

IMPLEMENTING EYE TESTING WITH SOCKET.IO

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in
Computer Science and Engineering

S SANTHOSH (37117004)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119**

MARCH 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **S SANTHOSH (37117004)** who carried out the project entitled “**IMPLEMENTING EYE TESTING WITH SOCKET.IO**” under my supervision from November 2020 to March 2021.

Internal Guide

Dr. R. AROUL CANESSANE

Head of the Department

Dr. S. VIGNESHWARI, M.E, Ph.D.,

Dr. L. LAKSHMANAN M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, S SANTHOSH (37117004) hereby declare that the Project Report entitled “**EYE TESTING with SOCKET.IO**” done under the guidance of **Dr. R. AROUL CANESSANE.**, at sathyabama institute of science and technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.SASIKALA M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI M.E., Ph.D.**, and **Dr. L. LAKSHMANAN M.E., Ph.D.**, Heads of the Department, Dept. of Computing for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R. AROUL CANESSANE.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computing who were helpful in many ways for the completion of the project.

ABSTRACT

Incorporating this technological advancement, medical supplementary equipments are a field that's yet to be virtually supported globally. This market gap is exploited in this project having a virtual eye test for eye glasses to purchased on eCommerce stores like lenscart.

To establish the connection, and to exchange data between client and server, Socket.IO uses Engine.IO. This is a lower-level implementation used under the hood. Engine.IO is used for the server implementation and Engine.IO-client is used for the client.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE No
	ABSTRACT	
1.	INTRODUCTION	
	NODE.JS	
	DEEP LEARNING	
	CNN	
	IMAGE CLASSIFIER	
	OBJECT RECOGNITION	
	IDENTIFICATION	
	1.3.2 DETECTION	
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	
	FACE CLASSIFIER	
	SCOPE OF FACE CLASSIFIER	
	ADVANTAGES	
	DISADVANTAGES	
3.	EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHM USED	
	HARDWARE REQUIREMENTS	
	SOFTWARE REQUIREMENTS	
	OVERVIEW OF SOFTWARE	
	DISCONNECT	
	CONNECT	
	HEARTBEAT	
	MESSAGE	

JSON MESSAGE

EVENT

ACK

ERROR

NOOP

ALGORITHM

CN NETWORK

IMAGE RECOGNITION

INCEPTION MODEL

TENSOR FLOW

TRANSFER LEARNING

DATA SET

RETRAINING IN DOCKER

4. RESULTS DISCUSSION AND PERFORMANCE ANALYSIS

RESULT

DISCUSSION

PERFORMANCE ANALYSIS

5. SUMMARY AND CONCLUSION

SUMMARY

CONCLUSION

FUTURE ENCHANCEMENTS

REFERENCE

APPENDIX

SOURCE CODE

SCREEN SHORTS

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
6.1	CONVERSION OF IMAGE	
6.2	IMPORTING TENSOR FLOW	
6.3	ROUND FACE DATA SET	
6.4	HEART FACE DATA SET	
6.5	DOCKER	
6.6	RESULT FOR ROUND FACE	
6.7	RESULT FOR HEART FACE	
6.8	ACCURACY PERCENTAGE	
6.9	UPLOAD A IMAGE	
6.10	OUTPUT FOR ROUND FACE	

CHAPTER 1

INTRODUCTION

NODE.JS

Node.js is a new platform and is still evolving (not even a version 1.0 has been released yet), but even in its infancy, it is probably one of the most popular platforms on the Web. It is already powering a large number of popular services. Let us take a look at what makes Node.js such a tempting and popular proposition.

DEEP LEARNING

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. A formal definition of deep learning is- neurons.

CNN

The name "convolution neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolution networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers

In deep learning, a convolution neural network (CNN or Convent) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

IMAGE CLASSIFIER

The classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity. Different varieties of the recognition problem are described in the literature:

OBJECT RECOGNITION

(also called object classification) – one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene. Blippar, Google Goggles and Like That provide stand-alone programs that illustrate this functionality.

IDENTIFICATION

An individual instance of an object is recognized. Examples include identification of a specific person's face or fingerprint, identification of handwritten digits, or identification of a specific vehicle.

DETECTION

The image data are scanned for a specific condition. Examples include detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system. Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analyzed by more computationally demanding techniques to produce a correct interpretation.

Currently, the best algorithms for such tasks are based on convolution neural networks. An illustration of their capabilities is given by the ImageNet Large Scale Visual Recognition Challenge; this is a benchmark in object classification and detection, with millions of images and hundreds of object classes. Performance of convolutional neural networks, on the Image Net tests, is now close to that of humans. The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble with images that have been distorted with filters (an increasingly common phenomenon with modern digital cameras). By contrast, those kinds of images rarely trouble humans. Humans, however, tend to have trouble with other issues. For example, they are not good at classifying objects into

finegrained classes, such as the particular breed of dog or species of bird, whereas convolution neural networks handle this with ease.

What I see



What a computer sees

08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

Fig 6.1 CONVERSION OF IMAGE

Instead of the image, the computer sees an array of pixels. For example, if the image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

AIM

EYE TEST

To find the right glasses according to the user face shape using sockey.io & deep learning

SCOPE OF FACE CLASSIFIER

First we input a known image and observed the Euclidean distance. This distance tells us how close the input image is from the image on our training set. Based on maximum and minimum distances we can make a decision of whether the face is a known face, an unknown face or not a face at all. This model can be designed using other various programming techniques and languages such as C++, JAVA, etc. but matlab and socket.io can do this with ease using its image processing applications and chat box.

