Sign Language Alphabet Recognition In Real-Time

Thesis submitted in fulfillment of the requirements for the Degree of

Bachelor of Technology

By

DEEPESH PANDEY (Enrolment no. 17102183)

ROHAN SINGH (Enrolment No. 17102073)

Department of Electronics and Communication Engineering

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY
(Declared Deemed to be University U/S 3 of UGC Act)
A-10, SECTOR-62, NOIDA, INDIA

June 2020

TABLE OF CONTENTS

DECLARATION BY THE SCHOLARS

We hereby declare that the work reported in the minor project thesis entitled "**SIGN LANGUAGE ALPHABET RECOGNITION IN REAL-TIME**" submitted at **Jaypee Institute of Information Technology, Noida, India**, is an authentic record of our work carried out under the supervision of **Dr. Gaurav Verma**. We have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my Minor Project Thesis.

(Signature ofthe Scholar)	(Signature of the Scholar)
ROHANSINGH	DEEPESH PANDEY
DATE:…………………………	**DATE:……………………………**

SUPERVISOR’S CERTIFICATE

This is to certify that the work reported in the final report for Minor Project entitled “**SIGN LANGUAGE ALPHABET RECOGNITION IN REAL-TIME**”, submitted by **Deepesh Pandey** and **Rohan Singh** at **Jaypee Institute of Information Technology, Noida, India**, is a bona fide record of their original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

(Signature of the Supervisor)

(Dr. Gaurav Verma)

Assistant Professor (Senior Grade), ECE Department,

JIIT, Noida (U.P.) INDIA

Date:………………………………………

ACKNOWLEDGEMENT

Without the guidance and motivation of our mentors, this project would not have achieved as much as it did. These groups of people include our advisors, other faculty of the Electronics and Computer Engineering Department at JIIT, and other students. This project would not exist without the support our advisor, Assistant Professor (Senior Grade), Dr. GauravVerma.

Throughout the project they kept us motivated, and whenever we had a problem, they always had plenty of suggestions about how to solve it. Other faculty of JIIT helped us with specific challenges.

ABSTRACT

This project is a sign language alphabet recognition model using Python, openCV and Tensorflow for training the InceptionV3 model, a Convolutional Neural Network model for classification. The project aims at building a machine learning model that will be able to classify the various hand gestures used for finger-spelling in sign language. In this model, classification machine learning algorithms are trained using a giant set of image data. The project uses the dataset (1Go).

A generic image classification program that uses Google's Machine Learning library, Tensorflow and a pre-trained Deep Learning Convolutional Neural Network model called InceptionV3. This model has been pre-trained for the ImageNet Large Visual Recognition Challenge using the data from 2012, and it can differentiate between 1,000 different classes, like Dalmatian, dishwasher etc. We have applied **Transfer Learning** to this existing model and re-train it to classify a new set of images (our dataset).

A script installs the Inception model and initiates the re-training process for the specified image datasets. Once the process is complete, it returns a training accuracy somewhere around 90%. The training summaries, trained graphs and trained labels are to be saved in a folder named logs in this process.The classifier will output the predictions for each dataset in real-time as we make hand gesture for various alphabets in front of the webcam.

Note: This project is an improvement of our last semester’s minor project which used plenty of hardware. Thus, to improve the portability, accuracy and to eliminate the challenges which presented due to hardware (unreliable), we decided to make this project completely using only machine learning algorithms.

LIST OF FIGURES

CHAPTER 1

MOTIVATION

Our motivation comes from seeing the various hardships faced by people cannot speak. They suffer difficulty in communicating with other fellow normal people. They face challenges to convey very basic messages because not everyone understands sign language.

INTRODUCTION

"Sign Language Alphabet Recognition" is a project with a social purpose behind it, in which we have tried to implement a system which narrows the communication gap between mute and normal people. Speaking and hearing impaired minority make use of gestures to communicate what they are trying to say but it is impossible to be understood by all the normal people.
Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, writing or through visual aids, speech being one of the most commonly used among them. However, unfortunately, for the speaking and hearing impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency.

Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign Language consists of finger-spelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. In this project we have made an attempt to classify and identify the gestures (alphabets) in “American Sign Language (ASL)”.

EXECUTIVESUMMARY

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Current focuses in the field include emotion recognition from face and hand gesture recognition. Users can use simple gestures to control or interact with devices without physically touching them. Many approaches have been made using cameras and computer vision algorithms to interpret sign language. However, the identification and recognition of posture, gait, polemics, and human behaviors is also the subject of gesture recognition techniques. Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse.

Gesture recognition enables humans to communicate with the machine (HMI) and interact naturally without any mechanical devices. Using the concept of gesture recognition, it is possible to point a finger at the computer screen so that the cursor will move accordingly. This could make conventional input devices such as mouse, keyboards and even touch-screens redundant.

Gesture Recognition Features:

- More accurate
- Impressive Stability

- Time saving to unlock device

The major application areas of gesture recognition in the current scenario are:

- Automotive sector
- Sign language interpretation
- Consumer electronics sector
- Transit sector
- Gaming sector
- To unlock smart-phones
- In Defence Systems
- Home automation

Gesture recognition technology has been considered to be the highly successful technology as it saves time to unlock any device.

By removing the distance between the user and traditional hardware devices, our goal is to feel more like an extension of the body as opposed to an external machine. As an investigation into this idea, the goal of this project is to capture simple hand gestures and use that input to obtain a desired output.

Humans naturally use gesture to communicate. It has been demonstrated that young children can readily learn to communicate with gesture before they learn to talk. A gesture is non-verbal communication made with a part of the body. We use gesture instead of or in combination with verbal communication. Using this process, human can interface with the machine without any mechanical devices. Human movements are typically analyzed by segmenting them into shorter and understandable format.

The movements vary person to person. It can be used as a command to control different devices of daily activities, mobility etc. So our natural or intuitive body movements or gestures can be used as command or interface to operate machines, communicate with intelligent environments to control home appliances, smart home, tele-care systems etc.

CHAPTER 2

Convolutional Neural Network is a class of deep neural network that is used for Computer Vision or analyzing visual imagery.

2.1 Convolutional Layer

Computers read images as pixels and it is expressed as a matrix (NxNx3) — (height by width by depth). Images make use of three channels (RGB), so that is why we have a depth of 3.

The Convolutional Layer makes use of a set of learnable filters. A filter is used to detect the presence of specific features or patterns present in the original image (input). It is usually expressed as a matrix (MxMx3), with a smaller dimension but the same depth as the input file.

