

**SYSTEM FOR
HUMAN ACTIVITY RECOGNITION
WITH IMAGES USING DEEP LEARNING**

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

BY

**Kishore Kumar .A (Reg. No – 39120056)
Partha Pratim Halder (Reg. No - 39120079)**



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

MAY – 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Kishore Kumar.A (39120056)** and **Partha Pratim Halder (39120079)** who carried out the Project Phase-1 entitled "**System for human activity recognition with images using Deep Learning**" under my supervision from December 2022 to April 2023.

[Signature]
26/4/2023

Internal Guide

C.GEETHA, M.Tech, (PH.D), Assistant professor, IT

[Signature]
9/6/23

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.

Submitted for Viva voce Examination held on 11/05/2023

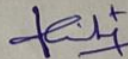
[Signature]
10/5/23
Internal Examiner

[Signature]
11/5/23
External Examiner

DECLARATION

I, **Kishore Kumar.A (Reg. No- 39120056)**, hereby declare that the Project Phase-1 Report entitled **“System for Human Activity Recognition with Images using Deep Learning”** done by me under the guidance of **C.Geetha, M.Tech, (PH.D), Assistant professor, IT** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Information Technology**.

DATE: 15/04/2023



PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.SASIKALA, M.E., Ph.D., Dean, School of Computing, Dr. R. SUBHASHINI, M.E., Ph.D. Head of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **DR. C.GEETHA, M.Tech, (PH.D), Assistant professor, IT** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my gratitude and thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

Recognizing human behavior in the realm of computer vision, HAR is an important but challenging research topic. In many applications, including surveillance, in-home monitoring, human-computer interaction, and health care, human activity recognition is essential. It is an important yet challenging research topic in the computer vision community.

To determine the specific subject activity in the image, we suggest context characteristics and a deep learning model in this work. To enhance the performance of recognition, we use images from google image search and used those images for our dataset. We create a deep neural network framework to provide a high-level depiction of human behavior identification based on a picture collection. Recognizing human activity necessitates forecasting a person's activity using image-based information. The photos are divided into recognized activities.

We propose context features along with a deep learning model to recognize the individual subject activity in the image. We take the dataset from multiple sources to improve the recognition performance. We design a deep neural network structure to obtain the high-level representation of human activity recognition based on image dataset. Human activity recognition requires predicting the action of a person by image-based data. It classifies the images into activity recognized.

The goal of our project is to detect the movement and actions of a person using image detection taken from a website or the local computer itself. In our project, the image is treated as an object. We are mainly focusing on four poses of human i.e sitting, sleeping, standing and bending via the image. Using deep learning based techniques for Human Activity results in best accuracy. We can implement CNN Algorithm to classify the images. More than two Architectures were compared to select the best architecture. Finally the model can be deployed in Django framework.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	v
	LIST OF FIGURE	ix
1.	INTRODUCTION	
	1.1 Artificial Intelligence	1
	1.2 Machine Learning	2
	1.3 Deep Learning	2
	1.4 Digital Image Processing	3
	1.5 Activity Detection and Recognition	4
	1.5.1 Detection	
	1.5.2 Recognition	
	1.6 Objectives	5
	1.7 Scope	5
2.	SYSTEM ANALYSIS	
	2.1 Existing System	6
	2.2 Proposed System	6
	2.3 Literature Survey	7
	2.4 Inferences From Literature Survey	12
	2.5 Open Problems In Existing System	13
	2.6 Computer Vision and Digital Image Processing	15
	2.7 Open CV in Image Processing	15
	2.8 Pattern Recognition and Classifiers	17
3.	REQUIREMENT ANALYSIS	

3.1	Feasibility Study	19
3.2	Software Requirements	20
3.2.1	Windows 10	
3.2.2	Anaconda	
3.2.3	Jupyter Notebook	
3.2.4	Django	
3.3	Hardware Requirements	23
3.3.1	Hard Disk	
3.3.2	RAM	
3.3.3	Processor	
3.3.4	Mouse	
3.3.5	Keyboard	
4.	DESCRIPTION	
4.1	Description Of Proposed System	28
4.2	Construction of a Detecting Model	28
4.3	Design Architecture	29
4.3.1	General	
4.3.2	Data Flow diagram	
4.3.3	System Architecture Diagram	
4.3.4	Use Case diagram	
4.3.5	Sequence diagram	
4.3.7	ER Diagram	
4.3.8	Data set used	
4.4	Module Description	36
4.5	Convolutional Neural Network	39
4.6	Libraries Required	47
4.6.1	Tensorflow	
4.6.2	Keras	
4.6.3	matplotlib	
4.6.4	os	

5.	RESULT, CONCLUSION AND FUTURE ENHANCEMENTS	
5.1	Result	50
5.2	Conclusion	52
5.3	Future Enhancements	52
	REFERENCES	53
	APPENDICES	
	A. SOURCE CODE	55
	B. SCREENSHOTS	67
	C. RESEARCH PAPER	71

LIST OF FIGURE

Figure No.	Figure Name	Page No.
1.1	Steps Deep Learning	3
3.1	Anaconda navigator	21
3.2	Django	23
4.1	Steps of data flow	28
4.2	Process of data flow	31
4.3	Architecture Diagram	31
4.4	Use Case Diagram	32
4.5	Sequence Diagram	33
4.6	ER diagram	34
4.7	Dataset	35
4.8	Lenet Architecture	44
4.9	Resnet Architecture	46
5.1	Frontend Page	50
5.2-5.4	output	51
6.1	Importing images	68
6.2	Model accuracy plots	69
6.3	Model accuracy	70

CHAPTER - 1

INTRODUCTION

Human activity recognition (HAR) is a well-known research topic that involves the correct identification of different activities, sampled in a number of ways. On this regard, automatic and deep methods are gaining momentum in the field of HAR. With the adoption of data-driven approaches for image classification, the process of selecting meaningful features from the data is deferred to the learning model. In particular, CNNs have the ability to detect the dependencies among image, and can effectively model scale invariant features. In this paper, we apply convolutional neural networks for the HAR problem. The dataset we collected is composed of 5 activities.

To classify the human activity to design a deep learning technique. It proposed a system for predicting human activity based on an image dataset. Samples of more images are collected that are compared with different classes. The primary attributes of the image are relied upon the shape and texture-oriented features. Initially, we are preprocessing our dataset and implementing more than two CNN architectures. Each architecture gives a different kind of accuracy, so we compare each architecture. After we save the model in .h5 file format. Because we don't need to train the architecture again and again. Once we trained and get the accuracy when saving the model in hierarchical file format. Now you can check the accuracy whenever you want. That trained model can be deployed in Django Framework.

1.1 Artificial Intelligence

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to

the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of “intelligent agents” any system that perceives its environment and takes actions that maximize its chance of achieving its goals.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, and speech recognition and machine vision.

1.2 Machine Learning

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed.

Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms.

It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm.

1.3 Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as

a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

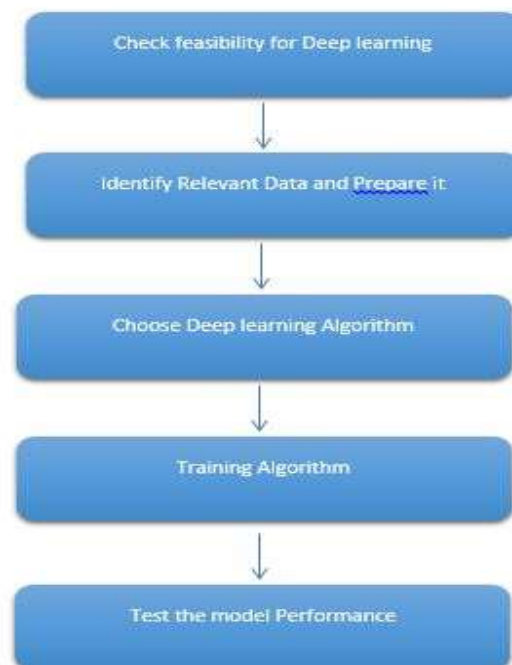


Fig 1.1 Deep learning steps

1.4 Digital Image Processing

Image processing is reckon as one of the most rapidly involving fields of the software industry with growing applications in all areas of work. It holds the possibility of developing the ultimate machine in the future, which would be able to perform the visual function of living beings. As such, it forms the basis of all kinds of visual

automation.

1.5 Activity Detection and Recognition

A Human activity Recognition System recognizes the Shapes and or orientation depending on implementation to task the system into performing some job. Movement is a form of nonverbal information. A person can make numerous movements at a time. As humans through vision perceive human gestures and for computers we need an image and a camera, it is a subject of great interest for computer vision researchers such as performing an action based on activity of the person.

1.5.1 Detection

Activity detection is related to the position of a human at a given time in a stiff image or sequence of images i.e. moving images. In case of a moving sequence, it can be followed by tracking of the movement in the scene but this is more relevant to the applications such as sign language. The underlying concept of activity detection is that human eyes can detect objects, which machines cannot, with that much accuracy as that of humans. From a machine point of view it is just like a man fumble with his senses to find an object.

1.5.2 Recognition

Human activity detection and recognition has been a significant subject in the field of computer vision and image processing in the past 30 years. There have been considerable achievements and numerous approaches developed in this field.

1.6 Objectives

The objectives of the project are:

1) Study and apply the needed tools namely:

- Image downloaded from computer or locally saved.
- anaconda Deployed Server and python 3.6.8 Community
- Algorithms for computer vision and deep learning.

2) Develop a front-end website to upload images to process.

3) Test the computer application and website running

4) Document the result of the project.

1.7 Scope

The scope of our project is to develop a real time activity recognition system which ultimately controls the image with a jpg extension or the camera samples of real-time webcam method. During the project, four gestures were chosen to represent four navigational commands that are sitting, standing, bending and sleeping.

A simple computer vision application was written for the detection and recognition of the five gestures and their translation into the corresponding commands for the actions and tracking.

CHAPTER - 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Human activity recognition (HAR) plays a vital role in many applications, such as surveillance, in-home monitoring, and health care. Portable radar sensor has been increasingly used in HAR systems in combination with deep learning (DL). However, it is both difficult and time-consuming to obtain a large-scale radar dataset with reliable labels. Insufficient labeled data often limit the generalization of DL models. As a result, the performance of DL models will drop when being applied to a new scenario. In this sense, only labeling a small portion of data in the large-scale radar dataset is more feasible. In this article, we have proposed a new JDS-TL method to recognize human activities in a semi supervised manner.

The proposed JDS-TL is composed of the unsupervised adversarial domain transfer module and the supervised semantic transfer module. The former can mitigate the harmful effects of domain shift and learn domain alignment feature representation. The latter can transfer the intra class correlation from the source domain to the target domain by utilizing a small amount of labeled target data. A public radar-based HAR dataset including six daily human activities was employed to evaluate the proposed method.

2.2 PROPOSED SYSTEM

To classify the human activity to design a deep learning technique. It proposed a system for predicting human activity based on an image dataset. Samples of more images are collected that are compared with different classes. The primary attributes of the image are relied upon the

shape and texture-oriented features. Initially, we are preprocessing our dataset and implementing more than two CNN architectures. Each architecture gives a different kind of accuracy, so we compare each architecture. After we save the model in .h5 file format. Because we don't need to train the architecture again and again. Once we trained and get the accuracy when saving the model in hierarchical file format. Now you can check the accuracy whenever you want. That trained model can be deployed in Django Framework.

2.3 Literature Survey

[1] Human Motion Detection with Deep Convolutional and Recurrent Models for Wearables

Author: Nils Y. Hammerla^{1,2}, Shane Halloran²

Year: 2016

In human activity identification (HAR) in pervasive computing, deep learning is beginning to displace well-established analytical methods that depend on hand produced feature extraction and categorization techniques. For problems varying from the division and recognizable proof of proactive tasks like running or rising steps to difficulties involving the detection of moving motions, it is difficult to comprehend the usefulness of bespoke deep structures.

They thoroughly research deep, neural, and recurrent methods using three sample datasets that comprise movement data obtained from external devices. We outline the training procedures for recurrent methods in this situation, present a fresh regularisation technique, and demonstrate how they beat the cutting edge on a sizable standard

dataset. We use huge number of acknowledgment tries different things with haphazardly inspected model setups to examine each model's suitability for human activity recognition. We then use the fANOVA framework to examine the effects of the hyperparameters. Finally, we provide recommendations for practitioners who want to use deep learning in their problem-solving.

[2] A Deep Learning Approach for Human Activity Recognition

Author: Susana Benavidez, Derek McCreight

Year: 2020

Every year, the hardware and sensors in smartphones and wearable devices become more powerful and precise; the valuable data collected by these sensors can be used for more precise Human Activity Recognition. In this article, we investigate the classification of 18 distinct tasks using convolutional neural networks and LSTMs, ranging from eating chips to dribbling a ball, using labelled, time-series gyroscope and accelerometer data from a smartwatch and smartphone. All of our work is available in our public repo, and the processed data is available here.

[3] Human activity recognition with smartphone sensors using deep learning neural networks

Author: Charissa Ann Ronao, Sung-Bae Cho*

Year: 2019

Translation constant and hierarchy behaviour is a natural part of human activity. Time-series sensing data is used to deduce actions in the area of human activity identification (HAR), which, as its use becomes more widespread, has been the subject of increased research and

development in recent years. In this article, we show how to use the sensors on your smartphone to its full potential for HAR by capitalising on the unique properties of actions and 1D time-series data. While doing so, a technique is given for autonomously and data-adaptively pulling trustworthy characteristics from unprocessed data. Despite a diminishing difference in feature complexity with each additional layer, studies show that convnets generate increasingly important and complicated features as the number of layers increases. It is possible to use a larger time range of temporal local correlation (1–9–14), and it has been demonstrated that a small pooling size (1–2–13) is advantageous. Additionally, Convnets classified movement activities almost perfectly, particularly those that were comparable and were previously thought to be very challenging to categorise. In terms of human action, convnets work better than other cutting-edge data mining methods. Accomplished a total score of 94.79% on the test set utilizing crude sensor information and 95.75% with additional data from the transient quick Fourier change of the HAR informational index, the two of which were perceived for the benchmark dataset gathered from 30 energetic members.

