A Project Report

on

LEVERAGING CNN AND TRANSFER LEARNING FOR VISION-BASED HUMAN ACTIVITY RECOGNITION

Dissertation Submitted in partial fulfilment of requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

CHARMISHA TAMMANA

20U91A0519

Under the Esteemed Guidance of

Ms. P. Swathi M.tech

Asst. Professor, Dept. of CSE

SRI MITTAPALLI COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi and Affiliated to JNTU Kakinada)

(An ISO 9001:2005 Certified institution and Accredited by NAAC & NBA)

TUMMALAPALEM, NH-5, GUNTUR, 522233, A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(2020-2024)

SRI MITTAPALLI COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi and Affiliated to JNTU Kakinada)

(An ISO 9001:2005 Certified institution and Accredited by NAAC & NBA)

TUMMALAPALEM, NH-5, GUNTUR, 522233, A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the thesis entitled "**LEVERAGING CNN AND TRANSFER LEARNING FOR VISION-BASED HUMAN ACTIVITY RECOGNITION**" **CHARMISHA TAMMANA (20U91A0519)** submitted in partial fulfilment of requirements for award of BACHELOR OF TECHNOLOGY Degree in Computer Science and Engineering by JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA is an authentic work carried out by her under my supervision and guidance in the year 2023-2024.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
Ms. P. SWATHI M.tech	**Dr. S. GOPI KRISHNA M.tech, ph. D**
Asst. Professor	**Professor & HOD**

EXTERNAL EXMINER

ACKNOWLEDGEMENT

We would like to express our utmost gratitude to our Chairman **Sri.M.V. Koteswara Rao** and Secretary **Sri. M.B.V. Satyanarayana** for providing their support and stimulating environment for the development of our project.

Furthermore, we are deeply grateful to **Dr. S. Gopi Krishna M.Tech, Ph.D**, the Principal of Sri Mittapalli College of Engineering, for their assistance in providing us with the necessary resources and support.

We would like to express our sincere gratitude to our Head of Department (HOD), **Dr. S. Gopi Krishna M. Tech, Ph.D.**, for their guidance and support throughout the course of my final year project. Their valuable suggestions and feedback have been instrumental in shaping the direction of my research and helping me to achieve my academic goals.

We are also immensely thankful to our guide **Miss. P. SWATHI M. Tech**, for her moral support and guidance throughout the project. Our sincere thanks also go to all the teaching and non- teaching staff for their constant support and advice. Lastly, we would like to thank our friends who have helped us in the successful completion of this project, either directly or indirectly.

CHARMISHA TAMMANA

20U91A0519

DECLARATION

I, **T. CHARMISHA** declare that the contents of this project, in full or part, have not been submitted to any other university or institution for the award of any degree or diploma.

We also declare that we have adhered to all principles of academic honest and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

We understand that any violation of the above will cause for disciplinary action by the institution and can also evoke penal action for the sources which have not been properly cited or from whom proper permission has not been taken when needed.

CHARMISHA TAMMANA

20U91A0519

DATE:

PLACE:

INDEX

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 Dataset 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.

CHAPTER-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 categorised 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. Recognising human activities from the streaming video is challenging. Video-based human activity recognition can be categorised as marker-based and vision-based according to motion features. Marker-based method make use of optic wearable marker based 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 recognise human activities from Weizmann Dataset. We first extracted the frames for each activity 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 dataset.

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 dataset such as ImageNet to the human activity recognition dataset.

There have been a lot of research 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. Therefore, video-based human activity recognition using deep learning models have gained a lot of interest in recent years. Zhu et al. 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 correlation 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. Wang et al. 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. Ji et al. proposed a 3D CNN model which perform s 3D convolutions and extract spatial and temporal features by capturing the motion information for activity recognition. Simonyan et al. introduced ConvNet, a two-stream convolution layer architecture that could achieve good results despite of limited training data. Khaire et al. proposed a model that train convnets from RGB-D dataset and combined the soft max scores from depth, motion and skeleton images at the classification level to identify the activities. Khaire et al. proposed the extension of CNN architecture in the first convolutional layers over a 40-video chunk. Similarly, Tran et al. used a deep 3D CNN architecture (quite similar to VGGnet) that utilise 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.

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 behavioural 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 behavioural symptoms in PwD. We evaluate various com bination of deep CNN features using SVM.

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

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

Human action and behaviour 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. 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, colour 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. This suggests the layers of deep models do indeed learn features that are fairly general. Automatic monitoring of the behavioural symptoms is often based on wearable sensors. In, Chikhaoui et al.

CHAPTER-2
LITERATURE SURVEY

2. LITERATURE SURVEY

1) Noninvasive sensor based automated smoking activity detection

AUTHORS: B. Bhandari, J. Lu, X. Zheng, S. Rajasekar, 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) Compressive representation for device-free activity recognition with passive rf id 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.

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.

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.

5) A Vision-Based Transfer Learning Approach For Recognizing Behavioural 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, behaviour and the ability to perform activities of daily living. Depression and aggressive behaviour are the most upsetting and challenging symptoms of dementia. Automatic recognition of these behaviours 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 behavioural 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.

CHAPTER-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**: Marker based 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 Oxford Net)

CHAPTER-4
SOFTWARE REQUIREMENTS

4. SYSTEM REQUIREMENTS SPECIFICATIONS

4.1. SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

➢ **System**	**:**	Intel i3
➢ **Hard Disk**	**:**	1 TB.
➢ **Monitor**	**:**	14' Colour Monitor.
➢ **Mouse**	**:**	Optical Mouse.
➢ **Ram**	**:**	4GB.

SOFTWARE REQUIREMENTS:

➢ **Operating system**	**:**	Windows 10.
➢ **Coding Language**	**:**	Python.

4.2. SYSTEM STUDY 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**

4.2.1. 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.

4.2.2TECHNICAL 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.

4.2.3. 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.

4.3. Input/Output Design

4.3.1. 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

1.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.

2. 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.

3.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

4.3.2. 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.

1. 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.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

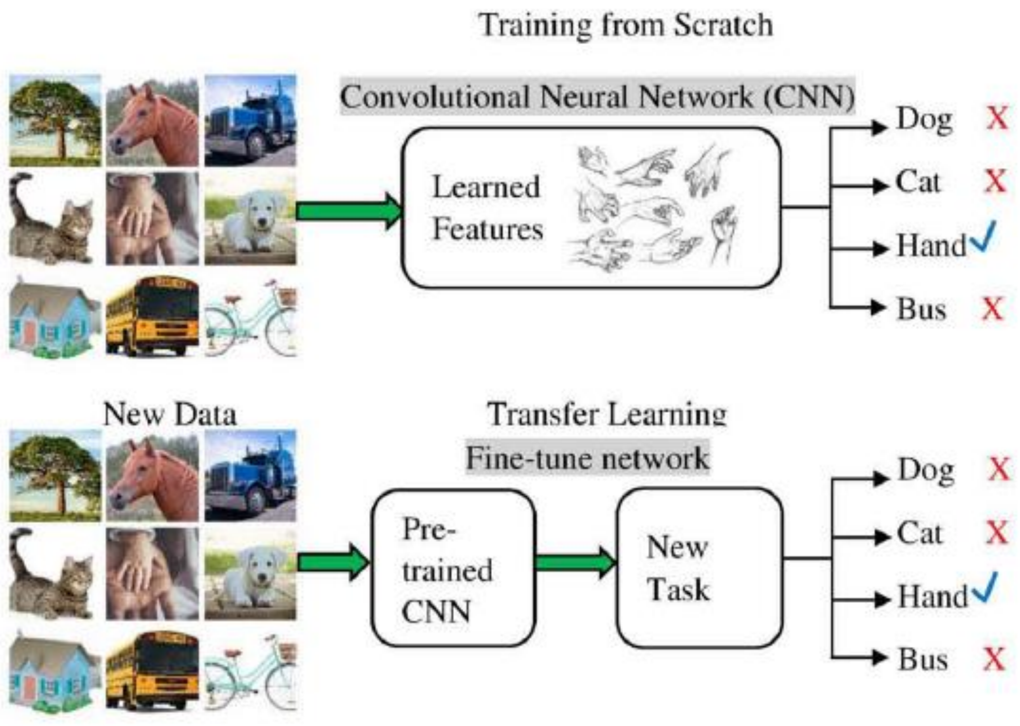
The output form of an information system should accomplish one or more of the following objectives.

- 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.

