VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590018, KARNATAKA

A Project Report

on

DEEP NEURAL FRAMEWORK FOR CONTINOUS SIGN LANGUAGE DETECTION

Submitted in partial fulfillment of the requirements for the VIII Semester of degree of ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi*

by

Suprith Satish
1RN19IS161

Vijay Payyavula
1RN19IS174

Vishal Veeraraj Shetty
1RN19IS178

Yash S Sindhe
1RN19IS186

Under the Guidance of

Ms.SHYLA N

Assistant Professor

Department of ISE

Department of Information Science and Engineering

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar Post,
Channasandra, Bengaluru – 560 098

2022-2023

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar Post
Channasandra, Bengaluru – 560 098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

ESTD:2001
An Institute with a Difference

CERTIFICATE

Certified that the project work entitled ***Deep Neural Framework for Continuous Sign Language Detection*** has been successfully completed by **Suprith Satish (1RN19IS161), Vijay Payyavula(1RN19IS174), Vishal Veeraraj Shetty (1RN19IS178), Yash S Sindhe (1RN19IS186)** bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of degree **Bachelor of Engineering** in **Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2022-2023**. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Ms. Shyla N Project Guide	**Dr. Prakash S / Dr. Bhagyashree Ambore** Project Coordinator
Dr. Suresh L HOD, Dept of ISE	**Dr. Ramesh Babu H S** Principal

External Viva

Name of the Examiners	**Signature with Date**
1. _____	**1.** _____
2. _____	**2.** _____

DECLARATION

We, **Suprith Satish [1RN19IS161], Vijay Payyavula [1RN19IS174], Vishal Veeraraj Shetty [1RN19IS178], Yash S Sindhe [1RN19IS186]** students of VIII Semester B.E. in Information Science and Engineering, RNS Institute of Technology hereby declare that the Project entitled ***Deep Neural Framework for Continuous Sign Language Detection*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester of degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2022-2023.

Place: Bengaluru
Date:

SUPRITH SATISH	**(1RN19IS161)**
VIJAY PAYYAVULA	**(1RN19IS174)**
VISHAL VEERARAJ SHETTY	**(1RN19IS178)**
YASH S SINDHE	**(1RN19IS186)**

ABSTRACT

Sign Language plays an indispensable role in the lives of people having speaking and hearing disabilities. Recognition of American Sign Language using Computer Vision is very challenging due to its increasing complexity and high inter class variations. In this paper, convolution neural networks (CNNs) are used to recognize the Alphabets. We are detecting the sign language using CNN algorithm.

More than 100 different sign languages are utilized across the world for understanding the deaf and speech impaired groups, they are Indian Sign Language, American Sign Language (ASL), Italian Sign Language, and many others. More than thousands and millions of people and about millions of people in India are using Sign Language as their primary mode of communication. American Sign Language is the widely used sign language and it ranks fourth most used languagein North America. It is a surprise to know that not only in the USA, ASL is also used in more than 30 other nations, where English is considered as the primary mode of communication. Around a million people in the USA and various parts of the globe use ASL as their primary mode of communication.

ACKNOWLEDGMENT

We would like to place our gratefulness to all those people who played an important role in guiding and making the project successful.

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

First of all, we would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of project work.

In this regard, we express our sincere gratitude to the principal **Dr. Ramesh Babu H S,** for providing us with all the facilities in this college.

We are extremely grateful to our own and beloved Professor and Head of the Department of Information science and Engineering, **Dr. Suresh L**, for having accepted to patronize us in the right direction with all his wisdom.

We place our heartfelt thanks to **Ms. Shyla N,** Assistant Professor, Department of Information Science and Engineering for having guided for project and all the staff members of our department for helping us out at all times.

We would like to thank Project coordinators **Dr. Prakasha S**, Associate Professor and **Dr. Bhagyashree Ambore**, Assistant Professor, Department of Information Science and Engineering for supporting and guiding us all through.

We thank our beloved friends and our parents for supporting and encouraging us throughout. We made an honest effort in this assignment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

APE	- Apple Photo Enhancer
AUVs	- Autonomous Underwater Vehicles
BN	- Batch Normalization
BReLU	- Bilateral Rectified Linear Unit
CNNs	- Convolutional Neural Networks
DPED	- DSLR Photo Enhancement Dataset
FP	- Fixation Prediction
GANs	- Generative Adversarial Networks
HVS	- Human Visual System
LSGANs	- Least Squares Generative Adversarial
MARE	- Marine Autonomous Robotic Explorer
Mbad-EN	- Multi-band fusion-based enhancement
MRF	- Markov Random Field
PAFs	- Part Affinity Fields
RGB	- Red Green Blue
ROVs	- Remotely Operated Vehicles
SOD	- Salient Object Detection
Synset	- Synonym Set

CHAPTER 1

INTRODUCTION

1.1 Background

People with hearing and speech impairments utilize American Sign Language (ASL) as a language. It is essential to their expression and communication. The intricacy of the language and substantial inter-class variances make it difficult to recognize ASL using computer vision. Convolution neural networks (CNNs) are a viable method for tackling this problem.

A deep learning algorithm used for image recognition is called a CNN. They have been successfully employed in a number of applications, such as gesture, face, and object identification. CNNs can be trained to recognize the hand motions that correspond to each letter of the ASL alphabet in the context of understanding ASL gestures. This entails teaching the model the patterns and characteristics that identify each letter by feeding it a sizable collection of photos of ASL motions.

Accessibility and communication possibilities for people with hearing and speaking impairments can be significantly improved by using CNNs to identify ASL. Computers can transform ASL motions into written or spoken words by identifying them, enabling seamless communication between individuals with and without impairments. Additionally, this technology may be incorporated into a variety of gadgets and platforms, including laptops, tablets, and smart-phones, making it readily accessible to those in need. In conclusion, CNNs' ability to recognize ASL gestures has the potential to significantly enhance the quality of life for those who have hearing and speech impairments.

1.2 Existing Systems and their Limitations

Sign language is the most natural and effective means for deaf and hearing individuals to communicate with one another. Due to challenges in hand segmentation and appearance variations among signers, American Sign Language (ASL) alphabet recognition (i.e. fingerspelling) with a marker-less visual sensor is a difficult process. Existing color-based sign language recognition systems face numerous obstacles, including complicated background, hand segmentation, high inter-class and intra-class variability, and a deep learning architecture

based on the Principal Component Analysis Network (PCANet). On this detection, two learning algorithms for the PCANet model are given.

Disadvantages of Existing System:

- Sign language recognition without CNNs requires the manual extraction of features from the video data. This can be time-consuming and requires expert knowledge of sign language and computer vision techniques.
- ASL signs can vary significantly in appearance and movement, even among signers with similar signing styles. This variability can make it difficult to extract meaningful features for recognition.
- Many sign language recognition systems without CNNs are designed to recognize a limited vocabulary of signs. This can limit their usefulness in real-world communication, where signers may use a wide variety of signs.
- Sign language recognition without CNNs can be sensitive to changes in lighting conditions, which can affect the appearance of signs and make recognition more difficult.
- Real-time recognition of sign language without CNNs can be challenging, especially for complex signs or signs that are produced quickly. This can result in errors or delays in recognition.
- Sign language recognition systems lacking CNNs may result in recognition errors when adapting to individual signers, especially for those with unique signing styles or physical differences that affect their signing.

1.3 Proposed System

We're developing a machine learning project that uses a Convolutional Neural Network (CNN) algorithm to detect and predict sign language. The system we're proposing will be capable of recognizing user-inputted hand gestures in real-time. To accomplish this, the system will use a camera or other input device to capture an image of the user's hand motions. The image data will then be fed into the CNN model for recognition. By using machine learning, the system will be able to learn and adapt to new data over time, resulting in faster and more accurate recognition of sign language.

Advantages of the proposed system:

- Using a CNN algorithm can lead to high accuracy in recognizing and translating hand motions in sign language.
- The system can provide reliable and efficient translations in real-time without the need for a human translator.
- This can save time in communication between deaf and hearing individuals and in situations requiring sign language interpretation.
- Implementing the CNN model in a real-time system can result in a faster recognition process, which can be useful in emergency situations or fast-paced environments.

The system can learn and adapt to new data, resulting in faster and more efficient recognition of hand movements over time.

CHAPTER 2

LITERATURE SURVEY

A Literature survey describes the concept of how the concept has emerged, how it has been implemented and what is the current status.

Hand gesture recognition is an important aspect of human-computer interaction, which enables people to communicate with computers or other digital devices using hand movements.

In **paper [1]**, the authors described a revolutionary real-time approach for hand gesture identification. Firstly, the hand region is extracted from the background using the background subtraction method, which involves separating the hand region from the rest of the image by subtracting the background pixels from the foreground pixels. Then, the palm and fingers are detected and recognized using an algorithm designed for this purpose. Finally, a rule classifier is used to predict the labels of the hand movements based on the extracted features.

The researchers conducted experiments on a dataset of 1300 photos and found that their method performs well and is highly efficient. They also compared their approach with a state-of-the-art method on another dataset of hand movements and found that their method outperforms the previous method.

The authors are discussing the importance of hand gesture recognition in human-computer interaction (HCI), and how it has become more necessary with the development of new technologies. They explain that there are two types of gestures, static and dynamic, and that they can be used as a tool of communication between computers and humans.

The authors then go on to describe various applications of hand gesture recognition, such as sign language recognition, augmented reality, and robot control. They also explain the workflow of hand gesture recognition, which involves detecting the hand region, extracting features to describe hand gestures, and recognizing hand gestures by measuring the similarity of the feature data. The authors discuss various classifiers that can be used to discriminate hand gestures, but note that the time cost is very high.

The authors then propose their own method for hand gesture recognition that is based

on finger recognition. They intend to present this method as a simple and efficient alternative that does not require expensive sensors or special tape. They further claim that their method is highly efficient and suitable for real-time applications. Overall, the authors aim to showcase the importance of hand gesture recognition and propose their own method as a viable solution.

Paper **[2]** deals with the breed classification of dogs. To classify dog breed is a challenging part under a deep convolutional neural network. A set of sample images of a breed of dogs and humans are used to classify and learn the features of the breed. The images are converted to a single label of dimension with image processing.

The images of human beings and dogs are considered for breed classification to find the existing percentage of features in humans of dogs and dogs of human. This research work has used principal component analysis to shorten the most similar features into one group to make an easy study of the features into the deep neural networks. And, the facial features are stored in a vector form. This prepared vector will be compared with each feature of the dog into the database and will give the most efficient result. In the proposed experiment, 13233 human images and 8351 dog images are taken into consideration.

The images under test are classified as a breed with the minimum weight between test and train images. This paper is based on research work that classifies different dogs breed using CNN. If the image of a dog is supplied then the algorithm will work for finding the breed of dog and features similarity in the breed and if the human image is supplied it determines the facial features existing in a dog of human and vice-versa.