[4] Machine Learning Algorithms for Human Activity Recognition

Author: K.R.Baskaran, M.N.Saroja

Year: 2019

By keeping note of their daily actions and helping those with serious health problems, it is crucial to watch the Activity of Daily Living (ADL) of senior people who live alone. In this study, various machine learning methods for recognising human behaviour are investigated. Additionally, a lot of study is being done to find out more about the tools that are

currently being used for movement detection. Typically, signals produced by sensors are used to identify activities. The data are then pre-processed, divided, characteristics are taken, and activity is identified. The Human Activity Detection System's primary objective is to examine the constraints of independent aged people and recommend solutions for them.

[5] Video-based human action recognition using deep learning

Author: Huy-Hieu Pham · Louahdi Khoudour

Year: 2019

An essential area of computer vision application is human movement detection. The primary goal is to provide an exact description of human actions and encounters based on a series of data never before seen by sensors. Having the ability to recognise, comprehend, and anticipate complicated human actions has far-reaching implications, including the development of intelligent monitoring systems, human-computer interactions, healthcare, security, and combat uses. The computer vision industry has recently paid deep learning a lot of attention. This paper surveys the current state of the art for action recognition through the lens of video analysis and deep learning techniques. In this article, we introduce the most important deep learning models for human action recognition and analyse their strengths and flaws to demonstrate the current status of deep learning algorithms used to address practical challenges in human action recognition. By conducting a quantified analysis of the identification accuracies described in the literature, we are able to identify state-of-the-art deep structures in the field of action recognition and show current trends and unresolved issues for future

works in this area.

[5] Human Activity Recognition Based On The Blob Features

Author: Jie Yang, Jian Cheng, Hanqing Lu

Year: 2019

The authors of this article describe a new method for identifying human activity in film. Since human actions can be considered to be a time entity with a number of frames, we analyse them in succession. We first build a statistical background model and then use background removal to isolate visible items from the video feed. After that, we create a new feature picture based on a set of guidelines using closer view objects from the current edge and a few casings before it. The non-zero pixels in the feature picture are then combined into clumps using the linked component technique. Then, for every glob, there is an action that can be identified by the way the blob looks. By identifying amorphous characteristics, we can recognise actions. Gaussian Mixture is used in this model to account for each category of human action, and distance measure is used to determine closeness.

[5] Human Activity Recognition via 3-D Joint Angle Features and Hidden Markov

Models

Author: Md. Zia Uddin, Nguyen Duc Thang, and Tae-Seong Kim

Year: 2018

In this article, a new method for Human Activity Identification (HAR) is presented, which is based on the 3-D joint orientations of the human body. Our approach does not depend on instruments affixed to the

person; All things being equal, a 3D body model is co-enrolled to the sound system data from each gathering of movement video pictures taken by the sound system camera to gauge the body joint directions. To start with, we decipher the expected joint point highlights from the time-successive action video outlines into codewords, which forms the basis of the concealed markov model for that activity (HMM). Each activity HMM is learned and then used to identify activities based on these patterns.

As we investigated further, we found that our combined angle-based Human activity Detection yielded significantly improved results in the identification rate, particularly for those activities that are not detectable with traditional methods. Indeed, when we compared the efficiency of our joint angle-based Human activity Recognition to that of the standard binary silhouette-based Human activity Recognition, we found the same thing to be true.

2.4 INFERENCES FROM LITERATURE SURVEY

It is possible to see from the aforementioned material that significant study has been done on this subject and that many instances have been put forth. Of course, each of the aforementioned methods has advantages and disadvantages. Even though some of the work utilizes the contemporary mix period and provides greater precision, it is still some distance from what is desired. Higher precision necessitates reduced numerical expenses, high-quality working speed, and, especially, simplicity of use.

Human action recognition has been extensively researched for many years. The majority of action recognition methods necessitate manual annotation of

the action. In recent years, there has been a great deal of research into action recognition. There is a distinction between action recognition and action detection. The most effective method is deep learning.

2.5 OPEN PROBLEMS IN EXISTING SYSTEM

Human endeavor numerous uses, such as tracking, in-home monitoring, and medical treatment, depend on recognition. Portable radar sensors, in conjunction with deep learning, are increasingly being used in Human activity Recognition systems (DL).

However, obtaining a large-scale radar dataset with reliable labels is both difficult and time-consuming. Inadequate labelled data frequently limits the generalization of Deep Learning models. As a result, when applied to a new scenario, the performance of DL models will suffer. In this regard, labelling only a small portion of the large-scale radar dataset is more feasible.

Drawbacks

- They are not using CNN Algorithm for getting better prediction.
- They are using un-labeled data for classification, so prediction can vary from the real activity.
- Deployment cannot be done here.

2.6 Computer Vision and Digital Image Processing

The sense of sight is arguably the most important of man's five senses. It provides a huge amount of information about the world that is rich in detail and delivered at the speed of light. However, human vision is not without its limitations, both physical and psychological. Through digital imaging technology and computers, man has transcending many visual

limitations. He can see into far galaxies, the microscopic world, the sub-atomic world, and even “observe” infrared, x-ray, ultraviolet and other spectra for medical diagnosis, meteorology, surveillance, and military uses, all with great success.

While computers have been central to this success, for the most part man is the sole interpreter of all the digital data. For a long time, the central question has been whether computers can be design to analyses and acquire information from images autonomously in the same natural way humans can.

Russell and Norvig identified three broad approaches used in computer vision to distil useful information from the raw data provided by images. The first is the feature extraction approach, which focuses on simple computations applied directly to digital images to measure some useable characteristic, such as size. This relies on generally known image processing algorithms for noise reduction, filtering, object detection, edge detection, texture analysis, computation of optical flow, and segmentation, which techniques are commonly used to pre-process images for subsequent image analysis. This is also considered as an “uninformed” approach.

The second is the recognition approach, where the focus is on distinguishing and labelling objects based on knowledge of characteristics that sets of similar objects have in common, such as shape or appearance or patterns of elements, sufficient to form classes. Here computer vision uses the techniques of artificial intelligence in knowledge representation to enable a “classifier”

to match classes to objects based on the pattern of their features or structural description. A classifier has to “learn” the patterns by being fed a training set of objects, their classes, achieving the goal of minimizing mistakes, and maximizing successes through a systematic process of improvement. There are many techniques in artificial intelligence that can be used for object or pattern recognition, including statistical pattern recognition, neural nets, genetic algorithms and fuzzy systems.

The third is the reconstruction approach, where the focus is on building a geometric model of the world suggested by the image or images and which is used as a basis for action. This corresponds to the stage of image understanding, which represents the highest and most complex level of computer vision processing. Here the emphasis is on enabling the computer vision system to construct internal models based on the data supplied by the images and to discard or update these internal models as they are verified against the real world or some other criteria. If the internal model is consistent with the real world, then image understanding takes place. Thus, image understanding requires the construction, manipulation and control of models and now relies heavily upon the science and technology of artificial intelligence.

2.5 Open CV in Image Processing

OpenCv is a widely used tool in computer vision. It is a computer vision library for real-time applications, written in C and C++, which works with the Windows, Linux and Mac platforms. It is freely available as open source software from <http://sourceforge.net/projects/opencvlibrary/>.

OpenCv was started by Gary Brodsky at Intel in 1999 to encourage computer vision research and commercial applications and, side-by-side with these, promote the use of ever-faster processors from Intel. OpenCV contains optimised code for a basic computer vision infrastructure so developers do not have to re-invent the proverbial wheel. Brodsky and Koehler provide the basic tutorial documentation. According to its website, OpenCV has been downloaded more than two million times and has a user group of more than 40,000 members. This attests to its popularity.

- smoothing (blurring) functions to reduce noise,
- Dilation and erosion functions for isolation of individual elements,
- Flood fill functions to isolate certain portions of the image for further processing,
- Filter functions, including Sobel, Laplace and Canny for edge detection,
- Hough transform functions for finding lines and circles,
- Affine transform functions to stretch, shrink, warp and rotate images,
- Integral image function for summing sub regions (computing Haar wavelets),
- Histogram equalization function for uniform distribution of intensity values,
- Contour functions to connect edges into curves,
- Bounding boxes, circles and ellipses,
- Moments functions to compute Hu's moment invariants,

- Optical flow functions (Lucas-Kanade method),
- Motion tracking functions (Kalman filters), and
- Face detection/ Haar classifier.

OpenCV has a module containing basic image processing and computer vision algorithms. These include:

OpenCV also has an ML (machine learning) module containing well-known statistical classifiers and clustering tools. These include:

- Normal/ naïve Bayes classifier,
- Decision trees classifier,
- Boosting group of classifiers,
- Neural networks algorithm,.

2.6 Pattern Recognition and Classifiers

In computer, vision a physical object maps to a particular segmented region in the image from which object descriptors or features may be derive. A feature is any characteristic of an image, or any region within it, that can be measure. Objects with common features may be group into classes, where the combination of features may be considered a pattern. Object recognition may be understood to be the assignment of classes to objects based on their respective patterns. The program that does this assignment is called a classifier.

The most important step is the design of the formal descriptors because

choices have to be made on which characteristics, quantitative or qualitative, would best suit the target object and in turn determines the success of the classifier.

In statistical pattern recognition, quantitative descriptions called features are used. The set of features constitutes the pattern vector or feature vector, and the set of all possible patterns for the object form the pattern space X (also known as feature space).

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should

Provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations.

3.1.1 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources.

3.1.2 Economic Feasibility

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of

updating, etc. in order to estimate whether the new system will perform adequately or not.

3.1.3 Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

3.2 SOFTWARE REQUIREMENT

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it.

The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

- ❖ Windows 10 OS 64-bit (or above)
- ❖ Anaconda
- ❖ Jupyter Notebook
- ❖ Django

3.2.1 Windows 10 or above

Windows 10 is a Microsoft operating system for personal computers, tablets, embedded devices and internet of things devices.

Microsoft released Windows 10 in July 2015 as a follow-up to Windows 8. The company has said it will update Windows 10 in perpetuity rather than release a new, full-fledged operating system as a successor.

3.2.2 Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012.[8] As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

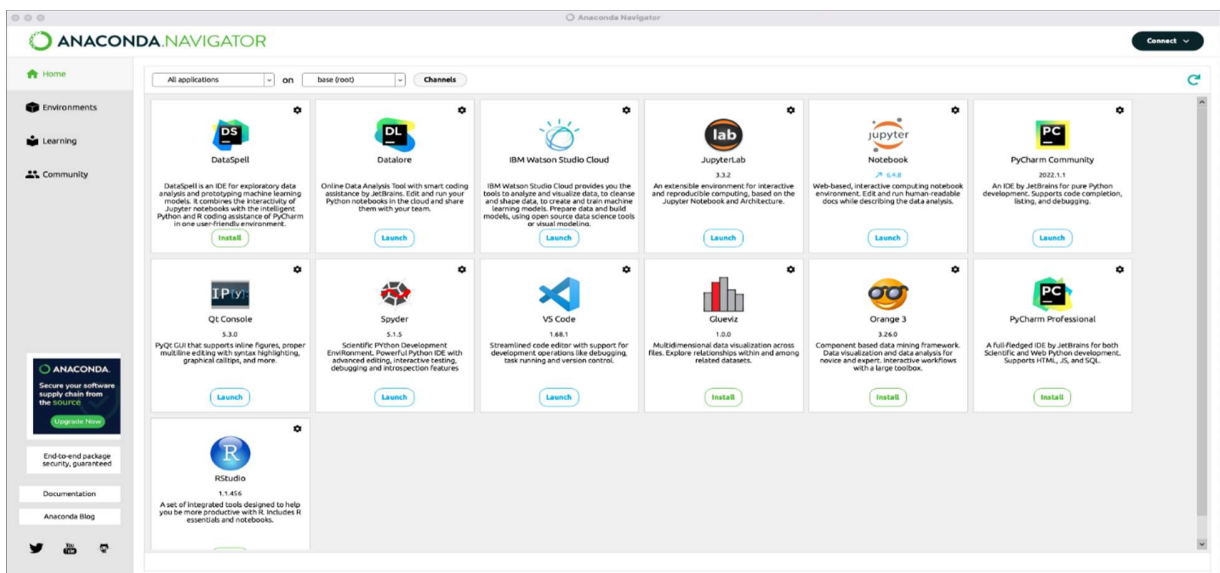


Fig.3.1 Anaconda navigator

3.2.3 Jupyter Notebook

Jupyter Notebook is a software used to develop open-source software, open standards, and services for interactive computing across multiple programming languages. It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are



Julia, Python and R. Its name and logo are an homage to Galileo's discovery of the moons of Jupiter, as documented in notebooks attributed to Galileo. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab. Jupyter is financially sponsored by NumFOCUS.

3.2.4 Django

Django is a free and open-source, Python-based web framework that follows the model–template–views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established in the US as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and

configured via admin models.

Some well-known sites that use Django include Instagram, Mozilla, Disqus, Bitbucket, Nextdoor and Clubhouse.