CHAPTER-5
SYSTEM DESIGN

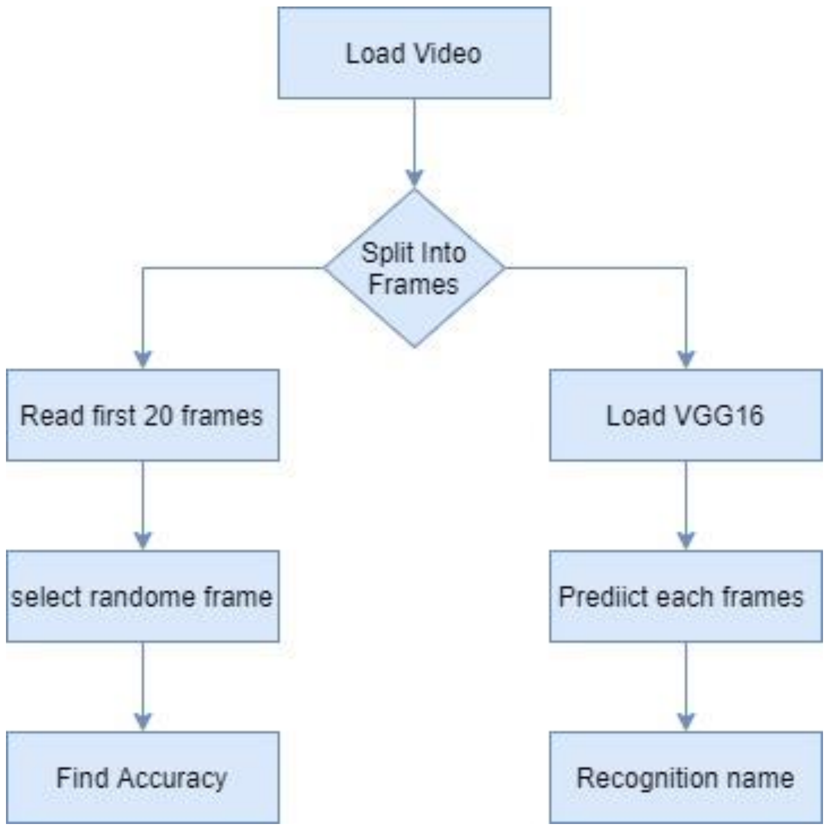
5. SYSTEM DESIGN

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.

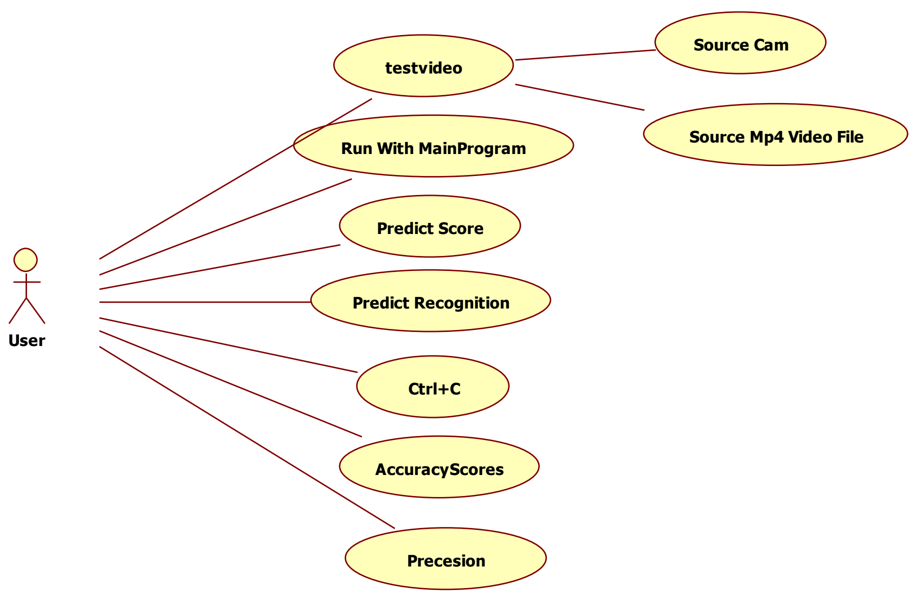
GOALS:

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 extendibility 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.
7. **7.** Integrate best practices.

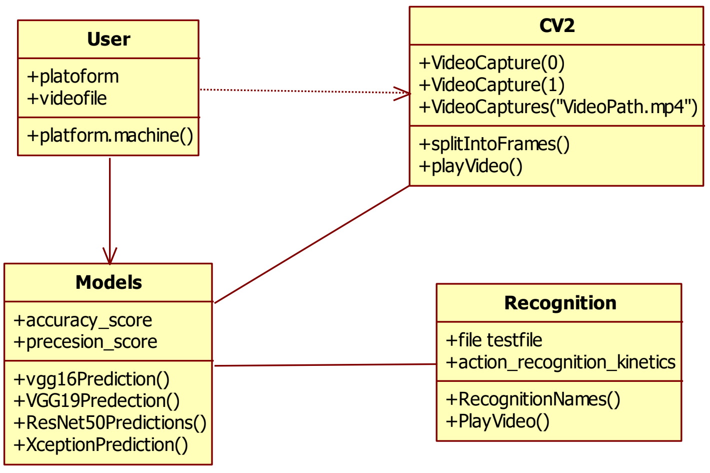
5.3.1. USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural 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.



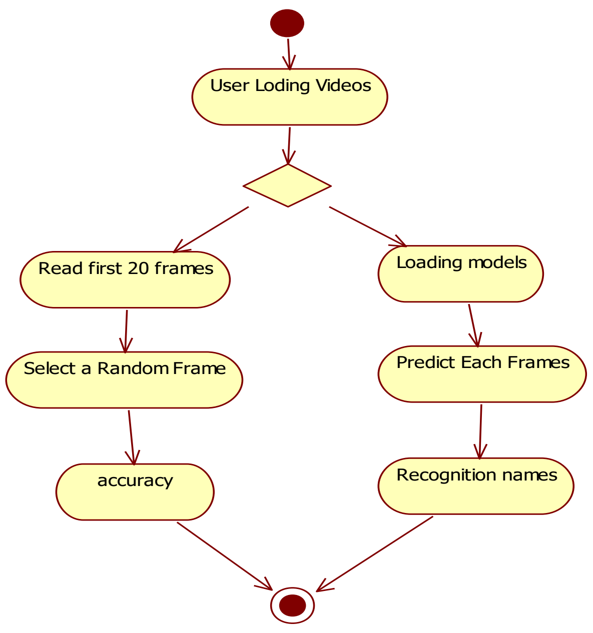
5.3.2. CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modelling 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.



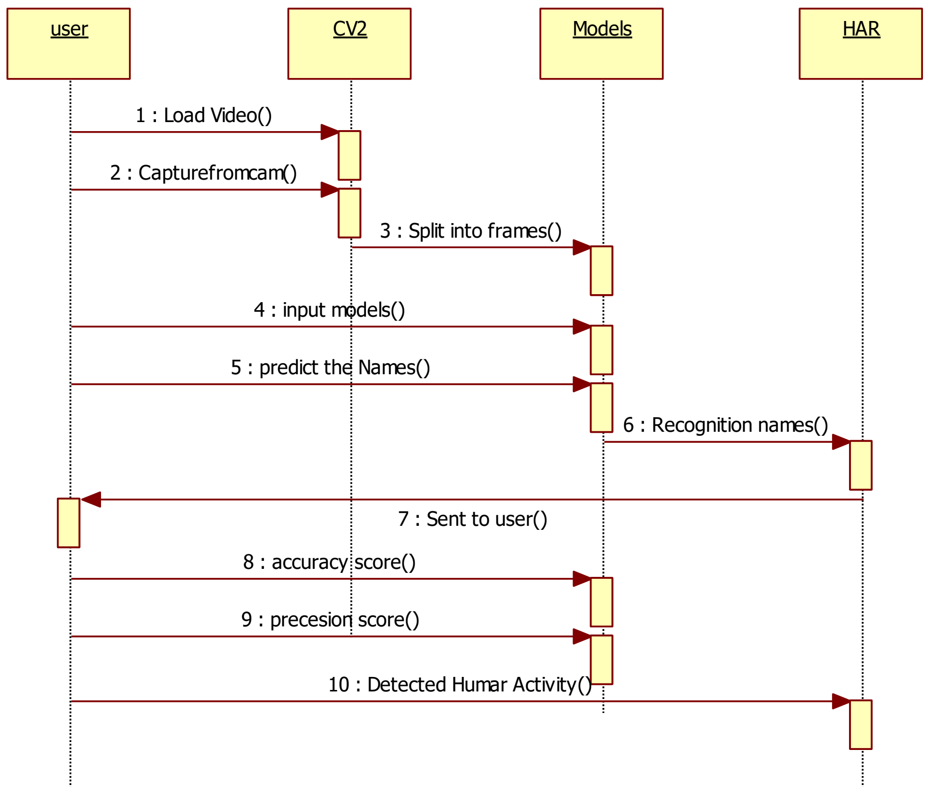
5.3.3. ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling 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.



5.3.4. SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling 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.



CHAPTER-6

IMPLEMENTATION

6. Implementation

MODULES:

- **User**
- **HAR System**
- **VGG16**
- **Transfer Learning**

6.1. MODULES DESCRIPTION:

- **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 without camera we can load the video file from the disk. Vgg16, Vgg19 has programmatically configured. User can change the model selection in the code and can run in multiple ways.

- **HAR System:** Video-based human activity recognition can be categorised 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.

- **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 models submitted to ILSVRC-2014. It makes the improvement over Alex Net 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.

- **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. In this post, you will discover how you can use transfer learning to speed up training and improve the performance of your deep learning model.

CHAPTER-7

TECHNOLOGY DESCRIPTION

7. TECHNNOLOGY DESCRIPTION

TECHNOLOGIES USED

For a project on leveraging Convolutional Neural Networks (CNNs) and transfer learning for vision-based human activity recognition, several technologies and tools are commonly used:

7.1. Python: Python is the primary programming language used for implementing deep learning models due to its extensive libraries and frameworks.