This filter is convolved (slided) across the width and height of the input file, and a dot product is computed to give an activation map.

Different filters which detect different features are convolved on the input file and a set of activation maps is outputted which is passed to the next layer in the CNN.

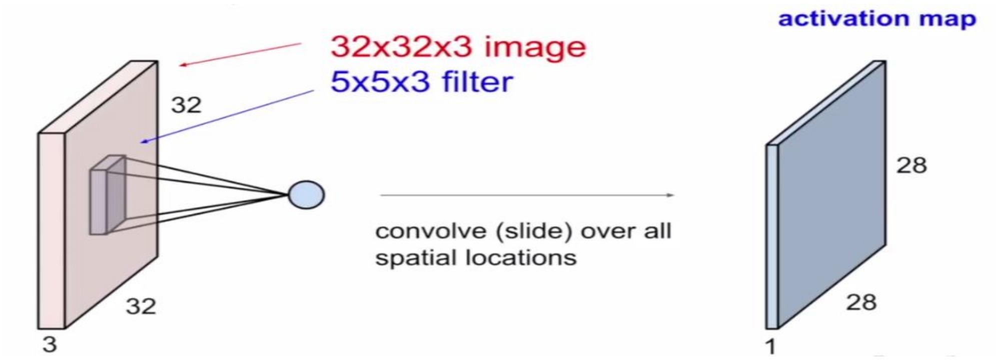


Figure 2.1 The use of filters to get Activation Maps

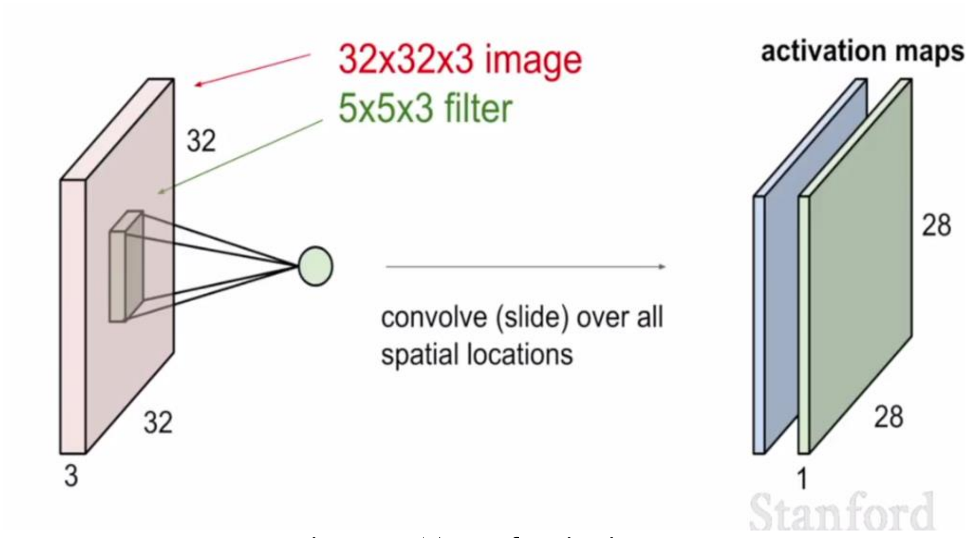


Figure 2.2 (a) Set of Activation maps

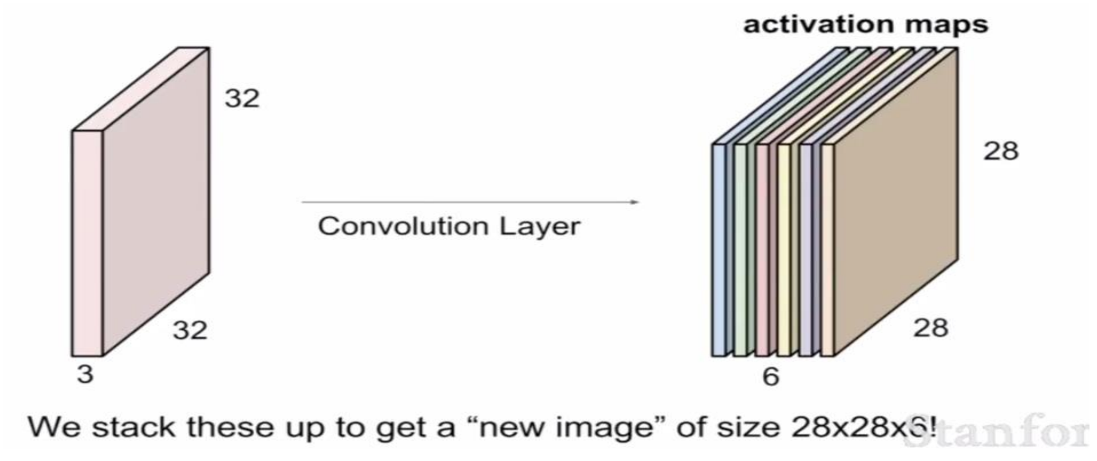


Figure2.2 (b) Set of Activation maps

There is a formula which is used in determining the dimension of the activation maps:

(N + 2P — F)/ S + 1; where N = Dimension of image (input) file

- P = Padding
- F = Dimension of filter
- S = Stride

2.2 Activation Function

The activation function is a node that is put at the end of or in between Neural Networks. They help to decide if the neuron would fire or not.

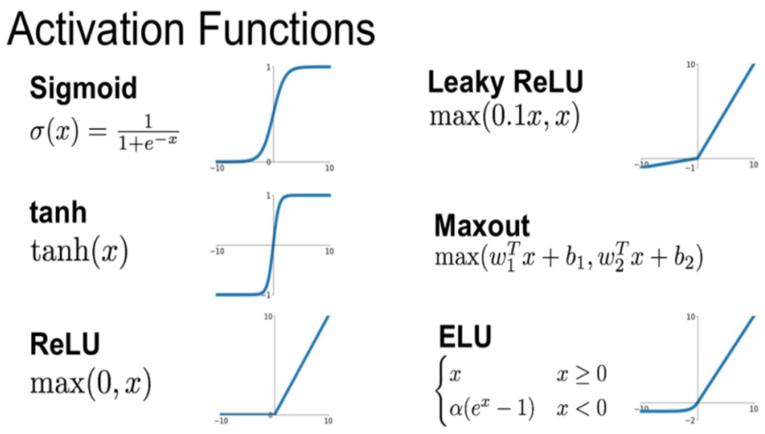


Figure 2.3 Activation Functions

We have different types of activation functions just as the figure above, but for this post, my focus will be on **Rectified Linear Unit** (ReLU).