ADVANTAGES

- Fast; good for non-normal distributions; can limit classification to specific land cover.
- Reuse the data for future use
- Building better user interface for users

DISADVANTAGES

classes can include pixels spectrally distant from the signature mean; does not incorporate variability; not all pixels are classified; allows class overlap.

privacy considered threat in this issue

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHM USED

HARDWARE REQUIREMENTS

1. Processor : Pentium processor intel i5 2.5GHz
2. RAM : Minimum 8GB primary memory
3. Hard Disk : Minimum 500B hard disk space
- 4 Monitor : preferably colour monitor(16-bit colour) or above
6. Compact disk drive
7. A mouse and a keyboard

SOFTWARE REQUIREMENTS

1. Operating system : Microsoft Windows 8
2. Language : Java
4. Application used : Node.js, HTTP

OVERVIEW OF SOFTWARE

DISCONNECT

When the type is zero (0), the message is a disconnect signal. This will tell socket.io to close the connection and the mentioned socket. If the endpoint is not specified, the message will be sent to the default socket, which will cause the whole socket to be closed and all the endpoints on that socket will be terminated. For example:

- Message: 0 – The result is that the socket is closed and all the connections/endpoints are terminated.
- Message: 0 : /**endpoint** – The socket connection to /**endpoint** will be closed and no messages can be sent to or from that endpoint. Other endpoints will continue to operate.

CONNECT

This message is only used for multiplexing, and is sent from the client to the server to open a new connection. Thus, this message must always have an endpoint. The first (default) socket connection is established by the handshake explained earlier. The endpoint may be followed by query parameters in a URL query format. If the connection is successful, the server will echo the same message, else the server can send an error message. For example:

- Message: 1 : /**endpoint** – Requests the server to open a multiplexed socket to the endpoint.

Message: 0 : /**endpoint?param=one** – Requests the server to open a multiplexed socket to the endpoint, passing a parameter called **param** with the value **one**.

HEARTBEAT

This is the heartbeat message. It must be sent from the client to the server within the timeout negotiated during the handshake. The server will reply with a heartbeat message too. In this case, we don't have an endpoint and nor is any other information required. This is because it serves the whole socket. For example:

- Message: 2 – Sends a heartbeat message to the other end.

MESSAGE

This is the message sent over the socket. In the API, this message will be sent when you are using **socket.send**, and will result in a message event on the receiving end. This message will carry data, treating it as plain text. For example:

- Message: `3:1::Some message` – This will send a message to the other end, where the message event handler will be triggered with the **Some message** message in the event data.
- Message: `3:1:/endpoint:Some message` – Again, the message will be sent to other end of the socket, but on the multiplexed endpoint.

JSON MESSAGE

This is similar to sending the message, but in this case the message has to be serialized using JSON, and it will be parsed at the other end before being sent to the handler. In version 0.6, this was done using the same API as `send()` for message, just passing a JSON message instead of a string message. But since this introduces a performance penalty over sending plain text from version 0.7 onwards, we have to use the **json** flag to send a JSON message; for example, **socket.json.send**. For example:

- Message: `4:1::{"some":"content"}` – Sends the JSON message to the other end of the socket.

EVENT

The Event message is a special kind of JSON message that is used to send events over the socket. In events, the data payload is of the form `{"name":"eventName", "args":{"some":"content"}}`.

Here, **name** is the name of the event and **args** are the parameters to be sent to the handler. The **socket.emit** call is used to send events in the applications.

The following event names are reserved and cannot be used in applications:

- **message**
- **connect**
- **disconnect**
- **open**
- **close**
- **error**
- **retry**
- **reconnect**

For example:

- Message: 5:1::{"name": "myEvent", "args":{"some": "data"}} – The result is that the event will be sent to the other end and the appropriate event handler will be invoked, passing the args to it.

ACK

The acknowledgment (ACK) message will be sent when the message is received, with ACK request enabled; or, it can be sent out by the application. The data section in the ACK message can be the message ID for the message that is being acknowledged. If the message ID is followed by + and additional data, it is treated as an event packet. For example:

- Message: 6::1 – Sends an acknowledgment for the receipt of a message with ID 1.
- Message: 6::1+["A", "B"] – This will send an acknowledgment for the message along with the data.

ERROR

This is sent by the server in case there's an error, such as failure during the processing of a `connect` request to an endpoint. The data section of this message will contain the error message and, optionally, advice, separated by the + sign. For example:

- Message: 7::Unauthorized – The result is that the error will be sent to the client.

NOOP

This message implies no operation, and is used to close a poll after the polling times out.

ALGORITHM

CN NETWORK

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

IMAGE RECOGNITION

CNNs are often used in image recognition systems. In 2012 an error rate of 0.23 percent on the MNIST database was reported. Another paper on using CNN for image classification reported that the learning process was "surprisingly fast"; in the same paper, the best published results as of 2011 were achieved in the MNIST database and the NORB database. Subsequently, a similar CNN called AlexNet won the ImageNet Large Scale Visual Recognition Challenge 2012.