Python is a high-level, general-purpose, and interpreted programming language used in various sectors including machine learning, artificial intelligence, data analysis, web development, and many more. Python is known for its ease of use, powerful standard library, and dynamic semantics. It also has a large community of developers who keep on contributing towards its growth. The major focus behind creating it is making it easier for developers to read and understand, also reducing the lines of code.

Features of Python

Python has plenty of features that make it the most demanding and more popular. Let's read about a few of the best features that Python has:

→Easy to read and understand

→Interpreted language

→Object-oriented programming language

→Free and open-source

→Versatile and Extensible

→Multi-platform

→Hundreds of libraries and frameworks

→Flexible, supports GUI

→Dynamically typed

→Huge and active community

7.2 Deep Learning Frameworks:

- **TensorFlow**:

TensorFlow can be used to develop models for various tasks, including natural language processing, image recognition, handwriting recognition, and different computational-based simulations such as partial differential equations.

TensorFlow is a popular framework of machine learning and deep learning. It is a free and open-source library which is released on 9 November 2015 and developed by Google Brain Team. It is entirely based on Python programming language and use for numerical computation and data flow, which makes machine learning faster and easier.

TensorFlow can train and run the deep neural networks for image recognition, handwritten digit classification, recurrent neural network, word embedding, natural language processing, video detection, and many more. TensorFlow is run on multiple CPUs or GPUs and also mobile operating systems.

TensorFlow is the better library for all because it is accessible to everyone. TensorFlow library integrates different API to create a scale deep learning architecture like CNN (Convolutional Neural Network) or RNN (Recurrent Neural Network).

It is mainly used for deep learning or machine learning problems such as Classification, Perception, Understanding, Discovering Prediction, and Creation.

1) Voice/Sound Recognition
2) Image Recognition
3) Time Series
4) Video Detection
5) Text-Based Applications

- **Keras:**

Keras is a deep learning API written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch. Keras is: Simple – but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter. Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

Keras is relatively easy to learn and work with because it provides a python frontend with a high level of abstraction while having the option of multiple back-ends for computation purposes. This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly.

Keras allows you to switch between different back ends. The frameworks supported by Keras are:

1) TensorFlow
2) Theano
3) PlaidML
4) MXNet
5) CNTK (Microsoft Cognitive Toolkit)

Out of these five frameworks, TensorFlow has adopted Keras as its official high-level API. Keras is embedded in TensorFlow and can be used to perform deep learning fast as it provides inbuilt modules for all neural network computations. At the same time, computation involving tensors, computation graphs, sessions, etc can be custom made using the Tensor flow Core API, which gives you total flexibility and control over your application and lets you implement your ideas in a relatively short time.

Overall, Keras is a powerful and user-friendly tool for building, training, and deploying deep learning models. Its simplicity, flexibility, and extensive ecosystem make it an excellent choice for both beginners and experienced researchers and practitioners in the field of artificial intelligence and machine learning.

- **OpenCV:**

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. Originally developed by Intel, it was later maintained by Willow Garage and is now maintained by Itseez, which was later acquired by Intel. OpenCV is written in C++ and has interfaces for Python, Java, and MATLAB/Octave, making it widely accessible across different programming languages and platforms.

Here are some key aspects and features of OpenCV in Python:

➢ **Image Processing**: OpenCV provides a wide range of functionalities for image processing, including reading and writing images in various formats, resizing, cropping, rotating, and transforming images, adjusting brightness, contrast, and color balance, applying filters and convolution operations, and performing morphological operations such as erosion and dilation.

➢ **Computer Vision Algorithms:** OpenCV includes implementations of various computer vision algorithms and techniques, such as feature detection ,feature matching ,object detection,object tracking camera calibration, stereo vision, and 3D reconstruction.

- **Deep Learning Integration:** OpenCV provides integration with deep learning frameworks such as TensorFlow, PyTorch, and Caffe through its Deep Neural Networks (DNN) module. This allows users to perform inference with pre-trained deep learning models directly within OpenCV, enabling tasks such as image classification, object detection, semantic segmentation, and image generation.
- **Video Processing:** OpenCV supports video processing capabilities, including reading and writing video files, capturing frames from cameras, applying real-time video effects, performing background subtraction, optical flow estimation, object tracking in videos, and video stabilization.
- **GUI Tools:** OpenCV provides graphical user interface (GUI) tools for visualizing and interacting with images and videos, such as displaying images and videos in windows, drawing shapes and text on images, handling mouse and keyboard events, creating trackbars and buttons for parameter adjustment, and saving screenshots and video frames.

Overall, OpenCV is a powerful and versatile library for computer vision and image processing tasks in Python, suitable for a wide range of applications, including robotics, augmented reality, surveillance, medical imaging, and more. Its extensive documentation, large community, and active development make it a popular choice for both researchers and practitioners in the field.

7.3. Integration with ONNX Model:

Integration with an ONNX model refers to the usage of a deep learning model that has been trained and saved in the Open Neural Network Exchange (ONNX) format. ONNX is an open-source format designed to represent deep learning models in a standardized way, enabling interoperability between different deep learning frameworks such as PyTorch, TensorFlow, and others.

- **Loading the ONNX Model**: The code uses the cv2.dnn.readNet() function to load a pre-trained human activity recognition model specified by the --model argument. This function is capable of loading models saved in various formats, including ONNX.
- **Model Inference:** Once the ONNX model is loaded, it can be used to perform inference on input data. In this case, the input data consists of video frames captured from a video stream or file. The frames are processed and passed through the ONNX model to obtain predictions about human activity recognition.
- **Interoperability**: By using the ONNX format, the model trained in any deep learning framework that supports ONNX export can be used seamlessly with OpenCV's DNN

module. This provides flexibility and interoperability, allowing developers to leverage pre-trained models trained in different frameworks without the need for additional conversion or compatibility issues.

Overall, integration with an ONNX model enables the code to utilize a pre-trained deep learning model for human activity recognition, providing efficient and accurate inference capabilities within the OpenCV environment

Subprocess Module:

The subprocess module present in Python (both 2.x and 3.x) is used to run new applications or programs through Python code by creating new processes. It also helps to obtain the input/output/error pipes as well as the exit codes of various commands. In this tutorial, we'll delve into how to effectively use sub processing modules to execute external functions from your Python code, pass data to them, and get results

Uses of Subprocess Module:

1) Run system commands or external programs from your Python script.
2) They can handle input, output, and errors caused by child programs in your Python code

7.4. Commonly Used Libraries:

Usage of NumPy: NumPy, a fundamental package for numerical computing in Python, is utilized in the code for array manipulation and operations. For example, NumPy arrays are used to handle blob data generated by the DNN module and to perform array operations like transpose and expansion.

- **Argparse:** The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv . The argparse module also automatically generates help and usage messages.
- **Immutils:** A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much easier with OpenCV and both Python 2.7 and Python 3.
- **SYS:** The sys module provides a number of functions and variables that can be used to manipulate different parts of the Python runtime environment.

- **Shutil:** This method is used to replicate the complete directory. It copies an entire directory tree rooted at source to the destination directory. The destination directory must not already present.
- **SubProcess:** The subprocess is a standard Python module designed to start new processes from within a Python script. It's very helpful, and, in fact, it's the recommended option when you need to run multiple processes in parallel or call an external program or external command from inside your Python code.
- **Random:** The Python Random module is a built-in module for generating random integers in Python. These numbers occur randomly and does not follow any rules or instructuctions. We can therefore use this module to generate random numbers, display a random item for a list or string, and ect.
- **Data Augmentation Libraries:** Libraries like imgaug or built-in augmentation modules in frameworks like TensorFlow and PyTorch are used to artificially increase the size of the training dataset by applying transformations such as rotation, scaling, flipping, etc., to the images.
- **GPU Acceleration:** Deep learning models, especially CNNs, are computationally intensive and training them can be significantly accelerated using GPUs. Technologies like NVIDIA CUDA or cloud-based GPU services (e.g., Google Colab, AWS EC2 with GPU instances) are commonly employed for this purpose.
- **Data Collection Tools:** Tools for collecting and labelling human activity datasets are essential. These may include smartphone apps, wearable devices, or specialized sensors.
- **Version Control:** Version control systems like Git are used for managing codebase changes, collaborating with team members, and tracking experimental results.
- **Visualization Tools:** Tools like TensorBoard(for TensorFlow) or matplotlib are used for visualizing model architectures, training/validation curves, and other metrics.
- **Deployment Platforms:** Once the model is trained, it needs to be deployed for inference. Technologies like TensorFlow Serving, TensorFlow Lite, ONNX Runtime, or cloud-based platforms such as AWS SageMaker or Google Cloud AI Platform are used for deployment.

By leveraging these technologies, developers can effectively implement and deploy CNN-based models for vision-based human activity recognition tasks.

- **Pre-trained Models:** Utilizing pre-trained CNN models for transfer learning is common in vision-based tasks. Models like VGG, ResNet, Inception, and MobileNet are often used as feature extractors.

