HUMAN ACTIVITY RECOGNITION USING DEEP LEARNING

A mini project report submitted for the partial fulfilment of the requirement for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Submitted By

S. Noor Basha (Reg.No.22691F00B2)

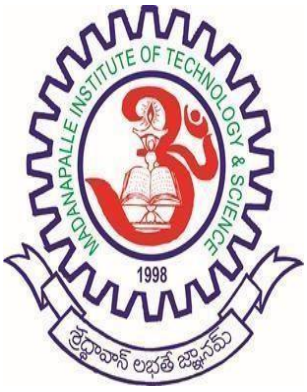
C. Rajesh (Reg.No.22691F00C7)

Under the Guidance of

Dr. N.Nirmala Devi, MCA., Ph.D.,

Assistant Professor

Department of Computer Applications



DEPARTMENT OF COMPUTER APPLICATIONS

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE MADANAPALLE

(UGCAUTONOMOUS)

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

www.mits.ac.in

2023–2024

MADANAPALLEINSTITUTEOFTECHNOLOGY&SCIENCEMADANAPALLE

(UGCAUTONOMOUS)

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

DEPARTMENT OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

This is to certify that the Mini project work entitled is a Bonafide work carried out by **S. Noor Basha**(R**eg.No.22691F00B2), C. Rajesh** (**Reg.No.22691F00C7)** submitted in partial fulfilment of the requirements for the award of the degree of **Master of Computer Applications** in **Madanapalle Institute of Technology and Science, MADANAPALLE**, affiliated to **Jawaharlal Nehru Technological University, Anantapuramu** during the **academic year 2023-2024.**

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
Dr. N. Nirmala Devi , MCA., Ph.D.,	**Dr. N. Naveen Kumar, MCA., Ph.D.,**
Assistant Professor	Associate Professor & HOD
Department of Computer Applications	Department of Computer Applications

DECLARATION

We, **S. Noor Basha** (R**eg.No.22691F00B2),C. Rajesh** (**Reg.No.22691F00C7)** hereby declare that the project entitled **"HUMAN ACTIVITY RECOGNITION USING DEEP LEARNING**" is done by us under the guidance of **Dr. N. Nirmala Devi, MCA., Ph.D., Assistant Professor,** submitted in partial fulfilment of the requirements for the award of the degree of Master of Computer Applications at **MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE, MADANAPALLE,** affiliated to **Jawaharlal Nehru Technological University, Anantapuramu,** during the academic year 2023-2024. This work has not been submitted by anybody to the award of any degree.

Date:

Place: Madanapalle **Signature of the Students**

ACKNOWLEDGEMENT

First, I must thank the almighty who has granted me knowledge, wisdom, strength, and courage to serve in this world and carry out my project work in a successful way.

We express our sincere thanks to **Dr. N. Vijaya Bhaskar Choudhry, Ph.D**. **Secretary & Correspondent**, Madanapalle Institute of Technology & Science for his continuous encouragement towards practical education and constant support in all aspects, including the provision of very good infrastructure facilities in the institute.

It is my duty to thank our **Principal**, **Dr. C. Yuvaraj**, Madanapalle Institute of Technology & Science, for his guidance and support at the time of my course and project.

I also wish to express my thanks to **Dr. P. Ramanathan, VP-Academics** for his continuous support throughout my MCA career.

It is my foremost duty to thank the **Head of the Department, Dr. N. Naveen Kumar, MCA, Ph.D., Associate Professor,** who gave me constant support during the project time and continuous encouragement towards the completion of the project successfully.

We thank our faculty guide **Dr.N. Nirmala Devi., MCA., Ph.D.**, **Assistant Professor**, for her continuous support in conducting periodical reviews and guidance until the completion of the project in a successful manner.

.

ABSTRACT

The primary objective of our project is to develop a system capable of detecting and categorizing human movements and actions through image detection sourced from websites or local computers. This involves treating each image as an object and focusing on key human poses such as sitting, sleeping, standing, and bending. Human Activity Recognition (HAR) is crucial in various domains like healthcare, sports analysis, and smart environments. In this paper, we propose an innovative HAR approach utilizing Convolutional Neural Networks (CNNs) to automatically classify and recognize human activities based on sensor data.

Our system leverages a deep learning architecture that effectively captures intricate spatial and temporal patterns in sequential data, making it well-suited for the complexity of human activities. The input data comprises time-series sensor readings, such as accelerometer and gyroscope data, obtained from wearable devices or other sources. The CNN architecture eliminates the need for manual feature engineering by autonomously learning hierarchical features from raw sensor data.

To assess the model's performance, we employ a comprehensive dataset encompassing diverse human activities, including walking, running, sitting, and other common daily tasks. The CNN model is trained on labelled data and rigorously tested, demonstrating high accuracy and robustness in recognizing various activities. The results showcase the effectiveness of our CNN-based HAR system, highlighting its potential for real-world applications. Its adaptability to different activities, environments, and individuals positions it as a promising solution for human activity recognition in diverse scenarios, contributing to the advancement of intelligent systems and assistive technologies. The integration of CNNs not only enhances accuracy but also establishes a scalable and automated framework for activity recognition.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The problem addressed in this project revolves around the accurate and automated recognition of human activities through the utilization of Convolutional Neural Networks (CNNs). In the realm of Human Activity Recognition (HAR), the challenge lies in developing a system capable of discerning complex activities from sequential sensor data, specifically accelerometer and gyroscope readings from wearable devices or other sources. The traditional hurdle involves the manual engineering of features, which is labour-intensive and may not capture intricate spatial and temporal patterns inherent in human activities.

The objective is to employ CNNs to address these challenges by allowing the model to autonomously learn hierarchical features from raw sensor data. The problem encompasses designing an architecture that spatially scans and captures local patterns through convolutional layers, while subsequent layers integrate these features to recognize more complex and abstract patterns associated with various human activities.

The project aims to overcome the limitations of existing approaches by providing a scalable, automated, and accurate framework for activity recognition. The challenge is to train a CNN model on diverse datasets containing different human activities and evaluate its performance in terms of accuracy, robustness, and adaptability across various scenarios, environments, and individuals. Ultimately, the goal is to contribute to the advancement of intelligent systems and assistive technologies through the effective integration of CNNs in Human Activity Recognition.

1.2 Process Description

Human Activity Recognition (HAR) using Convolutional Neural Networks (CNNs) involves a systematic process for accurate and automated classification of human actions. The initial step encompasses data collection, wherein time-series sensor readings, such as accelerometer and gyroscope data, are obtained from wearable devices or other sources. Subsequently, a comprehensive dataset comprising diverse human activities is curated for training and evaluation purposes. The CNN architecture is designed to automatically learn hierarchical features from the raw sensor data, eliminating manual feature engineering.

The training phase involves feeding labelled data into the CNN model, enabling it to discern patterns associated with various activities. Rigorous testing follows to assess the model's accuracy and robustness in recognizing distinct human actions. The convolutional layers spatially scan and capture local patterns, while subsequent layers integrate these features to recognize more complex and abstract activity patterns. The results demonstrate the effectiveness of the CNN-based HAR system, showcasing its potential for real-world applications in diverse scenarios, contributing to the advancement of intelligent systems and assistive technologies.

CHAPTER 2

SYSTEM ANALYSIS

2.1. Existing System

The Ancient way of monitoring human activities was a manual process, with someone having to sit in front of a monitor to guide and keep an eye on things. This was not only tiring and time-consuming but also expensive. Additionally, relying on people for this task led to mistakes and oversights. Some newer systems tried using sensors to recognize activities, but they required users to wear these sensors, limiting the ability to monitor activities in open environments. Our goal is to replace this cumbersome and restricted approach with an automated system using sensor data that doesn't require users to wear anything, making it more efficient and versatile.

2.2. Disadvantages

- Time consuming
- High complexity
- Requires continuous data collection
- Can be expensive to implement
- Limited by technology constraints

2.3. Proposed System

The Proposed system operates by taking videos and images as input to identify the activities captured in them. It stands out for its speed and cost-effectiveness, utilizing deep learning to comprehend and categorize the activities being performed. This system's versatility extends to various applications, making it an ideal foundational framework. Whether integrated or expanded upon, it can adapt to a wide array of tasks. Additionally, its potential to diminish the necessity for extra staff in data entry tasks is noteworthy. Essentially, this system not only swiftly recognizes actions in videos and images but also proves to be a practical and economical solution with the capability to serve as a fundamental platform for diverse applications, thereby streamlining processes and reducing the requirement for additional personnel in data-related activities.

