An Integrated Hand Gesturing Recognition Software With The American Sign Language Manuscript Using Convolution Neural Networks (CNN)

A Project report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ROBOTICS AND AUTOMATION ENGINEERING

Submitted by

Nouman Jinabade 21070127032

Under the guidance of

S. Satish Kumar

(ASSOCIATE PROFESSOR)

CONTENTS

ABSTRACT

Sign language is used by deaf and hard-hearing people to exchange information between their community and with society. It is the only tool of communication for such people, Sign language is a boon for physically challenged people to express their thoughts and emotion.

In this work, a novel scheme of sign language recognition has been proposed for identifying the alphabets, words, and gestures in sign language.

Computer recognition will deal with the sign gesture acquisition and continues till text/speech generation. Sign gestures can be classified as static and dynamic. Static gesture recognition is easier to implement than dynamic gesture recognition, but both forms of recognition have significant value to society.

This survey outlines the steps involved in sign language recognition, including data acquisition and preprocessing, exploratory data analysis, neural network implementation, feature extraction, classification, and the results obtained. Additionally, future research directions in this field are proposed.

1. INTRODUCTION

People with speech impairments communicate using hand signs and gestures, but it can be challenging for non-impaired individuals to comprehend. Therefore, there is a need for a system that can recognize and translate these gestures for the benefit of both parties and close the communication gap between the two groups.

Image processing refers to the process of performing operations on an image to improve it or extract relevant information. It is a form of signal processing where an image is the input and the output may be the image itself or its characteristics. This technology is rapidly advancing and is a central focus of research in engineering and computer science. The process of image processing consists of three steps:

1. Importing/acquiring the frame via image acquisition tools.
2. Analysing and manipulating the image.
3. Producing the final result, which can be either the improved image or a report based on the image analysis.

Image processing can either be done through analogue or digital methods. Analogue processing is applied to physical copies such as prints or photos, and involves image analysis using visual techniques. Digital processing, on the other hand, involves manipulating digital images with the use of computers. This process is broken down into three phases: pre-processing, enhancement, and display/information extraction.

We would be discussing 'Digital Image Processing' under the scope for this part of the survey as that is what is required for this imperative software :

1.1 Digital Image Processing

Digital image processing involves using computer algorithms to manipulate images. In recent years, its usage has skyrocketed and it has a wide range of applications, from healthcare to entertainment and even in areas like geology and remote sensing. Digital image processing plays a crucial role in the modern information society, particularly in multimedia systems.

Digital image processing involves manipulating numerical representations of images. In this the image goes throw the following process's

1. Enhancement of the Image
2. Restoration of the key points in the Image
3. Analysis of the resultant
4. Compression of all

Enhancement involves modifying an image using heuristic methods to make it easier for a person to understand. Further using computer algorithms to process images. It is defined as applying a series of operations to a numerical representation of an object to achieve a specific outcome. Processing also involves converting physical images into digital form and using algorithms to extract important information from the digital image.

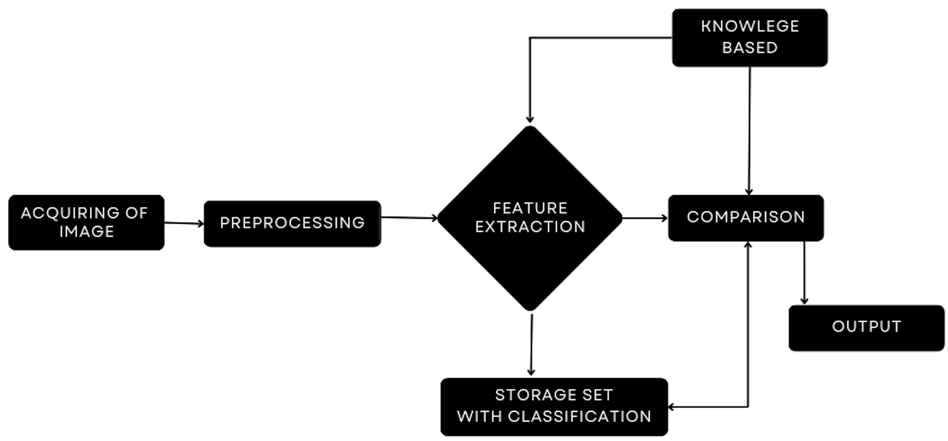


Fig1.1: Summary for 'Digital Image Processing'

1.1.2 Pattern Recognition

Pattern recognition-based image processing- involves separating objects from images using pattern recognition techniques and then using statistical decision theory methods to identify and categorize them. When an image contains multiple objects, pattern recognition is divided into three stages, as depicted in the figure.

The three phases of pattern recognition are:

1. Image segmentation and object separation where objects are separated from the background
2. Feature extraction where objects are quantitatively measured and described with a feature vector
3. Classification where objects are categorised based on the feature vector.

The input for this process is an image and the output is the object type and structural analysis of the image, which describes the important information in the image.

1.2 Sign Language

Sign language is a form of communication that uses gestures, facial expressions, and body postures to convey meaning. It is primarily used by deaf and mute individuals. Different countries have their own sign languages, such as British Sign Language (BSL), Indian Sign Language (ISL), and American Sign Language (ASL), which are not mutually intelligible.

The goal of a sign language recognition system is to facilitate communication between signers and non-signers without the need for an interpreter.

1.3 Sign Language With Hand Gesture Recognition

Sign language recognition is the process of translating sign language into text, providing a communication bridge between non-speaking individuals and the general public. This is achieved through the use of image processing algorithms and neural networks, which map the gestures to corresponding text based on training data. As a result, raw images or videos are transformed into readable text. Individuals with limited cognitive abilities often struggle with social communication due to their difficulty in understanding and using common gestures and expressions.

1.4 Motivation

The 2011 Indian census recorded approximately 1.3 million individuals with hearing impairment, while the National Association of the Deaf in India estimated that 18 million people, or roughly 1% of the Indian population, are deaf. This discrepancy inspired our project, which aims to provide a communication solution for speech impaired and deaf individuals. As not all individuals can understand sign language, our project focuses on converting sign language gestures into text that can be easily comprehended by those without speech impairments.

1.5 Problem Statement

Speech-impaired individuals communicate using hand gestures and signs, which can be difficult for non-impaired individuals to understand. To address this, there is a need for a system that can recognize these gestures and convey the information to non-impaired individuals, bridging the communication gap between these two groups. In this evolution of tech in the 21st century a collinear voice between all humans is very essential ,this software would atleast give such impaired individuals a voice which could to be heard.