This research paper discusses the use of machine learning, specifically Convolutional Neural Networks (CNN), which is a type of deep learning, in various fields such as business analytics and healthcare improvement. The paper explains that CNN is a set of neurons with different biases and weights, and it receives inputs to produce outputs. The architecture of CNN helps reduce features and increase accuracy.

The author goes on to explain the three steps of CNN: the convolutional layer, the pooling layer, and the output layer. In the convolutional layer, images are converted into 2D matrices with weighted entries to recognize features. The pooling layer is used to reduce the size of the image when it is too large, and this can be done through methods such as max pooling or average pooling. The output layer produces a compressed image with extracted features.

The author then describes about the CNN architecture, where there are multiple convolutional layers, each of which uses different layers. These filters are essentially small grids of numbers that perform a type of mathematical operation called convolution. Each filter is designed to detect a specific feature in the input image, such as edges, curves, or shapes. The filters "slide" across the input image, performing element-wise multiplication and summation operations at each position, and creating a new feature map with the results. This process is repeated for each filter in the layer, resulting in multiple feature

The author of the paper **[3]** is proposing a new user-independent recognition system for American Sign Language (ASL) alphabet recognition using depth images captured from a low-cost Microsoft Kinect depth sensor. The paper discusses the challenges faced by existing color-based sign language recognition systems and highlights the advantages of using depth information over color images for hand segmentation and feature extraction. The proposed method uses a deep learning architecture, namely Principal Component Analysis Network (PCANet), for feature learning and linear Support Vector Machine (SVM) classifier for recognition. The author presents two strategies for learning the PCANet model, namely training a single PCANet model for all users and training separate

PCANet models for each user. The paper concludes that the proposed method outperforms the state-of-the-art recognition accuracy using a leave-one-out evaluation strategy on a public dataset of real depth images captured from various users.

In this research paper the author is discussing how sign language and gestures can be used as a means of communication for people who are deaf or hearing-impaired. These individuals may not be able to rely on spoken language to communicate, so sign language can be a very useful alternative.

The paper focuses specifically on automatic sign language recognition (SLR) systems, which are computer programs that can recognize and translate sign language gestures into written or spoken language. These systems typically use sensors or cameras to capture hand gestures and then analyze them to determine what they mean. The idea is that SLR systems can be used to improve communication between people who use sign language and those who do not. For example, a person who is deaf could use sign language to communicate with someone who does not know sign language, and an SLR system could automatically translate their signs

into spoken language for the other person to understand.

The paper describes the two main approaches to SLR: sensor-based and vision-based methods. Sensor-based methods involve wearing special gloves or sensors to capture hand movements, while vision-based methods use cameras to capture images of hand gestures. The paper then goes on to describe the advantages of vision-based methods, particularly those that use low-cost depth cameras such as Microsoft Kinect.

The research paper describes a new approach for recognizing fingerspelling in sign language that does not depend on the specific person performing the gestures, meaning it is "signer-independent." The proposed method uses depth images, which capture the distance from a camera to the object being imaged, to capture hand gestures.

The approach involves several steps. First, the hand region is segmented from the depth image and the depth values are normalized. Next, a PCANet model, which is a type of deep learning model, is used to learn features from the segmented hand regions. The PCANet model identifies patterns and features in the hand images that are relevant for fingerspelling recognition. Finally, the extracted features are fed into a linear Support Vector Machine (SVM) classifier, which is a machine learning algorithm that can classify data into different categories. The SVM classifier uses the learned features to identify the letters of the fingerspelling alphabet in the input images.

In paper **[4]** author is describing a research paper that presents the development and comparison of two computer vision models (You Only Look Once - YOLO and Transfer Learning - TL) to detect and recognize ventilation objects (masks and tubes) and their positions on the patient's face. The study aims to improve the accuracy of detecting these objects during medical procedures. The paper presents the development processes and compares the performance of the two models, with the TL model showing a better performance (98%) compared to the YOLO model (93%).

The author of the research paper is discussing the potential benefits of using computer vision technology to detect and recognize ventilation objects such as masks and tubes, and their positions on a patient's face in real-time. This technology can help healthcare providers monitor patients round-the-clock and detect adverse events and patient deterioration early on, which is crucial in critical care settings. This type of monitoring is often time-consuming and labor-intensive for healthcare providers, and Electronic Medical Records (EMR) may not

provide enough quantitative data to evaluate patient-device interaction and detect deteriorations or adverse events.

The paper explains how computer vision technology can recognize and locate objects in an image frame, such as a patient's face, and assign a specific class or label to the objects of interest. The study also compares two computer vision models, You Only Look Once (YOLO) and Transfer Learning (TL) model, to detect ventilation objects, with the TL model performing better with 98% accuracy. The research paper concludes that computer vision technology can be helpful in healthcare settings, particularly in the ICU, and suggests future directions for this technology, such as recognizing events based on object changes and patterns that can predict adverse events.

The authors of paper **[5]** are discussing a system for recognizing hand gestures using computer vision. The authors explain that hand gestures can convey rich information and are therefore useful in a variety of applications such as robot control and intelligent furniture. The paper presents a method for recognizing hand gestures that involves several steps.

First, the authors use a skin color model and an AdaBoost classifier based on Haar features to segment the human hand from the background in a video frame. This step is necessary because the hand typically appears in a variety of backgrounds and can be difficult to distinguish from other objects in the image.Next, the authors use the CamShift algorithm to track the hand in real-time as it moves within the video stream. This allows the system to continuously recognize the hand gestures even as the hand moves. Finally, the authors use a convolutional neural network (CNN) to recognize the hand gestures that are being made. They trained the CNN to recognize 10 common digits, and in their experiments.

The author of this research paper is discussing the development of an interactive system for hand gesture recognition and its significance in the field of human-computer interaction. The paper emphasizes the importance of recognizing hand gestures due to their ability to convey enriched information and their increasing use in various fields, such as somatosensory games, sign language recognition, and UAV control.

One of the primary challenges in hand gesture recognition is segmenting the hand gesture from the background of the image or video. The author discusses several methods used for hand gesture segmentation, including skin color model, edge detection, motion information, and statistical template matching. Each of these methods has its own advantages and

disadvantages. For example, skin color model is not affected by hand postures, but it cannot exclude objects similar to the skin color such as the human face. On the other hand, edge detection can segment hand gestures based on the discontinuity of gray value in the margin area of the image region, but it can be easily interrupted by noise and has strict requirements for the background.

The author of this paper then proposes a method for hand gesture segmentation that pre-processes images, establishes a Gaussian mixture model according to skin colors, and segments hand gestures by combining with AdaBoost classifier based on Haar features. This method was found to be effective in segmenting hand gestures from complicated backgrounds with high accuracy.

The paper also introduces the CamShift algorithm for hand gesture tracking, which involves real-time monitoring and location of hand gestures in a video, as well as the LeNet-5 network for hand gesture recognition. The system is tested using 10 common hand gestures ranging from 1-10, and the results show 98.3% accuracy in recognizing the gestures in a complicated background.

The paper **[6]** presents a system for automatic translation of static gestures of alphabets in American Sign Language (ASL) using three different feature extraction methods and a neural network. The system deals with images of bare hands, allowing users to interact with the system naturally. The image is processed and converted to a feature vector that is compared with the feature vectors of a training set of signs. The system is flexible, allowing for rotation, scaling, and translation variations within the image.

The paper reports the implementation and testing of the system using datasets of a number of samples of hand images for each sign. The three feature extraction methods are compared, and the best one is suggested based on the results obtained from the artificial neural network (ANN). The system achieved a recognition accuracy of 92.33% for selected ASL signs.

The author is discussing the concept of deep learning, which is a subset of machine learning in artificial intelligence that imitates the working of a human brain in processing data and creating approach-patterns for use in decision making. Deep learning is part of a broader family of machine learning methods built on artificial neural networks with representation learning. The author highlights the advantage of deep learning over traditional machine

learning methods in that there is no need for feature extraction, which is a preprocessing layer that requires detailed knowledge of the problem domain and can be quite complicated.

The author proposes a system which aims in identifying the expression of voice disorder and hearing loss people and recognize hand gestures and provide a description as text. The system will also include an audio feature for easy understanding. The system is application-based and can be easily accessed with a webcam, providing a learning platform online.

The author explains the concept of Convolutional Neural Networks (CNNs) and their various applications, which include image classification, object detection, object tracking, text detection and recognition, speech and natural language processing. The author specifically discusses the use of CNNs in emotion recognition models, which are based on deep learning approaches. The author mentions that CNNs are inspired by the natural visual perception mechanism of living creatures and can be used for various applications. In this particular case, the author focuses on the emotion recognition model, which is built using a deep learning approach based on CNNs.

The author also explains the process of data collection for the emotion recognition model. The data collection interface was created by fine-tuning a pre-trained CNN model on an existing dataset of face expression images collected from the web. This new dataset was then used to train a fine-tuned CNN model, which serves as the basis for the Emotion Training Platform.

The author of the paper **[8]** describes the development of a video-based continuous sign language recognition system for German Sign Language (GSL). The system is based on continuous density Hidden Markov Models (HMM), with one model for each sign, and feature vectors reflecting manual sign parameters are used for training and recognition. The system employs beam search to reduce computational complexity during the recognition task. The system aims for an automatic signer-dependent recognition of sign language sentences, based on a lexicon of 97 signs of German Sign Language (GSL), using a single color video camera for image recording.

The author describes sign languages and their unique characteristics, such as manual and non-manual parameters, 3D signing space, and simultaneous structure with a parallel temporal and spatial configuration. The syntax of sign language sentences is also described as being less strict than in spoken language, with a configuration that typically refers to time,

location, person, and predicate. The paper notes that deaf individuals may have a limited vocabulary of spoken language and difficulty with reading and writing, making sign language recognition an important technology.

The paper then outlines the basic difficulties that can arise during the recognition of continuous sign language, including occlusion of fingers or hands, the need to detect sign boundaries automatically, co-articulation (the influence of the preceding and subsequent signs), variation in signing speed and position, loss of depth information in 2D projections, and the computational complexity of processing a large amount of image data in real-time. To address these challenges, the author proposes using Hidden Markov Models (HMMs) based on statistical methods, which can compensate for time and amplitude variances in signals and detect sign boundaries automatically. The paper notes that HMMs have been proven effective for speech and character recognition, making them an ideal approach to sign language recognition.

The author then describes the system architecture, which is based on feature vectors reflecting manual sign parameters as input for training and recognition. Beam search is employed to reduce computational complexity during recognition. The system concentrates on manual sign parameters and does not yet use non-manual parameters for recognition. The paper examines the influence of different features reflecting different manual sign parameters on the recognition results. Results are given for varying features and varying sized vocabulary. The system achieves an accuracy of 91.7% based on a lexicon of 97 signs. In summary, the paper presents the development of a video-based sign language recognition system and analyzes its accuracy and performance for the German Sign Language using a lexicon of 97 signs.

The authors of paper **[9]** discusses the importance of gesture recognition as a means of human-computer interaction and how people are seeking a more natural way to perform gesture recognition. The author notes that computer vision-based gesture recognition can be a convenient and efficient way to transfer human feelings and instructions to computers, which can significantly improve the efficiency of human-computer interaction.

