Establishing Control using Hand Gestures

Pritish Arora[1], Pratik Mali[2], Syed Ausaf Haider[3], Noor Mohammad Shaikh[4]

And Prof. Netra S Patil [5]

[1]Student, [2]Student, [3]Student, [4]Student and [5]Professor
Department of Computer Engineering (ACIT, NBA, NAAC A+ grade) Bharati Vidyapeeth Deemed to be University College of Engineering, Pune, India

---***---

Abstract*: The processing power of computational devices has grown at an exponential rate. Phenomenon's like artificial intelligence, Internet of things, virtual and stimulated reality etc are making major contributions to every sector of the society. Consequentially, the diversity of input methods to devices with computational abilities have become a domain of research. Powerful devices equipped with a variety of sensors can process various types of input signals simultaneously and effectively. Thus, leading to an expansion of the spectrum of inputs which intern enables a user to interact using sound, touch and even body language. Gestures are a common form of communication; we usually interact amongst ourselves using hand gestures. The Purpose of this particular study is to inherit gesture recognition into system enhancing human-machine interaction. This enables us to control a wide variety of devices remotely through hand postures with the help of vision systems. The following research focuses on controlling of a music player android application using the YOLO(v3) [1] algorithm for implementing hand gesture recognition [2] and establishing the control.*

***Keywords* - Human Computer Interaction (HCI), Hand Gesture, Gesture recognition system, Vision System,Yolo.**

1. INTRODUCTION

Body language is one of the key ways of interaction among humans. It adds emphasis on the complete message itself. Gesture is a form of non-verbal communication using various body parts, mostly hand and face [3]. We can inherit gesture recognition into system to enhance human-machine interaction. This would enable us to control a wide variety of devices remotely through hand postures.

We live in a world where essentially all the devices are interconnected in a closed ecosystem. With increasing need to maintain communication within a system, we have to come up with a centralized method for communication. With advent of IOT, more and more devices are getting interconnected, we cannot add more devices for controlling them. The complexity of system must be controlled as devices are getting compact and portable. Gesture system provides a novel method to facilitate the requirement of centralized technique for communication. Hand gesture best suits this way of interaction as it is the most convenient method for humans to interact.

Gesture eliminates the need of peripherals or external devices to be added to the ecosystem for interaction. Hand gesture can be classified in two types i.e. Static and Dynamic gestures. Static gesture [4] are the postures which refers only a single image to a single command. Stop sign is an example of static gesture. Static gesture is simple and need less computational power. Dynamic gesture change over a period of time and it is complex. A waving hand means goodbye is an example of dynamic gesture. Now a days, visual sensors are used in the domain of robotics, visual surveillance, manufacturing and healthcare.

This article proposes an android music player application which is controlled using hand gestures. A computer vision system is implemented using an open source neural network framework called darknet.

2. LITERATURE SURVEY

There are several key domains that are encompassed under the broad umbrella of "gesture recognition"; here we highlight some of the most significant. The earliest key instances were from Ishibuchi et al. [5] introducing hand gesture recognition using a 3D prediction model in 1993, and in the same year Ahmad et al. [6] introducing the use of a Bayesian techniques in order to recognise 3D hand gestures even with large amounts of noise and uncertain or missing input data. Mu-Chun et al. [7] discuss the use of a neural network to recognise only static postures, whilst Jung et al. [8] demonstrate methods for finding a human hand in an image sequence using colour separation; from the same group, Min et al [9] introduce the use of Hidden Markov Models to recognise gestures. Eickeler et al [10] followed this and presented a continuous Hidden Markov Model that essentially introduced dynamic feature extraction and evaluation methods. More recently, Bretzner et al. [11] have used a multi-scale colour segmentation technique, coupled with particle filters in order to track hand postures. Ming- Hsuan et al. [12] use 2D motion trajectories in order to track hand gestures, whilst Xia et al. [13] use depth data to provide information on hand gestures.

In the article "real time Hand gesture Recognition using finger segmentation," achieved the task of gesture recognition by first isolating the image of a hand from the background and then converting the image to grayscale or a binary image. This methodology of gesture recognition was based on essential identification of the fingers, so the palm was separated by drawing a maximal inner circle on the palm region and the erect fingers in the binary image were identified by the labelling algorithm which uses image processing techniques like edge detection. Similar approach followed by Ma, Xuhong; Peng, Jinzhu [3] who used Kinect sensor to sense the hand gesture and get various readings of the distance and depth of the hand from the source, then manipulating the gathered information by performing hand segmentation, fingertip detection and palm point detection. Focusing at the limitation of distance of gesture recognition, their article proposed an improved threshold segmentation method with depth information of the gesture by the Kinect sensor and also presents the cosine curvature algorithm for detection of fingertips. In 2010, the skin pigment parameters such as hue, saturation and value of RGB along with Laplacian filter was used to extract the hand postures.