1.6 Organisation of paper

1. Introduction of technologies studied, statement of problem and motivation for project.
2. Literature survey outlining prior works and technologies used for Sign Language Recognition.
3. Detailed explanation of methodologies, including architecture and algorithms used.
4. Presentation of project designs.
5. Experimental analysis, code, and results obtained.
6. Conclusion of project and discussion of potential future extensions.

2. LITERATURE

2.1 Application Areas Of Hand Gesture In Different Papers :

1. Hand gesture recognition systems have a wide range of applications in different domains such as sign language translation, virtual environments, smart surveillance, robot control, and medical systems.

2. Sign language recognition has received special attention due to its importance in facilitating communication for people with hearing impairments. Numerous systems have been proposed to recognize gestures using different sign languages such as American Sign Language (ASL), Japanese Sign Language (JSL), and Arabic Sign Language (ArSL) using various techniques such as boundary histogram, neural networks, and dynamic programming matching.

3. Hand gesture recognition is also used in robot control, where gestures are used to give commands to the robot to perform specific tasks. For example, a system was proposed that uses finger counting gestures, where each sign represents a specific function such as "move forward" or "stop." In virtual environments, hand gesture recognition is used to interact with virtual objects or control the virtual environment.

4. Hand gesture recognition is used to interact with virtual objects or control the virtual environment. One popular application of hand gesture recognition systems is in virtual environments (VEs) in especially for communication media systems. roposed system uses 3D pointing gesture recognition from binocular views for natural human-computer interaction (HCI) in real-time, which is accurate and independent of user characteristics and environmental changes.

5. Another recent application of hand gestures is recognizing numbers, where an automatic system was proposed to isolate and recognize Arabic numbers 0-9 in real-time using Hidden Markov Models (HMM).

6. Hand postures and gestures are also used for controlling television devices, such as turning the TV on and off, increasing and decreasing the volume, muting the sound, and changing the channel using open and close hand gestures.

7. In building 3D modeling, hand shape determination is needed to create and view 3D shapes of the hand. Some systems use hand silhouettes to build 2D and 3D objects, while 3D hand modeling is still a promising field of research.

2.2 Existing work : Summary of Research Results

During our literature review, we studied previous projects related to sign language recognition. The tables presented offer condensed information on various hand gesture recognition systems. Table 1 compares the recognition techniques used, while Table 2 provides an overview of the extraction methods, feature representations, and recognition approaches employed in the selected systems, including the hand extraction technique, features vector representation, and recognition method.

Table 1 : A comparison was made between different recognition methods

Method (Cit No.)	Captured gestures	Gestures Used In Train / Test	Database	Recognition %
[5]	5 v 12 (static/ dynamic)	240 trained	Self Generated	98.3% cumulative gestures
[2]	26	208	ASL	92.18%
[3]	6	60	Self Generated	100% for more than 4 gestures
[4]	20	200 trained	Self Generated	100% to 90% for different gestures
[1]	6	60	Self Generated	Without scaled : 84% After scaled (using normalization) : 95%

Table 2 : A Brief On Extraction Method Vs Features Representation.			
Method (Cit No.)	Classifier	FVR	Extraction Mh.
[9]	EDM (Euclidean Distance Metric)	The system extracted a total of 78 features, which included both local and global features calculated using moments.	HSV
[7]	FCM (Fuzzy C-Means)	To create a feature vector for hand gesture recognition, 13 parameters are used. The first parameter indicates the ratio aspect of the bounding hand-box, while the remaining 12 parameters represent the average values of brightness pixels in the image.	HSV
[8]	LVQ (Linear Vector Quantization)	9 numerical characteristics are created by measuring the distance between the palm and each finger, as well as four angles formed by these distances.	HSV
[1]	ERN (Elman Recurrent Network)	The system uses either 13 or 16 data items, depending on the configuration. The 13 data items include 10 for bending and 3 for coordinate angles. The 16 data items include 10 for bending, 3 for coordinate angles, and 3 for positional data.	Thresholding
[6]	Gaussian Dist.	The hand shape is obtained using Self-Growing and Self-Organized Neural, followed by the calculation of three angles: RC Angle, TC Angle, and Distance from the Palm Center.	YCbCr

2.3 Drawbacks

In this section, limitations of several discussed methods are elaborated. The orientation histogram method utilized in [10] suffers from certain issues such as similar gestures having different orientation histograms and different gestures having similar orientation histograms. Moreover, this method performs well for any dominant object in the image, even if it is not a hand gesture. The Neural Network classifier used for gesture classification in [12][11] is time-consuming and requires a significant amount of time to classify data when the number of training examples increases. For instance, in [11], the NN took several hours to learn 42 characters and four days to learn ten words. Fuzzy c-means clustering algorithm employed in [6] has some drawbacks such as the problem of wrong object extraction if the objects are larger than the hand. The performance of the recognition algorithm deteriorates when the distance between the user and the camera is more than 1.5 meters. In addition, this algorithm is sensitive to variations in lighting conditions and unwanted objects may overlap with the hand gesture. The system proposed in [14] is vulnerable to changes in environmental lighting conditions, leading to erroneous segmentation of the hand region. Although HMM tools are suitable for recognizing dynamic gestures [13], they are computationally intensive.

2.4 Proposed System

The proposed system is a sign language recognition system that uses convolutional neural networks and LSTM to identify hand gestures and give out a sentance for the non-impaired to understand. The system works by capturing video, converting it into frames, segmenting the hand pixels, and comparing the resulting images with a pre-trained model. This is expected to lead to more accurate recognition of the sign language letters/words/poses.