2.4. Advantages

- Surveillance and Security
- Healthcare monitoring
- Enhances sports training
- Improves personal safety
- Education
- Human-Computer Interaction
- Entertainment
- Gaming

2.5 Software Requirements

Operating System	:	Windows 10
Platform (Or) Software	:	Jupiter
Coding Language	:	Python IDLE
Libraries	:	TensorFlow

2.6 Hardware Requirements

- **CPU** **:** Intel Core i3
- **RAM** **:** Minimum 4 GB
- **Storage** **:** Minimum 16 GB
- **Sensors** **:** Camera

2.7 Feasibility Study

A feasibility study is simply an assessment of the practicality of a proposed project plan or method. This is done by analyzing technical, economic, legal, operational and time feasibility factors.

2.7.1 Technical Feasibility

Using CNN for Human Activity Recognition is technically feasible. We collect sensor data from wearables, train a CNN to automatically learn patterns, and test its accuracy. The CNN's ability to recognize activities in real-world scenarios showcases its practicality, making it a viable technology for automated and accurate human activity recognition. CNNs to capture intricate spatial and temporal patterns, making

them a viable and effective solution for real-world applications in Human Activity Recognition.

2.7.2 Economic Feasibility

Implementing Human Activity Recognition (HAR) using Convolutional Neural Networks (CNNs) proves economically viable due to its efficiency and scalability. The CNN-based system reduces costs by automating the recognition process, eliminating the need for extensive manual labour. Moreover, the integration of CNNs enhances accuracy, reducing potential financial losses associated with misclassifications. The technology's adaptability to various activities and environments ensures a versatile and cost-effective solution applicable across different scenarios. Overall, the economic feasibility lies in the streamlined, automated framework that enhances accuracy while providing a scalable solution for human activity recognition in diverse settings, contributing to efficient resource utilization and economic value.

2.7.3 Legal Feasibility

Implementing Human Activity Recognition (HAR) using Convolutional Neural Networks (CNN) must align with legal frameworks to ensure privacy and compliance. Respect for data protection laws, like GDPR or HIPAA, is paramount. Consent for data collection, especially if it involves personal information, is necessary. Transparent policies on data usage, storage, and sharing must be established. Additionally, measures to anonymize or aggregate data should be employed to safeguard individuals' identities. Regular audits and updates to comply with evolving legal standards will ensure the responsible and legal feasibility of the CNN-based HAR system, fostering trust and adherence to ethical guidelines.

2.7.4 Operational Feasibility

Implementing human activity recognition with CNNs is operationally feasible as it involves practical and achievable steps. Firstly, data collection from wearable devices is straightforward. The CNN architecture's ability to automatically learn patterns removes the need for complex manual setups. Training the model with labelled data is accessible, and subsequent testing ensures reliability. The spatial scanning capability of CNN layers simplifies pattern recognition. The system's adaptability and accuracy in recognizing diverse activities make it operationally effective for real-world applications, showcasing its practicality and potential for use in everyday scenarios.

2.7.5 Social Feasibility

Implementing Human Activity Recognition (HAR) with Convolutional Neural Networks (CNNs) is socially feasible as it offers practical benefits for individuals and communities. The system primarily relies on wearable devices, ensuring user-friendly integration into daily life without intrusive measures. This approach aligns with the growing trend of health-conscious lifestyles and fitness monitoring. Moreover, HAR using CNNs has the potential to enhance safety in various settings, such as healthcare, by detecting anomalies in activities.

The technology's adaptability to diverse human activities fosters inclusivity, making it applicable to a wide range of individuals and communities. With a focus on common daily tasks, it provides a non-invasive means to monitor well-being. Additionally, the automation of activity recognition contributes to the development of assistive technologies, catering to the needs of individuals with varying abilities. Overall, the social feasibility of HAR using CNNs lies in its user-friendly nature, potential for widespread application, and positive impact on health and safety.

CHAPTER 3

SYSTEM DESIGN

3.1 Block Diagram

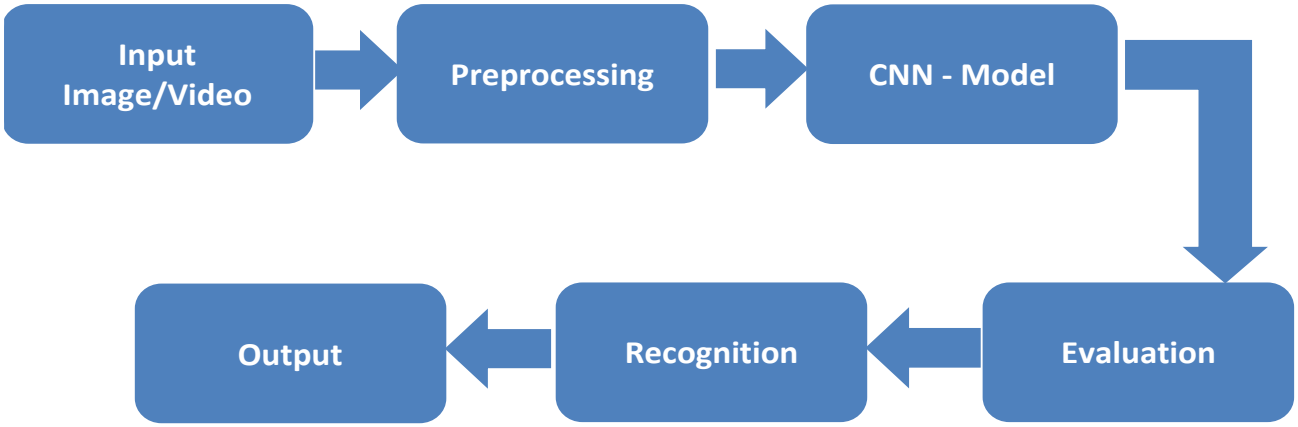


Fig.3.1 Block Diagram

.

3.2. Module Description

- Data Collection
- Data Pre-processing
- Convolutional Neural Network Architecture
- Training
- Evaluation
- Recognition
- Output

Data Input:

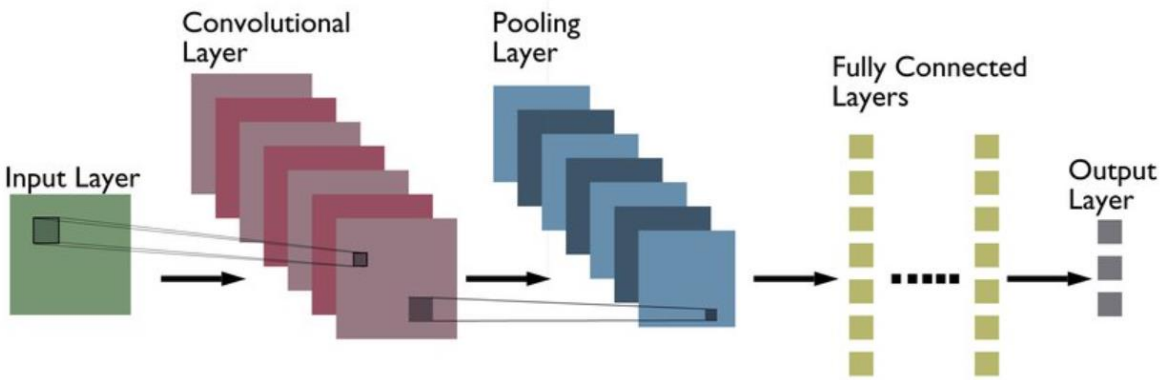
The Data Collection Module is a vital component in our Human Activity Recognition (HAR) system using Convolutional Neural Networks (CNNs). This module is responsible for assembling a sequence of images that act as input data for recognizing human activities. Imagine it as talking to people or using special sensors to collect information. The module captures images in chronological order from diverse sources, such as cameras. It's akin to creating a visual timeline of activities. This curated dataset becomes the training ground for our CNN, allowing it to learn and understand the patterns associated with various human movements. So, in essence, the Data Collection Module acts as the storyteller, providing the necessary visual elements for our HAR system to make sense of human actions.

Data Pre-processing:

The Data Processing Module in Human Activity Recognition (HAR) using Convolutional Neural Networks (CNN) is like the brain that analyses and understands the collected information. Once we've gathered sequential images, this module takes charge. It sorts through the images, making sense of the details. Think of it as a super-smart organizer arranging a jumble of pictures into a clear story.

