A

Major Project

On

VISION-BASED HUMAN ACTIVITY RECOGNITION

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

B MANISH KUMAR (177R1A05J5)
P ROHIT (177R1A0599)
V SRI RAM REDDY (167R1A05P7)
V SAI CHANDANA (177R1A0559)

Under the Guidance of

M MADHUSUDHAN

(ASSISTANT PROFESSOR)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGC Act.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify **"VISION-BASED HUMAN ACTIVITY RECOGNITION"** that the project entitled being submitted by **B MANISH KUMAR (177R1A05J5), P ROHIT (177R1A0599), V SRI RAM REDDY (167R1A05P7) & V SAI CHANDANA (177R1A0559)** in partial fulfillment of the requirements for the award of the degree of BTech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of Bonafede work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. M. Madhusudhan
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
Head of the department

EXTERNAL EXAMINER

Submitted or voice Examination held on _____

ACKNOWLEGDEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. M. Madhusudhan**, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project the work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. A. Uday Kiran, Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mrs. G. Latha, Mr. A. Kiran Kumar,** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all of the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

B MANISH KUMAR (177R1A05J5)

P ROHIT (177R1A0599)

V SRI RAM REDDY (167R1A05P7)

V SAI CHANDANA (177R1A0559)

ABSTRACT

With the advent of the Internet of Things (IoT), there have been significant advancements in the area of human activity recognition (HAR) in recent years. HAR is applicable to wider application such as elderly care, anomalous behaviour detection and surveillance system. Several machine learning algorithms have been employed to predict the activities performed by the human in an environment. However, traditional machine learning approaches have been outperformed by feature engineering methods which can select an optimal set of features. On the contrary, it is known that deep learning models such as Convolutional Neural Networks (CNN) can extract features and reduce the computational cost automatically. In this paper, we use CNN model to predict human activities from Image Data set model. Specifically, we employ transfer learning to get deep image features and trained machine learning classifiers. Our experimental results showed the accuracy of 96.95% using VGG-16. Our experimental results also confirmed the high performance of VGG-16 as compared to rest of the applied CNN models.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

Human activity recognition (HAR) is an active research area because of its applications in elderly care, automated homes and surveillance system. Several studies has been done on human activity recognition in the past. Some of the existing work are either wearable based or non-wearable based . Wearable based HAR system make use of wearable sensors that are attached on the human body. Wearable based HAR system are intrusive in nature. Non-wearable based HAR system do not require any sensors to attach on the human or to carry any device for activity recognition. Non-wearable based approach can be further categorized into sensor based and vision-based HAR systems. Sensor based technology use RF signals from sensors, such as RFID, PIR sensors and Wi- Fi signals to detect human activities. Vision based technology use videos, image frames from depth cameras or IR cameras to classify human activities. Sensor based HAR system are non-intrusive in nature but may not provide high accuracy. Therefore, vision-based human activity recognition system has gained significant interest in the present time. Recognizing human activities from the streaming video is challenging. Video-based human activity recognition can be categorized as marker-based and vision-based according to motion features . Marker-based method make use of optic wearable markerbased motion capture (MoCap) framework. It can accurately capture complex human motions but this approach has some disadvantages. It require the optical sensors to be attached on the human and also demand the need of multiple camera settings. Whereas, the vision based method make use of RGB or depth image. It does not require the user to carry any devices or to attach any sensors on the human. Therefore, this methodology is getting more consideration nowadays, consequently making the HAR framework simple and easy to be deployed in many applications.

Most of the vision-based HAR systems proposed in the literature used traditional machine learning algorithms for activity recognition. However, traditional machine learning methods have been outperformed by deep learning methods in recent time. The most common type of deep learning method is Convolutional Neural Network (CNN). CNN are largely applied in areas related to computer vision. It consists series of convolution layers through which images are passed for processing. In this paper, we use CNN to recognize human activities from Weizmann Data set. We first extracted the frames for each activities from the videos. Specifically, we use transfer learning to get deep image features and trained machine learning classifiers. We applied

3 different CNN models to classify activities and compared our results with the existing works on the same data set.

In summary, the main contributions of our work are as follows:

1) We applied three different CNN models to classify human recognition activities and we showed the accuracy of 96.95% using VGG-16.

2) We used transfer learning to leverage the knowledge gained from large-scale data set such as ImageNet to the human activity recognition data set.

There have been a lot of research h on vision-based human activity recognition in recent years . Most of the studied methods have depend on handcrafted feature extraction from the videos/images and employed traditional classifiers for activity recognition. The traditional approaches often achieved optimum results and exhibited high performances. However, traditional methods are not feasible to deploy in real life because handcrafted features are highly dependent on data and are not robust to the environment change.

Hidden Markov Model (HMMs) methods have been largely used as the recognition techniques in the past because of its capability of temporal pattern decoding. However, researchers are more interested in using deep learning techniques because of its ability to automatically extract the features and learn deep pattern structures. Deep learning methods have clearly ruled out traditional classification methods in the domain of computer vision. Deep learning techniques have been largely employed recently in the domain of computer vision and have achieved tremendous results. There fore, video-based human activity recognition using deep learning models have gained a lot of interest in recent years. Zhuetal proposed an action classification method by adding a mixed-norm regularization function to a deep LSTM network. One of the most popular deep learning methods in frames/image processing is Convolutional Neural Network (CNN). The re have been several works that utilized 2D-CNNs that take advantages of spatial cor relation between the video frame s and combine the outputs employing different strategies Many have also used additional input such as optical flow to 2D-CNN to get temporal correlations information. Subsequently, 3D-CNNs were introduced that demonstrated exceptional results in the classification of videos and frames. Wangetal. applied CNN to RGB and depth frames to automatically extract the features. The obtained features were passed

through a fully connected neural network and achieved an improved accuracy. Jietal. proposed a 3D CNN model which perform s 3D convolutions and extract spatial and temporal features by capturing the motion information for activity recognition. Simonyanetal. introduced ConvNet, a two-stream convolution layer architecture that could achieve good results despite of limited training data. Khaireetal. proposed a model that train convents from RGB-D data set and combined the softmax scores from depth , motion and skeleton images at the classification level to identify the activities. Karpathyetal. proposed the extension of CNN architecture in the first Convolutional layers over a 40 video chunk. Similarly, Tranetal. used a deep 3D CNN architecture (quiet similar to VGGnet) that utilize spatiotemporal convolutions and pooling in all layers to improve the accuracy of the model.

In comparison, we are more interested to explore how transfer learning can be leveraged with CNN models on benchmark dataset to improve classification accuracy.

According to the Alzheimer' s Society, around 46.8M people are living with dementia and the numbers will rise to 115.4M in 2050. One in three PwD shows aggressive behavior, which is very stressful and upsetting for the person with dementia and their carers. Depression is also common at all stages of dementia. It occurs in about 20– 40% of PwD. Identifying depression in PwD can be difficult. To date, there is no single test or questionnaire to detect the depression due to the complexities and multifaceted nature of the condition. The common approach to monitor and manage the above-mentioned behavioral symptoms is via direct observation by caregivers, family members and health care professionals. However, this is labor-intensive, subjective, time consuming, costly and could increase the workload of caregivers and health care professionals. Recently, ambient technologies have been explored extensively in a variety of settings, such as " smart homes" and hospitals, for health monitoring. These technologies could be adapted into the early detection of behavioral symptoms that would aid caregivers and guide the headway of tailored interventions. Most of the above-mentioned technologies often use on body bio-sensing devices (e.g. actigraphs, accelerometer, biomarkers and bio-patches) for measuring signals linking behavioral symptoms. However, it is suggested that PwD requires monitoring systems that are "unobtrusive, and preferably collected in a transparent way without patient intervention due to their cognitive impairment " . Therefore, more recently researchers have explored monitoring systems involving unobtrusive methods that includes video surveillance using

cameras and Kinetic sensor. Monitoring and recognition of aggression and depression using such systems is still very much in its infancy. This could be due to the challenge faced by the researchers to develop standard algorithms that can adequately and concisely recognize behavioral symptoms. In this paper, we propose a novel method for recognizing behavioral symptoms involving aggression and depression. The proposed approach benefits from the power of transfer learning (TL) by using appearance features as deep CNN features, which are extracted from various state-of-the-art deep models (e.g. VGG16, Inception-V3 and Inception ResNet-V2). We also explore the various level of abstraction by exploring different extraction points in a given CNN model (e.g. VGG16). This work includes the following novel contributions:

• To our knowledge, we are the first to report vision based recognition of behavioral symptoms (aggressive, depressive, happy and neutral) in PwD.

• We demonstrate the effectiveness of TL using different state-of-the-art deep CNN models for recognizing behavioral symptoms in PwD. We evaluate various combinations of deep CNN features using SVM.

• We introduce a novel image dataset to advance video based surveillance research for behavior recognition.

It is from the well-known ITV's Emmerdale episodes involving dementia storyline.

Human action and behavior recognition has many potential applications including intelligent surveillance, assistive technologies, robotics and human-computer interaction. It is a fundamental and well-studied problem in computer vision with a long list of literature over the years. Traditional approaches often depend on the hand-crafted feature extraction and representation (e.g. HOG and SIFT), hand-object interactions , articulated pose and part-based/structured models. Many of these approaches explore the spatial configuration of body parts and hand-object interactions that often require body parts and/or object detector. Recently, major advances in CNN-based deep models have challenged these approaches. These CNN models are trained and evaluated on very large and highly diverse datasets often consisting human-human, human-objects and human-animals interactions. In contrast, the targeted behavioral symptoms are often expressed via body language (e.g. gestures) and facial expression, and usually a hard problem for a machine to differentiate varius symptoms shown by the same

person. It is also known as fine-grained recognition. Deep CNN models are comprised of multiple layers to learn representation of images/videos with multiple levels of abstractions through a hierarchical learning process. Such models learn from very general (e.g. Gabor filters, edges, color blobs) to task-specific features as we move from first-layer to the last-layer . Thus, these models are explored for TL in solving visual recognition tasks. In TL, a base network is trained on a base dataset. Then, the learned features (e.g. weights) are adapted, or transferred to a second target network/model to be trained on a target dataset. This would work if the learned features are task-independent, which means they are suitable for both base and target task. More recently, it has been shown that it is possible to obtain state-of-the-art results using TL . This suggests the layers of deep models do indeed learn features that are fairly general. In this paper, we explore strategies to strengthen this generalizability. Automatic monitoring of the behavioral symptoms is often based on wearable sensors. In, Chikhaouietal. have used Kinect and accelerometer to classify aggressive and agitated behavior using ensemble learning classifier. Whaleetal. used a smartphone app to collect context-sensitive information to monitor behavioral patterns that might be indicative of depressive symptoms.

2. LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Noninvasive sensor based automated smoking activity detection

AUTHORS: B. Bhandari, J. Lu, X. Zheng, S. Rajasegarar, and C. Karmakar

Although smoking prevalence is declining in many countries, smoking related health problems still leads the preventable causes of death in the world. Several smoking intervention mechanisms have been introduced to help smoking cessation. However, these methods are inefficient since they lack in providing real time personalized intervention messages to the smoking addicted users. To address this challenge, the first step is to build an automated smoking behavior detection system. In this study, we propose an accelerometer sensor based non-invasive and automated framework for smoking behavior detection. We built a prototype device to collect data from several participants performing smoking and other five confounding activities. We used three different classifiers to compare activity detection performance using the extracted features from accelerometer data. Our evaluation demonstrates that the proposed approach is able to classify smoking activity among the confounding activities with high accuracy. The proposed system shows the potential for developing a real time automated smoking activity detection and intervention framework.

2.2 Compressive representation for device-free activity recognition with passive rfid signal strength

AUTHORS: L. Yao, Q. Z. Sheng, X. Li, T. Gu, M. Tan, X. Wang, S. Wang, and W. Ruan

Understanding and recognizing human activities is a fundamental research topic for a wide range of important applications such as fall detection and remote health monitoring and intervention. Despite active research in human activity recognition over the past years, existing approaches based on computer vision or wearable sensor technologies present several significant issues such as privacy (e.g., using video camera to monitor the elderly at home) and practicality (e.g., not possible for an older person with dementia to remember wearing devices). In this paper, we present a low-cost, unobtrusive, and robust system that supports independent living of older people. The system interprets what a person is doing by deciphering signal fluctuations using radio-frequency identification (RFID) technology and machine learning algorithms. To deal with

noisy, streaming, and unstable RFID signals, we develop a compressive sensing, dictionary-based approach that can learn a set of compact and informative dictionaries of activities using an unsupervised subspace decomposition. In particular, we devise a number of approaches to explore the properties of sparse coefficients of the learned dictionaries for fully utilizing the embodied discriminative information on the activity recognition task. Our approach achieves efficient and robust activity recognition via a more compact and robust representation of activities. Extensive experiments conducted in a real-life residential environment demonstrate that our proposed system offers a good overall performance and shows the promising practical potential to underpin the applications for the independent living of the elderly.

2.3 Sparse Composition Of Body Poses And Atomic Actions For Human Activity Recognition In Rgb-D Videos

AUTHORS : I. Lillo, J. C. Niebles, and A. Soto

This paper presents an approach to recognize human activities using body poses estimated from RGB-D data. We focus on recognizing complex activities composed of sequential or simultaneous atomic actions characterized by body motions of a single actor. We tackle this problem by introducing a hierarchical compositional model that operates at three levels of abstraction. At the lowest level, geometric and motion descriptors are used to learn a dictionary of body poses. At the intermediate level, sparse compositions of these body poses are used to obtain meaningful representations for atomic human actions. Finally, at the highest level, spatial and temporal compositions of these atomic actions are used to represent complex human activities. Our results show the benefits of using a hierarchical model that exploits the sharing and composition of body poses into atomic actions, and atomic actions into activities. A quantitative evaluation using two benchmark datasets illustrates the advantages of our model to perform action and activity recognition.

2.4 Imagenet: A Large-Scale Hierarchical Image Database

AUTHORS : **J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei**

The explosion of image data on the Internet has the potential to foster more sophisticated and robust models and algorithms to index, retrieve, organize and interact with images and

multimedia data. But exactly how such data can be harnessed and organized remains a critical problem. We introduce here a new database called “ImageNet”, a large-scale ontology of images built upon the backbone of the WordNet structure. ImageNet aims to populate the majority of the 80,000 synsets of WordNet with an average of 500-1000 clean and full resolution images. This will result in tens of millions of annotated images organized by the semantic hierarchy of WordNet. This paper offers a detailed analysis of ImageNet in its current state: 12 subtrees with 5247 synsets and 3.2 million images in total. We show that ImageNet is much larger in scale and diversity and much more accurate than the current image datasets. Constructing such a large-scale database is a challenging task. We describe the data collection scheme with Amazon Mechanical Turk. Lastly, we illustrate the usefulness of ImageNet through three simple applications in object recognition, image classification and automatic object clustering. We hope that the scale, accuracy, diversity and hierarchical structure of ImageNet can offer unparalleled opportunities to researchers in the computer vision community and beyond.

2.5 A Vision-Based Transfer Learning Approach For Recognizing Behavioral Symptoms In People With Dementia

AUTHORS: Z. Wharton, E. Thomas, B. Debnath, and A. Behera

With an aging population that continues to grow, dementia is a major global health concern. It is a syndrome in which there is a deterioration in memory, thinking, behavior and the ability to perform activities of daily living. Depression and aggressive behavior are the most upsetting and challenging symptoms of dementia. Automatic recognition of these behaviors would not only be useful to alert family members and caregivers, but also helpful in planning and managing daily activities of people with dementia (PwD). In this work, we propose a vision-based approach that unifies transfer learning and deep Convolutional neural network (CNN) for the effective recognition of behavioral symptoms. We also compare the performance of state-of-the-art CNN features with the hand-crafted HOG-feature, as well as their combination using a basic linear SVM. The proposed method is evaluated on a newly created dataset, which is based on the dementia storyline in ITVs Emmerdale episodes. The Alzheimer's Society has described it as a "realistic portrayal" 1 of the condition to raise awareness of the issues surrounding dementia.

3. SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

In the existing work with wearable based or non-wearable based. Wearable based HAR system make use of wearable sensors that are attached on the human body. Wearable based HAR system are intrusive in nature. Non-wearable based HAR system do not require any sensors to attach on the human or to carry any device for activity recognition. Non-wearable based approach can be further categorized into sensor based HAR systems. Sensor based technology use RF signals from sensors, such as RFID, PIR sensors and Wifi signals to detect human activities. Sensor based HAR system are non-intrusive in nature but may not provide high accuracy.

DISADVANTAGES OF EXISTING SYSTEM:

- Require the optical sensors to be attached on the human and also demand the need of multiple camera settings.
- Wearable dives cost are high.
- **Algorithm**: Markerbased motion Capture (MoCap) Framework.

3.2 PROPOSED SYSTEM:

The proposed System Vision based technology use videos, image frames from depth cameras or IR cameras to classify human activities. Video-based human activity recognition can be categorized as vision-based according to motion features. The vision based method make use of RGB or depth image. It does not require the user to carry any devices or to attach any sensors on the human. Therefore, this methodology is getting more consideration nowadays, consequently making the HAR framework simple and easy to be deployed in many applications. The most common type of deep learning method is Convolutional Neural Network (CNN). CNN are largely applied in areas related to computer vision. It consists series of convolution layers through which images are passed for processing.

ADVANTAGES OF PROPOSED SYSTEM:

- We use CNN to recognize human activities action recognition kinetics dataset.
- We use transfer learning to get deep image features and trained machine learning classifiers.
- Does not require the user to carry any devices or to attach any sensors on the human

Algorithm: Convolutional Neural Networks(CNN), VGG-16 (also called OxfordNet)

4. REQUIREMENTS

4. REQUIREMENT

4.1 REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the instigation from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

4.2 REQUIREMENT SPECIFICATION

4.2.1 Functional Requirements

- Graphical User interface with the User.

4.2.2 Software Requirements

For developing the application the following are the Software Requirements:

- Python

Operating Systems supported

Windows 10 64 BIT

Technologies and Languages used to Develop

- Python

Debugger and Emulator

- Any Browser (Particularly Chrome)

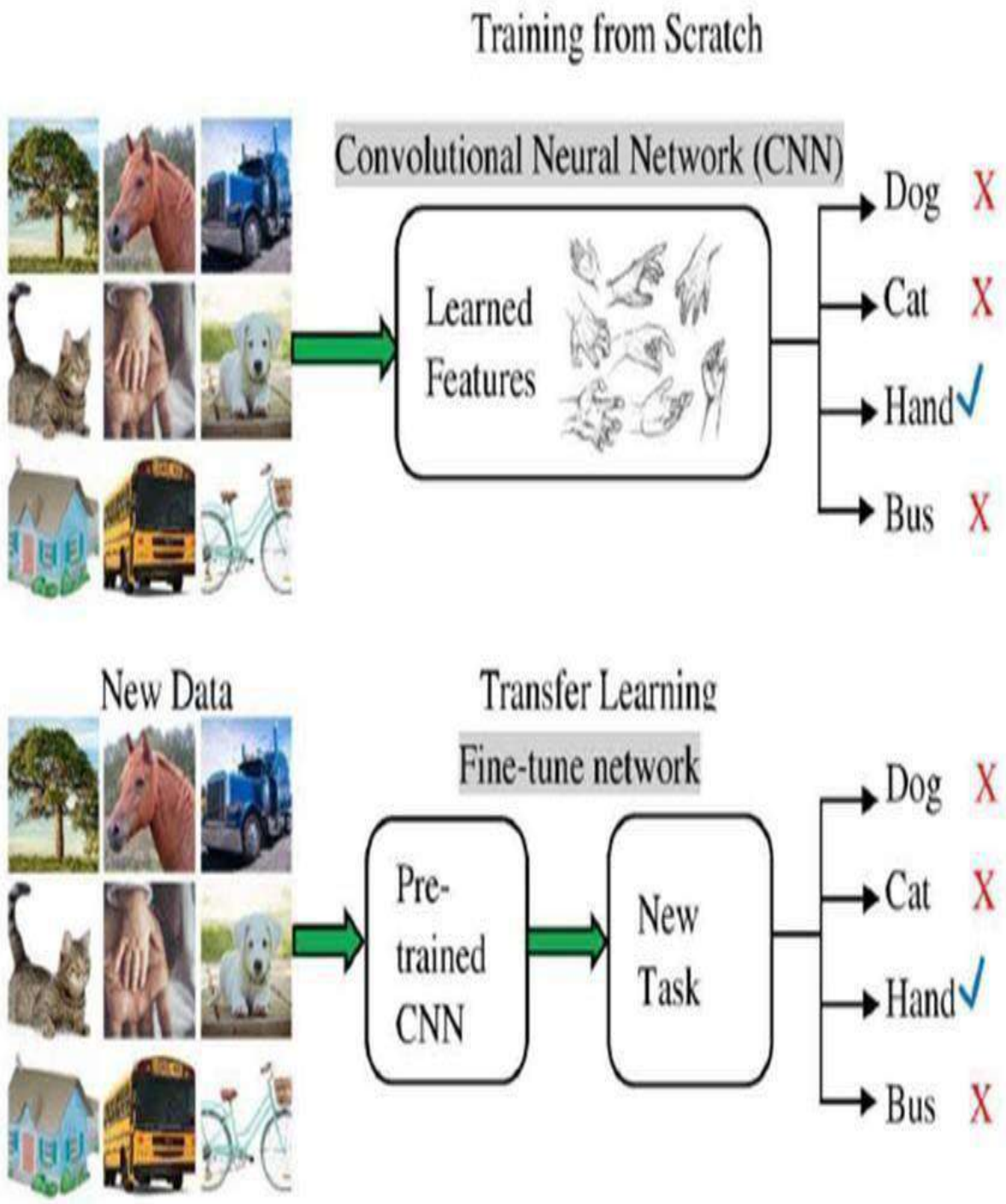
4.2.3 Hardware Requirements

- Processor: Intel i3
- RAM: 4GB
- Space on Hard Disk: Minimum 1 TB

5. SYSTEM DESIGN

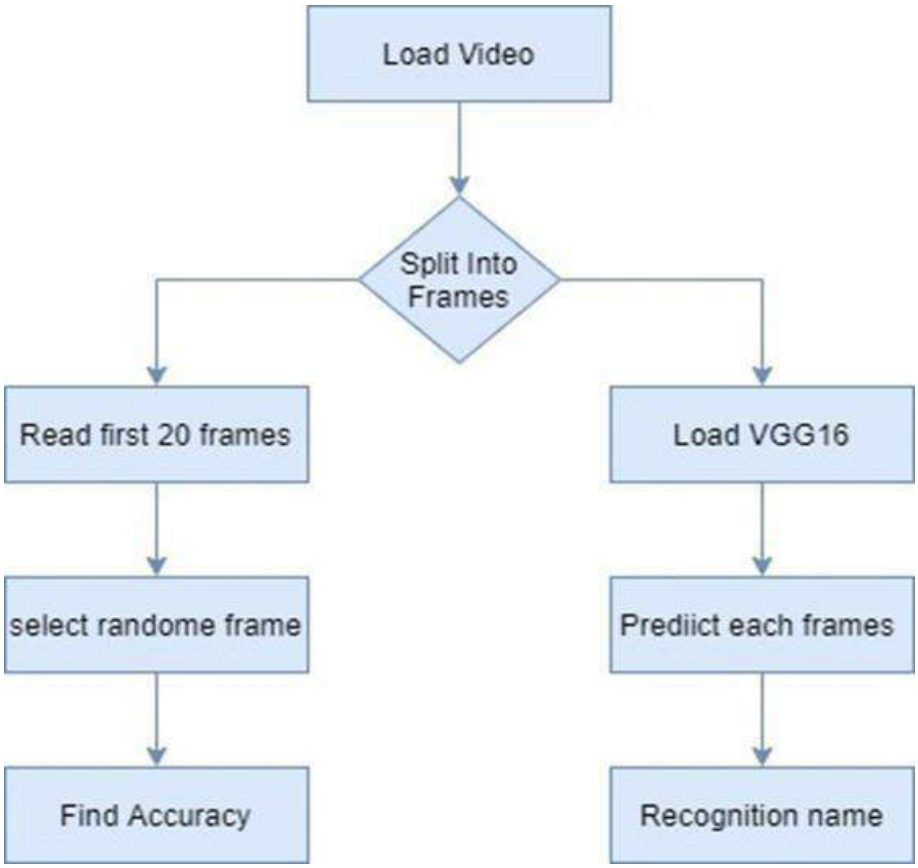
5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:



5.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



5.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

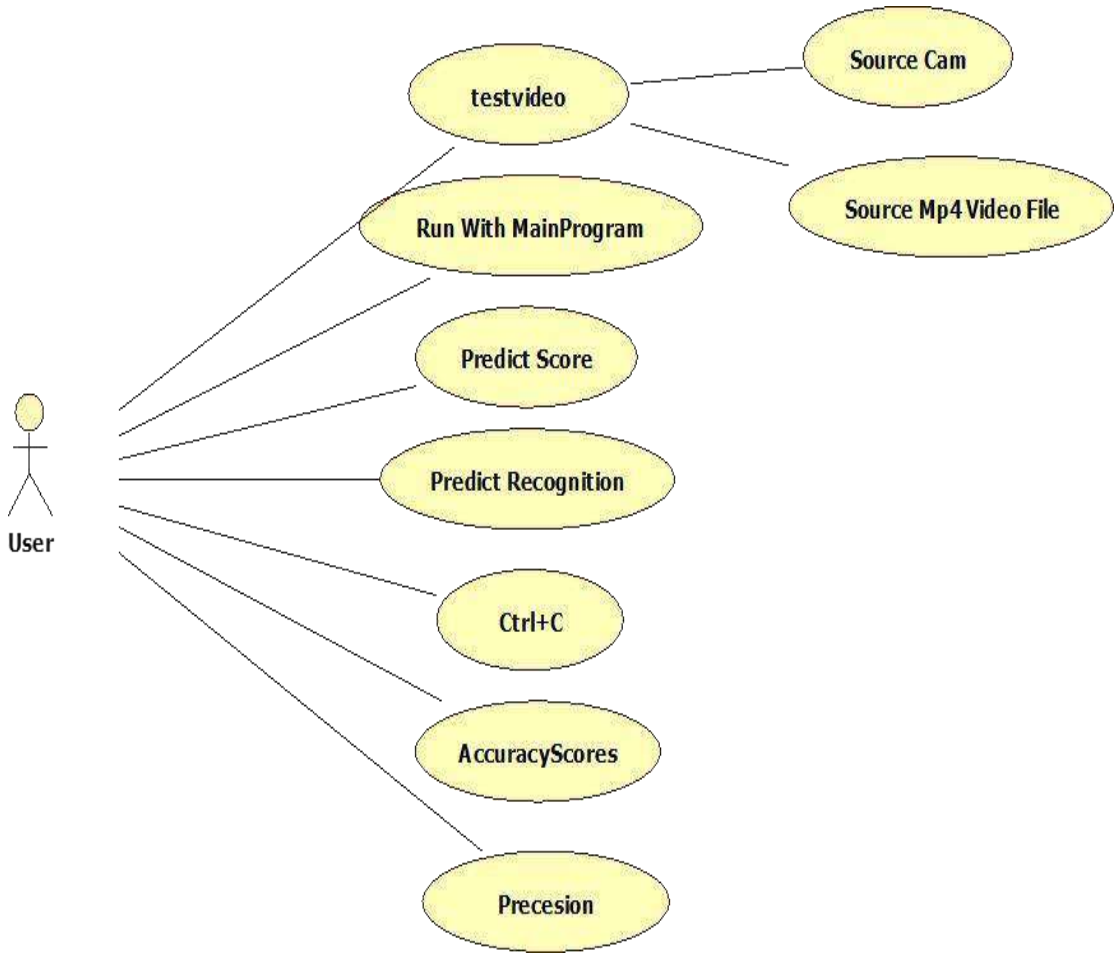
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

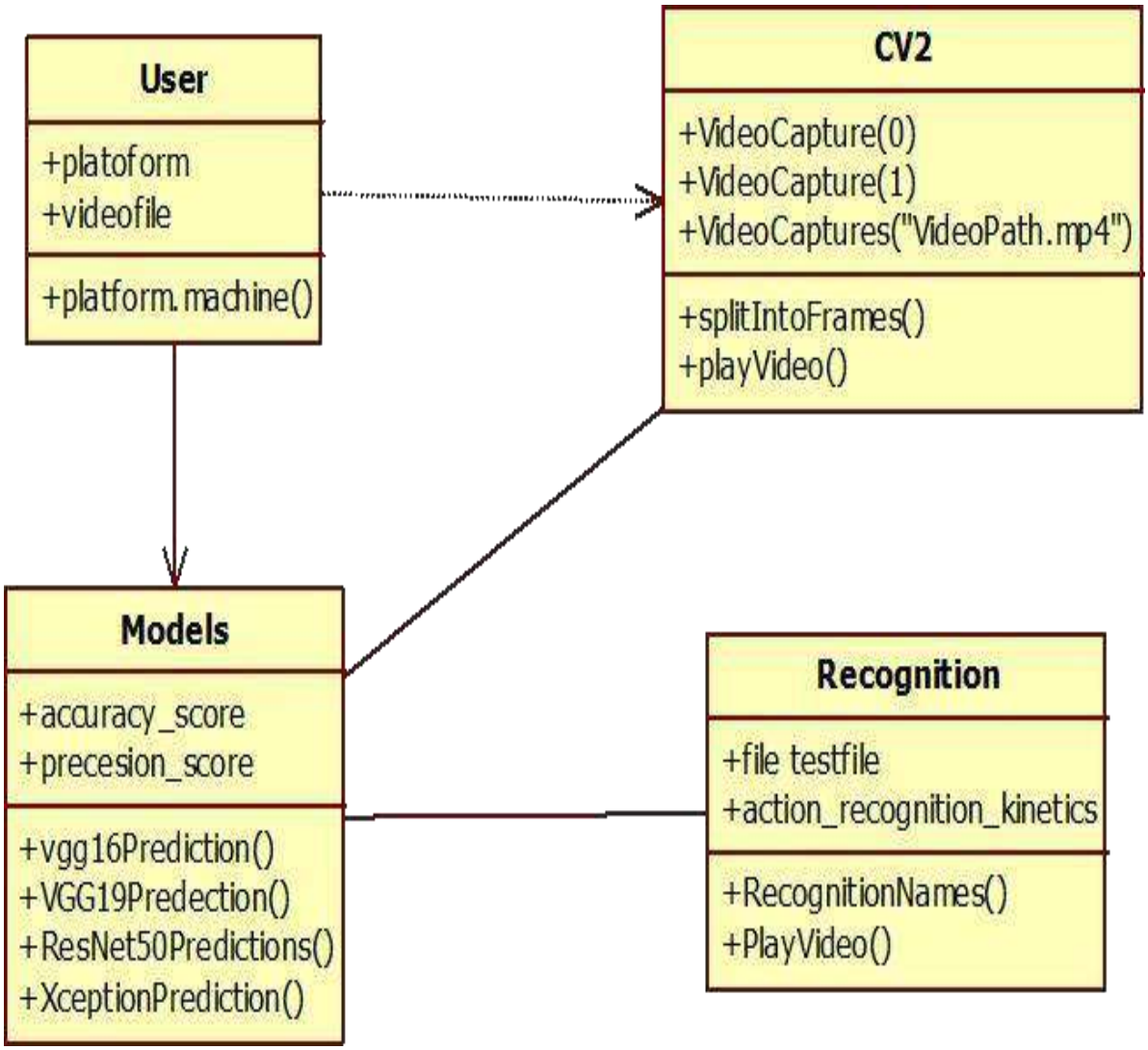
5.4 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