3. METHODOLOGY

3.1 IMPORTING LIBRARIES

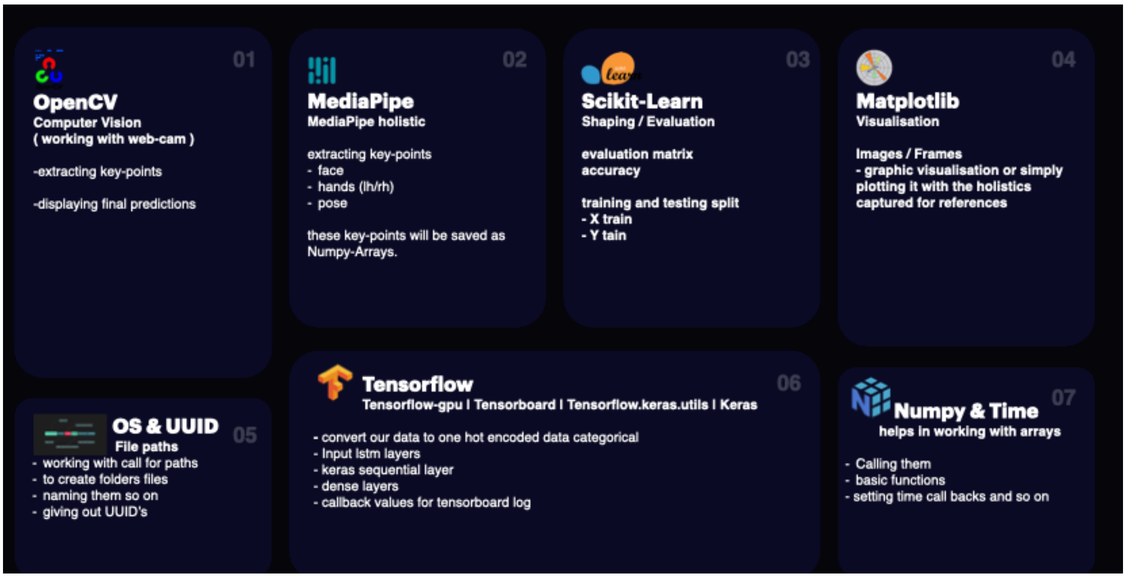


Fig. 3.1 Libraries With Summary

A Closer Look On Why OpenCV?

OpenCV (Open Source Computer Vision Library) is a library for programming functions mainly focused on real-time computer vision.
The library also includes a statistical machine learning library that includes algorithms such as boosting, decision tree learning, SVM, deep neural networks, etc.

Image processing

"Image processing is the analysis and manipulation of a digitized image to improve its quality or extract useful information."

A digital Image representation of a physical image as a finite set of digital values, called picture elements or pixels. Each pixel has a numerical value that represents the intensity of the corresponding point in the original image. The digital image is stored in a computer in a matrix form where each element represents the intensity of the corresponding pixel. The quality of a digital image is defined by its resolution, which is the number of pixels used to represent the image, and the color depth, which defines the number of bits used to represent the intensity of each pixel. In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

The essence of image processing involves three stages:

1. Acquisation of the image
2. Examining and modifying the image
3. The result, which can be either a transformed image or a report generated from analyzing the image.

3.2 FLOW OF METHODOLOGY :

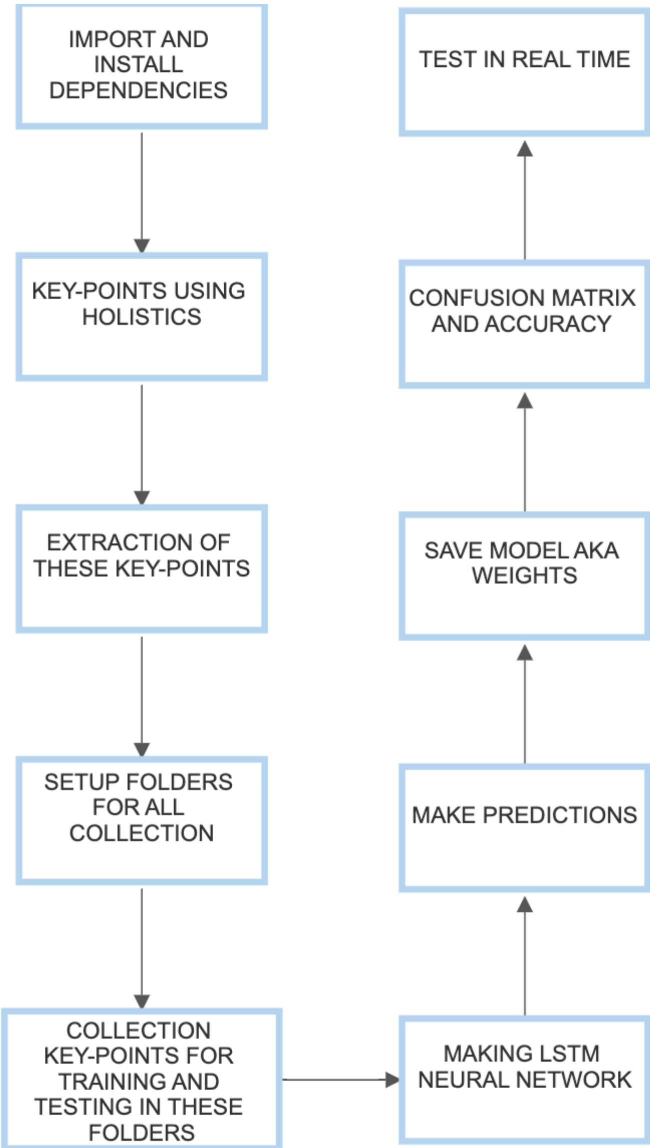


Fig. 3.2 Flowchart

3.3 DATA COLLECTION

Mediapipe provides an open-source cross-platform framework for processing various forms of perceptual data like video and audio. The framework includes multiple solutions such as face recognition and posture estimation. In this paper, we focus on mediapipe Hands for hand tracking.The X and Y coordinates are normalized by the bounding box's width and height, while the Z coordinate represents depth information, with a smaller value meaning closer to the camera and larger meaning further away, using the wrist as the origin. This method enables advanced hand and finger tracking with less processing and is suitable for low-performance environments like mobile devices.