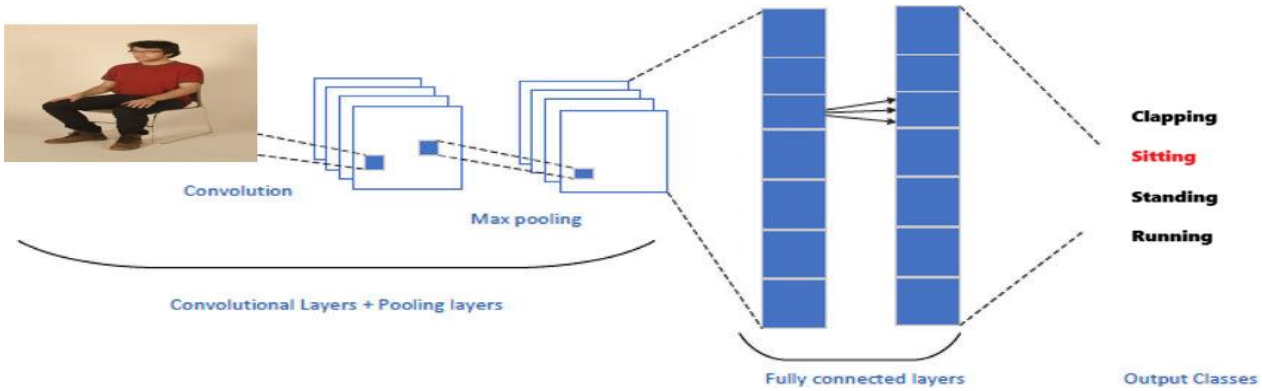
The module checks for patterns and important features in the images, like recognizing different poses or actions. It's like a detective spotting clues! By doing this, it helps the CNN understand the complexities of human activities. This smart processing step is crucial, as it transforms raw images into valuable insights, enabling the system to accurately recognize and classify various activities with precision. In simple terms, it's the brainpower that makes our HAR system sharp and reliable.

CNN Model:



The heart of our system lies in this essential module, tasked with crafting the Convolutional Neural Network (CNN) architecture. Think of it as the brain behind the scenes, orchestrating the process of understanding and recognizing human activities. Picture it like a series of filters in your camera: the convolutional layers focus on picking out unique features from the images, like recognizing shapes or movements. Then, the pooling layers step in to shrink things down, simplifying the gathered information. Finally, the fully connected layers act like the decision-makers, putting all the puzzle pieces together to classify what's happening.

In simpler terms, this module is like the blueprint of our CNN, mapping out how it should look and function. It's like giving the system a set of instructions – telling it to pay attention to certain details, ignore others, and ultimately learn the patterns that define different human actions from the images it receives.



Training:

In the training module, we're basically teaching our computer brain, the Convolutional Neural Network (CNN), to recognize and understand human activities. Imagine it's like showing the computer lots of pictures of people doing different things, like sitting, walking, or bending. We label these pictures so that the computer knows what each activity is. The process is like teaching a clever pet: we want the computer to learn patterns and connections between the pictures and the activities they show.

To make our computer even smarter, we use a method called backpropagation during training. It's like a feedback system that helps the computer adjust its 'thinking' process. The computer tweaks its internal settings, called weights and biases, to get better at guessing the right activity. The goal is to minimize the difference between what the computer predicts and the actual activities in the labeled pictures. So, through this training process, our computer gets better at recognizing human actions accurately.

Evaluation:

Now that we've trained our computer brain (CNN) to recognize human activities, it's time to check how well it's doing. We use a separate set of pictures that the computer has never seen before; think of it like a new set of test questions. This helps us know if our computer has really learned or if it's just memorizing stuff.

We then use some measures to see how good our computer is. One is accuracy, which is like checking how many questions it gets right. Another is precision, which is about how exact it is in its answers. Recall is like checking if the computer remembers all the important things correctly. These measures help us understand how effective our computer brain is in recognizing different activities, making sure it's not just good with what it learned but can also handle new situations.

Recognition:

Now, after our computer brain, the Convolutional Neural Network (CNN), has learned a lot about human activities, we move to the recognition part. This is like testing what the computer has learned. In the recognition module, we give the computer new pictures it has never seen before, and we want it to tell us what activity is happening in each picture. It's like asking our clever pet to identify actions in a new game.

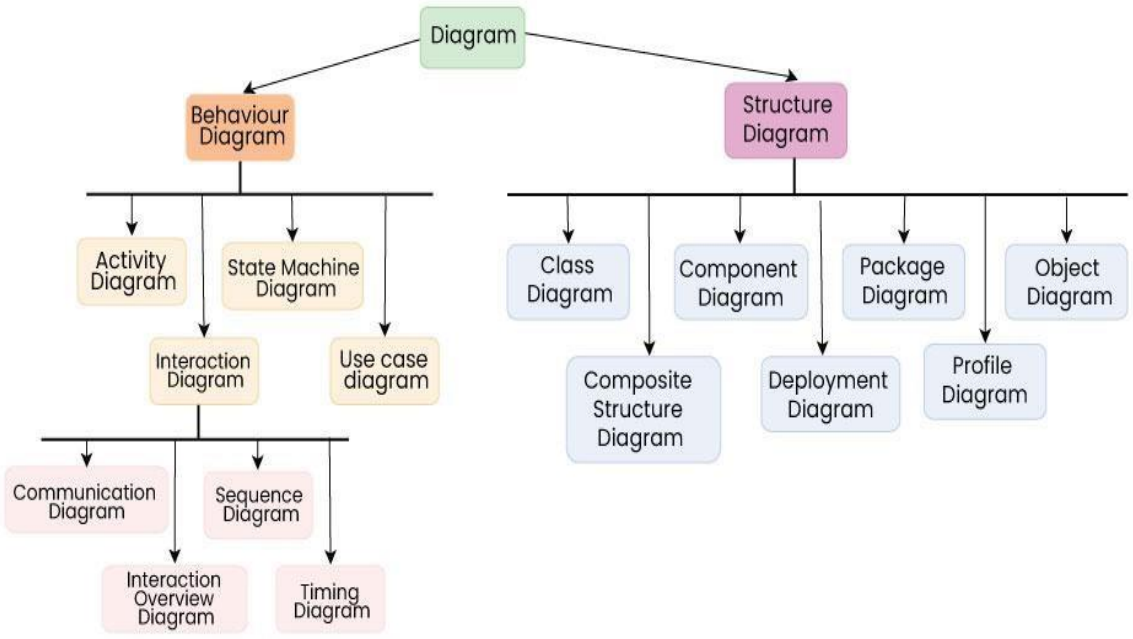
The recognition module takes all the knowledge the computer gained during training and uses it to make predictions about the activities in the new pictures. The computer translates what it learned into practical guesses, telling us if someone is sitting, standing, or doing something else in the new images. It's like putting the computer's learning to work, helping us understand and categorize activities in real-life situations based on the pictures we provide.

Result:

The result module in our Human Activity Recognition system using CNN is like the messenger that shares the computer's findings with you. After the computer learns from lots of pictures and becomes good at recognizing your actions, the result module reveals its conclusions. It might show on your screen or tell you directly – like a friendly assistant saying, "You're walking!" It's the final step where you get the accurate report on your activities, making sure you and the computer are on the same page. This module acts as your personal announcer, providing clear and understandable feedback about what the computer saw you doing.

3.3. UML Diagram

Unified Modelling Language (UML) serves as a universal language for creating models. Think of it like a set of blueprints for understanding and designing systems, just like architects use plans for building structures. Unlike programming languages, UML is more like a visual tool. Through UML diagrams, we paint pictures that show how a system behaves and its structure. It's a handy tool for software experts, business folks, and system architects to model, design, and analyse different aspects. UML is closely tied to object-oriented design, using elements and connections to craft diagrams that help us grasp and communicate complex system designs.



- Class Diagram
- Activity Diagram
- Sequence Diagram
- Use case diagram

3.3.1. Class Diagram:

Class diagrams are one of the most widely used diagrams. It is the backbone of all the object-oriented software systems. It depicts the static structure of the system. It displays the system's class, attributes, and methods. It is helpful in recognizing the relation between different objects as well as classes.