Similarly, some others used YCbCr color model, input gestures such as hand gloves and HSI indexing models to detect edges of hand postures. The most promising way of implementation of hand gesture recognition system are by using Vision systems. Vision systems leverage the power of machine learning to help the system recognize gestures on basis of what it has previously learnt. Computer vision attempts to replicate a human's cognitive ability of interpretation of what he/she sees. Computer Vision systems basically are intelligent enough to make sense out of an image, identify objects, context of the image. Vision systems are mostly based upon neural networks which are just a bio-mimicry of a human brain's network of neurons. One of the implementations used Convolutional neural networks to detect and recognize the hand gestures and the also use the error back propagation algorithm for training their model. A research implements Max Pool Convolutional neural networks or MPCNN along with morphological image processing to remove the noise which increases the accuracy of the model.

3. PROPOSED SYSTEM

Our proposed architecture uses neural network to generate a prediction label from the input image of a hand. The label is then sent to the client music player application over the network. This received label is then used to initiate a function call of the application. The goal of this system is to provide real time detection and hence establish control of the music player application. It functions on a server-client architecture. The server system has a capable Graphics Processing Unit which is vital to the system as higher accuracy and immediate detection is an important feature. The detection of the hand gesture is performed using YOLO(v3) algorithm. This algorithm has proven to be one of the fastest when it comes to static detection. It is based on CNN i.e. Convolutional neural networks but unlike CNN, it uses the input image only once. The architecture of the proposed system is shown in fig 3.1

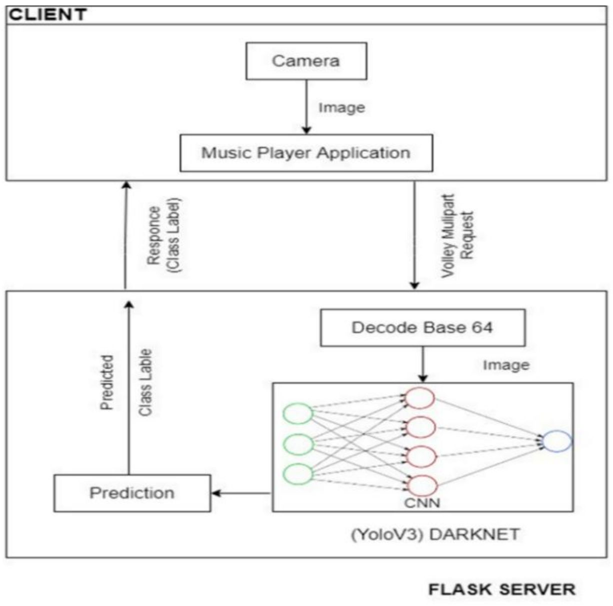


Fig. 3.1: Architecture

Firstly, the flask server is connected to the mobile device. The android application then initiates the camera after one of the audio files is first played manually by touch. The captured image is then passed on to the flask server using Volley-Multipart-Request. HTTP works as a request-response protocol between a client and server. We employ the POST method of HTTP. Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. One of the biggest advantages of Volley is that it holds all responses in memory during parsing along with Effective request cache and memory management. On receiving the image, the flask server then evokes the python script for darknet framework. The image is provided as input to the script.

Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Third, YOLO learns generalizable representations of objects. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. YOLO uses the following steps: -

YOLO first takes an input image. The framework then divides the input image into grids (say a 3X3 grid)

The network predicts 4 coordinates for each bounding box, tx, ty, tw, th. If the cell is offset from the top left corner of the image by (cx; cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to:

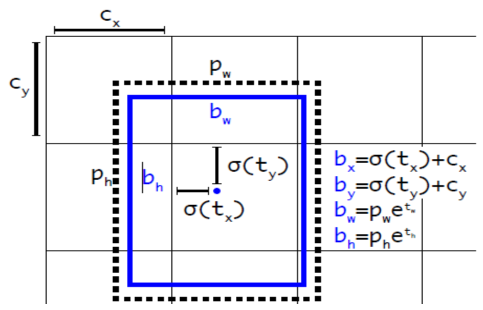
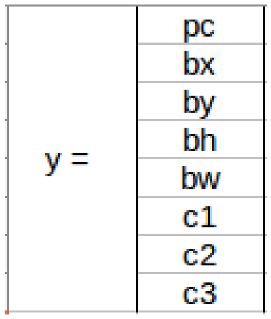


Fig. 3.2: Yolo: Bounding Box

Image classification [13](13) and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects. We need to pass the labelled data to the model in order to train it. Then, a vector is generated for each image using the notations to detect objects (if any) in the image. Each box predicts the classes the bounding box may contain using multilabel classification. During training we use binary cross-entropy loss for the class predictions.



y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

Fig. 3.3: Yolo: Vector

Here, pc defines whether an object is present in the grid (probability) bx, by, bh, bw specify the bounding box if there is an object c1, c2, c3 represent the classes. So, if the object is a car, c2 will be 1 and c1 & c3 will be 0, and so on.