Media pipe has two tasks in this program
a)Palm detection Model
b)Hand Landmark Model

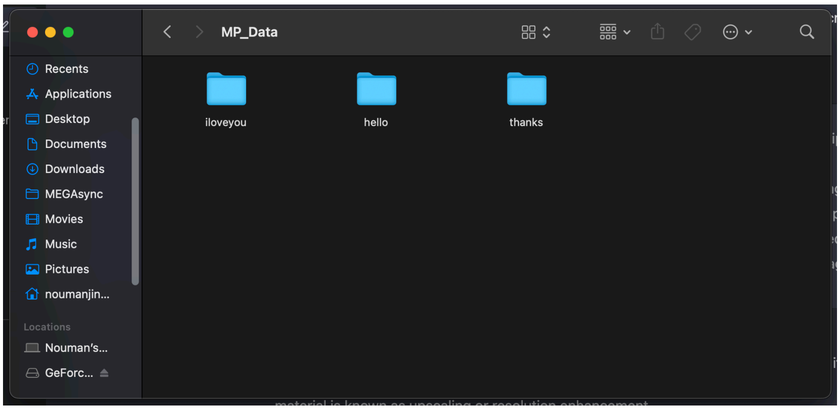
Detection Model

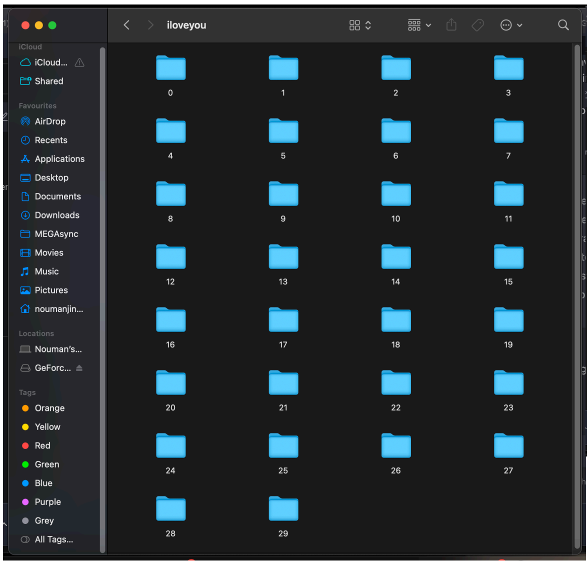
We use a single-shot detector model similar to Blaze Face, optimized for mobile real-time applications, to detect initial hand/face/pose locations. In detecting hands is a complex task as the model must handle a wide range of hand sizes and detect occluded and self-occluded hands. Unlike faces, hands lack high contrast patterns, making it difficult to detect them solely based on visual features. Our solution addresses these challenges by using different strategies. Firstly, we train a palm detector instead of a hand detector, as detecting rigid objects like palms and fists is easier than detecting hands with articulated fingers. Secondly, we use an encoder-decoder feature extractor for larger scene awareness, even for small objects. Lastly, we minimize focal loss during training to support a large number of anchors, resulting from the high scale variance. Detection of pose is very easy we just take reference points from the whole scheme of hands and face.

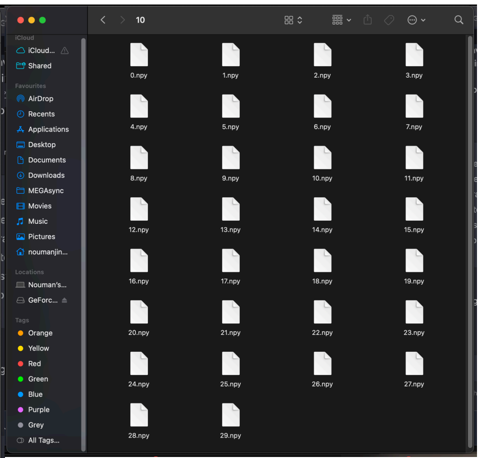
Landmark Model

After the detection is over the whole image our subsequent our landmark model precisely locates 3D hand-knuckle, face expression, and pose coordinates within the detected areas through regression, which is direct prediction of coordinates. The model learns a consistent internal representation of the hand/face/pose and is robust even with partially visible, blurred, or self-occluded hands. (Landmarks Model is also styled to draw these landmarks for a reference preview, this is done in a completely independent function)

DATASETS USED FOR TRAINING:







Figs 3.3 Datasets used for training the model

Sample pictures of training data

*(This dataset is pre collected in the code with a sequence of 30 frames with 30 cycles for each case 3 (as the trial run was of 3 cases) : **30 frames x 30 x 3 instances with 1662 points** = **4,487,400 array points**)*

CALCULATIONS FOR DATA COLLECTION :

our initial system is : 30 videos/sequences ,each will have 30 frames in length , each with 3 actions, multiplied by 1662 key-points

which will make our np array system as: 90 videos with 30 frames each with 1662 capture points in a single array

3.4 TRAINING AND TESTING MODULE

In supervised machine learning, the model is trained by providing both input and desired output data. The process involves several steps including:

1. model construction
2. model training (functions (def) are written before construction so some part of training such as landmark model and basic structure are the start for this software)
3. model testing
4. model evaluation

✳ **Model Construction**

The project relies on machine learning algorithms, specifically neural networks. The process of building a neural network model using Keras involves:

1. Initializing the model using: model = Sequential()

“ **Preprocessing for this step:**

This code seems to be for preparing data for training a machine learning model. The code:

- **Creates two lists: sequences and labels.**
- **Loops through the actions in the actions list and then through each sequence in a directory.**
- **Loads the data for each frame in a sequence and appends it to the window list.**
- **Appends the window list to the sequences list and appends the corresponding label to the labels list.**
- **Imports the Sequential class from the tensorflow.keras.models module, which is used for creating a sequential model in TensorFlow's Keras API.**

2. Adding different layers to the model :

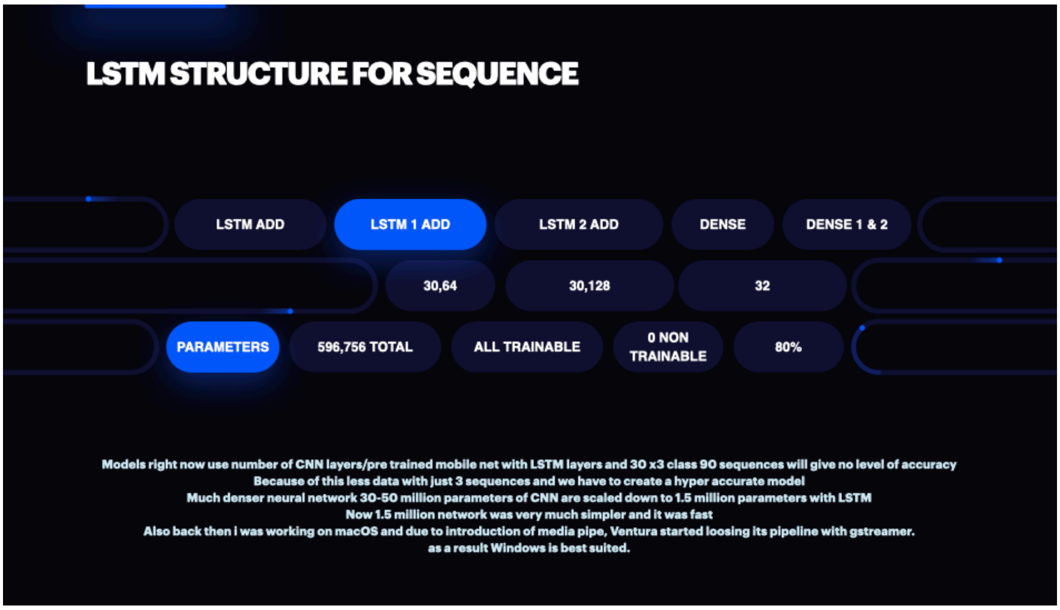


Fig. 3.4 A closer look on step 2

3. Compiling the model with a chosen loss function and optimizer algorithm using: model.compile(loss='name_of_loss_function', optimizer='name_of_optimizer_alg')