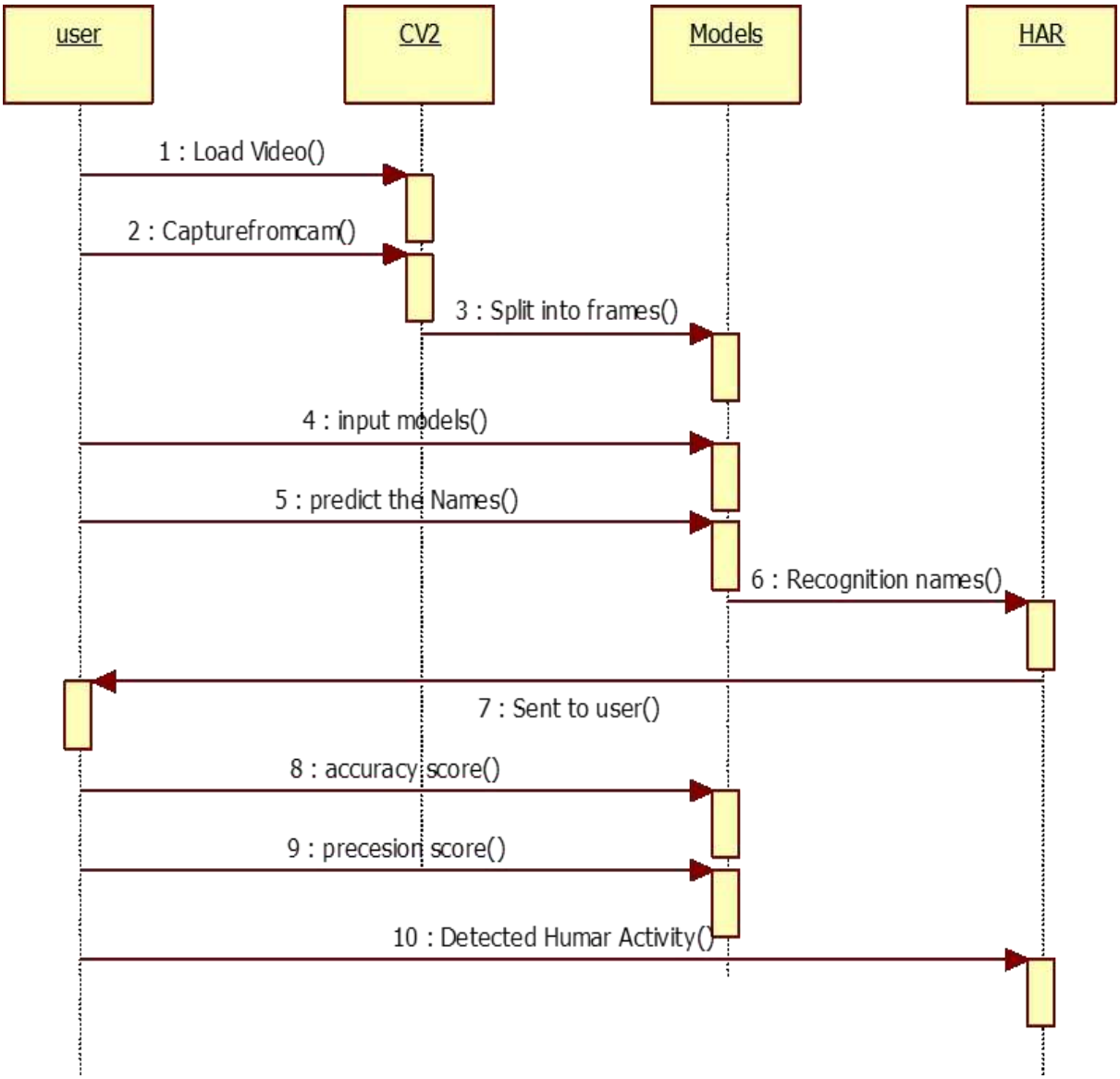
5.5 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



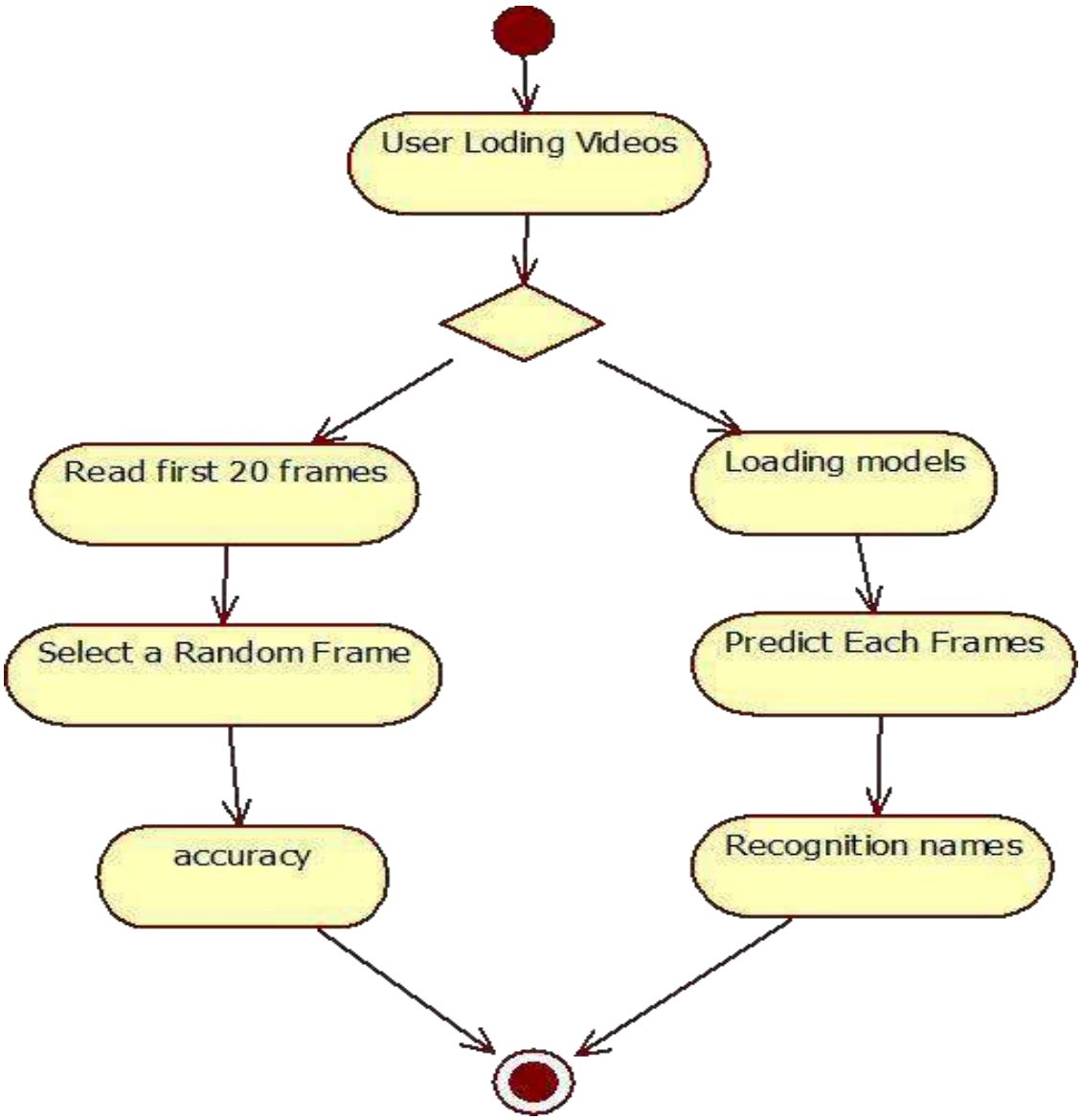
5.6 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



5.7 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



6. MODULES

6.MODULES:

6.1 User:

The User can start the project by running mainrun.py file. User has to give –input (Video file path).The open cv class VideoCapture(0) means primary camera of the system, VideoCapture(1) means secondary camera of the system. VideoCapture(Videfile path) means with out camera we can load the video file from the disk. Vgg16, Vgg19 has programitaically configured. User can change the model selection in the code and can run in multiple ways.

6.2 HAR System:

Video-based human activity recognition can be categorized as vision-based according. The vision based method make use of RGB or depth image. It does not require the user to carry any devices or to attach any sensors on the human. Therefore, this methodology is getting more consideration nowadays, consequently making the HAR framework simple and easy to be deployed in many applications. We first extracted the frames for each activities from the videos. Specifically, we use transfer learning to get deep image features and trained machine learning classifiers.

6.3 VGG16:

VGG16 is a Convolutional neural network model. Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second Convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

6.4 Transfer Learning:

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

7. IMPLEMENTATION

7. IMPLEMENTATION

7.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

7.1.1 Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

Hello, Python!

7.1.2 Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result −

```
Hello, Python!
```

- **Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.
Starting an identifier with two leading underscores indicates a strongly private identifier.
If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

- **Reserved Words**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not
assert finally or
break for pass
class from print
continue global raise
def if return
del import try
elif in while
else is with
except lambdayield

- **Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
```

```
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
import sys
try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
```

```
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline
comment.
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying "Press the enter key to exit", and waits for the user to take action −

```
#!/usr/bin/python
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   Suite
```

- **Command Line Arguments**

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d    : debug output from parser (also PYTHONDEBUG=x)
-E    : ignore environment variables (such as PYTHONPATH)
-h    : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

● Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
```

```
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]: physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']: Zara
dict['Age']: 7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']: DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
```

TypeError: 'type' object is unsubscriptable

Note − del() method is discussed in subsequent section.

● Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']: Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python
dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python
tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

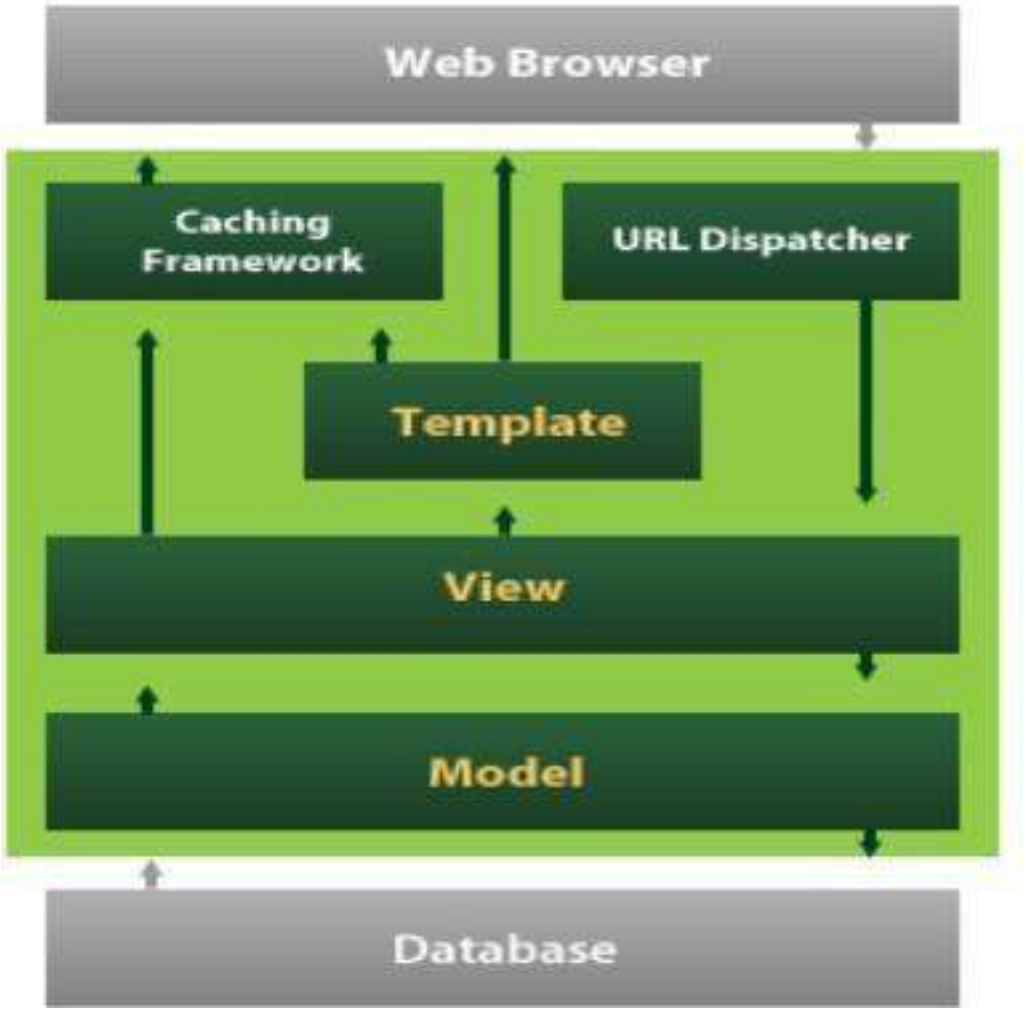
This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
```

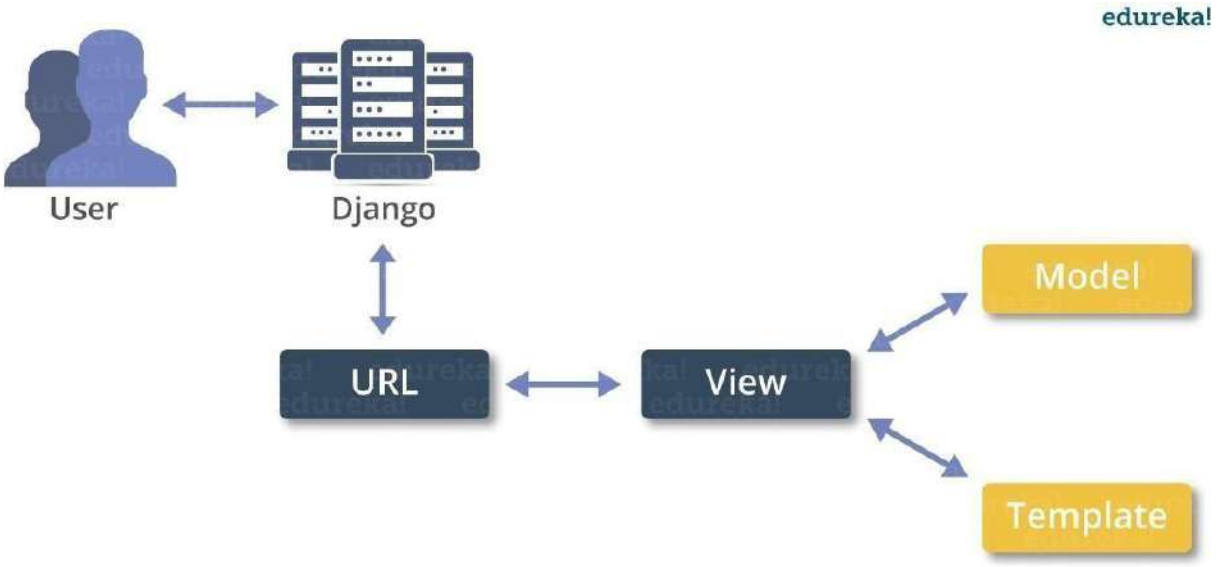
7.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, 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



7.2.1 Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure −

myproject/
  manage.py
  myproject/
    __init_.py
    settings.py
    urls.py
    wsgi.py

