TOPIC OF THE PROJECT

MAJOR PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INSTRUMENTATION AND CONTROL ENGINEERING

by

Kunal Maggo	**Samyak Jain**	**Mayank Negi**
Enrollment No: 01611503018	**Enrollment No: 02511503018**	**Enrollment No: 01711503018**

Guided by

Dr. Arati Kane
HOD & Associate Professor

DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING
BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)
NEW DELHI – 110063
MAY 2022

CANDIDATE DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Major project Report entitled **"SIGN LANGUAGE TO TEXT/SPEECH"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted to the **Department of Instrumentation and Control Engineering** of **BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **February 2022 to May 2022** under the guidance of **Dr. Arati Kane, HOD & Associate Professor.**

The matter presented in the B. Tech Major Project Report has not been submitted by me for the award of any other degree at this or any other Institute.

Kunal Maggo	**Samyak Jain**	**Mayank Negi**
Enrollment No: 01611503018	**Enrollment No: 02511503018**	**Enrollment No: 01711503018**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. He/She/They are permitted to appear in the External Major Project Examination

Dr. Arati Kane
HOD & Associate Professor

Dr. Aarti Kane
Head, ICE

The B. Tech Major Project Viva-Voce Examination of **Kunal Maggo(01611503018), Samyak Jain(02511503018), Mayank Negi(01711503018)** has been held on

……………………………….

(Signature of External Examiner)

Dr. Aarti Kane
Project Coordinator

ABSTRACT

Conversing to a person with hearing disability is always a major challenge. Sign language has indelibly become the ultimate panacea and is a very powerful tool for individuals with hearing and speech disability to communicate their feelings and opinions to the world. It makes the integration process between them and others smooth and less complex. However, the invention of sign language alone, is not enough. There are many strings attached to this boon. The sign gestures often get mixed and confused for someone who has never learnt it or knows it in a different language. However, this communication gap which has existed for years can now be narrowed with the introduction of various techniques to automate the detection of sign gestures. In this paper, we introduce a Sign Language recognition using American Sign Language.

In this study, the user must be able to capture images of the hand gesture using web camera and the system shall predict and display the name of the captured image. We used the Yolo Object Detection algorithm to detect the word level gestures.

KeyWords: Sign LanguageRecognition1, Convolution Neural Network2, Image Processing3, Yolo Object Detection4, Hand Gesture Recogniton5.

ACKNOWLEDGEMENT

We express our deep gratitude to the **Dr. Arati Kane**, **HOD & Associate Professor**, Department of Instrumentation And Control Engineering for her valuable guidance and suggestion throughout our project work. We are thankful to **Dr. Arati Kane**, project coordinator**,** for her valuable guidance.

We would like to extend our sincere thanks to **the Head of the Department, Dr. Arati Kane** for her time to time suggestions to complete our project work. We are also thankful to **Prof. Dharmender Saini, Principal** for providing us the facilities to carry out our project work.

Kunal Maggo	**Samyak Jain**	**Mayank Negi**
Enrollment No: 01611503018	**Enrollment No: 02511503018**	**Enrollment No: 01711503018**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

Speech impaired people use hand signs and gestures to communicate. Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

1.1 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

• Importing the image via image acquisition tools.

• Analysing and manipulating the image.

• Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre- processing, enhancement, and display, information extraction.

Digital image processing:

Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing.

society, rely heavily on digital image processing.

Digital image processing consists of the manipulation of those finite precision numbers. The processing of digital images can be divided into several classes: image enhancement, image restoration, image analysis, and image compression. In image enhancement, an image is manipulated, mostly by heuristic techniques, so that a human viewer can extract useful information from it.

Digital image processing is to process images by computer. Digital image processing can be defined as subjecting a numerical representation of an object to a series of operations in order to obtain a desired result. Digital image processing consists of the conversion of a physical image into a corresponding digital image and the extraction of significant information from the digital image by applying various algorithms.

Pattern recognition: On the basis of image processing, it is necessary to separate objects from images by pattern recognition technology, then to identify and classify these objects through technologies provided by statistical decision theory. Under the conditions that an image includes several objects, the pattern recognition consists of three phases, as shown in Fig.

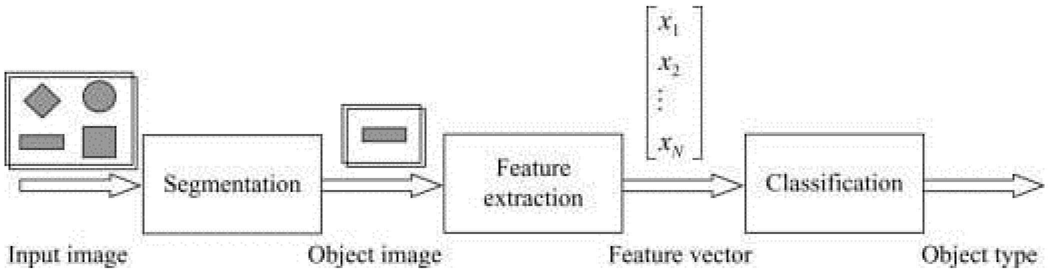


Fig1.1: Phases of pattern recognition

The first phase includes the image segmentation and object separation. In this phase, different objects are detected and separate from other background. The second phase is the feature extraction. In this phase, objects are measured. The measuring feature is to quantitatively estimate some important features of objects, and a group of the features are combined to make up a feature vector during feature extraction. The third phase is classification. In this phase, the output is just a decision to determine which category every object belongs to. Therefore, for pattern recognition, what input are images and what output are object types and structural analysis of images. The structural analysis is a description of images in order to correctly understand and judge for the important information of images.

1.2 SIGN LANGUAGE

It is a language that includes gestures made with the hands and other body parts, including facial expressions and postures of the body.It used primarily by people who are deaf and dumb. There are many different sign languages as, British, Indian and American sign languages. British sign language (BSL) is not easily intelligible to users of American sign Language (ASL) and vice versa .

A functioning signing recognition system could provide a chance for the inattentive communicate with non-signing people without the necessity for an interpreter. It might be wont to generate speech or text making the deaf more independent. Unfortunately there has not been any system with these capabilities thus far. during this project our aim is to develop a system which may classify signing accurately.

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing, and is used by many hearing people as well.

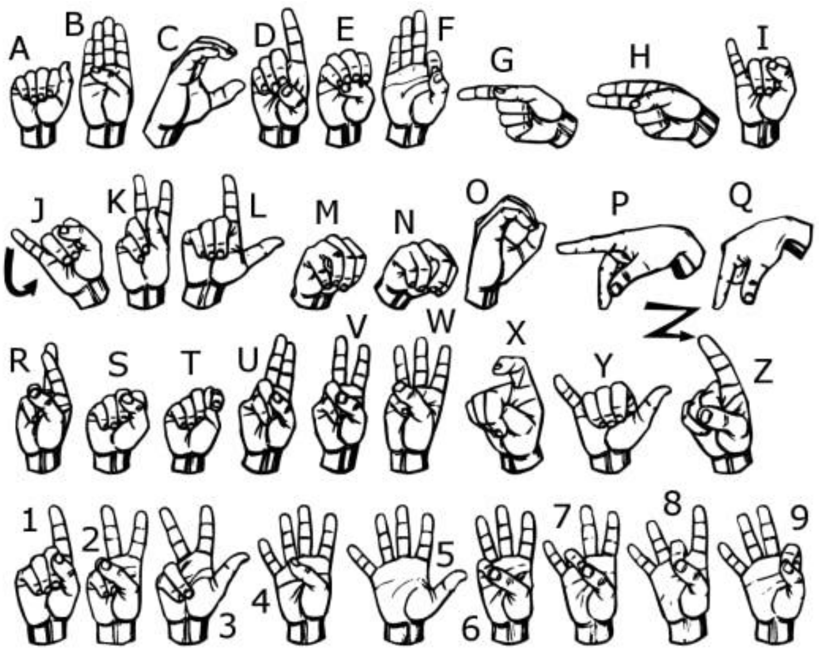


Fig 1.2 Sign Language Gestures

1.3 SIGN LANGUAGE AND HAND GESTURE RECOGNITION

The process of converting the signs and gestures shown by the user into text is called sign language recognition. It bridges the communication gap between people who cannot speak and the general public. Image processing algorithms along with neural networks is used to map the gesture to appropriate text in the training data and hence raw images/videos are converted into respective text that can be read and understood.

Dumb people are usually deprived of normal communication with other people in the society. It has been observed that they find it really difficult at times to interact with normal people with their gestures, as only a very few of those are recognized by most people. Since people with hearing impairment or deaf people cannot talk like normal people so they have to depend on some sort of visual communication in most of the time. Sign Language is the primary means of communication in the deaf and dumb community. As like any other language it has also got grammar and vocabulary but uses visual modality for exchanging information. The problem arises when dumb or deaf people try to express themselves to other people with the help of these sign language grammars. This is because normal people are usually unaware of these grammars. As a result it has been seen that communication of a dumb person are only limited within his/her family or the deaf community. The importance of sign language is emphasized by the growing public approval and funds for international project. At this age of Technology the demand for a computer based system is highly demanding for the dumb community. However, researchers have been attacking the problem for quite some time now and the results are showing some promise. Interesting technologies are being developed for speech recognition but no real commercial product for sign recognition is actually there in the current market. The idea is to make computers to understand human language and develop a user friendly human computer interfaces (HCI). Making a computer understand speech, facial expressions and human gestures are some steps towards it. Gestures are the non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers. The project aims to determine human gestures by creating an HCI. Coding of these gestures into machine language demands a complex programming algorithm. In our project we are focusing on Image Processing and Template matching for better output generation.

1.4 MOTIVATION

The 2011 Indian census cites roughly 1.3 million people with “hearing impairment”. In contrast to that numbers from India’s National Association of the Deaf estimates that 18 million people –roughly 1 per cent of Indian population are deaf. These statistics formed the motivation for our project. As these speech impairment and deaf people need a proper channel to communicate with normal people there is a need for a system. Not all normal people can understand sign language of impaired people. Our project hence is aimed at converting the sign language gestures into text that is readable for normal people.

