required to developed meaningful information and insight of business data, as well as significantly enhance data quality.Data augmentation techniques using Keras were explored and these could be used to train the model to reduce overfitting.These changes are to be made in the future versions of this model.Some of the results of the augmented image of stop are shown below.Keras was used for achieving data augmentation.

Fig 17 Data augmentation results

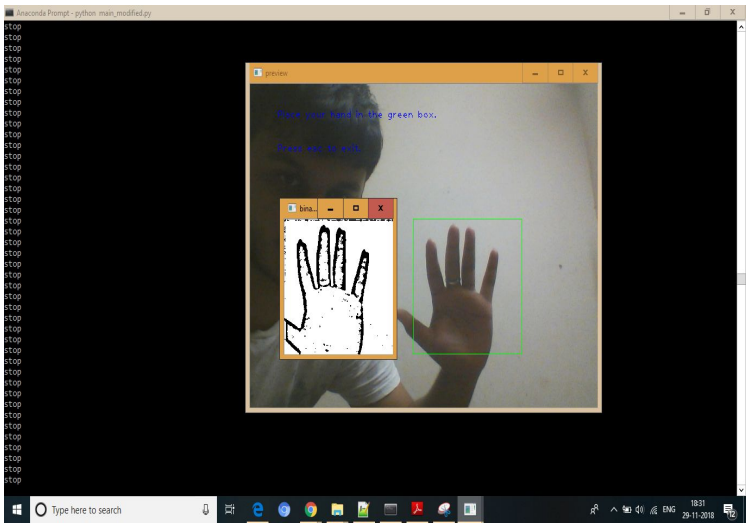


Fig 16. Code working to classify the stop gesture in real time with the classification result in the command window.

4.Conclusions

In this paper,a method to recognize static hand gestures by using deep learning and image processing.The model was built on Python 3 library Keras and open source image processing library OpenCV.Two different types of image processing based pre-processing techniques namely binary mode processing and skin mode processing were employed.Further,A convolutional neural network was used to classify the images in to the respective gesture class. The model could be trained just with a small duration of 10 epochs and also dataset was self-generated

instead of using a publicly available dataset.The model parameters were evaluated based on the testing set and accuracy of 99.49% was achieved on training data and 96.46% on previously unseen data.

Previously,I was using simply images with clean background to recognize gestures.Now,with the previously discussed pre-processing techniques it was possible to recognize gestures even with not clear backgrounds.

This has direct application in the automobile sector where this model can be used to give commands to the vehicle for other things while the driver is driving the car.This can save valuable lives and improve traffic safety.It can also be used in the field of gaming where the player can give commands to the characters in the game using this model.

5.Acknowledgment

I would like to express my deepest gratitudes to my supervisor Dr Yashwant Kashyap for providing his invaluable guidance,comments and suggestions throughout the course of this project.Without his support and guidance,the completion of this project was never possible.

6.References

1. Ali A.Alani and others R.2018.Hand gesture recognition using an adapted convolutional neural network with data augmentation,2018 4th International Conference on Information Management (ICIM)
2. D. Valencia and others R.2018.Hand gestures recognition using machine learning for control of multiple quadrotors 2018 IEEE Sensors Applications Symposium (SAS)
3. S.Hussain and others R.2017 Hand gestures recognition using machine learning for control of multiple quadrotors 2018 IEEE Sensors Applications Symposium (SAS),2017 International SoC Design Conference (ISOCC)
4. V.Bobic and others R.2016 Hand gesture recognition using neural network based techniques 2016 13th Symposium on Neural Networks and Applications (NEUREL)
5. G.R.S Murthy and others R.2010 ,Hand gesture recognition using neural networks,2010 IEEE 2nd International Advance Computing Conference (IACC)
6..John and A Boyali Deep Learning -based Fast Hand Gesture Recognition Using representative frames
7. T.Du and others R. 2018.Gesture Recognition Method Based On Deep Learning,33rd Youth Academia Annual Conference of Chinese Association of Automation May 18-20,2018,Nanjing,China
8. S. Hussain and others, Hand Gesture recognition using deep learning,IIT Guwahatin/
9. *M. Abdallah and others, R. 2012.An overview of Gesture Recognition,6th SETIT*
10. . M. panwar and others, Hand Gesture recognition for Human computer interaction