BEETHOVEN : INNOVATIVE SIGN LANGUAGE TRANSLATION USING MACHINE LEARNING

A PROJECT REPORT

submitted by

AADEL ABOOBACKER (TLY16CS001)

ABHIJITH N (TLY16CS002)

AMAL P K (TLY16CS008)

SAFDAR SHAHIR (TLY16CS044)

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the degree

of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING THALASSERY

THALASSERY, KERALA - 670107

JUNE 2020

DECLARATION

I/We undersigned hereby declare that the project **"Innovative Sign Language Translation"** , submitted for partial fulfillment of the requirements for the award of degree of Master of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Nikhil Dharman M K** . This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Aadel Aboobacker
TLY16CS001

Abhijith N
TLY16CS002

Amal P K
TLY16CS008

Safdar Shahir
TLY16CS044

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING THALASSERY

CERTIFICATE

This is to certify that the report entitled **INNOVATIVE SIGN LANGUAGE TRANSLATION USING MACHINE LEARNING** submitted by **Aadel Aboobacker(TLY16CS001), Abhijith N(TLY16CS002), Amal P K(TLY16CS008)** and **Safdar Shahir(TLY16CS044)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Nikhil Dharman M K	**H A Nisha Rose**
Project Guide	HOD
Assistant Professor	Professor
Dept. of CSE	Dept. of CSE
College of Engineering Thalassery	College of Engineering Thalassery

Place: Thalassery

Date: 19 June 2020

ACKNOWLEDGEMENTS

I take this opportunity to express my thanks to **Prof. H A NISHA ROSE**, *Head of the Department, Computer Science and Engineering, for her continuous support and inspiration given during the study and analysis of the Project topic.*

I express my sincere gratitude to Project Guide Prof. **Nikhil Dharman M .K** *for his dutifulness and support.*

Finally I express my humble gratitude and thanks to all my teachers and other faculties of the Department of Computer Science and Engineering for their sincere and friendly cooperation in completing this Project.

ABSTRACT

Muteness/mutism or the inability to speak is considered to be a true disability.There are number of ways in which they can communicate and one such way is using sign language. Sign languages are full-fledged natural languages with their own grammar and lexicon. It is currently the most commonly used and effective communication language for hearing and speech impaired people. The foretold being a language which takes some time to absorb, most of us cannot remember or practice the signs as it is, also they do not need it other than communicating with the mentioned community.

The main focus of this work is to find means to effectively deal with the above mentioned communication issue using Machine Learning algorithms to train the path captured using mobile Camera. The dataset is thenceforth utilized to analyze the gestures and output the word. Developing sign language application for deaf people can be very important, as they'll be able to communicate easily with even those who don't understand sign language. The project aims at taking the basic step in bridging the communication gap between normal people, deaf and dumb people using sign language.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND ESSENTIALS

More than 30 million people around the globe have speech impairments and must rely on sign language, which poses a language barrier when seeking to communicate with non-sign language users But learning sign language is a tedious and time taking process. At this age of technology, it is quintessential to make these people feel part of the society by helping them communicate smoothly.

Automatic sign language recognition is a research area that encompasses human-computer interaction, computer vision and machine learning. Robust automatic recognition of sign language could assist in the translation process and the integration of hearing-impaired people, as well as the teaching of sign language to the hearing population.

The project aims at identifying alphabets in Indian Sign Language from the corresponding gestures. Gesture recognition and sign language recognition have been a well researched topic for American Sign Language(ASL), but few research works have been published regarding Indian Sign Language(ISL). But instead of using high-end technology like gloves or kinect, we aim to solve this problem using state of the art computer vision and machine learning algorithms

1.1 MOTIVATION

People who suffer from hearing and/or speech disorder can be a victim of social isolation. Hearing impaired or speech disabled persons can become targets of bullying among normal people due to their disorders. The bullying can result in lowering of self-esteem in hearing or speech impaired persons, especially among young children.

As foretold, sign language is one of the most efficient methods to communicate amongst the speech and hearing impaired community and outside the community. Unfortunately, there might not be a human interpreter available to us each time we have to communicate. Also, Professional sign language interpretation requires specific skills such as perfect knowledge of each sign - both finger spelling, word level sign vocabulary and non manual features. The interpreter must also be a good observer and must be able to think quickly. The person must also have experience in many fields so that he/she understands the specific words concerned with the field.

Learning any sign language is not easier than learning spoken English or any other spoken language. It takes at least 2 to 3 years of courses to attain a beginning-intermediate skill. To attain an intermediate fluent skill, it takes another 2 years in the ISL/English interpretation training. To become fluent, it takes some time, at least a few years or more, after the graduation from ISL/English interpretation program.

But we cannot solely rely on the availability and skills of such interpreters for each time we communicate. Especially when we have no idea even what a hand gesture means. Like all the necessity demands a solution, this dilemma should have a technological solution which includes translation of finger spelled and word level sign language to text.

1.2 INDIAN SIGN LANGUAGE

ISL is the predominant sign language in South Asia, used by at least several hundred thousand deaf signers (2003). As with many sign languages, it is difficult to estimate numbers with any certainty, as the Census of India does not list sign languages and most studies have focused on the north and on urban areas.

The Indian deaf population of 1.1 million is 98% illiterate[citation needed]. In line with oralist philosophy, deaf schools attempt early intervention with hearing aids etc., but these are largely dysfunctional in an impoverished society. As of 1986, only 2% of deaf children attended school.[citation needed] SIGN LANGUAGE INTERPRETER India have 250 plus sign language interpreter who are active in interpretation most notable interpreter are Monica Punjabi ,gaurav Verma ,Sandeep Dubey.

Deaf schools in the region are overwhelmingly oralist in their approach.[9] Unlike American Sign Language (ASL) and sign languages of European countries, ISL is in rudimentary stage of its development. The Deaf communities of India are still struggling for ISL to gain the status of sign language as a minority language. Though sign language is used by many deaf people in India, it is not used officially in schools for teaching purposes.In 2005, India the National Curricular Framework (NCF) gave some degree of legitimacy to sign language education, by hinting that sign languages may qualify as an optional third language choice for hearing students. NCERT in March 2006 launched a class III text includes a chapter on sign language, emphasising the fact that it is a language like any other and is "yet another mode of communication." The aim was to create healthy attitudes towards the differently-abled.

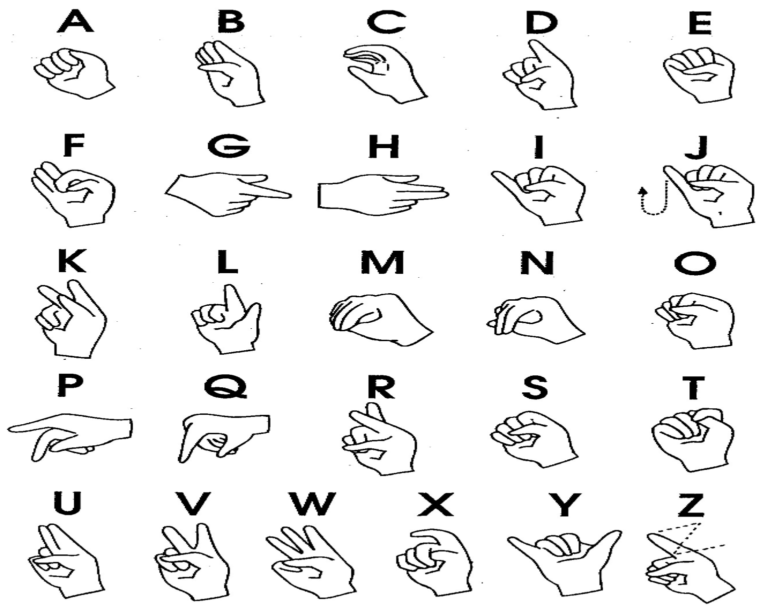


Figure 1.1: ISL Single handed English Alphabet notations

With the advancement of the technology, different methods of interaction with the computers have developed. More traditionally with a keyboard, mouse and thereafter joysticks, track-pads, electro-mechanical gloves, etc. have been used. Apart from these methods, gesture recognition has also been used and this can be considered as a more natural mode of interaction since it mimics normal conversation with a human being. Therefore, it provides a good interface for human computer interaction (HCI). Hand gesture recognition has many applications like general computer interaction, game play, sign language recognition, robotic device manipulation, etc. Sign language is a visual language and has 3 major components.

FINGER-SPELLING	WORD LEVEL	NON-MANUAL FEATURES
letter by letter	Vocabulary	Facial expressions

Table 1.1: Major components of Sign Language in visual aspects

Gestures can be identified as a meaningful body motion which can involve hands, head, body in order to convey a meaningful information or to interact with the environment. These gestures can be static, dynamic or both. In dynamic gesture recognition, it is required to identify both spatial and temporal movements and this paper proposes a methodology for dynamic gesture recognition.

Sign language is often represented using both hands and in some special occasions using single hands. Figure 1.1 showcases the single handed patterns and the one below is done using both the hands.

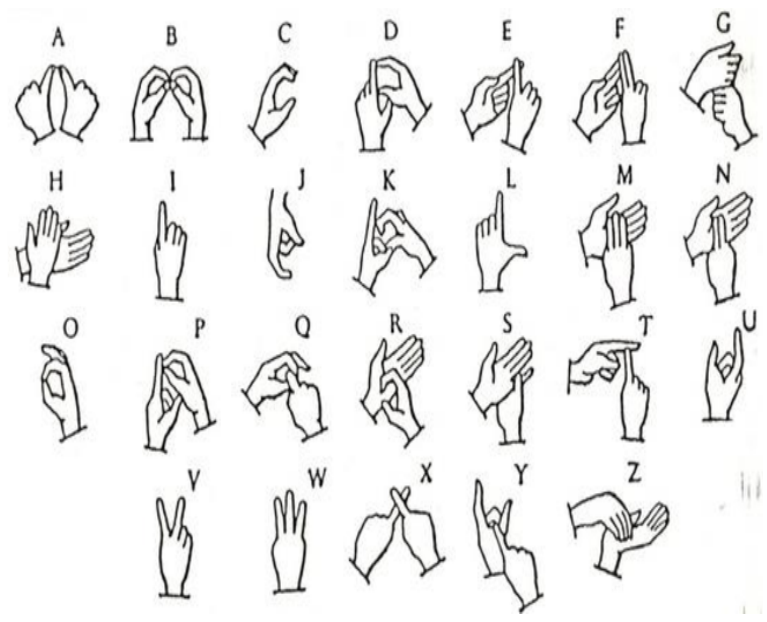


Figure 1.2: ISL two handed English Alphabet notations

In overall, gesture recognition can be divided into two broad areas and those are vision-based methods and methods require special hardware like gloves, armbands and body kits. The second type is comparatively difficult for the user since it is required to carry required hardware when the user needs to use the system. These types of systems might be required for specialized applications like Unmanned Arial Vehicle (UAV) control but not for a general computer user. On the other hand, vision-based systems use techniques like image processing, pattern recognition, clustering, image extraction, image segmentation and object detection. Using vision-based approaches provides a non-obstructive interface for the user and thus created a boost in this area .

1.3 OVERVIEW OF THE PROPOSED SYSTEM

The project presents a robust and efficient method of sign language detection. The final product will include a **mobile application that performs real time sign language translation.** The user focuses the video camera onto the gesture after launching the app. The textview section below displays the corresponding English equivalent

1.4 ORGANISATION OF THE THESIS

The thesis is organized into six chapters. Chapter 2 contains literature survey of existing techniques. Chapter 3 deals with data collection. The requires software and utilities are listed and explained in Chapter 4. The proposed methodology is explained in Chapter 5. Chapter 6 describes conclusion and scope for future work.