The paper also describes the different algorithms used for gesture recognition, such as hidden Markov models, dynamic time rounding algorithm, and neural network algorithm, and outlines the three main steps involved in the process: image collection, hand segmentation, and gesture recognition and classification. The paper reviews the development of computer vision-based gesture recognition methods over the past 20 years, discusses the research status at home

and abroad, and summarizes the advantages and disadvantages of different gesture recognition methods.

The author notes that gesture recognition technology has evolved over the years, from early methods that used wearable devices to more recent optical marking methods and computer vision-based approaches. However, there are still challenges that need to be addressed in order to improve the accuracy and usability of gesture recognition systems. One challenge is poor universality, which refers to the fact that different users may use different gestures to convey the same meaning. For example, one person may use a thumbs-up gesture to indicate approval, while another person may use a nod of the head.

This variability makes it difficult for gesture recognition systems to accurately interpret user gestures across different contexts and users. Another challenge is sensitivity to illumination changes and occlusion. Illumination changes can occur when lighting conditions change, such as when a user moves from a bright room to a dimly lit room. Occlusion occurs when part of the user's hand or body is blocked from view, such as when the hand is partially hidden behind an object. These challenges can make it difficult for gesture recognition systems to accurately interpret user gestures in real-world environments.

Despite these challenges, the author notes that advances in computer operation speed and algorithm development can help improve the performance of gesture recognition systems. For example, faster processing speeds can help reduce the latency between gesture input and system response, while improved algorithms can help better handle variability in user gestures and illumination changes.

The author of paper **[10]** is discussing a method for semantic segmentation using fully convolutional neural networks (FCNs). The paper presents the idea of building FCNs that can take inputs of arbitrary sizes and produce correspondingly-sized outputs with efficient inference and learning. The key insight is that the FCN can be trained end-to-end, pixels-to-pixels, and can outperform state-of-the-art methods for semantic segmentation. The paper details the space of fully convolutional networks, explains their application to spatially dense prediction tasks, and draws connections to prior models.

The paper also discusses the adaptation of contemporary classification networks such as AlexNet, VGG net, and GoogLeNet into fully convolutional networks and the transfer of their learned representations by fine-tuning to the segmentation task. The paper introduces a

skip architecture that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. The method achieves state-of-the-art segmentation performance on several benchmark datasets including PASCAL VOC, NYUDv2, and SIFT Flow, while inference takes less than one fifth of a second for a typical image.

The author discusses a method for performing semantic segmentation of images using fully convolutional neural networks (FCNs). The authors draw on recent advances in deep neural networks for image classification and transfer learning, and adapt and fine-tune classification nets for direct, dense prediction of semantic segmentation. The paper charts the space of FCNs and situates prior models, both historical and recent, in this framework.

The author describes the idea of extending a convolutional neural network (convnet) to arbitrary-sized inputs, which was first introduced by Matan et al. for recognizing strings of digits. The authors also discuss historical works on fully convolutional inference and learning for detection, such as Wolf and Platt's work on expanding convnet outputs to 2-dimensional maps of detection scores for postal address blocks, and Ning et al.'s work on defining a convnet for coarse multiclass segmentation of C. elegans tissues.

The paper focuses on adapting and extending deep classification architectures, using image classification as supervised pre-training, and fine-tuning fully convolutionally to learn from whole image inputs and whole image ground truths. The authors fuse features across layers to define a nonlinear local-to-global representation that is tuned end-to-end. The method is compared to other semantic segmentation methods, including Hariharan et al. and Gupta et al.'s hybrid proposal-classifier models, which fine-tune an R-CNN system by sampling bounding boxes and/or region proposals for detection, semantic segmentation, and instance segmentation.

In paper **[11]**, The author is conveying that they have evaluated the possibility of using features extracted from the activation of a deep convolutional network that was trained on a large set of object recognition tasks in a fully supervised manner, for repurposing to novel generic tasks. The generic tasks may differ significantly from the original tasks and there may not be sufficient labeled or unlabeled data to conventionally train or adapt a deep architecture to the new tasks. The author investigates and visualizes the semantic clustering of deep convolutional features with respect to several tasks, including scene recognition, domain adaptation, and fine-grained recognition challenges.

The author also compares the efficacy of relying on various network levels to define a fixed feature and reports novel results that significantly outperform the state-of-the-art on several important vision challenges. Finally, the author is releasing an open-source implementation of these deep convolutional activation features, along with all associated network parameters, to enable vision researchers to experiment with deep representations across a range of visual concept learning paradigms.

This author discusses the use of deep convolutional networks as a method for learning effective representations of visual data. The authors argue that conventional visual representations, which are based on flat feature representations involving quantized gradient filters, have plateaued in recent years and that deep or layered compositional architectures should be able to capture more salient aspects of a given domain through discovery of salient clusters, parts, mid-level features, and/or hidden units.

The author then describes their recent research on the application of deep models to large-scale visual recognition tasks, which have performed extremely well in domains with large amounts of training data. However, with limited training data, fully-supervised deep architectures will generally dramatically overfit the training data. To address this problem, the authors investigate semi-supervised multi-task learning of deep convolutional representations, where representations are learned on a set of related problems but applied to new tasks which have too few training examples to learn a full deep representation.

The model can be considered as a deep architecture for transfer learning based on a supervised pre-training phase, or simply as a new visual feature defined by the convolutional network weights learned on a set of pre-defined object recognition tasks.

The authors report that their model, which is based on a deep convolutional network trained on ImageNet, outperforms a host of conventional visual representations on standard benchmark object recognition tasks, including Caltech-101, the Office domain adaptation dataset, the Caltech-UCSD Birds fine-grained recognition dataset, and the SUN-397 scene recognition database. They also analyze the semantic salience of deep convolutional representations, comparing visual features defined from such networks to conventional representations, and find that convolutional features appear to cluster semantic topics more readily than conventional features.

The paper **[12]** discusses the development of two real-time systems for recognizing

sentence-level continuous American Sign Language (ASL) using a single camera to track the user's unadorned hands. The first system uses a desk-mounted camera and achieves 92 percent word accuracy, while the second system mounts the camera in a cap worn by the user and achieves 98 percent accuracy (97 percent with an unrestricted grammar). Both experiments use a 40-word lexicon.

The authors explain that sign language is a highly structured form of communication, with each gesture having an assigned meaning and strong rules of context and grammar. American Sign Language (ASL) is the language of choice for most deaf people in the United States, and it uses approximately 6,000 gestures for common words and finger spelling for communicating obscure words or proper nouns. However, the majority of signing is with full words, allowing signed conversations to proceed at about the pace of spoken conversation. ASL's grammar allows more flexibility in word order than English and sometimes uses redundancy for emphasis. Another variant, Signed Exact English (SEE), has more in common with spoken English but is not as widespread in America.

The authors note that previous attempts at machine sign language recognition have focused on isolated signs or fingerspelling, and most attempts have relied on instrumented gloves or desktop-based camera systems using a form of template matching or neural nets for recognition. However, the authors argue that hidden Markov models (HMMs) are well-suited for visual recognition of complex, structured hand gestures as are found in sign languages. HMMs have been used prominently and successfully in speech recognition and more recently in handwriting recognition.

The authors then describe the results of their experiments with HMM-based sign language recognition systems. They show that their system achieves high accuracy in recognizing sentence-level ASL selected from a 40-word lexicon. The first experiment demonstrates how the system can be used to communicate with a desk-based computer, while the second experiment shows how a wearable computer might use this method as part of an ASL to English translator. The experiments suggest that HMMs will be a powerful method for sign language recognition, much as they have been for speech and handwriting recognition.

Table 2.1 Literature Survey Summary

Ref. No.	Author	Technique	Results	Limitations
[1]	Zhi-Hua Chen, Jung-Tae Kim, Jianning Liang and Jing Zhang.	Hand Detection, Hand Segmentation, Finger Detection and Finger Segmentation and finally, Gesture Recognition	This method out performs the state-of-art FEMD on an image collection of hand gestures.	The presence of moving objects with similar skin color degrades the performance of hand gesture recognition.
[2]	Bickey Kumar shah, Aman Kumar and Amrit Kumar.	Classifying dog breeds based on facial recognition using CNNs.	Classifies the dog breed and detects the probability of features of dog breed that match the human features of a particular sample.	Limited dataset, generalizability, interpretability, and unspecified performance in real-world scenarios.
[3]	Walaa Aly, Saleh Aly and Sultan Almotairi.	Recognition of ASL alphabets based on depth image and PCANet features.	Outperforms other state-of-the-art methods.	The need for a depth camera, the limited focus on only the ASL alphabet

[4]	Quan T. Doa and Jamil Chaudri.	YOLO (You Only Look Once) and TL (Transfer Learning).	The current YOLO model for object detection of ventilation devices on patient's faces reached an accuracy rate of 95%.	Larger ventilation machines needed for better accuracy.
[5]	Jing-Hao Sun Ting-Ting Ji, Shu-Bin Zhang, Jia-Kui Yang and Guang-Rong Ji	Skin Color Model, AdaBoost Classifier and CamShift algorithm, CNN.	Successful in identifying hand gestures.	Limited Dataset, Overfitting of model and Lack of comparison with other methods.
[6]	Vaishali S. Kulkarni and S.D.Lokhande. Dr	ANN Approach	Successful in identifying ASL with accuracy of 92.33%.	Lack of judgment of how good the differentiation one can achieve.
[8]	B. Bauer and H. Hienz	Hidden Markov Model(HMM)	Sign Language recognition successful with accuracy of 94%.	Very static recognition of hand gestures.

[9]	Wang, Xianghan, Jiang, Jie, Wei, Yingmei, Kang, Lai and Gao, Yingying	Hidden Markov Model (HMM),, Dynamic Time Rounding Algorithm and Neural Networks.	Recognizes hand gestures accurately.	Poor universality, sensitivity to illumination changes and occlusion, and poor real-time performance
[10]	Jonathan Long, Evan Shelhamer and Trevor Darrell.	CNN Approach	Achieves state of art segmentation of PASCAL VOC, NYUDv2 and SIFT.	Lacks multi-resolution layer.
[11]	Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng and Trevor Darrell	Using DeCAF	Revives high accuracy on visual recognition tasks.	Requires heavy graphics card for training the model.
[12]	T. Starner, J. Weaver and A. Pentland	Hidden Markov Model, Wearable based and Desk Based Recognizer.	Recognizes with sign language with an accuracy of 97.6%.	Need to incorporate finger and palm tracking.

CHAPTER 3

ANALYSIS

3.1 Problem Statement

The problem statement is to develop a machine learning model that can accurately recognize and interpret sign language gestures. This involves analyzing and processing video data of hand gestures and converting it into meaningful language output. The challenge lies in overcoming the complexity and variability of hand motions, as well as the diversity of sign languages used across different regions and cultures.