4. Note that the loss function is used to evaluate the accuracy of each prediction made by the model.

This code uses the multilabel_confusion_matrix and accuracy_score functions from the sklearn.metrics module to evaluate the performance of a machine learning model. The steps involved in this code are:

- **The model is run on the test data X_test and the predictions are stored in yhat.**
- **The true labels y_test are transformed into a list of the index of the highest value for each example using np.argmax(y_test, axis=1).tolist().**
- **The predicted labels are transformed into a list of the index of the highest value for each example using np.argmax(yhat, axis=1).tolist().**
- **The function multilabel_confusion_matrix is called with the true labels and predicted labels as inputs, which calculates a confusion matrix for the multi-label classification problem.**

- **The accuracy of the predictions is calculated using accuracy_score by passing the true labels and predicted labels as inputs.**

(Before training the model, it's important to scale the data for better performance.)

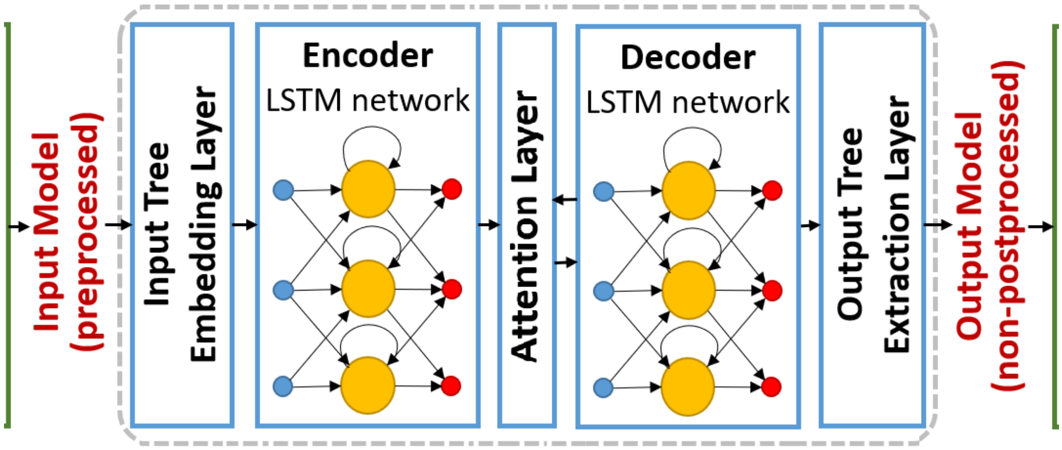
An Overview on LSTM with Neural Networks:

(This is domain analysis for the project and understanding neural networks and LSTM help.)

A typical LSTM (Long Short-Term Memory) network in a neural network can be represented as follows:

Input layer -> LSTM layer(s) -> Output layer

The input layer takes in sequences of data, and the LSTM layer processes the sequences using gates (input, forget, and output) to regulate the flow of information and maintain a state of memory. The output layer provides the final predictions based on the processed information. The number of LSTM layers can vary based on the complexity of the problem being solved.



Fir 3.5 LSTM GENERAL MODEL

✳ Model training:

This code trains a machine learning model using training data and the expected output for this data. The steps involved in this code are:

1. The fit function of the model is called with the training data and expected output as inputs. model.fit(training_data, expected_output)
2. The progress of the training is visible on the console as the script runs.
3. At the end of the training, the final accuracy of the model is reported.

(COLLECT KEYPOINTS VALUES FOR TRAINING AND TESTING)

A code should be written in model training at the start to collect KeyPoints and store them also these would be past for training and testing . The code should capture video from the default camera using OpenCV's VideoCapture class and sets up a mediapipe holistic model. The code then enters a loop for a list of actions, for a range of sequences, and for a fixed length of frames in each sequence. For each frame, the code reads the video feed, makes detections using a mediapipe function, draws landmarks on the image, and displays it on the screen. The code also saves the keypoints extracted from the detections as numpy arrays at a specified file path. The loop continues until the 'q' key is pressed, at which point the camera is released and the windows are destroyed.

(TRAIN LSTM NEURAL NETWORK)

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])
model.summary()

The model is being set up with the 'Adam' optimizer and the loss function used is 'categorical crossentropy'. The model will be trained using the X_train and y_train data for 2000 epochs with the use of a tensorboard callback. Finally, a summary of the model is displayed.

✳ Model Testing:

This code tests the performance of a machine learning model using a second set of data that has not been seen by the model before. The steps involved in this code are:

1. A second set of data is loaded for testing the model.
2. The accuracy of the model is verified by comparing its predictions with the true values.
3. If the model is performing well, it can be saved using the save function with a file name as input. model.save("name_of_file.Z1")
4. The saved model can then be used in the real world to evaluate new data, which is also known as model evaluation.

(LIVE TESTING FOR THE MODEL)

