

PROFESSIONAL TRAINING REPORT

At

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Technology Degree in
Information Technology

By

ARUNAV SINGH (38110050)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119. TAMIL NADU**

MARCH 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344
www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ARUNAV SINGH (38110050)** who carried out the “**SIGN LANGUAGE RECOGNITION USING MACHINE LEARNING**” under our supervision from December 2021 to March 2022.

Internal Guide

Dr, S. Murugan (M.E., Ph.D.)

Head of the Department

Dr, Lakshmanan (M.E., Ph.D.)

Submitted for Viva Voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, ARUNAV SINGH (38110050) hereby declare that the Professional Training Report on “**SIGN LANGUAGE RECOGNITION USING MACHINE LEARNING**” done under the guidance of **Dr, S. Murugan (M.E., Ph.D)**, at Sathyabama Institute Of Science And Technology has submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** 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 ,(**Dr. Lakshmanan M.E., Ph.D.**) , Head of the Department of Information and Technology 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, S. Murugan (M.E., Ph.D.)**, for his 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 Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

Sign language is an overlooked concept even though there being a large social group which could benefit by it. Not everyone knows how to interpret a sign language when having a conversation with a deaf and dumb person. There is always a need to communicate using sign language. One finds it hard to communicate without a translator. To solve this, we need a common translator that is understood by common people and will help them to communicate without any barriers. Image classification and machine learning can be used to help computers recognize sign language, which could then be interpreted by other people. Pre-processing will be performed on images to get cleaned input. After that convolutional neural network (CNN) will be used to recognize sign language gestures. The main aim of this project is to eliminate the barrier between the deaf and dumb and the rest.

Table of Contents

1. INTRODUCTION.....	7
1.1 OUTLINE OF THE PROJECT	7
1.2 LITERATURE VIEW	7
1.3 SYSTEM IMPLEMENTATION	10
1.4 OBJECTIVES	10
2. AIM AND SCOPE OF THE PRESENT INVESTIGATION	11
2.1 SIGN LANGUAGE RECOGNITION	11
2.2 SERVERPORTAL.....	11
3. SYSTEM REQUIRMENT	12
3.1 REQUIRMENTS SPECIFICATION	12
3.1.1 HARDWARE REQUIRMENTS	12
3.1.2 SOFTWARE REQUIRMENTS	12
3.2 ABOUT THE SOFTWARE	12
3.2.1 SPYDER	12
4. EXPERIMENTAL OR MATERIALS AND METHODS ALGORITHM USED	13
4.1 METHODOLOGY	13
4.2 ALGORITHM	16
5. RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS.....	22
5.1 OVERVIEW OF THE PLATFORM	22
5.2 TESTING	23
6. SUMMARY AND CONCLUSIONS	23
6.1 CONCLUSION.....	23
6.2 FURTHER ENHANCEMENT	23
7. REFERENCES	23
8. APPENDIX	24
A) SAMPLE CODE: -	24
B) SCREEN SHOTS: -	27

1. INTRODUCTION

1.1 OUTLINE OF THE PROJECT

Indian sign language is a predominant sign language Since the only disability D&M people have is communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behaviour and visuals. Deaf and dumb(D&M) people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

American sign language is a predominant sign language Since the only disability D&M people have is communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behaviour and visuals. Deaf and dumb(D&M) people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

For interaction between normal people and D&M people a language barrier is created as sign language structure which is different from normal text. So, they depend on vision-based communication for interaction. If there is a common interface that converts the sign language to text, the gestures can be easily understood by the other people. So, research has been made for a vision-based interface system where D&M people can enjoy communication without really knowing each other's language. The aim is to develop a user-friendly human computer interface (HCI) where the computer understands the human sign language. There are various sign languages all over the world, namely American Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language, Japanese Sign Language and work has been done on other languages all around the world

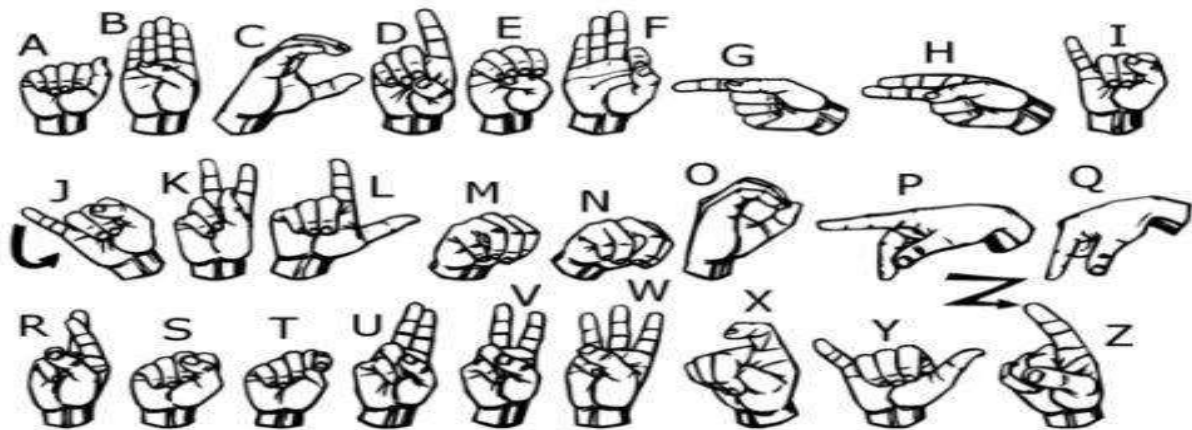
The importance of the very use of this method is increasing day by day as this gives the people with disabilities and opportunity to try their hands in different fields that require communication. With the help of this project, they can communicate with the majority in any industry, thus giving them an even playing field to cope up with. Here, we use the method of object identification followed by recognition to help us differentiate between the many symbols used in the sign.