We use a technique called as Intersection over Union which help us to predict bounding boxes. It calculates the intersection over union of the actual bounding box and the predicted bonding box (IoU = Area of the intersection / Area of the union)

Another technique is introduced to overcome multiple detections by the object detection algorithm. This is called Non-Max Suppression which ensures only single detection per object. It essentially suppresses boxes with higher IoU to improve the output of YOLO significantly.

Finally, using these vectors we can identify classes of the object detected and use it for further application. We generate appropriate labels for the classes.

The resultant label generated is then converted to string and sent using response class to the android application. The music player application on receiving the response string then triggers the function that corresponds to the hand gesture label. And hence an interface where a user can interact with the application by making hand gestures is established.

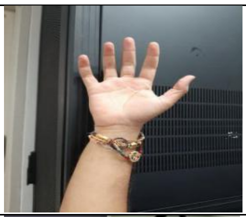
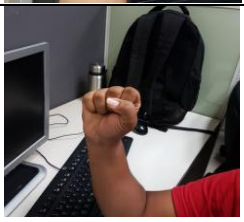
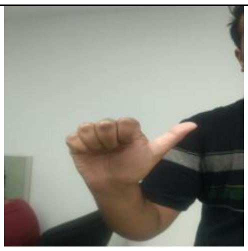
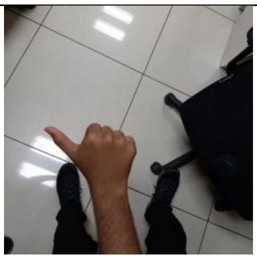
4. IMPLEMENTATION

After the architecture of the system was defined, the implementation of the system was carried out in phases as follows-

4.1 Use Case Definition:

The Music player was defined as the use case. Firstly, the most convenient static hand gestures were identified. These gestures were then assigned to a specific operation of the music player. The table shows the corresponding gestures defined for given operations.

Table 4.1: Input Gestures

Sr.no	Command	Gesture
1.	Play	
2.	Pause	
3.	Forward	
4.	Previous	

4.2 Collection of data:

For the given problem statement the data set was created manually by capturing pictures as no data set particular was found specific to the problem on the internet. The goal of create a dataset which would contribute in providing a more accurate model. Therefore, pictures of the multiple people's hand were taken in randomly varying light conditions. Also, the dataset contained pictures of all required hand gestures from different angles and all potential deviations of the ideal gestures. For example, different pictures of the play gestures contained hands of varying space between the fingers skin tones, light conditions and different angles of the hand gesture etc.

4.3 Data Preparation:

In this step of preparation, the Dataset containing more than 2000 images was firstly cleaned by going through each picture individually to make sure that the captured picture would not deviate the model in the training phase.

4.4 Data Labelling:

The model is based on CNNs which follow the supervised approach of machine learning. So, the cleaned dataset was then labelled using openCV in python. Labelling of data was done in the form of creating bounding boxes around the gesture in the images. The marked bounding boxes generate an annotation file which contains the class of the object, x- centre, y-centre, width and height of the bounding box.

The OpenCv python script will create .txt-file for each .jpg-image-file - in the same directory and with the same name, but with .txt-extension, and put to file: object number and object coordinates on this image, for each object in new line:

<object-class> <x_center> <y_center> <width> <height>

Where:

<object-class> - integer object number from 0 to (classes-1)

<x_center> <y_center> <width> <height> - float values relative to width and height of image, it can be equal from (0.0 to 1.0]

for example: <x> = <absolute_x> / <image_width> or <height> = <absolute_height> / <image_height>

atention: <x_center> <y_center> - are center of rectangle (are not top-left corner)