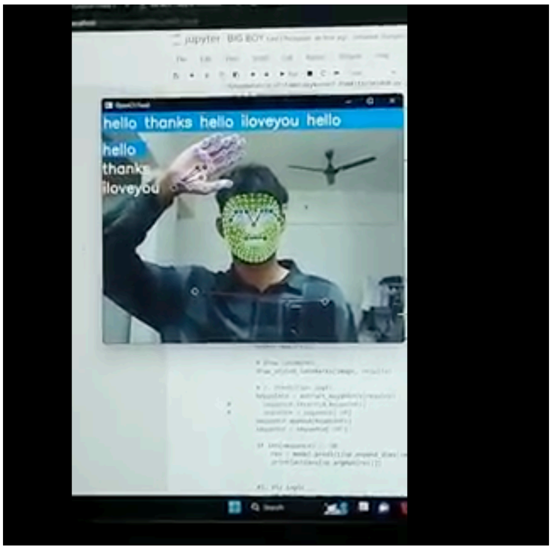


Fig. 3.6 TEST 1

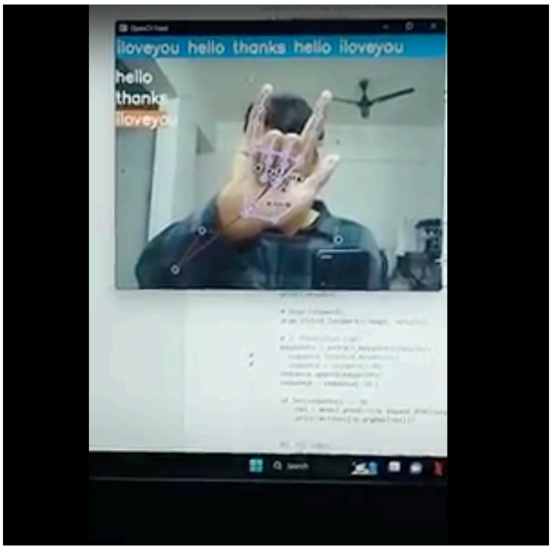


Fig. 3.7 TEST 2

Using SciPy library and the OpenCV library to visualize the probability of a machine learning model's predictions. The code has a the following steps:

1. It defines a list of RGB colors to be used for the visualization.
2. A prob_viz function is created that takes in the model's predictions (res), a list of actions, an input frame (image), and the list of colors.
3. The function creates a copy of the input frame and uses a for loop to add rectangles for each action based on the probabilities. The color of the rectangle corresponds to the color in the list of colors.
4. The function also adds text for each action using the putText function from OpenCV.

3.5 THE ALGORITHM FOR TRAINING A NEURAL NETWORK:

Optimizer(Adam): Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the

Learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network. Now, what is moment ? N-th moment of a random variable is defined as the expected value of that variable to the power of n. More formally:

Adam is an adaptive learning rate method, meaning it calculates individual learning rates for each parameter. It's named "adaptive moment estimation" because it uses estimates of the first and second moments of the gradient to adapt the learning rate for each weight in the neural network. The moment of a random variable refers to the expected value of that variable raised to the power of n.

Categorical crossentropy is a type of loss function used for single-label categorization, where each data point only belongs to one class. It is used to calculate the error rate between the predicted output and the actual target. The activation function used before the target block must be Softmax.

3.6 SEGMENTATION (ADD ON)

Image segmentation refers to the process of dividing a digital image into multiple segments or sets of pixels, also known as image objects. The purpose of image segmentation is to simplify or modify the representation of an image, making it easier to analyze. Image segmentation is a crucial step in fields such as autonomous driving, where sensory input devices like cameras, radar, and lasers generate digital maps. Deep learning techniques, such as various deep learning architectures, are widely used for image segmentation. Image segmentation involves converting an image into a collection of regions of pixels represented by a mask or labeled image. This allows processing of only the important segments of the image, rather than the entire image. Image segmentation can be performed through detecting abrupt discontinuities in pixel values, looking for similarities in the regions of an image, or using domain-specific knowledge to solve segmentation problems in specific application areas.

3.7 CONVOLUTION NEURAL NETWORK (CNN)

(This method exists but we decided to switch to LSTM Neural Network model due to less data ,as models right now use number of CNN layers/ pre trained mobile net with LSTM layers and 30 x3 class 90 sequences will give no level of accuracy , also much denser neural network 30-50 million parameters of CNN are scaled down to 1.5 million parameters with LSTM)

Convolution Formula:

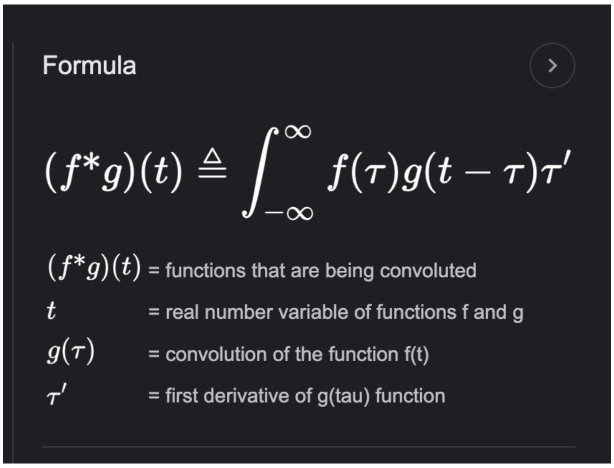


Fig. 3.8 Formula

The convolution formula is a mathematical expression that describes the way two functions interact with each other. It is defined as the integral of the product of one function with a flipped and shifted version of the other function. In computer vision and image processing, convolution is used to filter images and extract features.

The rectified linear unit (ReLU) layer is used to scale the parameters to non-negative values, which is achieved by setting negative values to 0. The purpose of this layer is to increase the non-linearity in the images, as images are naturally non-linear. This is achieved by breaking up the linearity even further and compensating for the linearity that might be imposed during the convolution operation.

The pooling layer reduces the height and width of the input, which helps to reduce computation and make feature detectors more invariant to their position in the input. The fully connected layer combines the features into a wider variety of attributes that make the convolutional network more capable of classifying images. This layer is made up of neurons that are linked to each other and activate when they identify patterns, which are then communicated to the output layer. The output layer gives the output class based on weight values, and the accuracy of the network is determined by the loss function, which is used to optimize the network. This optimization process involves adjusting the weights and feature detectors.

4. CONCLUSION (CREATED LSTM MODEL & EVALUATION)

As we created a logs folder in training plan of our methodology we can easily show / visualise our results in tensor board by opening up a terminal and running the file from logs < train < // folder,
using ***tensorboard - - logdir.***