1.2 LITERATURE VIEW

Sign language is a visual language and consists of 3 major components

Fingerspelling	Word level sign vocabulary	Non-manual features
Used to spell words	Used for the majority of communication	Facial expressions and tongue, mouth, and body position.

In our project we basically focus on producing a model which can recognise Fingerspelling based hand



gestures in order to form a complete word by combining each gesture. The gestures we aim to train are as given in the image below.

In the recent years there has been tremendous research done on the hand gesture recognition.

With the help of literature survey done we realized the basic steps in hand gesture recognition are:-

- i. Data-acquisition
- ii. Data-pre-processing
- iii. Gesture-classification

Data acquisition:

The different approaches to acquire data about the hand gesture can be done in the following ways:

1. Use of sensory devices

It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly.

2.Vision based approach

In vision-based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describe in artificial vision systems that are implemented in software and/or hardware.

The main challenge of vision-based hand detection is to cope with the large variability of human hand's appearance due to a huge number of hand movements, to different skin colour possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

Data pre-processing and Feature extraction for vision-based approach:

- In [1] the approach for hand detection combines threshold-based colour detection with background subtraction. We can use Ad boost face detector to differentiate between faces and hands as both involve similar skin-color.
- We can also extract necessary image which is to be trained by applying a filter called Gaussian blur. The filter can be easily applied using open computer vision also known as OpenCV and is described in[3].
- For extracting necessary image which is to be trained we can use instrumented gloves as mentioned in [4]. This helps reduce computation time for pre-processing and can give us more concise and accurate data compared to applying filters on data received from video extraction.
- We tried doing the hand segmentation of an image using colour segmentation techniques but as mentioned in the research paper skin colour and tone is highly dependent on the lighting conditions due to which output, we got for the segmentation we tried to do were no so great. Moreover we have a huge number of symbols to be trained for our project many of which look similar to each other like the gesture for symbol 'V' and digit '2', hence we decided that in order to produce better accuracies for our large number of symbols, rather than segmenting the hand out of a random background we keep background of hand a stable single colour so that we don't need to segment it on the basis of skin colour. This would help us to get better results.

Gesture Classification:

- In [1] Hidden Markov Models (HMM) is used for the classification of the gestures. This model deals with dynamic aspects of gestures. Gestures are extracted from a sequence of video images by tracking the skin-color blobs corresponding to the hand into a body– face space centered on the face of the user. The goal is to

recognize two classes of gestures: deictic and symbolic. An image is filtered using a fast look-up indexing table. After filtering, skin colour pixels are gathered into blobs. Blobs are statistical objects based on the location (x, y) and the colorimetry (Y,U,V) of the skin colour pixels in order to determine homogeneous areas.

- In Naïve Bayes Classifier is used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation. Thus, unlike many other recognition methods, this method is not dependent on skin colour. The gestures are extracted from each frame of the video, with a static background. The first step is to segment and label the objects of interest and to extract geometric invariants from them. Next step is the classification of gestures by using a K nearest neighbour algorithm aided with distance weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes “classifier.
- According to paper on “Human Hand Gesture Recognition Using a Convolution Neural Network” by Hsien-I Lin , Ming-Hsiang Hsu, and Wei-Kai Chen graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan, they construct asking model to extract the handout of an image and then apply binary threshold to the whole image. After obtaining the threshold image they calibrate it about the principal axis in order to centre the image about it. They input this image to a convolutional neural network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using their model they produce an accuracy of around 95% for those 7 gestures.

1.3 SYSTEM IMPLEMENTATION

For interaction between normal people and D&M people a language barrier is created as sign language structure which is different from normal text. So, they depend on vision-based communication for interaction.

If there is a common interface that converts the sign language to text the gestures can be easily understood by the other people. So, research has been made for a vision-based interface system where D&M people can enjoy communication without really knowing each other's language.

If there is a common interface that converts the sign language to text the gestures can be easily understood by the other people. So, research has been made for a vision-based interface system where D&M people can enjoy communication without really knowing each other's language.

1.4 OBJECTIVES

- 1) Decision making power is provided by this system.
- 2) Accurate result can be obtained.
- 3) This system makes selection process more effective.

- 4) To increase efficiency proposed system is dependent on classification method.
- 5) Proposed system is used to reduce confusion at the time of processing data average.