CHAPTER-8
CODING

8. CODING

Python code of Main Run:

```
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)
```

Python code for 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
```

Python code for 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)
```

Python code for Report Generation:

```
# 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)
```

Python code for Test LSTM Model:

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_'+gr
oup+'.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()
```

CHAPTER-9
SYSTEM TESTING

9. SYSTEM TESTING

What do you mean by software testing?

Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should. It is oriented to 'detection'.

9.1. TESTING METHODOLIGES

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

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.

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.

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 centred 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 class 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.

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.

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.

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.

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.

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.

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.

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.

Sample 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 won't 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 won't 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 won't work

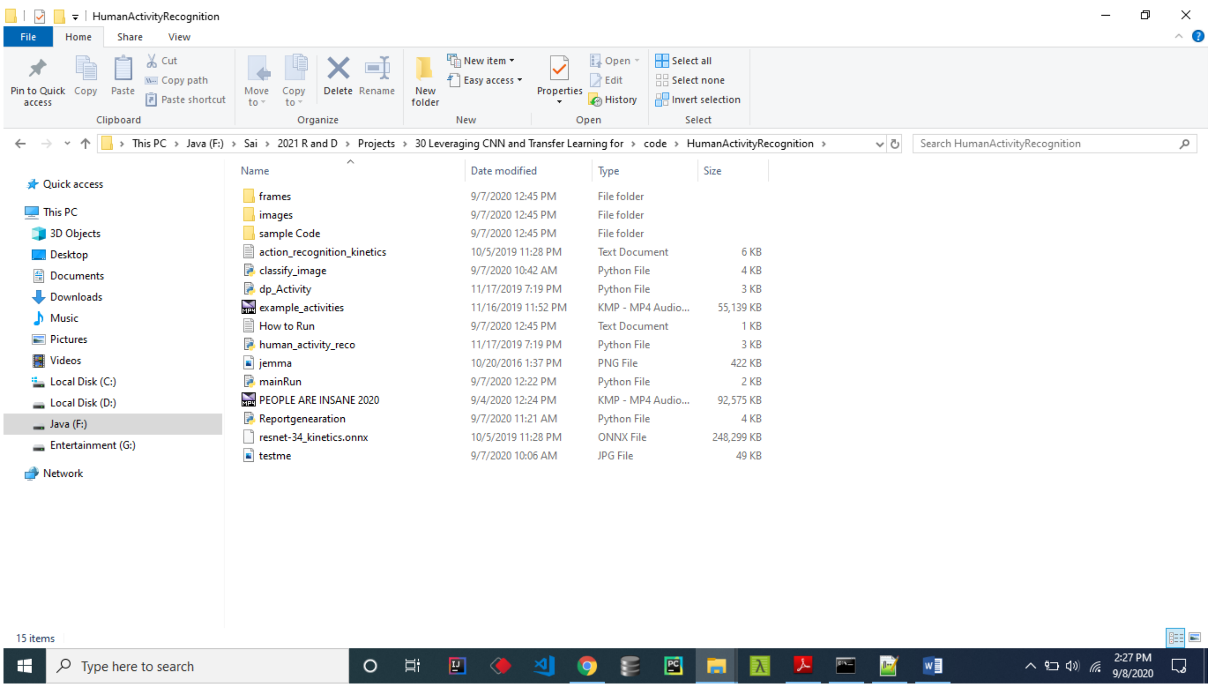
CHAPTER-10

OUTPUT (SCREEN SHOTS & REPORTS)

10. OUTPUT (SCREEN SHOTS & REPORTS)

10.1 Execution Screen Shots

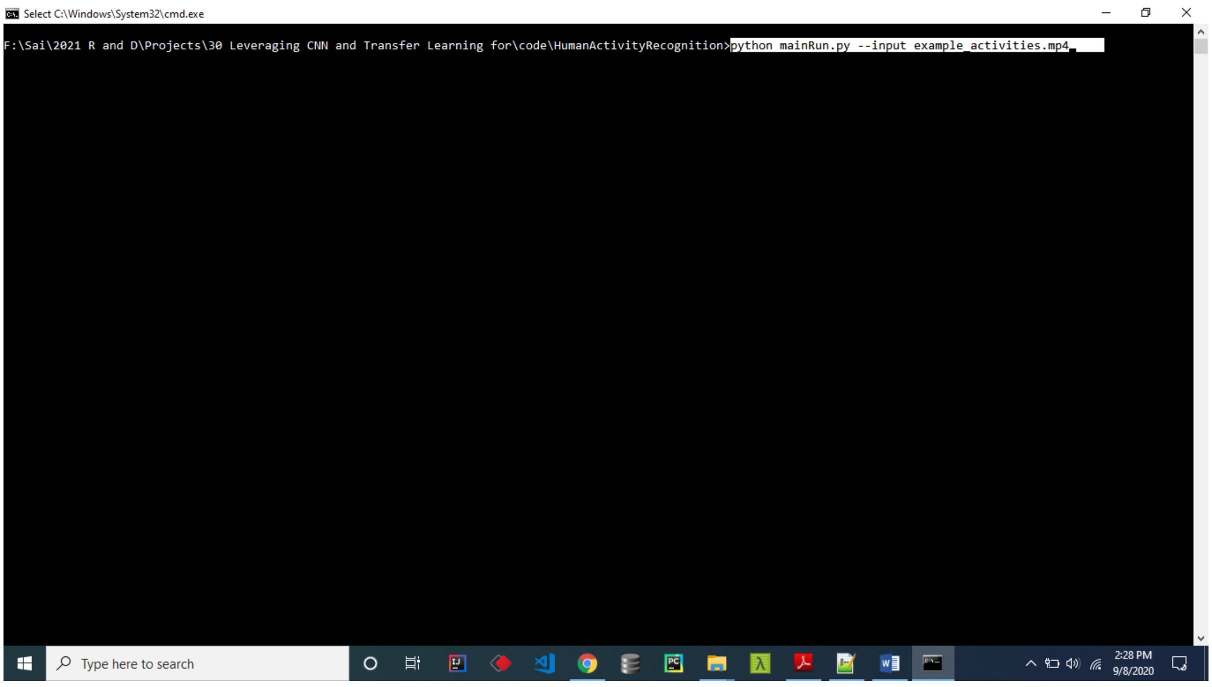
Starting Project:



This PC › Java (F:) › Sai › 2021 R and D › Projects › 30 Leveraging CNN and Transfer Learning for › code › HumanActivityRecognition ›

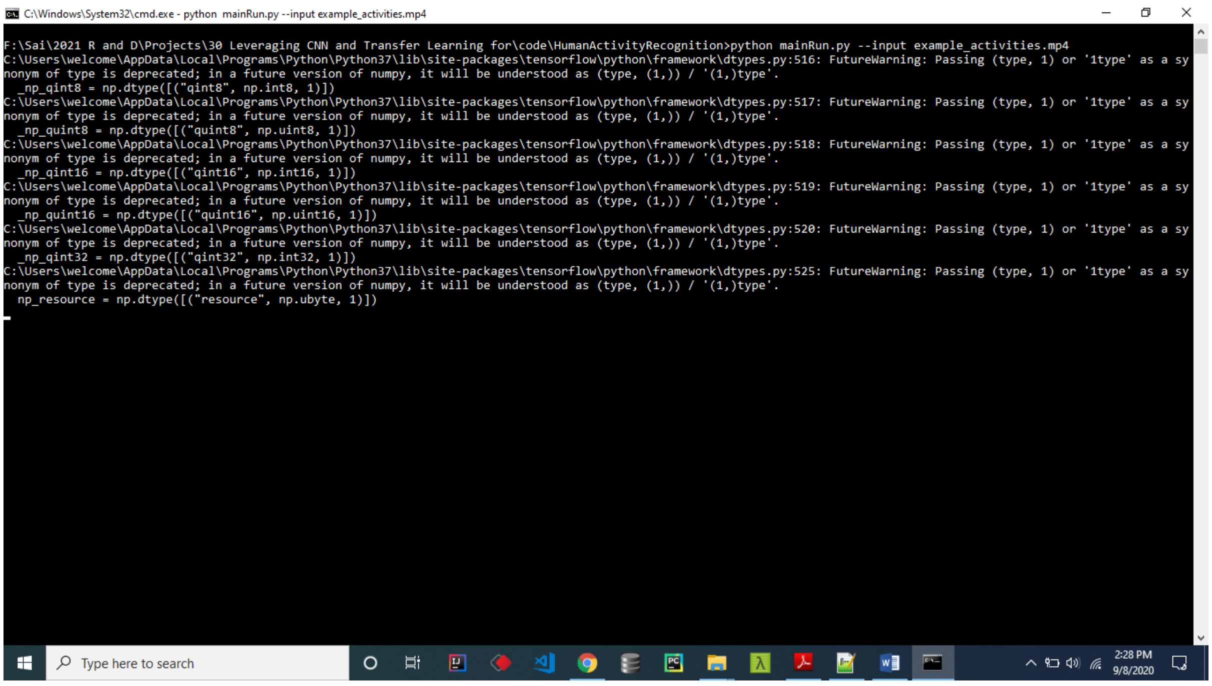
Name	Date modified	Type	Size
frames	9/7/2020 12:45 PM	File folder	
images	9/7/2020 12:45 PM	File folder	
sample Code	9/7/2020 12:45 PM	File folder	
action_recognition_kinetics	10/5/2019 11:28 PM	Text Document	6 KB
classify_image	9/7/2020 10:42 AM	Python File	4 KB
dp_Activity	11/17/2019 7:19 PM	Python File	3 KB
example_activities	11/16/2019 11:52 PM	KMP - MP4 Audio...	55,139 KB
How to Run	9/7/2020 12:45 PM	Text Document	1 KB
human_activity_reco	11/17/2019 7:19 PM	Python File	3 KB
jemma	10/20/2016 1:37 PM	PNG File	422 KB
mainRun	9/7/2020 12:22 PM	Python File	2 KB
PEOPLE ARE INSANE 2020	9/4/2020 12:24 PM	KMP - MP4 Audio...	92,575 KB
Reportgenearation	9/7/2020 11:21 AM	Python File	4 KB
resnet-34_kinetics.onnx	10/5/2019 11:28 PM	ONNX File	248,299 KB
testme	9/7/2020 10:06 AM	JPG File	49 KB

Run the main Program:



```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
```

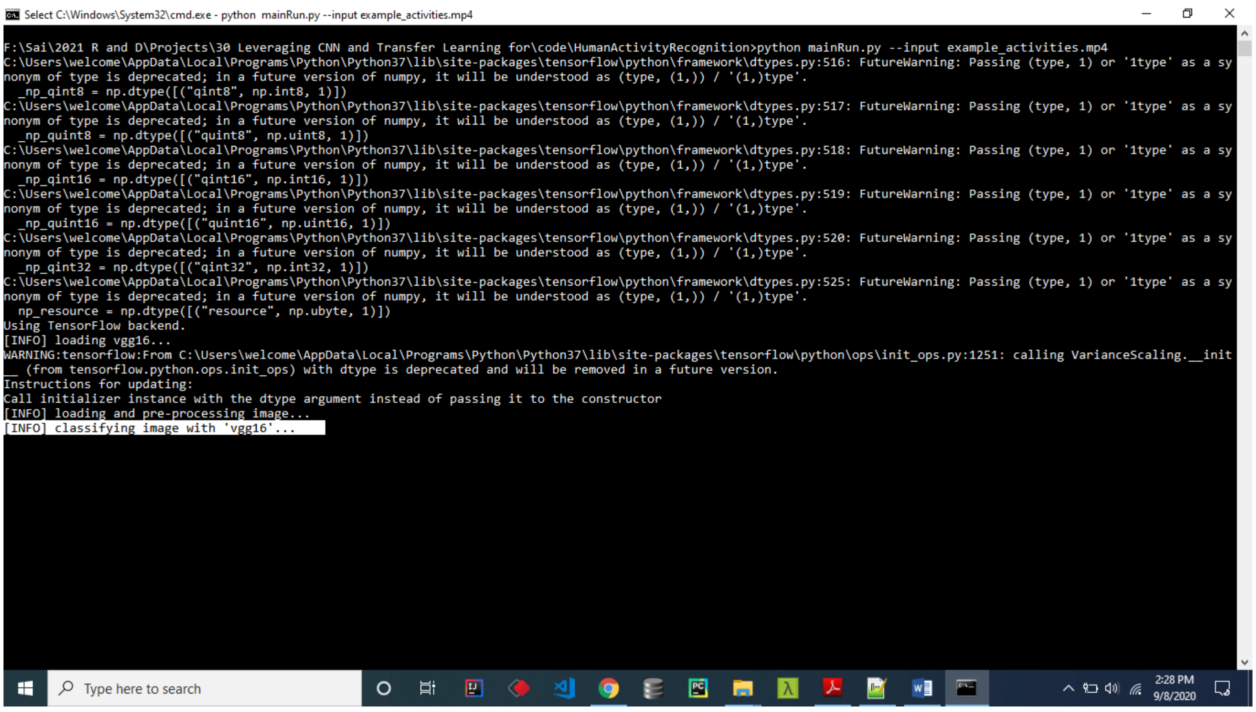
Loading Tensor flow Libraries:



C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4

```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

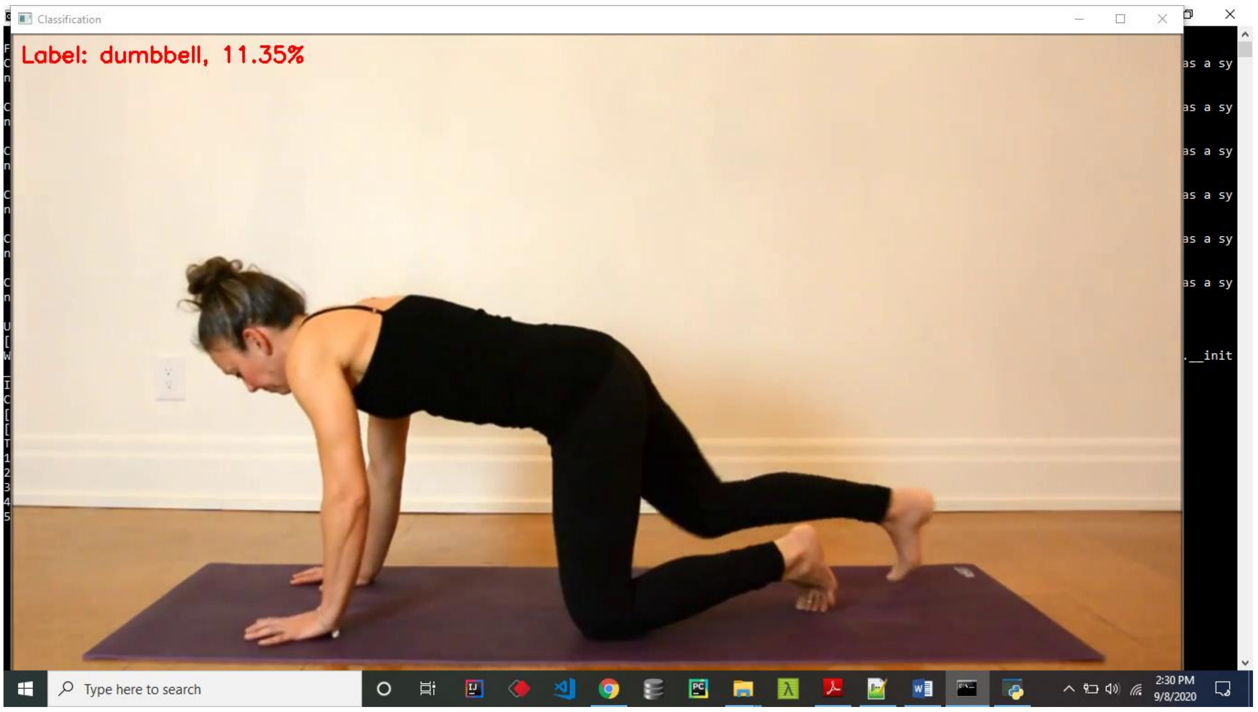
Classification with vgg16:



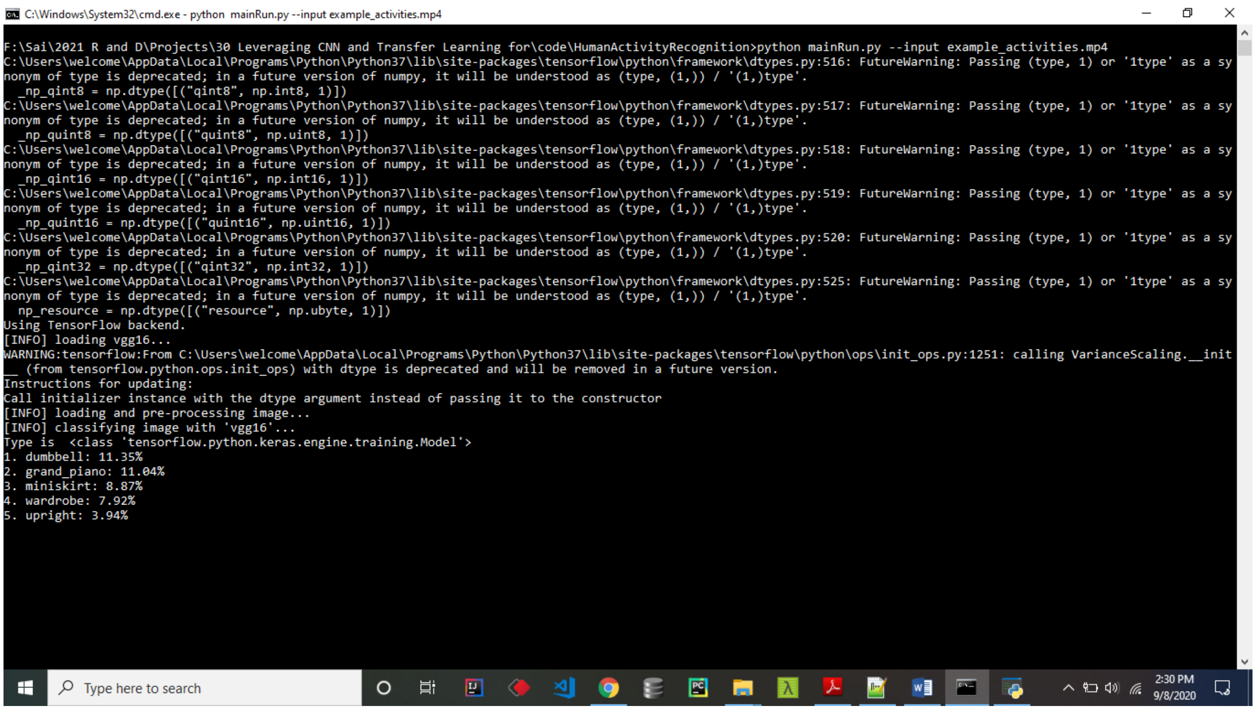
Select C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4

```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
[INFO] loading vgg16...
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init
__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
[INFO] loading and pre-processing image...
[INFO] classifying image with 'vgg16'...
```