1.5 PROBLEM STATEMENT

Speech impaired people use hand signs and gestures to communicate.

Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

2. LITERATURE SURVEY

2.1 INTRODUCTION:

The domain analysis that we have done for the project mainly involved understanding the neural networks

2.1.1 TensorFlow:

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.

Features: TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell Julia, MATLAB, R, Scala, Rust, OCaml, and Crystal. "New language support should be built on top of the C API. However, not all functionality is available in C yet." Some more functionality is provided by the Python API.

Application: Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as Deep Dream.

2.1.2 OpenCv:

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.[1] Originally developed by Intel, it was later supported by Willow Garage then Itseez(which was later acquired by Intel[2]). The library is cross-platform and free for use under the open-source BSD license.

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Ego motion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition

Stereopsis stereo vision: depth perception from 2 cameras

- Structure from motion (SFM).
- Motion tracking
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbour algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

AForge.NET, a computer vision library for the Common Language Runtime (.NET Framework and Mono).

ROS (Robot Operating System). OpenCV is used as the primary vision package in ROS.

VXL, an alternative library written in C++.

Integrating Vision Toolkit (IVT), a fast and easy-to-use C++ library with an optional interface to OpenCV.

CVIP tools, a complete GUI-based computer-vision and image-processing software environment, with C function libraries, a COM-based DLL, along with two utility programs for algorithm development and batch processing.

OpenNN, an open-source neural networks library written in C++. List of free and open source software packages

- OpenCV Functionality
- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu) Image-Processing:

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”.

Digital-Image:

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps : Importing the image

Analysing and manipulating the image

Output in which result can be altered image or report that is based on image analysis Applications of Computer Vision:

Here we have listed down some of major domains where Computer Vision is heavily used.

- Robotics Application
- Localization − Determine robot location automatically
- Navigation
- Obstacles avoidance
- Assembly (peg-in-hole, welding, painting)
- Manipulation (e.g. PUMA robot manipulator)
- Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people
- Medicine Application
- Classification and detection (e.g. lesion or cells classification and tumor detection)
- 2D/3D segmentation
- 3D human organ reconstruction (MRI or ultrasound)
- Vision-guided robotics surgery
- Industrial Automation Application
- Industrial inspection (defect detection)
- Assembly
- Barcode and package label reading
- Object sorting
- Document understanding (e.g. OCR)
- Security Application
- Biometrics (iris, finger print, face recognition)
- Surveillance − Detecting certain suspicious activities or behaviors
- Transportation Application
- Autonomous vehicle
- Safety, e.g., driver vigilance monitoring

2.1.3Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Openended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

Features: Keras contains numerous implementations of commonly used neural- network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU) principally in conjunction with CUDA.

Keras applications module is used to provide pre-trained model for deep neural networks. Keras models are used for prediction, feature extraction and fine tuning. This chapter explains about Keras applications in detail.

Pre-trained models

Trained model consists of two parts model Architecture and model Weights. Model weights are large file so we have to download and extract the feature from ImageNet database. Some of the popular pre-trained models are listed below,

- ResNet
- VGG16
- MobileNet
- InceptionResNetV2
- InceptionV3

2.1.5 Neural Networks:

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

A neural network works similarly to the human brain’s neural network. A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

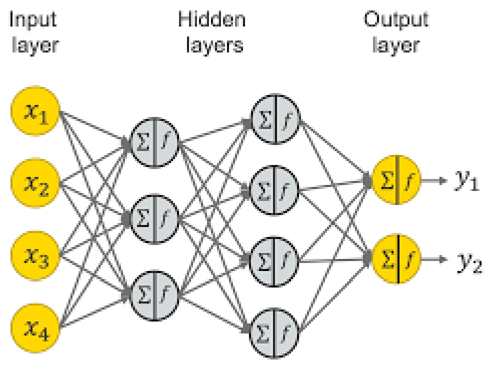


Fig 2.1 Layers in a Neural Network

Areas of Application

Followings are some of the areas, where ANN is being used. It suggests that ANN has an interdisciplinary approach in its development and applications.

Speech Recognition

Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand.

Great progress has been made in this field, however, still such kinds of systems are facing the problem of limited vocabulary or grammar along with the issue of retraining of the system for different speakers in different conditions. ANN is playing a major role in this area. Following ANNs have been used for speech recognition −

Multilayer networks

Multilayer networks with recurrent connections Kohonen self-organizing feature map

The most useful network for this is Kohonen Self-Organizing feature map, which has its input as short segments of the speech waveform. It will map the same kind of phonemes as the output array, called feature extraction technique. After extracting the features, with the help of some acoustic models as back-end processing, it will recognize the utterance.

Character Recognition

It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are some ANNs which have been used for character recognition −

Multilayer neural networks such as Backpropagation neural networks. Neocognitron

Though back-propagation neural networks have several hidden layers, the pattern of connection from one layer to the next is localized. Similarly, neocognitron also has several hidden layers and its training is done layer by layer for such kind of applications.

Signature Verification Application

Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique.

For this application, the first approach is to extract the feature or rather the geometrical feature set representing the signature. With these feature sets, we have to train the neural networks using an efficient neural network algorithm. This trained neural network will classify the signature as being genuine or forged under the verification stage.

Human Face Recognition

It is one of the biometric methods to identify the given face. It is a typical task because

of the characterization of "non-face" images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be preprocessed. Then, the dimensionality of that image must be reduced. And, at last it must be classified using neural network training algorithm. Following neural networks are used for training purposes with preprocessed image −

Fully-connected multilayer feed-forward neural network trained with the help of back-propagation algorithm.

For dimensionality reduction, Principal Component Analysis PCA is used. Deep Learning:

Deep-learning networks are distinguished from the more commonplace single-hidden- layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as "deep" learning. So deep is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven't heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer's output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear

functions.

Above all, these neural nets are capable of discovering latent structures within unlabeled, unstructured data, which is the vast majority of data in the world. Another word for unstructured data is raw media; i.e. pictures, texts, video and audio recordings. Therefore, one of the problems deep learning solves best is in processing and clustering the world's raw, unlabeled media, discerning similarities and anomalies in data that no human has organized in a relational database or ever put a name to.

For example, deep learning can take a million images, and cluster them according to their similarities: cats in one corner, ice breakers in another, and in a third all the photos of your grandmother. This is the basis of so-called smart photo albums.

Deep-learning networks perform automatic feature extraction without human intervention, unlike most traditional machine-learning algorithms. Given that feature extraction is a task that can take teams of data scientists years to accomplish, deep learning is a way to circumvent the chokepoint of limited experts. It augments the powers of small data science teams, which by their nature do not scale.

When training on unlabeled data, each node layer in a deep network learns features automatically by repeatedly trying to reconstruct the input from which it draws its samples, attempting to minimize the difference between the network's guesses and the probability distribution of the input data itself. Restricted Boltzmann machines, for examples, create so-called reconstructions in this manner.

In the process, these neural networks learn to recognize correlations between certain relevant features and optimal results – they draw connections between feature signals and what those features represent, whether it be a full reconstruction, or with labeled data.

A deep-learning network trained on labeled data can then be applied to unstructured data, giving it access to much more input than machine-learning nets.

Convolution neural network:

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users' photos.

Instead of the image, the computer sees an array of pixels. For example, if 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.

To solve this problem the computer looks for the characteristics of the baselevel. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

Applications of convolution neural network: Decoding Facial Recognition:

Facial recognition is broken down by a convolutional neural network into the following major components -

- Identifying every face in the picture
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features

Comparing all the collected data with already existing data in the database to match a face with a name.

A similar process is followed for scene labeling as well. Analyzing Documents:

Convolutional neural networks can also be used for document analysis. This is not just useful for handwriting analysis, but also has a major stake in recognizers. For a machine to be able to scan an individual's writing, and then compare that to the wide database it has, it must execute almost a million commands a minute. It is said with the use of CNNs and newer models and algorithms, the error rate has been brought down to a minimum of 0.4% at a character level, though it's complete testing is yet to be widely seen.

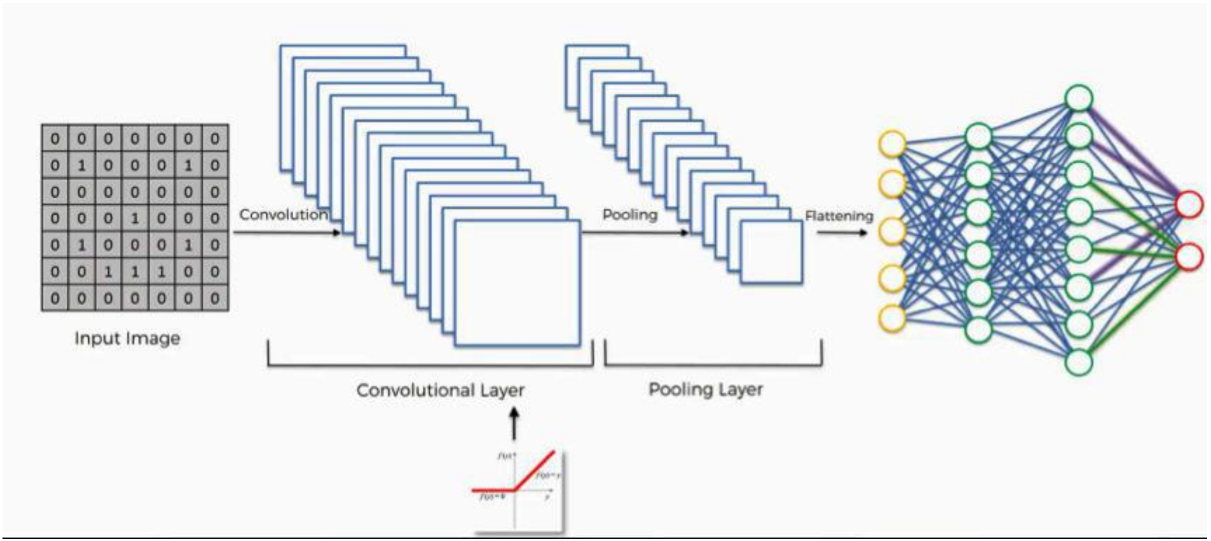


Fig2.2: Layers involved in CNN

2.1.6 Yolo Object Detection:

YOLO is an abbreviation for the term ‘You Only Look Once’. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