When applied to facial recognition, CNNs achieved a large decrease in error rate.^[75] Another paper reported a 97.6 percent recognition rate on "5,600 still images of more than 10 subjects". CNNs were used to assess video quality in an objective way after manual training; the resulting system had a very low root mean square error¹

The ImageNet Large Scale Visual Recognition Challenge is a benchmark in object classification and detection, with millions of images and hundreds of object classes. In the ILSVRC 2014, a large-scale visual recognition challenge, almost every highly ranked team used CNN as their basic framework. The winner GoogLeNet (the foundation of DeepDream) increased the mean average precision of object detection to 0.439329, and reduced classification error to 0.06656, the best result to date. Its network applied more than 30 layers. That performance of convolutional neural networks on the ImageNet tests was close to that of humans. The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble with images that have been distorted with filters, an increasingly common phenomenon with modern digital cameras. By contrast, those kinds of images rarely trouble humans. Humans, however, tend to have trouble with other issues. For example, they are not good at classifying objects into fine-grained categories such as the particular breed of dog or species of bird, whereas convolutional neural networks handle this.

In 2015 a many-layered CNN demonstrated the ability to spot faces from a wide range of angles, including upside down, even when partially occluded, with competitive performance. The network was trained on a database of 200,000 images that included faces at various angles and orientations and a further 20 million images without faces. They used batches of 128 images over 50,000 iterations.

INCEPTION MODEL

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batch norm is used extensively throughout the model and applied to activation inputs. Loss is computed via SoftMax.

A high-level diagram of the model is shown below:

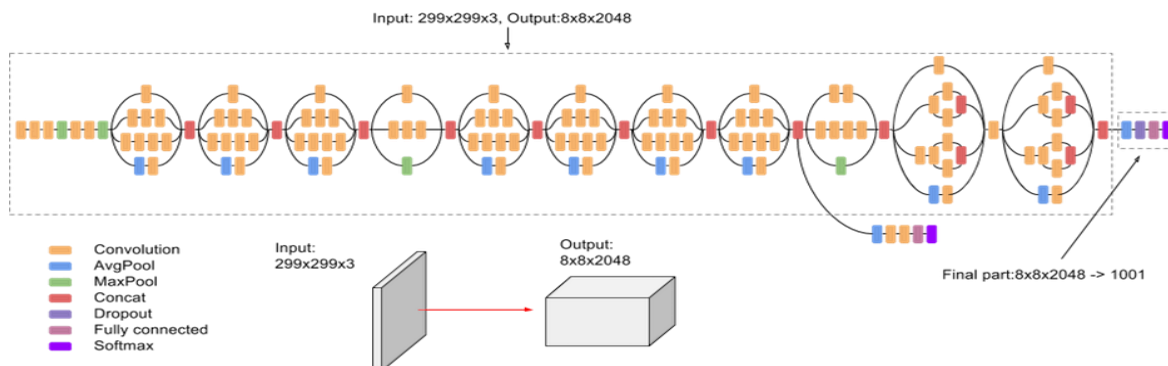


FIG 6.2 INCEPTION MODEL

TENSOR FLOW

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google

```
import os
import tensorflow as tf
from base64 import b64encode
from flask import Flask, render_template, request
```

FIG 6.3 IMPORTING TENSOR FLOW

TRANSFER LEARNING

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems

DATA SET

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files. In this we are using image data set we are collecting two different types of data set round face data set and heart face data set.