The Project Structure

The "myproject" folder is just your project container, it actually contains two elements −

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The "myproject" subfolder − This folder is the actual python package of your project. It contains four files −

__init_.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to 'True' for a live project. However, this has to be set to 'True' if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database.sql',
    'USER': '',
    'PASSWORD': '',
    'HOST': '',
    'PORT': '',
  }
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)

PostGreSQL (django.db.backends.postgresql_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

$ python manage.py runserver

You will get something like the following on running the above code −

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

7.2.2 Create Application

We assume you are in your project folder. In our main "myproject" folder, the same folder then manage.py −

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a "myapp" folder with the application structure −

myapp/

__init_.py

admin.py

models.py

tests.py

views.py

__init_.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.messages',
  'django.contrib.staticfiles',
  'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms
class LoginForm(forms.Form):
  user = forms.CharField(max_length = 100)
  password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm
def login(request):
  username = "not logged in"
   if request.method == "POST":
     #Get the posted form
     MyLoginForm = LoginForm(request.POST)
      if MyLoginForm.is_valid():
        username = MyLoginForm.cleaned_data['username']
  else:
     MyLoginForm = Loginform()
          return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
  <body>
     <form name = "form" action = "{% url "myapp.views.login" %}"
       method = "POST" >{% csrf_token %}
       <div style = "max-width:470px;">
         <center>
           <input type = "text" style = "margin-left:20%;"
             placeholder = "Identifiant" name = "username" />
         </center>
       </div>
                          <br>
       <div style = "max-width:470px;">
         <center>
           <input type = "password" style = "margin-left:20%;"
             placeholder = "password" name = "password" />
         </center>
       </div>
```

```
                    <br>
    <div style = "max-width:470px;">
      <center>
    <button style = "border:0px; background-color:#4285F4; margin-top:8%;
          height:35px; width:80%;margin-left:19%;" type = "submit"
          value = "Login" >
          <strong>Login</strong>
        </button>
        </center>
     </div>
 </form>
</body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>
  <body>
    You are : <strong>{{username}}</strong>
  </body>
 </html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
  url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
  url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered −
Setting Up Sessions
In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −
'django.contrib.sessions.middleware.SessionMiddleware'
And INSTALLED_APPS should have −
'django.contrib.sessions'
By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.
Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.
For this, first lets change our login view to save our username cookie server side −

```
def login(request):
   username = 'not logged in'
   if request.method == 'POST':
      MyLoginForm = LoginForm(request.POST)
       if MyLoginForm.is_valid():
         username = MyLoginForm.cleaned_data['username']
         request.session['username'] = username
      else:
         MyLoginForm = LoginForm()
  return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
   if request.session.has_key('username'):
      username = request.session['username']
      return render(request, 'loggedin.html', {"username" : username})
   else:
      return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
   url(r'^connection/','formView', name = 'loginform'),
   url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

8. SYSTEM STUDY

8. SYSTEM STUDY

8.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

8.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

8.1.2 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. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

8.1.3 SOCIAL 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. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

9. SYSTEM TESTING

9. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

9.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

9.3 Functional test

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

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

9.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

9.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

9.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is

treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

9.7 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

9.8 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

9.9 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9.10 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9.11 Test Cases

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	Set Video Source	By Help of CV2 open library we can set the video	Pass	If opencv not installed then it wont work
2.	Live capture	System primary cam we can use for HAR System	Pass	If system does not have then it will fail
3.	Load video file from disk	The video file format important here	Pass	Based on video format it will work
4.	Loading Models	Use can load the Models	Pass	First model has to download
5.	Test with Resnet50	This models download and got predict results	Pass	Depends on net speed
6.	Test with VGG16 Models	Model Loaded and Got results	Pass	Model Depends on net
7.	Split video file into frames	Video file splitted in to frames	Pass	If format not supports then it wont work
8.	Predict the Recognition names	Loaded the Recognitions name and displayed in the frames	Pass	If dataset not available then it will fail
9.	Calculate accuracy	Accuracy score calculated	Pass	Accuracy score based on image pixel values
10.	Calculate precession	By Help of sklearn package we can calculate the precession scores	Pass	Input value changes the it wont work

10. SAMPLE CODE

10. SAMPLE CODE

Mainrun.py

```
import cv2
import argparse
import os
import subprocess
from random import randrange
folder = 'frames'
for filename in os.listdir(folder):
    file_path = os.path.join(folder, filename)
    try:
        if os.path.isfile(file_path) or os.path.islink(file_path):
            os.unlink(file_path)
        elif os.path.isdir(file_path):
            shutil.rmtree(file_path)
    except Exception as e:
        print('Failed to delete %s. Reason: %s' % (file_path, e))
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="",
            help="optional path to video file")
args = vars(ap.parse_args())
```

```
def FrameCapture(path):
    vidObj = cv2.VideoCapture(path)
    count = 0
    success = 1
    #while success:
    while count<=25:
        # vidObj object calls read
        # function extract frames
        success, image = vidObj.read()
        # Saves the frames with frame-count
        cv2.imwrite("frames/frame%d.jpg" % count, image)
        count += 1
  # Driver Code
if__name__== '_main_':
    # Calling the function
    FrameCapture(args["input"])
    inputImage = "frames/frame"+str(randrange(25))+".jpg"
    runvalue = "classify_image.py -i "+inputImage
    subprocess.call("python "+runvalue)
    harActivity = "human_activity_reco.py --model resnet-34_kinetics.onnx --classes
action_recognition_kinetics.txt --input "+args["input"]
```

```
    subprocess.call("python "+harActivity)

    scores = "Reportgenearation.py"

    subprocess.call("python "+scores)
```

HAR Model

```
# USAGE

# python human_activity_reco.py --model resnet-34_kinetics.onnx --classes
action_recognition_kinetics.txt --input example_activities.mp4

# python human_activity_reco.py --model resnet-34_kinetics.onnx --classes
action_recognition_kinetics.txt

# import the necessary packages

import numpy as np

import argparse

import imutils

import sys

import cv2

# construct the argument parser and parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-m", "--model", required=True,

        help="path to trained human activity recognition model")

ap.add_argument("-c", "--classes", required=True,

        help="path to class labels file")

ap.add_argument("-i", "--input", type=str, default="",
```

```
        help="optional path to video file")
args = vars(ap.parse_args())
# load the contents of the class labels file, then define the sample
# duration (i.e., # of frames for classification) and sample size
# (i.e., the spatial dimensions of the frame)
CLASSES = open(args["classes"]).read().strip().split("\n")
SAMPLE_DURATION = 16
SAMPLE_SIZE = 112
# load the human activity recognition model
print("[INFO] loading human activity recognition model...")
net = cv2.dnn.readNet(args["model"])
# grab a pointer to the input video stream
print("[INFO] accessing video stream...")
vs = cv2.VideoCapture(args["input"] if args["input"] else 0)
# loop until we explicitly break from it
while True:
        # initialize the batch of frames that will be passed through the
        # model
        frames = []
        # loop over the number of required sample frames
        for i in range(0, SAMPLE_DURATION):
```

```
            # read a frame from the video stream
            (grabbed, frame) = vs.read()
            # if the frame was not grabbed then we've reached the end of
            # the video stream so exit the script
            if not grabbed:
                print("[INFO] no frame read from stream - exiting")
                sys.exit(0)
            # otherwise, the frame was read so resize it and add it to
            # our frames list
            frame = imutils.resize(frame, width=400)
            frames.append(frame)
# now that our frames array is filled we can construct our blob
        blob = cv2.dnn.blobFromImages(frames, 1.0,
            (SAMPLE_SIZE, SAMPLE_SIZE), (114.7748, 107.7354, 99.4750),
            swapRB=True, crop=True)
        blob = np.transpose(blob, (1, 0, 2, 3))
        blob = np.expand_dims(blob, axis=0)
# pass the blob through the network to obtain our human activity
        # recognition predictions
        net.setInput(blob)
        outputs = net.forward()
```

```
    label = CLASSES[np.argmax(outputs)]

# loop over our frames
    for frame in frames:
        # draw the predicted activity on the frame
        cv2.rectangle(frame, (0, 0), (300, 40), (0, 0, 0), -1)
        cv2.putText(frame, label, (10, 25), cv2.FONT_HERSHEY_SIMPLEX,
            0.8, (255, 255, 255), 2)
        # display the frame to our screen
        cv2.imshow("Activity Recognition", frame)
        key = cv2.waitKey(1) & 0xFF
        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break
```

Classify Image:

```
# USAGE
# python classify_image.py --image images/soccer_ball.jpg --model vgg16
# import the necessary packages
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications import Xception # TensorFlow ONLY
from tensorflow.keras.applications import VGG16
```

```
from tensorflow.keras.applications import VGG19
#from tensorflow.keras.applications import imagenet_utils
#from keras.applications.vgg16 import preprocess_input, decode_prediction
from keras.applications import imagenet_utils
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
import numpy as np
import argparse
import cv2
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
	help="path to the input image")
ap.add_argument("-model", "--model", type=str, default="vgg16",
	help="name of pre-trained network to use")
args = vars(ap.parse_args())
# define a dictionary that maps model names to their classes
# inside Keras
MODELS = {
	"vgg16": VGG16,
```

```
        "vgg19": VGG19,
        "inception": InceptionV3,
        "xception": Xception, # TensorFlow ONLY
        "resnet": ResNet50
}
```

 # esnure a valid model name was supplied via command line argument

```
if args["model"] not in MODELS.keys():
        raise AssertionError("The --model command line argument should "
                "be a key in the `MODELS` dictionary")
```

 # initialize the input image shape (224x224 pixels) along with

 # the pre-processing function (this might need to be changed

 # based on which model we use to classify our image)

```
inputShape = (224, 224)
preprocess = imagenet_utils.preprocess_input
```

 # if we are using the InceptionV3 or Xception networks, then we

 # need to set the input shape to (299x299) [rather than (224x224)]

 # and use a different image pre-processing function

```
if args["model"] in ("inception", "xception"):
        inputShape = (299, 299)
        preprocess = preprocess_input
```

```
# load our the network weights from disk (NOTE: if this is the
# first time you are running this script for a given network, the
# weights will need to be downloaded first -- depending on which
# network you are using, the weights can be 90-575MB, so be
# patient; the weights will be cached and subsequent runs of this
# script will be *much* faster)
print("[INFO] loading {}...".format(args["model"]))
Network = MODELS[args["model"]]
model = Network(weights="imagenet")
# load the input image using the Keras helper utility while ensuring
# the image is resized to `inputShape`, the required input dimensions
# for the ImageNet pre-trained network
print("[INFO] loading and pre-processing image...")
image = load_img(args["image"], target_size=inputShape)
image = img_to_array(image)
# our input image is now represented as a NumPy array of shape
# (inputShape[0], inputShape[1], 3) however we need to expand the
# dimension by making the shape (1, inputShape[0], inputShape[1], 3)
# so we can pass it through the network
image = np.expand_dims(image, axis=0)
```

```
# pre-process the image using the appropriate function based on the
# model that has been loaded (i.e., mean subtraction, scaling, etc.)
image = preprocess(image)
# classify the image
print("[INFO] classifying image with '{}'...".format(args["model"]))
preds = model.predict(image)
print("Type is ",type(model))
P = imagenet_utils.decode_predictions(preds)
# loop over the predictions and display the rank-5 predictions +
# probabilities to our terminal
for (i, (imagenetID, label, prob)) in enumerate(P[0]):
        print("{}. {}: {:.2f}%".format(i + 1, label, prob * 100))
# load the image via OpenCV, draw the top prediction on the image,
# and display the image to our screen
orig = cv2.imread(args["image"])
(imagenetID, label, prob) = P[0][0]
cv2.putText(orig, "Label: {}, {:.2f}%".format(label, prob * 100),
        (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
cv2.imshow("Classification", orig)
cv2.waitKey(0)
```

Reportgeneration.py

```
# demonstration of calculating metrics for a neural network model using sklearn
from sklearn.datasets import make_circles
from sklearn.datasets import load_sample_image
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics  import roc_auc_score
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.layers import Dense
from sklearn.feature_extraction import image
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
# generate and prepare the dataset
def get_data():
    # generate dataset
```

```
    X, y = make_circles(n_samples=1000, noise=0.1, random_state=1)
    #print("Which type you are ..?= am ",len(X),len(y))
    #print(X)
    #print(" ----------")
    #print(y)
    # split into train and test
    n_test = 500
    trainX, testX = X[:n_test, :], X[n_test:, :]
    trainy, testy = y[:n_test], y[n_test:]
    one_image = load_sample_image("flower.jpg")
    print('Image shape: {}'.format(one_image.shape))
    patches = image.extract_patches_2d(one_image, (2, 2))
    print('Patches shape: {}'.format(patches.shape))
    print(patches[1])
    print(patches[800])
    return trainX, trainy, testX, testy
# define and fit the model
def get_model(trainX, trainy):
    # define model
    model = Sequential()
    model.add(Dense(100, input_dim=2, activation='relu'))
```

```
    model.add(Dense(1, activation='sigmoid'))
    # compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    # fit model
    model.fit(trainX, trainy, epochs=300, verbose=0)
    return model
# generate data
trainX, trainy, testX, testy = get_data()
# fit model
model = get_model(trainX, trainy)
# predict probabilities for test set
yhat_probs = model.predict(testX, verbose=0)
# predict crisp classes for test set
yhat_classes = model.predict_classes(testX, verbose=0)
# reduce to 1d array
yhat_probs = yhat_probs[:, 0]
yhat_classes = yhat_classes[:, 0]
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(testy, yhat_classes)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
```

```
precision = precision_score(testy, yhat_classes)

print('Precision: %f' % precision)

# recall: tp / (tp + fn)

recall = recall_score(testy, yhat_classes)

print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)

f1 = f1_score(testy, yhat_classes)

print('F1 score: %f' % f1)

# kappa

kappa = cohen_kappa_score(testy, yhat_classes)

print('Cohens kappa: %f' % kappa)

# ROC AUC

auc = roc_auc_score(testy, yhat_probs)

print('ROC AUC: %f' % auc)

# confusion matrix

f,ax = plt.subplots(figsize=(8, 8))

matrix = confusion_matrix(testy, yhat_classes)

sns.heatmap(matrix, annot=True, linewidths=0.01,cmap="Blues",linecolor="gray", fmt=
'.1f',ax=ax)

plt.xlabel("Predicted Label")

plt.ylabel("True Label")
```

```
plt.title("Confusion Matrix")

plt.show()

print(matrix)
```

Test lstm model:

```
# cnn lstm model

from numpy import mean

from numpy import std

from numpy import dstack

from pandas import read_csv

from keras.models import Sequential

from keras.layers  import Dense

from keras.layers import Flatten

from keras.layers import Dropout

from keras.layers import LSTM

from keras.layers import TimeDistributed

from keras.layers.convolutional import Conv1D

from keras.layers.convolutional import MaxPooling1D

from keras.utils import to_categorical

from matplotlib import pyplot

# load a single file as a numpy array

def load_file(filepath):
```

```
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values
# load a list of files and return as a 3d numpy array
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded
# load a dataset group, such as train or test
def load_dataset_group(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt',
'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt',
'body_acc_z_'+group+'.txt']
```

```
        # body gyroscope
        filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt',
'body_gyro_z_'+group+'.txt']
        # load input data
        X = load_group(filenames, filepath)
        # load class output
        y = load_file(prefix + group + '/y_'+group+'.txt')
        return X, y
# load the dataset, returns train and test X and y elements
def load_dataset(prefix=''):
        # load all train
        trainX, trainy = load_dataset_group('train', prefix + 'HARDataset/')
        print(trainX.shape, trainy.shape)
        # load all test
        testX, testy = load_dataset_group('test', prefix + 'HARDataset/')
        print(testX.shape, testy.shape)
        # zero-offset class values
        trainy = trainy - 1
        testy = testy - 1
        # one hot encode y
        trainy = to_categorical(trainy)
```