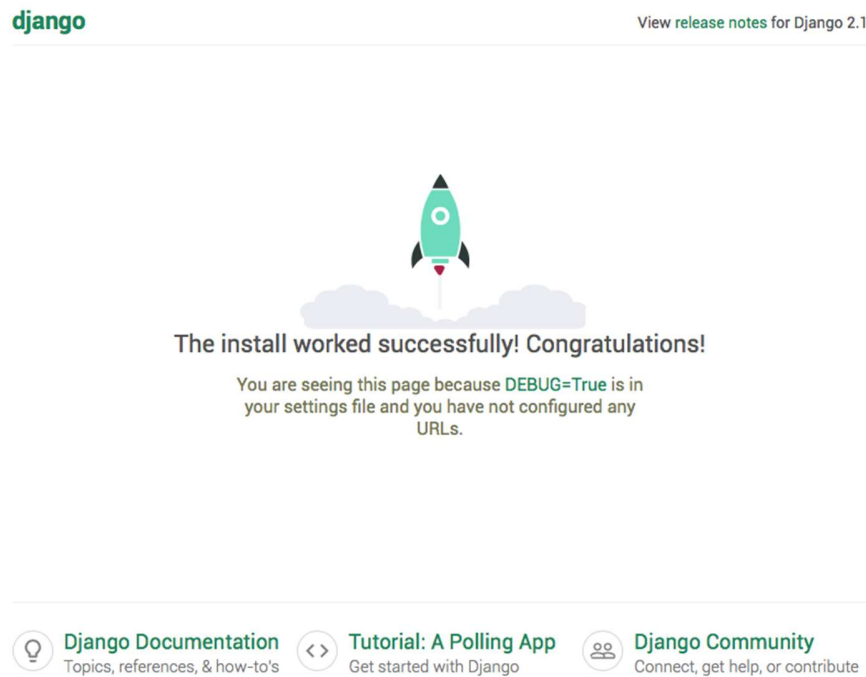


Fig.3.2 Django

3.3 HARDWARE REQUIREMENT

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

- Hard Disk
- RAM – 4 GB
- Processor – Dual Core or Above

- Mouse
- Keyboard
- Monitor
- Speaker

3.3.1 Hard Disk

A hard disk or fixed disk, is an electro-mechanical data storage device that uses magnetic storage to store and retrieve digital information using one or more rigid rapidly rotating disks (platters) coated with magnetic material.

During the mid-1990s the typical hard disk drive for a PC had a capacity of about 1 gigabyte. As of January 2019, desktop hard disk drives typically had a capacity of 1 to 6 terabytes, with the largest-capacity drives reaching 15 terabytes

3.3.2 RAM

Random access memory (RAM) is a type of data storage used in computers that is generally located on the motherboard. This type of memory is volatile and all information that was stored in RAM is lost when the computer is turned off.

The RAM chip may be individually mounted on the motherboard or in sets of several chips on a small board connected to the motherboard. Older memory types were in the form of chips called dual in-line package (DIP). Although DIP chips are still used today, the majority of memory is in the form of a module, a narrow printed circuit board attached to a connector on the motherboard. The three main memory circuit boards types containing chips are: RIMMs (Rambus in-line memory modules),

DIMMs (dual in-line memory modules) and SIMMs (single in-line memory modules). Most motherboards today use DIMMs. Sometimes, people get confused about the difference between memory and storage, in part because both can be measured in megabytes (MB), gigabytes (GB) and terabytes (TB).

By contrast, storage is slower, but it can retain data when the device is powered down. So, for example, if a document has been saved to a hard drive prior to a power outage or system crash, the user will still be able to retrieve it when the system is back up and running.

3.3.3 Processor

A processor is an integrated electronic circuit that performs the calculations that run a computer. A processor performs arithmetical, logical, input/output (I/O) and other basic instructions that are passed from an operating system (OS). Most other processes are dependent on the operations of a processor.

Dual core is a CPU that has two distinct processors that work simultaneously in the same integrated circuit. This type of processor can function as efficiently as a single processor but can perform operations up to twice as quickly.

3.3.4 Mouse

A mouse is a small handheld input device that controls a computer screen's cursor or pointer in conjunction with the way it is moved on a flat surface. The mouse term name originates from its likeness to a small, corded and elliptical shaped device that looks like a mouse tail.

Some mouse devices have integrated features, such as extra buttons that may be programmed and assigned with different commands. Because the mouse reduces the use of a keyboard, its invention and continuous innovation is considered one of the most important breakthroughs in computer ergonomics.

3.3.5 Keyboard

A computer keyboard is an input device used to enter characters and functions into the computer system by pressing buttons, or keys. It is the primary device used to enter text. A keyboard typically contains keys for individual letters, numbers and special characters, as well as keys for specific functions. A keyboard is connected to a computer system using a cable or a wireless connection.

A computer keyboard is an input device used to enter characters and functions into the computer system by pressing buttons, or keys. It is the primary device used to enter text. A keyboard typically contains keys for individual letters, numbers and special characters, as well as keys for specific functions.

3.3.6 Monitor

A computer monitor is an electronic device that shows pictures for computers. Monitors often look similar to televisions. The main difference between a monitor and a television is that a monitor does not have a television tuner to change channels. Monitors often have higher display resolution than televisions.

The advent of display technology has paved the way for the continuous evolution of the monitor, whether it's for computers, television, mobile

devices or any device that has a display. The current contenders for top-tier technology being used for display devices includes Super LCD 3 (SLCD3) and Super AMOLED. It should be noted that LED displays are actually just a kind of LCD display that use LED lights as backlight illumination.

CHAPTER 4

DESCRIPTION

4.1 DESCRIPTION OF PROPOSED SYSTEM

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensorflow library in Python using the Keras method.

4.2 Construction of a Detecting Model

Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly.

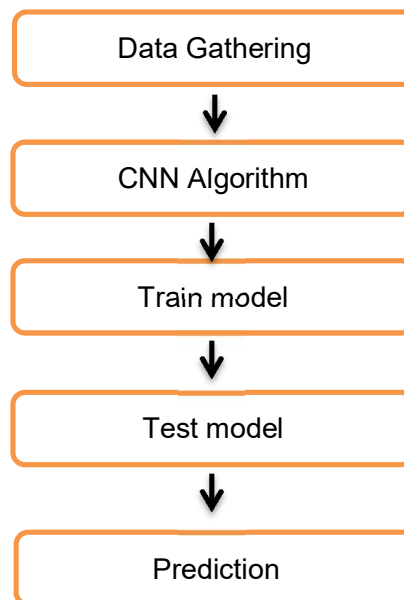


Fig 4.1 Steps of dataflow

4.3 DESIGN ARCHITECTURE

4.3.1 General

Design is meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system.

4.3.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design. Symbols and Notations Used in DFDs Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

External entity: an outside system that sends or receives data,

communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data store: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

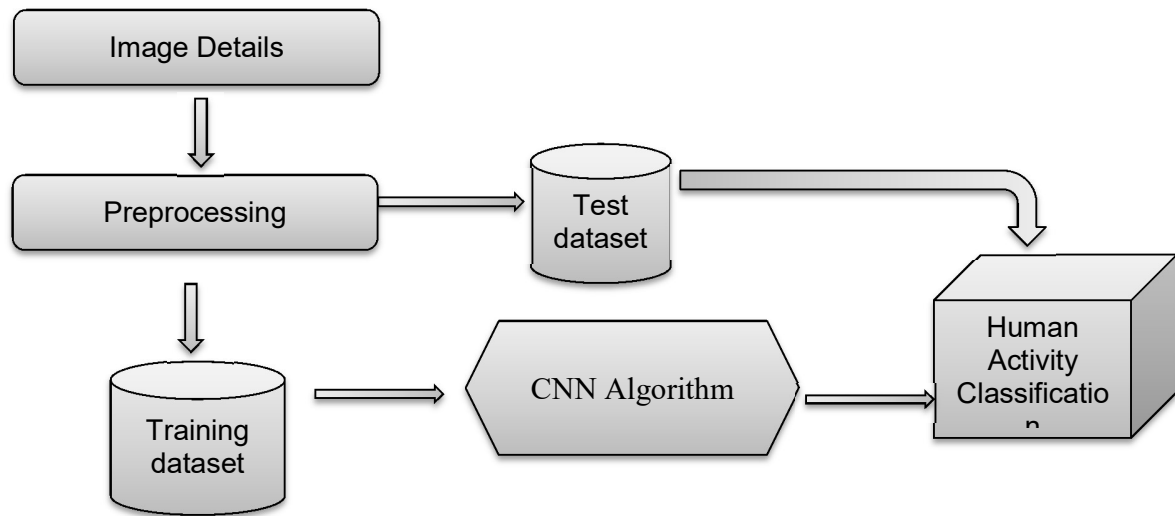


Fig 4.2 Process of dataflow

4.3.3 System Architecture Diagram:

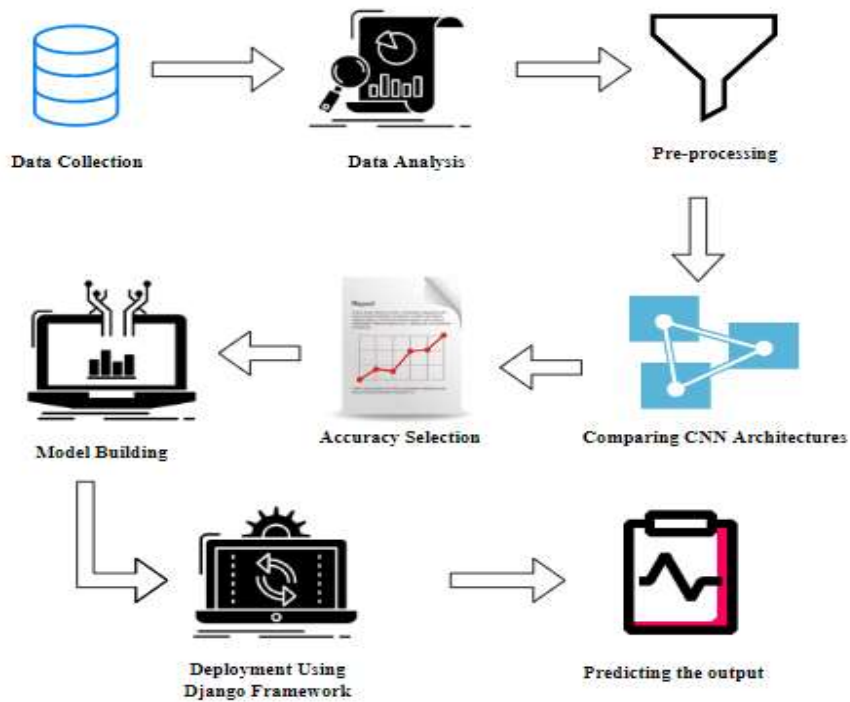


Fig 4.3 architecture diagram

The systems architect establishes the basic structure of the system, we propose a Hash code Solomon algorithm and a we can put a small part of data in local machine and fog server in order to protect the privacy. Moreover, based on computational intelligence, this algorithm can compute the distribution proportion stored in cloud, fog, and local machine, respectively. Through the theoretical safety analysis and experimental

Evaluation, the feasibility of our scheme has been validated, which is really a powerful supplement to existing cloud storage scheme.

4.3.4 Use case Diagram

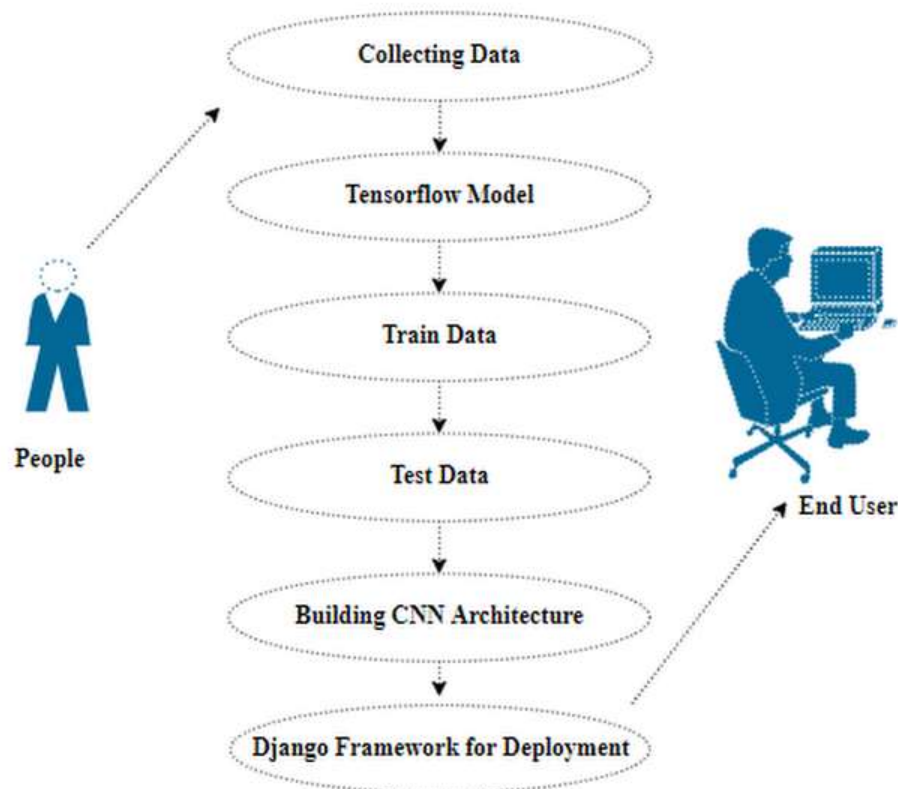


Fig 4.4 Use Case Diagram

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

4.3.5 SEQUENCE DIAGRAM

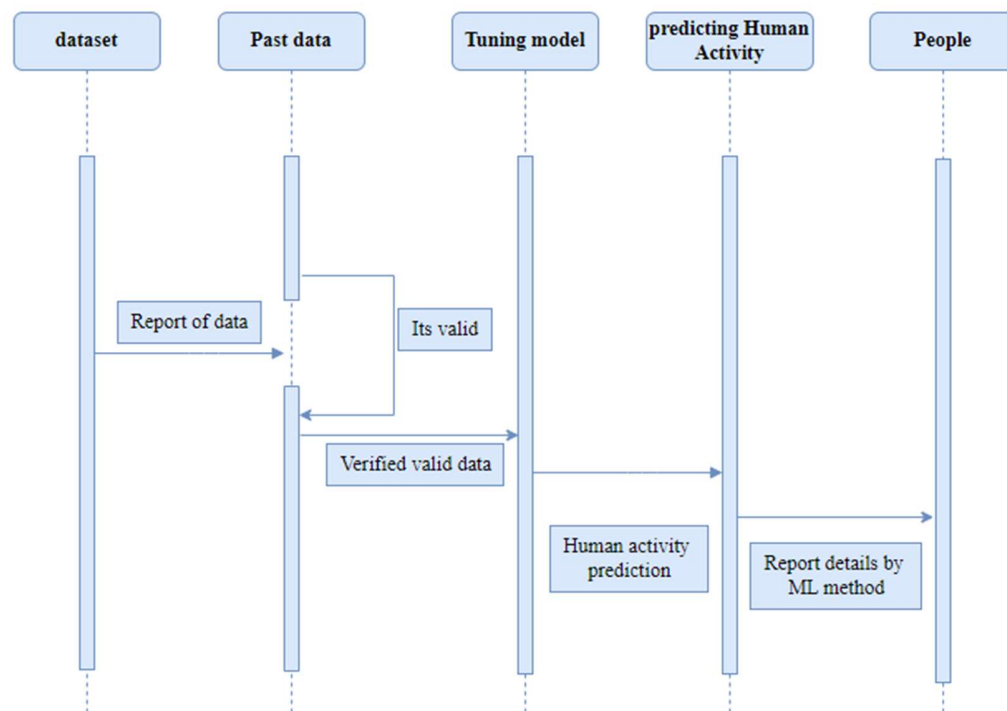


Fig 4.5 SEQUENCE DIAGRAM

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on

identifying the behavior within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