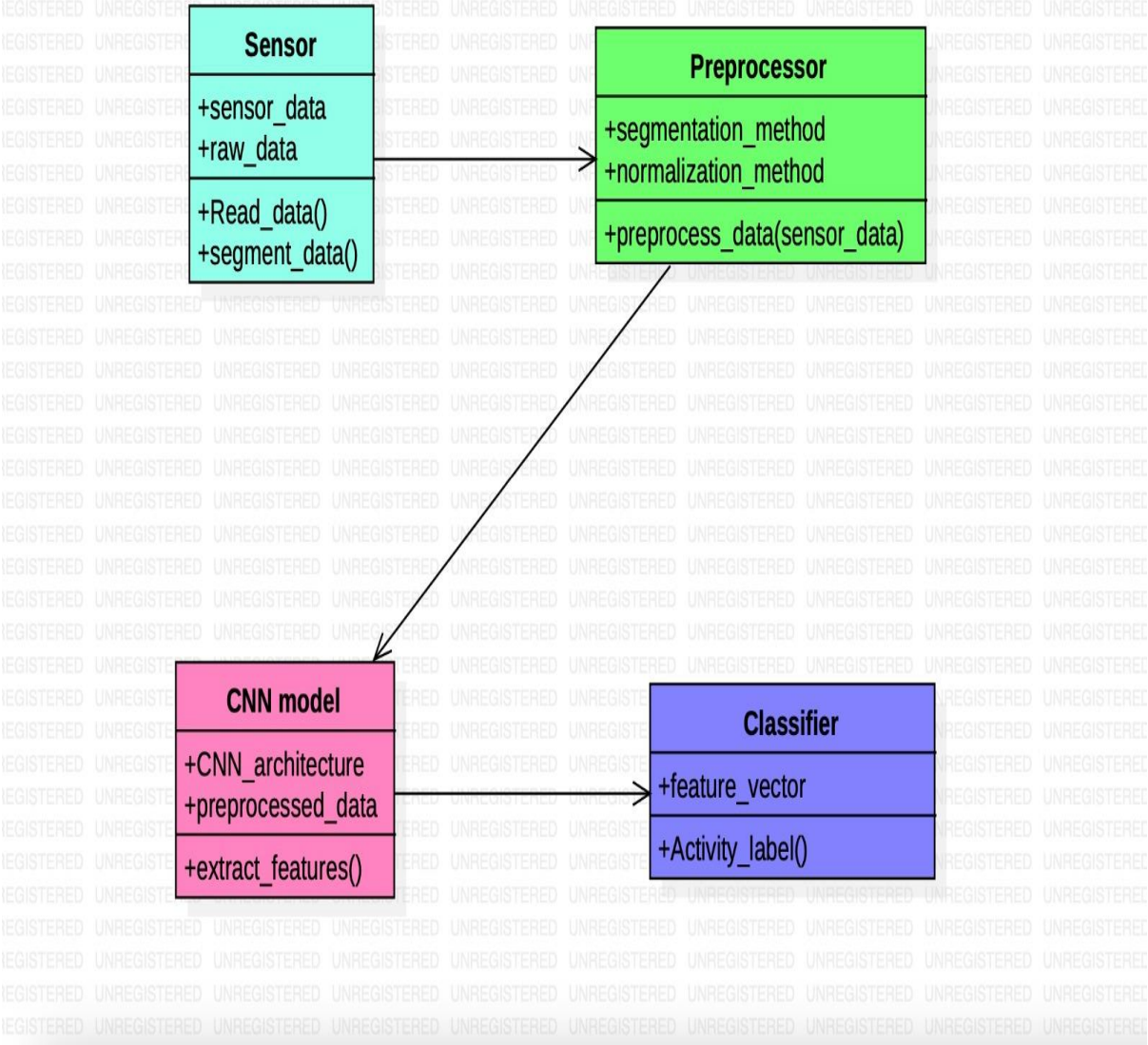


Fig.3.3.1. Class Diagram

3.3.2. Activity Diagram:

It models the flow of control from one activity to the other. With the help of an activity diagram, we can model sequential and concurrent activities. It visually depicts the workflow as well as what causes an event to occur.

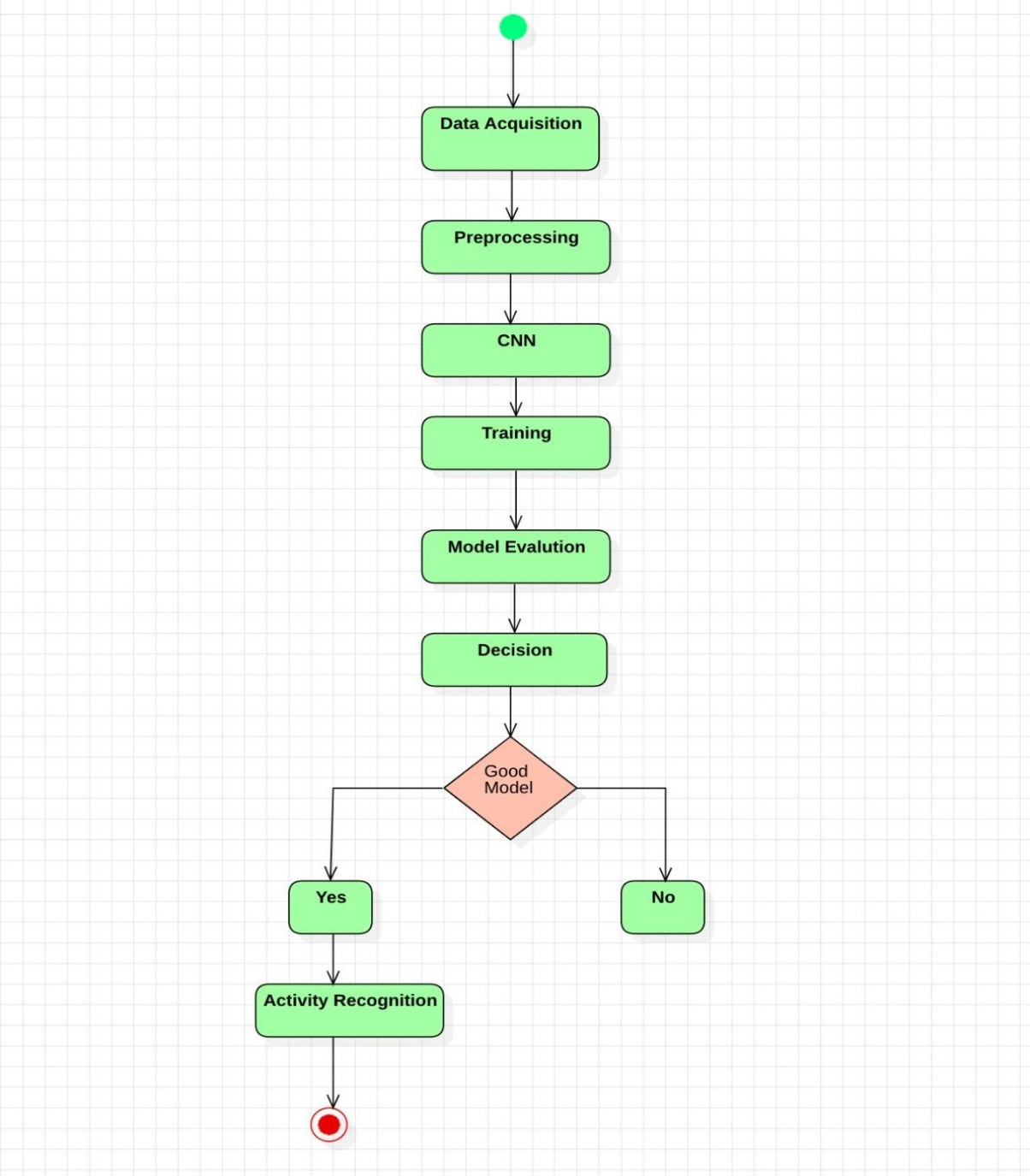


Fig.3.3.2. Activity Diagram

3.3.3. Sequence Diagram:

Sequence Diagram shows the interactions between the objects in terms of messages exchanged over time. It delineates in what order and how the object functions are in a system.