```
    testy = to_categorical(testy)
    print(trainX.shape, trainy.shape, testX.shape, testy.shape)
    return trainX, trainy, testX, testy
# fit and evaluate a model
def evaluate_model(trainX, trainy, testX, testy):
    # define model
    verbose, epochs, batch_size = 0, 25, 64
    n_timesteps, n_features, n_outputs = trainX.shape[1], trainX.shape[2], trainy.shape[1]
    # reshape data into time steps of sub-sequences
    n_steps, n_length = 4, 32
    trainX = trainX.reshape((trainX.shape[0], n_steps, n_length, n_features))
    testX = testX.reshape((testX.shape[0], n_steps, n_length, n_features))
    # define model
    model = Sequential()
    model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3, activation='relu'),
input_shape=(None,n_length,n_features)))
    model.add(TimeDistributed(Conv1D(filters=64, kernel_size=3, activation='relu')))
    model.add(TimeDistributed(Dropout(0.5)))
    model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(100))
```

```
        model.add(Dropout(0.5))
        model.add(Dense(100, activation='relu'))
        model.add(Dense(n_outputs, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        # fit network
        model.fit(trainX, trainy, epochs=epochs, batch_size=batch_size, verbose=verbose)
        # evaluate model
        _, accuracy = model.evaluate(testX, testy, batch_size=batch_size, verbose=0)
        return accuracy
# summarize scores
def summarize_results(scores):
        print(scores)
        m, s = mean(scores), std(scores)
        print('Accuracy: %.3f%% (+/-%.3f)' % (m, s))
# run an experiment
def run_experiment(repeats=10):
        # load data
        trainX, trainy, testX, testy = load_dataset()
        # repeat experiment
        scores = list()
        for r in range(repeats):
```