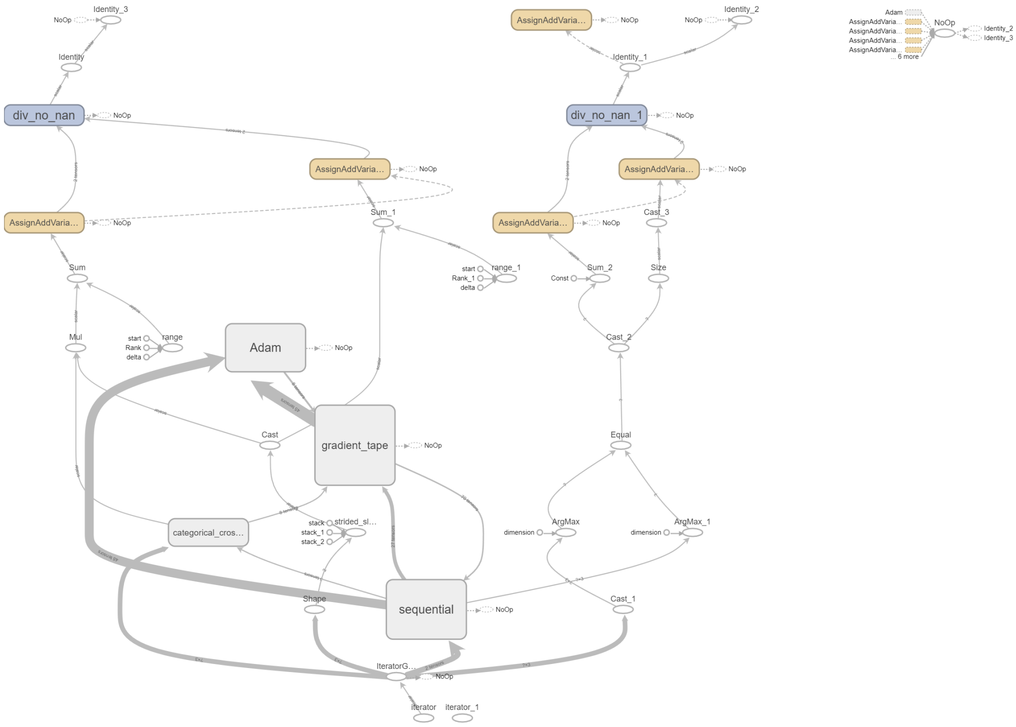


Fig. 3.9 VISUALISED BY TENSOR-BOARD

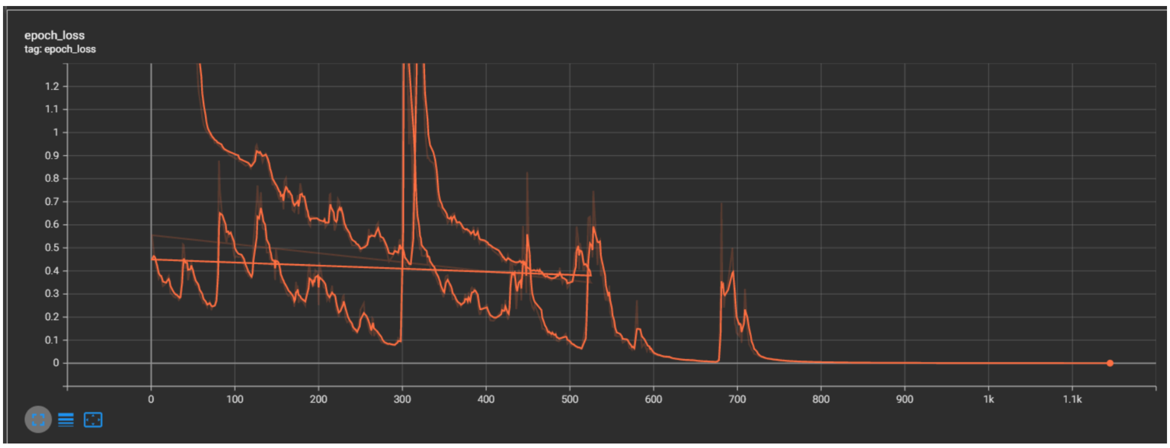


Fig. 3.10 EPOCH LOSS PLOT (NON SCALED)

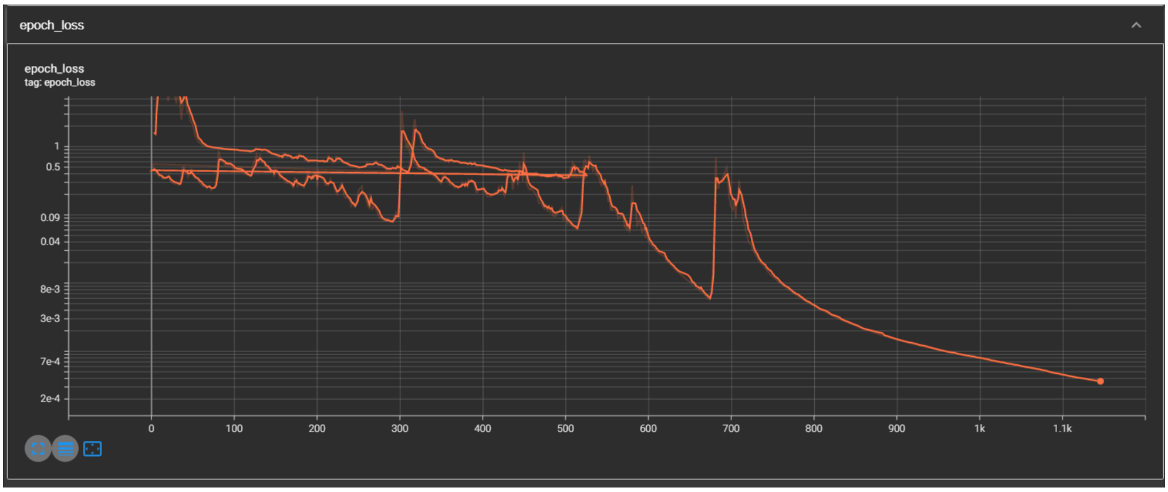


Fig. 3.11 EPOCH LOSS PLOT (SCALED)