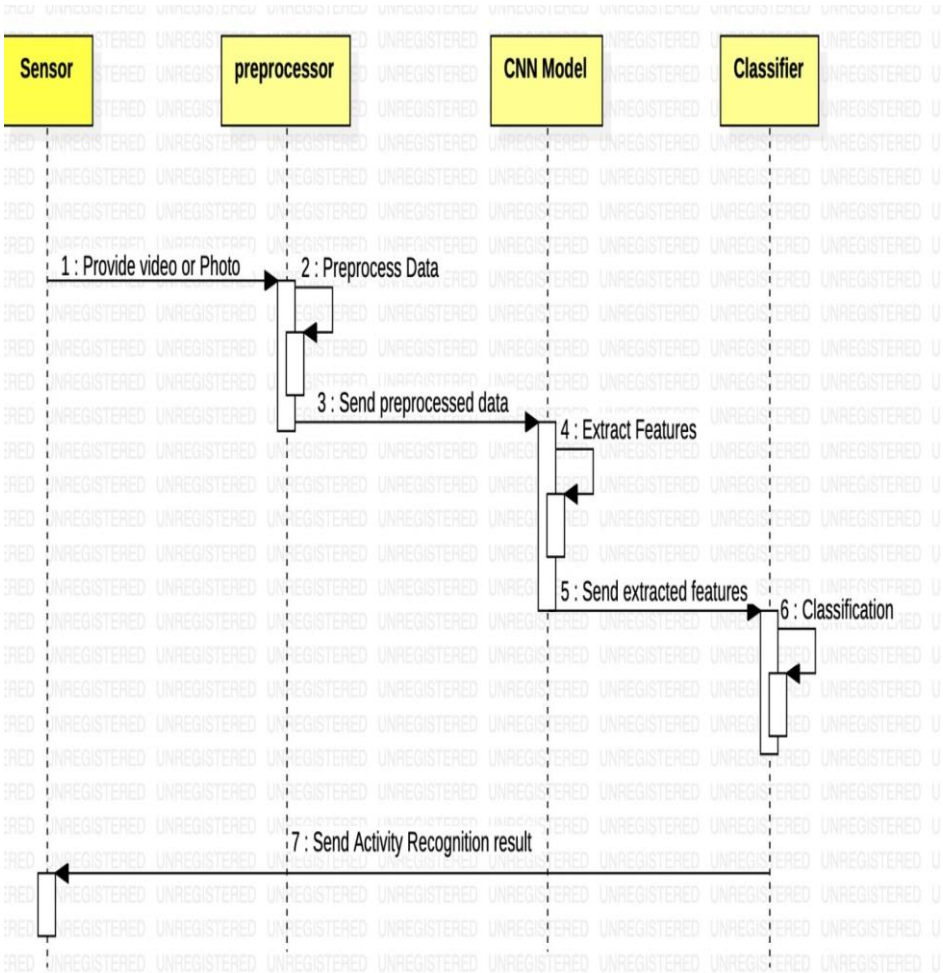


Fig.3.3.3. Sequence Diagram

3.3.4. Use Case Diagram:

A use case diagram is a visual representation in software engineering that illustrates how users interact with a system. It outlines various scenarios or use cases, depicting the relationships between actors (users) and the system, highlighting the system's functionalities. It helps in understanding, clarifying, and communicating the intended functionality of a system from a user's perspective.

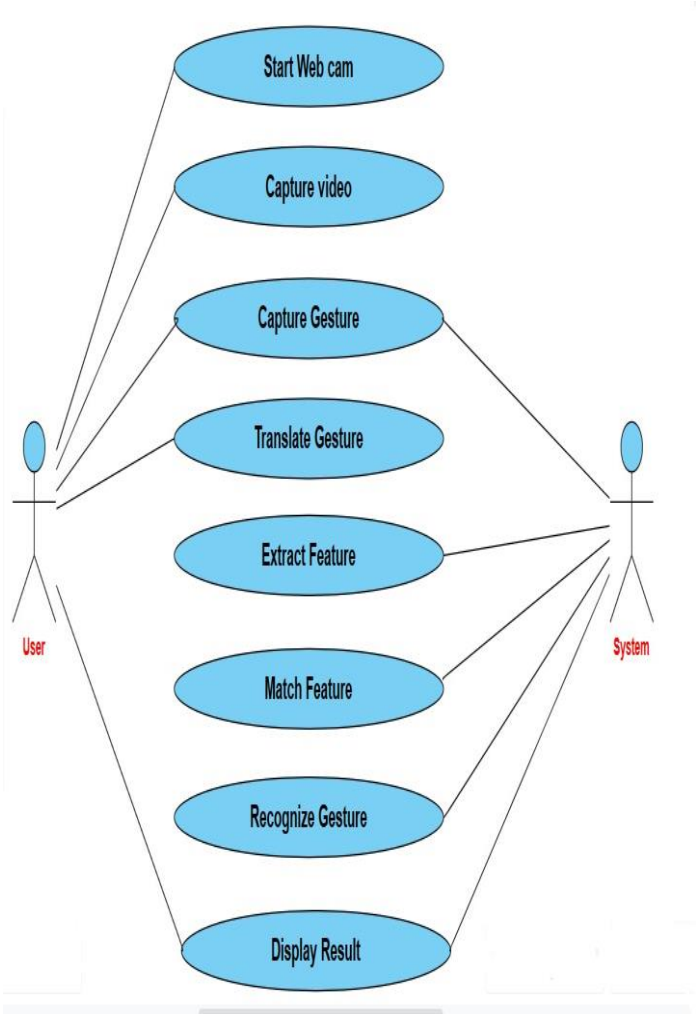


Fig.3.3.4. Use Case Diagram

3.3.5. Data Flow Diagram:

The flow of data of a system or a process is represented by DFD. DFD also gives insight into the inputs and outputs of each entity and the process itself. *DFD does not have control flow and no loops or decision* rules are present. Specific operations depending on the type of data can be explained by a flowchart. It is a graphical tool, useful for communicating with users ,managers and other personnel. it is useful for analyzing existing as well as proposed system.

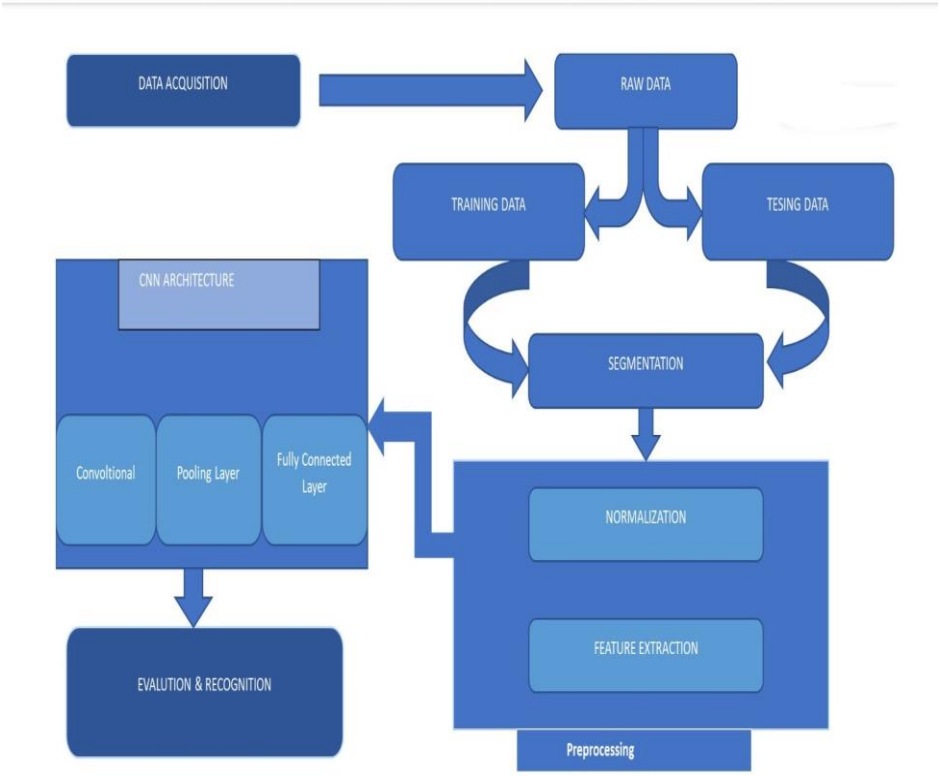


Fig.3.3.5. Data Flow Diagram

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1. Language Selection:

Python is a preferred choice for social media data analysis through data mining due to its versatility, extensive libraries, and ease of use. Python offers powerful tools like Pandas, NumPy, and Scikit-learn that facilitate data manipulation, analysis, and machine learning tasks. Its simplicity and readability make it accessible for both beginners and experienced developers.

Python is the preferred language for human activity recognition using Convolutional Neural Networks (CNNs) due to its simplicity, versatility, and extensive ecosystem. Its readability streamlines complex CNN architecture development, while libraries like Tensor Flow and OpenCV enhance efficiency. Python's rich set of data tools facilitates diverse dataset handling and seamless integration with machine learning workflows, making it an ideal choice for researchers and practitioners.

The language's popularity and community support also contribute to readily available resources and advancements in the field of human activity recognition, solidifying Python's position as the best-suited language for CNN-based applications in this domain.