```
            score = evaluate_model(trainX, trainy, testX, testy)
            score = score * 100.0
            print('>#%d: %.3f' % (r+1, score))
            scores.append(score)
    # summarize results
    summarize_results(scores)
# run the experiment
run_experiment()
```

11. INPUT & OUTPUT DESIGN

11.INPUT AND OUTPUT DESIGN

11.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

11.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information
- Create document, report, or other formats that contain information produced by the system.

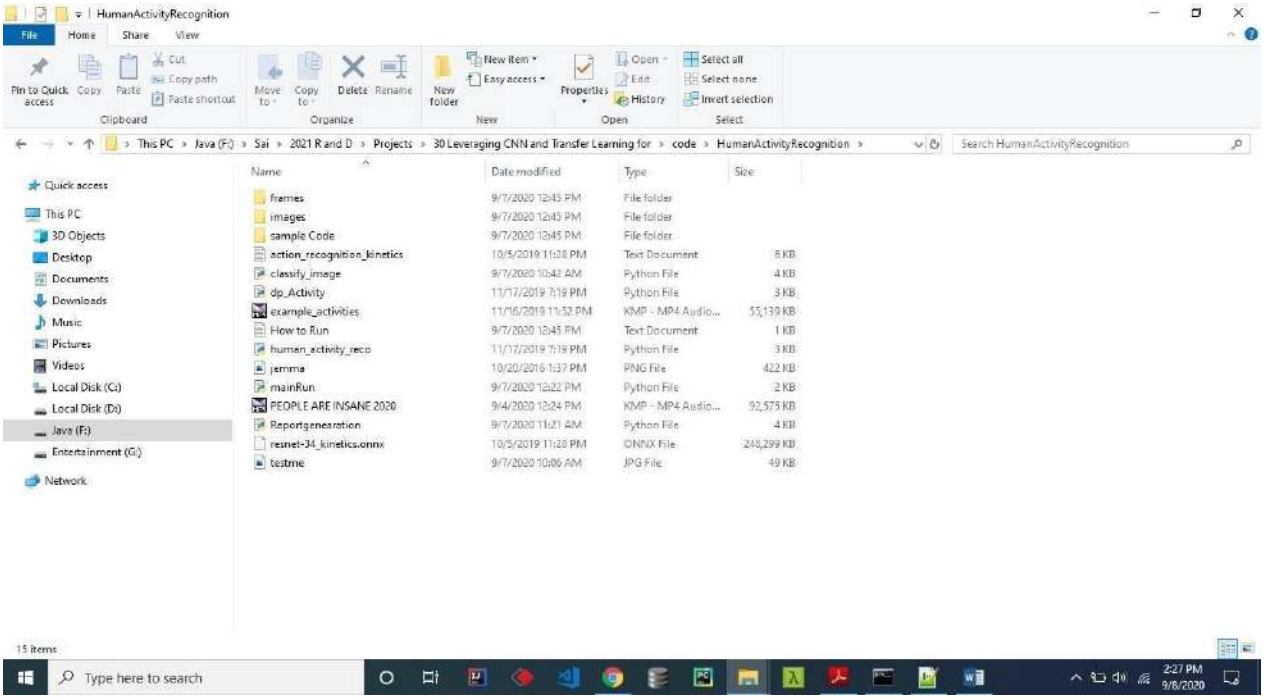
The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

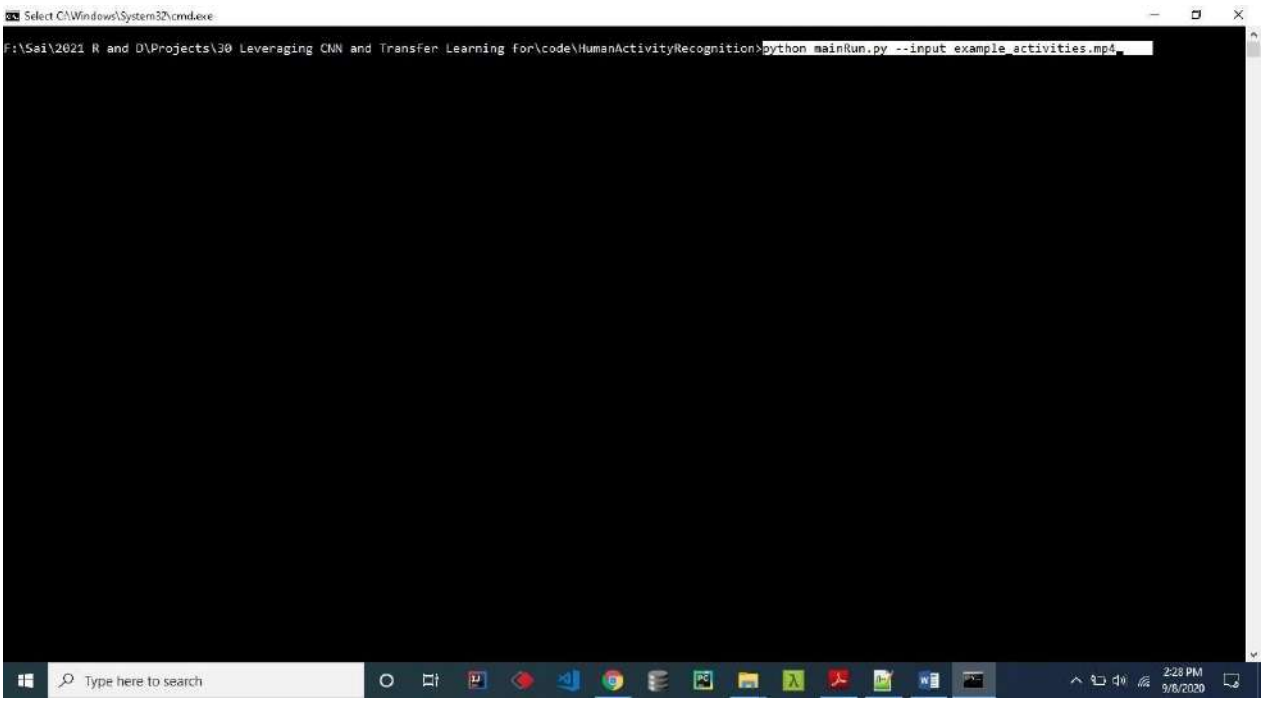
12. SCREENSHOTS

12.SCREEN SHOTS

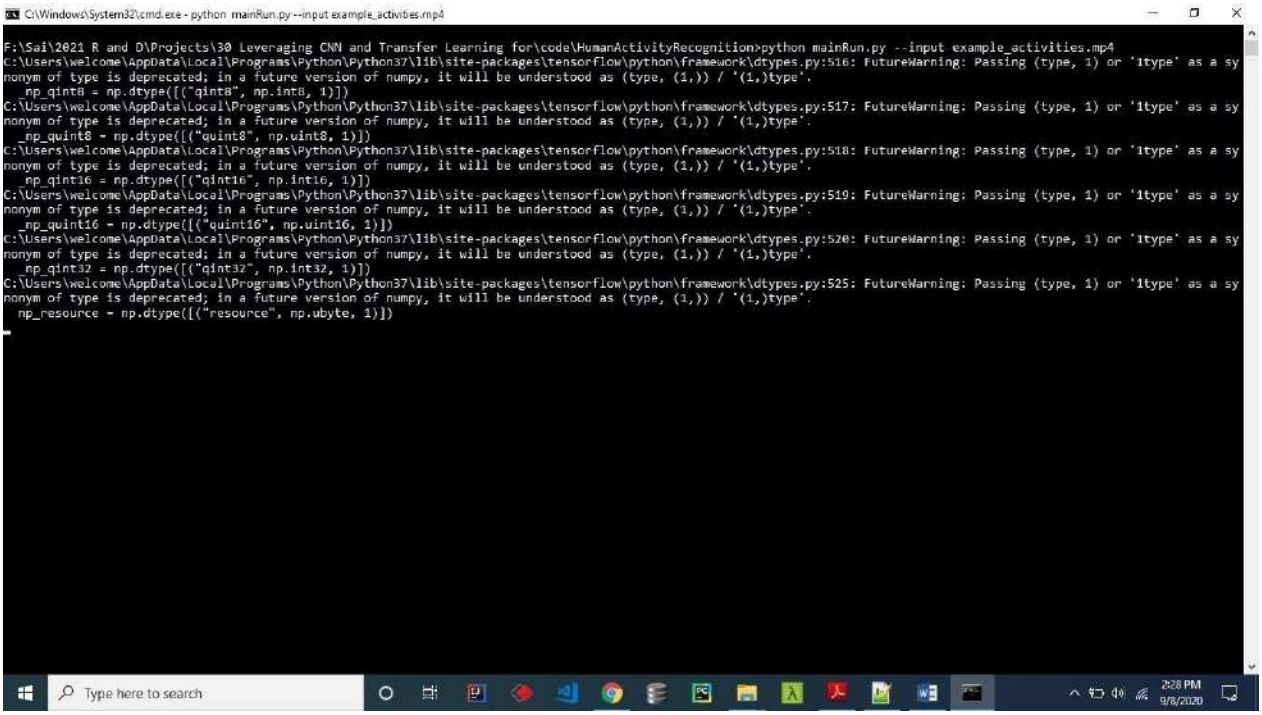
12.1 Starting Project:



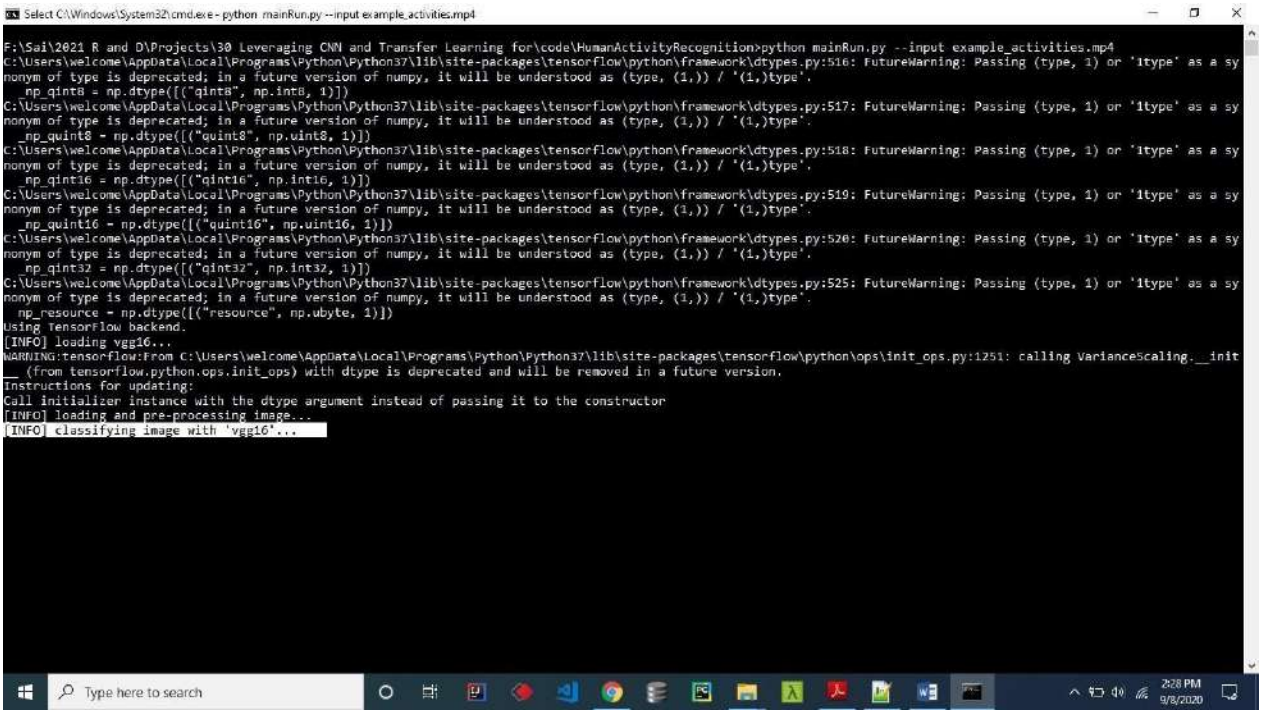
12.2 Run the main Program:



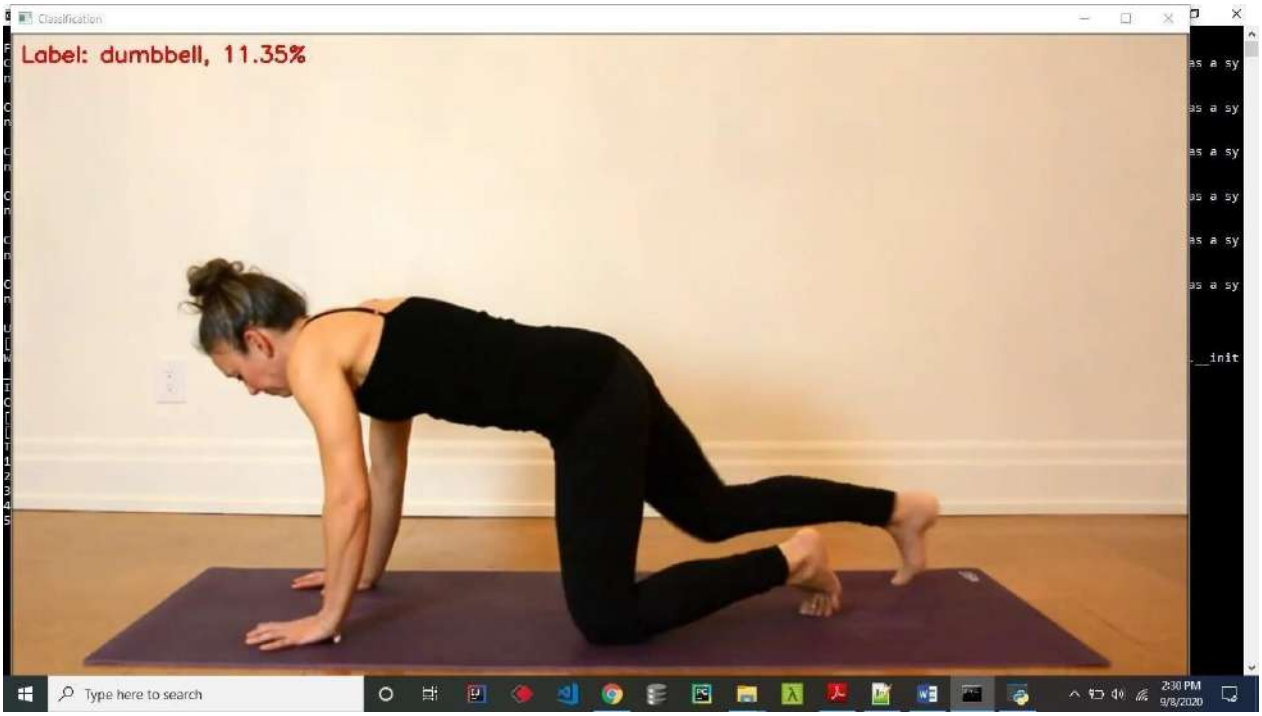
12.3 Loading Tensor flow Libraries:



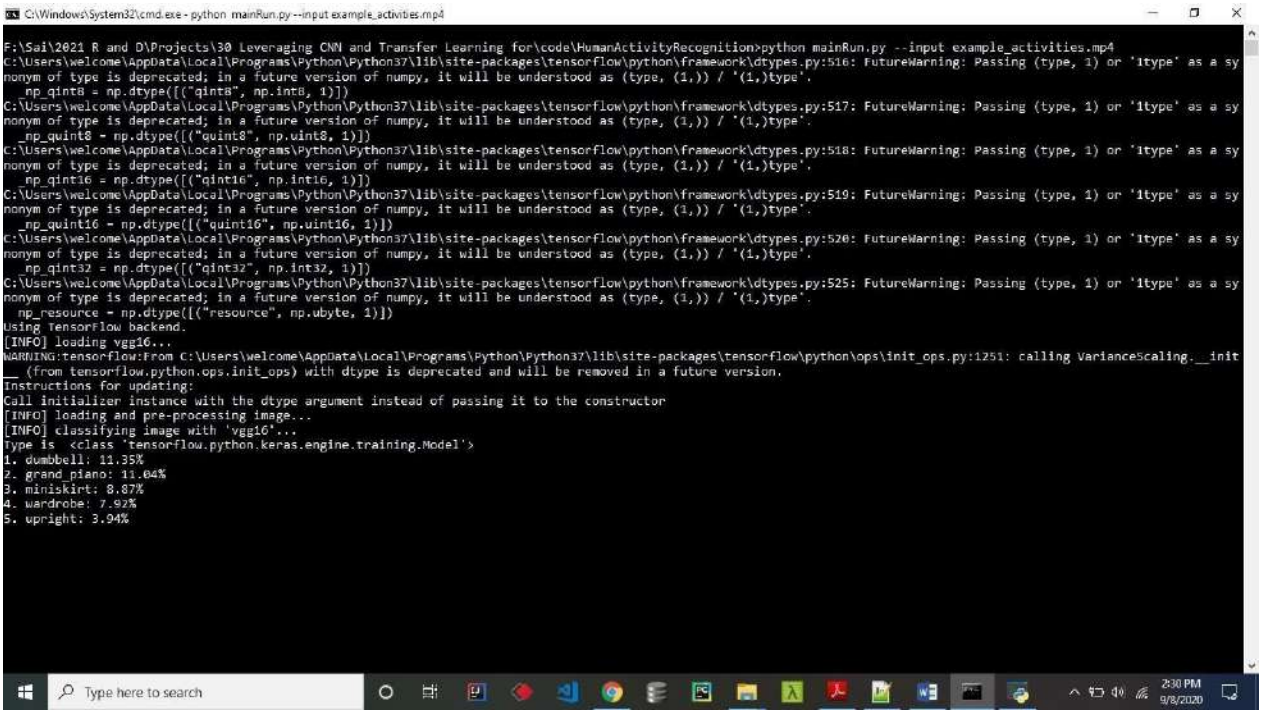
12.4 Classification with vgg16:



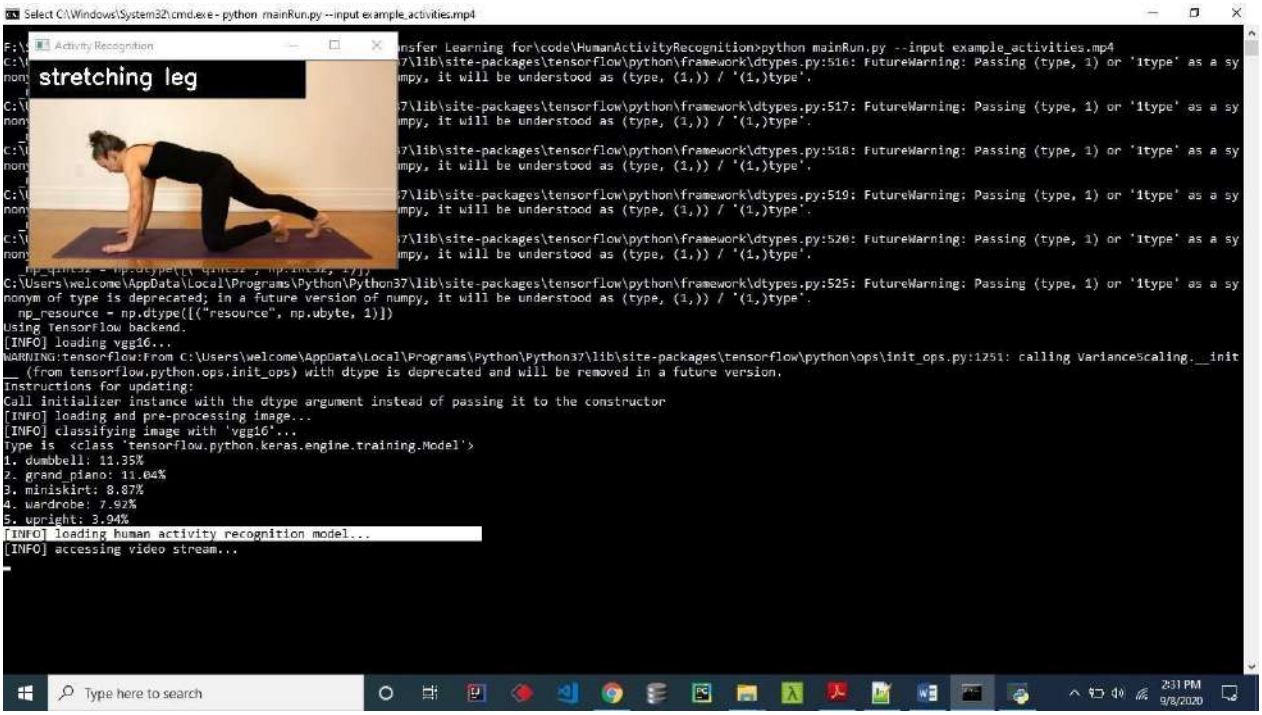
12.5 Get Image label:



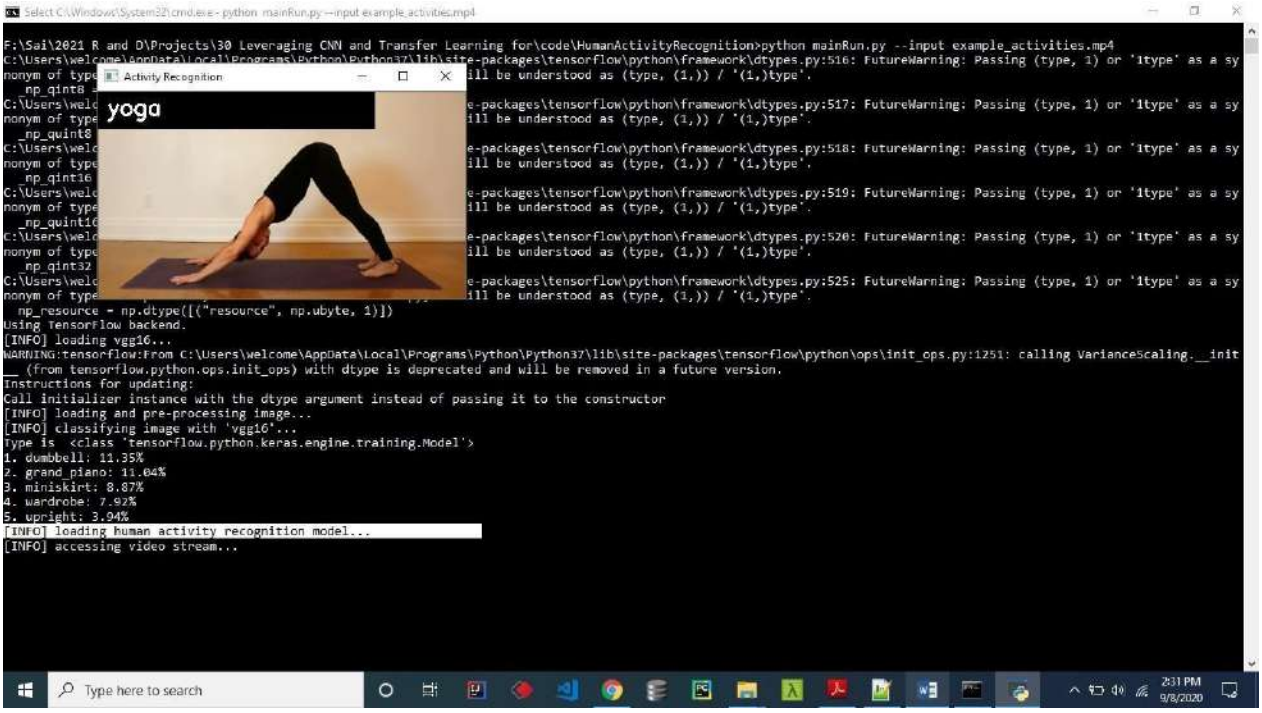
12.6 Result from image:



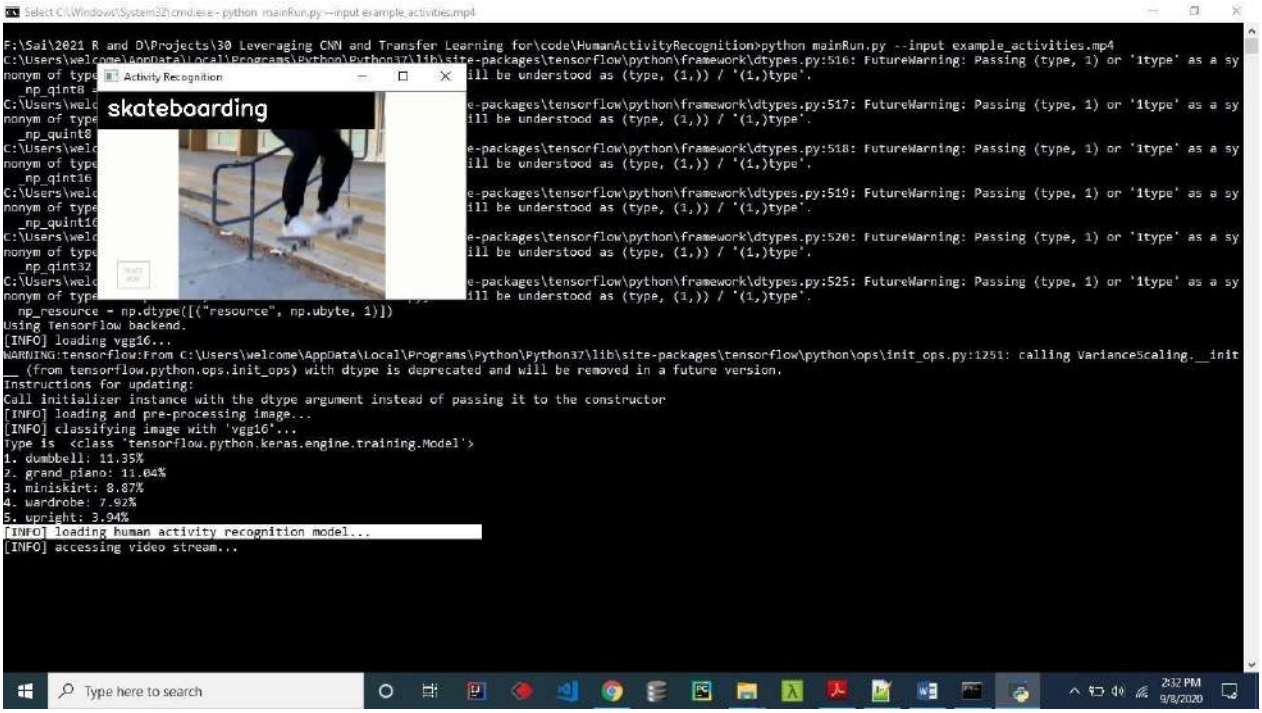
12.7 Loading model HAR:



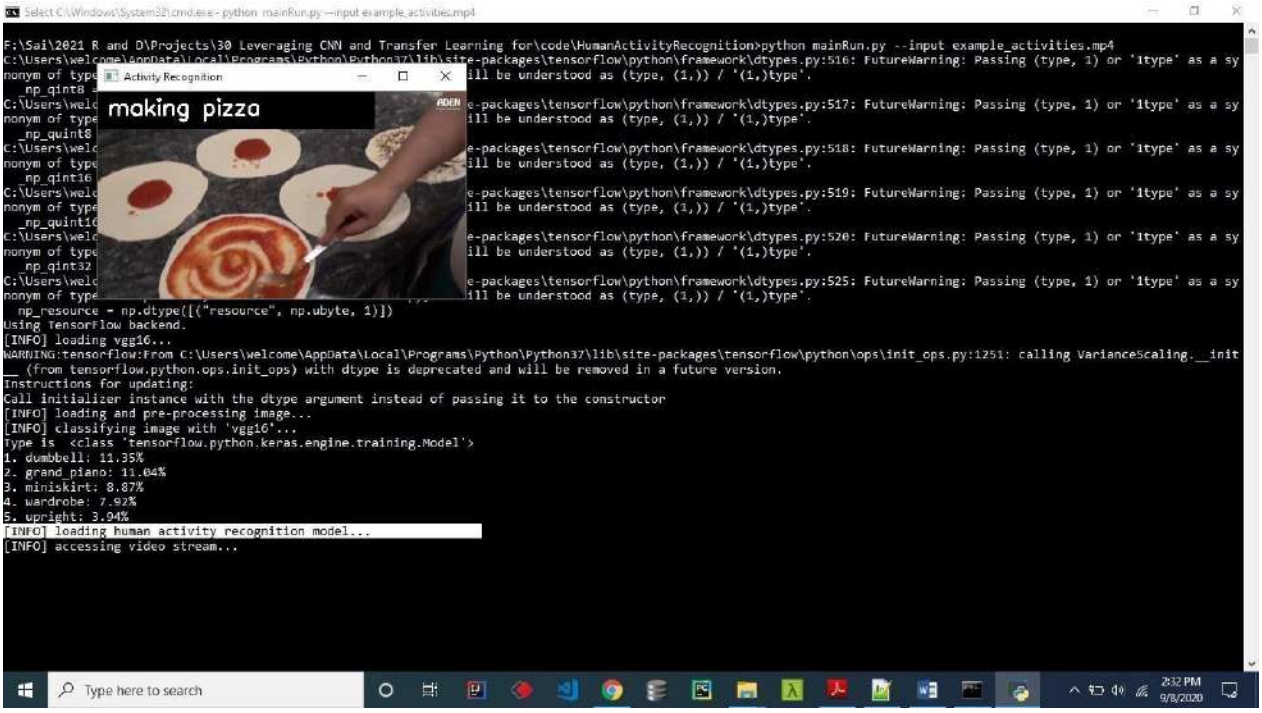
12.8 Result 1:



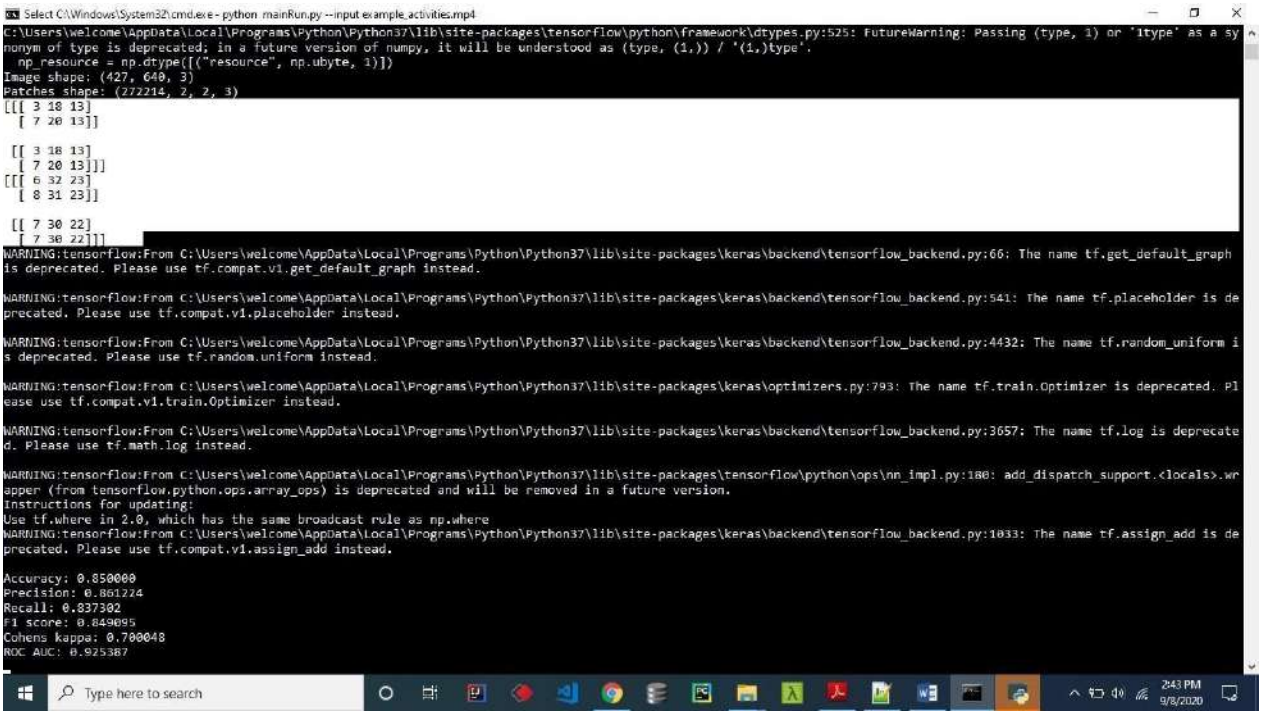
12.9 Result 2:



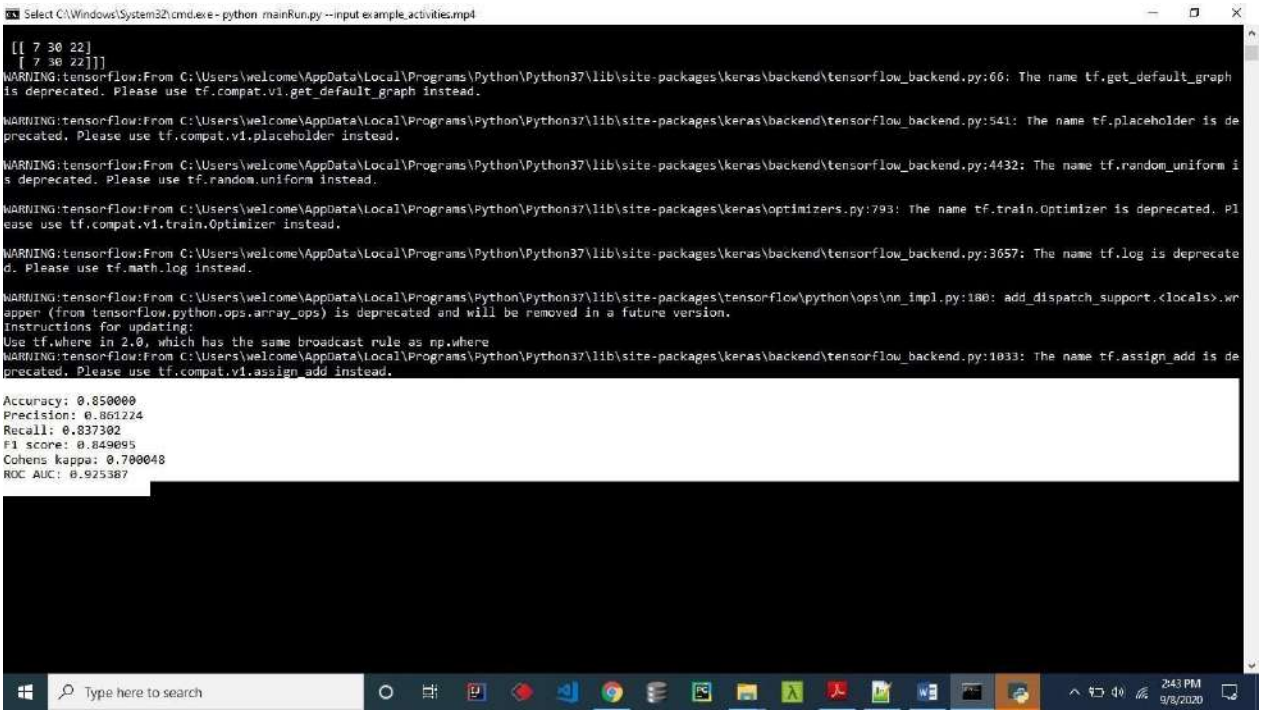
12.10 Result 3:



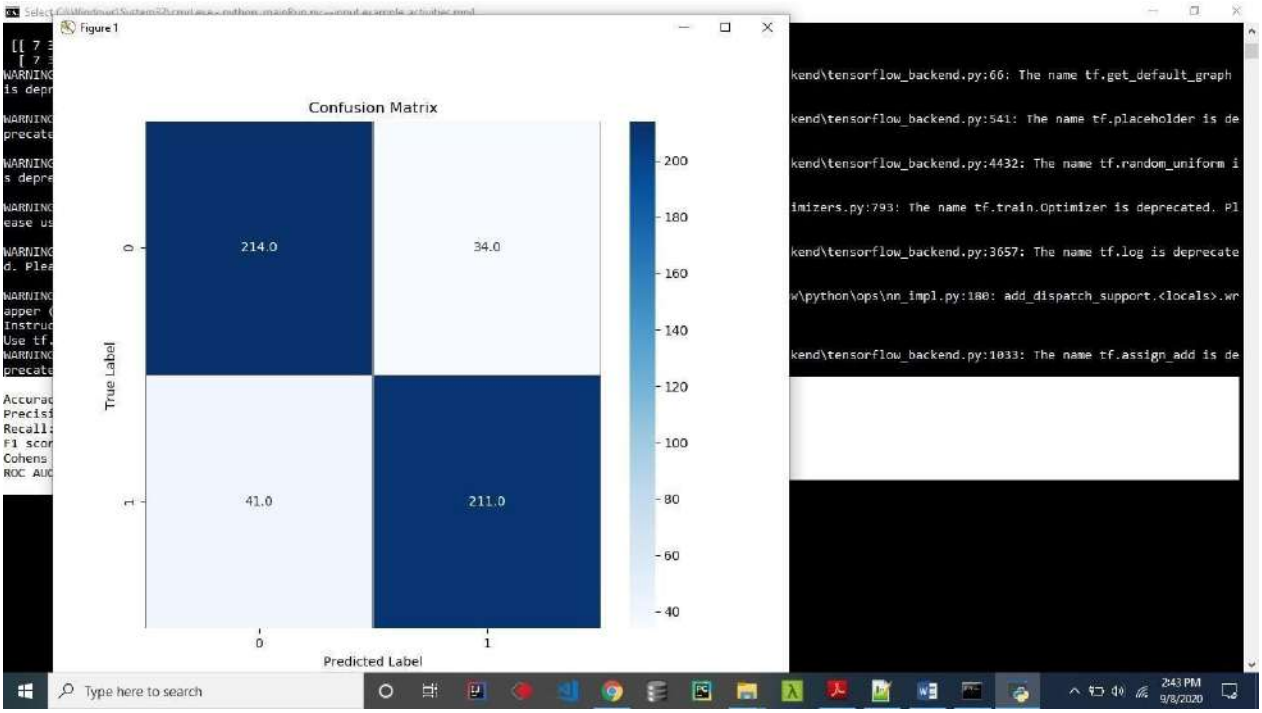
12.11 Patches from image:



12.13 Accuracy:



12.14 Confusion matrix:



13. CONCLUSION

13. CONCLUSION

We used CNN models to predict the human activities from Wiezmann Dataset. We experimented with 3 different Convolutional Neural Networks (CNN) for activity recognition. We have employed transfer learning to get the deep image features and trained machine learning classifiers. Our experimental results showed the accuracy of 96.95% using VGG-16 with the implementation of transfer learning. Our experimental results showed that VGG-16 outperformed other CNN models in terms of feature extraction. Our experimental results with transfer learning technique also showed high performance of VGG-16 as compared to state-of-the-art methods.

14. BIBILOGRAPHY

14. BIBILOGRAPHY

14.1 REFERENCES

1. B. Bhandari, J. Lu, X. Zheng, S. Rajasegarar, and C. Karmakar, "Noninvasive sensor based automated smoking activity detection," in Pro-ceedings of Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2017, pp. 845–848.

2. L. Yao, Q. Z. Sheng, X. Li, T. Gu, M. Tan, X. Wang, S. Wang, and W. Ruan, "Compressive representation for device-free activity recognition with passive rfid signal strength," IEEE Transactions on Mobile Computing, vol. 17, no. 2, pp. 293–306, 2018.

3. I. Lillo, J. C. Niebles, and A. Soto, "Sparse composition of body poses and atomic actions for human activity recognition in rgb-d videos," Image and Vision Computing, vol. 59, pp. 63–75, 2017.

4. W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks," in ThirtiethAAAI Conference on Artificial Intelligence, 2016.

5. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

6. J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June 2009, pp. 248–255.

7. A. Jalal, N. Sarif, J. T. Kim, and T.-S. Kim, "Human activity recognition via recognized body parts of human depth silhouettes for residents monitoring services at smart home," Indoor and built environment, vol. 22, no. 1, pp. 271–279, 2013.

8. K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in Advances in neural information processing systems, 2014, pp. 568–576.

9. G. Gkioxari, R. Girshick, and J. Malik, “Contextual action recognition with r* cnn,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1080–1088.

10. L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, “Towards good practices for very deep two- stream convnets,” arXiv preprint arXiv:1507.02159, 2015.

11. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Deep end2end voxel2voxel prediction,” in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2016, pp. 17–24.

12. P. Wang, W. Li, J. Wan, P. Ogunbona, and X. Liu, “Cooperative trainingof deep aggregation networks for rgb-d action recognition,” in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

13. S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 1, pp. 221–231, 2013.

14. P. Khaire, P. Kumar, and J. Imran, “Combining cnn streams of rgb-d and skeletal data for human activity recognition,” Pattern Recognition Letters, vol. 115, pp. 107–116, 2018.

15. A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

16. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 4489–4497.

17. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

18. Z. Wharton, E. Thomas, B. Debnath, and A. Behera, “A vision-based transfer learning approach for recognizing behavioral symptoms in people with dementia,” in 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2018, pp. 1–6.

19. J. Cai, X. Tang, and R. Zhong, “Silhouettes based human action recognition by procrustes analysis and fisher vector encoding,” in International Conference on Image and Video Processing, and Artificial Intelligence, vol. 10836. International Society for Optics and Photonics,2018, p. 1083612.

20. S. S. Kumar and M. John, “Human activity recognition using optical flow based feature set,” in Proceedings of IEEE International Carnahan conference on security technology (ICCST). IEEE, 2016, pp. 1–5.

21. W. Feng, H. Tian, and Y. Xiao, “Research on temporal structure for action recognition,” in Chinese Conference on Biometric Recognition. Springer, 2017, pp. 625–632.

22. P. Y. Han, K. E. Yee, and O. S. Yin, “Localized temporal representation in human action recognition,” in Proceedings of International Confer-ence on Network, Communication and Computing. ACM, 2018, pp.261–266..

14.2 GIT HUB REPOSITORY LINK

1. https://github.com/bmanishkumar11/VISION-BASED-HUMAN-ACTIVITY-RECOGNITION

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 8, Issue 2 - V8I2-1320)
Available online at: https://www.ijariit.com

Vison-based human activity recognition

M. Madhusudhan
madhusudhan.cse@cmrtc.ac.in
CMR Technical Campus, Hyderabad, Telangana

B. Manish Kumar
177r1a05j5@cmrtc.ac.in
CMR Technical Campus, Hyderabad, Telangana

P Rohit
177r1a0599@cmrtc.ac.in
CMR Technical Campus, Hyderabad, Telangana

V. Sri Ram Reddy
167r1a05p7@cmrtc.ac.in
CMR Technical Campus, Hyderabad, Telangana

V. Sai Chandana
177r1a0559@cmrtc.ac.in
CMR Technical Campus, Hyderabad, Telangana

ABSTRACT

There have been substantial advances in HAR (Human Activity Recognition) in the past several years due to the advancement of the IoT (Internet of Things). HAR may be used in a range of contexts, including elder care, surveillance systems, and anomalous behavior detection. Different machine learning techniques have been used to anticipate human actions in a particular circumstance. Feature engineering techniques, which may pick an ideal collection of features, have outperformed typical machine learning techniques. Deep learning models, like CNN (Convolutional Neural Networks), on the other hand, are known to extract features and minimize computing costs automatically. We employ the CNN model for predicting actions from the Weizmann Dataset in this article. To extract deep image features and trained machine learning classifiers, transfer learning is used in particular. We found that VGG-16 has an accuracy of 96.95 percent in our experiments. We also found that VGG-16 outperformed the rest of the CNN models that were used in our experiments.