Get Image label:

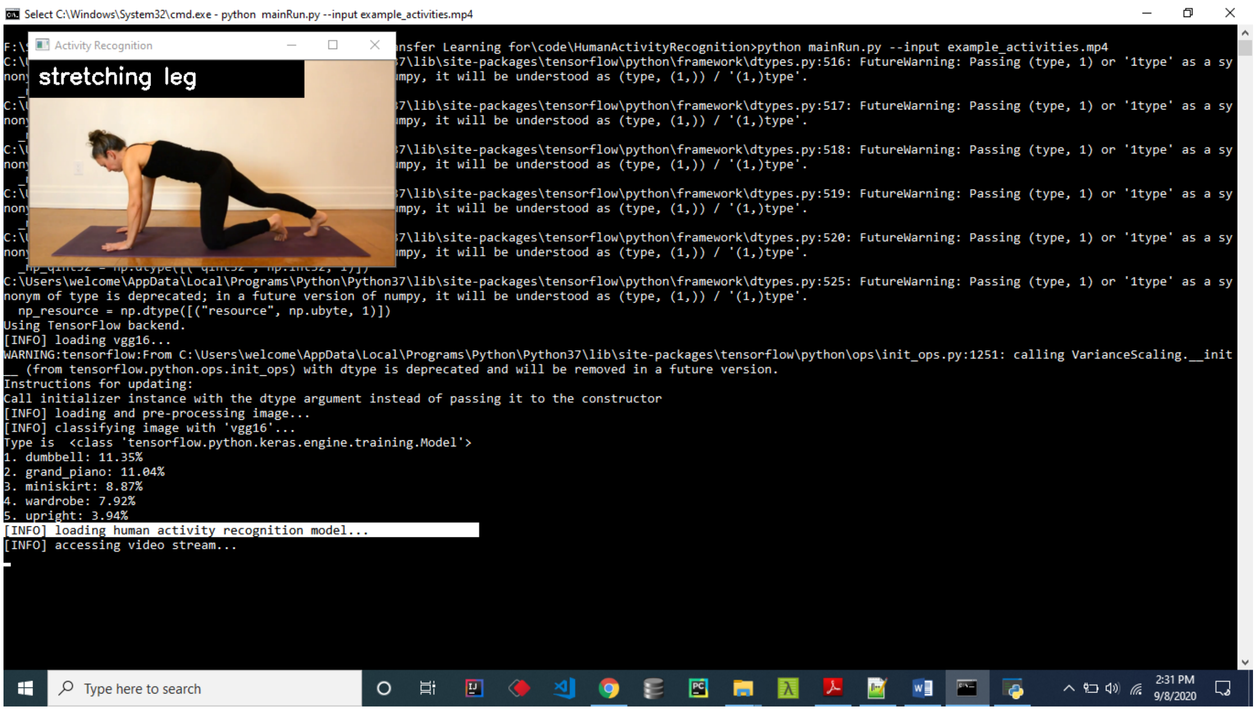


Result from image:


C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4
Type here to search
2:30 PM
9/8/2020

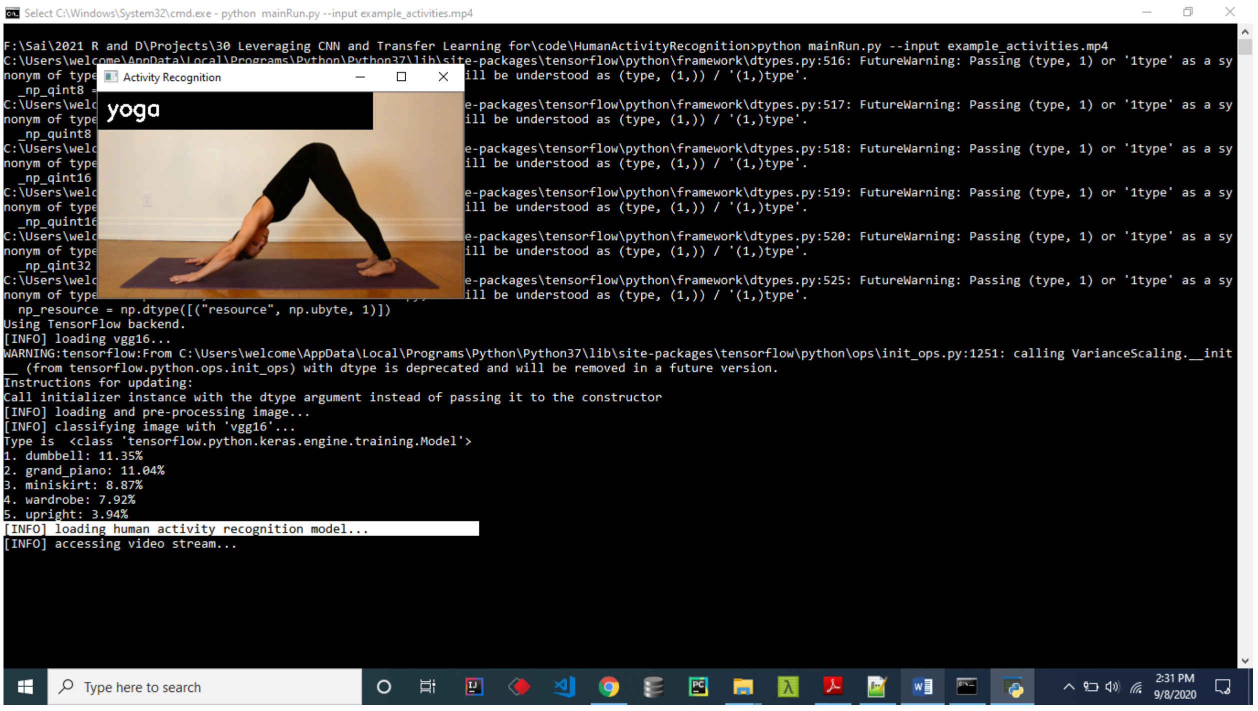
```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
[INFO] loading vgg16...
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init
__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
[INFO] loading and pre-processing image...
[INFO] classifying image with 'vgg16'...
Type is  <class 'tensorflow.python.keras.engine.training.Model'>
1. dumbbell: 11.35%
2. grand_piano: 11.04%
3. miniskirt: 8.87%
4. wardrobe: 7.92%
5. upright: 3.94%
```

Loading model HAR:


Select C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4
Activity Recognition
stretching leg
Type here to search
2:31 PM
9/8/2020

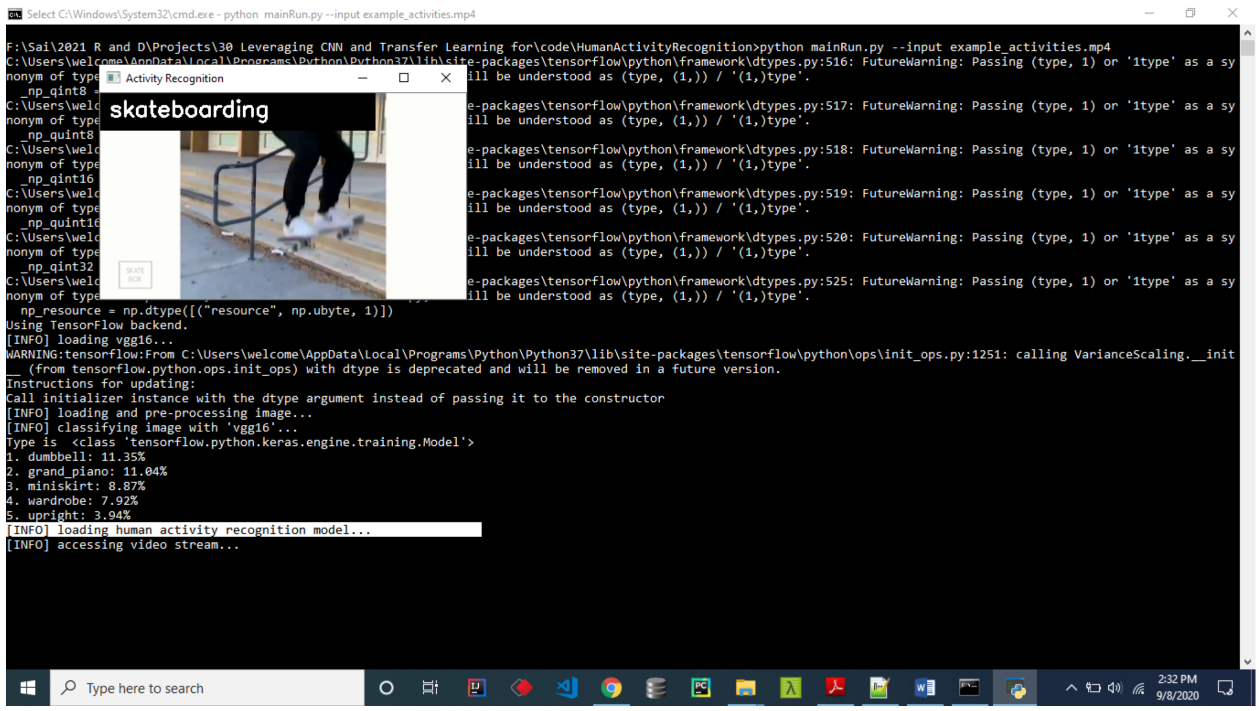
```
F:\                            nsfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\                            7\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
non                            mpy, it will be understood as (type, (1,)) / '(1,)type'.
C:\                            7\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
non                            mpy, it will be understood as (type, (1,)) / '(1,)type'.
C:\                            7\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
non                            mpy, it will be understood as (type, (1,)) / '(1,)type'.
C:\                            7\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
non                            mpy, it will be understood as (type, (1,)) / '(1,)type'.
C:\                            7\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
non                            mpy, it will be understood as (type, (1,)) / '(1,)type'.
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
[INFO] loading vgg16...
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init
__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
[INFO] loading and pre-processing image...
[INFO] classifying image with 'vgg16'...
Type is  <class 'tensorflow.python.keras.engine.training.Model'>
1. dumbbell: 11.35%
2. grand_piano: 11.04%
3. miniskirt: 8.87%
4. wardrobe: 7.92%
5. upright: 3.94%
[INFO] loading human activity recognition model...
[INFO] accessing video stream...
```