The model needs to be robust enough to handle different lighting conditions, hand orientations, and background noise. The ultimate goal is to create an effective tool that enables communication and accessibility for individuals with hearing or speech impairments.

One of the main challenges in this task is the complexity and variability of hand motions. Sign language gestures can involve a wide range of hand movements, including hand shapes, movements, and orientations, which can vary significantly between different sign languages and even between different signers of the same language.

Another challenge is the diversity of sign languages used across different regions and cultures. There are many different sign languages used around the world, each with its own unique vocabulary and grammar, and the machine learning model needs to be able to recognize and interpret a wide range of sign language systems.

The model also needs to be robust enough to handle different lighting conditions, hand orientations, and background noise. This is important because sign language is often performed in a variety of environments and conditions, and the model needs to be able to recognize gestures accurately in all of these settings.

3.2 Objectives

The objectives for a sign language recognition system using Convolutional Neural Networks (CNN) may include:

1. **Accurately recognize and classify hand gestures**: The primary objective of the system is to recognize and classify sign language hand gestures accurately. The CNN model should be trained to identify different sign language gestures from the video input and convert them into meaningful language output.
2. **Robustness to variations**: The CNN model should be robust enough to handle variations in hand orientation, lighting conditions, and background noise. This would ensure that the system can recognize sign language gestures accurately in different settings.
3. **Real-time processing**: The system should be designed to process video input in real-time to enable seamless communication between individuals who use sign language and those who do not.
4. **Multilingual support**: The system should be able to recognize and interpret sign language gestures from different sign languages used around the world.
5. **Accessibility**: The system should provide accessibility for individuals with hearing or speech impairments and enable them to communicate more effectively. This would help to bridge the communication gap between individuals who use sign language and those who do not.

3.2.1 Aims of project

- Develop a software system that can accurately recognize and interpret sign language gestures in real-time.
- Apply computer vision techniques, such as image processing and machine learning, to capture and process video data of hand gestures.
- Address the challenges associated with sign language recognition, such as the variability and complexity of hand motions, and the diversity of sign languages used around the world.
- Create a system that is robust enough to handle different lighting conditions, hand orientations, and background noise.
- Improve communication and accessibility for individuals with hearing or speech impairments by providing a tool that can accurately interpret sign language gestures.

- Potentially use the system in a variety of settings, such as schools, workplaces, and public spaces, to facilitate communication between sign language users and non-users.

Continuously improve and optimize the system over time to ensure high accuracy and reliability.

3.3 Methodology

Artificial Neural Networks are used in various classification task like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural Network. In this blog, we are going to build basic building block for CNN.

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels in case of an image).
2. **Hidden Layer:** The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layers can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we back propagate into the model by calculating the derivatives. This step is called back propagation which basically is used to minimize the loss.

3.3.1 CNN Introduction

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image) and height (as image generally have red, green, and blue channels).

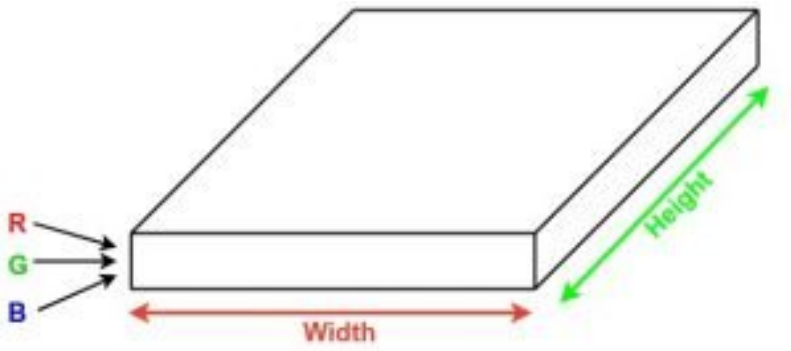


Fig 3.3.1 Image Representation

Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G and B channels now we have more channels but lesser width and height. His operation is called Convolution. If patch size is same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

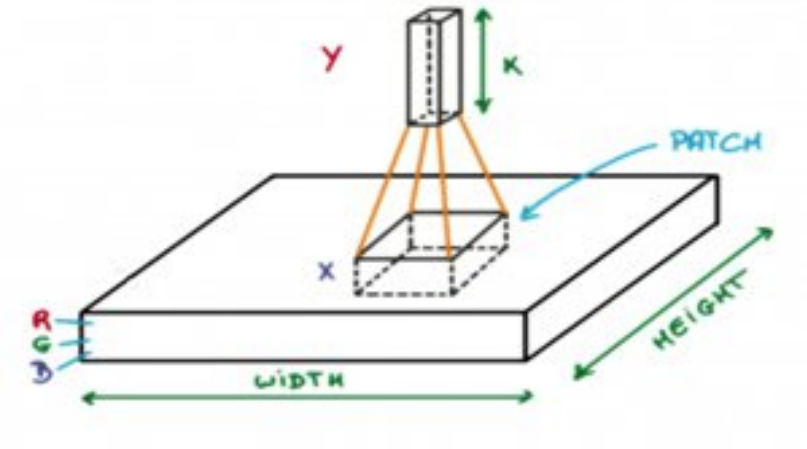


Fig 3.3.2 Convolution Process

Now let's talk about a bit of mathematics which is involved in the whole convolution process.

Convolution layers consist of a set of learnable filters (patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).

For example, if we have to run convolution on an image with dimension 34x34x3. Possible size of filters can be axax3, where 'a' can be 3, 5, 7, etc but small as compared to image dimension.

During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.

- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

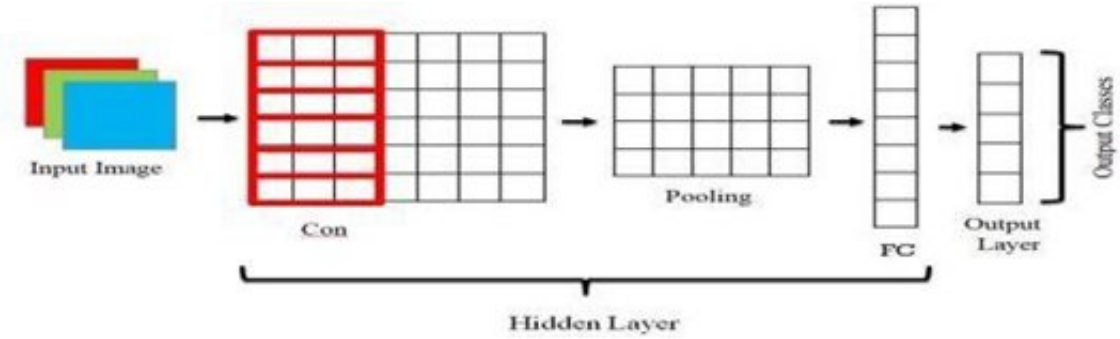


Fig 3.3.3 Hidden Layer

3.4 Software Requirement Specifications

System requirement specifications gathered by extracting the appropriate information to implement the system. It is the elaborate conditions which the system needs to attain. Moreover, the SRS delivers a complete knowledge of the system to understand what this project is going to achieve without any constraints on how to achieve this goal. This SRS does not provide the information to outside characters but it hides the plan and gives little implementation details.

3.4.1 Software Requirement Specification

- Operating system: Windows 10
- Coding Language: Python
- Software Tool: Keras
- Software: Ananconda

3.4.2 Hardware Requirement Specification

- Processor: Intel core
- Processor Speed: 1.86 GHz.
- RAM: 4GB+
- Hard Disk Space: 500 GB+
- Monitor: 15 VGA Color

3.5 Functional Requirement

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements. System should do minimal computations on its own. System should capture images.

3.6 Non-Functional Requirement

Non-Functional Requirements are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software systems.

The proposed system will utilize a camera to capture video data for analysis. All required data will be stored in a Python database to ensure reliable storage and easy access. The system will be designed to be both reliable and flexible for future enhancements, allowing for changes and updates as needed. Overall, the proposed system aims to provide a reliable, flexible, and cost-effective solution for capturing and analyzing video data.

CHAPTER 4

SYSTEM DESIGN

4.1 System Design

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way, the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering, and systems architecture.

Generally speaking, the principle of Sign language detection is shown in the figure below.The proposed system includes five modules. The initial stage is the image acquisition stage through which the real world sample is recorded in its digital form using a digital camera.

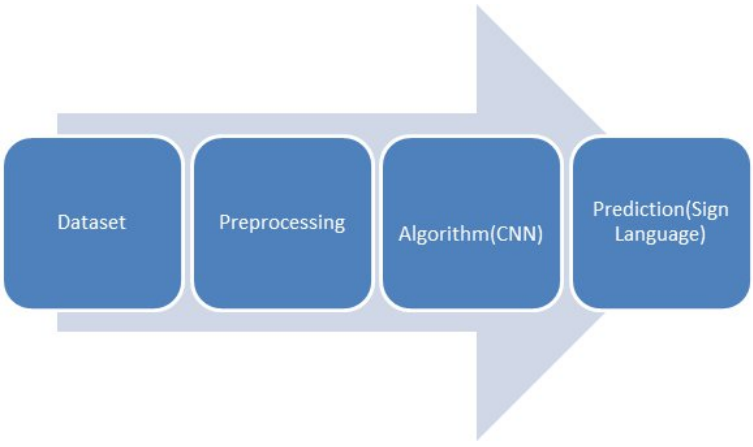


Fig 4.1 Process of Sign language detection

In the next stage of the research, the image was subjected to a pre-processing stage. Making use of it the size and complexity of the image is reduced. The precise digital information is subjected to a segmentation process which separates the rotten portion of the Animal samples. The feature extraction aspect of an image analysis focuses on identifying inherent

High-level design (HLD) explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly nontechnical to mildly technical terms that should be understandable to the administrators of the system. In contrast low level design further exposes the logical detailed design of each of these elements for programmers.

High level design is the design which is used to design the software related requirements.In this chapter complete system design is generated and shows how the modules, sub modules and the flow of the data between them are done and are integrated. It consists of very simple phases and shows the implementation process.

4.2 Design Consideration

High level design is the design which is used to design the software related requirements.In this chapter complete system design is generated and shows how the modules, sub modules and the flow of the data between them are done and are integrated. It consists of very simple phases and shows the implementation process.

4.2.1 Input

The input to the system is a video or sequence of images of a person making a sign.

4.2.2 Preprocessing

The input video or images are preprocessed to extract relevant features, such as hand shape, position, and motion. This may involve applying image processing techniques, such as cropping, resizing, and normalizing the images.. Images are taken as the input and output for image processing techniques.

It is the analysis of image to image transformation which is used for the enhancement of image. Firstly, we convert the RGB image to a gray scale image. It helps to reduce the complexity in the image and also make the work easy. Then by using min-max scalar method converts the gray scale values into binary values.

The obtained binary values are taken as the input for the further process. In the obtained binary matrix consider one value region as white and zero value region black. By using these values, the region of interest can be identified. So that the values are useful for feature extraction and identification of regions of interest.