4.3.6 ER DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

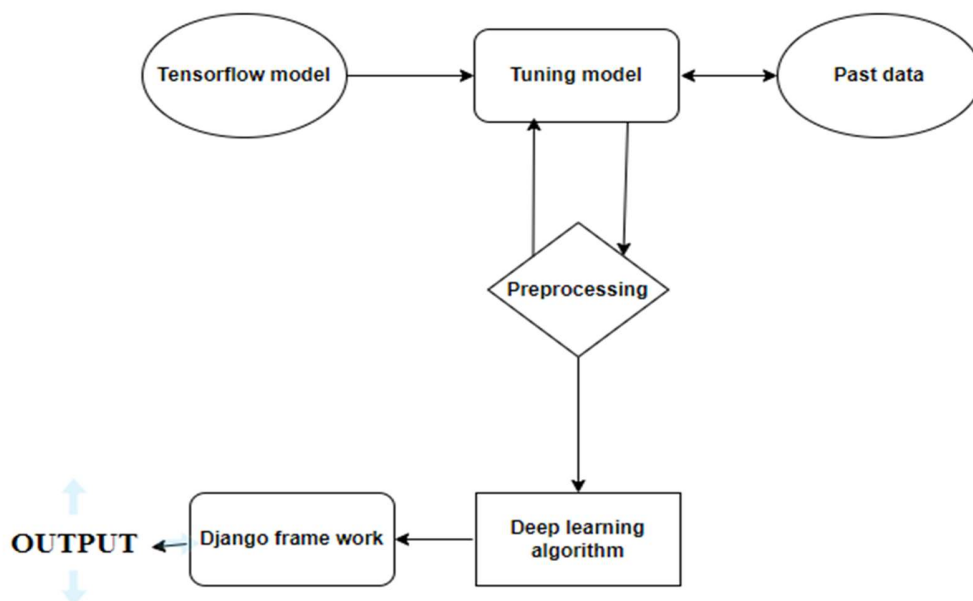


Fig 4.6 ER DIAGRAM

4.3.7 DATA SET USED

Good deep learning research starts with an exceptional dataset. The finest model or a high-fidelity system can only be produced by deep learning models, which are data-hungry and demand a lot of data. The quality of the data is just as crucial as the amount even if you have excellent deep learning techniques. The following quote best describes how a deep learning model works: Garbage In, Garbage Out (GIGO): If we feed low-quality data to an DL model, the result will be similar.

It was difficult to find a specific dataset for our project because there is no prearranged dataset for human activity recognition. We searched all over the famous dataset sources such as kaggle.com, registry.opendata.aws, and github.com. We had to manually collected data for our model, then manually we had arranged our images into specific folders. After arranging enough data images we divided it into two categories: testing and training.

The data's in each testing and training section are again classified as:

- Calling
- Clapping
- Cycling
- Dancing
- Drinking
- Eating
- Fighting
- Hugging
- Laughing

- Sitting

The model gains knowledge from the training set, which includes a given outcome, in order to subsequently generalize it to other data. It will utilise the Tensor flow module in Python and the Keras technique to evaluate our models using the test dataset (or subgroup). This method had consumed a significant amount of our time.



Fig 4.7 Dataset

4.4 Module Description

4.4.1 Import The Given Image From Dataset:

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of CNN.

4.4.2 To Train The Module By Given Image Dataset:

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.

4.4.3 Working Process Of Layers In Cnn Model:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

4.4.4 Input Layer:

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension $28 \times 28 = 784$, it need to convert it into 784×1 before feeding into input.

4.4.5 Convo Layer:

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

4.4.6 Pooling Layer:

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.

4.4.7 Fully Connected Layer (FC):

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

4.4.8 Softmax / Logistic Layer:

Softmax or Logistic layer is the last layer of CNN. It resides at the end of

FC layer. Logistic is used for binary classification and softmax is for multi-classification.

4.4.9 Output Layer:

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

4.5 Convolutional Neural Network

A Convolutional neural network (CNN) is one type of Artificial Neural Network. A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

4.5.1 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Types of Keras Models :

Models in keras are available in two types:

- Keras Sequential Model
- Keras Functional API

1. Sequential Model in Keras

It allows us to create models layer by layer in sequential order. But it does not allow us to create models that have multiple inputs or outputs.

It is best for simple stack of layers which have 1 input tensor and 1 output tensor.

This model is not suited when any of the layer in the stack has multiple inputs or outputs. Even if we want non-linear topology, it is not suited.

2. Functional API in Keras

It provides more flexibility to define a model and add layers in keras. Functional API allows us to create models that have multiple input or output.

It also allows us to share these layers. In other words, we can make graphs of layers using Keras functional API.

As functional API is a data structure, it is easy to save it as a single file that helps in recreating the exact model without having the original code. Also its easy to model the graph here and access its nodes as well.

Model Subclassing in Keras

Sequential model does not allow you much flexibility to create your models. Functional API also only has a little of customization available for you. But you may create your own fully-customizable models in Keras. This is done by subclassing the Model class and implementing a call method.

Input() is used to instantiate a Keras tensor.

A Keras tensor is a symbolic tensor-like object, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model.

For instance, if a, b and c are Keras tensors, it becomes possible to do: `model = Model(input=[a, b], output=c)`

4.5.2 kernels:

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations, each convolution is a type of affine function.

Padding:

To handle the edge pixels there are several approaches:

- Losing the edge pixels
- Padding with zero value pixels
- Reflection padding

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasingly worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

Strides:

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1,1), then stride to (1,3), then to (3,1) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one.

Rectified Linear Unit (ReLU):

A Rectified Linear Unit is used as a non-linear activation function. A ReLU says if the value is less than zero, round it up to zero.

Normalisation:

Normalisation is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between -1 and 1 making the data use the same scale, sometimes called Min-Max scaling.

It is common to normalize the input features, standardising the data by removing the mean and scaling to unit variance. It is often important the input features are centred around zero and have variance in the same order. With some data, such as images the data is scaled so that it's range is between 0 and 1, most simply dividing the pixel values by 255.

Batch normalisation:

Batch normalisation has the benefits of helping to make a network output more stable predictions, reduce overfitting through regularisation and speeds up training by an order of magnitude.

Batch normalisation is the process of carrying normalisation within the scope activation layer of the current batch, subtracting the mean of the batch's activations and dividing by the standard deviation of the batches activations.

This is necessary as even after normalizing the input as some activations can be higher, which can cause the subsequent layers to act abnormally and makes the network less stable.

Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

Importantly, batch normalization works differently during training and during inference.

During training (i.e. when using `fit()` or when calling the layer/model with the argument `training=True`), the layer normalizes its output using the mean and standard deviation of the current batch of inputs. That is to say, for each channel

being normalized, the layer returns $\gamma * (\text{batch} - \text{mean}(\text{batch})) / \sqrt{\text{var}(\text{batch}) + \epsilon} + \beta$, where:

- ϵ is a small constant (configurable as part of the constructor arguments)
- γ is a learned scaling factor (initialized as 1), which can be disabled by passing `scale=False` to the constructor.
- β is a learned offset factor (initialized as 0), which can be disabled by passing `center=False` to the constructor.

During inference (i.e. when using `evaluate()` or `predict()` or when calling the layer/model with the argument `training=False` (which is the default), the layer normalizes its output using a moving average of the mean and standard deviation of the batches it has seen during training. That is to say, it returns $\gamma * (\text{batch} - \text{self.moving_mean}) / \sqrt{\text{self.moving_var} + \epsilon} + \beta$.

`self.moving_mean` and `self.moving_var` are non-trainable variables that are updated each time the layer is called in training mode, as such:

- $\text{moving_mean} = \text{moving_mean} * \text{momentum} + \text{mean}(\text{batch}) * (1 - \text{momentum})$
- $\text{moving_var} = \text{moving_var} * \text{momentum} + \text{var}(\text{batch}) * (1 - \text{momentum})$

As such, the layer will only normalize its inputs during inference *after having been trained on data that has similar statistics as the inference data*.

TYPES OF CNN:

- LeNet
- ResNet

LeNet:

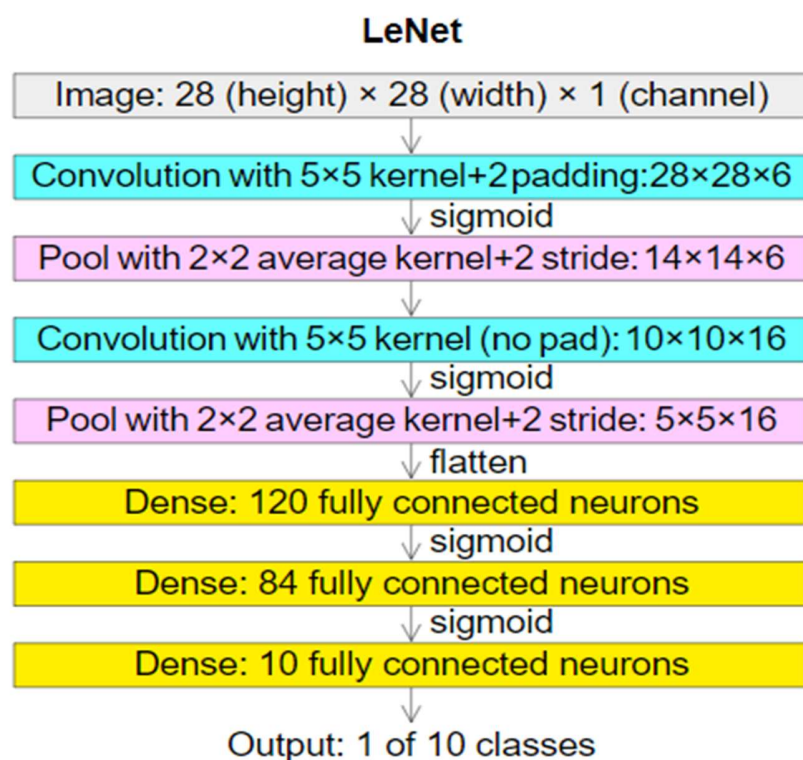
LeNet is a convolutional neural network structure proposed by LeCun et al. in

1998 . In general, LeNet refers to LeNet-5 and is a simple convolutional neural network. Convolutional neural networks are a kind of feed-forward neural network whose artificial neurons can respond to a part of the surrounding cells in the coverage range and perform well in large-scale image processing.

LeNet-5 was one of the earliest convolutional neural networks and promoted the development of deep learning. Since 1988, after years of research and many successful iterations, the pioneering work has been named LeNet-5.

As a representative of the early convolutional neural network, LeNet possesses the basic units of convolutional neural network, such as convolutional layer, pooling layer and full connection layer, laying a foundation for the future development of convolutional neural network. As shown in the figure (input image data with 32*32 pixels) LeNet-5 consists of seven layers. In addition to input, every other layer can train parameters.

Architecture of Lenet:



4.8 Architecture of LeNet

Convolutional layers:

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers:

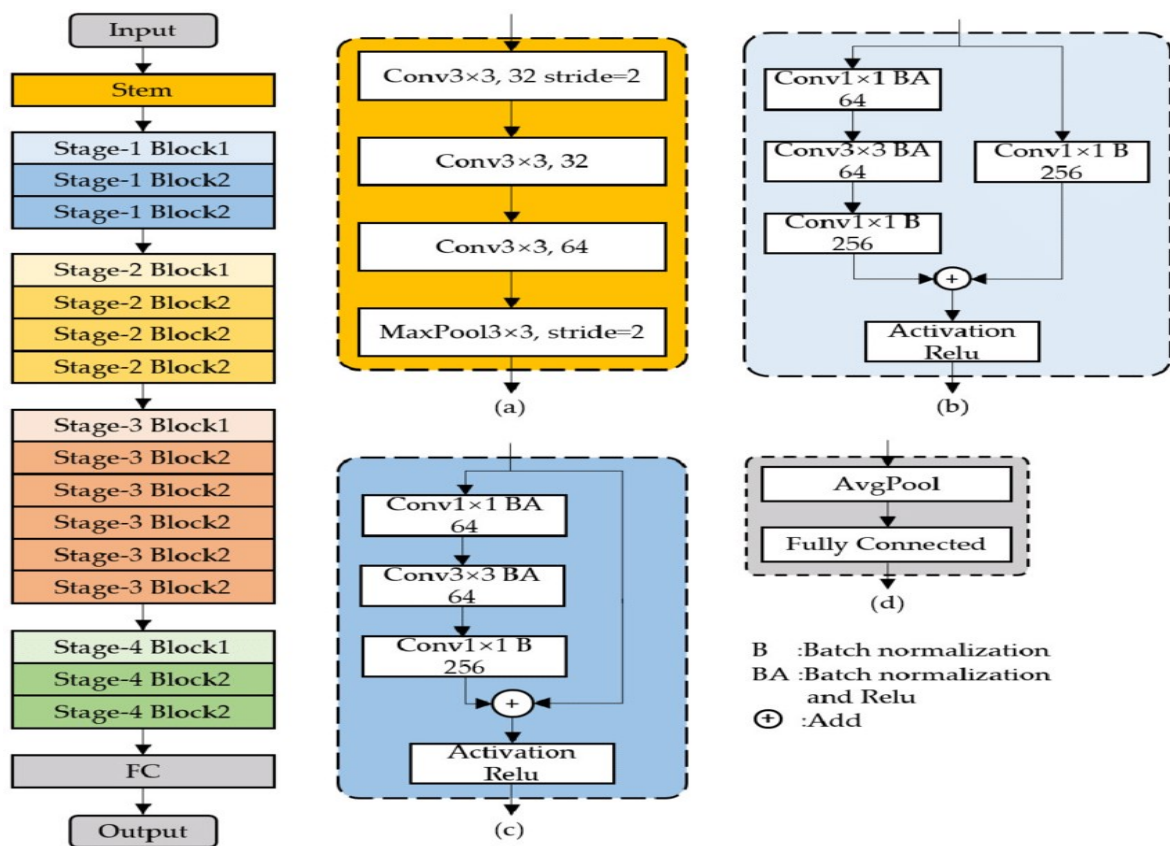
Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

ResNet:

ResNet is an artificial neural network that introduced a so-called “identity shortcut connection,” which allows the model to skip one or more layers. This approach makes it possible to train the network on thousands of layers without affecting performance. It’s become one of the most popular architectures for various computer vision tasks.