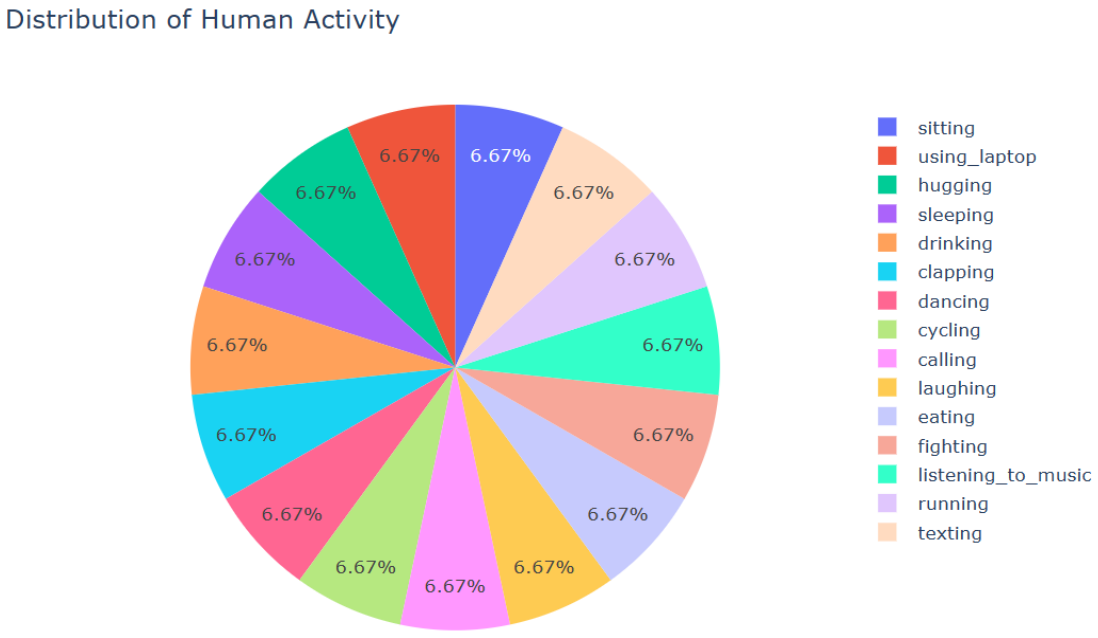
3.2. Algorithm:

1) Import necessary libraries and modules.
2) Read training and testing data from CSV files.
3) Load the paths of training and testing image folders.
4) Display the distribution of human activity labels using a pie chart.
5) Define a function (`displaying_random_images()`) to display a random image along with its label from the training set.

6) Preprocess the data:
    a. Create empty lists (`img_data` and `img_label`) to store image data and labels.
7) - Loop through each training folder, open each image, resize it to (160,160), and append the Image data to `img_data`.
    a. Append the corresponding label to `img_label`.
    b. Convert `img_data` to a NumPy array and store it in `iii`.
    c. Convert labels to categorical using `to_categorical`.
8) Define the shape of the images (`img_shape`).
9) Build a CNN model using the EfficientNetB7 architecture and add Dense layers for classification.
10) Compile the model with Adam optimizer, categorical crossentropy loss, and accuracy metrics.
11) Display a summary of the model architecture.
12) Train the model using the training data (`iii`) and labels (`y_train`) for 40 epochs.
13) Plot the training loss and accuracy over epochs.
14) Define a function (`read_img`) to read and resize images.
15) Define a function (`test_predict`) to make predictions on test images and display the result.
16) Make predictions on specific test images using the trained model.
    a. Test Image 1: "Image_1001.jpg"
    b. Test Image 2: "Image_101.jpg"
    c. Test Image 3: "Image_1056.jpg"
    d. Test Image 4: "Image_1024.jpg"
17) Display Accuracy with Output Image……

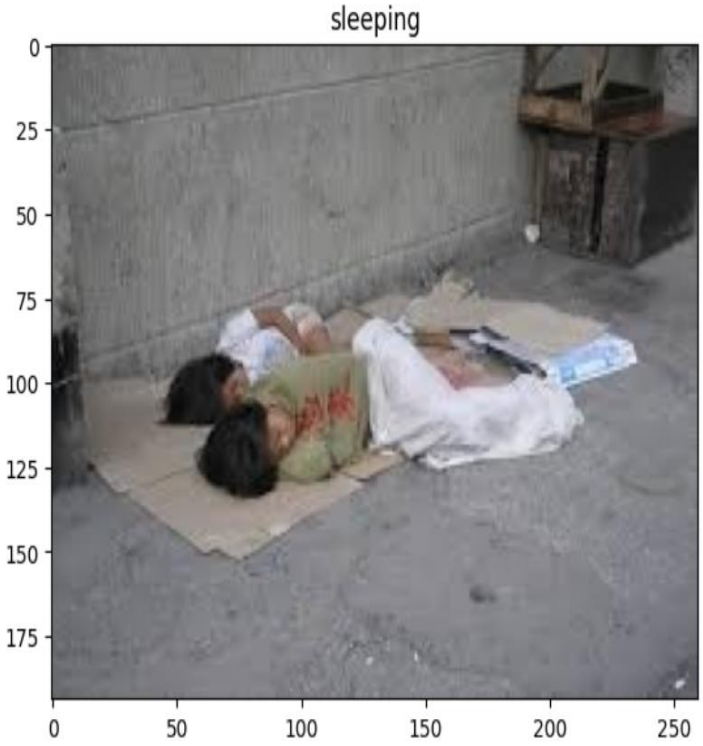
4.3. Screenshots:

```
Out[9]:
0            Image_1.jpg
1            Image_2.jpg
2            Image_3.jpg
3            Image_4.jpg
4            Image_5.jpg
                ...
12595    Image_12596.jpg
12596    Image_12597.jpg
12597    Image_12598.jpg
12598    Image_12599.jpg
12599    Image_12600.jpg
Name: filename, Length: 12600, dtype: object
```



In [11]:
```
displaying_random_images()
```


dancing
T.in '16-'17
OURITES
5:41
0
25
50
75
100
125
150
175
200
0
50
100
150
200


sleeping
0
25
50
75
100
125
150
175
0
50
100
150
200
250

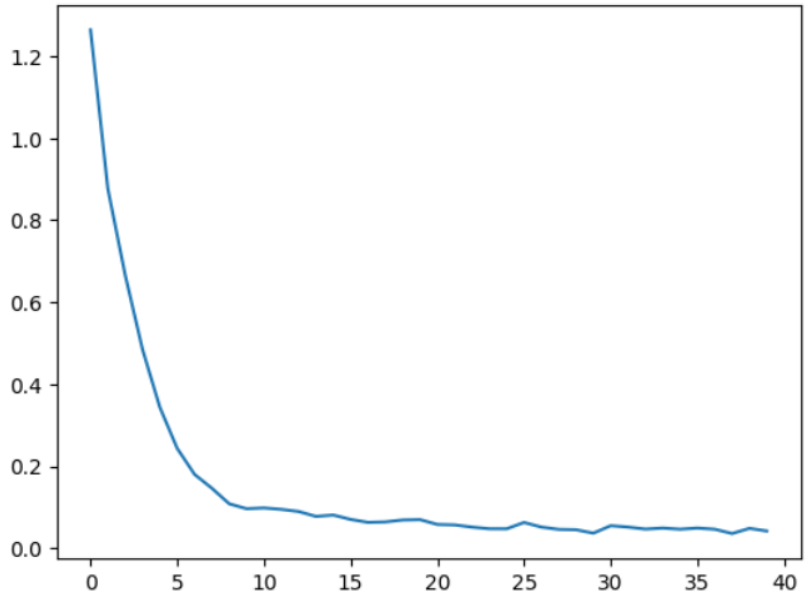
```
history = efficientnet_model.fit(iii,y_train,epochs=40)
```

```
Epoch 1/40

2024-01-11 23:18:29.280149: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed: INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin shape insequential/efficientnetb7/block1b_drop/dropout/SelectV2-2-TransposeNHWCToNCHW-LayoutOptimizer

394/394 [==============================] - 73s 123ms/step - loss: 1.2640 - accuracy: 0.6019
Epoch 2/40
394/394 [==============================] - 49s 123ms/step - loss: 0.8765 - accuracy: 0.7177
Epoch 3/40
394/394 [==============================] - 49s 123ms/step - loss: 0.6662 - accuracy: 0.7786
Epoch 4/40
394/394 [==============================] - 49s 124ms/step - loss: 0.4848 - accuracy: 0.8429
Epoch 5/40
394/394 [==============================] - 49s 123ms/step - loss: 0.3436 - accuracy: 0.8919
Epoch 6/40
394/394 [==============================] - 49s 124ms/step - loss: 0.2435 - accuracy: 0.9241
Epoch 7/40
394/394 [==============================] - 49s 123ms/step - loss: 0.1802 - accuracy: 0.9433
Epoch 8/40
394/394 [==============================] - 49s 123ms/step - loss: 0.1464 - accuracy: 0.9557
Epoch 9/40
394/394 [==============================] - 49s 123ms/step - loss: 0.1085 - accuracy: 0.9687
Epoch 10/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0965 - accuracy: 0.9707
Epoch 11/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0984 - accuracy: 0.9702
Epoch 12/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0949 - accuracy: 0.9706
```

```
Epoch 14/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0778 - accuracy: 0.9748
Epoch 15/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0811 - accuracy: 0.9747
Epoch 16/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0704 - accuracy: 0.9775
Epoch 17/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0633 - accuracy: 0.9783
Epoch 18/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0644 - accuracy: 0.9798
Epoch 19/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0690 - accuracy: 0.9787
Epoch 20/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0699 - accuracy: 0.9785
Epoch 21/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0582 - accuracy: 0.9810
Epoch 22/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0570 - accuracy: 0.9812
Epoch 23/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0517 - accuracy: 0.9829
Epoch 24/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0476 - accuracy: 0.9856
Epoch 25/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0476 - accuracy: 0.9841
Epoch 26/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0630 - accuracy: 0.9810
Epoch 27/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0517 - accuracy: 0.9840
Epoch 28/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0461 - accuracy: 0.9857
Epoch 29/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0453 - accuracy: 0.9855
Epoch 30/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0369 - accuracy: 0.9873
Epoch 31/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0551 - accuracy: 0.9828
394/394 [==============================] - 49s 124ms/step - loss: 0.0551 - accuracy: 0.9828
Epoch 32/40
394/394 [==============================] - 49s 124ms/step - loss: 0.0518 - accuracy: 0.9840
Epoch 33/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0469 - accuracy: 0.9844
Epoch 34/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0494 - accuracy: 0.9839
Epoch 35/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0466 - accuracy: 0.9853
Epoch 36/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0494 - accuracy: 0.9844
Epoch 37/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0465 - accuracy: 0.9870
Epoch 38/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0360 - accuracy: 0.9893
Epoch 39/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0486 - accuracy: 0.9846
Epoch 40/40
394/394 [==============================] - 49s 123ms/step - loss: 0.0422 - accuracy: 0.9858
```