Why the YOLO algorithm is important

YOLO algorithm is important because of the following reasons:

- **Speed:** This algorithm improves the speed of detection because it can predict objects in real-time.
- **High accuracy:** YOLO is a predictive technique that provides accurate results with minimal background errors.
- **Learning capabilities:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

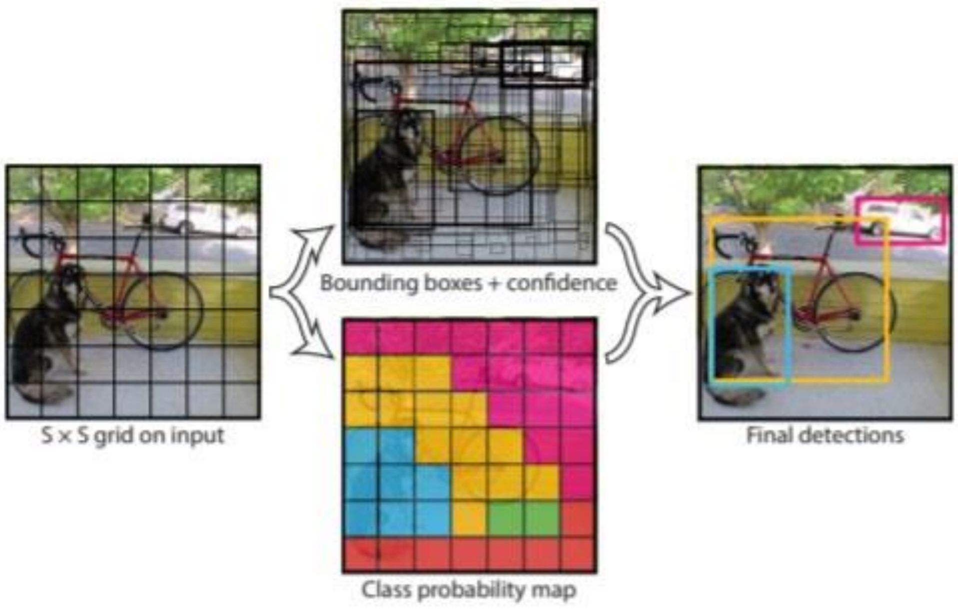


Fig 2.3 Object Detection in Yolo Model

2.2 EXISTING SYSTEM

In Literature survey we have gone through other similar works that are implemented in the domain of sign language recognition. The summaries of each of the project works are mentioned below

A Survey of Hand Gesture Recognition Methods in Sign Language Recognition

Sign Language Recognition (SLR) system, which is required to recognize sign languages, has been widely studied for years. The studies are based on various input sensors, gesture segmentation, extraction of features and classification methods. This paper aims to analyse and compare the methods employed in the SLR systems, classifications methods that have been used, and suggests the most promising method for future research. Due to recent advancement in classification methods, many of the recent proposed works mainly contribute on the classification methods, such as hybrid method and Deep Learning. This paper focuses on the classification methods used in prior Sign Language Recognition system. Based on our review, HMM- based approaches have been explored extensively in prior research, including its modifications.

This study is based on various input sensors, gesture segmentation, extraction of features and classification methods. This paper aims to analyze and compare the methods employed in the SLR systems, classifications methods that have been used, and suggests the most reliable method for future research. Due to recent advancement in classification methods, many of the recently proposed works mainly contribute to the classification methods, such as hybrid method and Deep Learning. Based on our review, HMM-based approaches have been explored extensively in prior research, including its modifications. Hybrid CNN-HMM and fully Deep Learning approaches have shown promising results and offer opportunities for further exploration.

Communication between Deaf-Dumb People and Normal People Chat applications have become a powerful media that assist people to communicate in different languages witheach other. There are lots of chat applications that are used different people in different languages but there are not such chat application that has facilitate to communicate with sign languages. The developed system is based on Sinhala Sign language. The system has included four main components as text messages are converted to sign messages, voice messages are converted to sign messages, sign messages are converted to text messages and sign messages are converted to voice messages. Google voice recognition API has used to develop speech character recognition for voice messages. The system has been trained for the speech and text patterns by using some text parameters and signs of Sinhala Sign language is displayed by emoji. Those emoji and signs that are included in this system will bring the normal people more close to the disabled people. This is a 2 way communication system but it uses pattern of gesture recognition which is not very reliable in getting appropriate output.

A System for Recognition of Indian Sign Language for Deaf People using Otsu's Algorithm

In this paper we proposed some methods, through which the recognition of the signs becomes easy for peoples while communication. And the result of those symbols signs will be converted into the text. In this project, we are capturing hand gestures through webcam and convert this image into grey scale image. The segmentation of grey scale image of a hand gesture is performed using Otsu thresholding algorithm. Total image level is divided into two classes one is hand and other is background. The optimal threshold value is determined by computing the ratio between class variance and total class variance. To find the boundary of hand gesture in image Canny edge detection technique is used. In Canny edge detection we used edge based segmentation and threshold based segmentation. Then Otsu's algorithm is used because of its simple calculation and stability. This algorithm fails, when the global distribution of the target and background vary widely.

Intelligent Sign Language Recognition Using Image Processing

Computer recognition of sign language is an important research problem for enabling communication with hearing impaired people. This project introduces an efficient and fast algorithm for identification of the number of fingers opened in a gesture representing an alphabet of the Binary Sign Language. The system does not require the hand to be perfectly aligned to the camera. The project uses image processing system to identify, especially English alphabetic sign language used by the deaf people to communicate. The basic objective of this project is to develop a computer based intelligent system that will enable dumb people significantly to communicate with all other people using their natural hand gestures. The idea consisted of designing and building up an intelligent system using image processing, machine learning and artificial intelligence concepts to take visual inputs of sign language's hand gestures and generate easily recognizable form of outputs. Hence the objective of this project is to develop an intelligent system which can act as a translator between the sign language and the spoken language dynamically and can make the communication between people with hearing impairment and normal people both effective and efficient. The system is we are implementing for Binary sign language but it can detect any sign language with prior image processing

Sign Language Recognition Using Image Processing

One of the major drawback of our society is the barrier that is created between disabled or handicapped persons and the normal person. Communication is the only medium by which we can share our thoughts or convey the message but for a person with disability (deaf and dumb) faces difficulty in communication with normal person. For many deaf and dumb people , sign language is the basic means of communication. Sign language recognition (SLR) aims to interpret sign languages automatically by a computer in order to help the deaf communicate with hearing society conveniently. Our aim is to design a system to help the person who trained the hearing impaired to communicate with the rest of the world using sign language or hand gesture recognition techniques. In this system, feature detection and feature extraction of hand gesture is done with the help of SURF algorithm using image processing. All this work is done using MATLAB software. With the help of this algorithm, a person can easily trained a deaf and dumb.

Sign Language Interpreter using Image Processing and Machine Learning

Speech impairment is a disability which affects one's ability to speak and hear. Such individuals use sign language to communicate with other people. Although it is an effective form of communication, there remains a challenge for people who do not understand sign language to communicate with speech impaired people. The aim of this paper is to develop an application which will translate sign language to English in the form of text and audio, thus aiding communication with sign language. The application acquires image data using the webcam of the computer, then it is pre-processed using a combinational algorithm and recognition is done using template matching. The translation in the form of text is then converted to audio. The database used for this system includes 6000 images of English alphabets. We used 4800 images for training and 1200 images for testing. The system produces 88% accuracy.

GESTURE RECOGNITION SYSTEM

Communication plays a crucial part in human life. It encourages a man to pass on his sentiments, feelings and messages by talking, composing or by utilizing some other medium. Gesture based communication is the main method for Communication for the discourse and hearing weakened individuals. Communication via gestures is a dialect that utilizations outwardly transmitted motions that consolidates hand signs and development of the hands, arms, lip designs, body developments and outward appearances, rather than utilizing discourse or content, to express the individual's musings. Gestures are the expressive and important body developments that speaks to some message or data. Gestures are the requirement for hearing and discourse hindered, they pass on their message to others just with the assistance of motions. Gesture Recognition System is the capacity of the computer interface to catch, track and perceive the motions and deliver the yield in light of the caught signals. It enables the clients to interface with machines (HMI) without the any need of mechanical gadgets. There are two sorts of sign recognition methods: image- based and sensor- based strategies. Image based approach is utilized as a part of this project that manages communication via gestures motions to distinguish and track the signs and change over them into the relating discourse and content.

2.3 PROPOSED SYSTEM

Our proposed system is sign language recognition system using convolution neural networks which recognizes various word level hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained Yolo Object Detection model. Thus our system is more robust in getting exact text labels of letters.

2.3.1 System Architecture

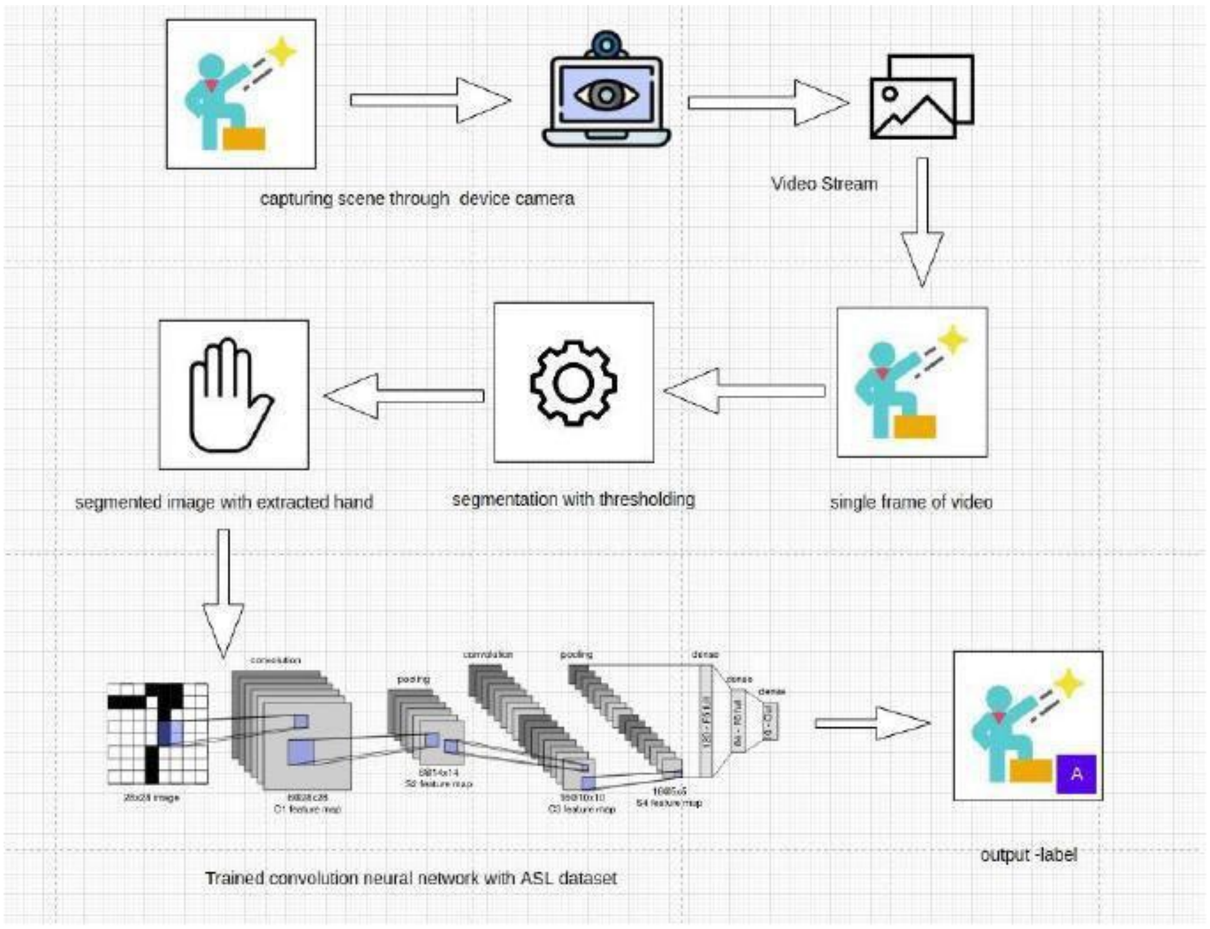

capturing scene through device camera
Video Stream
segmented image with extracted hand
segmentation with thresholding
single frame of video
Trained convolution neural network with ASL dataset
output -label
A

Fig2.4 Architecture of Sign Language recognition System

3. METHODOLOGY

3.1 TRAINING MODULE:

Supervised machine learning:It is one of the ways of machine learning where the model is trained by input data and expected output data. To create such model, it is necessary to go through the following phases:

1. model construction

2. model training

3. model testing

4. model evaluation

Model construction: It depends on machine learning algorithms. In this projectscase, it was neural networks.Such an agorithm looks like:

1. begin with its object: model = Sequential()

2. then consist of layers with their types: model.add(type_of_layer())

3. after adding a sufficient number of layers the model is compiled. At this moment Keras communicates with TensorFlow for construction of the model. It looks like: model.comile(loss= ‘name_of_loss_function’, optimizer= ‘name_of_opimazer_alg’) The loss function shows the accuracy of each prediction made by the model.

Before model training it is important to scale data for their further use.

Model training:

After model construction it is time for model training. In this phase, the model is trained using training data and expected output for this data. It’s look this way:

model.fit(training_data, expected_output). Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model.

Model Testing:

During this phase a second set of data is loaded. This data set has never been seen by the model and therefore it’s true accuracy will be verified. After the model training is complete, and it is understood that the model shows the right result, it can be saved by: model.save(“name_of_file.h5”). Finally, the saved model can be used in the real world. The name of this phase is model evaluation.

DATASET USED FOR TRAINING

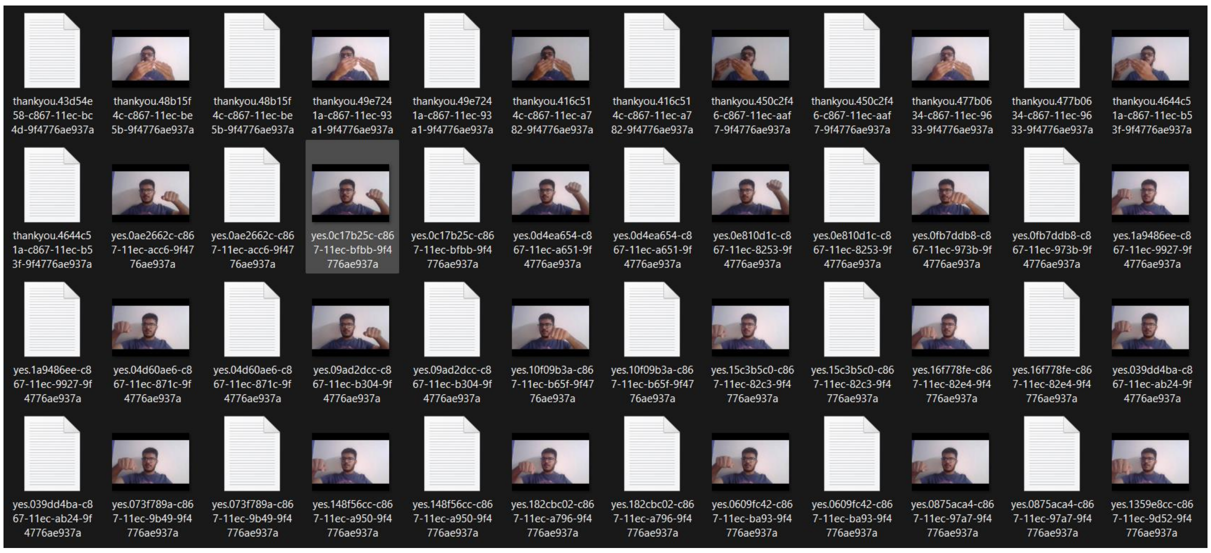
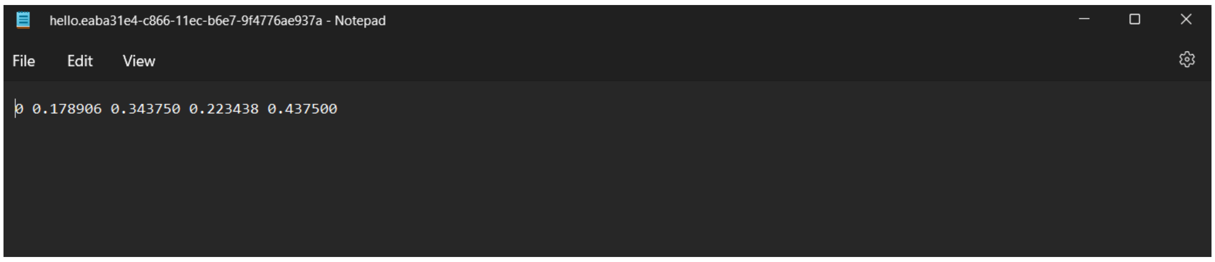

thankyou.43d54e58-c867-11ec-bc4d-9f4776ae937a
thankyou.48b15f4c-c867-11ec-be5b-9f4776ae937a
thankyou.48b15f4c-c867-11ec-be5b-9f4776ae937a
thankyou.49e7241a-c867-11ec-93a1-9f4776ae937a
thankyou.49e7241a-c867-11ec-93a1-9f4776ae937a
thankyou.416c514c-c867-11ec-a782-9f4776ae937a
thankyou.416c514c-c867-11ec-a782-9f4776ae937a
thankyou.450c2f46-c867-11ec-aaf7-9f4776ae937a
thankyou.450c2f46-c867-11ec-aaf7-9f4776ae937a
thankyou.477b0634-c867-11ec-9633-9f4776ae937a
thankyou.477b0634-c867-11ec-9633-9f4776ae937a
thankyou.4644c51a-c867-11ec-b53f-9f4776ae937a
thankyou.4644c51a-c867-11ec-b53f-9f4776ae937a
yes.0ae2662c-c867-11ec-acc6-9f4776ae937a
yes.0ae2662c-c867-11ec-acc6-9f4776ae937a
yes.0c17b25c-c867-11ec-bfbb-9f4776ae937a
yes.0c17b25c-c867-11ec-bfbb-9f4776ae937a
yes.0d4ea654-c867-11ec-a651-9f4776ae937a
yes.0d4ea654-c867-11ec-a651-9f4776ae937a
yes.0e810d1c-c867-11ec-8253-9f4776ae937a
yes.0e810d1c-c867-11ec-8253-9f4776ae937a
yes.0fb7ddb8-c867-11ec-973b-9f4776ae937a
yes.0fb7ddb8-c867-11ec-973b-9f4776ae937a
yes.1a9486ee-c867-11ec-9927-9f4776ae937a
yes.1a9486ee-c867-11ec-9927-9f4776ae937a
yes.04d60ae6-c867-11ec-871c-9f4776ae937a
yes.04d60ae6-c867-11ec-871c-9f4776ae937a
yes.09ad2dcc-c867-11ec-b304-9f4776ae937a
yes.09ad2dcc-c867-11ec-b304-9f4776ae937a
yes.10f09b3a-c867-11ec-b65f-9f4776ae937a
yes.10f09b3a-c867-11ec-b65f-9f4776ae937a
yes.15c3b5c0-c867-11ec-82c3-9f4776ae937a
yes.15c3b5c0-c867-11ec-82c3-9f4776ae937a
yes.16f778fe-c867-11ec-82e4-9f4776ae937a
yes.16f778fe-c867-11ec-82e4-9f4776ae937a
yes.039dd4ba-c867-11ec-ab24-9f4776ae937a
yes.039dd4ba-c867-11ec-ab24-9f4776ae937a
yes.073f789a-c867-11ec-9b49-9f4776ae937a
yes.073f789a-c867-11ec-9b49-9f4776ae937a
yes.148f56cc-c867-11ec-a950-9f4776ae937a
yes.148f56cc-c867-11ec-a950-9f4776ae937a
yes.182cbc02-c867-11ec-a796-9f4776ae937a
yes.182cbc02-c867-11ec-a796-9f4776ae937a
yes.0609fc42-c867-11ec-ba93-9f4776ae937a
yes.0609fc42-c867-11ec-ba93-9f4776ae937a
yes.0875aca4-c867-11ec-97a7-9f4776ae937a
yes.0875aca4-c867-11ec-97a7-9f4776ae937a
yes.1359e8cc-c867-11ec-9d52-9f4776ae937a

Fig 3.1 Training Dataset

Image Labels



```
0 0.178906 0.343750 0.223438 0.437500
```

Fig 3.2 Yolo Image Labels

3.2 ALGORITHM

HISTOGRAM CALCULATION:

Histograms are collected counts of data organized into a set of predefined bins

When we say data we are not restricting it to be intensity value. The data collected can be whatever feature you find useful to describe your image.

Let's see an example. Imagine that a Matrix contains information of an image (i.e. intensity in the range 0−255):

What happens if we want to count this data in an organized way? Since we know that the range of information value for this case is 256 values, we can segment our range in subparts (called bins) like:

$[0,255]=[0,15]\cup[16,31]\cup....\cup[240,255]range=bin1\cup bin2\cup....\cup binn=15$

and we can keep count of the number of pixels that fall in the range of each bins.

Back Propagation: Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

Optimizer(Adam): Adam can be looked at as a combination ofRMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network. Now, what is moment? Nth moment of a random variable is defined as the expected value of that variable to the power of n. More formally:

Loss Function(categorical cross entropy): Categorical cross entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

Note. The block before the Target block must use the activation function Softmax.

3.3 SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse. Modern image segmentation techniques are powered by deep learning technology. Here are several deep learning architectures used for segmentation:

Why does Image Segmentation even matter?

If we take an example of Autonomous Vehicles, they need sensory input devices like cameras, radar, and lasers to allow the car to perceive the world around it, creating a digital map. Autonomous driving is not even possible without object detection which itself involves image classification/segmentation.

How Image Segmentation works

Image Segmentation involves converting an image into a collection of regions of pixels that are represented by a mask or a labelled image. By dividing an image into segments, you can process only the important segments of the image instead of processing the entire image. A common technique is to look for abrupt discontinuities in pixel values, which typically indicate edges that define a region. Another common approach is to detect similarities in the regions of an image. Some techniques that follow this approach are region growing, clustering, and thresholding. A variety of other approaches to perform image segmentation have been developed over the years using domain-specific knowledge to effectively solve segmentation problems in specific application areas.

3.4 4 CLASSIFICATION CONVOLUTION NEURAL NETWORK

Image classification is the process of taking an input(like a picture) and outputting its class or probability that the input is a particular class. Neural networks are applied in the following steps:

1) One hot encode the data: A one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

2) Define the model: A model said in a very simplified form is nothing but a function that is used to take in certain input, perform certain operation to its beston the given input (learning and then predicting/classifying) and produce the suitable output.

3) Compile the model: The optimizer controls the learning rate. We will be using 'adam' as our optmizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

4) Train the model: Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

5) Test the model

A convolutional neural network convolves learned featured with input data and uses 2D convolution layers.

Convolution Operation:

In purely mathematical terms, convolution is a function derived from two given functions by integration which expresses how the shape of one is modified by the other.

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Fig 3.3 Convolution formula

Here are the three elements that enter into the convolution operation:

• Input image

• Feature detector

• Feature map

Steps to apply convolution layer:

• You place it over the input image beginning from the top-left corner within the borders you see demarcated above, and then you count the number of cells in which the feature detector matches the input image.

• The number of matching cells is then inserted in the top-left cell of the feature map

• You then move the feature detector one cell to the right and do the same thing. This movement is called a and since we are moving the feature detector one cell at time, that would be called a stride of one pixel.

• What you will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, and so you write “1” in the next cell in the feature map, and so on and so forth.

• After you have gone through the whole first row, you can then move it over to the next row and go through the same process.

There are several uses that we gain from deriving a feature map. These are the most important of them: Reducing the size of the input image, and you should know that the larger your strides (the movements across pixels), the smaller your feature map.

3.5 TESTING

The purpose of testing is to discover errors. Testing is a process of trying to discover every conceivable fault or weakness in a work product.

It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner.

Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for well planned through testing.

Testing Objectives:

There are several rules that can serve as testing objectives they are:

- Testing is a process of executing program with the intent of finding an error.
- A good test case is the one that has a high probability of finding an undiscovered error.

Types of Testing:

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at different phases of software development are :

Unit Testing:

Unit testing is done on individual models as they are completed and becomes executable. It is confined only to the designer's requirements. Unit testing is different from and should be preceded by other techniques, including:

- Inform Debugging
- Code Inspection

Black Box testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program.

This testing has been used to find error in the following categories: Incorrect or missing functions

- Interface errors
- Errors in data structures are external database access
- Performance error
- Initialisation and termination of errors
- In this testing only the output is checked for correctness
- The logical flow of data is not checked

id	Test case	Input description	Expected output	Test status
1	Loadind model	Initializing trained model and load it into ON	Loaded model without errors	pass
2	Converting video to frames	Capturing video and converting it into frames	Image frames of captured video stream	pass
3	Recognize hand gesture	Image frame that contains hand object	label	Pass

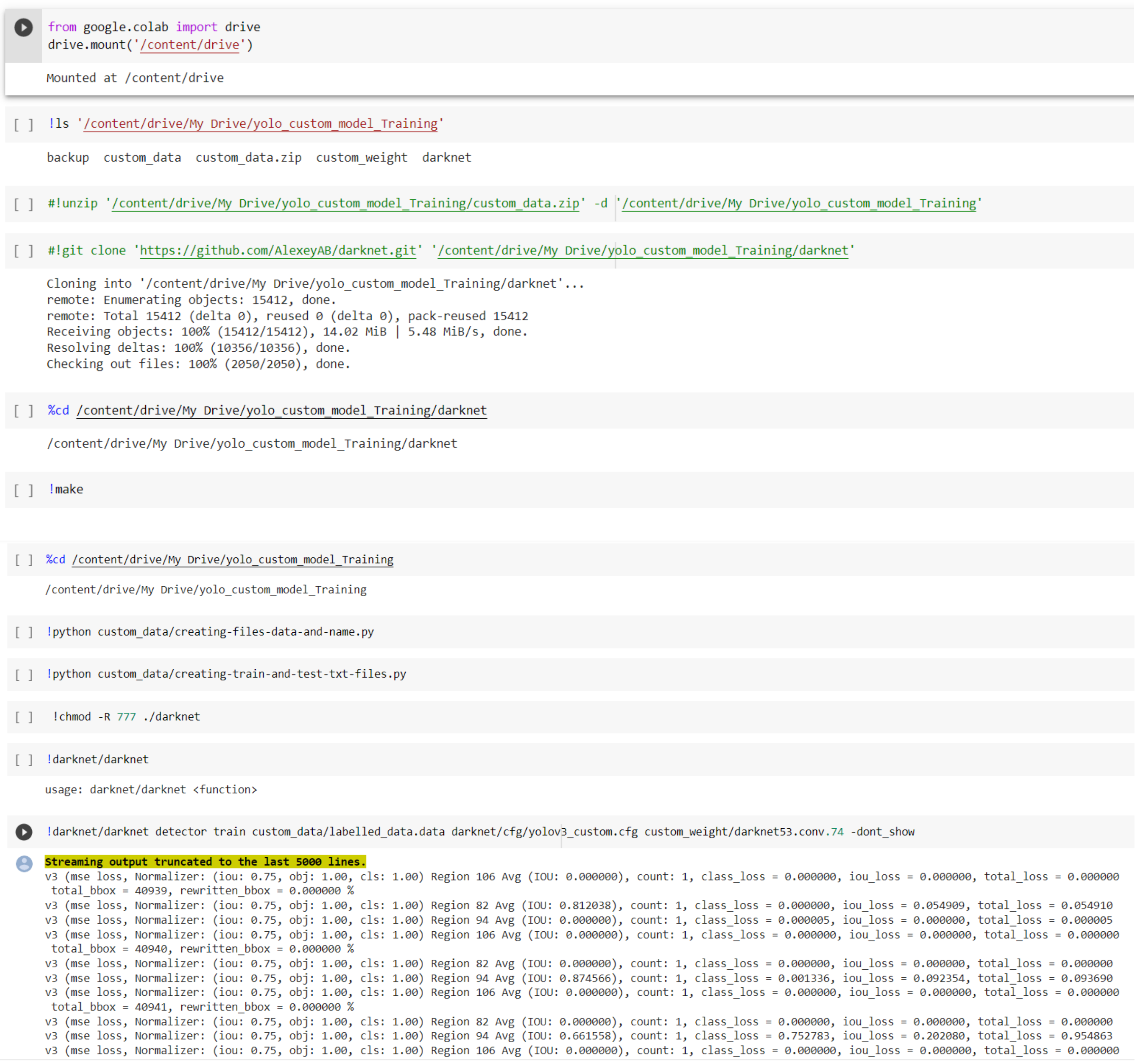
Table3.1: verification of testcases

3.6 Implementation

Code To Collect Data

```
In [ ]: import cv2 #opencv
        import os
        import time
        import uuid

In [ ]: IMAGES_PATH = Tensorflow / workspace / images / collectedimages

In [ ]: labels = [ ' hello ' , ' thanks ' , ' yes ' , ' no ' , " iloveyou " ] #Classes
        number_ings = 20 #Number of Images

In [ ]: for label in labels :
            Imkdir {"Tensorflow \ workspace \ images \ collectedimages \\ " + label}
            cap = cv2.VideoCapture(0)
            print("Collecting images for {}".format(label))
            time.sleep(5)
            for imgnum in range(number_imgs):
                ret,frame cap.read()
                imgname = os.path.join(IMAGES_PATH,label,label +"." + "{}.jpg".format(str(uuid.uuid1())))
                cv2.mwrite(imgname,frame)
                cv2.imshow('frame ',frame)
                time.sleep(2)
                if cv2.waitkey(1) && 0xFF == ord ('q'):
                    break
        cap.release()
```

Model Training



```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
[ ] !ls '/content/drive/My Drive/yolo_custom_model_Training'

    backup  custom_data  custom_data.zip  custom_weight  darknet
```

```
[ ] #!unzip '/content/drive/My Drive/yolo_custom_model_Training/custom_data.zip' -d '/content/drive/My Drive/yolo_custom_model_Training'
```

```
[ ] #!git clone 'https://github.com/AlexeyAB/darknet.git' '/content/drive/My Drive/yolo_custom_model_Training/darknet'

    Cloning into '/content/drive/My Drive/yolo_custom_model_Training/darknet'...
    remote: Enumerating objects: 15412, done.
    remote: Total 15412 (delta 0), reused 0 (delta 0), pack-reused 15412
    Receiving objects: 100% (15412/15412), 14.02 MiB | 5.48 MiB/s, done.
    Resolving deltas: 100% (10356/10356), done.
    Checking out files: 100% (2050/2050), done.
```

```
[ ] %cd /content/drive/My Drive/yolo_custom_model_Training/darknet

    /content/drive/My Drive/yolo_custom_model_Training/darknet
```

```
[ ] !make
```

```
[ ] %cd /content/drive/My Drive/yolo_custom_model_Training

    /content/drive/My Drive/yolo_custom_model_Training
```

```
[ ] !python custom_data/creating-files-data-and-name.py
```

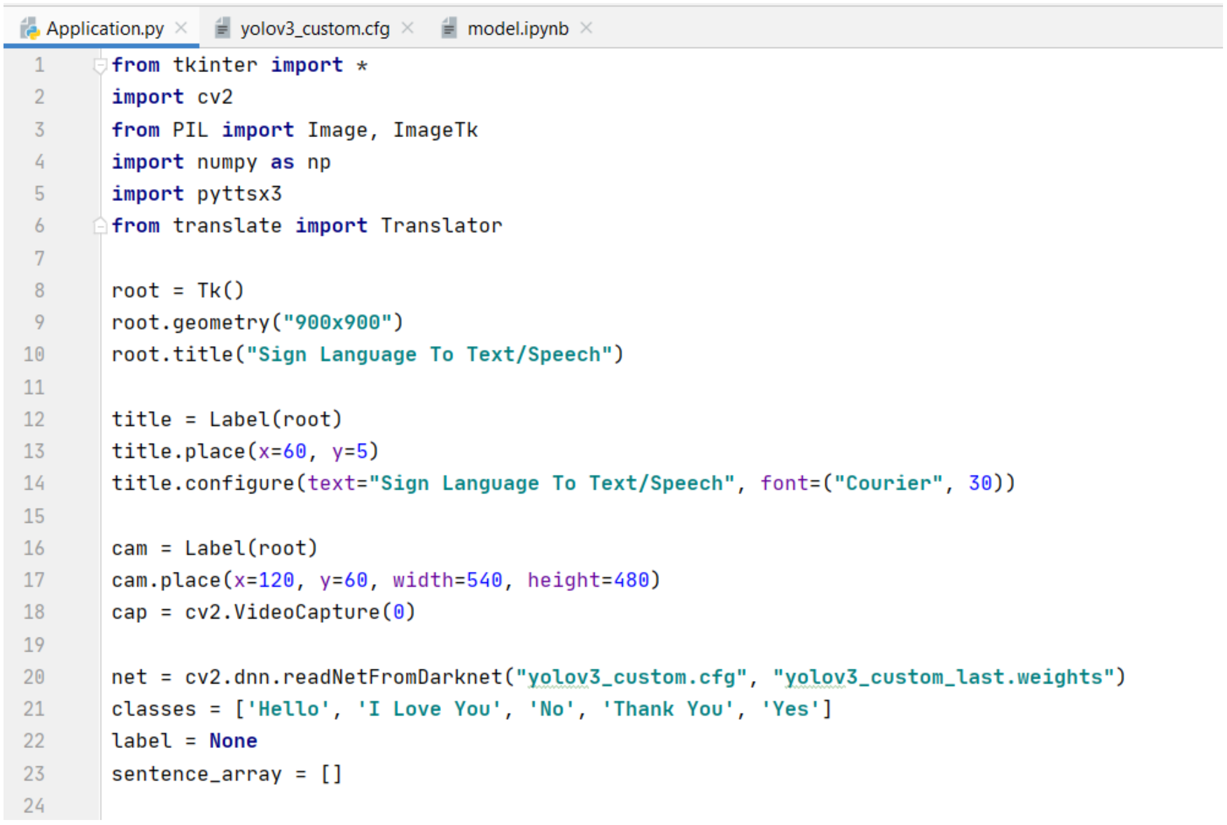
```
[ ] !python custom_data/creating-train-and-test-txt-files.py
```

```
[ ]  !chmod -R 777 ./darknet
```

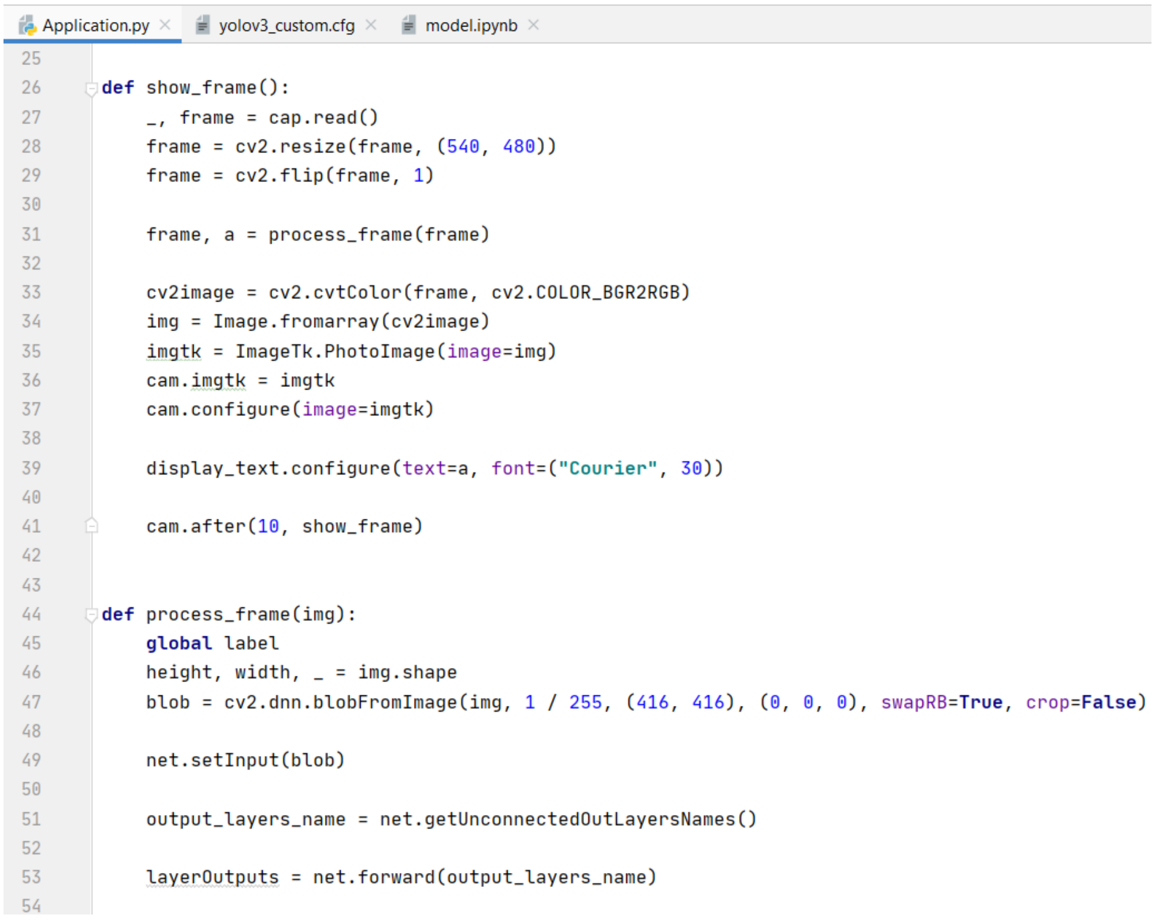
```
[ ] !darknet/darknet

    usage: darknet/darknet <function>
```

```
!darknet/darknet detector train custom_data/labelled_data.data darknet/cfg/yolov3_custom.cfg custom_weight/darknet53.conv.74 -dont_show

Streaming output truncated to the last 5000 lines.
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
 total_bbox = 40939, rewritten_bbox = 0.000000 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.812038), count: 1, class_loss = 0.000000, iou_loss = 0.054909, total_loss = 0.054910
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.000005, iou_loss = 0.000000, total_loss = 0.000005
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
 total_bbox = 40940, rewritten_bbox = 0.000000 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.874566), count: 1, class_loss = 0.001336, iou_loss = 0.092354, total_loss = 0.093690
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
 total_bbox = 40941, rewritten_bbox = 0.000000 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.661558), count: 1, class_loss = 0.752783, iou_loss = 0.202080, total_loss = 0.954863
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
```

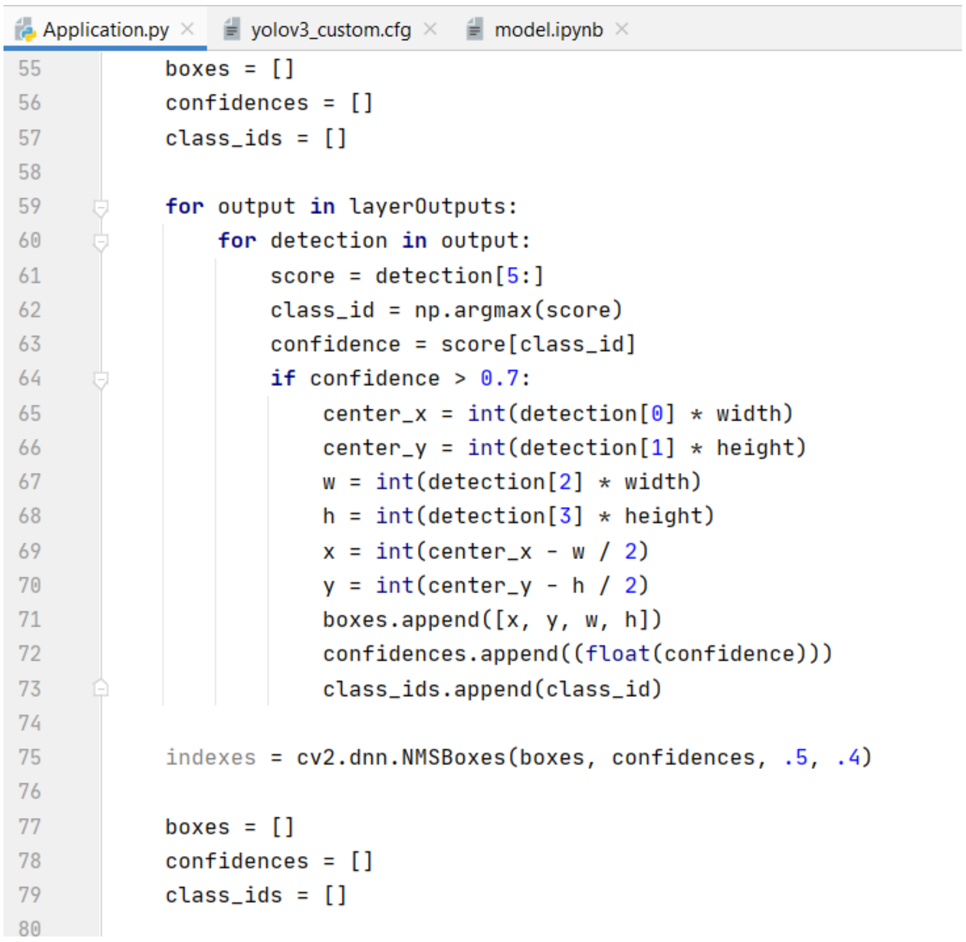
Application



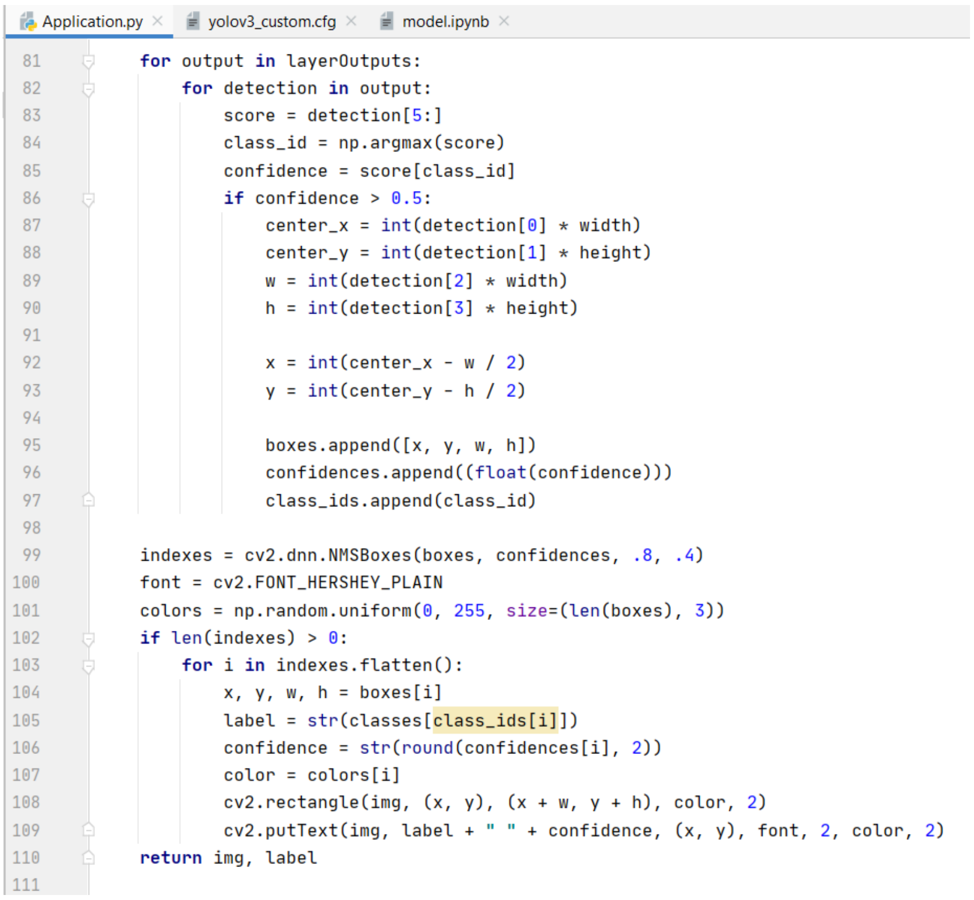
```python
from tkinter import *
import cv2
from PIL import Image, ImageTk
import numpy as np
import pyttsx3
from translate import Translator

root = Tk()
root.geometry("900x900")
root.title("Sign Language To Text/Speech")

title = Label(root)
title.place(x=60, y=5)
title.configure(text="Sign Language To Text/Speech", font=("Courier", 30))

cam = Label(root)
cam.place(x=120, y=60, width=540, height=480)
cap = cv2.VideoCapture(0)

net = cv2.dnn.readNetFromDarknet("yolov3_custom.cfg", "yolov3_custom_last.weights")
classes = ['Hello', 'I Love You', 'No', 'Thank You', 'Yes']
label = None
sentence_array = []

```



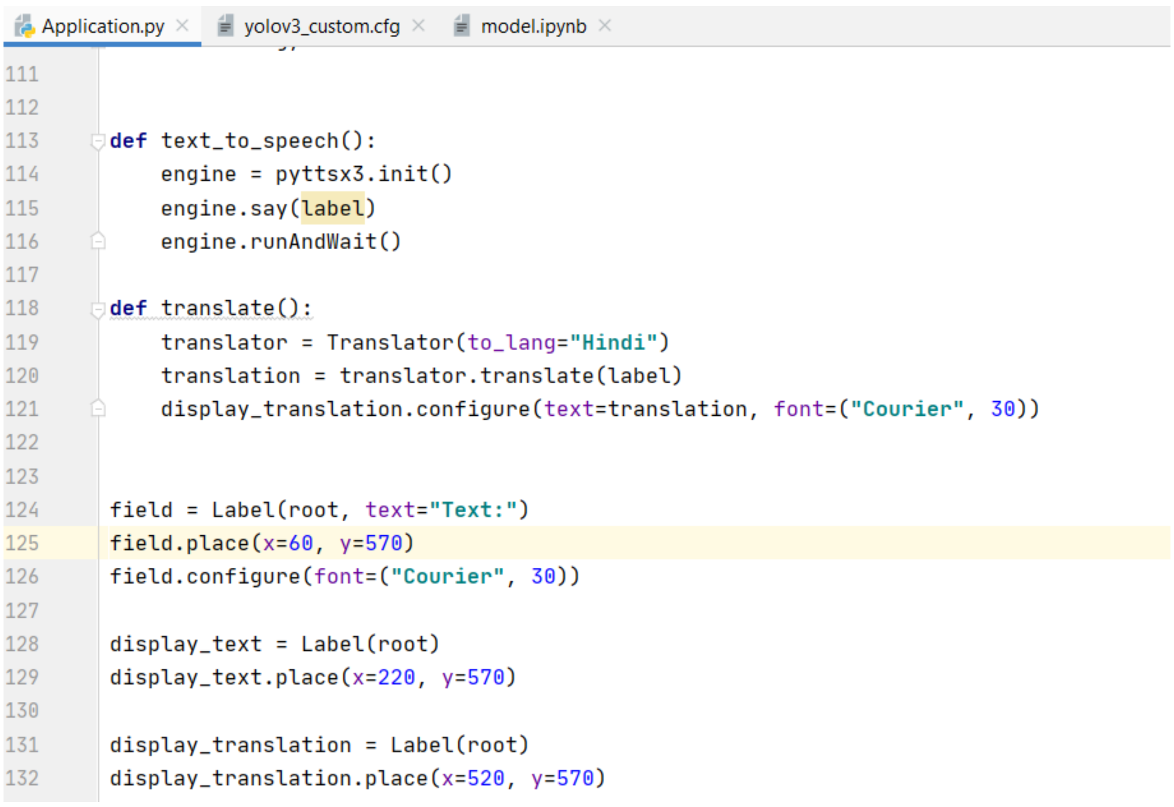
```python

def show_frame():
    _, frame = cap.read()
    frame = cv2.resize(frame, (540, 480))
    frame = cv2.flip(frame, 1)

    frame, a = process_frame(frame)

    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    cam.imgtk = imgtk
    cam.configure(image=imgtk)

    display_text.configure(text=a, font=("Courier", 30))

    cam.after(10, show_frame)


def process_frame(img):
    global label
    height, width, _ = img.shape
    blob = cv2.dnn.blobFromImage(img, 1 / 255, (416, 416), (0, 0, 0), swapRB=True, crop=False)

    net.setInput(blob)

    output_layers_name = net.getUnconnectedOutLayersNames()

    layerOutputs = net.forward(output_layers_name)