Keywords: *Convolutional Neural Network, Activity recognition, Deep Learning.*

1. INTRODUCTION

HAR is a prominent study topicdue to its wider applications in surveillance systems, automated homes, and elderly care. Human activity recognition has been the subject of several topics in the last few years. Existing works are either wearable or non-wearable. A wearable HAR system employs sensors that are fixed on the body. The nature of wearable-based HAR systems is intrusive. Non-wearable HAR does not need any sensors to be attached to the person or the carrying of any device to recognize the activity. These systems are further classified into two types: (i) Sensor-based and (ii) vision-based. The sensor-based system identifies human activity using RF signals from sensors including Wi-Fi signals, PIR sensors, and RFID [1]. To identify human actions, a vision-based system uses images, video frames from IR, or depth cameras. Sensor-based HAR systems are non-intrusive but can't offer a higher level of accuracy. As a result, vision-based systems are gaining popularity at present, however, extracting human activities from the live video is difficult.

According to motion features, video-based activity recognition may be classified as marker-based or vision-based. The optic wearable marker-based Mocap (motion capture) system is utilized in the marker-based technique. However, it can correctly record complicated human actions, this method has certain drawbacks. There must be an optical sensor fixed on the human body with different camera settings. The depth or RGB image is used in the vision-based approach. It does not need the use of external devices or the attachment of sensors to the body of the user. Since this method is gaining prominence, the HAR system is becoming simpler and easier to apply in a variety of settings [2].

There are a few vision-based HAR systems that use typical machine learning approaches for activity detection. The usage of deep learning techniques instead of standard approaches to machine learning has increased significantly. CNNs are often used in computer vision applications. Images are processed using a sequence of convolutional layers [3]. From the Weizmann Dataset, we utilize CNN to identify human activities. The frames for each action were first retrieved from the videos. Transfer learning is used to acquire deep image features as well as trained machine learning classifiers. To categorize activities, we used three different CNN algorithms and compared our findings to previous work on the same dataset.

2. RELATED WORK

Vision-based human activity identification has recently gotten significant attention. Handcrafted feature extraction from videos/images and conventional classifiers for activity identification has been used in the majority of the studies. In many cases, conventional methods yielded the best outcomes and showed the highest levels of performance. Traditional approaches, on the other hand, are impractical to utilize in the real world since handcrafted characteristics are heavily reliant on data and are not flexible enough to adapt to changing conditions [4].

Because of its ability to decode temporal patterns, HMM ("Hidden Markov Model") approaches are widely employed as recognition methods in recent years. However, researchers are increasingly turning to deep learning algorithms because of their proficiency in extracting features automatically and learning deep pattern frameworks. In the computer vision field, deep learning algorithms have ruled out classical categorization techniques. These algorithms have recently significantly gained attention in the computer vision field, with excellent results. Consequently, video-based human activity identification using deep learning algorithms has received considerable interest in recent years.

A mixed-norm regularization function may be added to a deep LSTM network, Zhu et al. suggested an action classification technique. CNNs are one of the most widely used deep learning approaches in frame/image processing. Several studies have used 2D-CNNs, which take advantage of spatial correlation between video frames and integrate the results using several methodologies. Many people have employed optical flow as an extra input to 2D-CNN to gain temporal correlations information [IO]. Subsequently, 3D-CNNs were introduced, and they showed remarkable performance in video and frame classification.

Wang et al. used CNN to extract features from depth and RGB frames automatically. The collected features were fed into a fully connected neural network, which resulted in a higher level of accuracy. Ji et al. suggested a 3D CNN model for activity recognition that performs 3D convolutions and extracts temporal and spatial properties by recording motion data. ConvNet, a two-stream convolution layer design devised by Simonyan et al., may obtain excellent results despite insufficient training data.

Khaire et al. developed an algorithm to detect activities by training convnets using RGB-D datasets and combining SoftMax scores from depth, skeleton, as well as motion images at the classification level. Over a 4D video chunk, Karpathy et al. suggested extending CNN architecture in the first convolutional layers. While Tran et al. employed a deep 3D-CNN model, (like VGG net) to increase the model's accuracy by using spatiotemporal convolutions and pooling in all layers.

In contrast, we're more interested in seeing how transfer learning may be employed with CNN models for improving classification accuracy on a benchmark dataset [5].

3. TRANSFER LEARNING

Transfer learning is a technique for transferring information from previous extensive training to the present model. Transfer learning allows deep network algorithms to be trained with considerably fewer data. It was utilized to shorten the training time and increase the model's accuracy. We employ transfer learning in this paper to use information from large-scale datasets like ImageNet. The frames for each action are first extracted from the videos. Transfer learning is used to acquire deep image features as well as trained machine learning classifiers. The pre-trained weights on ImageNet are utilized as the preliminary step for transfer learning in all CNN models. ImageNet is a database of 2000 images (activities from 1 category). Knowledge is transferred from ImageNet weights to Weizmann datasets since the actions detected in this study fall within the ImageNet domain. The penultimate CNN layer is used to extract the features. Figure 1 illustrates the fundamental concept of transfer learning.

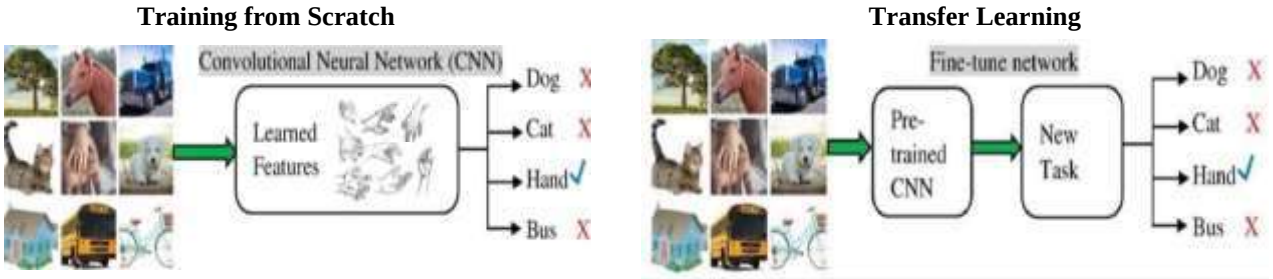


Fig. 1. Schematic representation of transfer learning

In transfer learning, the pre-trained neural model from a large-scale dataset is preserved while the weights are updated in the trained model and the pre-trained neural model is used to extract features.

4. IMPLEMENTATION

A. Dataset

We use trials on the Weizmann dataset to test the models' performance in terms of activity recognition. It consists of 90 low-resolution video frames depicting 9 different individuals doing ten different activities, including bend, jack (or jumping-jack), jump (or jump-forward-on-2-legs), pjump (or jump-in-place-on-2-legs), run, side (or gallop-sideways), slip, well, wave I (wave 1-hand), and wave2 (wave 2-hands). For our experiment, we employed nine different actions (excluding "pjump jump-in-place-on-2-legs"). All videos are first converted into separate frames depending on activity. Table 1 displays the total frames/activity for all 9 participants based on the extracted frames. The complete dataset is subdivided into three sections: testing 20%, training 70%, and validation 10%.