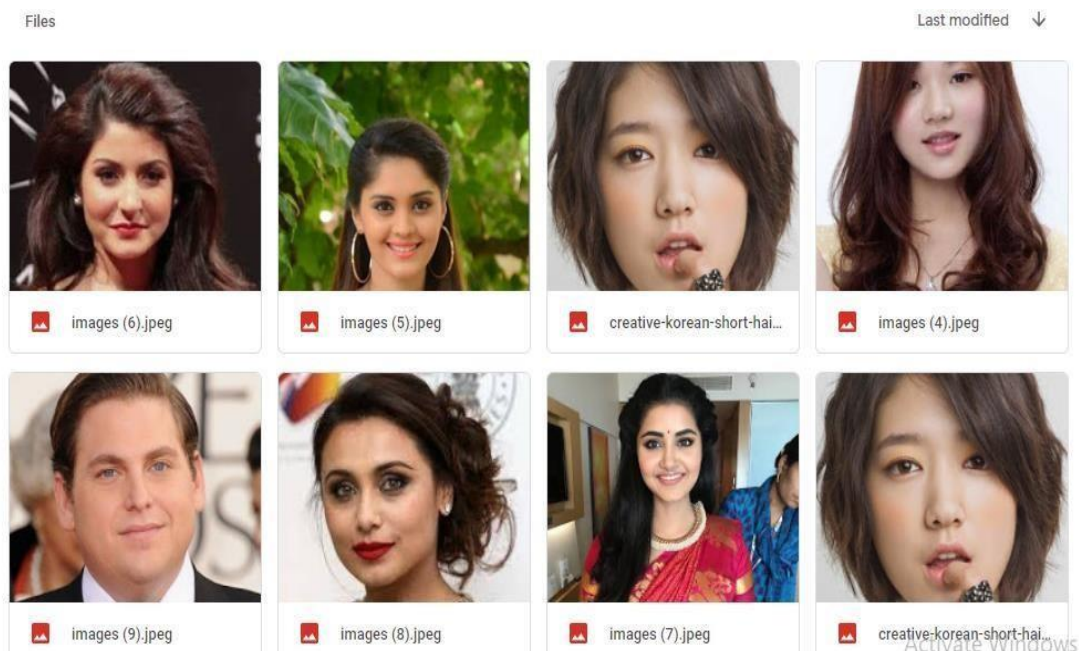


FIG 643 ROUND FACE DATA SET

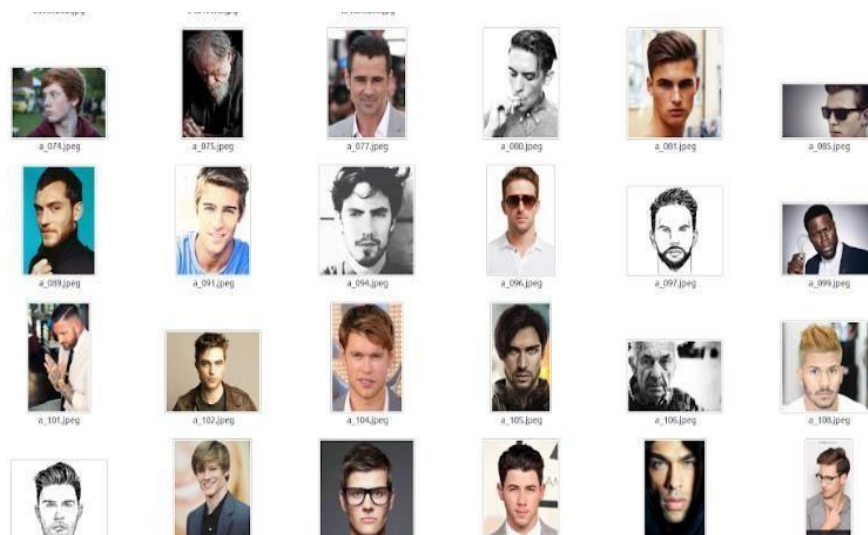
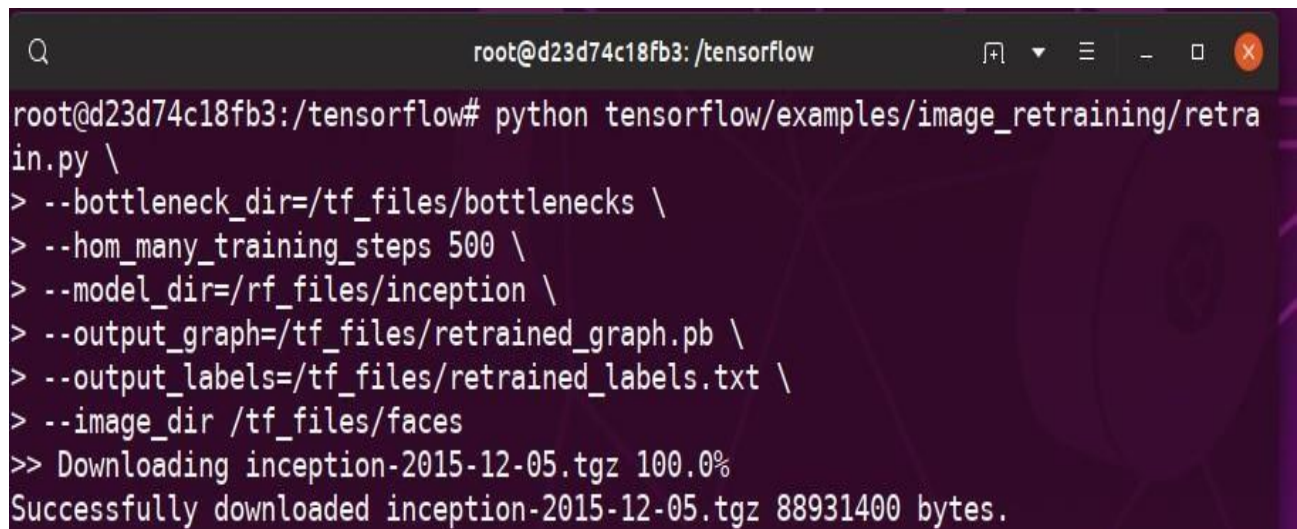


FIG 6.5 HEART FACE DATA SET

RETRAINING IN DOCKER

We are giving our heart face and round face data set to retrain the inception module by using tensor flow docker image, time duration for retraining will be based on the size of the data. The retraining script writes data to the following two files:

- `tf_files/retrained_graph.pb`, which contains a version of the selected network with a final layer retrained on your categories.
- `tf_files/retrained_labels.txt`, which is a text file containing labels.

A terminal window with a dark background and light-colored text. The window title is 'root@d23d74c18fb3: /tensorflow'. The command prompt is 'root@d23d74c18fb3:/tensorflow#'. The command entered is 'python tensorflow/examples/image_retraining/retrain.py \'. The output shows several command-line arguments being passed: '--bottleneck_dir=/tf_files/bottlenecks \', '--how_many_training_steps 500 \', '--model_dir=/rf_files/inception \', '--output_graph=/tf_files/retrained_graph.pb \', '--output_labels=/tf_files/retrained_labels.txt \', and '--image_dir /tf_files/faces'. The final output is '>> Downloading inception-2015-12-05.tgz 100.0%' followed by 'Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.' on the next line.

```
root@d23d74c18fb3:/tensorflow# python tensorflow/examples/image_retraining/retrain.py \  
> --bottleneck_dir=/tf_files/bottlenecks \  
> --how_many_training_steps 500 \  
> --model_dir=/rf_files/inception \  
> --output_graph=/tf_files/retrained_graph.pb \  
> --output_labels=/tf_files/retrained_labels.txt \  
> --image_dir /tf_files/faces  
>> Downloading inception-2015-12-05.tgz 100.0%  
Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.
```

FIG 6.6 DOCKER

CHAPTER 4 RESULTS DISCUSSION AND PERFORMANCE ANALYSIS

RESULT

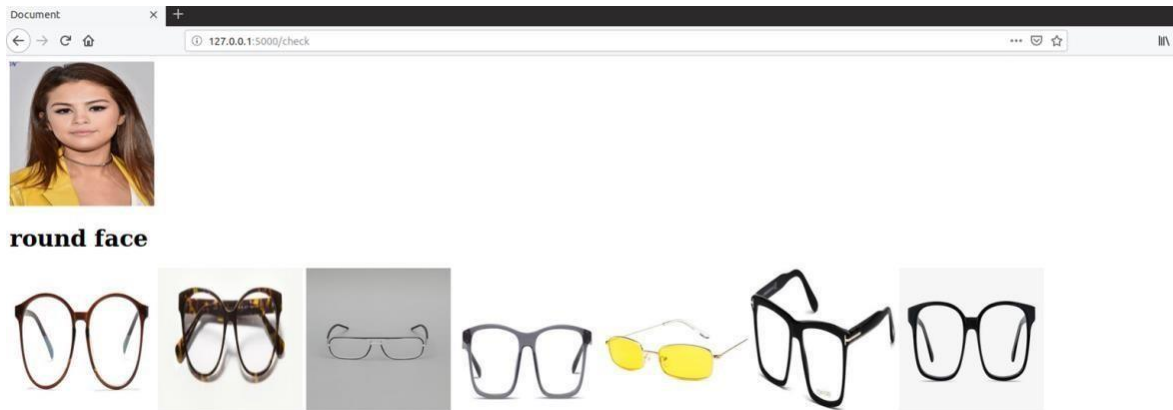


Fig 6.7 RESULT FOR ROUND FACE

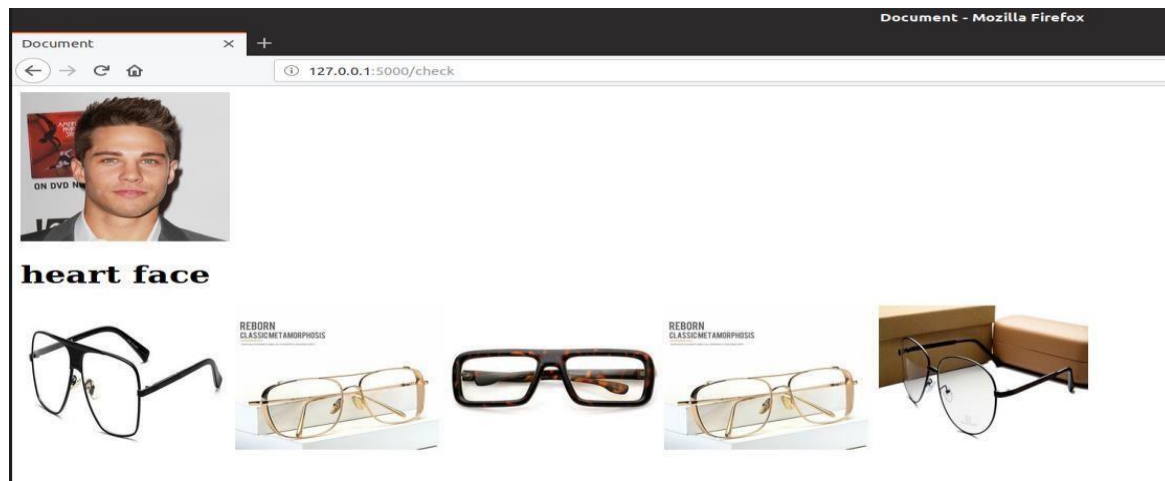
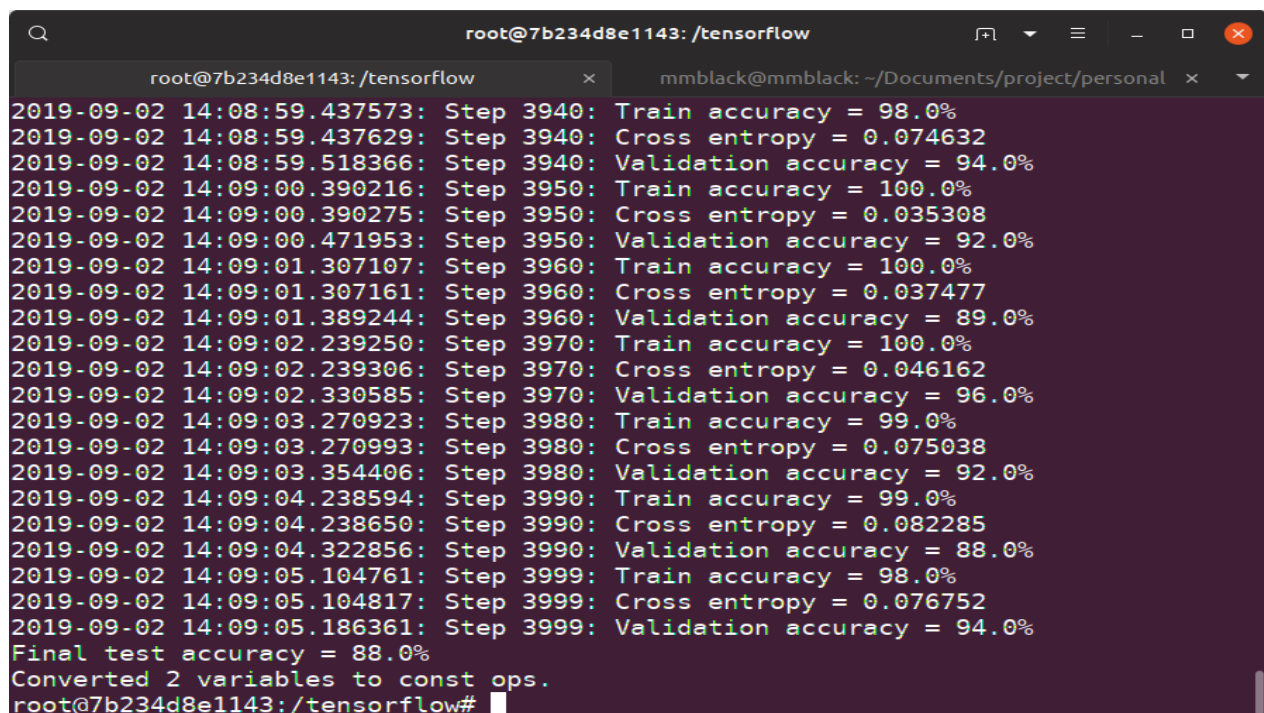


Fig 6.8 RESULT FOR HEART FACE

DISCUSSION

We started this project off by discussing the fact that convolutional neural networks trained on small images are both fast and easy to train and readily generalizable to larger image inputs. This makes them good models to build during the early experimental phase of a project, when you're still just trying to get to grips with a basic network architecture that works well. We can concentrate on dashing off quick one-off models now, and on scaling them up and fine-tuning performance later. Finally, we heard that models trained this way can often achieve equal or better performance than models trained from scratch.

PERFORMANCE ANALYSIS



```
root@7b234d8e1143: /tensorflow
2019-09-02 14:08:59.437573: Step 3940: Train accuracy = 98.0%
2019-09-02 14:08:59.437629: Step 3940: Cross entropy = 0.074632
2019-09-02 14:08:59.518366: Step 3940: Validation accuracy = 94.0%
2019-09-02 14:09:00.390216: Step 3950: Train accuracy = 100.0%
2019-09-02 14:09:00.390275: Step 3950: Cross entropy = 0.035308
2019-09-02 14:09:00.471953: Step 3950: Validation accuracy = 92.0%
2019-09-02 14:09:01.307107: Step 3960: Train accuracy = 100.0%
2019-09-02 14:09:01.307161: Step 3960: Cross entropy = 0.037477
2019-09-02 14:09:01.389244: Step 3960: Validation accuracy = 89.0%
2019-09-02 14:09:02.239250: Step 3970: Train accuracy = 100.0%
2019-09-02 14:09:02.239306: Step 3970: Cross entropy = 0.046162
2019-09-02 14:09:02.330585: Step 3970: Validation accuracy = 96.0%
2019-09-02 14:09:03.270923: Step 3980: Train accuracy = 99.0%
2019-09-02 14:09:03.270993: Step 3980: Cross entropy = 0.075038
2019-09-02 14:09:03.354406: Step 3980: Validation accuracy = 92.0%
2019-09-02 14:09:04.238594: Step 3990: Train accuracy = 99.0%
2019-09-02 14:09:04.238650: Step 3990: Cross entropy = 0.082285
2019-09-02 14:09:04.322856: Step 3990: Validation accuracy = 88.0%
2019-09-02 14:09:05.104761: Step 3999: Train accuracy = 98.0%
2019-09-02 14:09:05.104817: Step 3999: Cross entropy = 0.076752
2019-09-02 14:09:05.186361: Step 3999: Validation accuracy = 94.0%
Final test accuracy = 88.0%
Converted 2 variables to const ops.
root@7b234d8e1143: /tensorflow#
```

FIG 6.9 ACCURACY PERCENTAGE

performance for this model better than we expected. As a result of testing the model, I got a very good accuracy: 88% of correct classification samples after 500 images. The only drawback was that I had to wait about 40 minutes until 500 images come to the end (looking at the fact that I had a very small number of photos for training). If we collect more data we may achieve more accuracy.

CHAPTER 5

SUMMARY AND CONCLUSION

SUMMARY

We used transfer learning, which means we are starting with a model that has been already trained on another problem. We will then retrain it on a similar problem. Deep learning from scratch can take days, but transfer learning can be done in short order.

We are going to use a model trained on the ImageNet Large Visual Recognition Challenge dataset. These models can differentiate between 1,000 different classes, like Dalmatian or dishwasher. You will have a choice of model architectures, so you can determine the right tradeoff between speed, size and accuracy for your problem.

We will use this same model, but retrain it to tell apart a small number of classes based on our own examples.

CONCLUSION

In this work, I figured out what is deep learning. I assembled and trained the CNN model to classify photographs of round face and hear face.I have tested that this model works really well with a small number of photos. I measured how the accuracy depends on the number of data in order to detect potential overfitting problem. I determined that 2 different types of face are enough for a successful training of the model.

My next step would be to try this model on more data sets and try to apply it to practical tasks. I would also like to experiment with the neural network design in order to see how a higher efficiency can be achieved in various problems.

FUTURE ENCHANCEMENTS

- We can use this method for future use.
- We can add more facial structures .
- We can make changes by adding skin tone as a feature.
- We can create model with this same data suggesting hairstyle with it to.

REFERENCE

- [1] C.D. Manning, P. Raghavan and H. Schutze, "Introduction to Information Retrieval", Cambridge University Press, pp. 234-265, 2008.
- [2] C.D. Manning, P. Raghavan and H. Schutze, "Introduction to Information Retrieval", Cambridge University Press, pp. 234-265, 2008.
- [3] <https://codelabs.developers.google.com>.
- [4] G.T. Toussaint, "The Use of Context in Pattern Recognition," Pattern Recognition, vol. 10, 1977, pp. 189–204.
- [5] K. Abend, T.J. Harley, and L.N. Kanal, "Classification of Binary Random Patterns," IEEE Transactions on Information Theory, vol. 11, no. 4, October 1965, pp. 538–544.
- [6] C.K. Chow and C.N. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," IEEE Transactions on Information Theory, vol.14, no. 3, May 1965, pp. 462–467.
- [7] Daniel Graupe, Boris Vern, G. Gruener, Aaron Field, and Qiu Huang. "Decomposition of surface EMG signals into single fiber action potentials by means of neural network". Proc. IEEE International Symp. on Circuits and Systems, pp. 1008–1011, 1989.
- [8] Qiu Huang, Daniel Graupe, Yi Fang Huang, Ruey Wen Liu. "Identification of firing patterns of neuronal signals." In Proc. 28th IEEE Decision and Control Conf., pp. 266–271, 1989.

APPENDIX:

A. SOURCE CODE:

```
var io = require('socket.io');

exports.initialize = function(server) {
  io = io.listen(server);
  io.sockets.on("connection",function(socket){
    socket.send(JSON.stringify(
      {type:'serverMessage',
        message: 'Welcome to the most interesting chat room on earth!'}));
    socket.on('message', function(message){
      JSON.parse(message);
      if(message.type=="userMessage"){
        socket.broadcast.send(JSON.stringify(message));
        message.type="myMessage";
        socket.send(JSON.stringify(message));
      }
    });
  });
};

var socket = io.connect('/');

socket.on('message', function (data) {
  data = JSON.parse(data);
  $('#messages').append('<div class="'+data.type+'">' + data.message +
'</div>'); });

$(function(){
  $('#send').click(function(){
    var data = {
      message: $('#message').val(),    type:'userMessage'
    };
    socket.send(JSON.stringify(data));
    $('#message').val("");
  }); });

//EXISTING LAYOUT
section#nameform.modal
```

```
div.backdrop
div.popup
div.pophead Please enter a nickname
div.popbody
  input#nickname(type='text')
  input#setname(type='button', value='Set Name')
```

```
//EXISTING CSS
```

```
.modal{
  -moz-box-sizing: border-box;
  -webkit-box-sizing: border-box; box-sizing: border-box;
  -moz-box-sizing: border-box; height: -moz-calc(100% - 102px); height: -webkit-calc(100% - 102px); height: calc(100% - 102px);
  left: 0; position: absolute; top: 62px;
  width: 100%; z-index: 1000; }

.backdrop{          width: 100%;
height:100%;
  background-color: #777777; }

.popup { position: absolute; height: 100px; width: 300px; left: -moz-calc(50% - 150px); left: -webkit-calc(50% - 150px); left: calc(50% - 150px); top: -moz-calc(50% - 50px); top: -webkit-calc(50% - 50px); top: calc(50% - 50px); background: #FFFFFF; }

.pophead {
  background-color: #4192C1; color: #FFFFFF;
font-weight: bold; padding: 8px 3px; vertical-align: middle; }

.popbody { padding: 10px 5px;
}
```

```
var io = require('socket.io');

exports.initialize = function(server) {

  io = io.listen(server);

  io.sockets.on("connection", function(socket){

    socket.on('message', function(message){
```

```

message= JSON.parse(message);

if(message.type == "userMessage"){

socket.get('nickname',function(err,nickname){

message.username=nickname;

socket.broadcast.send(JSON.stringify(message));

message.type = "myMessage";

socket.send(JSON.stringify(message));

});

}

});

```

```

var socket = io.connect('/');

```

```

socket.on('name_set', function(data){
    $('#nameform').hide();
    $('#messages').append('<div class="systemMessage">' +
        'Hello '+data.name+'</ div>');
    $('#send').click(function(){
        var data = {
            message: $('#message').val(),
            type:'userMessage'
        };
        socket.send(JSON.stringify(data));
        $('#message').val("");
    });
    socket.on('message', function (data) {
        data = JSON.parse(data);
        if(data.username){
            $('#messages').append('<div class="'+data.type+
                '"><span class="name">' + data.username + ":</span> " +
data.message + '</div>');
        }else{
            $('#messages').append('<div class="'+data.type+'">' +
                data.message +
'</div>');

```

```

    }
  });
});

$(function(){ $('#setname').click(function(){ socket.emit("set_name", {name:

$('#nickname').val()}); });

});

```

B. SCREEN SHOTS

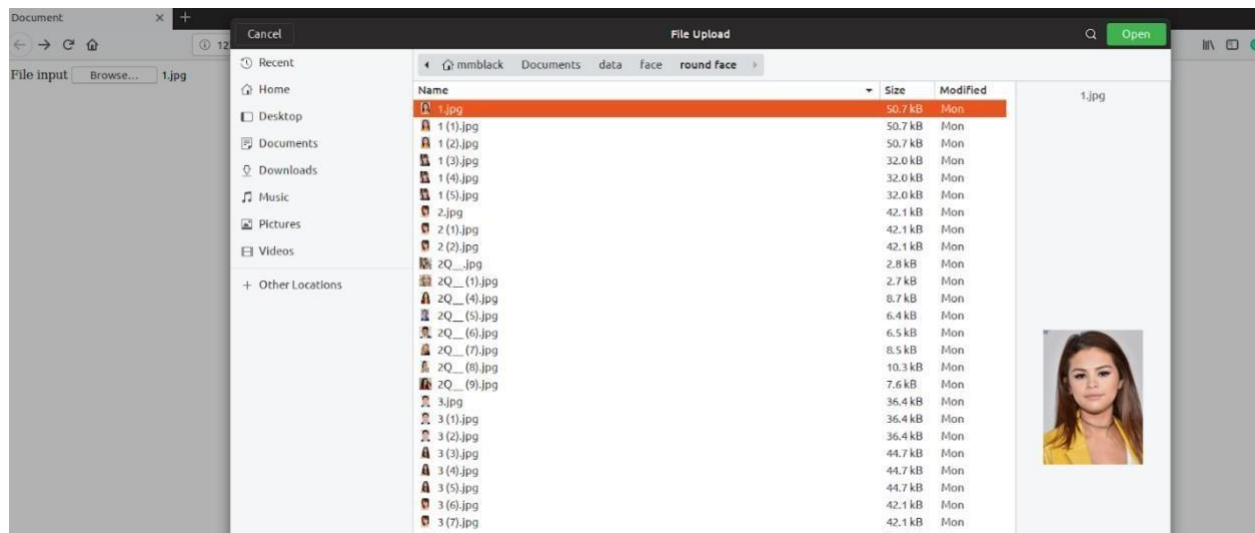


Fig 6.10 UPLOAD A IMAGE

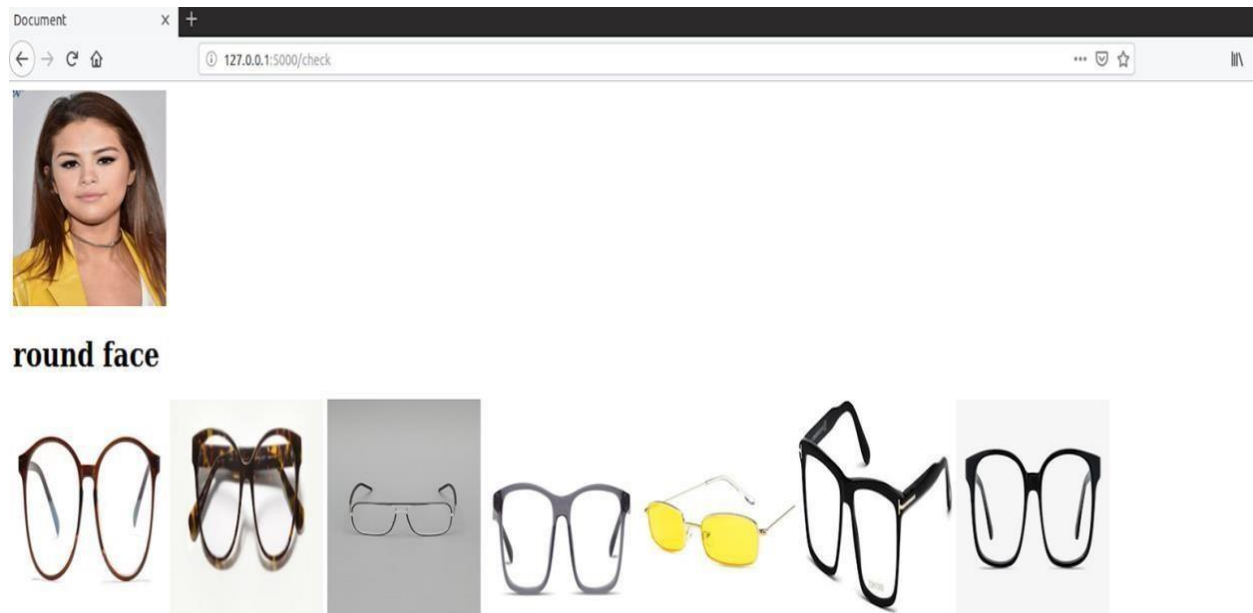


Fig 6.11 OUTPUT FOR ROUND FACE