CHAPTER 2

LITERATURE SURVEY

There is a standardized sign language namely Indo-Pakistani Sign Language which is practiced in India. These speech or hearing impaired people face a lot of difficult in their day to day life. Speech impaired people heavily rely on speech interpreters for medical, legal, educational and training sessions. There isn't any infrastructure available for speech impaired people to communicate with non-signers without the interpreter. There is not a pathway created for automation of sign language translation. So that's why there is need of automation of sign language translation so which would result in convenient communication between speech impaired people and a non-signer without the need of an interpreter for translation.

In the area of Computer Vision and Pattern Recognition, image segmentation plays an important role as a preliminary step for high level image processing. Segmentation subdivides an image into regions or objects ,one needs to isolate the regions and find relation among them. The process of separation of such objects is referred as image segmentation. . Object tracking is an important task within the field of Computer Vision, video analysis has generated a great deal of interest in object tracking algorithm. There are three key steps in video analysis: detection of moving object, tracking of such moving object and analysis of object tracks to recognize their behavior.

2.1 INDIA'S DISABILITY ESTIMATE : HEARING AND SPEECH

As in most developing countries, India uses the disability estimates from the Census to inform policy and programs for persons with disabilities. In the most recent Census of 2011, disability information became more detailed and important, and included information on a wide range of disabilities. According to this latest Census 2011, 2.2% of the Indian population had disability. Disability data in India are also available from large-scale population surveys such as the National Sample Survey, District-Level Household Survey-4, the Annual Health Survey, and the World Health Survey.

A previous comparison noted wide variations in the disability estimates from the Census 2001 and the National Sample Survey of persons with disabilities conducted in 2002, and the need for more qualitative studies on disability has been argued. As India prepares for the next decennial Census in the context of its commitment to SDGs and CRPD, we aim to provide a review of the disability estimates for India and its states from the Census 2011 and the most recent population-level survey estimates for disability in the current decade. In the background of the above stated need for high-quality data on disability, it is important to understand the variations across the data sources in India in order to provide a consistent and reliable information to formulate and implement policies for people with disability and to monitor trends in the prevalence of disabilities. Therefore, this paper aims to highlight the extent of variation in the disability estimates, and the methodological and definition related issues between the data sources that provide disability estimates for India for all age groups, and to recommend ways in which these variations could be minimized to reliably monitor the trends in disability over time.

Both the data sources considered hearing disability irrespective of the use of hearing aid, and persons with hearing disability in one ear were not considered as having disability in either data source (S1 Table). A person born with hearing disability who was also unable to speak (deaf and mute) was considered as having multiple disabilities in the Census; the survey does not categorically mention anything about persons who were both deaf and mute.

The hearing DR was 125.5% higher in the Census (425.9; 95% CI 425.5–426.3) than the survey (188.9; 95% CI 187.1–190.7, Table 3). Hearing disability accounted for 19.0% and 10.5% of the total disability in the Census and survey, respectively. The difference in hearing DR between the sources ranged from -67% in Nagaland to 436% in Uttar Pradesh (Fig 4 and S2 Table). The hearing DR declined between AHS baseline and round 1 in Uttar Pradesh (18.7%), while it declined between AHS rounds 1 and 2 in Odisha (-30.2%) and Uttarakhand (-13.2%) (Table 4). The increase in hearing DR was more between AHS baseline and round 1 than AHS rounds 1 and 2 in Rajasthan (128.6% vs 11.6%) Madhya Pradesh (16.4% vs 3.5%), and Chhattisgarh (48.3% vs

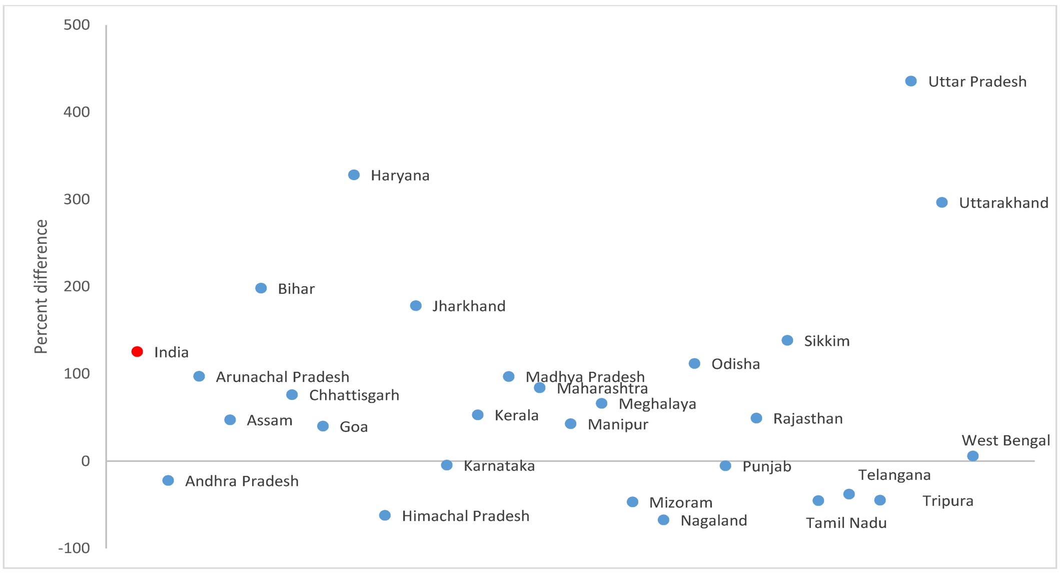


Figure 2.1: Percent difference in hearing disability rate between the Census 2011 and household survey 2012–2013 for each Indian state.

27.0%).

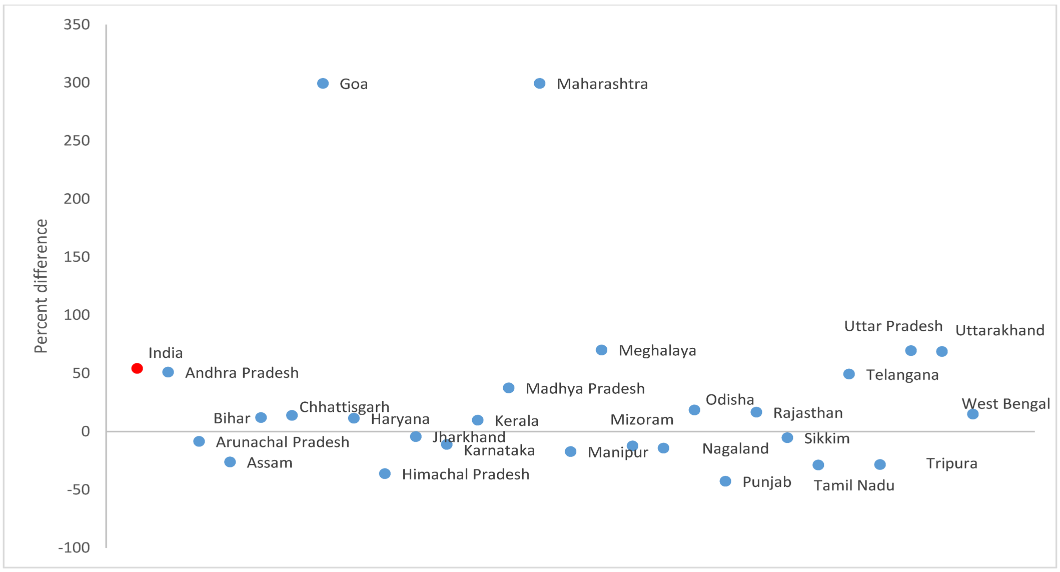


Figure 2.2: Percent difference in speech disability rate between the Census 2011 and household survey 2012–2013 for each Indian state.

Speech disability was assessed for persons aged 3 years and more in both the sources. The definition of speech disability was more or less similar between the two sources with the only difference being that the survey also included articulation defects along with stammering (S1 Table).

The speech DR was higher in the Census 2011 (169.9; 95% CI 169.7–170.1) by 54.2% than the surveys (110.2; 95% CI 108.8–115.5, Table 3). Speech disability accounted for 7.6% and 6.1% of the total disability in the Census and survey, respectively. The difference in speech disability between the sources ranged from -43% in Punjab to 300% in Maharashtra (Fig 6 and S2 Table). The speech DR increased between AHS baseline and round 1 (Table 4) and the range of increase varied substantially from Bihar (13.9%) to Odisha (116.9%). Between the AHS rounds 1 and 2, the speech DR declined in Odisha (-18.9%) and Rajasthan (-14.2%).

2.2 STUDY OF EXISTING SYSTEM AND RESEARCH SOLUTIONS

The existing system for communication between signer and non-signer takes place through an interpreter. The interpreter is well versed with Indo-Pakistani Sign Language. When the signer wants to convey a message to the non-signer he/she makes the appropriate sign language hand gestures to the interpreter and then the interpreter translates the hand gestures made by the signer into equivalent English.

Similarly, the non-signer communicates to signer through the interpreter. First the non-signer tells the interpreter what he/she wants to communicate to the signer then the interpreter through the hand gestures tell the signer. This process is quite tedious and time consuming and the signer may suffer if there is an absence of an interpreter. The existing system requires automation which would help the signers in communicating with the non-signers in absence of an interpreter. The standard sign language Indo-Pakistani can be automated with the help of computer vision, deep learning models which will classify the hand gestures which would aid in the reducing the communication gap between the signers and non-signers.

Some research papers we studied concentrated on improving the speech capability of hearing impaired persons. Some papers suggest an approach for requirement of picture communication system that enables the deaf to communicate over long distances through telephone lines. Some papers proposed a system to aid the hearing and speech impaired for communication with normal persons. These papers provide some beneficial methods to help the hearing and speech impaired persons. But these systems had certain drawbacks like lack of portability or were based on American Standard Sign

Language.

Following are excerpts and comments on some papers that were studied:

2.2.1 Computer-Aided Interpreter for Hearing and Speech Impaired

The proposed project was created with the aim of developing a system to enhance the quality of communication for hearing and speech impaired people. It seeks to establish a two-way communication by means of Human-Computer Interaction (HCI) and ComputerHuman Interaction (CHI).The proposed system is a potential human-computer and computer-human interaction for hearing and speech impaired people with normal people. The proposed device is programmed to perform two basic processes viz., a) recognizing the input voice signal and displaying the corresponding pictorial representation of the sign language gesture, and b) capturing the hand gesture and producing the corresponding voice output. This is achieved using Natural Voice processing and Digital Image Processing algorithms.

The above mentioned system makes use of the American Standard Sign Language, which as discussed earlier is not prevalent in India. Moreover, one of the major drawbacks of this system is lack of portability as it is developed for desktop. A camera and a microphone become the prerequisites for the system to convert voice into gestures and gestures to voice. Thus for the proposed system a complete setup of desktop with a microphone and camera will be required. This implies that it cannot be used for day-to-day communication on the go.

2.2.2 Graphical Speech Training System for Hearing Impaired

Computer aided Speech Training plays a significant role in developing speech and language skills of hearing impaired. The authors include a description of different types of hearing impairments, their causes and quantification; review global status of computer-based speech training system and address the growing need of such a system in India where the ratio of speech therapists to needy users is small, further strengthening our perspective on this issue. This paper also discusses an innovative an interactive graphical speech training system for hearing impaired persons which is cost effective and computer based to help hearing impaired children learn and practice speech skills at home even without the presence of a speech therapist and improve kinesthetic awareness among deaf children enabling them to learn and control the muscular activities of their vocal organs to produce intelligible speech.

The proposed system concentrates on improving the speech of persons of various age groups with partial hearing loss. Consequently, it would not be of use to a completely deaf person. Also even if the person does not regularly practice, then the system will not be of much use. This system is also developed as a desktop application, thus restricting its portability.