4.2.3 Feature extraction

The extracted features are then used to extract more meaningful and relevant features that can be used as input to the machine learning model. This may involve applying dimensionality reduction techniques, such as principal component analysis (PCA) or linear discriminant analysis (LDA). In this stage extract the required feature from the identified region which is obtained from the previous step. That region is compressed by converting a reduced size matrix to control over fitting. The reduction of the matrix size helps in reducing the memory size of the images. Then the flattening process is applied to the reduced matrix, in which the reduced matrix is converted to a one-dimension array, which is used for final detection.

4.2.4 Machine learning model

The extracted features are fed into a machine learning model, such as a support vector machine (SVM), decision tree, or neural network. The model is trained on a labeled dataset of sign language gestures, allowing it to learn and recognize different signs.

4.2.5 Output

The machine learning model outputs a prediction of the sign being made in the input video or images. This prediction can be used to generate text or audio output, allowing individuals who do not understand sign language to access the information being communicated.

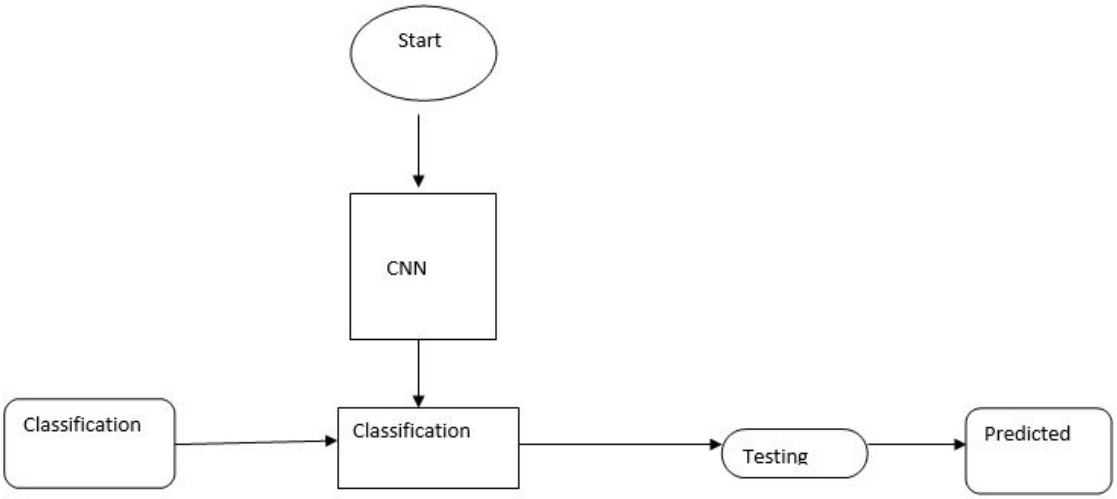
4.2.6 Evaluation

The performance of the system is evaluated using metrics such as accuracy, precision, and recall. The system can be fine-tuned and improved based on these results

4.3 System Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. A system architecture can consist of system components and the sub-systems developed that will work together to implement the overall system. The below figure shows the system architecture for the proposed system. The input image is preprocessed and

converted to gray scale image to get the clear vision of the image. Then it will be converted into binary values. In the next step identify the part which needs to proceed further. Then required features are extracted by In the CNN convolution layer. By passing those features into different layers of CNN we get a compressed image, that feature is used for detection of sign language.



4.3 System Architecture

4.4 Data flow diagram

A data flow diagram (DFD) is a graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context level DFD first which shows the interaction between the system and outside entities.

Data flow diagrams show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage.

There are only four symbols: Squares representing external entities, which are sources and destinationsof entering and leaving the system, Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it, Arrows representing the data flows, which can either be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly, The flat three-sided rectangle representing data stores should both receive information for storing and provide it for further processing, It is also used to analyze a

particular problem and the solution for it in steps, A user loads the data and the system reads the data provided by the user and Based on feature extraction and classifier the model will be trained and tested.

4.4.1 Data Flow Diagram for Pre-processing

The figure shows that the image is given as input. As we give the color image so that RGB image is converted into gray scale values to reduce complexity in the image. For efficient feature extraction gray scale values are converted into binary values. Then the image with reduced complexity is sent to the next process.

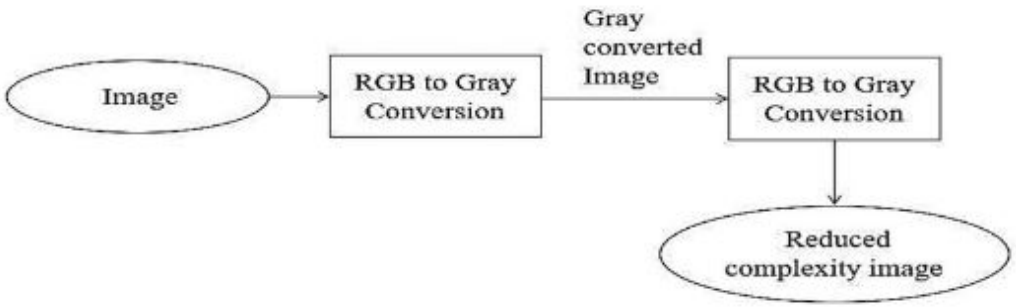


Fig 4.4.1 Dataflow diagram for Preprocessing

4.4.2 Data Flow Diagram for Identification

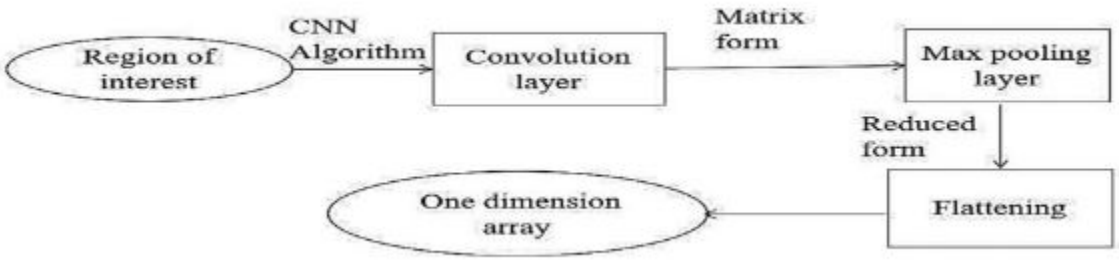


Fig 4.4.2 Dataflow diagram for identification

The figure shows that the image with reduced complexity is considered as input. Here the region with the value of one is considered as black that region is considered for the next process

4.4.3 Data Flow Diagram for Feature Extraction

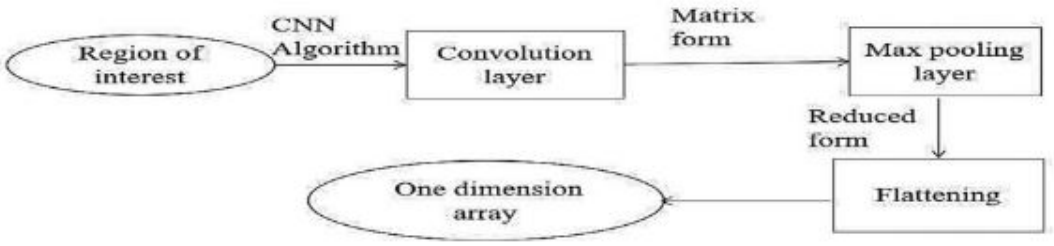


Fig 4.4.3 Dataflow diagram for feature extraction

The figure shows that the region of interest from the identification step is considered as input. The region of interest is obtained from converting RGB color image to the gray scale image by using minimax scalar method. For that region CNN algorithm is applied.

A CNN consists of an input layer and an output layer, as well as multiple hidden layers between them. The hidden layer basically consists of the convolution layer, pooling layer, relu layer and fully connected layers.

In this the RGB color image is converted into grayscale image by using minimax scalar method. The binary valued image is given as input to the convolution layer. In the convolution layer binary matrix is multiplied with a filter to extract features from the region.

4.4.4 Data Flow Diagram for Classification and Detection

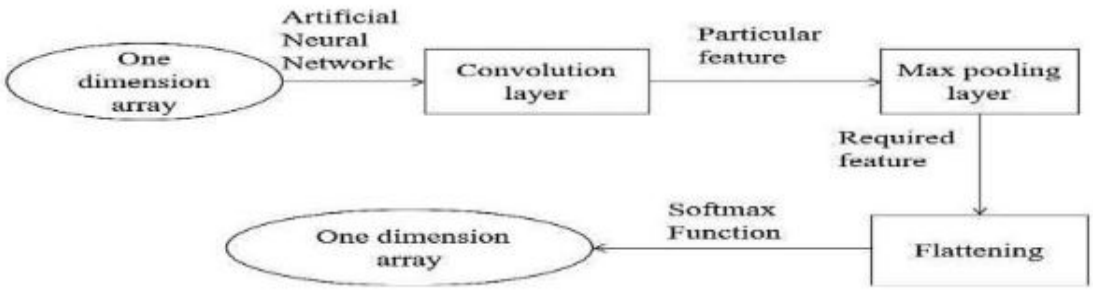


Fig 4.4.4 Dataflow diagram for Classification and detection

The figure shows that the one-dimension array is sent to a fully connected layer of CNN.Artificial neural network method is applied to this layer. Firstly, a one-dimension array

is sent to the input layer. Some particular feature which is required for the detection is identified by the hidden layer of ANN. The continued connection from hidden layer to output layer will help to identify accurate results. By considering all the features, the output layer gives the result with some predictive value. These values are calculated by using SoftMax activation function.

4.5 Activity Diagram

Here the preprocessing of the image by converting the RGB to grayscale image and feature extraction is done by the first layer that is the convolution layer of the neural network and detection done by using fully connected layers of the convolutional neural network.

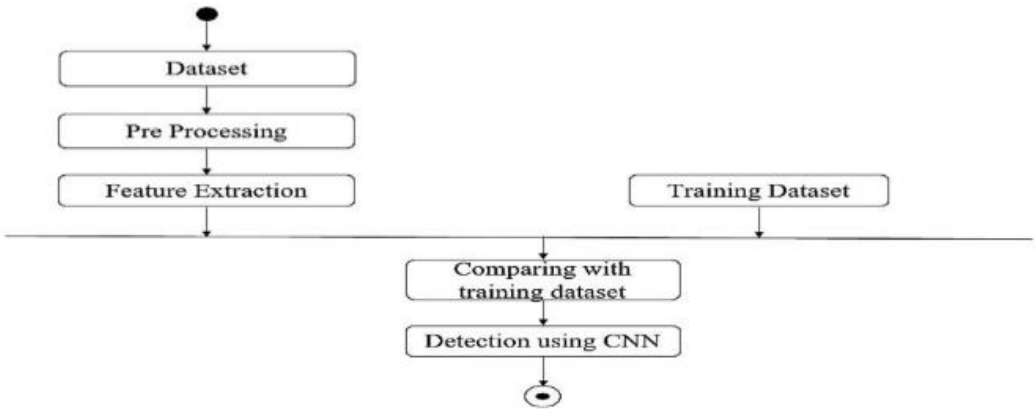


Fig 4.5 Activity diagram

4.6 Sequence diagram

Here the image dataset is given and pre-processing of the image is done. The processed data is given to feature extraction and here comparing and testing of image is done and by applying CNN algorithm the detection is done.