ReLU function is the most widely used activation function in neural networks today. One of the greatest advantage ReLU has over other activation functions is that it does not activate all neurons at the same time. From the image for ReLU function above, we'll notice that it converts all negative inputs to zero and the neuron does not get activated. This makes it very computational efficient as few neurons are activated per time. It does not saturate at the positive region. In practice, ReLU converges six times faster than tanh and sigmoid activation functions.

Some disadvantage ReLU presents is that it is saturated at the negative region, meaning that the gradient at that region is zero.

With the gradient equal to zero, during back-propagation all the weights will not be updated, to fix this, we use **Leaky ReLU**. Also, ReLU functions are not zero-centered. This means that for it to get to its optimal point, it will have to use a zig-zag path which may be longer.

2.3 Pooling Layer

The Pooling layer can be seen between Convolution layers in a CNN architecture. This layer basically reduces the number of parameters and computation in the network, controlling overfitting by progressively reducing the spatial size of the network.

There are two operations in this layer; Average pooling and Maximum pooling. Only Max-pooling will be discussed in this post.

Max-pooling, like the name states; will take out only the maximum from a pool. This is actually done with the use of filters sliding through the input; and at every stride, the maximum parameter is taken out and the rest is dropped. This actually down-samples the network.

Unlike the convolution layer, the pooling layer does not alter the depth of the network, the depth dimension remains unchanged.

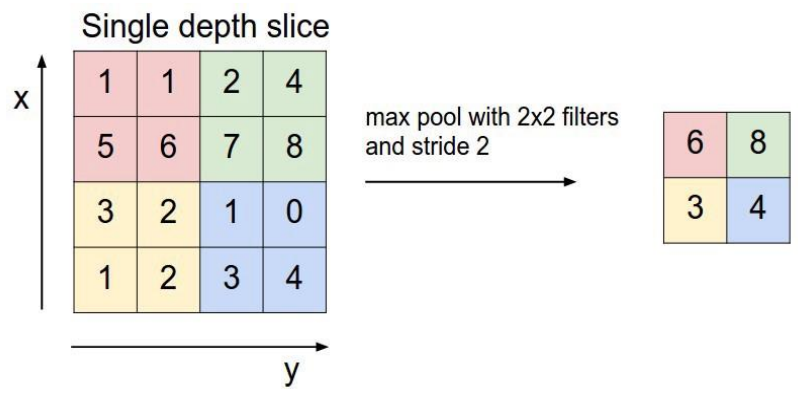


Figure 2.4 Max-pooling

Formulae for the output after Max-pooling:

- (N — F)/ S + 1; where N = Dimension of input to pooling layer
- F = Dimension of filter
- S = Stride

2.4 Fully-connected Layer

In this layer, the neurons have a complete connection to all the activations from the previous layers. Their activations can hence be computed with a matrix multiplication followed by a bias offset. This is the last phase for a CNN network.

The Convolutional Neural Network is actually made up of **hidden layers** and **fully-connected layer(s)**.

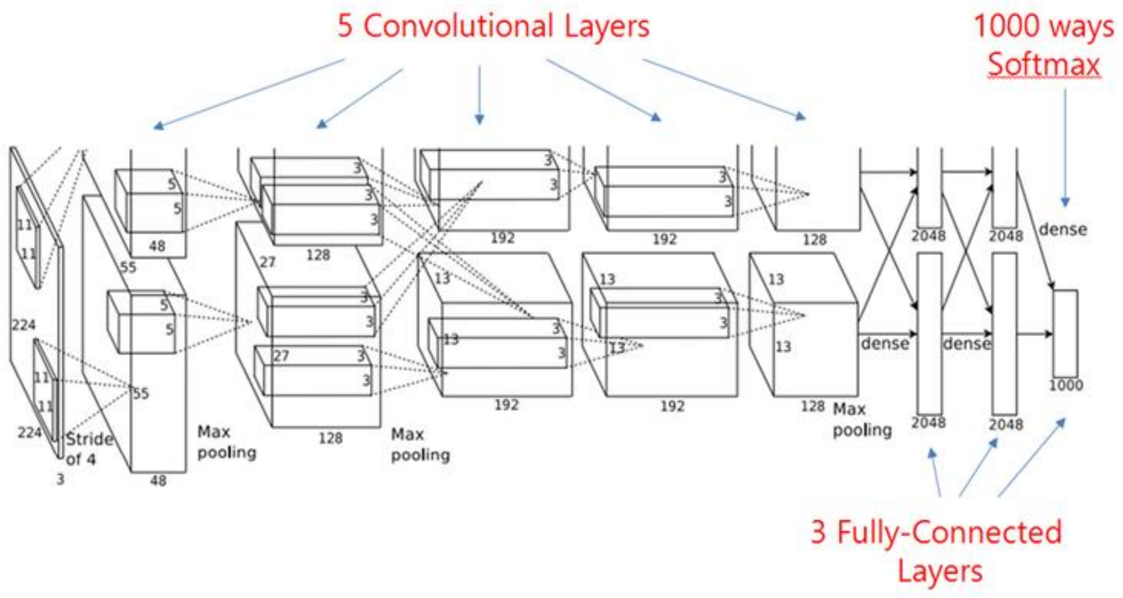


Figure 2.5 Layers of CNN

CHAPTER 3

3.1 TRANSFER LEARNING

3.1.1 What is Transfer Learning?

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related 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.

3.1.2 Examples of Transfer Learning with Deep Learning

Let’s make this concrete with common example of transfer learning with deep learning models.

3.1.2.1 Transfer Learning with Image Data

It is common to perform transfer learning with predictive modeling problems that use image data as input.

This may be a prediction task that takes photographs or video data as input.
For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition.

The research organizations that develop models for this competition and do well often release their final model under a permissive license for reuse. These models can take days or weeks to train on modern hardware.

These models can be downloaded and incorporated directly into new models that expect image data as input.

Three examples of models of this type include:

- Oxford VGG Model
- Google Inception Model(USED BY US)
- Microsoft ResNet Model

This approach is effective because the images were trained on a large corpus of photographs and require the model to make predictions on a relatively large number of classes, in turn, requiring that the model efficiently learn to extract features from photographs in order to perform well on the problem.

3.1.3 When to Use Transfer Learning ?

Transfer learning is an optimization, a shortcut to saving time or getting better performance.

In general, it is not obvious that there will be a benefit in using transfer learning in the domain until after the model has been developed and evaluated.