Architecture of ResNet:

There is a 34-layer plain network in the architecture that is inspired by VGG-19 in which the shortcut connection or the skip connections are added. These skip connections or the residual blocks then convert the architecture into the residual network as shown in the figure below.



4.9 Architecture of ResNet

Convolutional layers:

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers:

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

4.6 Libraries Required:

- ✓ **Tensorflow:** Just to use the tensor board to compare the loss and adam curve our result data or obtained log.
- ✓ **keras:** To pre-process the image dataset.
- ✓ **matplotlib:** To display the result of our predictive outcome.
- ✓ **os:** To access the file system to read the image from the train and test directory from our machines.

4.6.1 Tensorflow:

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow –

It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.

It includes a programming support of deep neural networks and machine learning techniques.

It includes a high scalable feature of computation with various data sets.

TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

4.6.2 Keras:

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features –

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backend.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

4.6.3 Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPython or Tkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source

equivalent of MATLAB.

4.6.4 OS:

The OS module in Python comes with various functions that enables developers to interact with the Operating system that they are currently working on. In this article we'll be learning mainly to create and delete a directory/folder, rename a directory and even basics of file handling.

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

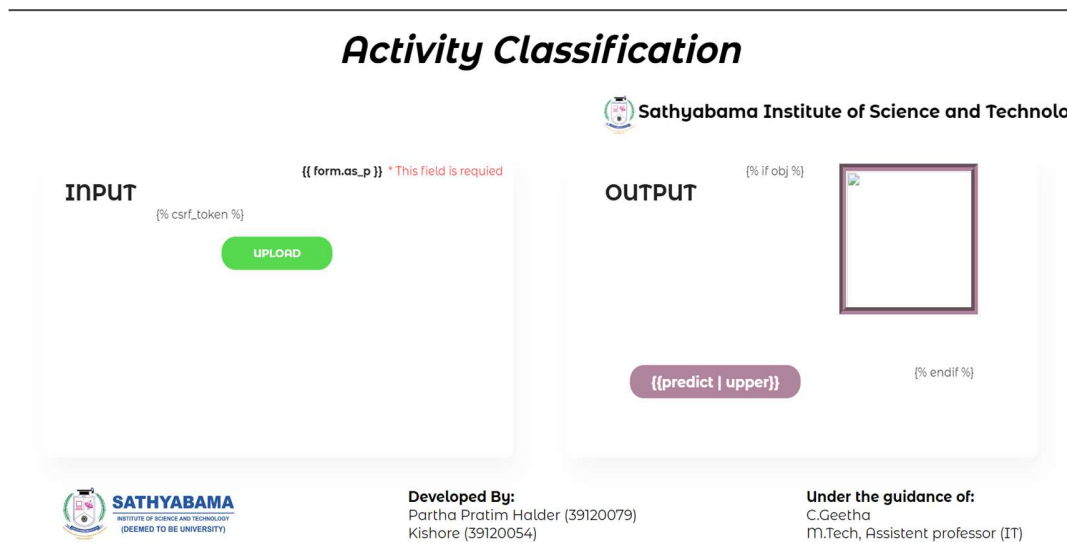
The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

CHAPTER 5

RESULT, CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Result

Here is what we have done, we have integrated our project and With the Django Framework we had created our front end and connected it with our Lenet Deep Learning Model as it shown the max Accuracy. Using Anaconda Prompt and visual studio we will run the Project. We now go to command prompt of V. studio and run the Django file which generates a localhost to access the front-end visual first look of our project. Once we enter our localhost, we need to upload image into the choose file option and the Deep learning will analyze the image and then generate the final desired output.



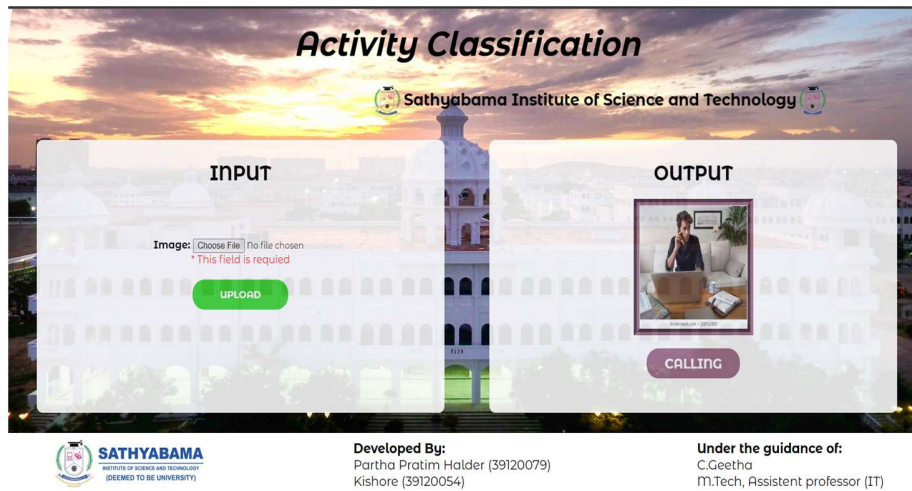


Fig 5.2 sample result for calling class



Fig 5.3 sample result for sitting class

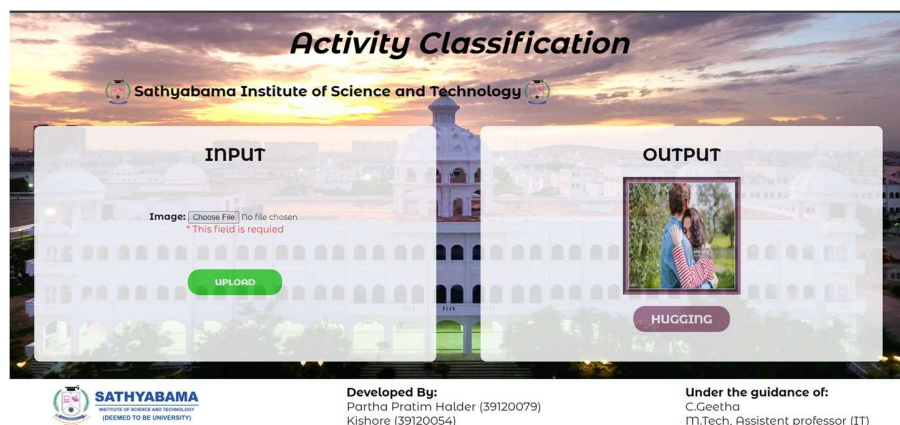


Fig 5.4 sample result for hugging class

5.1 Conclusion

We have proposed a Human activity Recognition system using deep learning techniques was developed. This is a complex problem that has already been approached several times with different techniques. While good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises. While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise on the input data. Nowadays, Human activity classification software includes the use of feature engineering. A solution totally based on feature learning does not seem close yet because of a major limitation. Thus, Human activity image classification could be achieved by means of deep learning techniques.

5.2 Future Enhancements

Human Activity Recognition can benefit various applications in fields like smart home monitoring, healthcare services, security surveillance, childcare etc. In future we can update this application by using object activity recognition in which activities performed by objects can also be tracked and analyzed. Application of integrated large datasets can be done to identify the activity taking place as slower rate of time. Even very subtle or minute variations should be recognized by the system. The data of actor performing the anomalous activity can be stored and identification of actor can be done if not caught in the first place. Activities that are of reoccurring manner should be stored to save time and space during recognition process. Implementation of such model can also be done in Government authority section. Much more developments for improvisation in accuracy and dealing with issues

related to optical identity and background clutter of image can be done.

REFERENCES

- [1] M. M. Arzani, M. Fathy, A. A. Azirani and E. Adeli, "Switching Structured Prediction for Simple and Complex Human Activity Recognition," in IEEE Transactions on Cybernetics, vol. 51, no. 12, pp. 5859-5870, Dec. 2021, doi: 10.1109/TCYB.2019.2960481.
- [2] Jie Yang, Jian Cheng and Hanqing Lu, "Human activity recognition based on the blob features," 2009 IEEE International Conference on Multimedia and Expo, 2009, pp. 358-361, doi: 10.1109/ICME.2009.5202508.
- [3] M. Z. Uddin, N. D. Thang and T. -S. Kim, "Human Activity Recognition via 3-D joint angle features and Hidden Markov models," 2010 IEEE International Conference on Image Processing, 2010, pp. 713-716, doi: 10.1109/ICIP.2010.5651953.
- [4] Y. Wang, H. Liu, K. Cui, A. Zhou, W. Li and H. Ma, "m-Activity: Accurate and Real-Time Human Activity Recognition Via Millimeter Wave Radar," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 8298-8302, doi: 10.1109/ICASSP39728.2021.9414686.
- [5] S. Perez-Gamboa, Q. Sun and Y. Zhang, "Improved Sensor Based Human Activity Recognition via Hybrid Convolutional and Recurrent Neural Networks," 2021 IEEE International Symposium on Inertial

- Sensors and Systems (INERTIAL), 2021, pp. 1-4, doi: 10.1109/INERTIAL51137.2021.9430460.
- [6] A. Gupta, A. Kembhavi and L. S. Davis, Observing Human-Object Interactions Using Spatial and Functional Compatibility for Recognition, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 10, pp. 1775-1789, Oct. 2009, doi 10.1109/TPAMI.2009.83.
- [7] S. Ullah and D. -H. Kim, Sparse Feature Learning for Human Activity Recognition, 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), 2021, pp. 309-312, doi 10.1109/BigComp51126.2021.00066.
- [8] S. S. Kumar and M. John, Human activity recognition using optical flow based feature set, 2016 IEEE International Carnahan Conference on Security Technology (ICCST), 2016, pp. 1-5, doi 10.1109/CCST.2016.7815694.
- [9] W. Li, X. Feng, Z. He and H. Zheng, "Human Activity Recognition Based on Data Fusion of FMCW Radar and Image," 2021 7th International Conference on Computer and Communications (ICCC), 2021, pp. 943-947, doi: 10.1109/ICCC54389.2021.9674543.
- [10] Y. Hanai, J. Nishimura and T. Kuroda, "Haar-Like Filtering for Human Activity Recognition Using 3D Accelerometer," 2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009, pp. 675-678, doi: 10.1109/DSP.2009.4786008.

APPENDICES

I. SOURCECODE

Manual architecture

```
# importing the necessary libraries
```

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

import tensorflow as tf
import glob
from PIL import Image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, MaxPooling2D, Flatten,
Convolution2D, Dropout, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
dir_name_train_calling='dataset/train/calling'
dir_name_train_cycling='dataset/train/cycling'
dir_name_train_dancing='dataset/train/dancing'
dir_name_train_fighting='dataset/train/fighting'
dir_name_train_hugging='dataset/train/hugging'
dir_name_train_laughing='dataset/train/laughing'
def image_details_print(data,path):
    print('===== Images in: ', path)
    for key,value in data.items():
        print(key,':\t', value)
```

```
def images_details(path):
    files=[f for f in glob.glob(path + "**/*.*", recursive=True)]
    data={}
    data['Images_count']=len(files)
    data['Min_width']=10**100
    data['Max_width']=0
    data['Min_height']=10**100
```

```

data['Max_height']=0

for f in files:
    img=Image.open(f)
    width,height=img.size
    data['Min_width']=min(width,data['Min_width'])
    data['Max_width']=max(width, data['Max_width'])
    data['Min_height']=min(height, data['Min_height'])
    data['Max_height']=max(height, data['Max_height'])

image_details_print(data,path)
def plot_images(path, n=6):
    files=[f for f in glob.glob(path + "**/*.*", recursive=True)]

    plt.figure(figsize=(80,40))
    for idx, img_path in enumerate(files[:n]):
        plt.subplot(7, n, idx+1) #splitting the image space so that all images can fit in
the space.
        img=plt.imread(img_path)
        plt.imshow(img)
        plt.axis('on')

    plt.tight_layout()
print("")
print('Trained data for calling')
print("")
images_details(dir_name_train_calling)
print("")
plot_images(dir_name_train_calling,12)
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2,
zoom_range=0.2)
test_datagen=ImageDataGenerator(rescale=1./255)
raining_set=train_datagen.flow_from_directory('dataset/train', batch_size=100,
class_mode='categorical', target_size=(255,255))
testing_set=test_datagen.flow_from_directory('dataset/test', batch_size=100,
class_mode='categorical', target_size=(255,255))
model=Sequential()

# Adding first convolution layer
model.add(Convolution2D(32,(3,3),input_shape=(255,255,3), padding='same',
activation='relu')) #rectified linear unit