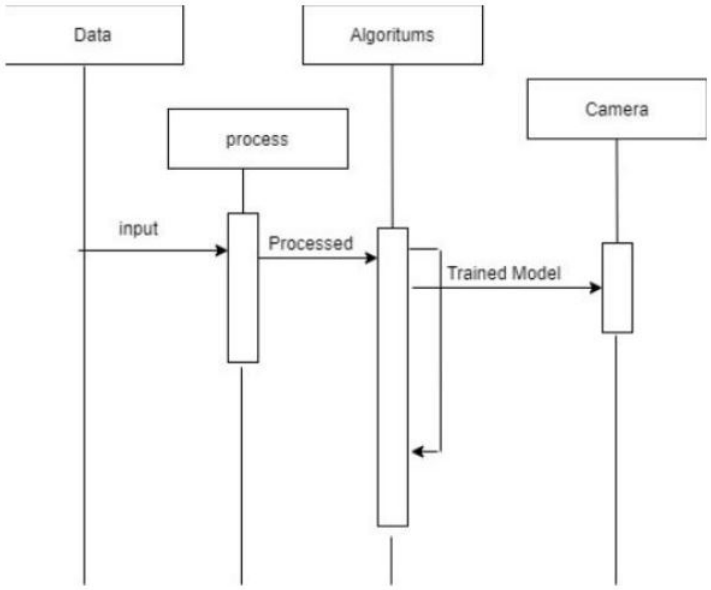


Fig 4.6 Sequence Diagram

Why TensorFlow?

- **Easy model building**: TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.
- **Robust ML production anywhere**: TensorFlow has always provided a direct path to production. Whether it's on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what language or platform you use.
- **Powerful experimentation for research**: TensorFlow gives you the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.
- **There is a TensorFlow for web, mobile, edge, embedded and more**: TensorFlow provides a range of services and modules within their existing ecosystem making them as one of the ground-breaking end-to-end tools to provide state-of-the-art deep learning.
- **TensorFlow Lite for mobile and embedded ML**: It is a TensorFlow lightweight solution used for mobile and embedded devices. It is fast since it enables on-device machine learning inference with low latency. It supports hardware acceleration with the Android Neural Networks API.

However, Tensorflow is not that user-friendly and has a steeper learning curve. To solve that, the high-level Keras API of Tensorflow provides building blocks to create and train deep learning models more easily. Also, Keras models are made by connecting configurable building blocks together with few restrictions. It makes it more modular and composable.

To do a simple classification using High-Level Keras API, some of the steps to be followed are:

1. Importing required modules.
2. Preparing the data to a suitable format for the Apis.
3. Building the neural network model using tf.keras (Tensorflow-Keras) APIs and compiling it.
4. Training the model with the prepared data while trying to resolve both under-fit and over-fit scenario.
5. Evaluating the model.
6. Saving and restoring of models to use it in production.

4.7 Low Level Design

Low Level Design in short LLD is like detailing HLD means it refers to component-level design process. It describes detailed description of each and every module means it includes actual logic for every system component and it goes deep into each modules specification. It is also known as micro level/detailed design. It is created by designers and developers. It converts the High-Level Solution into Detailed solution. It is created second means after High Level Design.

In order to convert High Level Design to Low Level Design we need to follow the following steps-

1. Import TensorFlow.
2. Download and prepare the dataset.
3. Verify the data.
4. Create the convolutional base.
5. Add Dense layers on top.
6. Compile and train the model.
7. Evaluate the model.

CHAPTER 5

IMPLEMENTATION

5.1 Overview of System Implementation

Computer vision is a rapidly growing technology that is set to revolutionize healthcare. The technology leverages powerful artificial intelligence algorithms with optical sensors and cameras. As a result, computer vision can help doctors, and medical professionals quickly identify diseases, provide an accurate diagnosis, offer personalized treatments for patients, monitor medication use, and even predict health outcomes.

Computer vision focuses on image and video understanding. It involves tasks such as object detection, image classification, and segmentation. Medical imaging can greatly benefit from recent advances in image classification and object detection. Research studies have demonstrated promising results in complex medical diagnostics tasks spanning dermatology, radiology, or pathology.

With the help of libraries of python like TensorFlow, we present and implement a conditional generative adversarial network-based model for real-time sign language detection. The process involves pre-processing the video frames, such as cropping and resizing, to ensure uniformity and then feeding them into the CNN for feature extraction. The extracted features are then passed through a classifier to recognize the sign language gesture. Features are extracted from the input video frames to capture the spatial information of hand movements and gestures. The CNN architecture uses convolutional layers to learn and extract these features.

5.2 Algorithm

1. Initialize the weights and biases of the network randomly.

- Let `W` be the weights matrix of size `(n_c, n_h, n_w)`, where `n_c` is the number of filters, `n_h` is the filter height, and `n_w` is the filter width.
- Let `b` be the bias vector of size `(n_c,)`.
- Initialize `W` and `b` with random values.

2. Feed the input data `X` into the network.

- `X` is a tensor of size `(n, c, h, w)`, where `n` is the batch size, `c` is the number of channels, `h` is the height, and `w` is the width.

3. Perform a convolution operation between the input data and the filters of the first convolutional layer.

- Let `Y` be the output tensor of size `(n, n_c, out_h, out_w)`, where `out_h` and `out_w` are the output height and width, respectively.
- The convolution operation is defined as follows:

$$Y[i, j, p, q] = \sum_k \sum_u \sum_v (X[i, k, p*u + r, q*v + s] * W[j, k, r, s])$$

where `i` is the index of the batch, `j` is the index of the filter, `p` and `q` are the spatial coordinates of the output feature map, `u` and `v` are the spatial coordinates of the input feature map, and `r` and `s` are the spatial coordinates of the filter.

4. Add a bias term to the result of the convolution operation.

- The bias operation is defined as follows:

$$Y[i, j, p, q] \mathrel{+}= b[j]$$

5. Apply an activation function (such as ReLU) to the output of the first layer.

- The ReLU activation function is defined as follows:

$$Y = \max(0, Y)$$

6. Perform pooling (such as max pooling) on the output of the activation function to reduce the spatial dimensions of the output.

- Let `Z` be the output tensor of size `(n, n_c, pool_h, pool_w)`, where `pool_h` and `pool_w` are the pooled height and width, respectively.
- The max pooling operation is defined as follows:

Z[i, j, p, q] = max(Y[i, j, p*stride_h : p*stride_h + pool_h, q*stride_w : q*stride_w + pool_w])

where `stride_h` and `stride_w` are the stride sizes.

7. Repeat steps 3-6 for each subsequent convolutional layer.

8. Flatten the output tensor of the last convolutional layer into a vector.

- Let `V` be the flattened vector of size `(n, n_c*pool_h*pool_w)`.

9. Pass the flattened vector through one or more fully connected layers.

- Let `W_f` be the weight matrix of size `(m, n_c*pool_h*pool_w)`, where `m` is the number of neurons in the fully connected layer.
- Let `b_f` be the bias vector of size `(m,)`.
- The fully connected layer operation is defined as follows:

```
U = W_f @ V.T + b_f
```

5.3 Pseudocode

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16,(3,3),activation = "relu" , input_shape = (180,180,3)) ,
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32,(3,3),activation = "relu") ,
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation = "relu") ,
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128,(3,3),activation = "relu"),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(550,activation="relu"),      #Adding the Hidden layer
    tf.keras.layers.Dropout(0.1,seed = 2019),
    tf.keras.layers.Dense(400,activation ="relu"),
    tf.keras.layers.Dropout(0.3,seed = 2019),
    tf.keras.layers.Dense(300,activation="relu"),
    tf.keras.layers.Dropout(0.4,seed = 2019),
    tf.keras.layers.Dense(200,activation ="relu"),
    tf.keras.layers.Dropout(0.2,seed = 2019),
    tf.keras.layers.Dense(26,activation = "softmax")   #Adding the Output Layer
])

model.summary()
```

Fig 5.3.1 Training the model

Fig 5.3.1 shows how the model is being trained.We use Keras which is an open-source high-level neural network API (Application Programming Interface) written in Python. It is designed to be user-friendly, modular, and easy to use, allowing developers to build and train deep learning models with minimal coding. Specific parameters has been set for each layer.

```
32  from tensorflow.keras.optimizers import RMSprop,SGD,Adam
33  adam=Adam(lr=0.001)
34  model.compile(optimizer='adam', loss='categorical_crossentropy', metrics = ['acc'])
35
36  bs=30         #Setting batch size
37  train_dir = "D:/abhiar/sign_speech/S5 speech/dataset/training_set/"   #Setting training directory
38  validation_dir = "D:/abhiar/sign_speech/S5 speech/dataset/test_set/"   #Setting testing directory
39
```

Fig 5.3.2 Adam optimizer

Fig 5.3.2 shows the snippet of the code where TensorFlow framework is being used and the optimizers imported re RMSprop, SGD and Adam. Adam optimizer has been taken for better performance of the program and the learning rate has been set to 0.1. The model is then compiled by setting the necessary parameters mentioned in the above diagram.

```
40  from tensorflow.keras.preprocessing.image import ImageDataGenerator
41
42  # All images will be rescaled by 1./255.
43  train_datagen = ImageDataGenerator( rescale = 1.0/255. )
44  test_datagen  = ImageDataGenerator( rescale = 1.0/255. )
45  train_generator=train_datagen.flow_from_directory(train_dir,batch_size=bs,
46              class_mode='categorical',target_size=(180,180))
47  # Flow validation images in batches of 20 using test_datagen generator
48  validation_generator =  test_datagen.flow_from_directory(validation_dir,
49                                                           batch_size=bs,
50                                                           class_mode  = 'categorical',
51                                                           target_size=(180,180))
52
53  history = model.fit(train_generator,
54                      validation_data=validation_generator,
55                      steps_per_epoch=150 // bs,
56                      epochs=2,
57                      validation_steps=50 // bs,
58                      verbose=2)
59  y_pred = np.asarray(model.predict(validation_generator))
60  y_true = validation_generator[1]
61  y_pred_class = np.argmax(y_pred, axis=1)
62  validation_generator=validation_generator[:-13]
63  print(validation_generator)
64
```

Fig 5.3.3 Testing the model

In Fig 5.3.3 the model which was trained is now being tested in this code snippet. The epochs value has been set to 2 because the results turned out to be efficient. A lower epochs value would iterate the training model less number of times and a higher epochs value would train the model more number of times, but if a higher epochs number is used, there is a possibility of the model to overfit and hence might produce poor results if a new dataset has been inputted to it.

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from keras_preprocessing.image import load_img, img_to_array
image_x, image_y = 64, 64
lb = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
      'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
classifier = load_model('model.h5')
cap = cv2.VideoCapture(0)
```