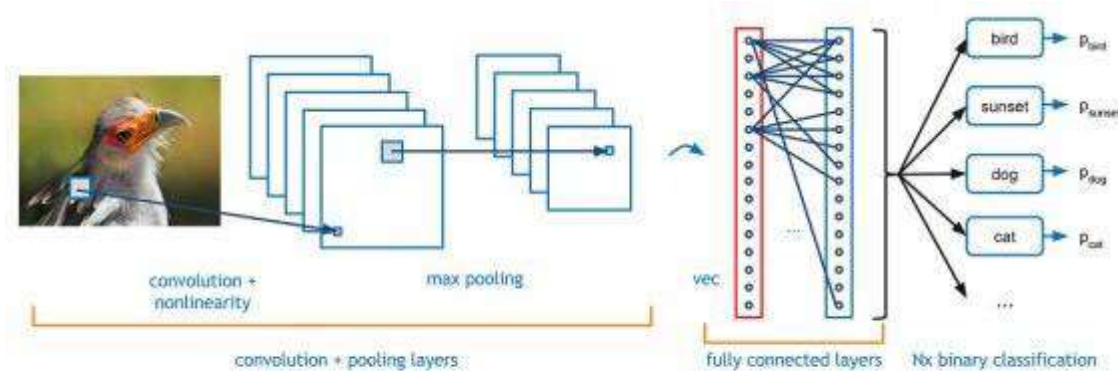


Figure 5.2: Convolution neural networks

2. AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 SIGN LANGUAGE RECOGNITION

Sign language recognition and translation is a research area with high potential impact. There are over 300 sign languages used around the world, and 70 million deaf people are using them. Sign language processing would break down all the barriers for sign language users. This can be very helpful for the deaf and dumb people in communicating with others as knowing sign language is not something that is common to all, moreover, this can be extended to creating automatic editors, where the person can easily write by just their hand gestures. The types of data available and the relative merits are explored allowing examination of the features which can be extracted. Classifying the manual aspects of sign (similar to gestures) is then discussed from a tracking and non-tracking viewpoint before summarizing some of the approaches to the non-manual aspects of sign languages. Methods for combining the sign classification results into full SLR are given showing the progression towards speech recognition techniques and the further adaptations required for the sign specific case. Finally, the current frontiers are discussed, and the recent research presented. This covers the task of continuous sign recognition, the work towards true signer independence, how to effectively combine the different modalities of sign, making use of the current linguistic research and adapting to larger more noisy data sets.

2.2 SERVERPORTAL

Gesture recognition information system is a project of the Department of Intelligent Systems in Control and Automation, which includes the development of subsystem for recognizing various types of communication between people with hearing and voice disabilities, fingerprints, gestures using the user's hands and his emotions. The recognition system is a combination of software and hardware tools that allow remotely using the recognition system.

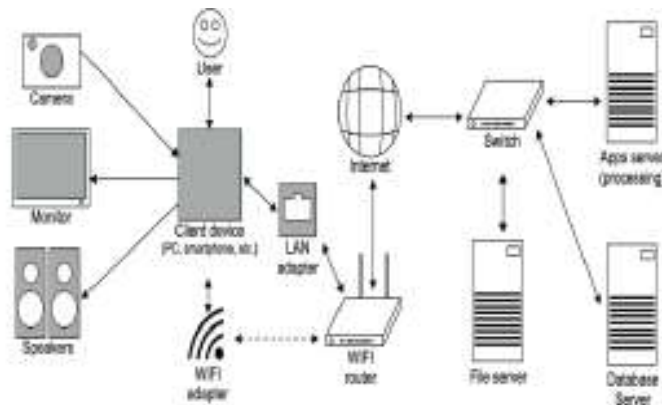


Figure:2

3. SYSTEM REQUIREMENT

3.1 REQUIREMENTS SPECIFICATION

3.1.1 HARDWARE REQUIREMENTS

Processor – Intel i5

Ram – 4GB

Hard Disk Drive – 40GB

Monitor – LCD

3.1.2 SOFTWARE REQUIREMENTS

Operating System – Windows 10

Software – SPYDER

3.2 ABOUT THE SOFTWARE

3.2.1 SPYDER

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features.

Spyder is an open-source cross-platform IDE. The PYTHON Spyder IDE is written completely in Python. It is designed by scientists and is exclusively for scientists, data-analyst, and engineers. It is also known as the Scientific Python Development IDE and has a huge set of remarkable features,

WHY SPYDER?

- i) It's an open-source cross-platform IDE for data science.
- ii) It integrates the essentials libraries for data science, such as NumPy, SciPy, Matplotlib and I-Python, besides that, it can be extended with plugins.

- iii) Spyder contains features like a text editor with syntax highlighting, code completion and variable exploring, which you can edit its values using a Graphical User Interface (GUI).

4. EXPERIMENTAL OR MATERIALS AND METHODS ALGORITHM USED

4.1 METHODOLOGY

Data Pre-processing Using Label-Img.
Object Detection using TensorFlow.
Real Time Detection.

A. DATA ACQUISITION

The different approaches to acquire data about the hand gesture can be done in the following ways:

1.USE OF SENSORY DEVICES

It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly.