# Adding first Maxpool layer
model.add(MaxPooling2D(pool_size=(2,2)))

# Adding Flatten layer

```

```

model.add(Flatten()) #flatten will covert 2D to 1D

# Adding input layer
model.add(Dense(units=164, activation='relu', )) #Dense for Neuron

# Adding dropout layer
model.add(Dropout(0.2))

# Adding Hidden layer
model.add(Dense(units=122, activation='relu'))
model.add(Dropout(0.2))

# Adding output layer
model.add(Dense(units=10, activation='softmax'))

model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

mc = ModelCheckpoint('Manual.h5', monitor = 'val_loss', save_best_only = True)
history=model.fit(training_set,
                  steps_per_epoch= training_set.samples//100,
                  validation_data=testing_set,
                  validation_steps= testing_set.samples//100,
                  epochs=10, callbacks=[mc])
def plot():
    # Plot training & validation accuracy values

    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    # Plot training & validation loss values

    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

```

plot()

LeNet Architecture

```
import glob
import pandas as pd
import tensorflow as tf
from tensorflow.keras.layers import Dense, Dropout, Activation, Convolution2D,
MaxPool2D, Flatten, Conv2D, BatchNormalization, MaxPooling2D,
AveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=
0.2,rotation_range=45 )
test_datagen=ImageDataGenerator(rescale=1./255)
Classifier=Sequential()
Classifier.add(Convolution2D(filters=6, kernel_size=(5,5), strides=(1,1),
input_shape=(225,225,3), padding='valid', activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
Classifier.add(Convolution2D(filters=128, kernel_size=(3,3), strides=(3,3),
padding='valid', activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
Classifier.add(Flatten())
Classifier.add(Dense(256, activation='relu'))
Classifier.add(Dense(10, activation='softmax'))

Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accu
racy'])
Classifier.summary()
epochs = 200
batch_size = 16
history=Classifier.fit( training_set, steps_per_epoch=training_set.samples //
batch_size,
                        epochs=epochs,
                        validation_data=testing_set,validation_steps=testing_set.samples //
batch_size,callbacks= [mc])
plt.figure(figsize=(20, 8))
```

```

plt.plot(history.history['loss'])

for i in range(epochs):
    if i%5 == 0:

plt.annotate(np.round(history.history['loss'][i]*100,2),xy=(i,history.history['loss'][i]))

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
import h5py                                #warning ::: to initilizing the resnet(best accuracy
model into UI)
Classifier.save('res.h5')

```

ResNet Architecture

```

import tensorflow as tf
import numpy as np
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D, MaxPool2D,
LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
import scipy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
from keras.initializers import glorot_uniform
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,
BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D,
GlobalMaxPooling2D
training_set=train_datagen.flow_from_directory('dataset/train',target_size=(224,224),

```

```

batch_size=16,class_mode='categorical')
test_set=test_datagen.flow_from_directory('dataset/test',target_size=(224,224),batch
_size=16,class_mode='categorical')
def identity_block(X, f, filters, stage, block):
    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Retrieve Filters
    F1, F2, F3 = filters

    # Save the input value. We'll need this later to add back to the main path.
    X_shortcut = X

    # First component of main path
    X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1), padding = 'valid',
name = conv_name_base + '2a', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    # Second component of main path
    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1), padding = 'same',
name = conv_name_base + '2b', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    # Third component of main path
    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1), padding = 'valid',
name = conv_name_base + '2c', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    # Final step: Add shortcut value to main path, and pass it through a RELU
activation
    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
def convolutional_block(X, f, filters, stage, block, s = 2):
    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Retrieve Filters
    F1, F2, F3 = filters

    # Save the input value

```

```

X_shortcut = X

##### MAIN PATH #####
# First component of main path
X = Conv2D(F1, (1, 1), strides = (s,s), name = conv_name_base + '2a',
kernel_initializer = glorot_uniform(seed=0))(X)
X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
X = Activation('relu')(X)

# Second component of main path
X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same',
name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
X = Activation('relu')(X)

# Third component of main path
X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid',
name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

##### SHORTCUT PATH #####
X_shortcut = Conv2D(F3, (1, 1), strides = (s,s), name = conv_name_base + '1',
kernel_initializer = glorot_uniform(seed=0))(X_shortcut)
X_shortcut = BatchNormalization(axis = 3, name = bn_name_base +
'1')(X_shortcut)

# Final step: Add shortcut value to main path, and pass it through a RELU
activation
addition=Add()
X = addition([X, X_shortcut])
X = Activation('relu')(X)

return X
def ResNet50(input_shape = (64, 64, 3), classes = 2):
    # Define the input as a tensor with shape input_shape
    X_input = Input(input_shape)

    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input)

    # Stage 1
    X = Conv2D(64, (7, 7), strides = (2, 2), name = 'conv1', kernel_initializer =
glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3)(X)

```

```

X = Activation('relu')(X)
X = MaxPooling2D((3, 3), strides=(2, 2))(X)

# Stage 2
X = convolutional_block(X, f = 3, filters = [64, 64, 256], stage = 2, block='a', s = 1)
X = identity_block(X, 3, [64, 64, 256], stage = 2, block='b',)
X = identity_block(X, 3, [64, 64, 256], stage = 2, block='c',)

# Stage 3
X = convolutional_block(X, f = 3, filters = [128, 128, 512], stage = 3, block='a', s =
2)
X = identity_block(X, 3, [128, 128, 512], stage = 3, block='b',)
X = identity_block(X, 3, [128, 128, 512], stage = 3, block='c',)
X = identity_block(X, 3, [128, 128, 512], stage = 3, block='d',)

# Stage 4
X = convolutional_block(X, f = 3, filters = [256, 256, 1024], stage = 4, block='a', s =
2)
X = identity_block(X, 3, [256, 256, 1024], stage = 4, block='b',)
X = identity_block(X, 3, [256, 256, 1024], stage = 4, block='c',)
X = identity_block(X, 3, [256, 256, 1024], stage = 4, block='d',)
X = identity_block(X, 3, [256, 256, 1024], stage = 4, block='e',)
X = identity_block(X, 3, [256, 256, 1024], stage = 4, block='f',)

# Stage 5
X = convolutional_block(X, f = 3, filters = [512, 512, 2048], stage = 5, block='a', s =
2)
X = identity_block(X, 3, [512, 512, 2048], stage = 5, block='b',)
X = identity_block(X, 3, [512, 512, 2048], stage = 5, block='c',)

# AVGPOOL.
X = AveragePooling2D((2, 2))(X)

# output layer
X = Flatten()(X)
X = Dense(classes, activation='softmax')(X)

# Create model
model = Model(inputs = X_input, outputs = X)

return model
model = ResNet50(input_shape = (225, 225, 3), classes = 10)
#### Fitting the model
history = model.fit(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,

```

```

        validation_data=test_set,validation_steps=test_set.samples // batch_size,
        callbacks= [mc])
plt.figure(figsize=(20, 8))
plt.plot(history.history['accuracy'])

for i in range(epochs):
    if i%5 == 0:

plt.annotate(np.round(history.history['accuracy'][i]*100,2),xy=(i,history.history['accuracy'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
plt.figure(figsize=(20, 8))
plt.plot(history.history['loss'])

for i in range(epochs):
    if i%5 == 0:

plt.annotate(np.round(history.history['loss'][i]*100,2),xy=(i,history.history['loss'][i]))

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()

```

#Django Frontend

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>FINAL YEAR PROJECT</title>
    <style>
        @import url('http://fonts.googleapis.com/css2?family=Montserrat
Alternates:wght@100;200;300;400;500;600;700;800;900&display=swap');
        *{
            margin:0px;
            padding:0px;
            box-sizing: border-box;
            font-family:'Montserrat Alternates','sans-serif';

```

```

}

body{
  background-image:url("static/DJI_0221.jpg");
  background-repeat: no-repeat;
  background-size: 100% 100%;
  width:100%;
}

h1{
  font-size: 50px;
  font-weight: 700;
  font-style: italic;
  text-align: center;
  padding: 10px;
  margin: 10px;
}

#container{
  width: 100%;
  height : 430px;
  display: flex;
  padding: 10px;
  justify-content:space-around;
  flex-wrap: wrap;
}

#container .block1{
  min-width: 400px;
  width: 45%;
  height: 400px;
  padding: 10px;
  margin: 10px;
  border: 0.5px solid rgb(255, 255, 255);
  border-radius: 10px;
  box-shadow: 10px 20px 30px rgba(0, 0, 0, 0.04);
  background-color: rgb(254, 254, 254);
  text-align: center;
  opacity: 87%;
}

#container .block2{
  min-width: 400px;
  width: 45%;
  height: 400px;
  padding: 10px;

```

```

margin: 10px;
border: 0.5px solid rgb(255, 255, 255);
border-radius: 10px;
box-shadow: 10px 20px 30px rgba(0, 0, 0, 0.04);
background-color: rgb(254, 254, 254);
text-align: center;
opacity: 87%;
}

```

```

#container .block2 h4{
text-align: center;
font-size: 20px;
color: white;
padding: 10px;
background-color: rgb(164, 114, 141);
border-radius: 20px;
width: fit-content;
}

```

```

#container .block2 img{
padding: 1px;
border : groove 10px rgb(164, 114, 141);
}

```

```

#container .block1 input[type=submit]{
height: fit-content;
border-radius: 20px;
border: 1px solid #3DD334;
background-color: #3DD334;
color: #FFFFFF;
font-size: 15px;
font-weight: bold;
padding: 12px 45px;
text-transform: uppercase;
display: block;
margin: 20px;
}

```

```

.marquee{
padding: 10px;
}

```

```

.marquee img{
vertical-align: middle;
height: 50px;
width: auto;
}

```

```

}

.heading{
    font-size: 30px;
    font-weight: bold;
    margin: 20px;
}

.marquee span{
    font-weight: bold;
    font-size: 25px;
}

#footer{
    background-color: rgb(255, 255, 255);
    width: 100%;
    height : fit-content;
    display: flex;
    padding: 10px 70px 10px 70px;
    justify-content:space-between;
    flex-wrap: wrap;
    margin-top: 20px;
}

#footer .logo img{
    height: 70px;
    width: auto;
}

#footer .name{
    font-size: 20px;
}

#footer .guide{
    font-size: 20px;
}
</style>
</head>
<body>
    <h1>Activity Classification</h1>
    <marquee class="marquee">
    <span>Sathyabama Institute of Science and Technology</span>
    
</marquee>
<div id="container">
  <div class="block1">
    <form action="" method="POST" enctype="multipart/form-data">
      <p class="heading">INPUT</p>
      <br><br><br>
      {% csrf_token %}
      <div style="margin-left: 75px;">
        <h4>{{ form.as_p }}</h4>
      </div>
      <p style="color:red;">* This field is required</p>
      <center><input type="submit" Value='UPLOAD' required></center>
    </form>
  </div>
  <div class="block2">
    <p class="heading">OUTPUT</p>
    {% if obj %}
      
      <br><br>
      <center><h4>&nbsp;{{predict | upper}}&nbsp;</h4></center>
    {% endif %}
  </div>
</div>
<div id="footer">
  <div class="logo">
    
  </div>
  <div class="name">
    <h4>Developed By:</h4>
    Partha Pratim Halder (39120079)<br>
    Kishore (39120054)
  </div>
  <div class="guide">
    <h4>Under the guidance of:</h4>
    C.Geetha <br>
    M.Tech, Assistent professor (IT)
  </div>
</div>
</body>
</html>

```

II. Screen Shots

Trained data for calling

```
===== Images in: dataset/train/calling
Images_count : 297
Min_width : 134
Max_width : 357
Min_height : 135
Max_height : 291
```



Trained data for cycling

```
===== Images in: dataset/train/cycling
Images_count : 308
Min_width : 177
Max_width : 414
Min_height : 121
Max_height : 285
```

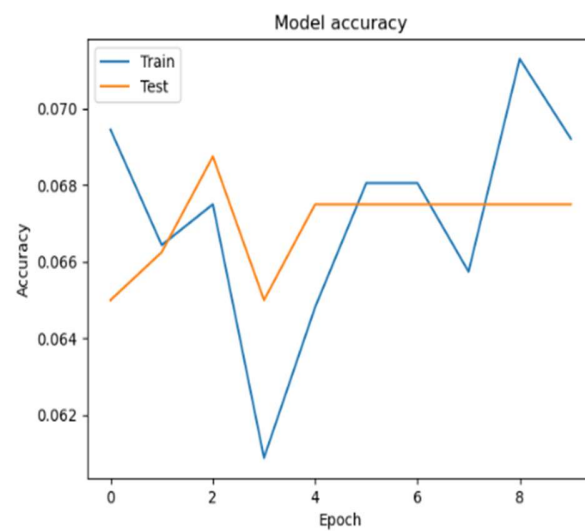


Trained data for fighting

```
===== Images in: dataset/train/fighting
Images_count : 297
Min_width : 154
Max_width : 346
Min_height : 104
Max_height : 300
```



Fig 6.01 importing images



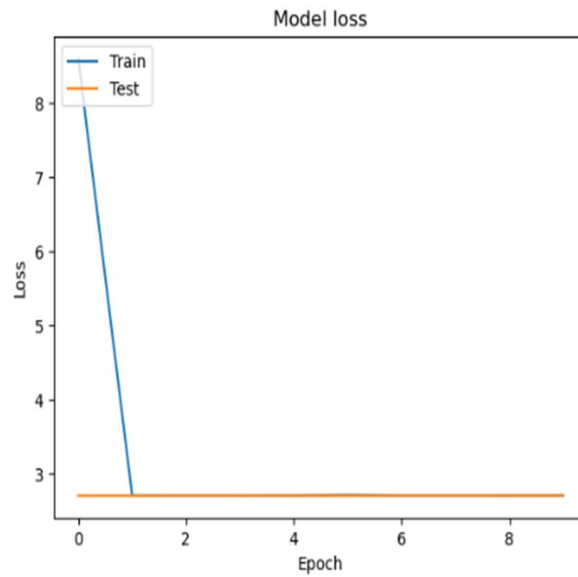


Fig 6.02 model accuracy plots

```

y: 0.2757
Epoch 168/200
276/276 [=====] - 84s 305ms/step - loss: 0.3289 - accuracy: 0.9183 - val_loss: 8.7854 - val_accurac
y: 0.3002
Epoch 169/200
276/276 [=====] - 84s 306ms/step - loss: 0.3698 - accuracy: 0.9171 - val_loss: 9.3329 - val_accurac
y: 0.2794
Epoch 170/200
276/276 [=====] - 84s 305ms/step - loss: 0.3498 - accuracy: 0.9173 - val_loss: 8.5342 - val_accurac
y: 0.2880
Epoch 171/200
276/276 [=====] - 84s 305ms/step - loss: 0.3650 - accuracy: 0.9155 - val_loss: 9.1142 - val_accurac
y: 0.2868
Epoch 172/200
276/276 [=====] - 84s 305ms/step - loss: 0.3152 - accuracy: 0.9214 - val_loss: 8.6688 - val_accurac
y: 0.2855
Epoch 173/200
276/276 [=====] - 84s 304ms/step - loss: 0.3207 - accuracy: 0.9221 - val_loss: 9.2787 - val_accurac
y: 0.2953
Epoch 174/200