Three possible benefits to look for when using transfer learning:

1. **Higher start**. The initial skill (before refining the model) on the source model is higher than it otherwise would be.
2. **Higher slope**. The rate of improvement of skill during training of the source model is steeper than it otherwise would be.
3. **Higher asymptote**. The converged skill of the trained model is better than it otherwise would be.

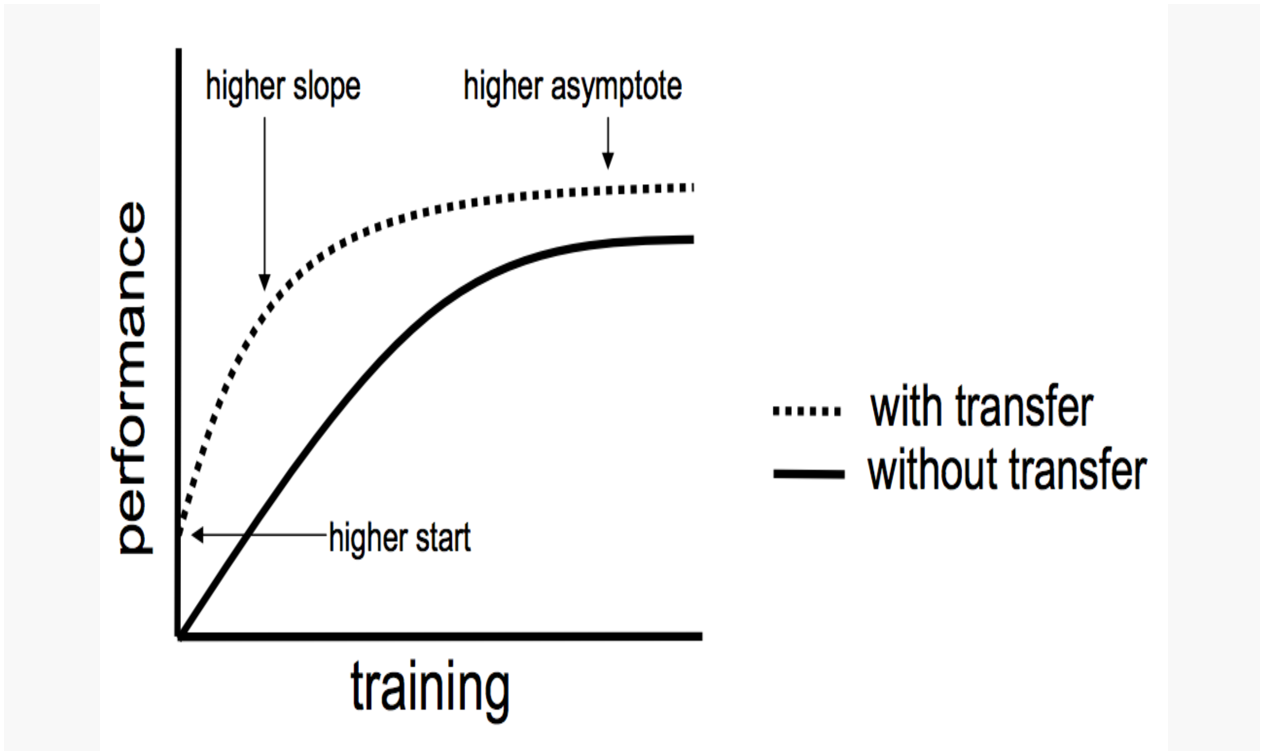


Figure 3.1 Three ways in which transfer might improve learning

Ideally, we would see all three benefits from a successful application of transfer learning.

It is an approach to try if you can identify a related task with abundant data and you have the resources to develop a model for that task and reuse it on your own problem, or there is a pre-trained model available that you can use as a starting point for yourown model.

On some problems where you may not have very much data, transfer learning can enable you to develop skillful models that you simply could not develop in the absence of transfer learning.

The choice of source data or source model is an open problem and may require domain expertise and/or intuition developed via experience.

CHAPTER 4

4.1 IMPLEMENTATION AND PROCEDURE INVOLVED

Our project is a sign language alphabet recognizer using Python, openCV and tensorflow for training InceptionV3 model, a convolutional neural network model for classification.

The project contains the dataset (1Go)

4.1.1 PACKAGES UTILISED

This project uses python 3.6 and the PIP following packages:

- openCV
- Tensorflow
- Matplotlib
- Numpy

4.1.2 Installation using PIP

All the packages are written into a text file and running this command in the python shell installs all the required packages we need.

- `pip3 install -r requirements.txt`

Requirements.txt file looks like this:

```
tensorflow==1.2.1
matplotlib==2.0.2
numpy==1.13.0
```

```
opencv-python==3.2.0.8
```

4.2 TRAINING

4.2.1 INITIATING TRANSFER LEARNING PROCESS

In the python shell we go to the project directory and run this command:

```
$ bash run.sh
```

The “run” file mentioned in above line contains a set of commands which are executed line by line which in turn on completion leads to the complete re-training of our model.

The “run” file looks like this:

```
python3 train.py \
  --bottleneck_dir=logs/bottlenecks \
  --how_many_training_steps=2000 \
  --model_dir=inception \
  --summaries_dir=logs/training_summaries/basic \
  --output_graph=logs/trained_graph.pb \
  --output_labels=logs/trained_labels.txt \
  --image_dir=./dataset
```

This script installs the Inception model and initiates the re-training process for the specified image data sets.
Once the process is complete, it returned a training accuracy somewhere around 90 %.

The training summaries, trained graphs and trained labels are saved in a foldernamed logs.

The following are few important function definitions of the “train.py” script that we run above to train our model.

defcreate_image_lists

Brief:

Builds a list of training images from the file system.Analyzes the sub folders in the image directory, splits them into stabletraining, testing, and validation sets, and returns a data structuredescribing the lists of images for each label and their paths.

--

defget_bottleneck_path

Brief:

Returns a path to a bottleneck file for a label at the given index.

----

defcreate_inception_graph

Brief:

Creates a graph from saved GraphDef file and returns a Graph object.

--

defrun_bottleneck_on_image

Brief:

Runs inference on an image to extract the 'bottleneck' summary layer.

----

defcreate_bottleneck_file(bottleneck_path, image_lists, label_name, index,

image_dir, category, sess, jpeg_data_tensor,

bottleneck_tensor):

Brief:

Create a single bottleneck file.

----

defget_or_create_bottleneck(sess, image_lists, label_name, index, image_dir,

category, bottleneck_dir, jpeg_data_tensor,

bottleneck_tensor):

Brief:

Retrieves or calculates bottleneck values for an image.If a cached version of the bottleneck data exists on-disk, return that,otherwise calculate the data and save it to disk for future use.

----

defget_random_distorted_bottlenecks(

sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,

distorted_image, resized_input_tensor, bottleneck_tensor):

Brief:

Retrieves bottleneck values for training images, after distortions. If we're training with distortions like crops, scales, or flips, we have torecalculate the full model for every image, and so we can't use cachedbottleneck values. Instead we find random images for the requested category,run them through the distortion graph, and then the full graph to get thebottleneck results for each.

---

defadd_input_distortions(flip_left_right, random_crop, random_scale,

random_brightness):

Brief:

Creates the operations to apply the specified distortions. During training it can help to improve the results if we run the imagesthrough simple distortions like crops, scales, and flips.

These reflect the kind of variations we expect in the real world, and so can help train themodel to cope with natural data more effectively. Here we take the suppliedparameters and construct a network of operations to apply them to an image.

Cropping

Cropping is done by placing a bounding box at a random position in the full image. The cropping parameter controls the size of that box relative to theinput image. If it's zero, then the box is the same size as the input and nocropping is performed. If the value is 50%, then the crop box will be half thewidth and height of the input.

Scaling

Scaling is a lot like cropping, except that the bounding box is alwayscentered and its size varies randomly within the given range. For example ifthe scale percentage is zero, then the bounding box is the same size as the input and no scaling is applied. If it's 50%, then the bounding box will be in

--

defvariable_summaries(var):

Brief:

Attach a lot of summaries to a Tensor (for TensorBoard visualization).

----

defadd_evaluation_step(result_tensor, ground_truth_tensor):

Brief:

Inserts the operations we need to evaluate the accuracy of our results.

4.3 CLASSIFYING

To test classification, we used the following command:

python3 classify.py path/to/image.jpg

4.4 USING WEBCAM

To use webcam, we use the following command after the training is complete in the python shell itself which runs the classify_webcam script:

```
python3 classify_webcam.py
```

CHAPTER 5

5.1 GESTURE RECOGNITION IN REAL-TIME

After successful training our project was ready to be tested in real time.

So I started off by putting gestures which map to a particular letter in the ASL Sign Language and test the accuracy of our project.

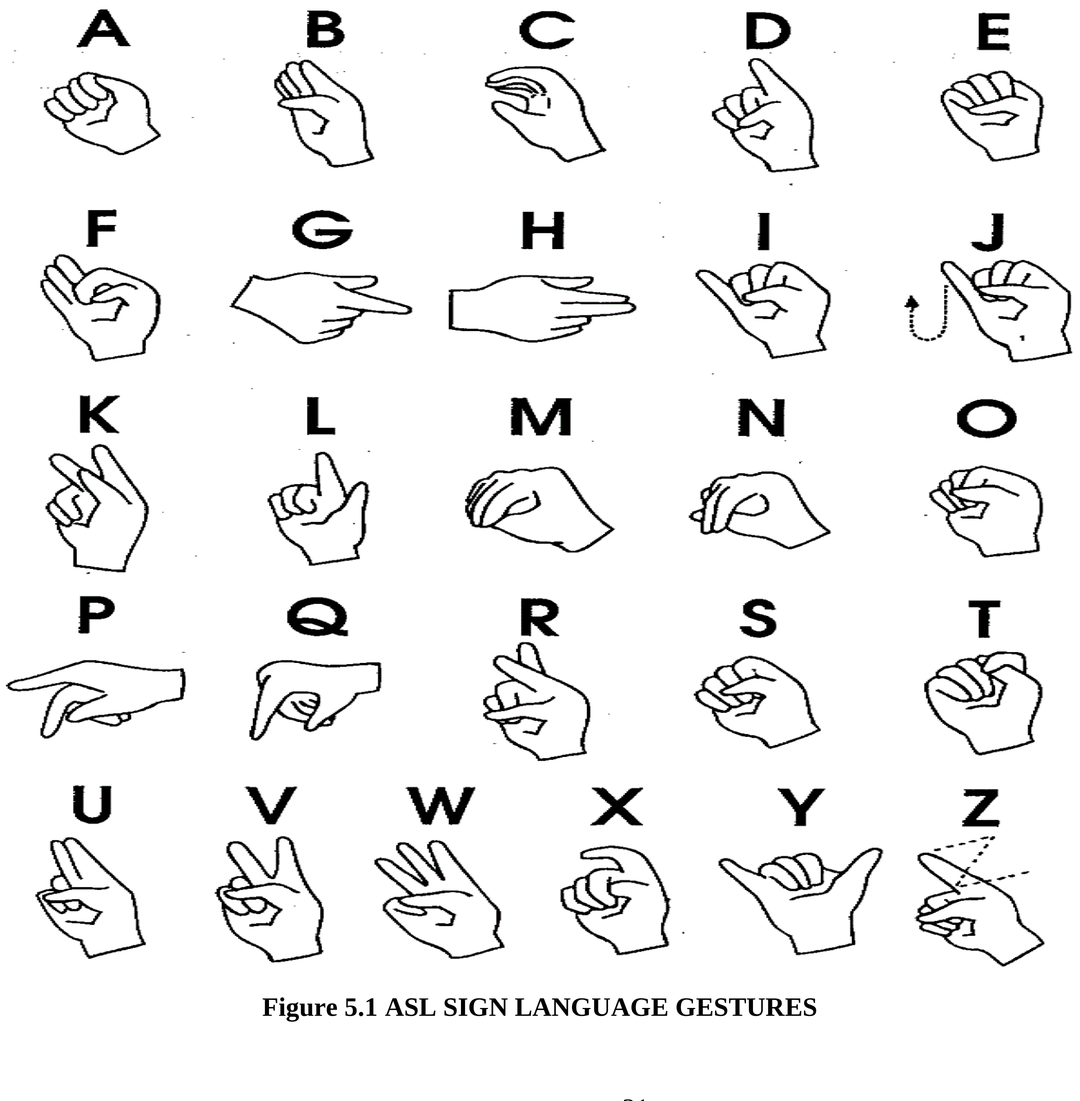


Figure 5.1 ASL SIGN LANGUAGE GESTURES

We included **three more gestures** in our dataset apart from the general **A-Z image dataset** just to increase our efficiency and minimize mistakes.

They are:

- SPACE
- DELETE
- NOTHING

SPACE:

For spacing between two words to avoid confusion and better understanding of the user.

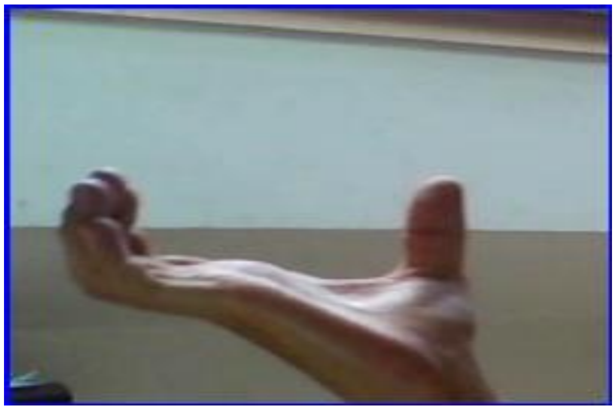


Figure 5.2 SPACE GESTURE

DELETE:

Implies that we want to delete the last entered alphabet which might have been entered by mistake or otherwise.

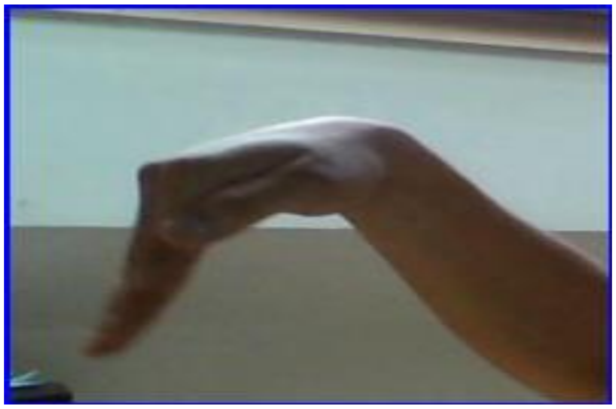


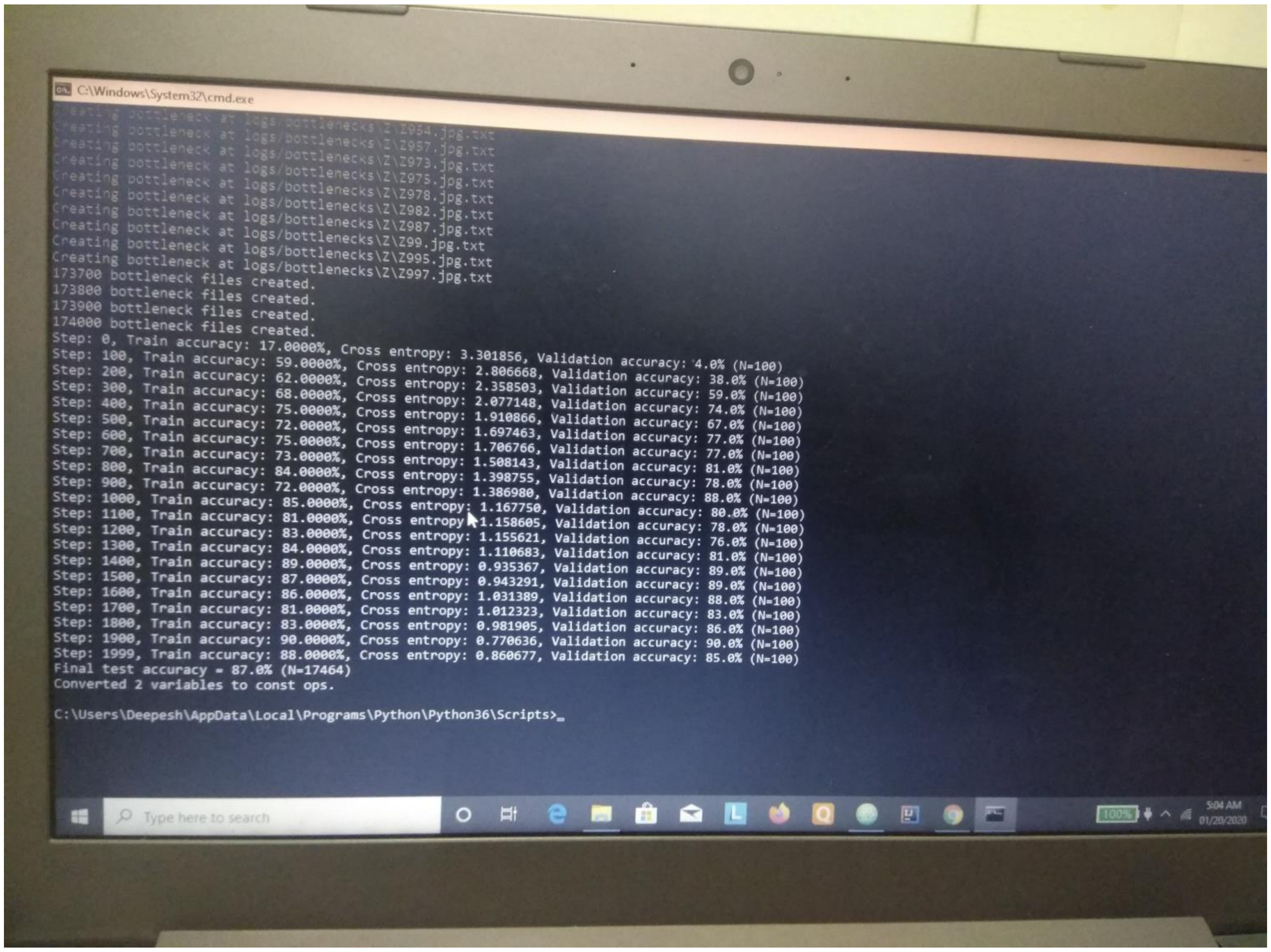
Figure 5.3 DELETE GESTURE

NOTHING:

This indicates that there is “nothing” in front of the webcam and the model leant this during its training so it doesn’t gets confused when it doesn’t sees any gesture.

Figure 5.4 NOTHING GESTURE

Then after training we got around 87 % training accuracy as shown below.



```
Creating bottleneck at logs/bottlenecks\Z\Z954.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z957.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z973.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z975.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z978.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z982.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z987.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z99.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z995.jpg.txt
Creating bottleneck at logs/bottlenecks\Z\Z997.jpg.txt
173700 bottleneck files created.
173800 bottleneck files created.
173900 bottleneck files created.
174000 bottleneck files created.
Step: 0, Train accuracy: 17.0000%, Cross entropy: 3.301856, Validation accuracy: 4.0% (N=100)
Step: 100, Train accuracy: 59.0000%, Cross entropy: 2.806668, Validation accuracy: 38.0% (N=100)
Step: 200, Train accuracy: 62.0000%, Cross entropy: 2.358503, Validation accuracy: 59.0% (N=100)
Step: 300, Train accuracy: 68.0000%, Cross entropy: 2.077148, Validation accuracy: 74.0% (N=100)
Step: 400, Train accuracy: 75.0000%, Cross entropy: 1.910866, Validation accuracy: 67.0% (N=100)
Step: 500, Train accuracy: 72.0000%, Cross entropy: 1.697463, Validation accuracy: 77.0% (N=100)
Step: 600, Train accuracy: 75.0000%, Cross entropy: 1.706766, Validation accuracy: 77.0% (N=100)
Step: 700, Train accuracy: 73.0000%, Cross entropy: 1.508143, Validation accuracy: 81.0% (N=100)
Step: 800, Train accuracy: 84.0000%, Cross entropy: 1.398755, Validation accuracy: 78.0% (N=100)
Step: 900, Train accuracy: 72.0000%, Cross entropy: 1.386980, Validation accuracy: 88.0% (N=100)
Step: 1000, Train accuracy: 85.0000%, Cross entropy: 1.167750, Validation accuracy: 80.0% (N=100)
Step: 1100, Train accuracy: 81.0000%, Cross entropy: 1.158605, Validation accuracy: 78.0% (N=100)
Step: 1200, Train accuracy: 83.0000%, Cross entropy: 1.155621, Validation accuracy: 76.0% (N=100)
Step: 1300, Train accuracy: 84.0000%, Cross entropy: 1.110683, Validation accuracy: 81.0% (N=100)
Step: 1400, Train accuracy: 89.0000%, Cross entropy: 0.935367, Validation accuracy: 89.0% (N=100)
Step: 1500, Train accuracy: 87.0000%, Cross entropy: 0.943291, Validation accuracy: 89.0% (N=100)
Step: 1600, Train accuracy: 86.0000%, Cross entropy: 1.031389, Validation accuracy: 88.0% (N=100)
Step: 1700, Train accuracy: 81.0000%, Cross entropy: 1.012323, Validation accuracy: 83.0% (N=100)
Step: 1800, Train accuracy: 83.0000%, Cross entropy: 0.981905, Validation accuracy: 86.0% (N=100)
Step: 1900, Train accuracy: 90.0000%, Cross entropy: 0.770636, Validation accuracy: 90.0% (N=100)
Step: 1999, Train accuracy: 88.0000%, Cross entropy: 0.860677, Validation accuracy: 85.0% (N=100)
Final test accuracy = 87.0% (N=17464)
Converted 2 variables to const ops.

C:\Users\Deepesh\AppData\Local\Programs\Python\Python36\Scripts>_
```

Figure 5.5 Training Accuracy of 87%

Then after successful training we try to implement and test it by running the following command on the command prompt:

```
py classify_webcam.py
```

This command launches the webcam with a rectangular box on the screen. Only the gestures made inside this rectangular box are detected and anything outside it is ignored. There is a "SCORE" field below this rectangular box which depicts how confident the prediction is on a scale of 0 to 1. A "SCORE" of 1 means 100% and a "SCORE" of 0 implies 0%. This "SCORE" is chosen on the basis of the array of all real numbers depicting the probabilities of each label being this gesture on the screen. The label having the highest probability gets selected and the "SCORE" corresponding to it returned.

As can be seen below, after I entered a random string I make an attempt to delete it using the "DELETE" gesture and is detected as shown under the rectangular box and its "SCORE" is also displayed underneath it.

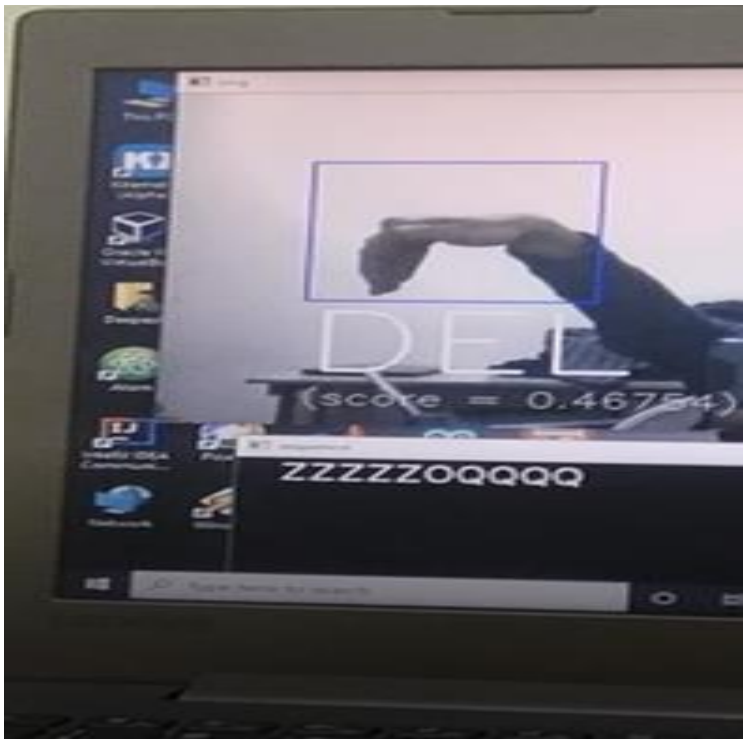
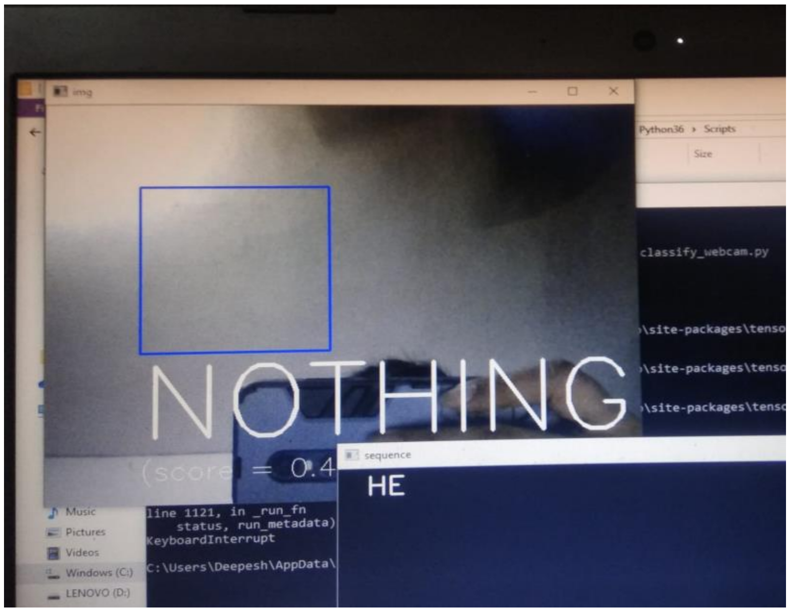