2.VISION BASED APPROACH

In vision-based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing 7 artificial vision systems that are implemented in software and/or hardware. The main challenge of vision-based hand detection is to cope with the large variability of human hand's appearance due to a huge number of hand movements, to different skin-color possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

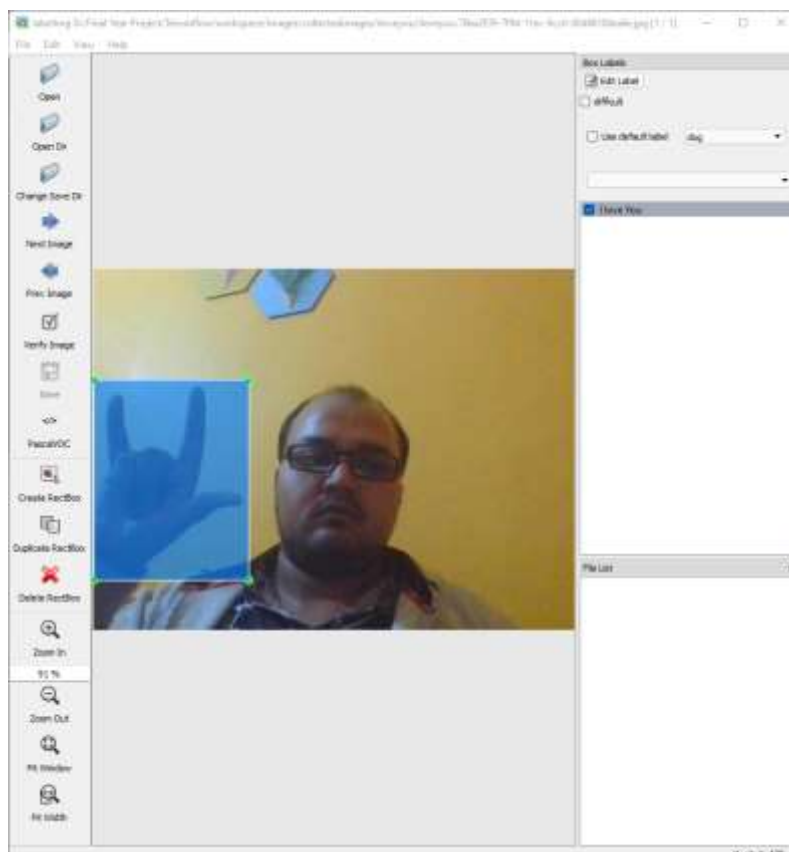
However, due to the rarity of the chosen project, there were a little to no datasets available online. As a result, we had to make our own datasets to help train the code to give us the best results possible. It was made possible with the use of Open computer vision also known as the Open CV library to obtain the customized datasets we use in this program.



Fig 4.1. Image captured using figure.

B. DATA PREPROCESSING USING LABELING

After you train the algorithm to help us identify sign language, we need to make sure the algorithm is able to ignore the backgrounds and focus solely on the given dataset. Labelling is a graphical image annotation tool which is used for bounding boxes in images.



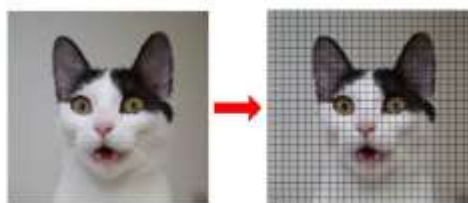
C. OBJECT DETECTION

Object detection is a method in which a program or a software can trace and detect the object from a given photo or any other visual data. The special attribute about object detection is that it identifies the class of object (person, table, chair, etc.) and their location-specific coordinates in the given image.

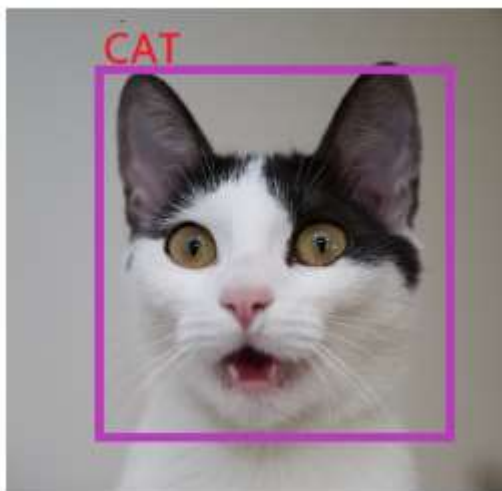
A bounding box around the object is used to indicate its location. The drawn bounding box may or may not pinpoint the exact location of the given piece. The potential to spot the provided object inside a photo explain the performance of the algorithm that has been used for detection. Sign language detection is one of the examples of object detection.

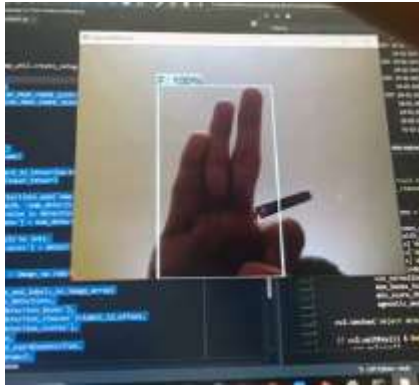
Object detection steps: -

Generation of small portion in the input as shown in the image below. As shown below in the figure large set of boxes are spanning the full image.



Feature extraction is the next step that will be done for each rectangular area to check whether or whether not the rectangle area contains a valid object.





4.2 ALGORITHM

Artificial Neural Networks:

Artificial Neural Network is a connection of neurons, replicating the structure of human brain. Each connection of neuron transfers information to another neuron. Inputs are fed into first layer of neurons which processes it and transfers to another layer of neurons called as hidden layers. After processing of information through multiple layers of hidden layers, information is passed to final output layer. 10 There are capable of learning, and they have to be trained.

Artificial neural networks are built like the human brain, with neuron nodes interconnected like a web. The human brain has hundreds of billions of cells called neurons. Each neuron is made up of a cell body that is responsible for processing information by carrying information towards (inputs) and away (outputs) from the brain.