Result 1:


Select C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4
Activity Recognition
yoga
Type here to search
2:31 PM
9/8/2020

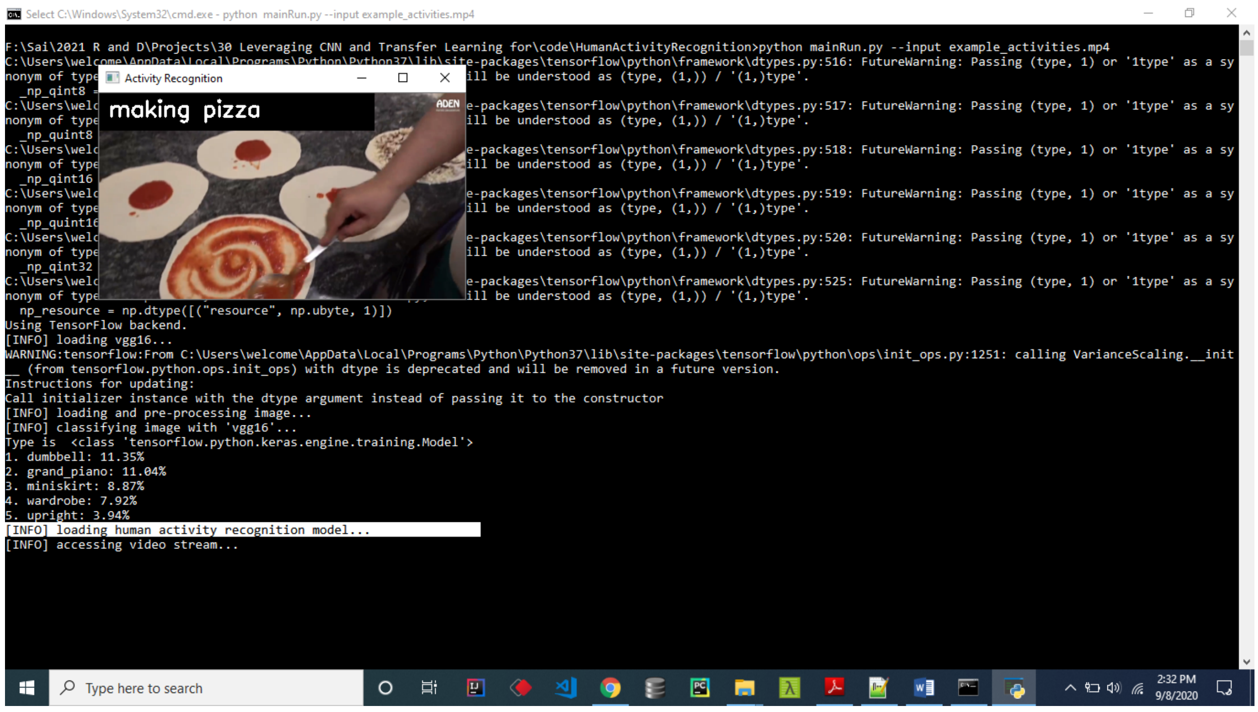
```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                          ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint8
C:\Users\welc                          e-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                          ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint8
C:\Users\welc                          e-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                          ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint16
C:\Users\welc                          e-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                          ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint16
C:\Users\welc                          e-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                          ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint32
C:\Users\welc                          e-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                          ill be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
[INFO] loading vgg16...
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init
__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
[INFO] loading and pre-processing image...
[INFO] classifying image with 'vgg16'...
Type is  <class 'tensorflow.python.keras.engine.training.Model'>
1. dumbbell: 11.35%
2. grand_piano: 11.04%
3. miniskirt: 8.87%
4. wardrobe: 7.92%
5. upright: 3.94%
[INFO] loading human activity recognition model...
[INFO] accessing video stream...
```

Result 2:


Select C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4
Activity Recognition
skateboarding
Type here to search
2:32 PM
9/8/2020

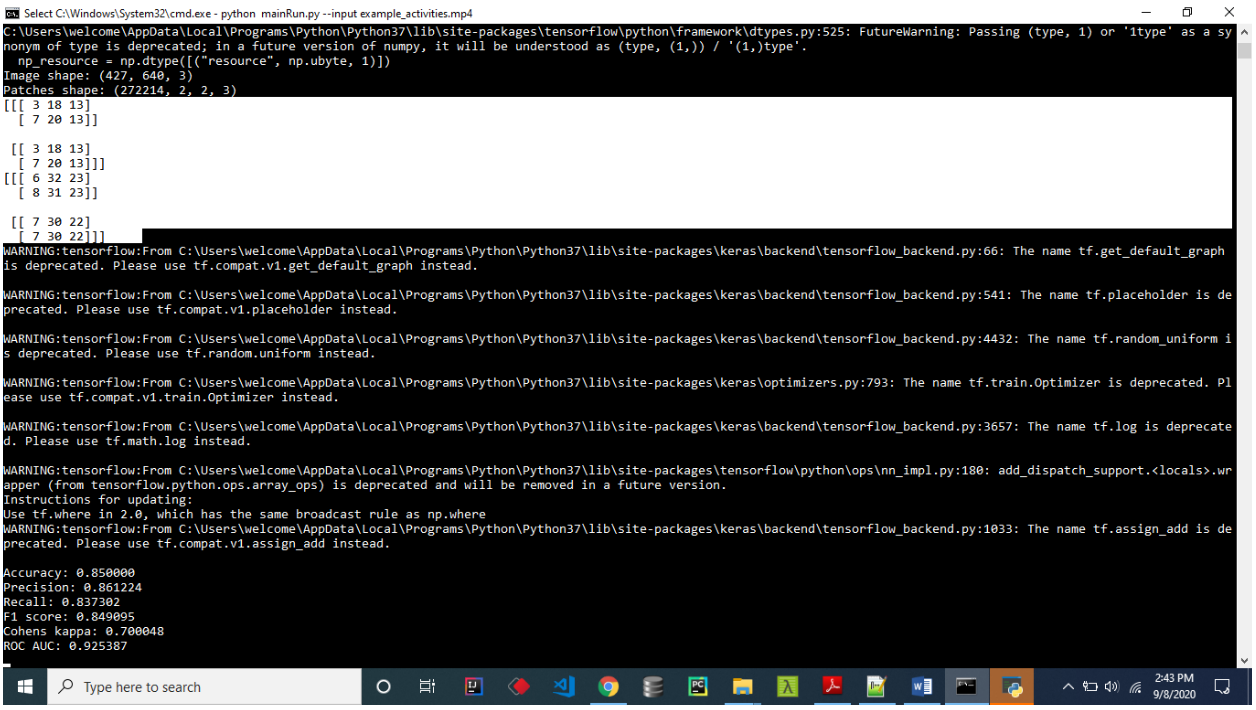
```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 =
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint8
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint16
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint16
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint32
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
[INFO] loading vgg16...
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init
__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
[INFO] loading and pre-processing image...
[INFO] classifying image with 'vgg16'...
Type is  <class 'tensorflow.python.keras.engine.training.Model'>
1. dumbbell: 11.35%
2. grand_piano: 11.04%
3. miniskirt: 8.87%
4. wardrobe: 7.92%
5. upright: 3.94%
[INFO] loading human activity recognition model...
[INFO] accessing video stream...
```

Result 3:


Select C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4
Activity Recognition
making pizza
Type here to search
2:32 PM
9/8/2020