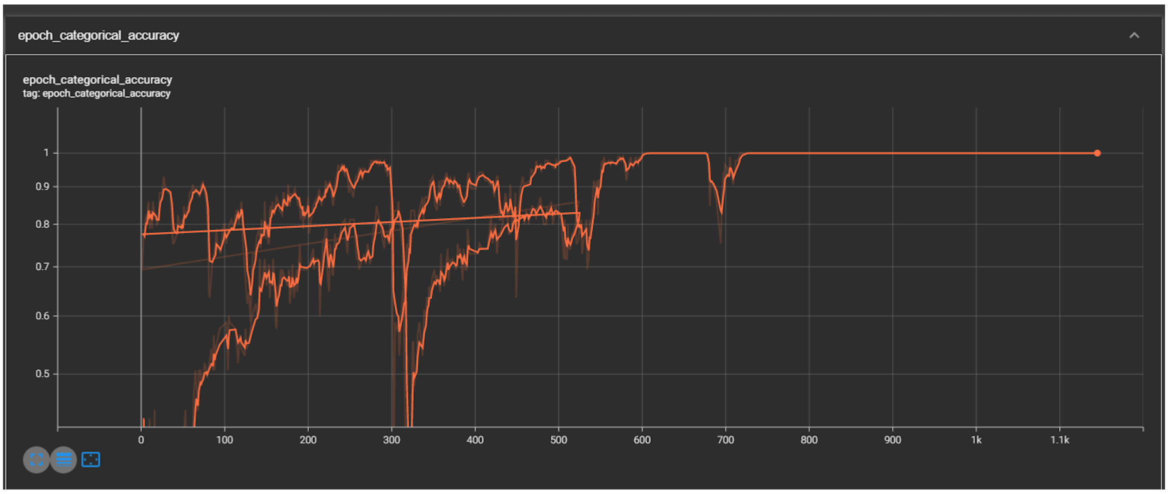


Fig. 3.12 EPOCH CATEGORICAL ACCURACY PLOT (NON SCALED) - 89.34% ACCURACY

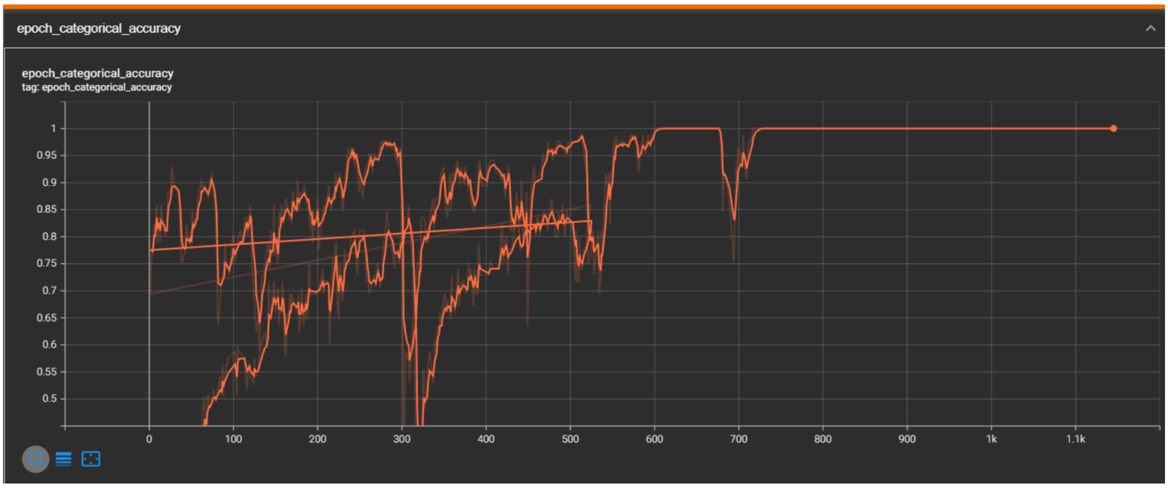


Fig. 3.13 EPOCH CATEGORICAL ACCURACY PLOT (SCALED) - 89.34% ACCURACY

REFERENCES

[1] Mokhtar M. Hasan, Pramoud K. Misra, (2011). "Brightness Factor Matching For Gesture
Recognition System Using Scaled Normalization", International Journal of Computer Science &
Information Technology (IJCSIT), Vol. 3(2).
[2] V. S. Kulkarni, S.D.Lokhande, (2010) "Appearance Based Recognition of American Sign Language Using Gesture Segmentation", International Journal on Computer Science and Engineering (IJCSE), Vol. 2(3), pp. 560-565.
3] Mokhar M. Hasan, Pramod K. Mishra, (2012) "Features Fitting using Multivariate Gaussian
Distribution for Hand Gesture Recognition", International Journal of Computer Science & Emerging
Technologies IJCSET, Vol. 3(2).
[4] Mokhar M. Hasan, Pramod K. Mishra, (2012). "Robust Gesture Recognition Using Gaussian
Distribution for Features Fitting', International Journal of Machine Learning and Computing, Vol. 2(3).
[5] Min B., Yoon, H., Soh, J., Yangc, Y., & Ejima, T. (1997). "Hand Gesture Recognition Using Hidden Markov Models". IEEE International Conference on computational cybernetics and simulation. Vol.
5, Doi: 10.1109/ICSMC.1997.637364
[6] E. Stergiopoulou, N. Papamarkos. (2009). "Hand gesture recognition using a neural network shape
fitting technique," Elsevier Engineering Applications of Artificial Intelligence, vol. 22(8), pp. 1141–
1158, doi: 10.1016/j.engappai.2009.03.008
[7] Xingyan Li. (2003). "Gesture Recognition Based on Fuzzy C-Means Clustering Algorithm",
Department of Computer Science. The University of Tennessee Knoxville.
[8] Luigi Lamberti, Francesco Camastra, (2011). "Real-Time Hand Gesture Recognition Using a Color
Glove", Springer Proceedings of the 16th international conference on Image analysis and processing:
Part I ICIAP.
[9] Mokhar M. Hasan, Pramod K. Mishra, (2012). "Robust Gesture Recognition Using Gaussian
Distribution for Features Fitting', International Journal of Machine Learning and Computing, Vol.
2(3).

[10] W. T. Freeman and Michal R., (1995) “Orientation Histograms for Hand Gesture Recognition”,
IEEE International Workshop on Automatic Face and Gesture Recognition.
[11] Simei G. Wysoski, Marcus V. Lamar, Susumu Kuroyanagi, Akira Iwata, (2002). “A Rotation
Invariant Approach On Static-Gesture Recognition Using Boundary Histograms And Neural 172
International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, July 2012
[12] Kouichi M., Hitomi T. (1999) “Gesture Recognition using Recurrent Neural Networks” ACM conference on Human factors in computing systems: Reaching through technology (CHI '91), pp. 237-242. doi: 10.1145/108844.108900
[13] Mahmoud E., Ayoub A., J¨org A., and Bernd M., (2008). “Hidden Markov Model-Based Isolated and Meaningful Hand Gesture Recognition”, World Academy of Science, Engineering and
Technology 41.
[14] Malima, A., Özgür, E., Çetin, M. (2006). “A Fast Algorithm for Vision-Based Hand Gesture
Recognition For Robot Control”, IEEE 14th conference on Signal Processing and Communications
Applications, pp. 1-4. doi: 10.1109/SIU.2006.1659822