An ANN has hundreds, or thousands of artificial neurons called processing units, which are interconnected by nodes. These processing units are made up of input and output units. The input units receive various forms and structures of information based on an internal weighting system, and the neural network attempts to learn about the information presented to produce one output report. Just like humans need rules and guidelines to come up with a result or output, ANNs also use a set of learning rules called backpropagation, an abbreviation for backward propagation of error, to perfect their output results.

An ANN initially goes through a training phase where it learns to recognize patterns in data, whether visually, aurally, or textually. During this supervised phase, the network compares its actual output produced with what it was meant to produce—the desired output. The difference between both outcomes is adjusted using backpropagation. This means that the network works backward, going from the output unit to the input units to adjust the weight of its connections between the units until the difference between the actual and desired outcome produces the lowest possible error.

During the training and supervisory stage, the ANN is taught what to look for and what its output should be, using yes/no question types with binary numbers. For example, a bank that wants to detect credit card fraud on time may have four input units fed with these questions: (1) Is the transaction in a different country from the user's resident country? (2) Is the website the card is being used at affiliated with companies or countries on the bank's watch list? (3) Is the transaction amount larger

than \$2,000? (4) Is the name on the transaction bill the same as the name of the cardholder?

The bank wants the "fraud detected" responses to be Yes, No, which in binary format would be 1 1 1 0. If the network's actual output is 1 0 1 0, it adjusts its results until it delivers an output that coincides with 1 1 1 0. After training, the computer system can alert the bank of pending fraudulent transactions, saving the bank lots of money.

There are different learning strategies:

- i. Unsupervised Learning.
- ii. Supervised Learning.
- iii. Reinforcement Learning.

Convolution Neural Network:

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most applied to analysing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that shift over input features and provide translation equivariant responses. Counter-intuitively, most convolutional neural networks are only equivariant, as opposed to invariant, to translation. They have applications in image and video recognition, recommender systems, classification, image segmentation, medical image analysis, natural language processing, brain computer-interfaces, and financial time-series.

CNNs use relatively little pre-processing compared to other image classification algorithm. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hard engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks makes them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their

filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

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.

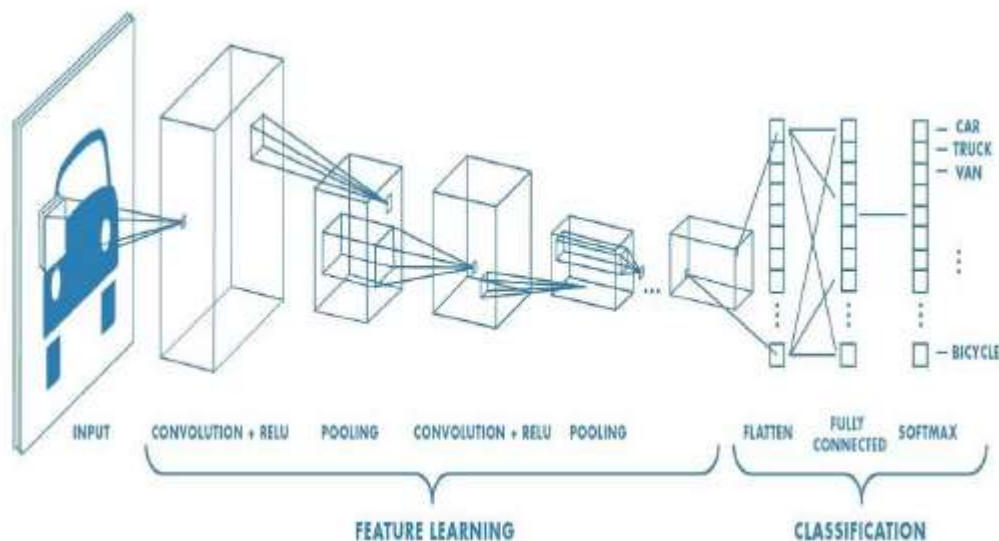


Fig 4.2

1)Convolution Layer: In convolution layer we take a small window size [typically of length 5×5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we slid the window by stride size [typically 1] and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour

2)Pooling Layer: We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two types of pooling:

- a) Max Pooling: In max pooling we take a window size [for example window of size 2×2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get a activation matrix half of its original Size.
- b) Average Pooling: In average pooling we take average of all values in a window.

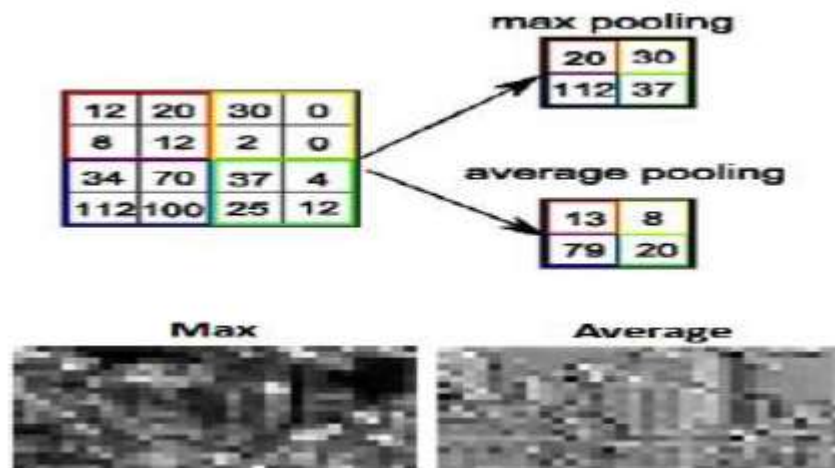


Fig 4.3

3) Fully Connected Layer: In convolution layer neurons are connected only to a local region, while in a fully connected region, we connect all the inputs to neurons.