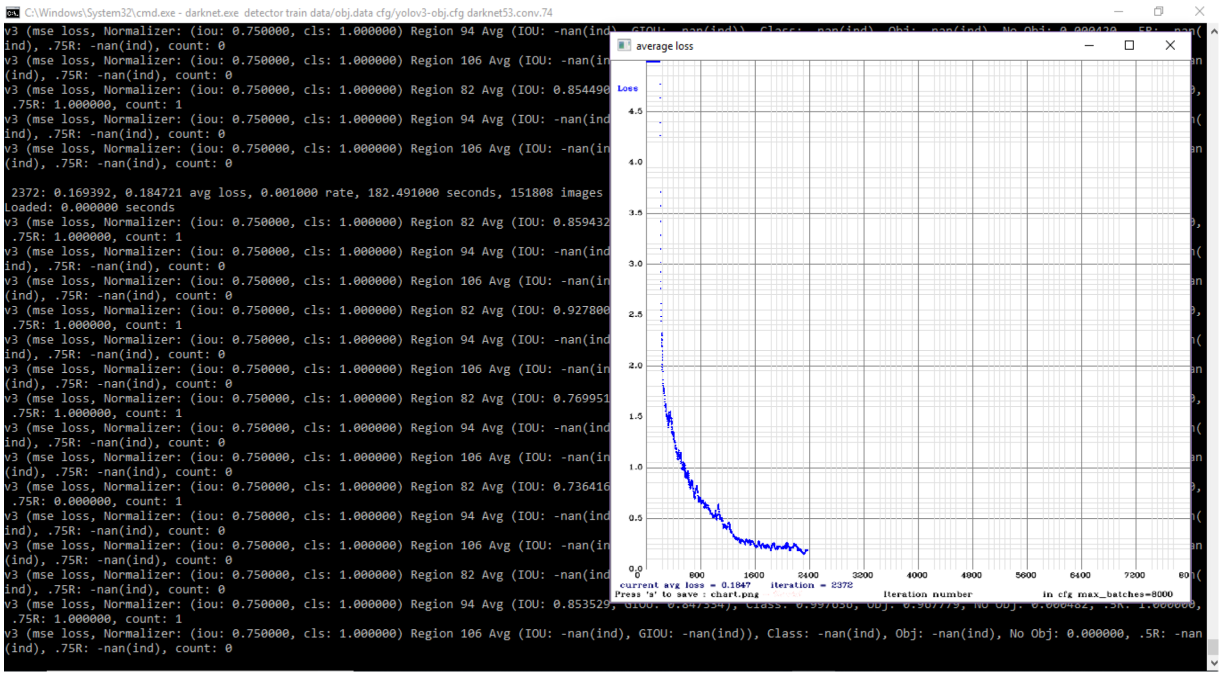
4.5 Training and testing:

Training is the most important part of the implementation. In this phase the required inputs provided to the neural network and it starts adjusting its weight values till a point where the predictions made by the neural network matches the expected outcome. The model selected for the application is Convolutional neural networks as mentioned earlier and the algorithm applied is Yolov3. The open source darknet framework provides a neural net which is customized by defining the classes and setting optimum batch sizes for the objects and modifying the the filters for different layers. These parameters of the darknet are first set to optimum values as per requirement.

Second step for training is to create a input train.txt file which will be used as input interface for the darknet. Create file train.txt in directory of the darknet build, with filenames of the labelled images, each filename in new line, with path relative to darknet.exe.

The training process here follows the transfer learning approach. Pre-trained weight values for some random objects is provided to the neural net and as the learning process goes on and the avg. loss parameter decreases the weight values are adjusted to suit the current application.

The following image represents the training process of the darknet framework:



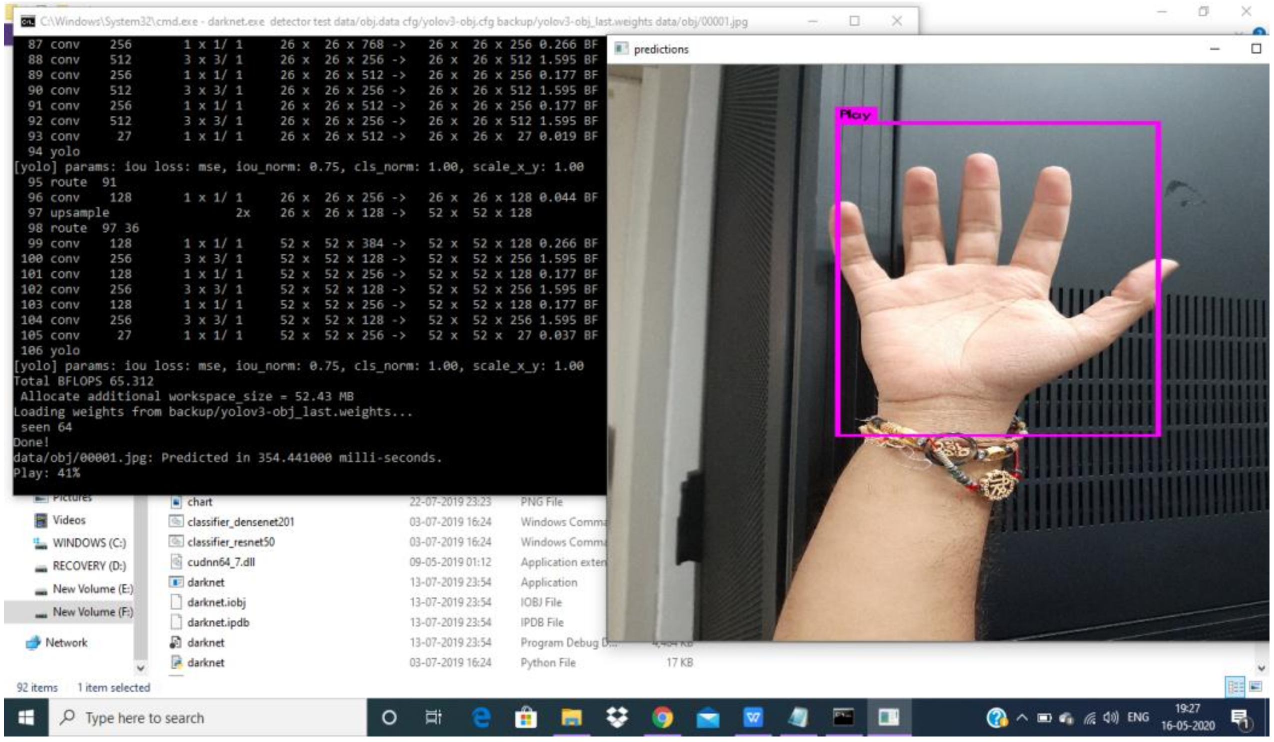
```
C:\Windows\System32\cmd.exe - darknet.exe  detector train data/obj.data cfg/yolov3-obj.cfg darknet53.conv.74
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: -nan(ind), GIOU: -nan(ind)), Class: -nan(ind), Obj: -nan(ind), No Obj: 0.000420, .5R: -nan(
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(in
(ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 82 Avg (IOU: 0.854490
 .75R: 1.000000, count: 1
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: -nan(ind
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(in
(ind), .75R: -nan(ind), count: 0

 2372: 0.169392, 0.184721 avg loss, 0.001000 rate, 182.491000 seconds, 151808 images
Loaded: 0.000000 seconds
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 82 Avg (IOU: 0.859432
 .75R: 1.000000, count: 1
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: -nan(ind
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(in
(ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 82 Avg (IOU: 0.927800
 .75R: 1.000000, count: 1
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: -nan(ind
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(in
(ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 82 Avg (IOU: 0.769951
 .75R: 1.000000, count: 1
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: -nan(ind
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(in
(ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 82 Avg (IOU: 0.736416
 .75R: 0.000000, count: 1
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: -nan(ind
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(in
(ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 82 Avg (IOU: -nan(ind
ind), .75R: -nan(ind), count: 0
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 94 Avg (IOU: 0.853529, GIOU: 0.847334), Class: 0.997036, Obj: 0.967779, No Obj: 0.000482, .5R: 1.000000,
 .75R: 1.000000, count: 1
v3 (mse loss, Normalizer: (iou: 0.750000, cls: 1.000000) Region 106 Avg (IOU: -nan(ind), GIOU: -nan(ind)), Class: -nan(ind), Obj: -nan(ind), No Obj: 0.000000, .5R: -nan
(ind), .75R: -nan(ind), count: 0
```

Fig 4.1: Training

For validation of the results of neural net a valid.txt file contains filenames of the labelled images similar to train.txt. The validation set contains 20% of the total dataset.

The graph in the Fig 4.1 represents the avg. loss value on the y-axis and number of iterations on the x-axis. The graph basically provides an insight on how well the neural net has learned. The avg. loss values corresponds to the error in the detection, lesser the value of avg loss better the accuracy of the prediction made by the model.

As the iterations progress the weights of the neural network are updates at regular intervals of 100 iterations (can be customized as needed). When the value of avg. loss becomes constant the training can be stopped. For our particular application 2400 iterations were performed and the latest weight file was stored to make predictions on real time data (fig 4.2).



```
  87 conv    256       1 x 1/ 1     26 x  26 x 768 ->   26 x  26 x 256 0.266 BF
  88 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  89 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  90 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  93 conv     27       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x  27 0.019 BF
  94 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  95 route  91
  96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
  97 upsample                2x     26 x  26 x 128 ->   52 x  52 x 128
  98 route  97 36
  99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
 100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 105 conv     27       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x  27 0.037 BF
 106 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.312
 Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3-obj_last.weights...
 seen 64
Done!
data/obj/00001.jpg: Predicted in 354.441000 milli-seconds.
Play: 41%
```

```
chart                    22-07-2019 23:23   PNG File
classifier_densenet201   03-07-2019 16:24   Windows Comma
classifier_resnet50      03-07-2019 16:24   Windows Comma
cudnn64_7.dll            09-05-2019 01:12   Application exten
darknet                  13-07-2019 23:54   Application
darknet.iobj             13-07-2019 23:54   IOBJ File
darknet.ipdb             13-07-2019 23:54   IPDB File
darknet                  13-07-2019 23:54   Program Debug D...
darknet                  03-07-2019 16:24   Python File          17 KB
```