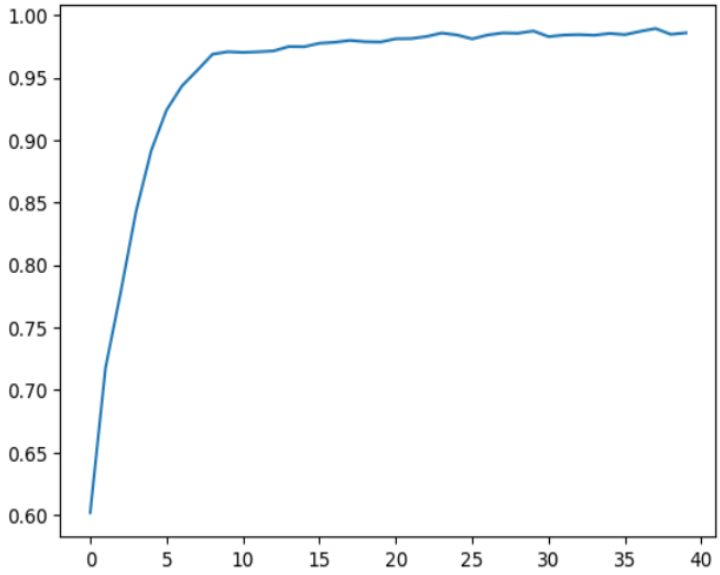
In [23]:
```
losses = history.history["loss"]
plt.plot(losses)
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x7d0427f6cd00>]
```

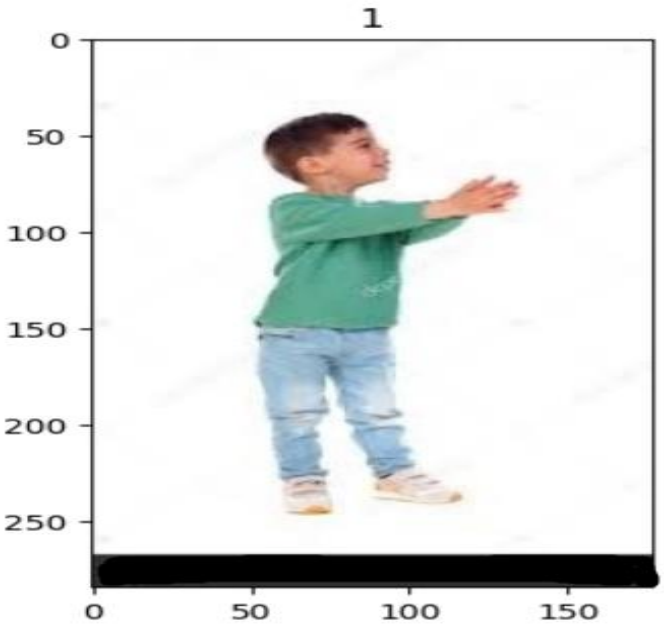


In [24]:
```
acc = history.history['accuracy']
plt.plot(acc)
```

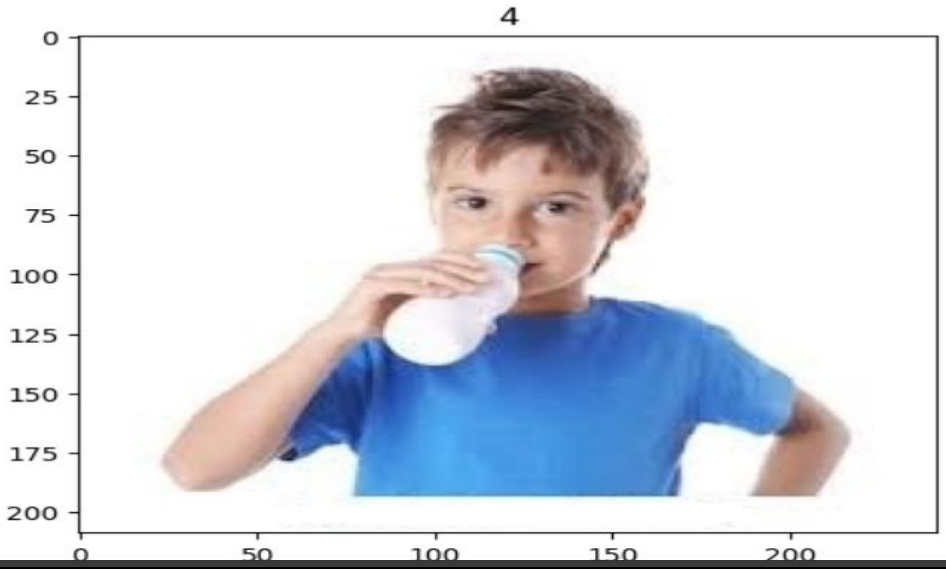
Out[24]:
```
[<matplotlib.lines.Line2D at 0x7d0427fd97b0>]
```



```
1/1 [==============================] - 6s 6s/step
probability: 99.83501434326172%
Predicted class :  1
```




```
1/1 [==============================] - 0s 40ms/step
probability: 99.9962568283081%
Predicted class :  4
```



```
1/1 [==============================] - 0s 40ms/step
probability: 98.4374463558197%
Predicted class :  5
```


```
1/1 [==============================] - 0s 40ms/step
probability: 99.85570311546326%
Predicted class :  14
```

4.4. Sample Code:

```
import os

import glob

import random

import numpy as np

import pandas as pd

import tensorflow_addons as tfa

import tensorflow as tf

from tensorflow import keras

from keras import layers

from keras.models import Sequential

from keras.layers import Conv2D,MaxPooling2D,Activation, Dropout, Flatten, Dense

from keras.preprocessing.image import ImageDataGenerator

from tqdm import tqdm

from PIL import Image

from tensorflow.keras.utils import to_categorical

import seaborn as sns

import matplotlib.image as img

import matplotlib.pyplot as plt

train_data = pd.read_csv("C:/Users/91807/OneDrive/Desktop/Training_set.csv")

test_data = pd.read_csv("C:/Users/91807/OneDrive/Desktop/Testing_set.csv")

train_fol = glob.glob("../input/human-action-recognition-har-dataset/Human Action
Recognition/train/*")
```

```
test_fol = glob.glob("../input/human-action-recognition-har-dataset/Human Action Recognition/test/*")

train_data

train_data.label.value_counts()

import plotly.express as px

HAR = train_data.label.value_counts()

fig = px.pie(train_data, values=HAR.values, names=HAR.index, title='Distribution of Human Activity')

fig.show()


#Making function that take random path and display the image

def displaying_random_images():

num = random.randint(1,10000)

imgg = "Image_{}.jpg".format(num)

   train = "../input/human-action-recognition-har-dataset/Human Action Recognition/train/"

   if os.path.exists(train+imgg):

testImage = img.imread(train+imgg)

plt.imshow(testImage)

plt.title("{}".format(train_data.loc[train_data['filename'] == "{}".format(imgg), 'label'].item()))

   else:

      #print(train+img)

      print("File Path not found \nSkipping the file!!")
```

```
displaying_random_images()

#Data preprocessing

img_data = []

img_label = []

length = len(train_fol)

for i in (range(len(train_fol)-1)):

   t = '../input/human-action-recognition-har-dataset/Human Action Recognition/train/' +
filename[i]

temp_img = Image.open(t)

img_data.append(np.asarray(temp_img.resize((160,160))))

img_label.append(situation[i])

img_shape= (160,160,3)

iii = img_data

iii = np.asarray(iii)

type(iii)

y_train = to_categorical(np.asarray(train_data["label"].factorize()[0]))

print(y_train[0])

#Make an CNN model

efficientnet_model = Sequential()

model = tf.keras.applications.EfficientNetB7(include_top=False,

input_shape=(160,160,3),

                                pooling ="avg",classes=15,

                                 weights="imagenet")
```

```
for layer in model.layers:
layer.trainable=False
efficientnet_model.add(model)
efficientnet_model.add(Flatten())
efficientnet_model.add(Dense(512,activation="relu"))
efficientnet_model.add(Dense(15,activation="softmax"))
efficientnet_model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy"])
efficientnet_model.summary()
history = efficientnet_model.fit(iii,y_train,epochs=40)
losses = history.history["loss"]
plt.plot(losses)
acc = history.history['accuracy']
plt.plot(acc)
# Model predictions
def read_img(fn):
img = Image.open(fn)
   return np.asarray(img.resize((160,160)))
def test_predict(test_image):
   result = efficientnet_model.predict(np.asarray([read_img(test_image)]))
itemindex = np.where(result==np.max(result))
   prediction = itemindex[1][0]
   print("probability: "+str(np.max(result)*100) + "%\nPredicted class : ", prediction)
   image = img.imread(test_image)
```

```
plt.imshow(image)

plt.title(prediction)

test_predict("/kaggle/input/human-action-recognition-har-dataset/Human Action Recognition/test/Image_1001.jpg")

test_predict("/kaggle/input/human-action-recognition-har-dataset/Human Action Recognition/test/Image_101.jpg")

test_predict("/kaggle/input/human-action-recognition-har-dataset/Human Action Recognition/test/Image_1056.jpg")

test_predict("/kaggle/input/human-action-recognition-har-dataset/Human Action Recognition/test/Image_1024.jpg")
```