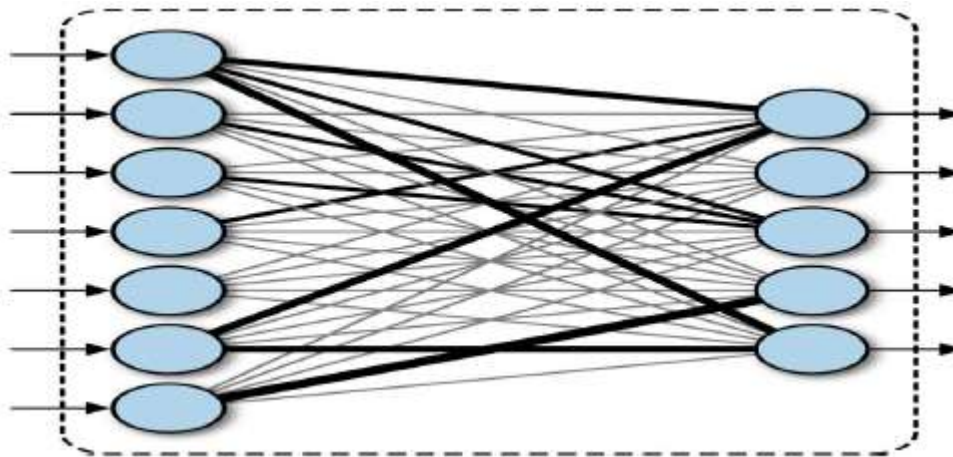


Fig 4.4

4) Final Output Layer: After getting values from fully connected layer, we connect them to final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

The approach which we used for this project is:

Our approach uses two layers of algorithm to predict the final symbol of the user.

Algorithm Layer 1:

- i. Apply gaussian blur filter and threshold to the frame taken with OpenCV to get the processed image after feature extraction.
- ii. This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and taken into consideration for forming the word
- iii. Space between the words is considered using the blank symbol.

Algorithm Layer 2:

- i. We detect various sets of symbols which show similar results on getting detected.
- ii. We then classify between those sets using classifiers made for those sets only.

Layer 1:

CNN Model:

- i. 1st Convolution Layer: The input picture has resolution of 128x128 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126X126 pixel image, one for each Filter-weights.
- ii. 1st Pooling Layer: The pictures are down sampled using max pooling of 2x2 i.e., we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.
- iii. 2nd Convolution Layer: Now, these 63 x 63 from the output of the first pooling layer is served as an input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 60 x 60-pixel image.
- iv. 2nd Pooling Layer: The resulting images are down sampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.
- v. 1st Densely Connected Layer: Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of $30 \times 30 \times 32 = 28800$ values. The input to this layer is an array of 28800 values. The output of these layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.
- vi. 2nd Densely Connected Layer: Now the output from the 1st Densely Connected Layer are used as an input to a fully connected layer with 96 neurons.
- vii. Final layer: The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

Activation Function:

We have used ReLu (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons). ReLu calculates $\max(x, 0)$ for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

Pooling Layer:

We apply Max pooling to the input image with a pool size of (2, 2) with ReLu activation function. This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.

Dropout Layers:

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples. This layer "drops out" a random set of activations in that layer by setting them to zero. The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out [5].

Optimizer: We have used Adam optimizer for updating the model in response to the output of the loss function. Adam combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp)

Layer 2:

We are using two layers of algorithms to verify and predict symbols which are more like each other so that we can get us close as we can get to detect the symbol shown. In our testing we found that following symbols were not showing properly and were giving other symbols also.

Finger spelling sentence formation implementation:

- i. Whenever the count of a letter detected exceeds a specific value and no other letter is close to it by a threshold, we print the letter and add it to the current string (In our code we kept the value as 50 and difference threshold as 20).
- ii. Otherwise, we clear the current dictionary which has the count of detections of present symbol to avoid the probability of a wrong letter getting predicted.
- iii. Whenever the count of a blank (plain background) detected exceeds a specific value and if the current buffer is empty no spaces are detected.
- iv. In other case it predicts the end of word by printing a space and the current gets appended to the sentence below.

Auto-correct Feature:

A python library Hunspell_suggest is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in which the user can select a word to append it to the current sentence. This helps in reducing mistakes committed in spellings and assists in predicting complex words.

Training and Testing: We convert our input images (RGB) into grayscale and apply gaussian blur to remove unnecessary noise. We apply adaptive threshold to extract our hand from the background and resize our images to 128 x 128. We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above. The prediction layer estimates how likely the image will fall under one of the classes. So, the output is normalized between 0 and 1 and such that the sum of each value in each class sums to 1. We have achieved this using SoftMax function. At first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labelled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labelled value and is zero exactly when it is equal to the labelled value. Therefore, we optimized the cross-entropy by minimizing it as close to zero. To do this in our network layer we adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy. As we have found out the cross-entropy function, we have optimized it using Gradient Descent in fact with the best gradient descent optimizer is called Adam Optimizer.

5. RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

5.1 OVERVIEW OF THE PLATFORM

We have achieved an accuracy of 95.8% in our model using only layer 1 of our algorithm and using the combination of layer 1 and layer 2 we achieve an accuracy of 98.0%, which is a better accuracy than most of the current research papers on American sign language. Most of the research papers focus on using devices like Kinect for hand detection. In [7] they build a recognition system for Flemish sign language using convolutional neural networks and Kinect and achieve an error rate of 2.5%. In [8] a recognition model is built using hidden Markov model classifier and a vocabulary of 30 words and they achieve an error rate of 10.90%. In [9] they achieve an average accuracy of 86% for 41 static gestures in Japanese sign language. Using depth sensors map [10] achieved an accuracy of 99.99% for observed signers and 83.58% and 85.49% for new signers. They also used CNN for their recognition system. One thing should be noted that our model doesn't use any background subtraction algorithm while some of the models present above do that. So, once we try to implement background subtraction in our project the accuracies may vary. On the other hand, most of the above projects use Kinect devices but our main aim was to create a project which can be used with readily available resources. A sensor like Kinect not only isn't readily available but also is expensive for most of the audience to

buy and our model uses a normal webcam of the laptop hence it is great plus point. Below are the confusion matrices for our results.

5.2 TESTING

We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above. The prediction layer estimates how likely the image will fall under one of the classes. So, the output is normalized between 0 and 1 and such that the sum of each value in each class sums to 1. We have achieved this using soft-max function first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labelled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labelled value and is zero exactly when it is equal to the labelled value. Therefore, we optimized the cross-entropy by minimizing it as close to zero. To do this in our network layer we adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy. As we have found out the cross-entropy function, we have optimized it using Gradient Descent in fact with the best gradient descent optimizer is called Adam Optimizer.

6. SUMMARY AND CONCLUSIONS

6.1 CONCLUSION

In this report, a functional real time vision based Indian sign language recognition for D&M people have been developed for ISL alphabets. We achieved final accuracy of 72.0% on our dataset. We can improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more like each other. This way we can detect almost all the symbols if they are shown properly, there is no noise in the background and lighting is adequate.

6.2 FURTHER ENHANCEMENT

We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. We are also thinking of improving the pre-processing to predict gestures in low light conditions with a higher accuracy.

7. REFERENCES

- [1] Stein, D., Forster, J., Zelle, U., Dreuw, P., and Ney, H.: A Study of German Sign Language Weather Forecast Corpora. Corpora and Sign Language Technologies, pp. 225–230, Workshop on Sign Language Representation and Processing, 2010.
- [2] Roussos, A., Theodorakis, S., Pitsikalis, P., Maragos, P.: Continuous sign language recognition using hand monitoring and affine shape presence hand shape sub-units. In: 11th European Conference on Computer Vision (ECCV) Workshop on Sign, Gesture, and Activity, 2010.

- [3] Website for the Dictasign project (2010). Visit <http://www.signlang.uni-hamburg.de/dictasign/portal> for more information.
- [4] Hadfield, S., and Bowden, R.: Depthbased generalised pose estimation. "Sign, Gesture, Activity," in Proc. ECCVInt. Workshop. Crete, Heraklion, 2010
- [5] S. Koelstra, M. Pantic, and I. Patras: A dynamic texture-based approach to facial action recognition and temporal models. Vol. 32, pp. 1940-1954, IEEE Transactions on Pattern Analysis and Machine Intelligence, November 2010.
- [6] Thai Sign Language Translation System Using Upright SpeedUp Robust Feature and CMeans Clustering, in WCCI 2012 IEEE World Congress on Computational Intelligence, pp. 10-15, June 2012. Phonkrit Chanda, Sansanee Auephanwiriyakul, and Nipon Theeraumpon, Thai Sign Language Translation System Using Upright Speed-Up Robust Feature and C-Means Clustering, in WCCI
- [7] Sara Bilal¹, Rini Akmeliawati², Momoh Jimoh El Salami, and Amir A. Shafie, "Visionbased Hand Posture Detection and Recognition for Sign Language Research," International Conference on Mechatronics (ICOM), 2011, pp. 17-19.
- [8] Pritee Khanna and Lalit Kane IEEE Proceedings of the 4th International Conference on Intelligent Human Computer Interaction, Dec. 2012. "Towards Establishing a Mute Communication: An Indian Sign Language Perspective," IEEE Proceedings of the 4th International Conference on Intelligent Human Computer Interaction, Dec. 2012.
- [9] Joyeeta Singha, Karen Das, International Journal of Advanced Computer Science and Applications, vol. 4, No. 2, pp. 188-194, 2013. Indian Sign Language Recognition Using Eigen Value Weighted Euclidean Distance Dependent Classification Technique, International Journal of Advanced Computer Science and Applications, vol. 4, No. 2, pp. 188-194, 2013.