```

```

Epoch 183/200
276/276 [=====] - 88s 320ms/step - loss: 0.3247 - accuracy: 0.9214 - val_loss: 10.1795 - val_accuracy: 0.2917
Epoch 184/200
276/276 [=====] - 85s 309ms/step - loss: 0.3372 - accuracy: 0.9194 - val_loss: 10.1525 - val_accuracy: 0.2831
Epoch 185/200
276/276 [=====] - 87s 314ms/step - loss: 0.3276 - accuracy: 0.9285 - val_loss: 9.9278 - val_accuracy: 0.3002
Epoch 186/200
276/276 [=====] - 87s 314ms/step - loss: 0.3193 - accuracy: 0.9237 - val_loss: 9.3446 - val_accuracy: 0.2868
Epoch 187/200
276/276 [=====] - 87s 316ms/step - loss: 0.3225 - accuracy: 0.9253 - val_loss: 11.1261 - val_accuracy: 0.2941
Epoch 188/200
276/276 [=====] - 88s 317ms/step - loss: 0.3596 - accuracy: 0.9201 - val_loss: 10.7212 - val_accuracy: 0.2892
Epoch 189/200
276/276 [=====] - 87s 317ms/step - loss: 0.3842 - accuracy: 0.9162 - val_loss: 8.0640 - val_accuracy: 0.2843
Epoch 190/200
276/276 [=====] - 88s 319ms/step - loss: 0.3333 - accuracy: 0.9257 - val_loss: 9.6206 - val_accuracy: 0.3088
Epoch 191/200
276/276 [=====] - 87s 315ms/step - loss: 0.3074 - accuracy: 0.9251 - val_loss: 10.0499 - val_accuracy: 0.2990
Epoch 192/200
276/276 [=====] - 86s 310ms/step - loss: 0.3338 - accuracy: 0.9228 - val_loss: 9.5910 - val_accuracy: 0.2794
Epoch 193/200
276/276 [=====] - 88s 317ms/step - loss: 0.3100 - accuracy: 0.9255 - val_loss: 9.9057 - val_accuracy: 0.2868
Epoch 194/200
276/276 [=====] - 85s 307ms/step - loss: 0.3423 - accuracy: 0.9271 - val_loss: 9.4132 - val_accuracy: 0.2770
Epoch 195/200
276/276 [=====] - 89s 323ms/step - loss: 0.3260 - accuracy: 0.9223 - val_loss: 10.8591 - val_accuracy: 0.2770
Epoch 196/200
276/276 [=====] - 89s 324ms/step - loss: 0.3483 - accuracy: 0.9237 - val_loss: 9.7863 - val_accuracy: 0.2647
Epoch 197/200
276/276 [=====] - 90s 328ms/step - loss: 0.3142 - accuracy: 0.9335 - val_loss: 9.9970 - val_accuracy: 0.2904
Epoch 198/200
276/276 [=====] - 92s 334ms/step - loss: 0.3489 - accuracy: 0.9262 - val_loss: 9.2257 - val_accuracy: 0.3002
Epoch 199/200
276/276 [=====] - 92s 335ms/step - loss: 0.3516 - accuracy: 0.9228 - val_loss: 10.4746 - val_accuracy: 0.2941
Epoch 200/200
276/276 [=====] - 89s 321ms/step - loss: 0.3131 - accuracy: 0.9307 - val_loss: 10.5644 - val_accuracy: 0.3039

```

Fig 6.03 model accuracy

III. Publication with Plagiarism Report

thesis.

by Turnitin Plagiarism Check

Submission date: 05-Mar-2023 07:54AM (UTC-0500)

Submission ID: 2029172258

File name: thesis.docx (766.54K)

Word count: 3983

Character count: 22142

SYSTEM FOR HUMAN ACTIVITY RECOGNITION WITH IMAGES USING DEEP LEARNING

~~Partha Pratim Haider~~

U.G Student, Dept. of Information Technology, Sathyabama University
Chennai, India – 600119
Email- pphaider.11@gmail.com

~~Kishore kumar~~

U.G Student, Dept. of Information Technology, Sathyabama University
Chennai, India – 600119
Email- kishoreashoke08@gmail.com

~~C. Geetha~~

M.Tech. (PH.D), Assistant professor, Dept. of Information Technology, Sathyabama University Chennai, India – 600119
Email- cgeetha.it@sathyabama.ac.in

ABSTRACT

Recognizing human behaviour in the realm of computer vision, HAR is an important but challenging research topic. In many applications, including surveillance, in-home monitoring, human-computer interaction, and health care, human activity recognition is essential.

To determine the specific subject activity in the image, we suggest context characteristics and a deep learning model in this work. To enhance the performance of recognition, we use images from google image search and used those images for our dataset. We create a deep neural network framework to provide a high-level depiction of human behaviour identification based on a picture collection. Recognizing human activity necessitates forecasting a person's activity using image-based information. The photos are divided into recognised activities. The goal is to forecast human activity using deep learning-based approaches with the highest degree of accuracy. For our project we will classify activities such as Calling, clapping, cycling, dancing, drinking, eating, fighting, hugging, laughing, sitting.

In order to categorize the images, we will utilize the convolutional neural network algorithm. To determine the ideal convolutional neural network architecture, initially we will use Residual Network and LeNet Architecture and compare these with our own (manual) architecture. We will select the architecture with the best accuracy.

Finally we will deploy the model with Django framework with some good to see User Interface.

Index Terms - Deep Learning, Images, Deep Neural Network, Human Activity, Convolution Neural Network, Classification.

1. INTRODUCTION

Human Computer Interaction (HCI) technology has impacted people's lives as a result of the rapid growth of artificial intelligence. Technology for recognising human behaviour is a crucial prerequisite for pleasant communication process.

One of the key elements of all automation apps that try to engage with a person is human activity recognition. Human activity recognition can be used for a variety of purposes, from improving human-computer

undertaking for a number of reasons, including the fact that people conduct activities in a variety of ways and that activities are frequently made up of multiple sub activities.

The systems and methods we use to communicate with computers are out-of-date and have a number of faults, despite the fact that they are now an essential part of life and are used in many various fields. In an attempt to handle these issues, a brand-new field called human computer interfaces has emerged.

Recognizing human activity involves action classification and action identification. the fundamental way in which humans interact with computers has not yet changed, despite significant advancements in both software and hardware. We still communicate with computers using a mouse and keyboard, or more sophisticated voice recognition systems, or even natural language processing in some cases. Contrasting how action detection and classification differ from one another. The category label for a given image only needs to be estimated for a classification task. We must estimate both the location of the action instance and its category for an action detection task or simply we can say that detection is the process of identification, while classification is the categorising of the object based on categories or classes that have already been established. While both are based on distinguishable characteristics of the object, categorization may have completely different boundaries depending on the problem domain.

The identification of human behaviour is a common topic for systems using machine vision. Human behaviour detection has become

increasingly accurate and lifelike as a result of advancements in artificial neural networks, such as the convolutional neural network (CNN). The recognition model engineering frequently utilizes convolutional brain organization to extricate spatial and fleeting attributes from the information picture. To accomplish the best precision, this work focuses on human activity detection using ResNet, LeNet and a manual architecture.

II.LITERATURE SURVEY

[1] Human Motion Detection with Deep Convolutional and Recurrent Models for Wearables

Author: Nils Y. Hammerla^{1,2}, Shane Halloran²

Year: 2016

In human activity identification (HAR) in pervasive computing, deep learning is beginning to displace well-established analytical methods that depend on hand produced feature extraction and categorization techniques. For problems varying from the division and recognizable proof of proactive tasks like running or rising steps to difficulties involving the detection of moving motions, it is difficult to comprehend the usefulness of bespoke deep structures.

They thoroughly research deep, neural, and recurrent methods using three sample datasets that comprise movement data obtained from external devices. We outline the training procedures for recurrent methods in this situation, present a fresh regularisation technique, and demonstrate how they beat the

cutting edge on a sizable standard dataset. We use huge number of acknowledgment tries different things with haphazardly inspected model setups to examine each model's suitability for human activity recognition. We then use the fANOVA framework to examine the effects of the hyperparameters. Finally, we provide recommendations for practitioners who want to use deep learning in their problem-solving.

[2] A Deep Learning Approach for Human Activity Recognition

Author: Susana Benavidez, Derek McCreight

Year: 2020

Every year, the hardware and sensors in smartphones and wearable devices become more powerful and precise; the valuable data collected by these sensors can be used for more precise Human Activity Recognition. In this article, we investigate the classification of 18 distinct tasks using convolutional neural networks and LSTMs, ranging from eating chips to dribbling a ball, using labelled, time-series gyroscope and accelerometer data from a smartwatch and smartphone. All of our work is available in our public repo, and the processed data is available here.

[3] Human activity recognition with smartphone sensors using deep learning neural networks

Author: Charissa Ann Ronao, Sung-Bae Cho*

Year: 2019

Translation constant and hierarchy behaviour is a natural part of human activity. Time-series sensing data is used to deduce actions in the

area of human activity identification (HAR), which, as its use becomes more widespread, has been the subject of increased research and development in recent years. In this article, we show how to use the sensors on your smartphone to its full potential for HAR by capitalising on the unique properties of actions and 1D time-series data. While doing so, a technique is given for autonomously and data-adaptively pulling trustworthy characteristics from unprocessed data. Despite a diminishing difference in feature complexity with each additional layer, studies show that convnets generate increasingly important and complicated features as the number of layers increases. It is possible to use a larger time range of temporal local correlation (1–9–14), and it has been demonstrated that a small pooling size (1–2–13) is advantageous. Additionally, Convnets classified movement activities almost perfectly, particularly those that were comparable and were previously thought to be very challenging to categorise. In terms of human action, convnets work better than other cutting-edge data mining methods. Accomplished a total score of 94.79% on the test set utilizing crude sensor information and 95.75% with additional data from the transient quick Fourier change of the HAR informational index, the two of which were perceived for the benchmark dataset gathered from 30 energetic members.

[4] Machine Learning Algorithms for Human Activity Recognition

Author: K.R.Baskaran, M.N.Saroja

Year: 2019

By keeping note of their daily actions and helping those with serious health problems, it is crucial to watch the Activity of Daily Living (ADL) of senior people who live alone. In this study, various machine learning methods for recognising human behaviour are investigated. Additionally, a lot of study is being done to find out more about the tools that are currently being used for movement detection. Typically, signals produced by sensors are used to identify activities. The data are then pre-processed, divided, characteristics are taken, and activity is identified. The Human Activity Detection System's primary objective is to examine the constraints of independent aged people and recommend solutions for them.

[5] Video-based human action recognition using deep learning

Author: Huy-Hieu Pham · Louahdi Khoudour
Year: 2019

An essential area of computer vision application is human movement detection. The primary goal is to provide an exact description of human actions and encounters based on a series of data never before seen by sensors. Having the ability to recognise, comprehend, and anticipate complicated human actions has far-reaching implications, including the development of intelligent monitoring systems, human-computer interactions, healthcare, security, and combat uses. The computer vision industry has recently paid deep learning a lot of attention. This paper surveys the current state of the art for action recognition through the lens of video analysis and deep learning techniques. In this article, we introduce the most important

deep learning models for human action recognition and analyse their strengths and flaws to demonstrate the current status of deep learning algorithms used to address practical challenges in human action recognition. By conducting a quantified analysis of the identification accuracies described in the literature, we are able to identify state-of-the-art deep structures in the field of action recognition and show current trends and unresolved issues for future works in this area.

[5] Human Activity Recognition Based On The Blob Features

Author: Jie Yang, Jian Cheng, Hanqing Lu
Year: 2019

The authors of this article describe a new method for identifying human activity in film. Since human actions can be considered to be a time entity with a number of frames, we analyse them in succession. We first build a statistical background model and then use background removal to isolate visible items from the video feed. After that, we create a new feature picture based on a set of guidelines using closer view objects from the current edge and a few casings before it. The non-zero pixels in the feature picture are then combined into clumps using the linked component technique. Then, for every glob, there is an action that can be identified by the way the blob looks. By identifying amorphous characteristics, we can recognise actions. Gaussian Mixture is used in this model to account for each category of human action, and distance measure is used to determine closeness.

[5] Human Activity Recognition via 3-D Joint Angle Features and Hidden Markov Models

Author: Md. Zia Uddin, Nguyen Duc Thang, and Tae-Seong Kim

Year: 2018

In this article, a new method for Human Activity Identification (HAR) is presented, which is based on the 3-D joint orientations of the human body. Our approach does not depend on instruments affixed to the person; All things being equal, a 3D body model is co-enrolled to the sound system data from each gathering of movement video pictures taken by the sound system camera to gauge the body joint directions. To start with, we decipher the expected joint point highlights from the time-successive action video outlines into codewords, which forms the basis of the concealed markov model for that activity (HMM). Each activity HMM is learned and then used to identify activities based on these patterns.