CHAPTER 5

SYSTEM TESTING

5.1. Testing Description:

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing. In most cases, the following professionals are involved in testing a system within their respective capacities-software tester, software developer, project lead/manager and end user. A test strategy is an outline that describes the testing approach of the software development cycle. The purpose of a test strategy is to provide a rational deduction from organizational, high-level objectives to actual test activities to meet those objectives from a quality assurance perspective. The test strategies are unit testing, integration testing and validation testing.

5.1.1. Unit Testing:

Unit testing focus on the smallest unit of the ware design like module components Test strategy conducted on each module interface to access the flow of input and output. The local data structure is accessible to verify integrity during execution by using unit testing the boundary conditions are tested all error handling.

5.1.2. Integration Testing:

1. It is used for the construction of software architecture
2. there are two Approaches to integration testing

> Non-incremental integration testing

> Incremental integration testing

The testing is the systematic technique for constructing the pr structure by informing the test in each module and later combining the individual module to form a large program.

5.1.3 Validation Testing:

Validation testing is the process of testing the input. Whether the given are valid or not. When working with databases it is important to validate entries, which can be done by using scripting code. The terms verification validation are used interchangeably we will describe both these method. 5.1.4 Verification Testing Verification testing is the process of evaluating the work progress and developing the application processes. confirm that a process meets the requirements as defined in the phase of developing the application processes

5.1.4 Verification Testing:

Verification testing is the process of evaluating the work progress and developing the application processes. confirm that a process meets the requirements as defined in the phase of developing the application processes.

5.2. Test Cases

```
import os
import glob
import random
import numpy as np
import pandas as pd

import tensorflow_addons as tfa
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Activation, Dropout, Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator

from tqdm import tqdm

from PIL import Image

from tensorflow.keras.utils import to_categorical

import seaborn as sns
import matplotlib.image as img
import matplotlib.pyplot as plt
```

In [5]:
```
train_data
```

Out[5]:

	filename	label
0	Image_1.jpg	sitting
1	Image_2.jpg	using_laptop
2	Image_3.jpg	hugging
3	Image_4.jpg	sleeping
4	Image_5.jpg	using_laptop
...	...	...
12595	Image_12596.jpg	sitting
12596	Image_12597.jpg	clapping
12597	Image_12598.jpg	sitting
12598	Image_12599.jpg	dancing
12599	Image_12600.jpg	listening_to_music

12600 rows × 2 columns

In [6]:

```
train_data.label.value_counts()
```

Out[6]:

```
label
sitting              840
using_laptop         840
hugging              840
sleeping             840
drinking             840
clapping             840
dancing              840
cycling              840
calling              840
laughing             840
eating               840
fighting             840
listening_to_music   840
running              840
texting              840
Name: count, dtype: int64
```

CONCLUSION

In conclusion, Convolutional Neural Networks (CNNs) have proven to be a pivotal technology in the field of human activity recognition (HAR). The robust capabilities of CNNs in processing visual data, coupled with their ability to capture spatial and temporal features, make them well-suited for the intricate task of identifying and classifying human activities. Through the utilization of Python as a programming language, the development and implementation of CNNs for HAR have been streamlined, thanks to Python's simplicity and its extensive ecosystem of libraries.

The adaptability of CNN architectures to diverse datasets, combined with Python's support for efficient data handling and analysis, empowers researchers and practitioners to create sophisticated models capable of accurately recognizing a wide range of human activities. The integration of technologies such as TensorFlow and OpenCV further enhances the efficiency of CNNs in processing visual information.

The significance of CNNs in HAR extends beyond their technological prowess. These networks have the potential to contribute significantly to fields such as healthcare, security, and human-computer interaction. As we continue to refine and optimize CNN models for HAR, we move closer to deploying reliable and efficient systems that enhance our understanding of human behaviour and contribute to the development of intelligent and context-aware applications. The future of human activity recognition using CNNs holds promise for innovations that can positively impact various aspects of our daily lives.

SCOPE FOR FUTURE ENHANCEMENT

Multi-Modal Fusion:

Integrating data from multiple sensors, such as accelerometers, gyroscopes, and cameras, can enhance the robustness and accuracy of HAR models. Fusion techniques can combine information from different modalities to provide a more comprehensive understanding of human activities.

Real-Time Recognition:

Streamlining CNN architectures and optimizing algorithms for real-time processing is crucial for applications like video surveillance, healthcare monitoring, and human-computer interaction. Achieving low-latency recognition will open doors to a wider range of practical applications.

Incremental Learning:

Developing models that can adapt and learn incrementally over time is essential for handling dynamic and evolving human behaviours. Incremental learning ensures that the system remains effective in recognizing new activities without the need for retraining on the entire dataset.

Privacy-Preserving Techniques:

As HAR systems are often deployed in sensitive environments, incorporating privacy-preserving methods, such as federated learning or on-device processing, can address concerns related to data security and privacy.

Transfer Learning:

Leveraging pre-trained CNN models on large datasets for related tasks and fine-tuning them for specific human activity recognition tasks can expedite model training and improve performance, especially in scenarios with limited labelled data.

REFERENCES

1. S. R. Ramamurthy and N. Roy, "human activity recognition using deep learning" Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery, vol. 8, no. 4, p. e1254, Jul. 2018.
2. Y. Huang, Y. Guo, and C. Gao, "Efficient parallel inflated 3D convolution architecture for action recognition," IEEE Access, vol. 8,pp. 45753–45765, 2020.
3. Y. Sun, X. Wu, W. Yu, and F. Yu, "Action recognition with motion map 3D network," Neurocomputing, vol. 297, pp. 33–39, Jul. 2018.
4. C. Zalluhoglu and N. Ikizler-Cinbis, "Region based multi-stream convolutional neural networks for collective activity recognition," J. Vis. Commun. Image Represent., vol. 60, pp. 170–179, Apr. 2019.
5. M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori, A hierarchical deep temporal model for group activity recognition, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 19711980, Jun. 2016.
6. W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F.Viola, T.Green, T.Back, P.Natsev, M.Suleyman, and A. Zisserman, The kinetics human action video dataset, arXiv:1705.06950. [Online]. Available: http://arxiv.org/abs/1705.06950 , 2017.
7. J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, A short note about kinetics-600, arXiv:1808.01340. [Online]. Available: http://arxiv.org/abs/1808.01340, 2018.
8. C. Chen, R. Jafari, and N. Kehtarnavaz, UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor, in Proc. IEEE Int. Conf. Image Process. (ICIP), pp. 168172, Sep. 2015.
9. M. Rohrbach, A. Rohrbach, M. Regneri, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele, Recognizing ne-grained and composite activities using hand-centric features and script data, Int. J. Comput. Vis., vol. 119, no. 3, pp. 346373, Sep. 2016.
10. S. M.Amiri, M.T.Pourazad, P. Nasiopoulos, and V. C. M. Leung, Non intrusive human activity monitoring in a smart home environment, in Proc. IEEE 15th Int. Conf. e-Health Netw., Appl. Services (Healthcom), Oct. 2013.