2.2.3 Hand Gesture Recognition For Indian Sign Language

1. A hand gesture recognition system to recognize the alphabets of Indian Sign Language
2. 4 Modules
   (a) real time hand tracking,
   (b) hand segmentation
   (c) feature extraction
   (d) gesture recognition.
3. Camshift method and Hue, Saturation, Intensity (HSV) color model are used for hand tracking and segmentation
4. Genetic Algorithm is used for gesture recognition

2.2.4 Sign Quiz

1. A sign language learning tool that uses automatic sign recognition
2. In this proposed system we capture the sign from the user using a camera.
3. The captured photograph is normalized and given to a Deep neural network with an Softmax classifier.
4. Softmax classifier is generally used in multiple class classification problems.
5. The photograph of the sign is detected using the neural network for an interactive sign language learning tool.
6. Results indicate that the signQuiz is better than printed medium for fingerspelled sign learning.

2.2.5 A Modified-LSTM Model for Continuous Sign Language Recognition using Leap motion

continuous sign language recognition system for sentence formation

1. In this paper they used a leap motion sensor to capture the sign.
2. This leap motion sensor will generate a set of points and its x,y,z coordinates
3. Using Feature extraction and normalization the input to the neural network is created in a uniform pattern.
4. A 2D Convolutional neural network is used for Feature extraction.
5. Using this feature maps are created that are fed to LSTM(Long Short Term Memory)
6. In this paper they propose a Modified LSTM that consist of an Reset(R) gate which helps in segmenting the continuous sign sequence as well as in improving the recognition performance.
7. The proposed system has been tested with 942 signed sentences of Indian Sign Language (ISL).
8. These sign sentences are recognized using 35 different sign words.
9. The average accuracy of 72.3% and 89.5% have been recorded on signed sentences and isolated sign words.

2.3 WHAT IS MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

The key difference from traditional computer software is that a human developer hasn't written code that instructs the system how to tell the difference between the banana and the apple.Instead a machine-learning model has been taught how to reliably discriminate between the fruits by being trained on a large amount of data, in this instance likely a huge number of images labelled as containing a banana or an apple.

2.3.1 SUPERVISED LEARNING

Supervised learning means having a full set of labeled data while training an algorithm. Fully labeled means that each example in the training dataset is tagged with the answer the algorithm should come up with on its own. So, a labeled dataset of flower images would tell the model which photos were of roses, daisies and daffodils. When shown a new image, the model compares it to the training examples to predict the correct label.There are two main areas where supervised learning is useful: Classification problems and Regression problems.

Classification problems ask the algorithm to predict a discrete value, identifying the input data as the member of a particular class, or group. In a training dataset of animal images, that would mean each photo was pre-labeled as cat, koala or turtle. The algorithm is then evaluated by how accurately it can correctly classify new images of other koalas and turtles.

On the other hand, regression problems look at continuous data. One use case, linear regression, should sound familiar from algebra class: given a particular x value, what's the expected value of the y variable?

2.3.2 UNSUPERVISED LEARNING

In unsupervised learning, a deep learning model is handed a dataset without explicit instructions on what to do with it. The training dataset is a collection of examples without a specific desired outcome or correct answer. The neural network then attempts to automatically find structure in the data by extracting useful features and analyzing its structure. Depending on the problem at hand, the unsupervised learning model can organize the data in different ways like clustering, anomaly detection, association and autoencoders.

2.3.3 REINFORCEMENT LEARNING

Reinforcement learning operates on the principle of gaming— and actually, video games are a common test environment for this kind of research. In this kind of machine learning, AI agents are attempting to find the optimal way to accomplish a particular goal, or improve performance on a specific task. As the agent takes action that goes toward the goal, it receives a reward. The overall aim: predict the best next step to take to earn the biggest final reward.

To make its choices, the agent relies both on learnings from past feedback and exploration of new tactics that may present a larger payoff. This involves a long-term strategy — just as the best immediate move in a chess game may not help you win in the long run, the agent tries to maximize the cumulative reward.

It's an iterative process: the more rounds of feedback, the better the agent's strategy becomes. This technique is especially useful for training robots, which make a series of decisions in tasks like steering an autonomous vehicle or managing inventory in a warehouse.

2.4 WHAT IS DEEP LEARNING

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

It utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

- Deep learning is an AI function that mimics the workings of the human brain in processing data for use in decision making.
- Deep learning AI is able to learn from data that is both unstructured and unla-

beled.

- Deep learning, a machine learning subset, can be used to help detect fraud or money laundering.
- Deep learning learns from vast amounts of unstructured data that could normally take humans decades to understand and process.

2.5 CONVOLUTIONAL NEURAL NETWORK(CNN)

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance(learnable weights and biases)to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While inprimitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.The CNN algorithm is trained rather than programmed so the technique does not require manual feature extraction.CNN automatically learns the features required to classify the images.

deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

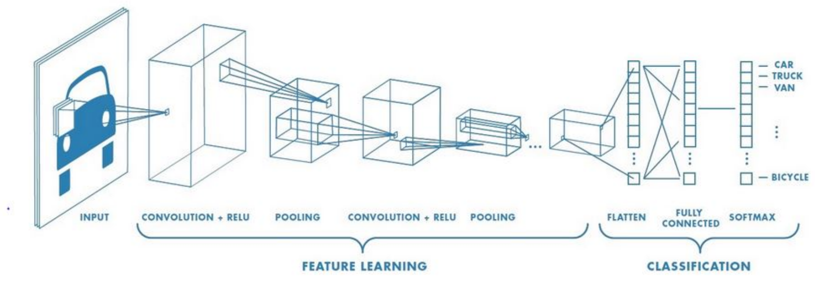


Figure 2.3: Neural network with many convolutional layers

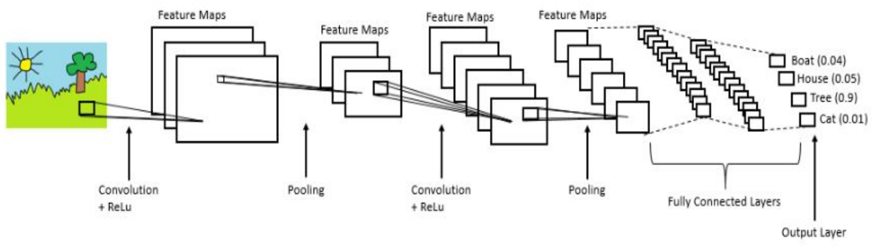


Figure 2.4: Complete CNN architecture

CONVOLUTION LAYER

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. Then the convolution of k x k image matrix multiplies with 3 x 3 filter matrix which is called “Feature Map” .

POOLING LAYER

The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On

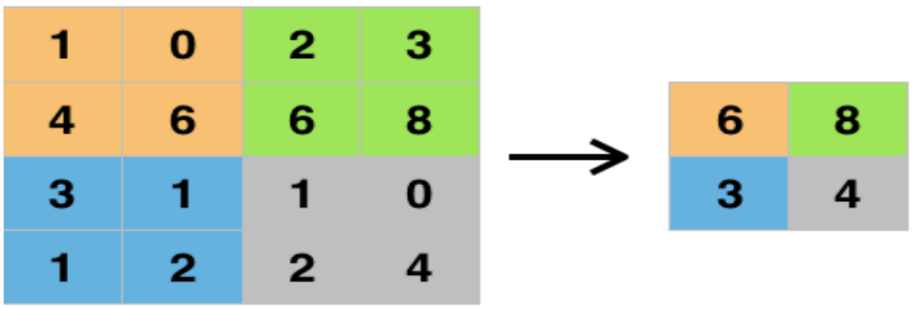


Figure 2.5: Max Pooling

the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Max pooling extracts the most important features like edges whereas average pooling extracts features so smoothly. Although both are used for same reason, max pooling is better for extracting the extreme features. Average pooling sometimes can't extract good features because it takes all into count and results an average value which may or may not be important for object detection type tasks. If don't need all inputs from Convolutional layer, will get bad accuracy for average pooling.

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

FULLY CONNECTED LAYER

The layer we call as fully connected layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network. In the above diagram, the feature map

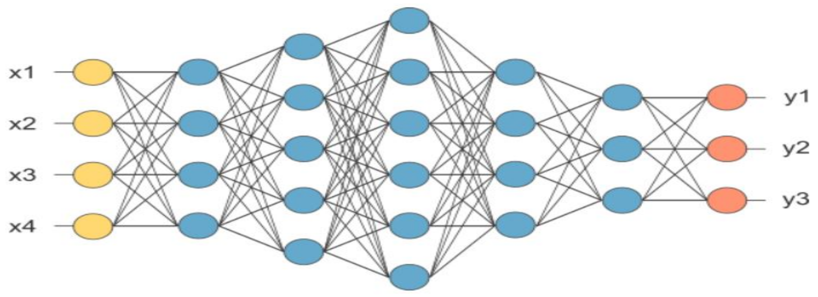


Figure 2.6: After pooling layer, flattened as FC layer

matrix will be converted as vector (x1, x2, x3, . . .). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

2.6 TRANSFER LEARNING USING MOBILENETS AND KERAS

Convolutional neural networks have become omnipresent in computer vision ever since AlexNet popularized deep convolutional neural networks by winning the ImageNet Challenge: ILSVRC 2012. The general trend has been to make deeper and more complicated networks in order to achieve higher accuracy .However, these advances to improve accuracy are not necessarily making networks more efficient with respect to size and speed. In many real world applications such as robotics, self-driving car and augmented reality, the recognition tasks need to be carried out in a timely fashion on a computationally limited platform.

2.6.1 Mobilenet model: Depthwise Separable Convolutions

The MobileNet model is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution. For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size.

MobileNets are built primarily from depthwise separable convolutions initially introduced in and subsequently used in Inception models to reduce the computation in the first few layers. Flattened networks build a network out of fully factorized convolutions and showed the potential of extremely factorized networks. Independent of this current paper, Factorized Networks introduces a similar factorized convolution as well as the use of topological connections. Subsequently, the Xception network demonstrated how to scale up depthwise separable filters to out perform Inception V3 networks. Another small network is Squeezenet which uses a bottleneck approach to design a very small network. Other reduced computation networks include structured transform networks and deep fried convnets

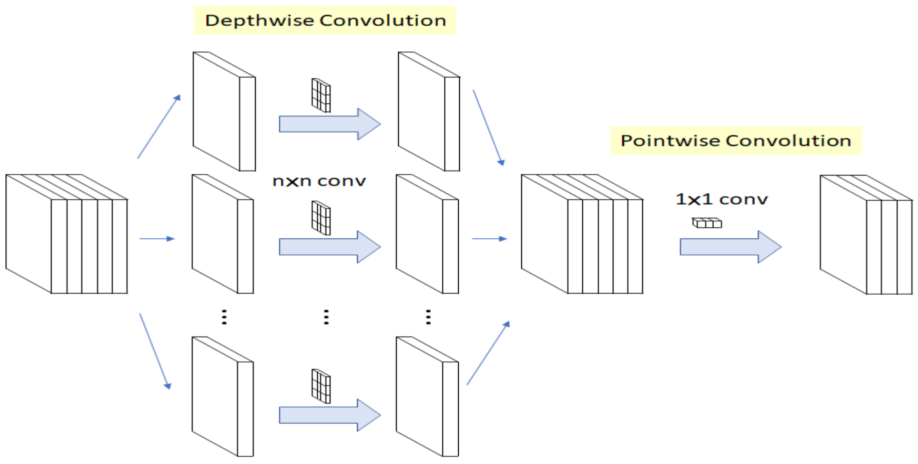


Figure 2.7

Depthwise convolution is the channel-wise DKxDK spatial convolution. Suppose in the figure above, we have 5 channels, then we will have 5 DKxDK spatial convolution. Pointwise convolution actually is the 1x1 convolution to change the dimension.

$$operation - cost = D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f$$

where the computational cost depends multiplicatively on the number of input channels M, the number of output channels N the kernel size Dk x Dk and the feature map size DF x DF .

MobileNet models address each of these terms and their interactions. First it uses depthwise separable convolutions to break the interaction between the number of output channels and the size of the kernel. The standard convolution operation has the effect of filtering features based on the convolutional kernels and combining features in order to produce a new representation. The filtering and combination steps can be split into two steps via the use of factorized convolutions called depthwise separable convolutions for substantial reduction in computational cost

$$Standard - Convolution - cost = D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f$$

By expressing convolution as a two step process of filtering and combining we get a reduction in computation of:

$$\frac{D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f}{D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f} = \frac{1}{N} + \frac{1}{D_k^2}$$

MobileNet uses 3 × 3 depthwise separable convolutions which uses between 8 to 9 times less computation than standard convolutions at only a small reduction in accuracy

2.6.2 Keras

Keras is a high-level API to build and train deep learning models. It is used for fast prototyping, advanced research, and production, with three key advantages:

1. User friendly : Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
2. Modular and composable : Keras models are made by connecting configurable building blocks together, with few restrictions.
3. Easy to extend: Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models.

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. 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 (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is FranÃ gois Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library.Chollet explained that Keras was conceived to be an interface rather than a standalone machine-learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used.Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0.

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.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).

2.7 TENSORFLOW

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google 's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

A couple of years ago, deep learning started to outperform all other machine learning algorithms when giving a massive amount of data. Google saw it could use these deep neural networks to improve its services like Gmail, photos and Google search engine.They build a framework called Tensorflow to let researchers and developers work together on an AI model. Once developed and scaled, it allows lots of people to use it.

It was first made public in late 2015, while the first stable version appeared in 2017. It is open source under Apache Open Source license. You can use it, modify it and redistribute the modified version for a fee without paying anything to Google. Tensorflow architecture works in three parts:

1. Preprocessing the data
2. Build the model
3. Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors.We can construct a sort of flowchart of operations (called a Graph) that we want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the operations are conducted inside a graph.The graph is a set of computation that takes place successively. Each operation is called an op node and are connected to each other.The graph outlines the ops and connections between the nodes. However, it does not display the values.The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

TensorFlow makes use of a graph framework. The graph gathers and describes all the series computations done during the training. The graph has lots of advantages:

- It was done to run on multiple CPUs or GPUs and even mobile operating system
- The portability of the graph allows to preserve the computations for immediate or later use. The graph can be saved to be executed in the future
- All the computations in the graph are done by connecting tensors together

A tensor has a node and an edge.The node carries the mathematical operation and produces an endpoints outputs.The edges the edges explain the input/output relationships between nodes.

2.7.1 Training a Model In Tensorflow

Training the neural network model requires the following steps:

1. 1. Feed the training data to the model - the images and labels arrays.
2. The model learns to associate images and labels.
3. We ask the model to make predictions about a test set - the test images array. We verify that the predictions match the labels from the test labels array.
4. To start training, call the model.fit method in tensorflow - the model is "fit" to the training data.

2.8 STUDY OF EXISTING APPLICATIONS IN GOOGLE PLAYSTORE

A few philanthropic developers have attempted to solve the problem faced by deaf and mute persons by developing text to audio converters, sign language interpreters and standard signs guides to name a few. However, our study of the existing solutions revealed that none of them provided a complete solution to the problem. While one application takes input as text and produces audio as output, another simply shows the corresponding sign for the entered text. There is no integration of all the features required for a conversation in one single application. Moreover, all the existing solutions use American Standard Sign Language which is not followed in Indian deaf schools, as per our survey.

S.No	Application Name	Cons
1	Virtual Voice	Needs external software support
2	Note Speak	Slow
3	Sign Language Interpreter	Uses ASL only
4	Sign Short Message Service	-No support for ISL

Table 2.1: Existing Applications on Google Play Store

Other papers and references are added in the reference page

CHAPTER 3

SOFTWARE REQUIREMENTS

The necessary software required for data collection, processing and deploying the final product is listed and explained below.

3.1 ANDROID STUDIO

Android Studio is a perfect option for front-end designing. The application would consist of a video camera and a textview that translates the sign that is captured using the camera

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.

Android is free and an open platform built on Linux. Android Application Development Company is also an open source solution for mobile devices offering a complete software stack including operating system, middle ware and key mobile applications. User acceptance to Android was very low when it was launched in 2007, as it was still in its early development cycle. But after Google's acquisition and development efforts, visibility for Android mobile technology grew.This is the reason why it is competing against Apple and other popular Smartphone operating system. The demand for Android app development grew through its robust offerings with many new android devices.

Android Studio is Android's official IDE. It is purpose-built for Android to accelerate your development and help you build the highest-quality apps for every Android device.

APPLY CHANGES

Android Studio's Apply Changes feature lets you push code and resource changes to your running app without restarting your app—and, in some cases, without restarting the current activity. This flexibility helps you control how much of your app is restarted when you want to deploy and test small, incremental changes while preserving your device's current state.

INTELLIGENT CODE EDITOR

The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code.

FAST AND FEATURE-RICH EMULATOR

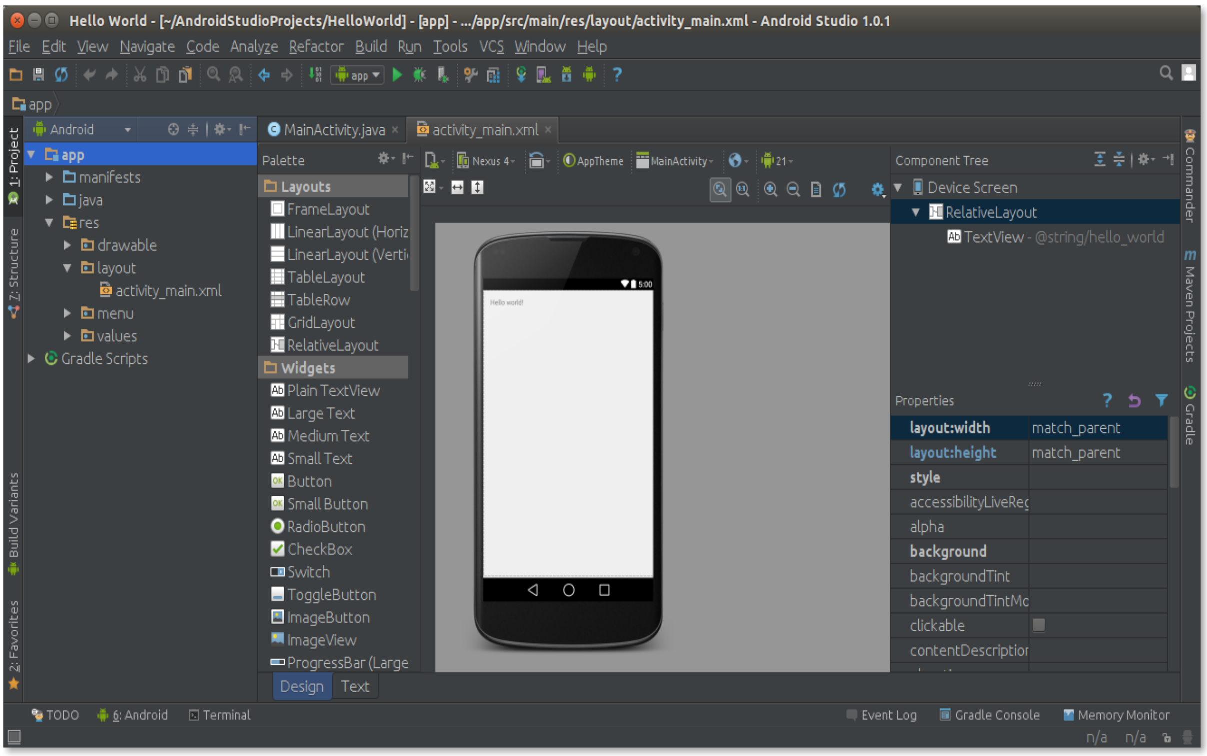
The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations: phones, tablets, Android Wear, and Android TV devices. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

DESIGNED FOR TEAMS

Android Studio integrates with version control tools, such as GitHub and Subversion, so you can keep your team in sync with project and build changes. The open source Gradle build system allows you to tailor the build to your environment and run on a continuous integration server such as Jenkins.

OPTIMISED FOR ALL ANDROID DEVICES

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code mod-



layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
backgroundTint	
backgroundTintMo	
clickable	☐
contentDescriptior	

Figure 3.1: Android Studio Editor And Emulator

ules allow you to divide your project into units of functionality that you can independently build, test, and debug.

3.2 TENSORFLOW LITE

TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

TensorFlow Lite consists of two main components:

- The TensorFlow Lite interpreter, which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
- The TensorFlow Lite converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

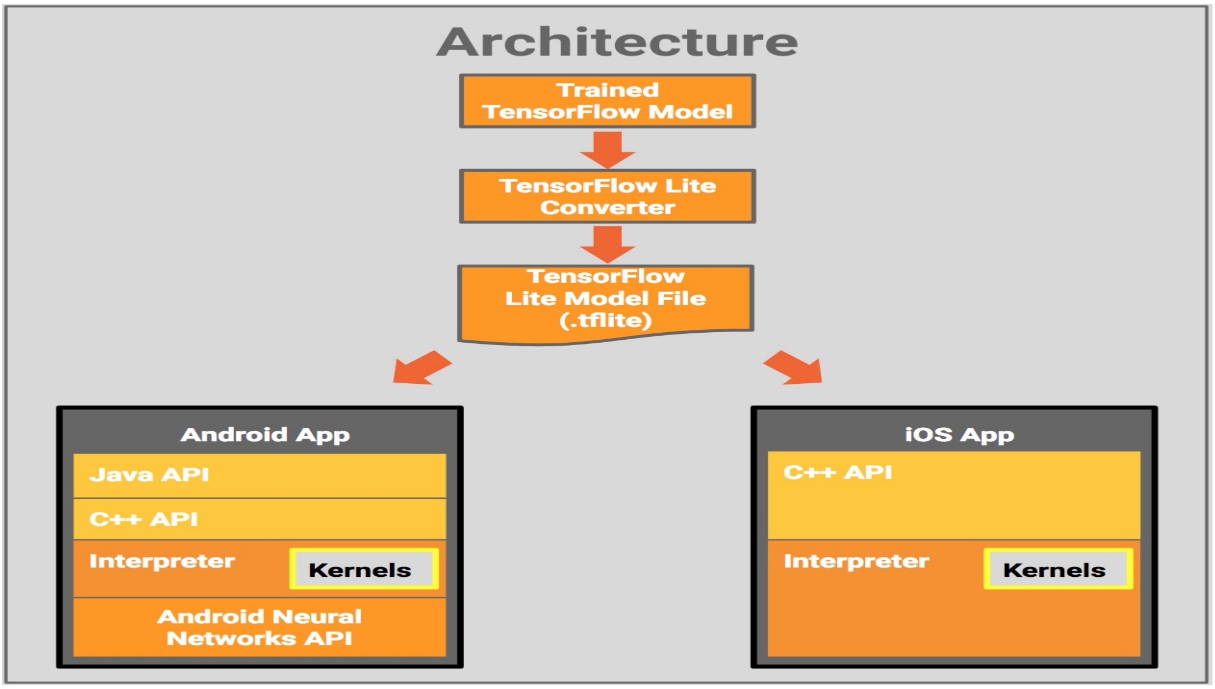


Figure 3.2: Tensorflow Lite Architecture

3.3 OPENCV

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license. (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

OpenCV's application areas include:

1. 2D and 3D feature toolkits
2. Egomotion estimation
3. Facial recognition system
4. Gesture recognition
5. Human–computer interaction (HCI)
6. Mobile robotics
7. Motion understanding
8. Object identification

9. Segmentation and recognition
10. Stereopsis stereo vision: depth perception from 2 cameras
11. Structure from motion (SFM)
12. Motion tracking
13. Augmented reality

We use opencv libraries for image pre-processing and image processing which includes gray-scaling, binary conversion, background subtraction, blurring etc.

CHAPTER 4

METHODOLOGY

Following methodology represents how computer vision based techniques, as well as CNN based classification techniques, have been used to develop a real-time dynamic gesture recognition system. The system is developed to identify four gestures (swipe left, swipe right, swipe down, shrink) as illustrated in Fig. 6 and it is possible to use the same approach to identify more gestures easily. Implementations of this approach have been done using python programming language and to ease the life of image processing, OpenCV (Open Source Computer Vision Library), which was initially developed by Intel has been used. For the machine learning tasks and to retain efficiency in large matrix manipulation tasks, python scikit-learn and NumPy libraries have been used. Images are captured using the inbuilt 2-5mp camera of laptops in 29 frames per second with an original resolution of 1280 x 720 and later reduced to ease computation. Prior to creating motion history image from captured frames, each frame is preprocessed using multiple image processing steps. Following are those steps and rationale for them.

4.1 DATA COLLECTION

The data required for training includes 26 letters of English lexicon, 0-9 digits, 45+ gestures. Signs without motion and with motion are collected separately. Sample data was collected with the help of our classmates and family members.

4.1.1 WHY NOT INSTRUMENTED GLOVES?

While designing the glove there are some circuits and some sensors are fixed on glove which detects every single motion made by hand or by even fingers. These instrumented gloves design and sensors used in it can be differ according to use or by companies made it. That's why it is not necessary every glove work same as other. In this glove there are some sensor whose working is Light based. In light based a tube is used which is flexible so that it can cooperate with hand movement in placed at one side and on the other side photocell is used. As we move or make movement by finger the light that strike on photocell changed, by this finger flexion could be measured. While using Instrumented gloved there are many difficulties faced by user like user have to always wear that glove while collecting the hand gestures. It is very difficult to give same posture same again and again because angle of hands and finger could be vary each time which make input gestures two different gestures.

4.1.2 VISION BAESD APPROACH

On time of collecting the data the user is not allowed to make movement for some amount of time until input is not recorded correctly. So, Computer Vision Based (19) came into existence. To collect data by using Vision Based Approach for Hand gesture and movement there are four constituent which make system as whole: Number of camera used and where to place them is first challenge in this approach. As we have to place cameras such that each and every movement of hand could be register. Main problem that encounter in this type of approach that hand movement is not visible by cameras and they also fail to record hand movement so this is primary concern in this method.

For Data Collection, We aimed at around 15 students of different skin tones and made around 6-11 seconds of video for every alphabet per person using a 30fps camera summing upto 1 minute video for every alphabet. While making the video the camera shall be tilted accordingly to obtain images in different positions within the frame. We would ask the students to show the most commonly used static representation for every alphabet(in case of multiple representations). We shall try to make the background as light as possible to make image segmentation easier. Later these videos were be converted into frames evaluating to 1800 - 2000 images per alphabet.

Methods	Glove	Vision
Price	High	Less
Alleviation for user	Less	High
Hand shape	To be the same	Can differ
Calculation	Complex	Less complex
Efficiency	Less	High

Table 4.1: Comparison of different approaches

4.2 DATA PREPARATION IN DETAIL

Python image processing packages - OpenCV, imutils, Scikit-image - were used to convert captured video into frames, collect and preprocess the images. The aim of preprocessing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

```
1 import cv2
2 import imutils
3 import numpy as np
4 from sklearn.metrics import pairwise
```

Line 1 here imports cv2 module and functions wrapped in the OpenCV package. The next line imports the imutils package which is a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3. Line 3 imprts numpy library. . It provides fast and efficient operations on arrays of homogeneous data. NumPy extends python into a high-level language for manipulating numerical data, similiar to MATLAB.

The next line imports pairwise function from scikit-learn library's metrics module. Scikit features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The sklearn.metrics.pairwise submodule implements utilities to evaluate pairwise distances or affinity of sets of samples.

4.2.1 BACKGROUND SUBTRACTION

It is crucial to identify moving objects of the scene since moving gestures are going to be identified. Z. Zivkovic suggested a method for this and it is based on Gaussian Mixture-based Background/Foreground Segmentation where it analyzes each pixel and he has developed an effective adaptive algorithm using Gaussian mixture probability density.Zivkovic proposed another improved version in 2006 and these two papers are based on the algorithm which has been used to subtract the background. This algorithm is implemented in OpenCV library and it can detect shadows as well. Fig 4.1 shows a frame background removed with a moving hand.

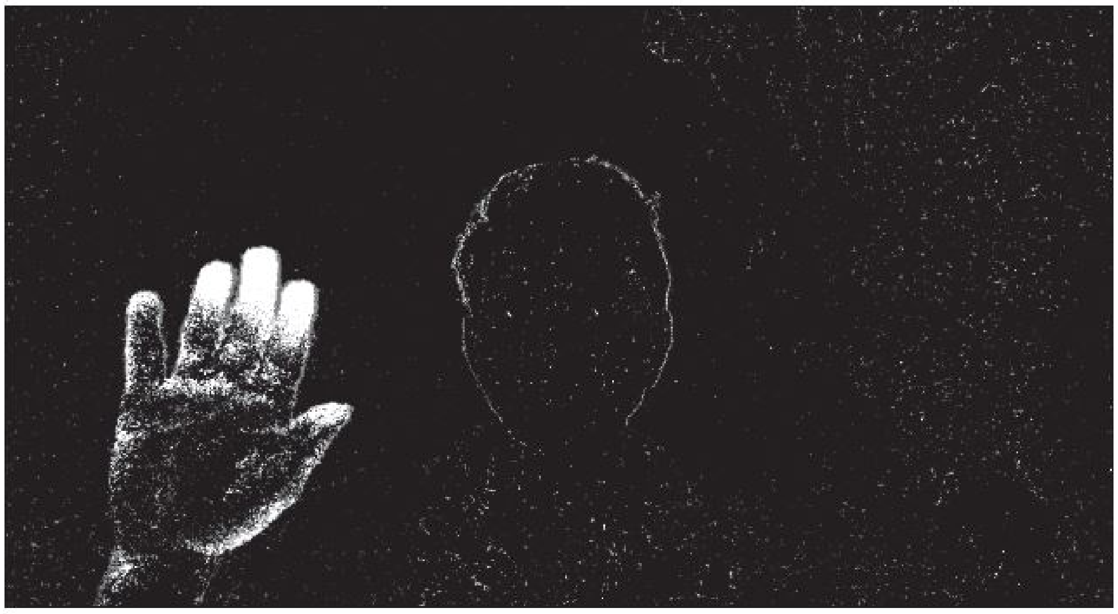


Figure 4.1: Background Subtracted Image

The **Running Average** of a function is used to separate foreground from background. In this concept, the video sequence is analyzed over a particular set of frames. During this sequence of frames, the running average over the current frame and the previous frames is computed. This gives us the background model and any new object introduced in the during the sequencing of the video becomes the part of the foreground. Then, the current frame holds the newly introduced object with the background. Then the computation of the absolute difference between the background model (which is a function of time) and the current frame (which is newly introduced object) is done.

Running average is computed using the equation given below :

$$dst(x, y = (1 - alpha) \cdot dst(x, y) + alpha.src(x, y)$$

The objective of the program is to detect active objects from the difference obtained from the reference frame and the current frame. We keep feeding each frame to the given function, and the function keeps finding the averages of all frames. Then we compute the absolute difference between the frames. The function used is

```
cv2.accumulateWeighted(src, dst, alpha)
```

The parameters passed in this function are :

1. src: The source image. The image can be colored or grayscaled image and either 8-bit or 32-bit floating point.
2. dst: The accumulator or the destination image. It is either 32-bit or 64-bit floating point. NOTE: It should have the same channels as that of the source image. Also, the value of dst should be predeclared initially.
3. alpha: Weight of the input image. Alpha decides the speed of updating. If you set a lower value for this variable, running average will be performed over a larger amount of previous frames and vice-versa.

4.2.2 BINARY THRESHOLDING (IMAGE SEGMENTATION)

From Fig. 4.2 it is observable that the pixels near the wrist are not black and white (Gray pixels). In order to convert those areas to white and to get a uniform binary image, binary thresholding is applied. In addition, it is required to invert the colors to get a black hand on a white background. Figure 3 shows how aforementioned changes have been applied. In general thresholding, it is required to provide a global threshold value. But with the different illumination conditions, determining this value can be difficult. Therefore, Otsu's binarization has been used and it will calculate a suitable threshold value by analyzing image histogram.

In Otsu Thresholding, a value of the threshold isn't chosen but is determined automatically. A bimodal image (two distinct image values) is considered. The histogram

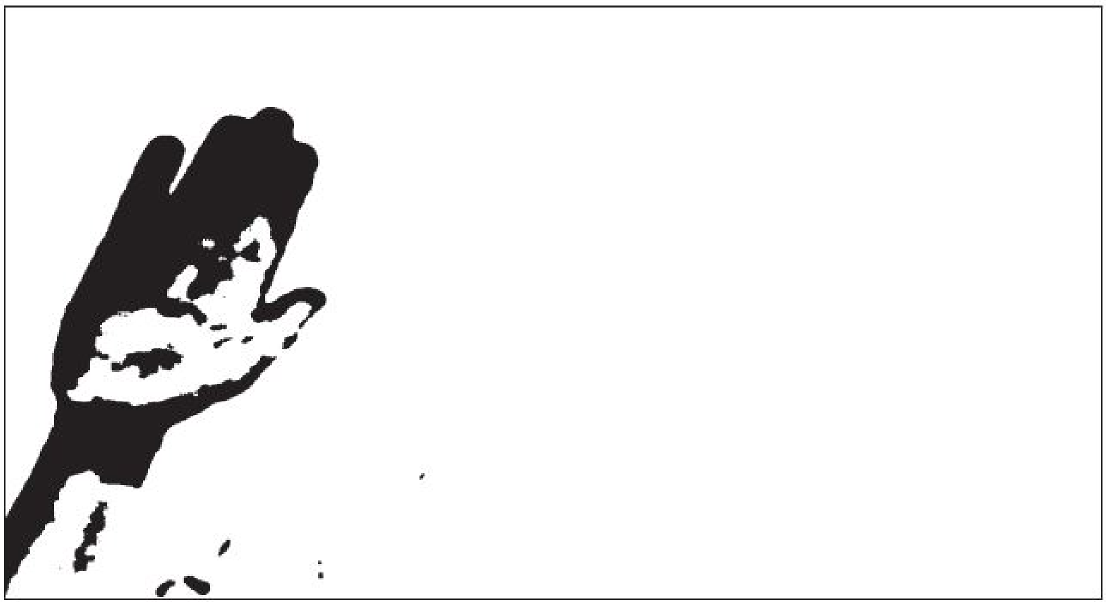


Figure 4.2: After Applying Binary Thresholding

generated contains two peaks. So, a generic condition would be to choose a threshold value that lies in the middle of both the histogram peak values. we use:

```
thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)
```

Syntax: cv2.threshold(source, thresholdValue, maxVal, thresholdingTechnique)
Parameters:
-> source: Input Image array (must be in Grayscale).
-> thresholdValue: Value of Threshold below and above which pixel values will change accordingly.
-> maxVal: Maximum value that can be assigned to a pixel.
-> thresholdingTechnique: The type of thresholding to be applied.

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come handy in shape analysis, finding the size of the object of interest, and object detection.

```
cv2.findContours(thresholded.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

OpenCV has findContours() function that helps in extracting the contours from the image. It works best on binary images.

4.2.3 EXTRACTING FRAMES

OpenCV provides the VideoCature() function which is used to work with the Camera. The following tasks can be done using OpenCV:

- Read video, display video, and save video.
- Capture from the camera and display it.

```
camera = cv2.VideoCapture(0)
```

A VideoCapture object is to be created inorder to capture a video. It accepts either the device index or the name of a video file. A number which is specifying to the camera is called device index. The camera is selected by passing O or 1 as an argument. After that we can capture the video frame-by-frame. We can play the video from the file. It is similar to capturing from the camera by changing the camera index with the file name. for eg;

```
camera = cv2.VideoCapture("F:\\Btech\\Beethoven\\DATACOLLECTION.mp4")
```

Set up an infinite while loop and use the read() method to read the frames using the above created object. camera.read() returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value.

```
    while(True):
        (grabbed, frame) = camera.read()
```

here grabbed is a boolean variable that returns true if the frame is available.If there is no frame, you wont get an error, you will get None.

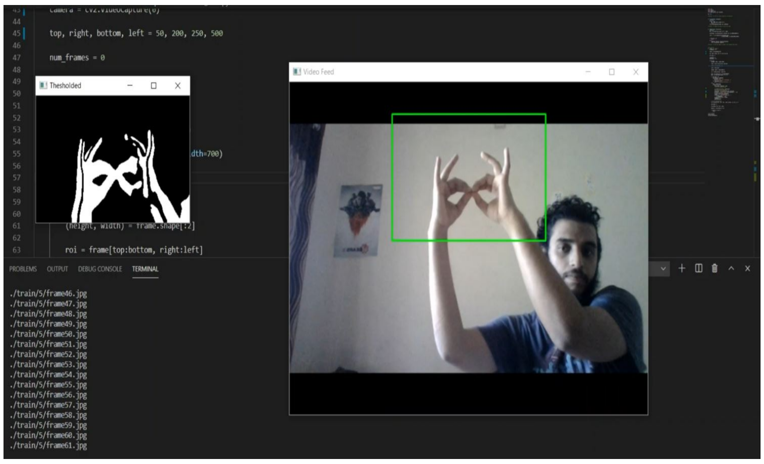
A color space plays an important role in color image processing and color vision applications. While compressing images/videos, properties of the human visual system are used to remove image details unperceivable by the human eye, appropriately called psychovisual redundancies. For color conversion, we use the function cv2.cvtColor(inputimage, flag) where flag determines the type of conversion. For BGR to Gray conversion we use the flags cv2.COLOR-BGR2GRAY

```
gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
```

cv2.imshow() method is used to display an image in a window, so that we can see the frames that we are capturing. The window automatically fits to the image size. cv2.imwrite() method is used to save an image to any storage device. This will save the image according to the specified format in current working directory. This, put in a loop, is our dataset.

```
# show the thresholded image
cv2.imshow("Thesholded", thresholded)
# Set the directory CORRECTLY
directory = './A/frame' + str(imageNumber) + '.jpg'
cv2.imwrite(directory, thresholded)
```

This dataset was prepared as shown below. Each time we collected a single letter or a digit, we changed the directory where the data is to be saved. To improve data and reduce redundancy and unwanted data, we constructed a rectangular frame setting coordinates and set the capture if the video detects variation in that rectangle and hence we can correctly focus our hands better. The following image shows us collecting data.



```
    camera = cv2.VideoCapture(0)

    top, right, bottom, left = 50, 200, 250, 500

    num_frames = 0

dth=700)

        (height, width) = frame.shape[:2]

        roi = frame[top:bottom, right:left]
```

```
./train/5/frame46.jpg
./train/5/frame47.jpg
./train/5/frame48.jpg
./train/5/frame49.jpg
./train/5/frame50.jpg
./train/5/frame51.jpg
./train/5/frame52.jpg
./train/5/frame53.jpg
./train/5/frame54.jpg
./train/5/frame55.jpg
./train/5/frame56.jpg
./train/5/frame57.jpg
./train/5/frame58.jpg
./train/5/frame59.jpg
./train/5/frame60.jpg
./train/5/frame61.jpg
```

Figure 4.3: Beethoven programmer collecting data of letter ’B’

(a) A (b) name

Figure 4.4

The images shown in Figure 4.4(a) and Figure 4.4(b) represents data that are labelled as A and "name" respectively. Those are signs for the foretold.

4.3 TRAINING MODEL

After collecting dataset, pre processing them, and labelling the dataset, provide it as input to tensorflow methods. Tensorflow takes care of the feature extraction process and creating labelled models.

```
import tensorflow as tf
import os
import numpy as np
import matplotlib.pyplot as plt
```

This imports tensorflow, numpy, matplotlib, and os libraries we need.

4.3.1 TRANSFER LEARNING

Feature extraction involves reducing the amount of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause a classification algorithm to overfit to training samples and generalize poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. In our system, the CNN does automatic feature extraction from the provided dataset. CNN learns general features in the first few layer and in deeper layers it extracts more robust features.

Feature extraction is an essential process for image data dimensionality reduction and classification. However, feature extraction is very difficult and often requires human intervention. In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and completely describing the original data set.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection.The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. There is a stark difference between the traditional approach of building and training machine learning models, and using a methodology following transfer learning principles.Traditional learning is isolated and occurs purely based on specific tasks, datasets and training separate isolated models on them. No knowledge is retained which can be transferred from one model to another. In transfer learning, you can leverage knowledge (features, weights etc) from previously trained models for training newer models and even tackle problems like having less data for the newer task.

4.3.2 TRAIN DATA GENERATOR

In order to train a model from scratch or even transfer learning from a pre-trained model, the model is fed by the data for either training or transfer learning. The data is firstly loaded into the machine RAM and then fed to the model. Unfortunately, some data might be very large to be loaded into the memory at once and this is a reason for the memory overflow. To overcome this issue, the data is not fed to the model at once but in batches where each batch contains a pre-defined number of images.

```
IMAGE_SIZE =96
BATCH_SIZE=64

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    validation_split=0.2,
)
```

The batch size is set as 64. This means that just 64 images will be fed to the model at a time. The data has 38126 samples, then dividing the number of samples by the batch size returns 596. This means that the entire training data will be fed into the model in 596 steps. All of these steps represent a single epoch for training the model. So, a single epoch, in this case, has 596 steps.

Keras supports a class named ImageDataGenerator for generating batches of tensor image data. It can also do real-time data augmentation. The validation_nsplit=0.2 will cause that 20% of the training data will be used for validation.

After building an instance of the ImageDataGenerator class and being able to specify the parameters to that generator inside the flow_from_directory() method, here is the complete code for building the data generator for the training data. The generator saved in the train_generator variable will be used later when transfer learning of MobileNet.

```
train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE,
    subset = 'training',
)
```

After preparing the training data generator, we should build the validation data generator.

4.3.3 VALIDATION DATA GENERATOR

The code used for this purpose is listed below. At first, the variable assigned to the directory argument of the flow_from_directory() method is now validation_dir rather than train_dir. The value of the validation_dir variable is created by joining the dataset directory saved in the base_dir variable to the word Test for loading the test data for model validation. The validation data generator is finally saved in the validation_generator variable. It will be used later when transfer learning of MobileNet

```
val_generator = datagen.flow_from_directory(
    base_dir,
    target_size =(IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE,
    subset = 'validation',
)
```

At this time, both the train and validation data generators are created.

4.3.4 LOADING MOBILENET

TensorFlow has a module tensorflow.keras.applications which holds a number of pre-trained deep learning models. When loading any of such models, not only their architectures are existing but also their trained weights. So, you are ready to either use them for making predictions or transfer learning. When a model is loaded for making predictions, then the entire model is loaded including the last fully connected (FC) layers. For transfer learning, these layers are not included. If a model is trained by a given dataset, then the last FC layers will have a number of neurons equal to the number of classes within this dataset.

```
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
base_model.trainable = False
```

To inform the network that it will not be retrained, the trainable parameter of the loaded model is set to False. This indicates that no layer will be trained.

Now, there are 0 trainable parameters and all other parameters are non-trainable. As a result, the values of all of these pre-trained parameters will be used.

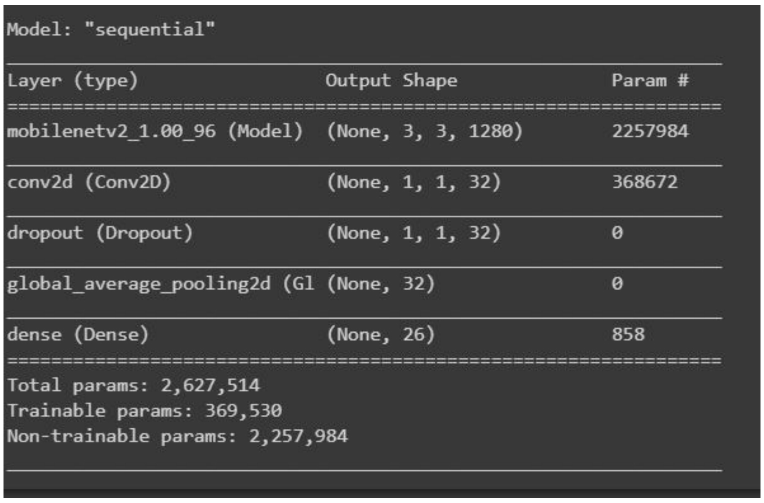
Transfer learning adapts the previously trained parameters to the new dataset. The more adaptation the more accurate the model will be in making predictions for the new dataset. Generally, the more layers to be adapted in transfer learning the more accurate predictions the model will make for the new dataset.

Before training the model we must specify **Optimizer for training the model and learning rate, Loss function, Metrics**

```
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

1. optimizer: Accepts the optimizer to be used. While creating the optimizer, the learning rate can be specified.
2. loss: Loss function. The binary crossentropy loss function is used.
3. metrics: Metrics to be evaluated while training and testing the model. Because we are working on a classification project, then just the accuracy is used.

The output of the model.summary() is as in Figure



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_96 (Model)  (None, 3, 3, 1280)        2257984
_________________________________________________________________
conv2d (Conv2D)              (None, 1, 1, 32)          368672
_________________________________________________________________
dropout (Dropout)            (None, 1, 1, 32)          0
_________________________________________________________________
global_average_pooling2d (Gl (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 26)                858
=================================================================
Total params: 2,627,514
Trainable params: 369,530
Non-trainable params: 2,257,984
_________________________________________________________________
```

Figure 4.5: model.summary()

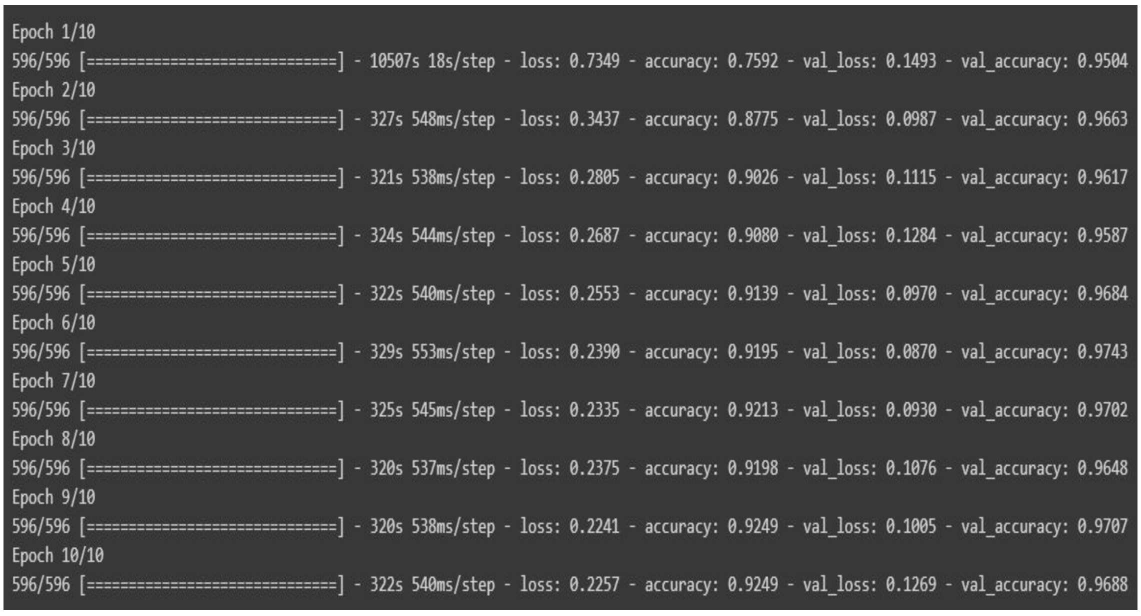
According to Figure 4.5, there are 369,530 trainable parameters which are the parameters of the newly compiled model. Note that the pooling layers have no trainable parameters. The number of non-trainable parameters is 2,257,984.

In order to train a model, the fit_generator() method is used. From the arguments accepted by this method, the following are used.

- generator: Train generator.
- steps_per_epoch: Always set to ceil(number of training samples / batch size).
- epochs: Number of training epochs.
- validation_data: Validation generator.
- validation_steps: Could be calculated as ceil(number of validation samples / batch size)

```
epochs = 10

history = model.fit(train_generator,
                    steps_per_epoch=len(train_generator),
                    epochs=epochs,
                    validation_data=val_generator,
                    validation_steps=len(val_generator))
```



```
Epoch 1/10
596/596 [==============================] - 10507s 18s/step - loss: 0.7349 - accuracy: 0.7592 - val_loss: 0.1493 - val_accuracy: 0.9504
Epoch 2/10
596/596 [==============================] - 327s 548ms/step - loss: 0.3437 - accuracy: 0.8775 - val_loss: 0.0987 - val_accuracy: 0.9663
Epoch 3/10
596/596 [==============================] - 321s 538ms/step - loss: 0.2805 - accuracy: 0.9026 - val_loss: 0.1115 - val_accuracy: 0.9617
Epoch 4/10
596/596 [==============================] - 324s 544ms/step - loss: 0.2687 - accuracy: 0.9080 - val_loss: 0.1284 - val_accuracy: 0.9587
Epoch 5/10
596/596 [==============================] - 322s 540ms/step - loss: 0.2553 - accuracy: 0.9139 - val_loss: 0.0970 - val_accuracy: 0.9684
Epoch 6/10
596/596 [==============================] - 329s 553ms/step - loss: 0.2390 - accuracy: 0.9195 - val_loss: 0.0870 - val_accuracy: 0.9743
Epoch 7/10
596/596 [==============================] - 325s 545ms/step - loss: 0.2335 - accuracy: 0.9213 - val_loss: 0.0930 - val_accuracy: 0.9702
Epoch 8/10
596/596 [==============================] - 320s 537ms/step - loss: 0.2375 - accuracy: 0.9198 - val_loss: 0.1076 - val_accuracy: 0.9648
Epoch 9/10
596/596 [==============================] - 320s 538ms/step - loss: 0.2241 - accuracy: 0.9249 - val_loss: 0.1005 - val_accuracy: 0.9707
Epoch 10/10
596/596 [==============================] - 322s 540ms/step - loss: 0.2257 - accuracy: 0.9249 - val_loss: 0.1269 - val_accuracy: 0.9688
```

Figure 4.6: Training data

4.4 MACHINE LEARNING ON MOBILE PLATFORMS

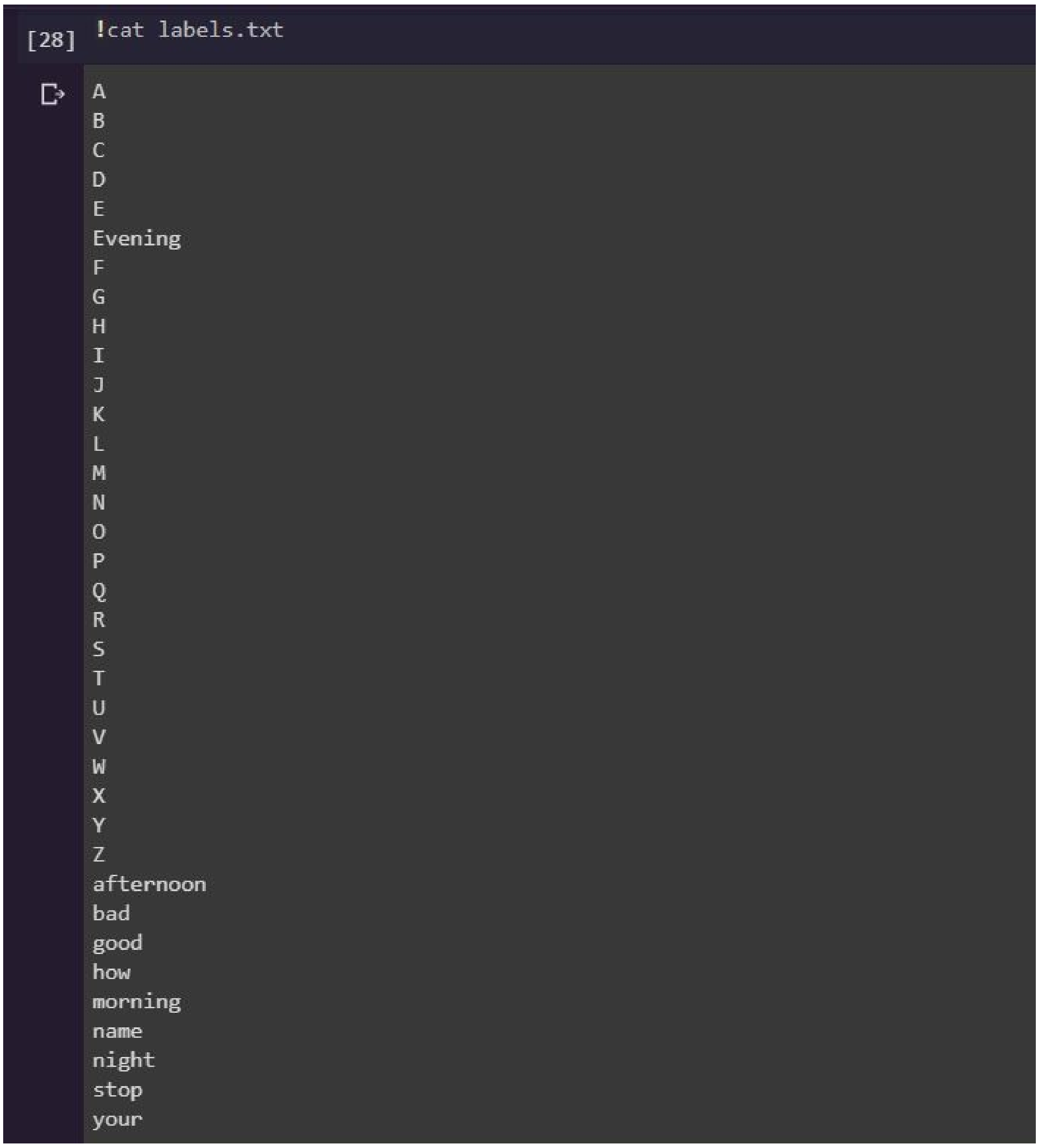
Mobile phones are already a huge part of our lives, and combining them with the power of machine learning is something that, in theory, can create user experiences that delight and impress users.

To use TensorFlow Mobile, you need to have a TensorFlow model that's successfully working in a desktop environment. Actually there are two options: TensorFlow for Mobile and TensorFlow Lite. TensorFlow Lite is in developer preview and is an evolution of TensorFlow Mobile, where models will have a smaller binary size, fewer dependencies, and better performance.It works with Android devices.

CHAPTER 5

RESULTS

A dataset of 68775 images belonging to 36 classes (36 labels) was prepared for training the model. The images were classified into 36 signs with almost 1900 images for each sign. Each image was resized and and resolution was reduced to 120x80. The content of the label.txt file that contains the label of each classes is given in the Figure below.



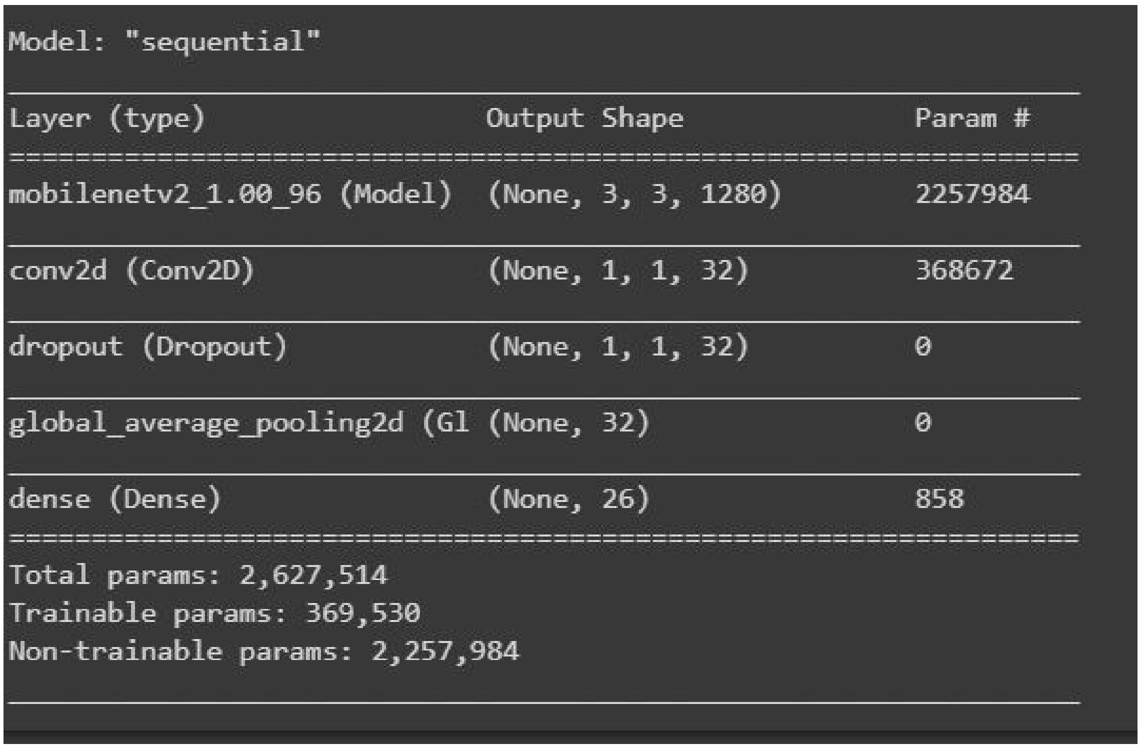
```
[28] !cat labels.txt
```

```
A
B
C
D
E
Evening
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
afternoon
bad
good
how
morning
name
night
stop
your
```

Figure 5.1: Label.txt file

5.1 MACHINE LEARNING MODEL

MobileNets Convolutional Neural Networks were trained with the image dataset created by the data preparation process. The figure shows the model summary, layers used for training the classifier. The model was trained in Google colab using Google compute engine.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_96 (Model)  (None, 3, 3, 1280)        2257984
_________________________________________________________________
conv2d (Conv2D)              (None, 1, 1, 32)          368672
_________________________________________________________________
dropout (Dropout)            (None, 1, 1, 32)          0
_________________________________________________________________
global_average_pooling2d (Gl (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 26)                858
=================================================================
Total params: 2,627,514
Trainable params: 369,530
Non-trainable params: 2,257,984
_________________________________________________________________
```

Figure 5.2: Model Summary

The figure 5.3 shows the output of training of the model in Googe colab using Tensorflow and Keras.

5.2 USER INTERFACE

The User Interface of the classifier was created using Android Studio.Prior to the main design the application was tested using tensorflow's image classifier api and required results were obtained. The API was able to classify each signs trained in the neural network. The Figure shows the Tensorflow API test results.

The figure 5.4 shows the user interface by Tensorflow Lite Image Classifier used for testing. . Figure 5.5 shows the layout view of the application while at build stage in

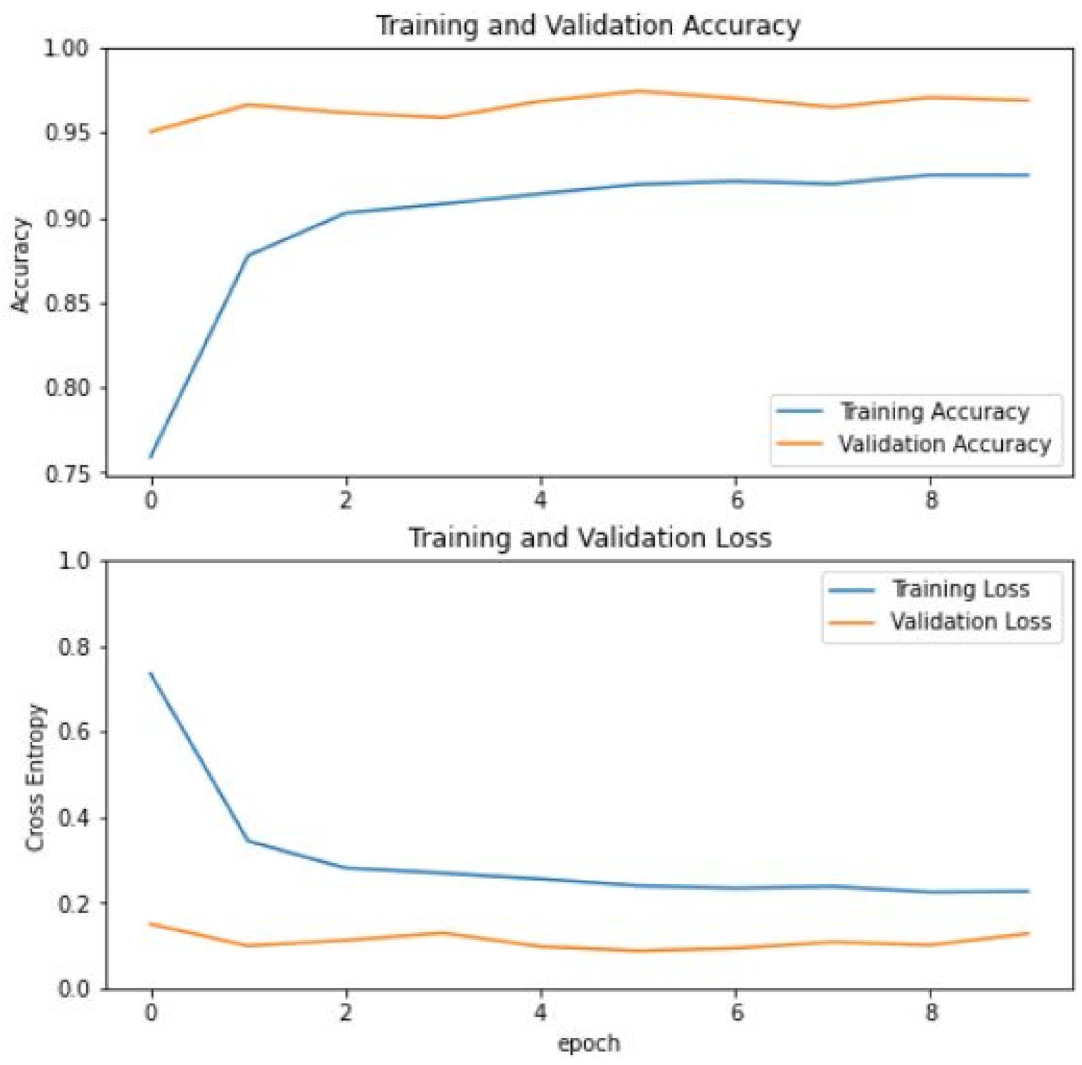
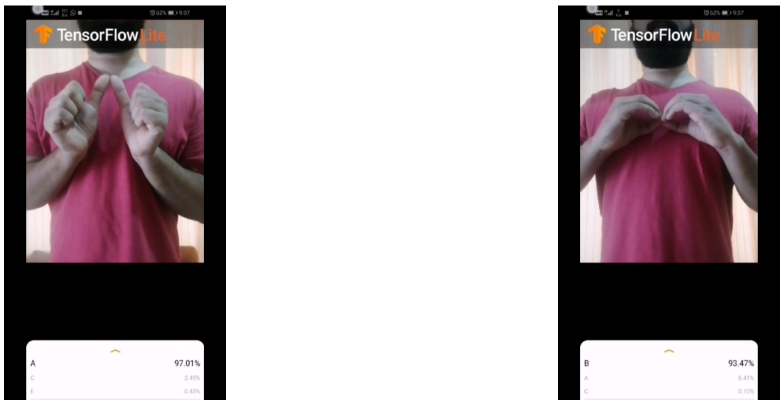


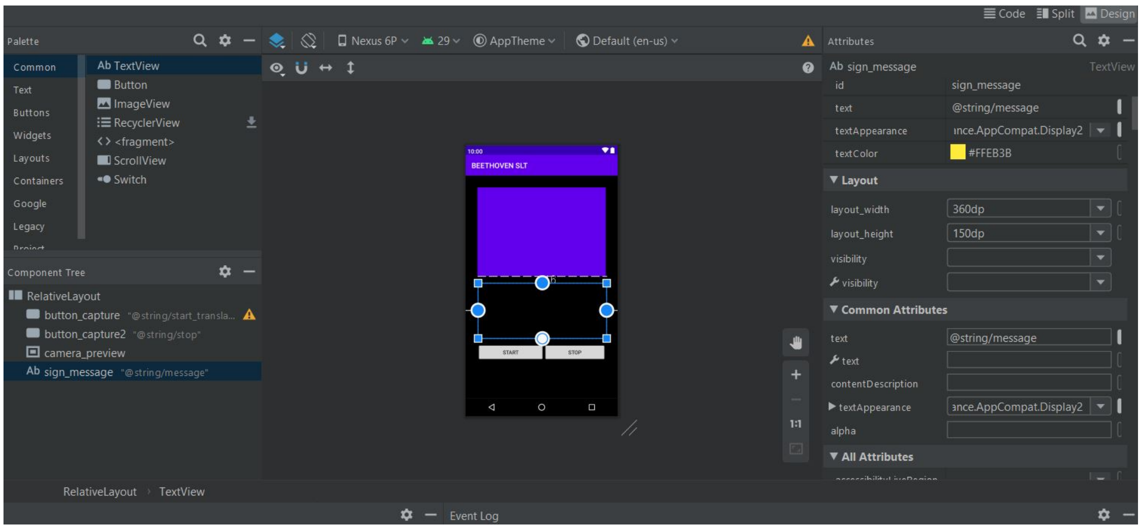
Figure 5.3: Training and validation graph depicting accuracy and loss

Android studio.The figure 5.6 shows the User Interface for Beethoven SLT. OpenCV Android library was imported inorder to process images for better efficiency while predicting. The application shall be compatible in all the android devices higher than Android Lollipop (Android 5).



(a) A (b) B

Figure 5.4



Component Tree
RelativeLayout
button_capture "@string/start_transla...
button_capture2 "@string/stop"
camera_preview
Ab sign_message "@string/message"

RelativeLayout › TextView

Attributes
Ab sign_message TextView

id	sign_message
text	@string/message
textAppearance	ınce.AppCompat.Display2
textColor	🟨 #FFEB3B
Layout	
layout_width	360dp
layout_height	150dp
visibility	
visibility	
Common Attributes	
text	@string/message
text	
contentDescription	
textAppearance	ınce.AppCompat.Display2
alpha	
All Attributes	

Figure 5.5: Android Studio Layout View

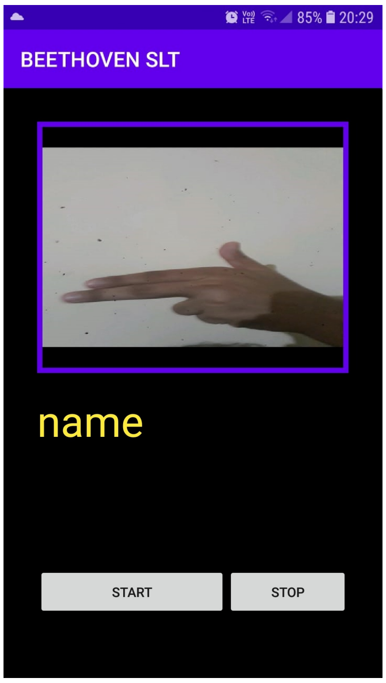


Figure 5.6: User Interface

CHAPTER 6

CONCLUSION

The project proposes a machine learning based real-time sign language and dynamic gesture recognition system. Proposed system generates texts/characters that are equivalent to the sign captured by the camera. The MobileNets Convolutional neural network is used to determine the probabilities for each category and if the category with the highest probability surpasses probability of 0.8, it is identified as a correct gesture.

We wish to extend our work further in recognising continuous sign language gestures with better accuracy. This method for individual gestures can also be extended for sentence level sign language. The future scope also includes an application program interface for various video conference software and a peer to peer sign language learning and validating platform.

REFERENCES

1. **Anil Kumar, D.**, **S. S**, **P. Kishore**, **K. Eepuri**, and **T. Maddala** (2018). S3drgf: Spatial 3d relational geometric features for 3d sign language representation and recognition. *IEEE Signal Processing Letters*, **26**, 1–1.

2. **El-Bendary, N.**, **H. M. Zawbaa**, **M. S. Daoud**, **A. E. Hassanien**, and **K. Nakamatsu** (2010). Arslat: Arabic sign language alphabets translator, 590–595.

3. **Khan, R. Z.** and **N. Ibraheem** (2012). Hand gesture recognition: A literature review. *International Journal of Artificial Intelligence Applications (IJAIA)*, **3**, 161–174.

4. **Lu, W.**, **Z. Tong**, and **J. Chu** (2016). Dynamic hand gesture recognition with leap motion controller. *IEEE Signal Processing Letters*, **23**(9), 1188–1192. ISSN 1558-2361.

5. **Melnyk, M.**, **V. Shadrova**, and **B. Karwatsky** (2014). Towards computer assisted international sign language recognition system: a systematic survey. *Int. J. Comput. Appl*, **89**(17), 44–51.

6. **Naguri, C. R.** and **R. C. Bunescu** (2017). Recognition of dynamic hand gestures from 3d motion data using lstm and cnn architectures, 1130–1133. ISSN null.

7. **Potter, L. E.**, **J. Araullo**, and **L. Carter**, The leap motion controller: a view on sign language. *In Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*. ACM, 2013.

Khan and Ibraheem (2012) Potter *et al.* (2013) Melnyk *et al.* (2014) Lu *et al.* (2016) Naguri and Bunescu (2017) Anil Kumar *et al.* (2018) El-Bendary *et al.* (2010)