Fig 5.3.4 Image Prediction

Fig 5.3.4 represents a code snippet where the final image prediction takes place. The libraries imported are cv2,numpy, tensorflow and keras. The entire input image has been rescaled to a size of 64x64 and a list of labels i.e. alphabets are initialized.

```
def Label(result):
    if result[0][0] == 1:
        return 'A'
    elif result[0][1] == 1:
        return 'B'
    elif result[0][2] == 1:
        return 'C'
    elif result[0][3] == 1:
        return 'D'
    elif result[0][4] == 1:
        return 'E'
    elif result[0][5] == 1:
        return 'F'
    elif result[0][6] == 1:
        return 'G'
    elif result[0][7] == 1:
        return 'H'
    elif result[0][8] == 1:
        return 'I'
    elif result[0][9] == 1:
        return 'J'
    elif result[0][10] == 1:
        return 'K'
    elif result[0][11] == 1:
        return 'L'
    elif result[0][12] == 1:
        return 'M'
    elif result[0][13] == 1:
        return 'N'
    elif result[0][14] == 1:
        return 'O'
    elif result[0][15] == 1:
        return 'P'
    elif result[0][16] == 1:
        return 'Q'
    elif result[0][17] == 1:
        return 'R'
    elif result[0][18] == 1:
        return 'S'
    elif result[0][19] == 1:
        return 'T'
    elif result[0][20] == 1:
        return 'U'
    elif result[0][21] == 1:
        return 'V'
    elif result[0][22] == 1:
        return 'W'
    elif result[0][23] == 1:
        return 'X'
    elif result[0][24] == 1:
        return 'Y'
    elif result[0][25] == 1:
        return 'Z'
```

Fig 5.3.5 Letter prediction

Before the threshold image is captured, the CNN architecture performs filtering and extracts the features from the grayscaled image and produces a threshold image. This threshold image is then compared with the datasets given for each alphabet and the required sign language has been detected.

```
while True:
    ret, frame = cap.read()
    img = cv2.rectangle(frame, (300, 300), (10, 10), (0, 255, 0), 0)
    crop_img = img[10:300, 10:300]
    gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(
        gray, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)[1]
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord(' '):
        img = cv2.resize(thresh, (64, 64))
        img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
        test_image = img_to_array(img)
        test_image = np.expand_dims(test_image, axis=0)
        print(test_image.shape)
        #result = np.argmax(classifier.predict(test_image))
        result = classifier.predict(test_image)
        print(result)
        # label=lb[int(result)]
        # print(label)
        output_lb = Label(result)
        print(output_lb)
        frame1 = frame.copy()
        cv2.putText(frame1, output_lb, (300, 25),  cv2.FONT_HERSHEY_SIMPLEX,
                    0.9, (0, 255, 255), 2)
    cv2.imshow('thresh', thresh)
    cv2.waitKey(1)

cv2.destroyAllWindows()
```

Fig 5.3.6 Image Processing Function

The above code snippet shows the image processing function where the image has been resized accordingly and been converted to a grayscale image. The threshold image is obtained by setting the appropriate parameters mentioned in the above code snippet. Image is then tested and the dimensions are set appropriately. The images are then sent for prediction of the appropriate letter shown by the sign language. The output is then finally generated.

5.4 Implementation Support

There are many features in Python, some of which are discussed below as follows:

- **Easy to code**: Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.
- **Free and Open Source**: Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open- source, this means that source code is also available to the public. So, you can download it as, use it as well as share it.
- **High-Level Language**: Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.
- **Extensible feature**: Python is an Extensible language. We can write some Python code into C or C++ language and also, we can compile that code in C/C++ language.
- **Python is Portable language**: Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

Basic syntax of Python

Python syntax can be executed by writing directly in the Command Line:

>>> print(“Hello, World!”) Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the **Command Line:**

C:\Users\Your Name>python myfile.py

Python Indentation

Indentation refers to the spaces at the beginning of a code line where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code.

Example:

```
if 5 > 2:
print("Five is greater than two!")
```

Python will give you an error if you skip the indentation.

Python Variables

In Python, variables are created when you assign a value to it. Example:

```
x = 5
y = "Hello, World!"
```

Python has no command for declaring a variable.

Python Comment

Python has commenting capability for the purpose of in-code documentation. Comments start with a #, and Python will render the rest of the line as a comment: Example:

```
#print ("Hello, World!")
```

Comparing TensorFlow, PyTorch and Keras

	Keras	PyTorch	TensorFlow
API Level	High	Low	High and Low
Architecture	Simple, concise, readable	Complex, less readable	Not easy to use
Datasets	Smaller datasets	Large datasets, high performance	Large datasets, high performance
Debugging	Simple network, so debugging is not often needed	Good debugging capabilities	Difficult to conduct debugging
Does It Have Trained Models?	Yes	Yes	Yes
Popularity	Most popular	Third most popular	Second most popular

Fig 5.4 Features of different python libraries

CHAPTER 6

TESTING

Testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing an transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested

Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, post-condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements.

6.1 White Box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. it is a test case design method that uses the control structure of the procedural design to derive test cases.

6.2 Black Box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

6.3 Testing strategies

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

Any software testing strategy should have the following characteristics:

- Testing begins at the module level and works outward toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.

Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

6.4 Unit Testing

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not.

Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application.

6.5 System Testing

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing.

System testing is carried out for our project where the models have been trained and integrated with other modules. OpenCV models were given 6 images for testing and the below images are produced as the result.

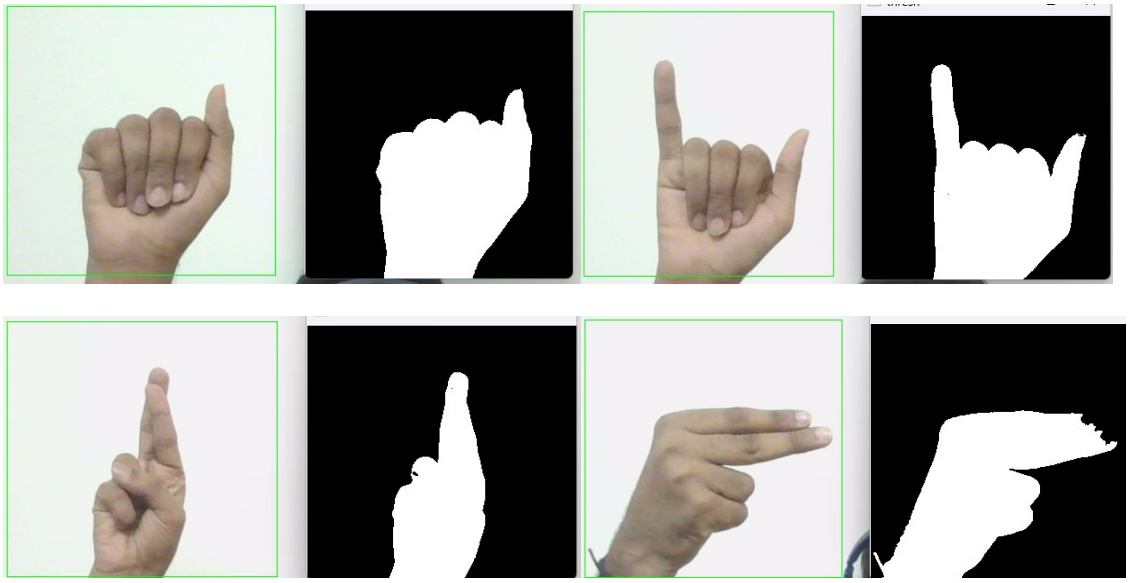


Fig 6.5 Outputs for OpenCV model

6.6 Validation Testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers.

Reasonable expectation is defined in the software requirement specification- a document that describes all user visible attributes of the software. The specification contains a section title "validation criteria". Information contained in that section forms the basis for validation testing approach.

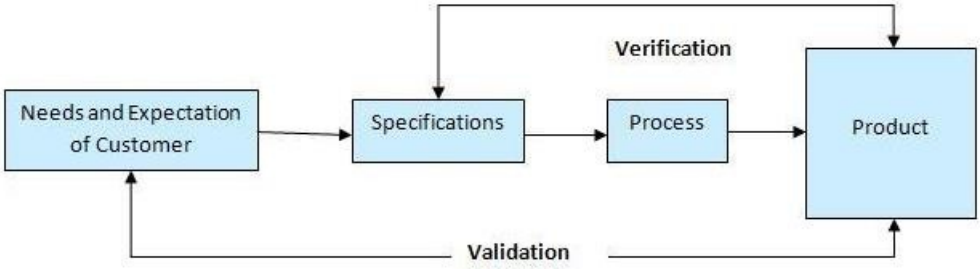


Fig 6.6 Software verification and validation

6.7 Integration Testing

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integration testing:

- *Top-Down Integration*: Top down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards throw the control hierarchy beginning with the main control module.

- *Bottom-Up Integration*: Bottom up integration as its name implies, begins Construction and testing with automatic modules.

- *Regression Testing*: In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagated unintended side effects.

6.8 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Table 6.8: Functional Testing

Input Images	Input must be accepted.
Train	Model analysis the models and trains it and then saves the model.
Tesing	Yes, pass.
Output	Yes, the model predicted, and Speech.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.9 Alpha Testing

It is virtually impossible for a software developer to forsee how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time.

If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer's site. The software is used in a natural setting with the developer "Looking over the shoulder" of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

6.10 Beta testing

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

6.11 System Testing and Acceptance Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc.

CHAPTER 7

DISCUSSION OF RESULTS

This section discusses the results from each module implemented.