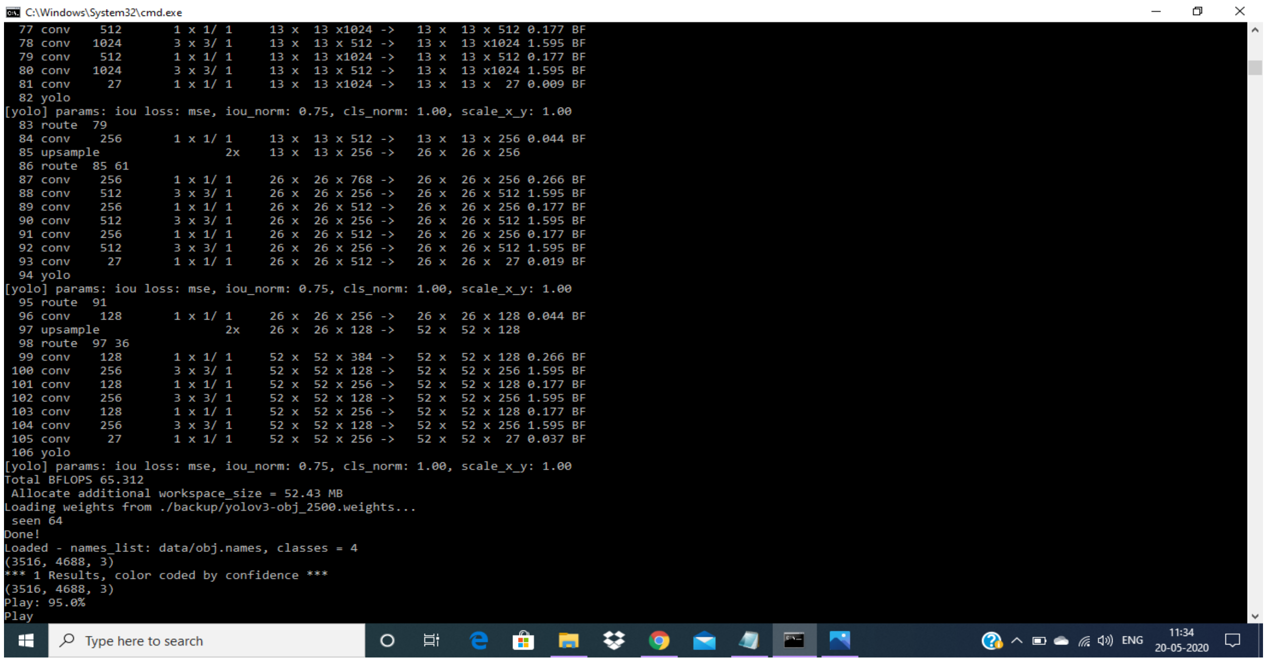
Fig 4.2: Real time detection

4.6 Integration with Android application:

A sample music player android app was developed which used request-response http methods to send image using VolleyMultipartRequest as base64 string to the Server.

A Python flask sever was created which achieved integration with the target android application by accepting the image from the client application. The server First decoded the base64string and then passed the decoded image to the trained neural network in the darknet framework. The output generated label string which represented the name of the detected gesture. This Label string is then passed using http response to the android application.

On receiving the response string, certain if conditions then allow the corresponding intended functions like play or pause to be called. Hence achieving an integration between the Darknet framework and the android application.


C:\Windows\System32\cmd.exe

```
  77 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  78 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  79 conv    512       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  80 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  81 conv     27       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x  27 0.009 BF
  82 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  83 route  79
  84 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
  85 upsample                2x     13 x  13 x 256 ->   26 x  26 x 256
  86 route  85 61
  87 conv    256       1 x 1/ 1     26 x  26 x 768 ->   26 x  26 x 256 0.266 BF
  88 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  89 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  90 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  91 conv    256       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  92 conv    512       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  93 conv     27       1 x 1/ 1     26 x  26 x 512 ->   26 x  26 x  27 0.019 BF
  94 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  95 route  91
  96 conv    128       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
  97 upsample                2x     26 x  26 x 128 ->   52 x  52 x 128
  98 route  97 36
  99 conv    128       1 x 1/ 1     52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
 100 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 101 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 102 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 103 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 105 conv     27       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x  27 0.037 BF
 106 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.312
 Allocate additional workspace_size = 52.43 MB
Loading weights from ./backup/yolov3-obj_2500.weights...
 seen 64
Done!
Loaded - names_list: data/obj.names, classes = 4
(3516, 4688, 3)
*** 1 Results, color coded by confidence ***
(3516, 4688, 3)
Play: 95.0%
Play
```

Fig 4.3: Server Execution

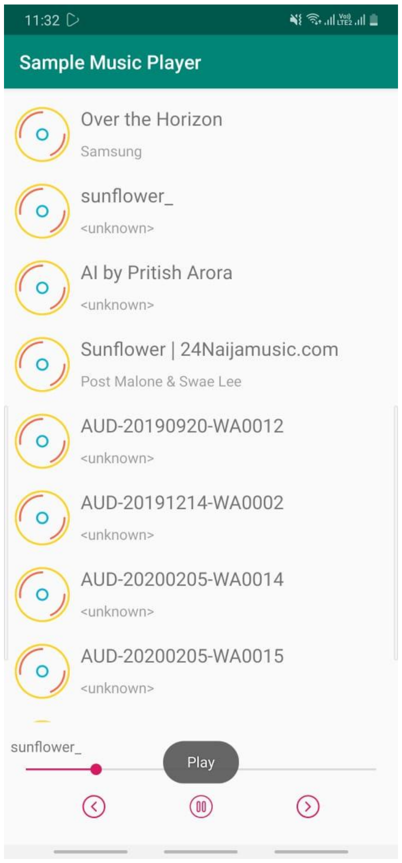
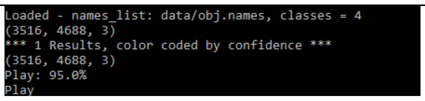
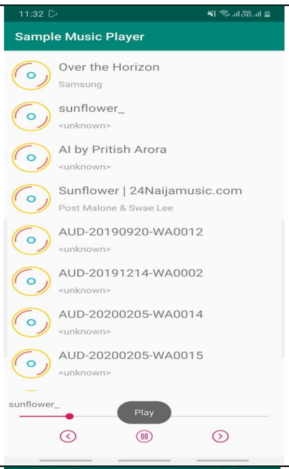


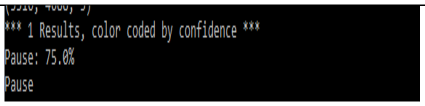
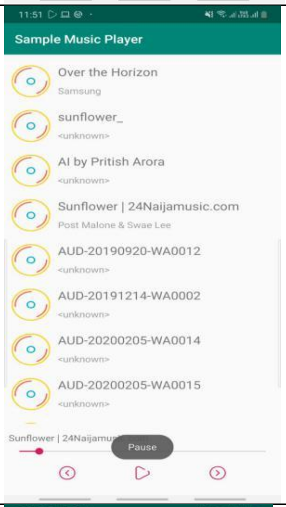
Fig 4.5: Client Application interface

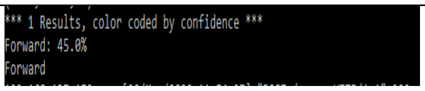
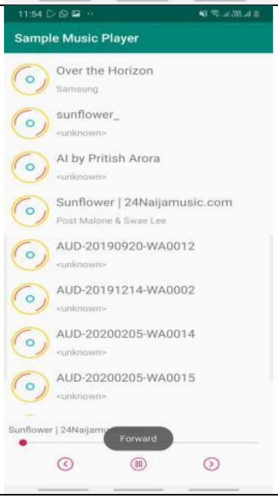
5. RESULT –

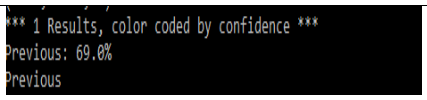
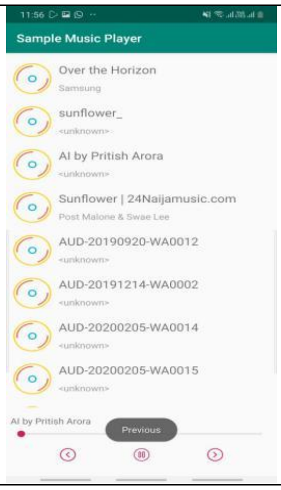
Table 4.1: Result

S.No.	Gesture	Server	Client

1	Play	
		`Loaded - names_list: data/obj.names, classes = 4`
		`(3516, 4688, 3)`
		`*** 1 Results, color coded by confidence ***`
		`(3516, 4688, 3)`
		`Play: 95.0%`
		`Play`
			
			11:32
			Sample Music Player
			Over the Horizon
			Samsung
			sunflower_
			<unknown>
			AI by Pritish Arora
			<unknown>
			Sunflower | 24Naijamusic.com
			Post Malone & Swae Lee
			AUD-20190920-WA0012
			<unknown>
			AUD-20191214-WA0002
			<unknown>
			AUD-20200205-WA0014
			<unknown>
			AUD-20200205-WA0015
			<unknown>
			sunflower_
			Play

2	Pause	
		`*** 1 Results, color coded by confidence ***`
		`Pause: 75.0%`
		`Pause`
			
			11:51
			Sample Music Player
			Over the Horizon
			Samsung
			sunflower_
			<unknown>
			AI by Pritish Arora
			<unknown>
			Sunflower | 24Naijamusic.com
			Post Malone & Swae Lee
			AUD-20190920-WA0012
			<unknown>
			AUD-20191214-WA0002
			<unknown>
			AUD-20200205-WA0014
			<unknown>
			AUD-20200205-WA0015
			<unknown>
			Sunflower | 24Naijamu
			Pause

3	Forward	
		`*** 1 Results, color coded by confidence ***`
		`Forward: 45.0%`
		`Forward`
			
			11:54
			Sample Music Player
			Over the Horizon
			Samsung
			sunflower_
			<unknown>
			AI by Pritish Arora
			<unknown>
			Sunflower | 24Naijamusic.com
			Post Malone & Swae Lee
			AUD-20190920-WA0012
			<unknown>
			AUD-20191214-WA0002
			<unknown>
			AUD-20200205-WA0014
			<unknown>
			AUD-20200205-WA0015
			<unknown>
			Sunflower | 24Naijam
			Forward