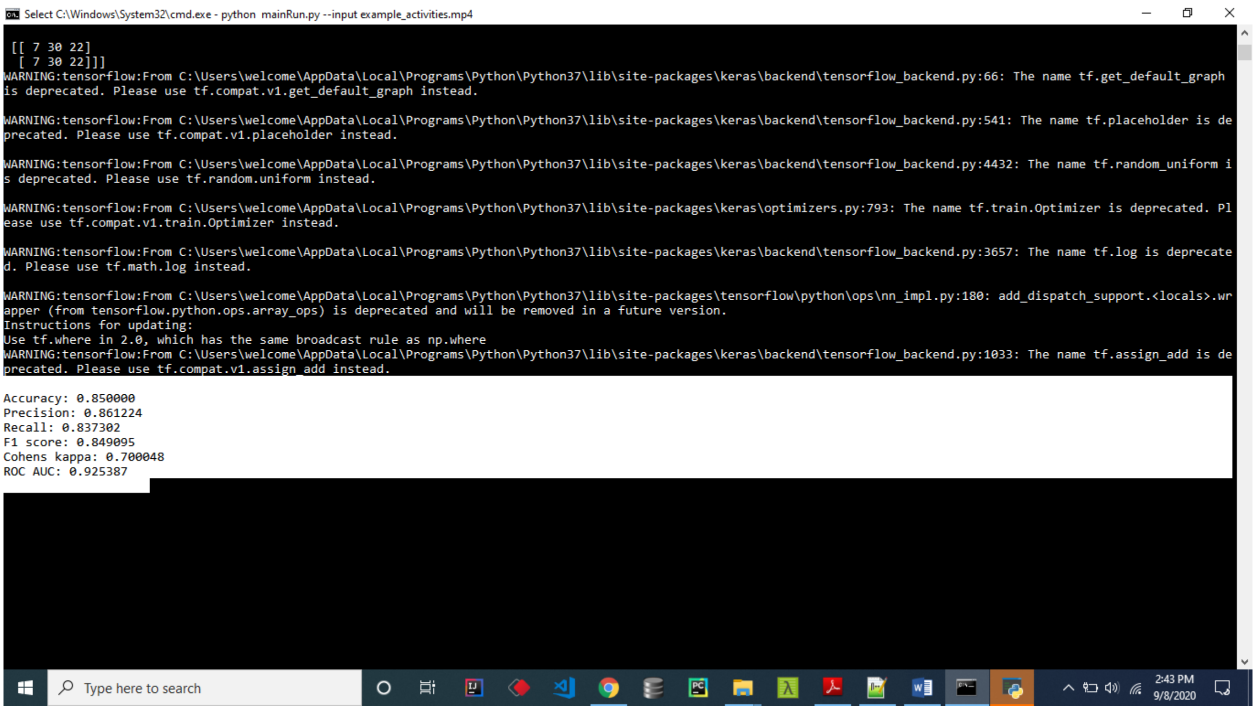
```
F:\Sai\2021 R and D\Projects\30 Leveraging CNN and Transfer Learning for\code\HumanActivityRecognition>python mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 =
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint8
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint16
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint16
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint32
C:\Users\welc                                                             e-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type                                                             ill be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
[INFO] loading vgg16...
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init
__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
[INFO] loading and pre-processing image...
[INFO] classifying image with 'vgg16'...
Type is  <class 'tensorflow.python.keras.engine.training.Model'>
1. dumbbell: 11.35%
2. grand_piano: 11.04%
3. miniskirt: 8.87%
4. wardrobe: 7.92%
5. upright: 3.94%
[INFO] loading human activity recognition model...
[INFO] accessing video stream...
```

Patches from image:



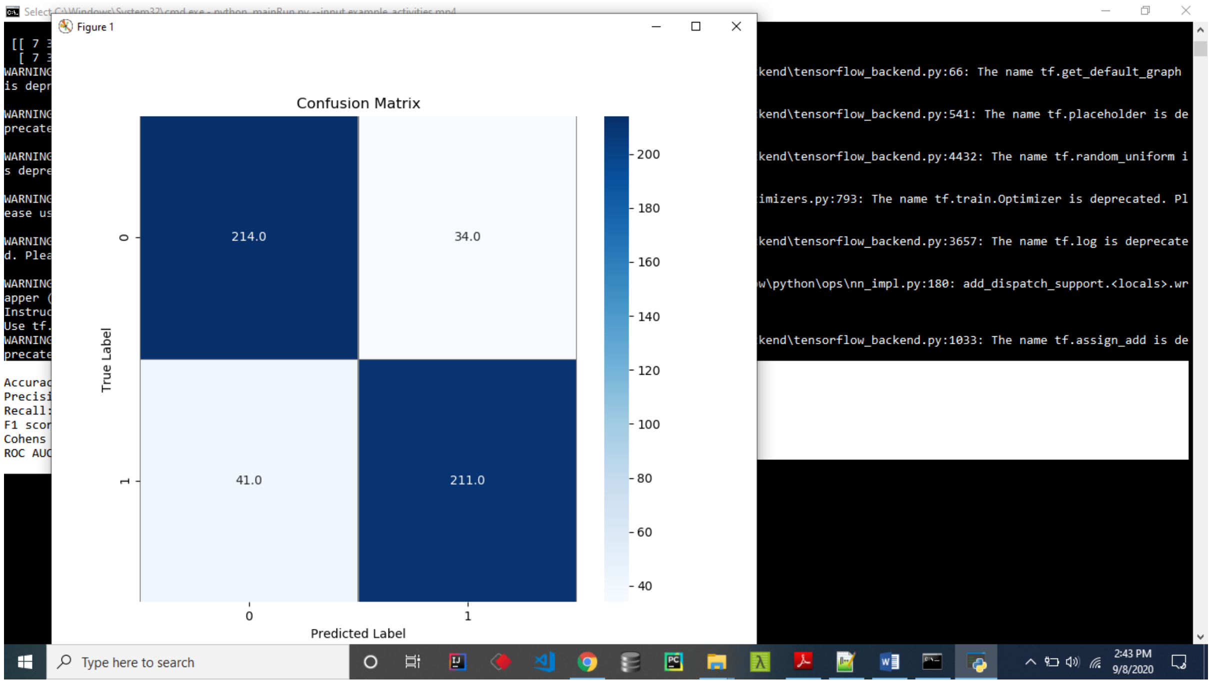
```
Select C:\Windows\System32\cmd.exe - python  mainRun.py --input example_activities.mp4
C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a sy
nonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Image shape: (427, 640, 3)
Patches shape: (272214, 2, 2, 3)
[[[ 3 18 13]
  [ 7 20 13]]

 [[ 3 18 13]
  [ 7 20 13]]]
[[[ 6 32 23]
  [ 8 31 23]]

 [[ 7 30 22]
  [ 7 30 22]]]
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:541: The name tf.placeholder is de
precated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:4432: The name tf.random_uniform i
s deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\optimizers.py:793: The name tf.train.Optimizer is deprecated. Pl
ease use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:3657: The name tf.log is deprecate
d. Please use tf.math.log instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wr
apper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:1033: The name tf.assign_add is de
precated. Please use tf.compat.v1.assign_add instead.

Accuracy: 0.850000
Precision: 0.861224
Recall: 0.837302
F1 score: 0.849095
Cohens kappa: 0.700048
ROC AUC: 0.925387
```

Accuracy:



```
Select C:\Windows\System32\cmd.exe - python  mainRun.py --input example_activities.mp4

 [[ 7 30 22]
  [ 7 30 22]]]
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:541: The name tf.placeholder is de
precated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:4432: The name tf.random_uniform i
s deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\optimizers.py:793: The name tf.train.Optimizer is deprecated. Pl
ease use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:3657: The name tf.log is deprecate
d. Please use tf.math.log instead.

WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wr
apper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\welcome\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:1033: The name tf.assign_add is de
precated. Please use tf.compat.v1.assign_add instead.

Accuracy: 0.850000
Precision: 0.861224
Recall: 0.837302
F1 score: 0.849095
Cohens kappa: 0.700048
ROC AUC: 0.925387
```

Confusion matrix:



Select C:\Windows\System32\cmd.exe - python mainRun.py --input example_activities.mp4

```
 [[ 7 3
  [ 7 3
WARNING
is depr

WARNING
precate

WARNING
s depre

WARNING
ease us

WARNING
d. Plea

WARNING
apper (
Instruc
Use tf.
WARNING
precate

Accurac
Precisi
Recall:
F1 scor
Cohens
ROC AUC
```

```
kend\tensorflow_backend.py:66: The name tf.get_default_graph

kend\tensorflow_backend.py:541: The name tf.placeholder is de

kend\tensorflow_backend.py:4432: The name tf.random_uniform i

imizers.py:793: The name tf.train.Optimizer is deprecated. Pl

kend\tensorflow_backend.py:3657: The name tf.log is deprecate

w\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wr


kend\tensorflow_backend.py:1033: The name tf.assign_add is de
```

CHAPTER—11
CONCLUSION

11. 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.

In future, we aim to extend this study by developing the context-aware recognition system to classify human activities. Also, we will extend our work to recognise complex human activities such as cooking, reading books, and watching TV.

CHAPTER-12
REFERENCE

12. REFERENCE

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 Thirtieth AAAI 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.