Table-1: Statistics for the dataset in terms of total frames/activity

Activity	No. of frames
Jack	729
Bend	639
Run	346
Jump	538
Skip	378
Side	444
Wave1	653
Wave2	624
Walk	566
Total	4917

B. Discussion and Results

We test three distinct CNNs for activity identification, such as Google's InceptionNet-v3, VGG-16, and VGG-19, to categorize activities. To use the information obtained from large-scale datasets like ImageNet, we applied transfer learning. Using the information gained from pre-trained weights on ImageNet, we experimented on the Weizmann dataset. CNNs' penultimate layers are used to extract the features. We used transfer learning on the VGG-16 CNN algorithm and obtained a 96.95 percent accuracy. VGG-16 takes a 224x224 image as an input and extracts features from the fc1 layer, yielding a 4096-D vector for each image.

To compare the performance of the various CNN models, we employed transfer learning on additional CNN models including Google's InceptionNet-v3 and VGG-19. VGG-19 obtained 96.54 percent and Google's InceptionNet-v3 obtained 95.63 percent, correspondingly. The experimental findings show that VGG-16 outperforms the other CNN models once transfer learning has been applied to all of the models. The accuracy, precision, f1-score, and recall of given models are reported in Table 2. Figures 2, 3, and 4 demonstrate the confusion matrix of three distinct CNN models [6].

On the Weizmann dataset, we were able to match the performance of other systems that did not include transfer learning. When using transfer learning to recognize the same dataset, the experiment results indicated that recognition scores improved. Transfer learning improves recognition accuracy by 1% to 6%. Table 3 compares the performance of the VGG-16 model using transfer learning to those of the other techniques. Transfer learning is compared against state-of-the-art techniques to see how successful it is when used with CNN algorithms for increasing recognition results [7].

Table-2: Activity recognition results based on several CNN models

Model	Precision	Accuracy	Recall	Precision	F1-Score
VGG-16	97.00%	96.95%	97.00%	97.00%	97.00%
Inception-v3	96.00%	95.63%	96.00%	96.00%	96.00%
VGG-19	97.00%	96.54%	97.00%	97.00%	96.00%

Table-3: Comparison of performance using the Weizmann dataset

Model	VGG-16	Kumar et al.	Cai et al.	Han et al.	Feng et al.
Accuracy	96.95	95.69%	95.70%	90.00%	94.10%

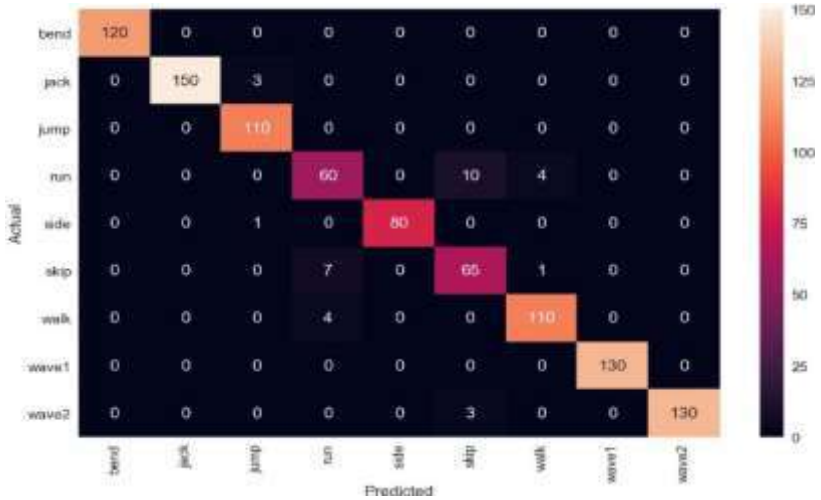


Fig. 2. Confusion Matrix for recognizing 9 activities with VGG-16 CNN on Weizmann Dataset

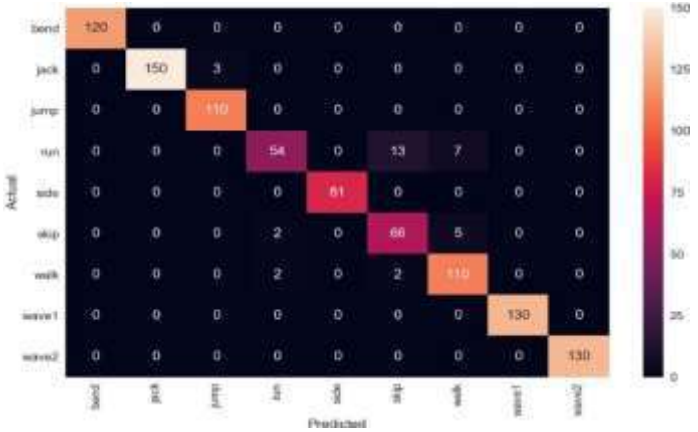


Fig. 3. Confusion Matrix for recognizing 9 activities with VGG-19 CNN on Weizmann Dataset

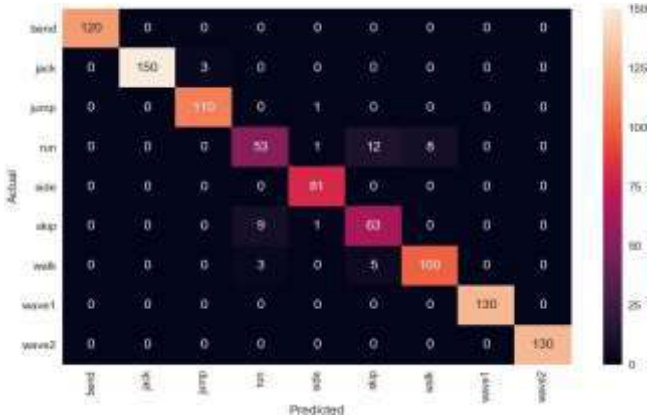


Fig. 4. Confusion Matrix for recognizing 9 activities with Inception-v3 CNN on Weizmann Dataset

Figures 2, 3, and 4 illustrate the confusion matrix of three distinct CNNs after transfer learning, which has been employed to categorize frames of various activities with Google's Inception Net-v3, VGG-16, and VGG-19, correspondingly. Figures 2, 3, and 4 show that VGG-16 has misclassification to forecast running activity as skip, VGG-19 has misclassification to forecast running activity as skip and skip as walk, and Google's InceptionNet-v3 has misclassification to forecast running activity as skip, all of which are visually comparable. The accuracy of activity identification has improved because of the use of transfer learning on CNN models. However, since ImageNet includes photos from different categories, the transfer learning approach employed in our research using information transferred from the pre-trained weight on ImageNet may be compromised.

5. CONCLUSION

This dataset was analyzed using a CNN model to predict human actions. Three CNNs were tested for activity recognition. To extract the deep image features as well as train machine learning classifiers, we have used transfer learning techniques. In our experiments, we found a 96.95 percent accuracy rate utilizing VGG-16 and transfer learning. In our experiments, VGG-16 performed better in comparison to CNN models to extract features. Additionally, we found that VGG-16 outperformed current best practices in our experiments using the transfer learning method.

6. REFERENCES

[1] B. Bhandari, J. Lu, X. Zheng, S. Rajasegarar, and C. Karmakar, "Noninvasive sensor-based automated smoking activity detection," in Proceedings of Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2017, pp. 845–848.

[2] L. Yao, Q. Z. Sheng, X. Li, T. Gu, M. Tan, X. Wang, S. Wang, and W. Ruan, "Compressive representation for device-free activity recognition with passive RFID signal strength," IEEE Transactions on Mobile Computing, vol. 17, no. 2, pp. 293–306, 2018.

[3] Lillo, J. C. Niebles, and A. Soto, "Sparse composition of body poses and atomic actions for human activity recognition in RGB-d videos," Image and Vision Computing, vol. 59, pp. 63–75, 2017.

[4] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton-based action recognition using regularized deep LSTM networks," in ThirtiethAAAI Conference on Artificial Intelligence, 2016.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[6] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June 2009, pp. 248–255.

[7] S. Deep and X. Zheng, "Leveraging CNN and Transfer Learning for Vision-based Human Activity Recognition," *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*, 2019, pp. 1-4, DOI: 10.1109/ITNAC46935.2019.9078016.

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

CERTIFICATE OF PUBLICATION

This is to certify that

B. Manish Kumar

published a paper entitled

Vison-based human activity recognition

Volume-8, Issue-2 - March-April, 2022

ISSN:2454-132X
IMPACT FACTOR 6.078

IJARIIT

Ref No....OAP/IJ/222/120 (V8I2-1320)

editor@ijariit.com, ijariit@gmail.com

Editor In Chief

Google Scholar · Cosmos · DRJI · RESEARCHERID · THOMSON REUTERS · IndianScience

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

CERTIFICATE OF PUBLICATION

This is to certify that

P Rohit

published a paper entitled

Vison-based human activity recognition

Volume-8, Issue-2 - March-April, 2022

ISSN:2454-132X
IMPACT FACTOR 6.078

Ref No.....OAP/I1/222/121 (V8I2-1320)

editor@ijariit.com, ijariit@gmail.com

Editor In Chief

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

CERTIFICATE OF PUBLICATION

This is to certify that

V. Sri Ram Reddy

published a paper entitled

Vison-based human activity recognition

Volume-8, Issue-2 - March-April, 2022

ISSN:2454-132X
IMPACT FACTOR 6.078

IJARIIT

Ref No....OAP/IJ/222/122 (V8I2-1320)

editor@ijariit.com, ijariit@gmail.com

Editor In Chief

Google Scholar · Cosmos · DRJI · RESEARCHERID · THOMSON REUTERS · IndianScience.in

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

CERTIFICATE OF PUBLICATION

This is to certify that

V. Sai Chandana

published a paper entitled

Vison-based human activity recognition

Volume-8, Issue-2 - March-April, 2022

ISSN:2454-132X
IMPACT FACTOR 6.078

Ref No....OAP/IJ/222/123 (V8I2-1320)

editor@ijariit.com, ijariit@gmail.com

Editor In Chief