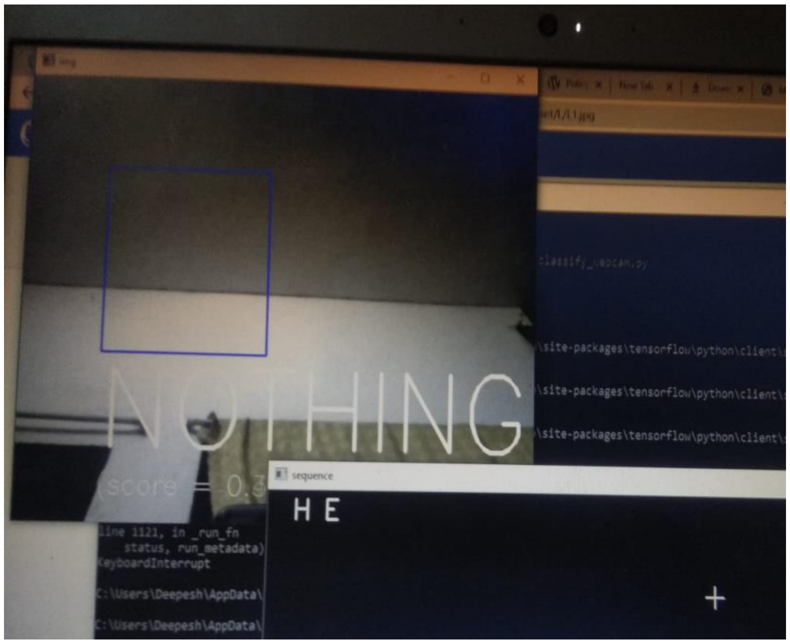
Figure 5.6 "DEL" Gesture detection in real-time

Similarly other gestures also get detected in real-time. As we can see in the first figure, we tried to display the word “HE”. In the second figure, we tried to show the usage of gesture “SPACE” by introducing it between letters “H” and “E”. In the third figure, I tried to display the message “HELP” on the screen.



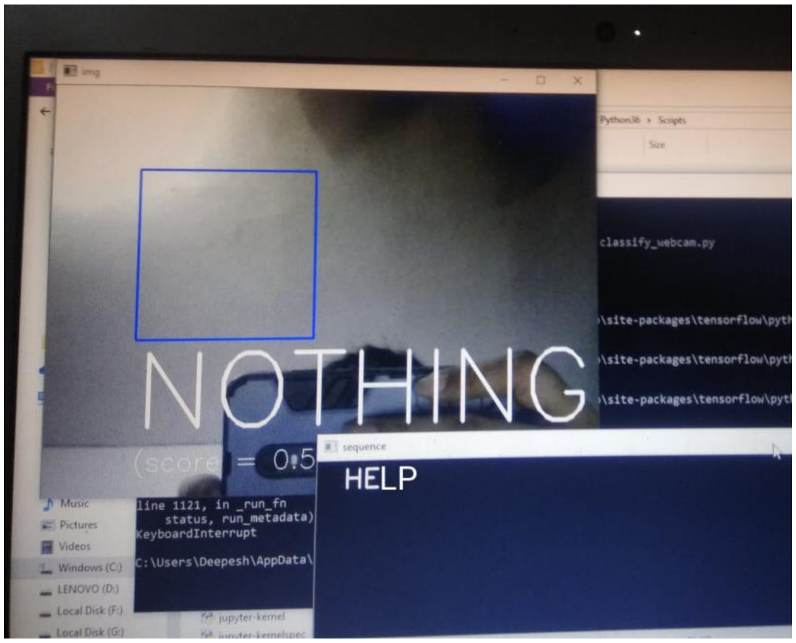
```
classify_webcam.py
\site-packages\tenso
\site-packages\tenso
\site-packages\tenso
line 1121, in _run_fn
    status, run_metadata)
KeyboardInterrupt
C:\Users\Deepesh\AppData\
```

FIGURE 5.7 DISPLAYING THE WORD “HE”



```
classify_webcam.py
\site-packages\tensorflow\python\client\
\site-packages\tensorflow\python\client\
\site-packages\tensorflow\python\client\
line 1121, in _run_fn
    status, run_metadata)
KeyboardInterrupt
C:\Users\Deepesh\AppData\
C:\Users\Deepesh\AppData\
```

FIGURE 5.8 DISPLAYING THE WORD “HE” WITH A “SPACE”



```
classify_webcam.py

\site-packages\tensorflow\pyth
\site-packages\tensorflow\pyth
\site-packages\tensorflow\pyth
```

```
line 1121, in _run_fn
    status, run_metadata)
KeyboardInterrupt

C:\Users\Deepesh\AppData\
```

FIGURE 5.8 MESSAGE “HELP” DISPLAYED USING GESTURES

Thus, we shown some examples to portray how our project works and displayed the messages in real-time according to the gestures made in front of the webcam in the rectangular box.

APPLICATIONS

Gesture Recognition Features:

- More accurate
- Impressive Stability
- Time saving to unlock a device

The major application areas of gesture recognition in the current scenario are:

- Automotive sector
- Sign language interpretation
- Consumer electronics sector
- Transit sector
- Gaming sector
- To unlock smart-phones
- In Defence Systems
- Home automation

Gesture recognition technology has been considered to be the highly successful technology as it saves time to unlock any device.

CONCLUSION

The most satisfying segment of this project has been the fact that the output of the whole procedure exactly resembles that of the expected outcome, resulting from the application of the particular theory which has been applied in this project. Moreover the extension of application that has been long associated with the following project i.e. adding audio effect to the fundamental aspect can serve as a great prospect in the future.

This experimental project can definitely prove as an ace in the hole for the speech impaired people if produced on large scale with improvised technology. They will be able to get an audio to a particular hand gesture. The extended provision of this fundamental work can also be sensor controlled consumer hardware.

REFERENCES

- Kang, Byeongkeun, SubarnaTripathi, and Truong Q. Nguyen. "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map." *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*. IEEE, 2015.
- scikit-learn.org
- deeplearningbooks.org : Convolutional Networks
- ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
- www.learnopencv.com