4	Previous	 `*** 1 Results, color coded by confidence ***`
		`Previous: 69.0%`
		`Previous`
			 Sample Music Player
			Over the Horizon
			Samsung
			sunflower_
			<unknown>
			AI by Pritish Arora
			<unknown>
			Sunflower | 24Naijamusic.com
			Post Malone & Swae Lee
			AUD-20190920-WA0012
			<unknown>
			AUD-20191214-WA0002
			<unknown>
			AUD-20200205-WA0014
			<unknown>
			AUD-20200205-WA0015
			<unknown>
			AI by Pritish Arora
			Previous

The above table shows the detection made by the server and the reflection of the resultant detection on the client application. The input images are similar to the ones shown in Table 4.1.

6. CONCLUSION

In today's world, there are many services available to provide data. Some need physical contact to every application and some without it. Using physical touch (talk, hand gesture etc.). There are applications that are regulated using the current and Intelligent input facility by hand gesture. User of this system can manage remote application without having to use mouse and keyboard. The program includes a flexibility of describing common user-interest movements. Command which makes the application more physically useful for challenged people, as the gesture can be described according to the its viability.

REFERENCES

[1] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018)
[2] “Real-Time Hand Gesture Recognition Using a Color Glove”-Luigi Lamberti & Francesco Camastra, (2011) DOI: https://doi.org/10.1007/978-3-642-24085-0_38
[3] "A Review on Vision-Based Hand Gesture Recognition and Applications" -Ananya Choudhury , Anjan Kumar Talukdar and Kandarpa Kumar Sarma (2015)
[4] "Real-Time System to Recognize Static Gestures of Brazilian Sign Language (Libras) alphabet using Kinect"-Mauro dos Santos Anjo, sEdnaldo Brigante Pizzolato, Sebastian Feuerstack (2016) DOI : https://doi.org/10.4018/978-1-4666-8493-5.ch011
[5] K. Ishibuchi, , F. Kishino, “Real Time Hand Gesture Recognition Using 3d Prediction Model”, IEEE International Conference On Systems: 'Systems Engineering In The Service Of Humans', 17-20 Oct. 1993, Vol. 5, pp. 324 – 328
[6] S. Ahmad, V. Tresp, “Classification With Missing And Uncertain Inputs”, IEEE International Conference On Neural Network”, 28 Mar. 1 Apr. 1993, Vol. 3, Pp. 1949
[7] S. Mu-Chun, J. Woung-Fei, C. Hsiao-Te, “A Static Hand Gesture Recognition System Using A Composite Neural Network” Proceedings Of The Fifth IEEE International Conference On Fuzzy Systems, 8-11 Sep. 1996, Vol. 2, pp. 786 - 792
[8] S. Jung, Y. Ho-Sub, W. Min, B. W. Min, “Locating Hands In Complex Images Using Color Analysis”, IEEE International Conference On Systems, Man, And Cybernetics, 1997. 'Computational Cybernetics And Simulation', 12-15 Oct. 1997, Vol. 3, pp. 2142 – 2146
[9] B. W. Min, Y. Ho-Sub, S. Jung, Y. Yun-Mo, E. Toshiaki, “Hand Gesture Recognition Using Hidden Markov Models”, IEEE International Conference On Systems, Man, And Cybernetics: 'Computational Cybernetics And Simulation', 12-15 Oct. 1997, Vol. 5, pp. 4232 – 4235
[10] S. Eickeler, A. Kosmala, G. Rigoll, “Hidden Markov Model Based Continuous Online Gesture Recognition”, International Conference On Pattern Recognition (ICPR), Aug. 1998, pp.1206-1208
[11] L. Bretzner, I. Laptev, T. Lindeberg, “Hand Gesture Recognition Using Multi-Scale Colour Features, Hierarchical Models And Particle Filtering”, Proc. Fifth IEEE International Conference On Automatic Face And Gesture Recognition, 20-21 May 2002, pp. 405 – 410

[12] Y. Ming-Hsuan, N. Ahuja, M. Tabb, "Extraction Of 2d Motion Trajectories And Its Application To Hand Gesture Recognition", IEEE Transactions On Pattern Analysis And Machine Intelligence, Aug. 2002, Vol. 24, No. 8, pp. 1061 – 1074

[13] L. Xia, K. Fujimura, "Hand Gesture Recognition Using Depth Data", Proc. Sixth IEEE International Conference On Automatic Face And Gesture Recognition, 17-19 May 2004, pp. 529 – 534

[14] Visual Interpretation of Hand Gestures for Human- Computer Interaction: A Review"- Pavlovic, V. I., Sharma, R. & Huang, T. S. (1997) DOI: 10.1109/34.598226