```



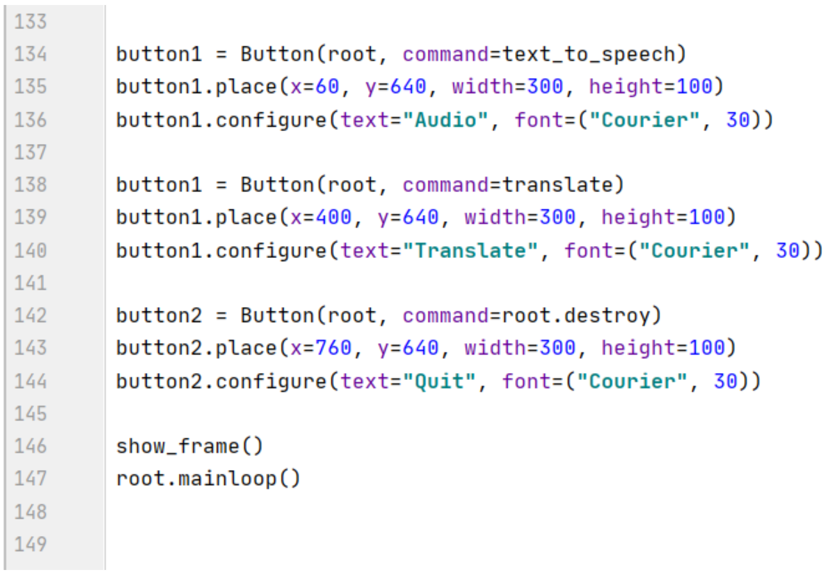
```
    boxes = []
    confidences = []
    class_ids = []

    for output in layerOutputs:
        for detection in output:
            score = detection[5:]
            class_id = np.argmax(score)
            confidence = score[class_id]
            if confidence > 0.7:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append((float(confidence)))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, .5, .4)

    boxes = []
    confidences = []
    class_ids = []

```



```
    for output in layerOutputs:
        for detection in output:
            score = detection[5:]
            class_id = np.argmax(score)
            confidence = score[class_id]
            if confidence > 0.5:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append((float(confidence)))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, .8, .4)
    font = cv2.FONT_HERSHEY_PLAIN
    colors = np.random.uniform(0, 255, size=(len(boxes), 3))
    if len(indexes) > 0:
        for i in indexes.flatten():
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = str(round(confidences[i], 2))
            color = colors[i]
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
            cv2.putText(img, label + " " + confidence, (x, y), font, 2, color, 2)
    return img, label

```



```python


def text_to_speech():
    engine = pyttsx3.init()
    engine.say(label)
    engine.runAndWait()

def translate():
    translator = Translator(to_lang="Hindi")
    translation = translator.translate(label)
    display_translation.configure(text=translation, font=("Courier", 30))


field = Label(root, text="Text:")
field.place(x=60, y=570)
field.configure(font=("Courier", 30))

display_text = Label(root)
display_text.place(x=220, y=570)

display_translation = Label(root)
display_translation.place(x=520, y=570)
```

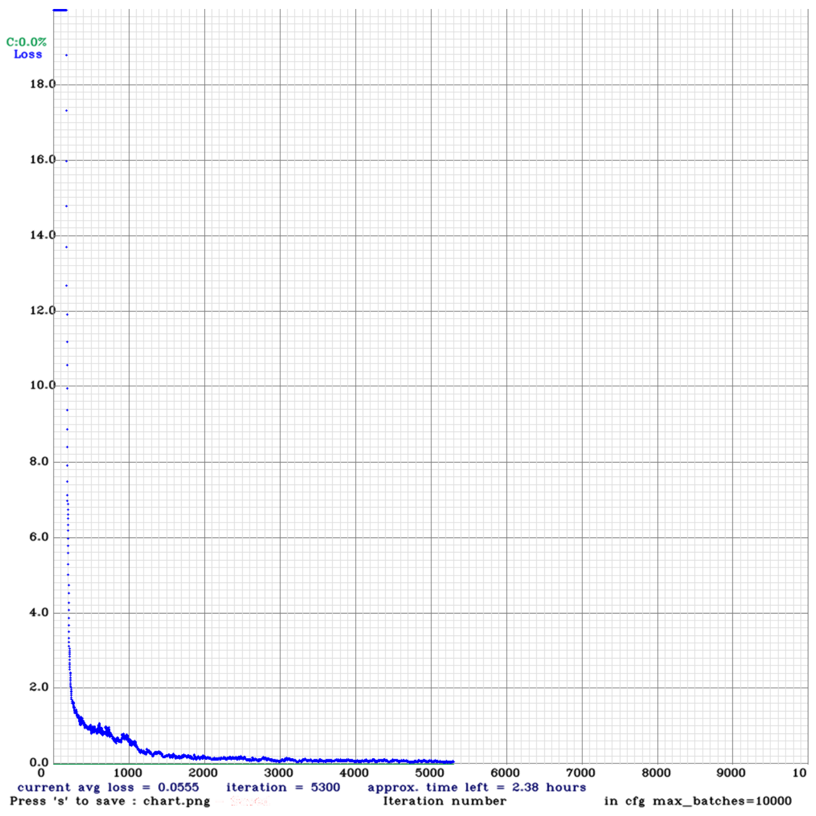


```python

button1 = Button(root, command=text_to_speech)
button1.place(x=60, y=640, width=300, height=100)
button1.configure(text="Audio", font=("Courier", 30))

button1 = Button(root, command=translate)
button1.place(x=400, y=640, width=300, height=100)
button1.configure(text="Translate", font=("Courier", 30))

button2 = Button(root, command=root.destroy)
button2.place(x=760, y=640, width=300, height=100)
button2.configure(text="Quit", font=("Courier", 30))

show_frame()
root.mainloop()


```

4. Results

For examining the effectiveness of the proposed model, several experiments were conducted for longer period of time. Parameters were heavily turned to enhance the performance of the proposed model. Several results were obtained and different conclusions were drawn but this study proposes the valid experiments. In this work, various models were used. Also, various learning rate variations were used in order to manage the small datasets to fit into the deep learning convolutional model.

Loss Vs Iterations



```
current avg loss = 0.0555    iteration = 5300    approx. time left = 2.38 hours
Press 's' to save : chart.png                                      in cfg max_batches=10000
```

Fig 4.1 Loss Vs Iterations Graph

Average Loss = 0.055

Iterations = 5300

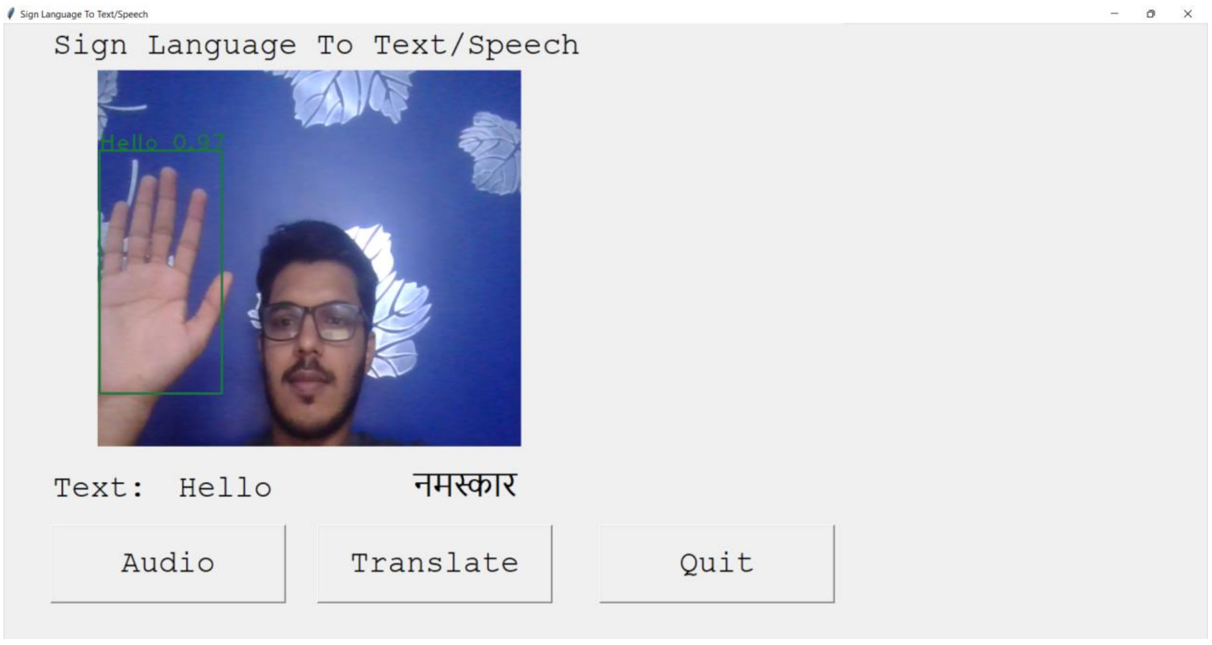


Fig 4.2 Application

REFRENCES

Base Paper:

[1] http://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf

[2] http://www.iosrjen.org/Papers/vol3_issue2%20(part-2)/H03224551.pdf

Other References:

[3]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&typ e=pdf

[4]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&typ e=pdf

[5] https://ieeexplore.ieee.org/document/7507939

[6] https://ieeexplore.ieee.org/document/7916786

[7] https://www.sciencedirect.com/science/article/pii/S1877050917320720

[8] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural- network-cnn-deep-learning-99760835f148

[9]https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/