8. APPENDIX

A) SAMPLE CODE: -

```
IMAGE PREPROCESSING-
import numpy as np
import cv2
import time
import uuid
import os
```

```
IMAGES_PATH='Tensorflow/workspace/images/collectedimages'
```

```
labels=['A','B','C','D','E','F']
```

```
no_of_imgs=10
```

```
for label in labels:
```

```
os.makedirs('Tensorflow/workspace/images/collectedimages\\'+ label)
```

```

cap=cv2.VideoCapture(0)
print('Images for {}'.format(label))
time.sleep(5)
for i in range(no_of_imgs):
    ret, frame=cap.read()
    if not ret:
        print("failes")
        break

```

```

imgname=os.path.join('Tensorflow/workspace/images/collectedimages',label,label+'.'+
'{}.jpg'.format(str(uuid.uuid1())))
cv2.imwrite(imgname,frame)
cv2.imshow('frame',frame)
time.sleep(2)
if cv2.waitKey(0) & 0xFF == ord('q'):

break
cap.release()
cap.release()

```

MAIN PROGRAM-

```

import os
WORKSPACE_PATH='Tensorflow/workspace'
SCRIPTS_PATH='Tensorflow/scripts'
APIMODEL_PATH='Tensorflow/models'
ANNOTATION_PATH=WORKSPACE_PATH+'/annotations'
IMAGE_PATH=WORKSPACE_PATH+'/images'
MODEL_PATH=WORKSPACE_PATH+'/models'
PRETRAINED_MODEL_PATH=WORKSPACE_PATH+'/pre-trained-models'
CONFIG_PATH=MODEL_PATH+'/my_ssd_mobnet/pipeline.config'
CHECKPOINT_PATH=MODEL_PATH+'/my_ssd_mobnet'

```

```

labels=[
{'name':'A','id':1},
{'name':'B','id':2},
{'name':'C','id':3},
{'name':'D','id':4},
{'name':'E','id':5},
{'name':'F','id':6},

]
with open(ANNOTATION_PATH+'\label_map.pbtxt','w') as f:
for label in labels:
f.write('item { \n'
f.write('\tname:{}'.format(label['name']))
f.write('\tid:{}'.format(label['id']))
f.write('}\n')
!python {SCRIPTS_PATH+'/generate_tfrecord.py'} -x {IMAGE_PATH + '/train'} -l
{ANNOTATION_PATH + '/label_map.pbtxt'} -o {ANNOTATION_PATH + '/train.record'}
!python {SCRIPTS_PATH+'/generate_tfrecord.py'} -x {IMAGE_PATH + '/test'} -l
{ANNOTATION_PATH + '/label_map.pbtxt'} -o {ANNOTATION_PATH + '/test.record'}

```

```

CUSTOM_MODEL_NAME='my_ssd_mobnet'
os.makedirs('tensorflow/workspace/models\\'+ CUSTOM_MODEL_NAME)

```

```

!cp {PRETRAINED_MODEL_PATH+'/'+'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config'} {MODEL_PATH+'/'+'CUSTOM_MODEL_NAME'}
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
CONFIG_PATH=MODEL_PATH+'/'+'CUSTOM_MODEL_NAME+'/'pipeline.config'
config=config_util.get_configs_from_pipeline_file(CONFIG_PATH)
pipeline_config=pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(CONFIG_PATH,"r") as f:
    proto_str=f.read()
    text_format.Merge(proto_str, pipeline_config)
    pipeline_config.model.ssd.num_classes=6
    pipeline_config.train_config.batch_size=4
    pipeline_config.train_config.fine_tune_checkpoint=PRETRAINED_MODEL_PATH+'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0'
    pipeline_config.train_config.fine_tune_checkpoint_type="detection"
    pipeline_config.train_input_reader.label_map_path=ANNOTATION_PATH+'/'+'label_map.pbtxt'
    pipeline_config.train_input_reader.tf_record_input_reader.input_path[:]=[ANNOTATION_PATH+'/'+'train.record']
    pipeline_config.eval_input_reader[0].label_map_path=ANNOTATION_PATH+'/'+'label_map.pbtxt'
    pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:]=[ANNOTATION_PATH+'/'+'test.record']
    config_text=text_format.MessageToString(pipeline_config)
    with tf.io.gfile.GFile(CONFIG_PATH, "wb") as f:
        f.write(config_text)
#training

print("""python {}/research/object_detection/model_main_tf2.py --model_dir={} --
pipeline_config_path={} --num_train_steps=8000""".format(API_MODEL_PATH,
MODEL_PATH,CUSTOM_MODEL_NAME,MODEL_PATH,CUSTOM_MODEL_NAME))
import os
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
detection_model =
model_builder.build(model_config=configs['model'],is_training=False)
ckpt=tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(CHECKPOINT_PATH,'ckpt-9')).expect_partial()
@tf.function
def detect_fn(image):
    image, shapes=detection_model.preprocess(image)
    prediction_dict=detection_model.predict(image, shapes)
    detections=detection_model.postprocess(prediction_dict, shapes)
    return detections
import cv2
import numpy as np
category_index =
label_map_util.create_category_index_from_labelmap(ANNOTATION_PATH+'/'+'label_map.pbtxt')
# Setup capture

```

```

cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while True:
    ret, frame = cap.read()
    image_np = np.array(frame)
    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)
    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections
    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
    label_id_offset = 1
    image_np_with_detections = image_np.copy()
    viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'],
        detections['detection_classes']+label_id_offset,
        detections['detection_scores'],
        category_index,
        use_normalized_coordinates=True,
        max_boxes_to_draw=5,
        min_score_thresh=.5,
        agnostic_mode=False)
    cv2.imshow('object detection', cv2.resize(image_np_with_detections, (800, 600)))
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        break
    cap.release()

```

B) SCREEN SHOTS: -