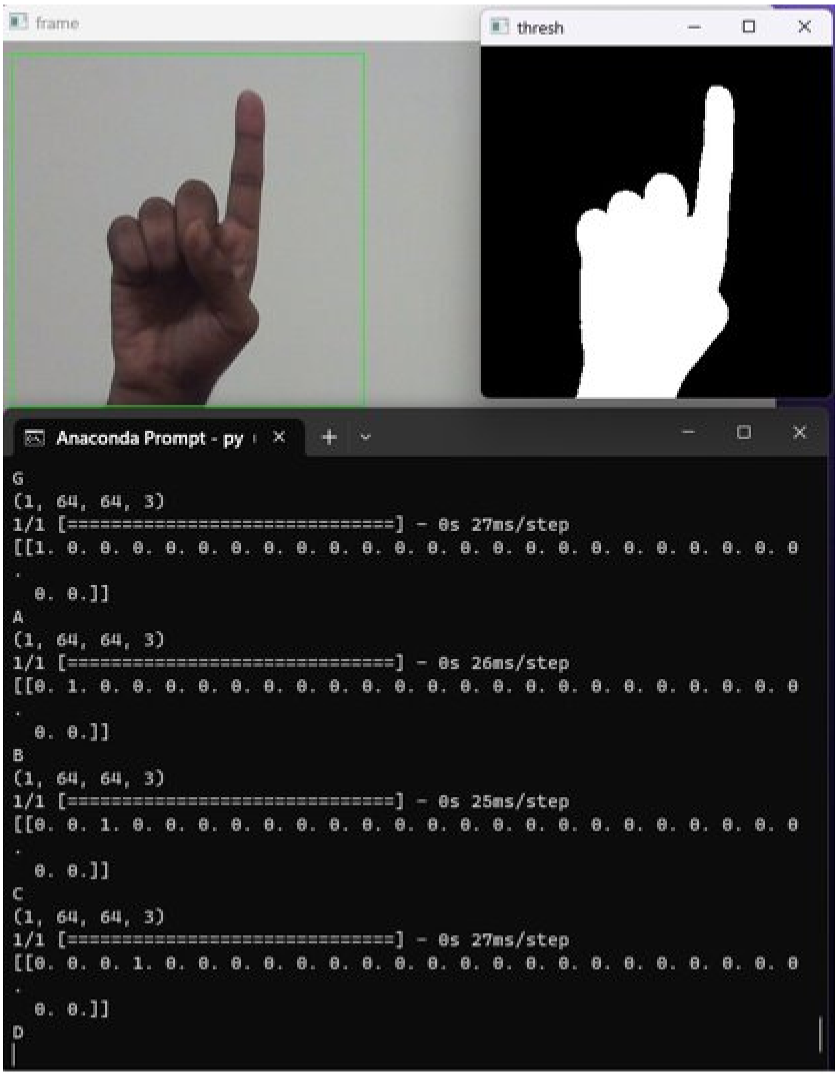


Fig 7.1 Detection of the letter D

Fig 7.1 Shows the results of the sign language detection which is presented in the green box. The user must present the sign language in the given green frame and a threshold image is extracted from the input sign language and model is made to run and identify the labels.

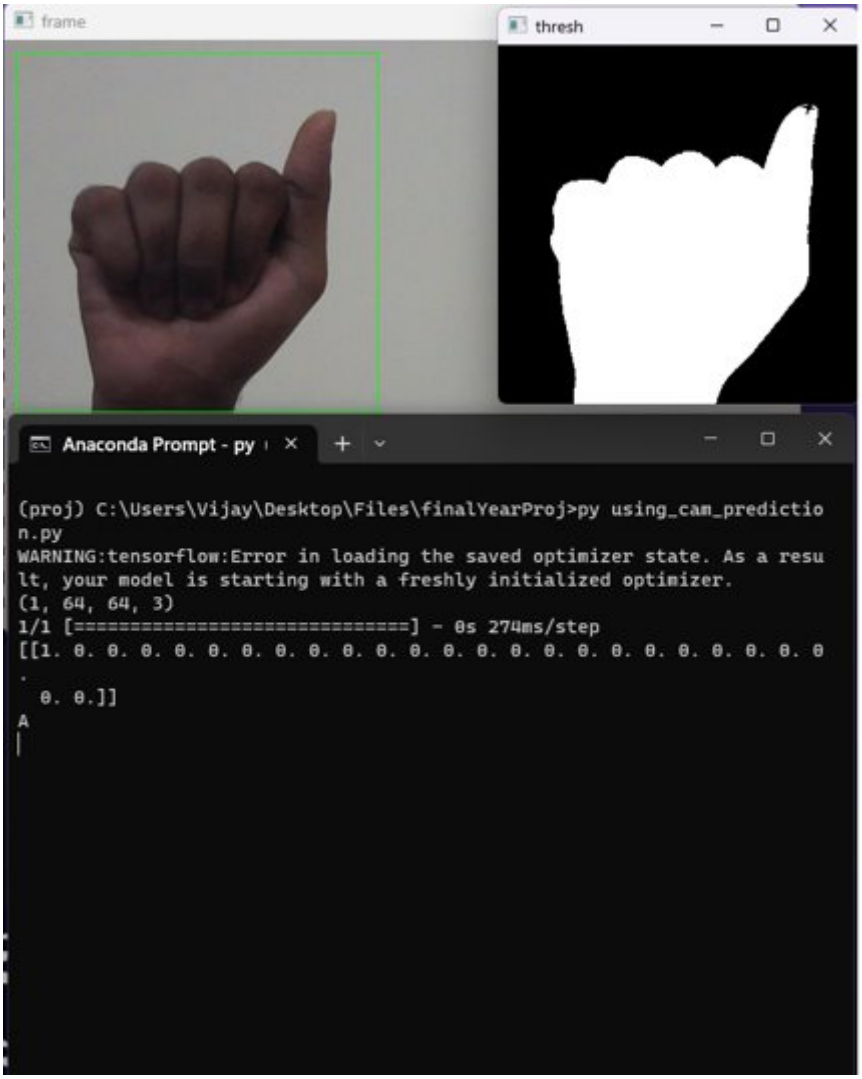


Fig 7.2 Detection of the letter A

Fig 7.2 depicts the identification of the letter A. The disabled user is made to present the sign language in the screen and the CNN interprets this sign language and produces the desired results. The array in the terminal screen represents the position of the alphabet . The entire image is then resized to 64x64 resolution.

Similarly another sign language from the ASL is detected by showing the hand gesture in the frame and is interpreted by our model which is then trained and tested to produce the desired results. The letter V is interpreted by extracting the necessary features from the input image and the alphabet is displayed.

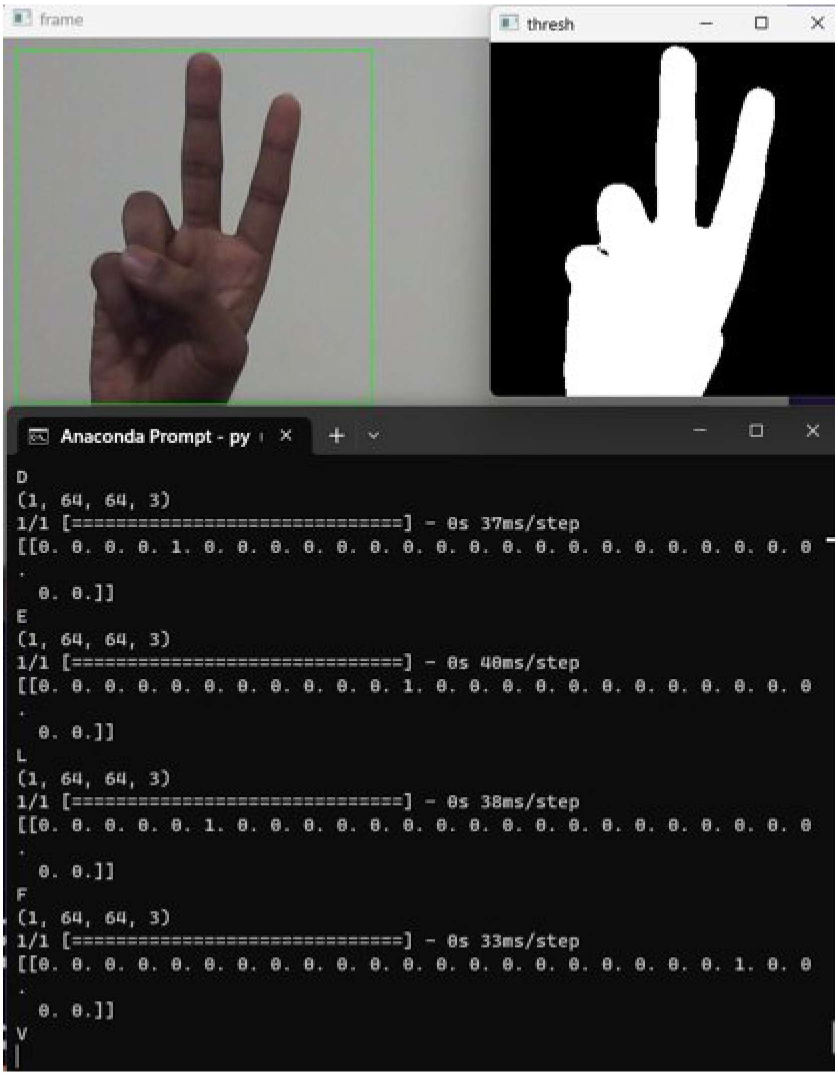


Fig 7.4 Detection of letter V

CHAPTER 8

CONCLUSION

Convolution Neural Networks (CNNs) are being utilized in this work to recognize alphabets in sign language. Sign language is a visual language that communicates using gestures, facial expressions, and body language.

The CNN method is used to recognize and categories sign language hand motions, which may subsequently be mapped to particular letters of the alphabet.
Machine learning is an artificial intelligence area that includes training algorithms to make predictions or judgements based on data patterns.

In this scenario, the CNN algorithm is being trained on a dataset of sign language motions and their accompanying alphabet letters to learn how to recognize and categorize new movements.

CHAPTER 9

FUTURE ENHANCEMENTS

- Currently, most sign language recognition systems focus on recognizing individual gestures. By incorporating NLP, the system could interpret the meaning of entire sentences or phrases, making it more effective in facilitating communication.

- One way to improve the accuracy of the CNN model is by using data augmentation techniques to increase the variety of training data. This would help the model to recognize more subtle differences in hand gestures and improve its overall accuracy.

- The system's user interface could be improved by incorporating a more intuitive and user-friendly interface, such as using virtual or augmented reality to enhance the user's experience.

CHAPTER 10

REFERENCES

[1] Dašić, P., Dašić, J., & Crvenković, B. (2017). Improving patient safety in hospitals through usage of cloud supported video surveillance. Open access Macedonian journal of medical sciences, 5(2), 101–106.

[2] Goran, S. F. (2010). A second set of eyes: an introduction to tele-ICU. Critical Care Nurse, 30(4), 46-55.

[3] Panlaqui, M., Broadfield, E., Champion, R., Edington, P., & Kennedy, S. (2017). Outcomes of telemedicine intervention in a regional intensive care unit: a before and after study. Anesthesia and intensive care, 45(5), 605-610.

[4] Rajagopalan, R., Litvan, I., & Jung, T. P. (2017). Fall prediction and prevention systems: recent trends, challenges, and future research directions. Sensors (Basel, Switzerland), 17(11), 2509.

[5] Su, L., Waller, M., Kaplan, S., Watson, A., Jones, M., & Wessel, D. L. (2015). Cardiac resuscitation events: one eyewitness is not enough. Pediatric Critical Care Medicine, 16(4), 335-342.

[6] Yagi, T., Koizumi, Y., Aoyagi, M., Kimura, M., & Sugizaki, K. (2005). Three-dimensional analysis of eye movements using four times high-speed video cameras. Auris Nasus Larynx, 32(2), 107-112.

[7] Haque, A., Guo, M., Alahi, A., Yeung, S., Luo, Z., Rege, A., ... & Platchek, T. (2017). Towards vision-based smart hospitals: A system for tracking and monitoring hand hygiene compliance.

[8] Yeung S, Rinaldo F, Jopling J, Liu BB, Mehra R, Downing NL, et al. A computer vision system for deep learning-based detection of patient mobilization activities in the ICU. Npj Digit Med. 2019;2..

[9] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10):1499–1503.