As we investigated further, we found that our combined angle-based Human activity Detection yielded significantly improved results in the identification rate, particularly for those activities that are not detectable with traditional methods. Indeed, when we compared the efficiency of our joint angle-based Human activity Recognition to that of the standard binary silhouette-based Human activity Recognition, we found the same thing to be true.

INFERENCES FROM LITERATURE SURVEY

It is possible to see from the aforementioned material that significant study has been done on this subject and that many instances have been put forth. Of course, each of the aforementioned methods has advantages and disadvantages. Even though some of the work utilises the contemporary mix period and provides greater precision, it is still some distance from what is desired. Higher precision necessitates reduced numerical expenses, high-quality working speed, and, especially, simplicity of use.

Human action recognition has been extensively researched for many years. The majority of action recognition methods necessitate manual annotation of the action. In recent years, there has been a great deal of research into action recognition. There is a distinction between action recognition and action detection. The most effective method is deep learning.

III. OPEN PROBLEMS IN EXISTING SYSTEM

Human endeavour Numerous uses, such as tracking, in-home monitoring, and medical treatment, depend on recognition. Portable radar sensors, in conjunction with deep learning, are increasingly being used in Human activity Recognition systems (DL).

However, obtaining a large-scale radar dataset with reliable labels is both difficult and time-consuming. Inadequate labelled data frequently limits the generalisation of Deep Learning models. As a result, when applied to a new scenario, the performance of DL models will

suffer. In this regard, labelling only a small portion of the large-scale radar dataset is more feasible.

Drawbacks

- They are not using CNN Algorithm for getting better prediction.
- They are using un-labeled data for classification, so prediction can vary from the real activity.
- Deployment cannot be done here.

IV. PROPOSED SYSTEM

To classify human activity in order to create a deep learning technique. It proposed an image-based system for predicting human activity. More image samples are collected and compared with different classes. The shape and texture-oriented features are the image's primary attributes.

We are initially preprocessing our dataset and implementing multiple convolutional neural network architectures. Because each architecture provides a different level of accuracy, we compare them. After saving the model in.h5 format. We don't need to train the architecture over and over again. When we have trained the model and obtained the accuracy, we save it in hierarchical file format. You can now check the precision whenever you want.

Advantages:

- We are using convolutional neural network algorithm for better image understanding.
- We can find the activity based on the image dataset, so prediction can be

effective when compared to labeled or unlabeled data.

- Deployment done using Django framework

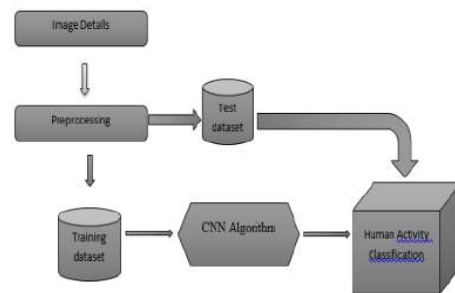


Fig I: Block diagram of proposed method

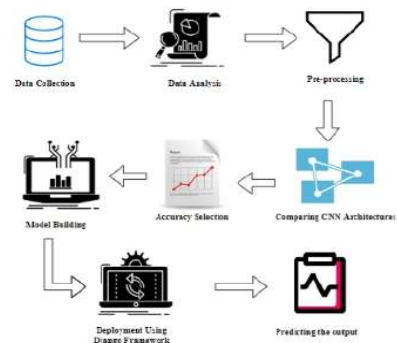


Fig II: System Architecture

DATA SET USED

Good deep learning research starts with an exceptional dataset. The finest model or a high-fidelity system can only be produced by deep learning models, which are data-hungry and demand a lot of data. The quality of the data is just as crucial as the amount even if you have excellent deep learning techniques. The

For the training we will use Residual Network and LeNet Architecture and one more manual architecture. We will select the architecture with the best accuracy.

IMPORTING IMAGES

Using the pre-processing picture data creator tool, we must first load our data collection into Keras while also creating size, rescale, range, zoom range, and horizontal reverse. Then, we load our picture dataset from the subdirectory using the data creator tool. Here, we define the goal size, group size, and class mode as well as the train, test, and validation settings. We must practise using our own network that we built by adding CNN levels.



Fig V: imported images and details for calling, cycling, fighting

TRAINING THE MODEL

We employ the classifier and match generator functions, as well as the training steps per

epoch, total number of epochs, validation data, and validation steps, to train our dataset.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 255, 255, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
flatten (Flatten)	(None, 516128)	0
dense (Dense)	(None, 164)	84645156
dropout (Dropout)	(None, 164)	0
dense_1 (Dense)	(None, 122)	20130
dropout_1 (Dropout)	(None, 122)	0
dense_2 (Dense)	(None, 15)	1545

=====
Total params: 84,668,027
Trainable params: 84,668,027
Non-trainable params: 0

Fig VI: CNN Model Summary details

WORKING PROCESS OF LAYERS

To distinguish between similar images, a Convolutional Neural Network (ConvNet/CNN) can take in an input image, assign weights and biases that can be taught to various characteristics and objects in the image, and then output a single label. When compared to competing classification strategies, ConvNet requires significantly less pre-processing work. Even though ConvNets don't start out with the criteria and characteristics used in advanced methods, they can learn them with enough training. The ConvNet, similar to the associating organization of neurons in the human cerebrum, was motivated by the association of the Visual Cortex. In any case, individual neurons just answer improvements in a tiny region of the vision field called the Open Field.

A convolutional brain network has 1,024 info units complete, with 256 in the info layer, 8 in

the main secret layer, and 2 in the last secret layer before the output.

Input Layer: The CNN input layer stores image data. The representation of picture data is done using three-dimensional vectors. It needs to be redesigned as a single section. Before feeding it into the input, a picture with the measurements $28 \times 28 = 784$ must be changed to 784×1 .

Convo Layer: Because it extracts picture characteristics, Convolutional network (ConvNet) layer, or ConvNet, is a feature extraction layer. An image region is connected to the Convo layer so that the previously observed convolution process can be carried out, and the dot product between the filter and the receptive field (a region of the inbound image with the same dimensions as the filter) can be calculated. A singular number that represents the output amount is the result of the procedure. Repeat the process by applying a Stride filter over the following receptive field of the same incoming picture. It will keep doing this until the complete picture has been processed. The input for the following layer will be the result.

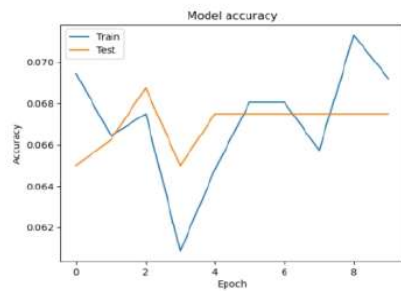


Fig VII: trained dataset accuracy

Pooling Layer: To decrease the geographic volume of the original picture after convolution, a pooling layer is applied. The centre of two convolution layers is where it is used. If FC is applied after the Convo layer without using pooling or maximum pooling, it will require a lot of processing resources. Thus, the original image's spatial volume can only be decreased using max pooling. In a single depth segment with a Span of 2, it used the greatest amount of pooling. The 4×4 input's reduction to 2×2 measurements can be seen.

Fully Connected Layer (FC): The completely linked layer includes synapses, weights, and biases. It connects synapses from different layers to one another. It has been taught to categorise pictures into various groups.

Logistic Layer: Logistic layer (also known as Softmax layer) is the last layer of a convolutional neural network. As its name suggests, it can be found at the base of the FC layer. For multiple classes, use softmax, while for binary classes, use logistic.

Output Layer: The name is contained in the output component, and we can use it to publish using the Django framework.

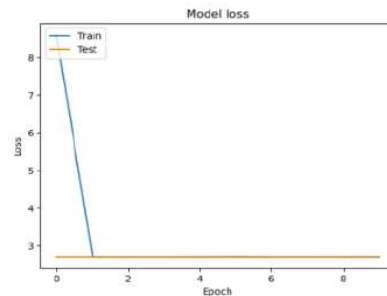


Fig VIII: trained dataset loss values

VI. DEPLOYMENT

We are deploying our model using Django framework. There are many options in Django to make the lives of web workers simpler, but none of those tools are useful if you can't quickly launch your sites. The aim of Django has been to make distribution simple ever since it was created.

In our deployment model we are creating a insertion button via which we will insert our image which we need to classify and in return we will get the inserted image as well as the classified activity as our output.



Fig IX. User interface for our model

VII. CONCLUSION

A study to classify human activity photos This initiative pioneered the use of deep learning methods. This is a challenging issue that has already been dealt with several times using different approaches. This endeavour focused on feature learning, one of DL's goals, even though excellent outcomes have been achieved using feature engineering. Image pre-processing increases categorization success even though feature building is not necessary. Consequently, it reduces disturbance in the

incoming data. Feature engineering is being used in human activity classification software. Because of a significant constraint, a solution entirely based on feature learning does not appear to be imminent. Deep learning algorithms could thus be used to classify human activity images.

VIII. FUTURE WORK

The tracking of smart homes, healthcare services, security video, daycare, and other uses are just a few examples of how Human Activity Detection can be helpful. The addition of object activity detection, which enables us to monitor and examine object behaviour, will enable us to enhance this application in the future. Identifying actions happening at a slowing rate of time can be done by using combined, enormous databases. Even the slightest or smallest changes should be detectable by the algorithm. Data on the actor engaging in the aberrant behaviour can be stored and used to identify the actor even if they are not initially discovered.

IX. REFERENCES

- [1] M. M. Arzani, M. Fathy, A. A. Azirani and E. Adeli, "Switching Structured Prediction for Simple and Complex Human Activity Recognition," in IEEE Transactions on Cybernetics, vol. 51, no. 12, pp. 5859-5870, Dec. 2021, doi: 10.1109/TCYB.2019.2960481.
- [2] Jie Yang, Jian Cheng and Hanqing Lu, "Human activity recognition based on the blob features," 2009 IEEE International Conference on Multimedia and Expo, 2009,

- pp. 358-361, doi: 10.1109/ICME.2009.5202508.
- [3] M. Z. Uddin, N. D. Thang and T. -S. Kim, "Human Activity Recognition via 3-D joint angle features and Hidden Markov models," 2010 IEEE International Conference on Image Processing, 2010, pp. 713-716, doi: 10.1109/ICIP.2010.5651953.
- [4] Y. Wang, H. Liu, K. Cui, A. Zhou, W. Li and H. Ma, "m-Activity: Accurate and Real-Time Human Activity Recognition Via Millimeter Wave Radar," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 8298-8302, doi: 10.1109/ICASSP39728.2021.9414686.
- [5] S. Perez-Gamboa, Q. Sun and Y. Zhang, "Improved Sensor Based Human Activity Recognition via Hybrid Convolutional and Recurrent Neural Networks," 2021 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), 2021, pp. 1-4, doi: 10.1109/INERTIAL51137.2021.9430460.
- [6] A. Gupta, A. Kembhavi and L. S. Davis, "Observing Human-Object Interactions Using Spatial and Functional Compatibility for Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 10, pp. 1775-1789, Oct. 2009, doi 10.1109/TPAMI.2009.83.
- [7] S. Ullah and D. -H. Kim, "Sparse Feature Learning for Human Activity Recognition," 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), 2021, pp. 309-312, doi 10.1109/BigComp51126.2021.00066.
- [8] S. S. Kumar and M. John, "Human activity recognition using optical flow based feature set," 2016 IEEE International Conference on Security Technology (ICCST), 2016, pp. 1-5, doi 10.1109/ICCST.2016.7815694.
- [9] W. Li, X. Feng, Z. He and H. Zheng, "Human Activity Recognition Based on Data Fusion of FMCW Radar and Image," 2021 7th International Conference on Computer and Communications (ICCC), 2021, pp. 943-947, doi: 10.1109/ICCC54389.2021.9674543.
- [10] Y. Hanai, J. Nishimura and T. Kuroda, "Haar-Like Filtering for Human Activity Recognition Using 3D Accelerometer," 2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009, pp. 675-678, doi: 10.1109/DSP.2009.4786008.

thesis.

ORIGINALITY REPORT

7%

SIMILARITY INDEX

4%

INTERNET SOURCES

4%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

www.semanticscholar.org

Internet Source

1%

2

Uddin, Md. Zia, Nguyen Duc Thang, and Tae-Seong Kim. "Human Activity Recognition via 3-D joint angle features and Hidden Markov models", 2010 IEEE International Conference on Image Processing, 2010.

Publication

1%

3

Submitted to University of North Texas

Student Paper

1%

4

www.researchgate.net

Internet Source

1%

5

Jharge S.R. Melvin, B. Rokesh, S. Dheepajyothieshwar, K. Akila. "Driver Yawn Prediction Using Convolutional Neural Network", Trans Tech Publications, Ltd., 2023

Publication

<1%

6

machinelearningmastery.com

Internet Source

<1%

7

link.springer.com

	Internet Source	<1 %
8	Submitted to University of Wollongong Student Paper	<1 %
9	scholar.archive.org Internet Source	<1 %
10	Submitted to Al-Nahrain University Student Paper	<1 %
11	huggingface.co Internet Source	<1 %
12	Jianfei Yang, Yuecong Xu, Haozhi Cao, Han Zou, Lihua Xie. "Deep learning and transfer learning for device-free human activity recognition: A survey", Journal of Automation and Intelligence, 2022 Publication	<1 %
13	docplayer.net Internet Source	<1 %
14	www.mdpi.com Internet Source	<1 %
15	Lecture Notes in Computer Science, 2015. Publication	<1 %
16	Wenlong Li, Xinxin Feng, Zhao He, Haifeng Zheng. "Human Activity Recognition Based on Data Fusion of FMCW Radar and Image", 2021	<1 %

7th International Conference on Computer and Communications (ICCC), 2021

Publication

17	www.e-sciencecentral.org Internet Source	<1 %
18	Maryam Koohzadi, Nasrollah Moghadam Charkari. "Survey on deep learning methods in human action recognition", IET Computer Vision, 2017 Publication	<1 %
19	irep.ntu.ac.uk Internet Source	<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography On