# VIDEO CLASSIFICATION BASED ON ACTIVITY RECOGNITION

## A PROJECT REPORT

Submitted to

**Jawaharlal Nehru Technological University Kakinada, Kakinada**

in partial fulfillment for the award of the degree of

**Bachelor of Technology**

**in**

**COMPUTER SCIENCE AND ENGINEERING**


Submitted by

**M.Venkata sai(16KN1A05E7)**

**Sk.Chandini (16KN1A05G1)** **P.Kavya sri (16KN1A05F7)**

Under the esteemed guidance of

**Mr.B.H.DASARADHA RAM**

**Associate Professor , CSE Department**




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NRI INSTITUTE OF TECHNOLOGY**

**Autonomous**

**(Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada)**

**Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' Grade**

**ISO 9001: 2015 Certified Institution**

**Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist, PIN: 521212, A.P, India.**


**2016-2020**

# NRI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada)
Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' Grade
ISO 9001: 2015 Certified Institution
Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist, PIN: 521212, A.P, India.

# Certificate

This is to certify that the Project entitled **"VIDEO CLASSIFICATION BASED ON ACTIVITY RECOGNITION"** is a bonafide work carried out by **M.Venkata sai (16KN1A05E7) Sk.Chandini(16KN1A05G1) and P.Kavya Sri(16KN1A05F7)** in partial fulfillment for the award of degree of Bachelor of Technology in **Computer Science and Engineering** of **Jawaharlal Nehru Technological University Kakinada, Kakinada** during the year 2019-2020.

**Project Guide**
**(Mr.B.H.Dasaradha Ram )**

**Head of the Department**
**(DrD.Suneetha)**

**EXTERNAL EXAMINER**

# DECLARATION

We hereby declare that the project report titled "**Video Classification based on Activity Recognition"** is a bonafide work carried out in the Department of Computer Science and Engineering, **NRI Institute of Technology**, Agiripalli,Vijayawada, during the academic year 2019-2020, in partial fulfilment for the award of the degree of **Bachelor of Technology** by JNTU Kakinada.

We further declare that this dissertation has not been submitted elsewhere for any Degree.

**16KN1A05E7**
**16KN1A05G1**
**16KN1A05F7**

# ACKNOWLEDGEMENT

We take this opportunity to thank all who have rendered their full support to my work. The pleasure, the achievement, the glory, the satisfaction, the reward, the appreciation and the construction of my project cannot be expressed with a few words for their valuable suggestions.

We are grateful to my Project Guide **Mr.B.H.Dasaradha Ram,Associate Professor** for rendering the valuable suggestions and extending his support to complete the project successfully.

We are expressing my heartfelt thanks to **Head of the Department, Dr. D.Suneetha** for his continuous guidance for completion of my Project work.

We are extending our sincere thanks to **Dean of the Department, Dr. K. V. Sambasiva Rao** for his continuous guidance and support to complete our project successfully.

We are thankful to the **Principal, Dr. C. Naga Bhaskar** for his encouragement to complete the Project work.

We are extending my sincere and honest thanks to the **Chairman, Dr. R. Venkata Rao** garu & **Secretary, Sri K. Sridhar** for their continuous support in completing the Project work.

Finally, I thank the Administrative Officer, Staff Members, Faculty of Department of CSE, NRI Institute of Technology and my friends, who directly or indirectly helped us in the completion of this project.

**16KN1A05E7**
**16KN1A05G1**
**16KN1A05F7**

# ABSTRACT

Action recognition in video sequences is a challenging problem in computer vision. In the context of videos, an action is represented as a sequence of frames, which computers can easily understand by analysing the contents of multiple frames in sequence. We recognize human actions similar to our observations of actions in real life. In this project only images will be considered as scenes of activities, these images will be used in video classification. these will be designed by using N number of scenes considered as datasets these can be understood as an activity, such activities will be fed into machine learning model using transfer learning for greater accuracy and with the help of this machine learning model the video classification is possible, hence this video classification can be used as activity recognition. We use Bi-LSTM to consider the sequence of information of frames of video in automatic understanding of actions.

# INDEX

# LIST OF FIGURES

# CHAPTER-1

# INTRODUCTION

## 1.1 Introduction:

Action recognition in video sequence is a challenging problem in computer vision due to similarity of visual content, changes in the view point for the same actions, camera motion with action performer, scale and pose of an actor, and different illumination conditions. Human actions range from simple activity through arm or leg to complex integrated activity of combined arms, legs and body. For example, the legs motions for kicking a football in a simple action, while jumping for a head-shoot is a collective motion of legs, arms, head and whole body. Generally, human action is a motion of body parts by interacting with objects in the environment. In the context of videos an action is represented using a sequence of frames, which humans can easily understand by analyzing the contents of multiple frames in sequence. We recognize human actions in a way similar to our observation of actions in real life. We use LSTM to consider the information of previous frames in automatic understanding of actions in videos.

## 1.2 Problem Definition:

Action recognition using video analysis is computationally expensive as processing a short video may take a long time due to its high frame rate. As each frame plays an important role in a video storing, keeping information of sequential frames for a long time makes the system more efficient. Researchers have presented many solutions for this problem such as motion, space time features, and trajectories.

## 1.3 Problem Solution:

The solution to this problem is LSTM. The main idea of LSTM architecture is its memory cell, input gate, output gate and forget gate , which can maintain its state over time T N, and non-linear gating units which regulate the information flow into/out of the cell. Researchers have presented different variations of LSTM such as multi-layer LSTM and bi-directional LSTM for processing sequential data. The proposed method analyses the complex pattern in the visual data of each frame, which cannot be efficiently identified using simple LSTM and multi-layer LSTM.

## 1.4 Process Diagram:

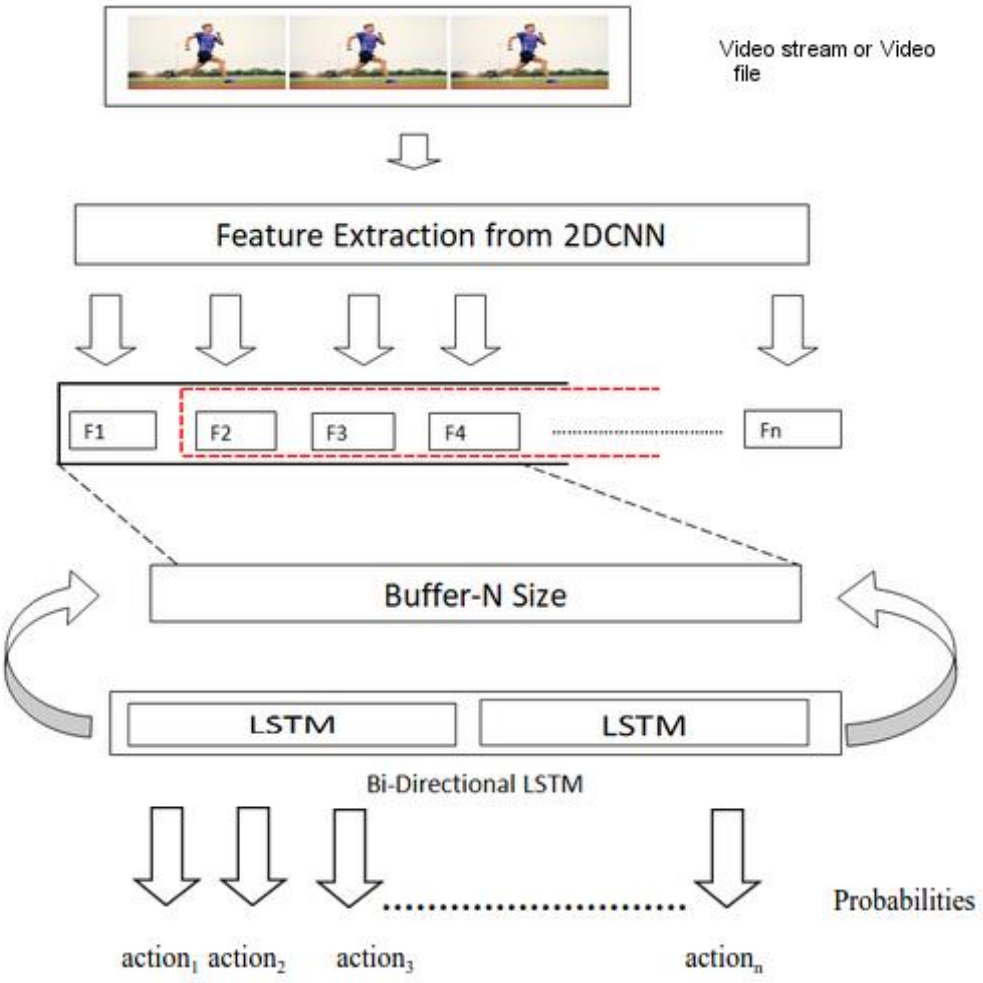


**Fig.1 : Continues Video Classification Process Diagram**

# CHAPTER – 2
# LITERATURE REVIEW

## 2. Literature Review:

1. Action recognition in video sequence using deep bi-directional lstm with cnn features AMIN ULLAH 1,(student member, IEEE), JAMIL AHMAD 1,(student member, IEEE),KHAN MUHAMMED 1,(student member, IEEE), MUHAMMED SAJJAD 2, SUNG WOOK BAIK 1,(member, IEEE).
2. Understanding LSTM Networks ,Source: Wikipedia,colah's blog
3. Effectiveness of Recurrent Neural Networks - From Andrej Karpathy Blog
4. A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning - Source: Towards Data Science

# CHAPTER-3

# SYSTEM ANALYSIS

## 3.1 Existing System:

Researchers have presented many solutions for this problem such as motion, space time features and trajectories. In the existing system it uses traditional machine learning models to train the data because using this technique machine requires a lot of data. In traditional models while training the data system requires to tune the channels to get high frequency.

## Disadvantages of existing system:

- By using this model machines use more computing power.
- It cannot perform accurate results.
- It takes more time to get the result and complexity of data.

## 3.2 Proposed System:

The proposed method uses Recurrent Neural Network “LSTM " to analyze frame to frame change of action videos. a novel action recognition method by processing the video data using Convolutional Neural Network(CNN) and Deep Bi-Directional LSTM(Bi-LSTM) network. For each frame we extract the features of images using CNN then the features are pushed into a deque, which is then passed to a LSTM which covers the action recognition part.

## Advantages of proposed system:

- This system helps to speed up the model training process and get an accurate result.
- Low computing power compared to existing systems.
- System will continuously

## 3.3 ANALYSIS MODEL:

- **Waterfall Model - Design**

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In the "The Waterfall" approach, the whole

process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model-

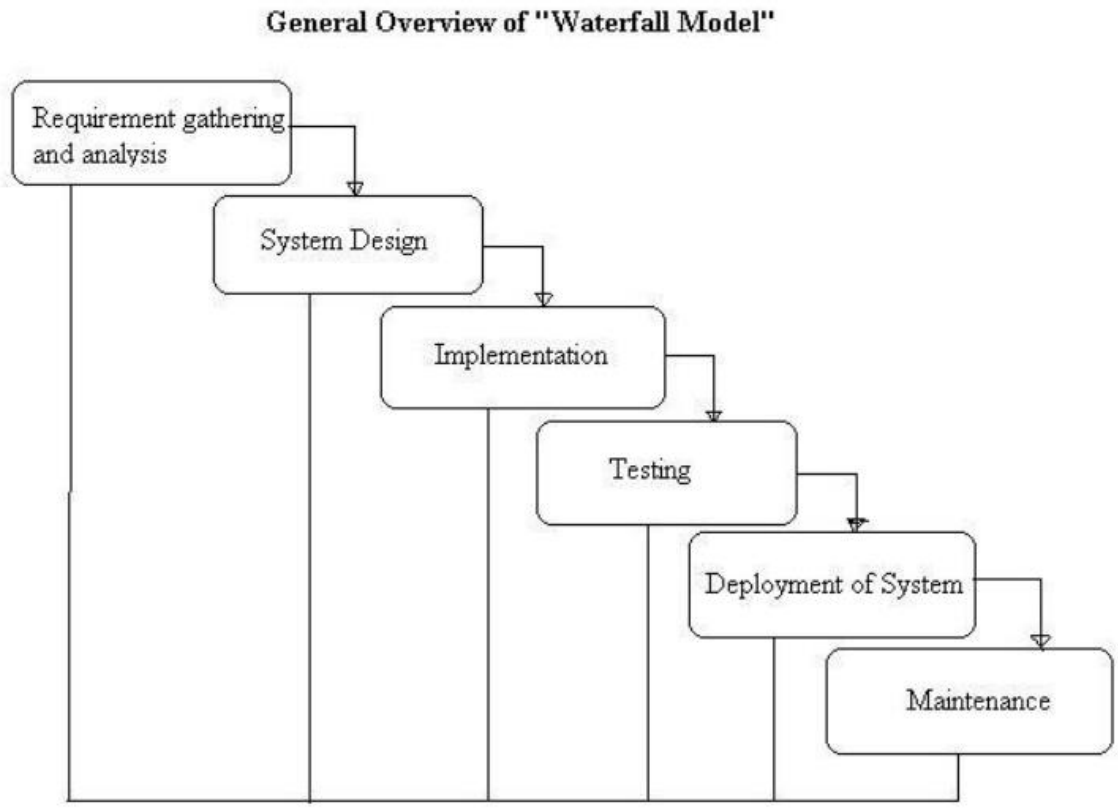


**Fig: 2. Waterfall Model diagram**

Sequential phases in Waterfall model are :

• **Requirement Gathering and Analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

• **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

• **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

• **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

• **Deployment of System** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

• **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

## 3.4 Modules:

- ### Create images from video:

  In this project we extract images from each video for classifying it in each class and store them as a trained data.

- ### CNN Trainer:

  Here we use the transfer learning concept and it works as a model for extracting features that will be trained here. The extracted images are taken as input.

- **Feature Extractor:**

After the CNN trainer it extracts features using this trained CNN model from video and saves the extracted features.

- **Bi-LSTM Trainer:**

Train based on extracted features as input and save the trained model.

- **Video Classifier:**

Using a CNN to extract features that predict images scene action, use trained LSTM for predicting the video class for the given video.

# CHAPTER-4

# FEASABILITY STUDY

## 4. Feasibility Study:

Feasibility study is a high level capsule version of the entire process intended to answer a number of questions like: what is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem is clearly understood. A feasibility study is necessary to determine that the proposed system is feasible by considering the technical, operational, and economical factors. By having the detailed feasibility study the management will have a clear-cut view of the proposed system.

The following feasibilities are considered for the project in order to ensure that the project is variable and it doesn't have any major obstructions. Feasibility study encompresses the following things

- Technical feasibility
- Economical feasibility
- Operational feasibility

In this phase, we study the feasibility of all proposed systems, and pick the best feasible solution for the problem. The feasibility is studied based on the three main factors as follows.

### 4.1 Technical Feasibility:

In this step, we verify whether the proposed systems are technically feasible or not. That is all the technologies required to develop the system are available readily or not.

Technical feasibility determines whether an organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds.

- This system is flexible and it can be expanded further.
- This system can guarantee accuracy, ease of use and reliability.
- Our project is technically feasible because all the technology needed for our project is readily available.

### 4.2 Economic Feasibility:

Economical feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process the entire project or whether the benefits obtained from the new system or not are worth the costs. Financial benefits must be equal or exceed the costs in this issue, we should consider:

- The development tool.
- The cost of hardware and software for the class of application being considered.
- The cost of maintenance etc.

Our project is economically feasible because the cost of development is very minimal when compared to the financial benefits of the application.

## 4.3 Operational Feasibility:

In this step, we verify different operational factors of the proposed systems like manpower, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility determines if the proposed system satisfied user objectives could be fitted into the current system operation.

- The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
- The clients have been involved in the planning and development of the system.
- The proposed system will not cause any problem under any circumstances.
- Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

# CHAPTER-5

# SOFTWARE REQUIREMENT SPECIFICATION

## 5.1 Introduction:

A Software Requirements specification (SRS) – a requirements specification for a software system is a complete description of behavior of a system to be developed. It includes a set of cases that describe all the interactions users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints). System Requirements Specification It is a collection of information that embodies the requirements of a system.

A system engineering requirement can be a description of what a system must do, referred to as Functional Requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Non-functional requirements, or 'Performance requirements' or 'Quality of service requirements'.

## 5.2 Functional Requirements:

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system.

## 5.3 Non-Functional Requirements:

In systems engineering and requirements engineering, a **non-functional requirement** is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors**.**

- **Availability:** A system's "availability" or "uptime" is the amount of time that is operational and available for use. It's related to the server providing the service to the users in displaying images. As our system will be used by thousands of users at any time our system must be available always. If there are any cases of updates they must

be performed in a short interval of time without interrupting the normal services made available to the users.

- **Flexibility:** If the organization intends to increase or extend the functionality of the software after it is deployed, that should be planned from the beginning; it influences choices made during the design, development, testing and deployment of the system. New modules can be easily integrated to our system without disturbing the existing modules or modifying the logical database schema of the existing applications.

- **Scalability:** Software that is scalable has the ability to handle a wide variety of system configuration sizes. The nonfunctional requirements should specify the ways in which the system may be expected to scale up (by increasing hardware capacity, adding machines etc.). Our system can be easily expandable. Any additional requirements such as hardware or software which increase the performance of the system can be easily added. An additional server would be useful to speed up the application.

- **Integrity:** Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software.

- **Usability:** The system is designed with a completely automated process hence there is no or less user intervention. Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users. Hyperlinks will be provided for each and every service the system provides through which navigation will be easier. A system that has high usability coefficient makes the work of the user easier.

- **Performance:** This system is developing in the high level languages and using the advanced back end technologies it will give response to the end user and client system within very less time. The performance constraints specify the timing characteristics

of the software. Making the application form filling process through online and providing the invigilation list information and examination hall list is given high priority compared to other services and can be identified as the critical aspect of the system.

- **Efficiency:** The back-end TensorFlow was developed in the C++ language which is considered to be efficient and is used in our project.

## 5.4 SYSTEM REQUIREMENTS:

### 5.4.1 Software Requirements:

- Operating System : Linux or Windows
- Environment : Python-3.x
- Library : Tensorflow
- High level module : Keras
- Datasets : UCF

### 5.4.2 Hardware Requirements:

- RAM : 8GB+
- Processor : i5
- GPU : Nvidia GTX 1080

# CHAPTER-6

# SYSTEM DESIGN

## 6.1 UML Modeling:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practice that has proven successful in the modeling of large and complex systems. The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the help UML helps project teams communicate, explore potential designs and validate the architectural design of the software.

## 6.2 CLASS DIAGRAM:

Class-based Modeling, or more commonly class-orientation, refers to the style of object-oriented programming in which inheritance is achieved by defining classes of objects; as opposed to the objects themselves (compare Prototype-based programming).

The most popular and developed model of OOP is a class-based model, as opposed to an object-based model. In this model, objects are entities that combine state (i.e., data), behavior (i.e., procedures, or methods) and identity (unique existence among all other objects). The structure and behavior of an object are defined by a class, which is a definition, or blueprint, of all objects of a specific type. An object must be explicitly created based on a class and an object thus created is considered to be an instance of that class. An object is similar to a structure, with the addition of method pointers, member access control, and an implicit data member which locates instances of the class (i.e. actual objects of that class) in the class hierarchy (essential for runtime inheritance features).

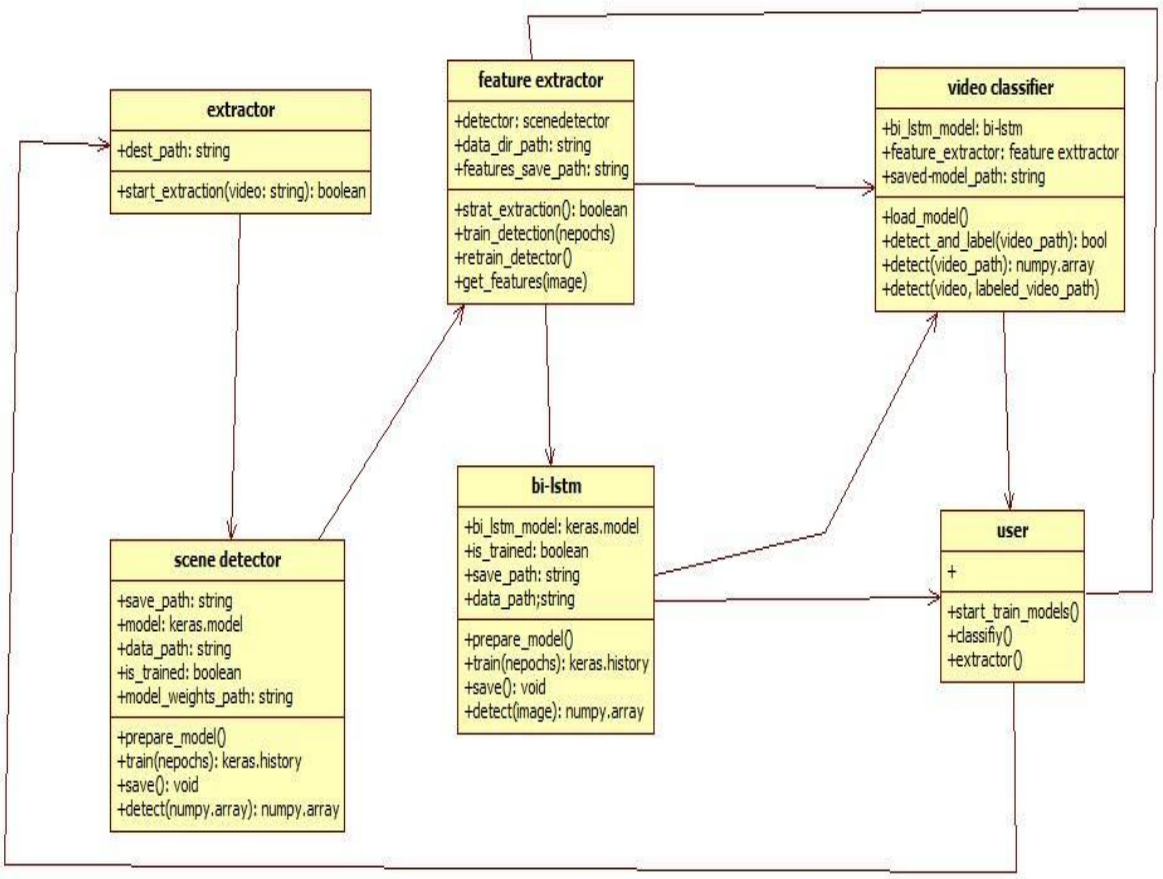


**Fig:2. Project-Class Diagram**

## 6.3 USECASE DIAGRAM:

Use case diagram represents the functionality of the system. Use case focus on the behavior of the system from an external point of view. Actors are external entities that interact with the system.

## Use cases:

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

## Actors:

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

## **System Boundary Boxes** (Optional):

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of the system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

## Include:

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case.

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»".There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

## Extend:

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

## Generalization:

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions wit0h a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation.

## Association:

Association between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicate the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

## Identified Use Cases:

The "user model view" encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of "use case diagrams". These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.

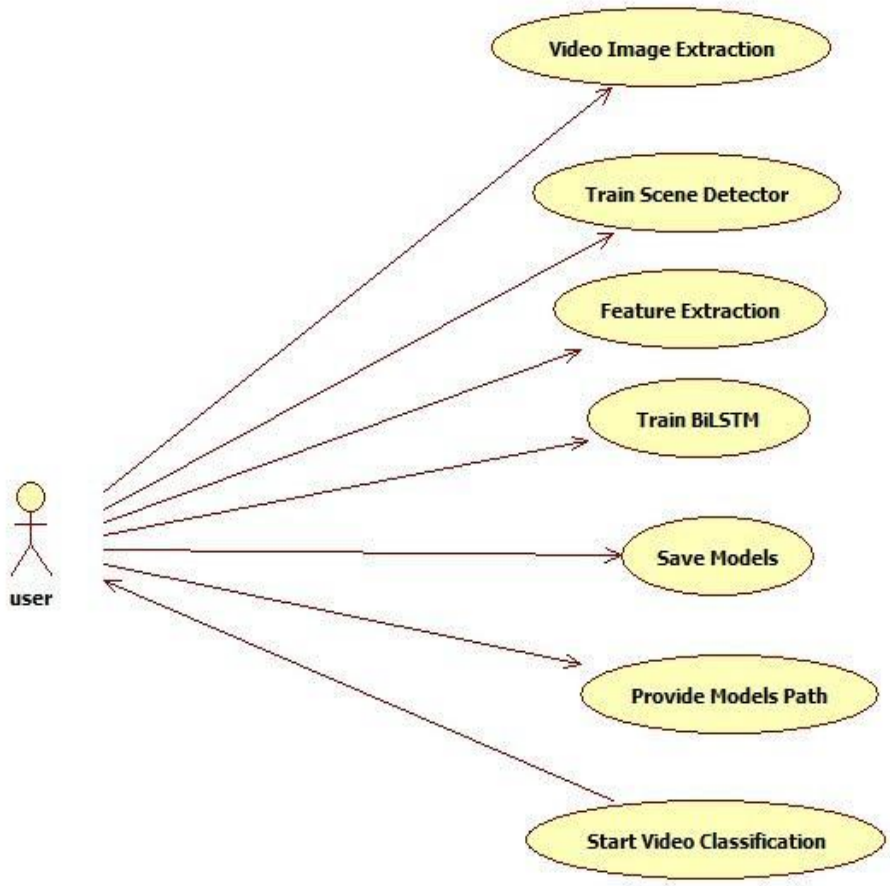


**Fig 3: Project-Usecase Diagram**

## 6.4 Activity Diagram:

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

Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows. The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start (split) or end (join) of concurrent activities.
- A black circle represents the start (initial state) of the workflow.
- An encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with the decisions or loops.

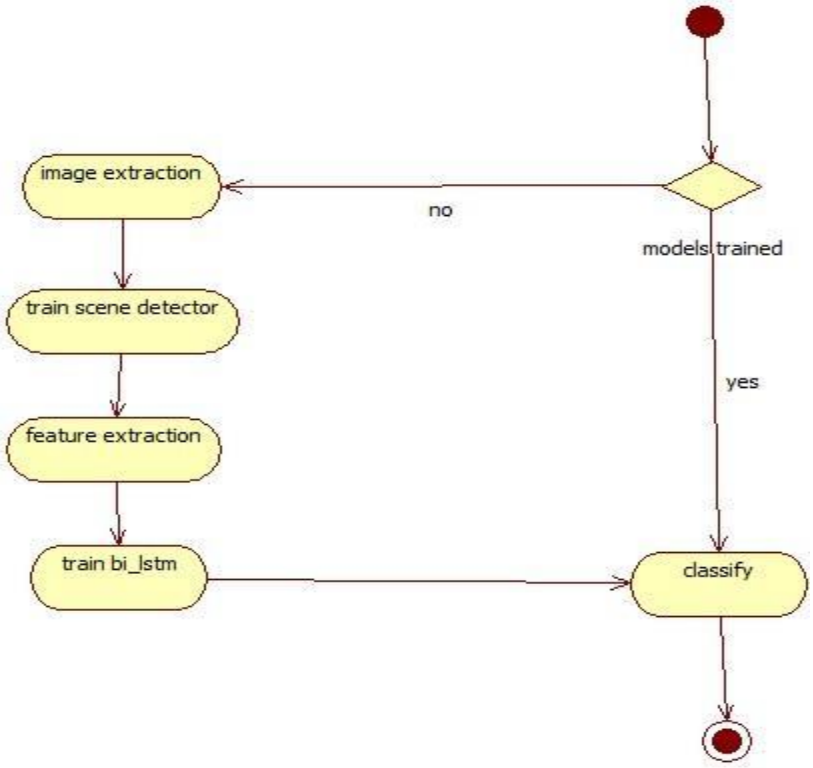


**Fig:4. Project-Activity Diagram**

## 6.5 Sequence Diagram:

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

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple run time scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Note that leaving

the instance name blank can represent anonymous and unnamed instances. In order to display interaction, messages are used. These are horizontal arrows with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages. This definition is true as of UML 2, considerably different from UML 1.x.

Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though). It should be the result of a message, either from the object itself, or another.

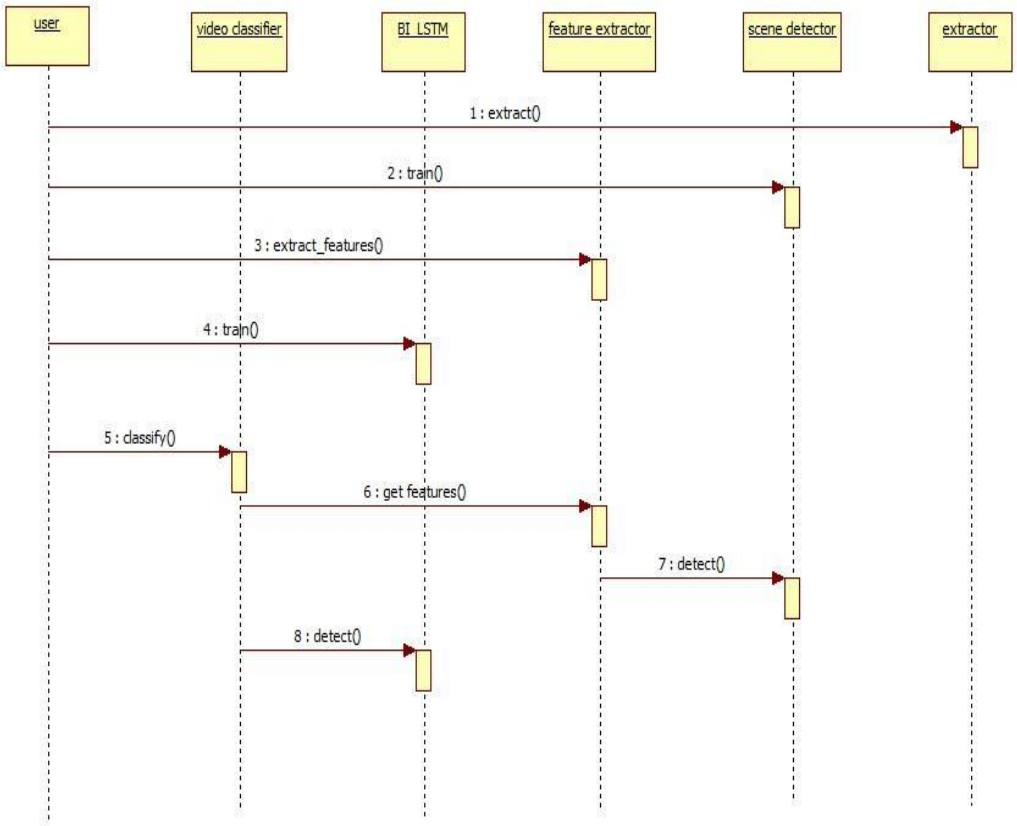


**Fig:5.Project-Sequence Diagram**

## 6.6 Component Diagram:

## What is a Component?

This section defines the term component and discusses the differences between object oriented, traditional, and process related views of component level design. Object Management Group OMG UML defines a component as “A modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces”.

## An Object Oriented View:

A component contains a set of collaborating classes. Each class within a component has been fully elaborated to include all attributes and operations that are relevant to its implementation. As part of the design elaboration, all interfaces (messages) that enable the classes to communicate and collaborate with other design classes must also be defined. To accomplish this, the designer begins with the analysis model and elaborates analysis classes (for components that relate to the problem domain) and infrastructure classes (or components that provide support services for the problem domain).

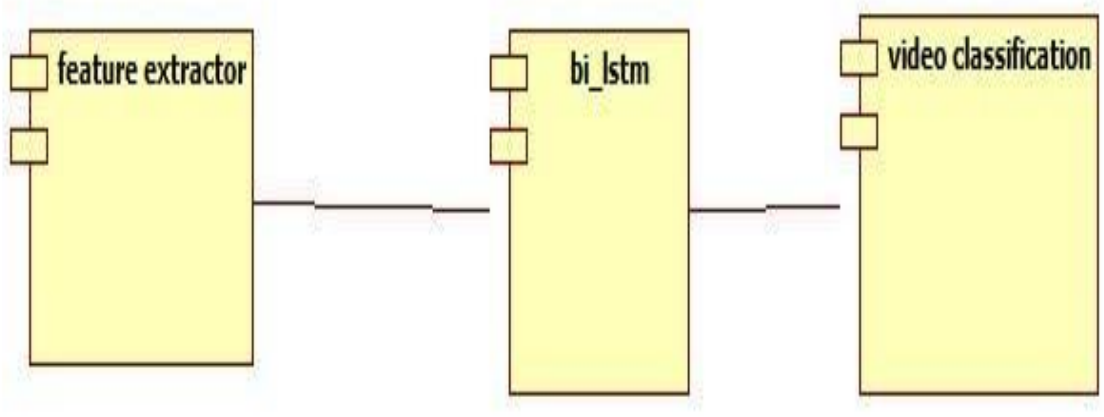


**Fig:6. Project-Component Diagram**

## 6.7 Deployment Diagram:

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

## Purpose:

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe run time processing nodes.

**How to draw a Deployment Diagram?**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application. Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important because it controls the following parameters

- Performance
- Scalability
- Maintainability
- Portability

So before drawing a deployment diagram the following artifacts should be identified:

- Nodes
- Relationships among nodes

The following deployment diagram is a sample to give an idea of the deployment view of order management system. Here we have shown nodes as:

- Monitor
- Modem
- Caching server
- Server

The application is assumed to be a web based application which is deployed in a clustered environment using server 1, server 2 and server 3. The user is connecting to the application using the internet. The control is flowing from the caching server to the clustered environment. So the following deployment diagram has been drawn considering all the points mentioned above.

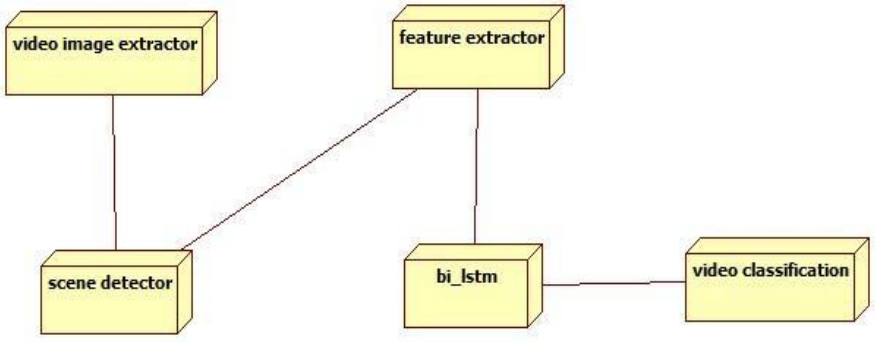


**Fig:7. Project Deployment Diagram**

# CHAPTER-7

# CODING

## 7.1 Software Description:

## Algorithm:

### 1. Convolution Neural Networks (CNN):

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels incase of an image).

2. Hidden Layer: The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.

3. Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

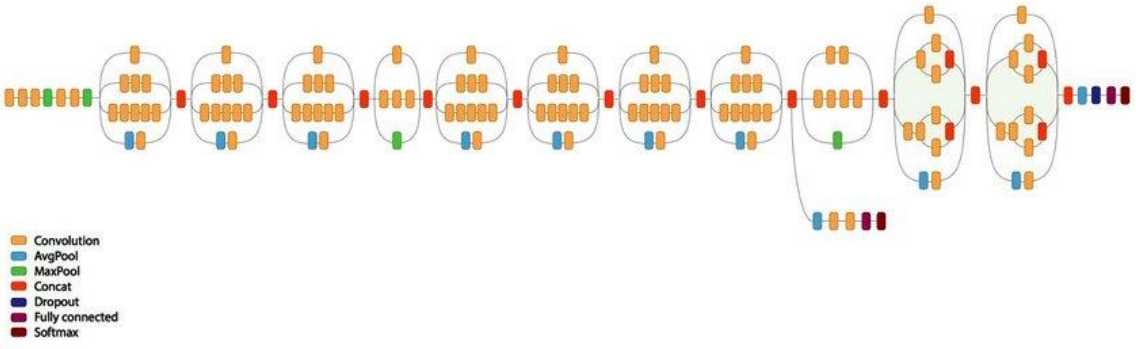


**Fig:8. Inception V3 Convolutional Neural Network**

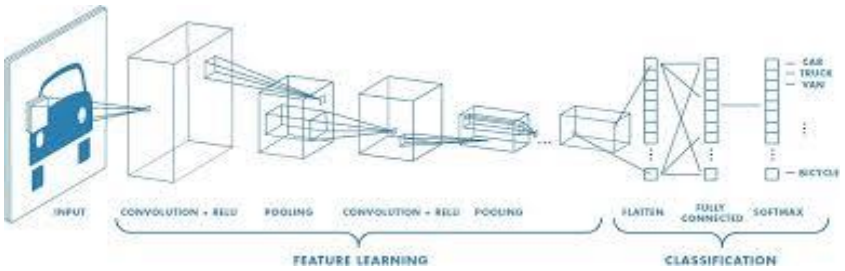


**Fig:9.Smaller Convolutional Neural Network Mobilnet**

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

**Layers used to build ConvNets**

A convnets is a sequence of layers, and every layer transforms one volume to another through differentiable functions.

Types of layers:

Let's take an example by running a covnets on of image of dimension 32 x 32 x

1) Input Layer: This layer holds the raw input of image with width 32, height 32 and depth 3.

2) Convolution Layer: This layer computes the output volume by computing dot product between all filters and image patch. Suppose we use a total 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.

3) Activation Function Layer: This layer will apply element wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.

4) Pool Layer: This layer is periodically inserted in the covenants and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resulting volume will be of dimension 16x16x12.

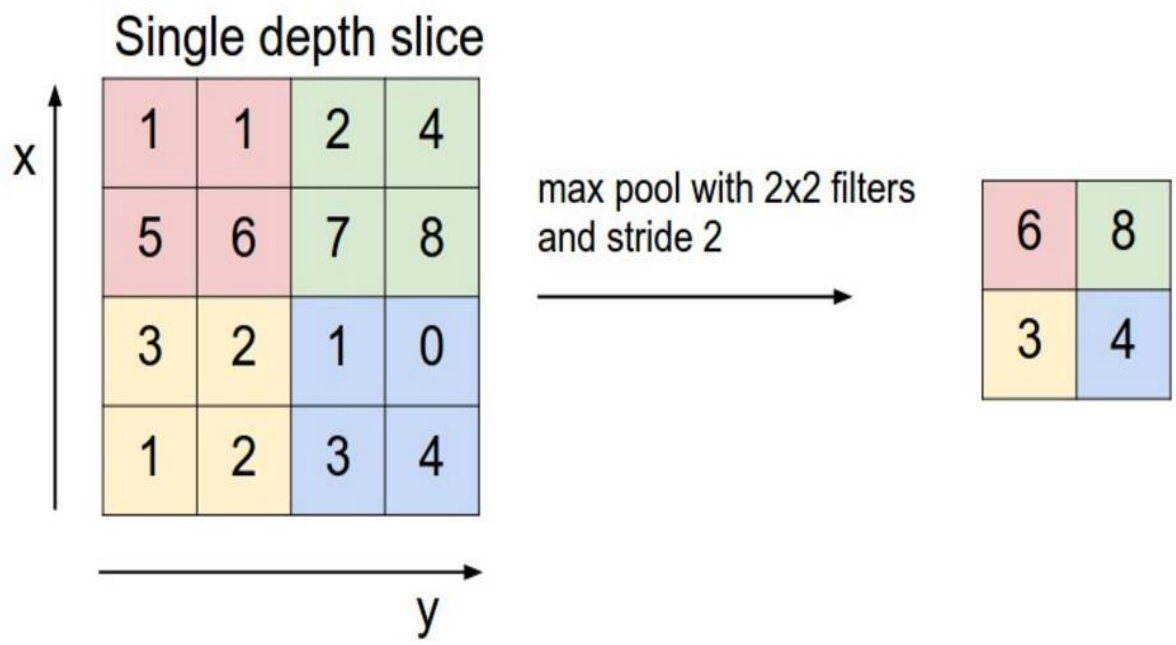


**Fig:10. Max Pooling For Feature Reduction**

5) Fully-Connected Layer: This layer is a regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

## 2. Bidirectional Long-Short Term Memory (Bi-LSTM):

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

Our aim is to use the power of CNNs to detect spatial features and RNNs for the temporal features, effectively building a CNN->LSTM network, or CRNN. Use transfer learning to retrain the Inception v3 network on our training data, like we did in our previous post

1. Pass our train and test data through the CNN, saving the output of a couple different layers to disk (see below)
2. Convert the output of individual frames into sequences of frames
3. Train the new LSTM on the train set
4. Evaluate the LSTM on the test set of data and see if it performs better than the 93.3% benchmark we set in the previous post.

Step 2 is unique so we'll expand on it a bit. There are two interesting paths that come to mind when adding a recurrent net to the end of our convolutional net:

1. We can pass the actual label predictions generated by the softmax layer of the CNN to the RNN. This gives us the probability that the frame is each of our classes, football or ad, which is the prediction we used in our previous post.
2. We can pass the output of the pool layer, before it's made into a prediction, to the RNN. The pool layer gives us a 2,048-d vector that represents the convoluted features of the image, but not a class prediction.

## Frames to Sequences:

- In order to turn our discrete predictions or features into a sequence, we loop through each frame in chronological order, add it to a queue of size *N*, and pop off the first frame we previously added.
- *N* represents the length of our sequence that we'll pass to the RNN. We could choose any length for N, but I settled on 40. At 10fps, which is the framerate of our video, that gives us 4 seconds of video to process at a time. This seems like a good balance of memory usage and information.

## Softmax Layer Method:

If one frame is an ad and the next is a football game, it's essentially impossible that the next will be an ad again. (I wish commercials were only 1/10th of a second long!)This is why it could be interesting to examine the temporal dependencies of the probabilities of each label before we look at the more raw output of the pool layer. We convert our individual predictions into sequences using the code above, and then feed the sequences to our RNN.

After training the RNN on our first batch of data, we then evaluate the predictions on both the batch we used for training and a holdout set that the RNN has never seen. No surprise, evaluating the same data we used to train gives us an accuracy of 99.55%! Good sanity check that we're on the right path.

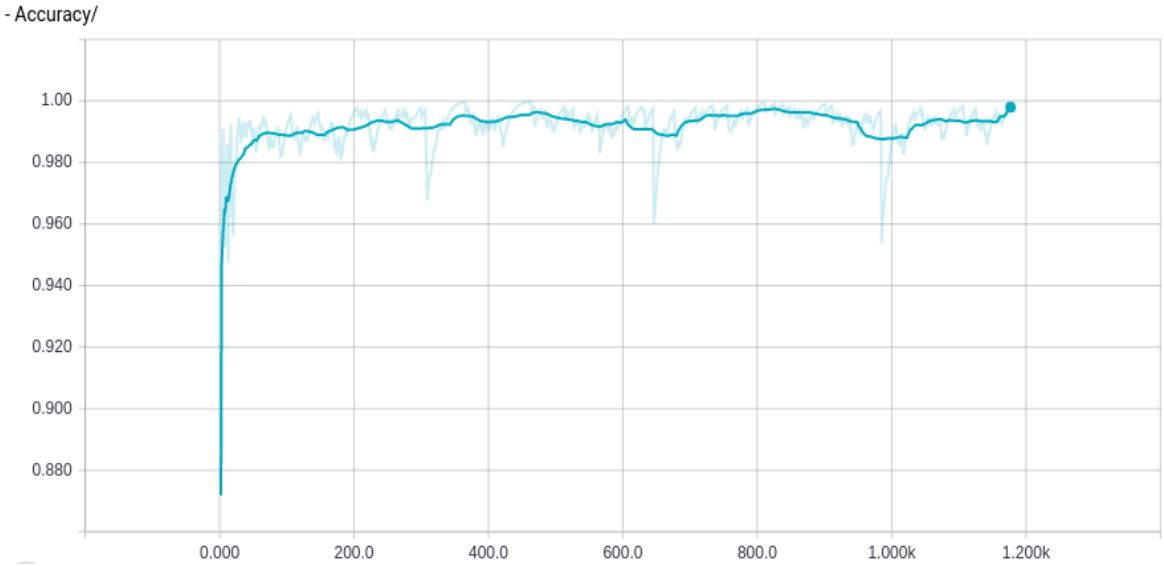


**Fig:11. Accuracy using Features from Softmax Layer**

## Pool layer method:

Instead of letting CNN do all the hard work, we'll give more responsibility to theBi-LSTM by using the output of CNN's pool layer, which gives us the feature representation (not a prediction) of our images. We again build sequences with this data to feed into our Bi-LSTM.

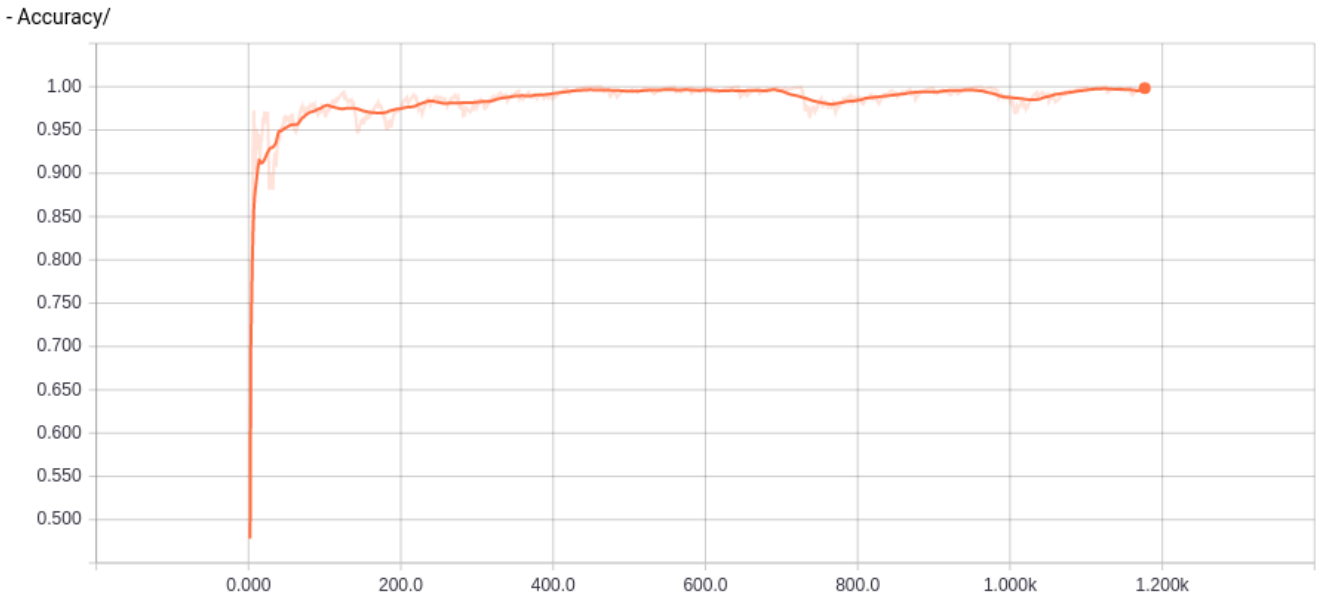


**Fig:12. Accuracy using Features from Pool Layer**

## Packages:

### 1. Keras:

**Module:** tf.keras

## Modules:

Activations module : Built-in activation functions.

Applications module : Keras Applications are canned architectures with pre-trained weights.

Backend module : Keras backend API.

Callbacks module : Callbacks: utilities called at certain points during model training.

Constraints module : Constraints: functions that impose constraints on weight values.

Datasets module : Public API for tf.keras.datasets namespace.

Estimator module : Keras estimator API.

Experimental module : Public API for tf.keras.experimental namespace.

Initializers module : Keras initializer serialization / deserialization.

Layers module : Keras layers API.

Losses module : Built-in loss functions.

Metrics module : Built-in metrics.

Mixed_precision module : Public API for tf.keras.mixed_precision namespace.

Models module Code for model cloning, plus model-related API entries.

Optimizers module : Built-in optimizer classes.

Preprocessing module : Keras data preprocessing utils.

Regularizers module : Built-in regularizers.

Utils module : Public API for tf.keras.utils namespace.

Wrappers module : Public API for tf.keras.wrappers namespace.

**Classes**

Class Model : Model groups layers into an object with training and inference features.

Class Sequential : Linear stack of layers.

## ➢ Layers Module:

Keras layers API.

## Modules

experimental module: Public API for tf.keras.layers.experimental namespace.

**Classes**

- class BatchNormalization : Normalize and scale inputs or activations. (Ioffe and Szegedy, 2014).
- class Conv2D : 2D convolution layer (e.g. spatial convolution over images).
- class Dense : Just your regular densely-connected NN layer.
- class Dropout : Applies Dropout to the input.
- class Flatten : Flattens the input. Does not affect the batch size.
- class InputLayer : Layer to be used as an entry point into a Network (a graph of layers).
- class Layer : Base layer class.
- class LayerNormalization: Layer normalization layer (Ba et al., 2016).
- class MaxPooling2D : Max pooling operation for spatial data.
- class ReLU : Rectified Linear Unit activation function.
- class Wrapper : Abstract wrapper base class.

## Functions

Input(...) : Input() is used to instantiate a Keras tensor.

add(...) : Functional interface to the Add layer.

| | | |
|---|---|---|
| average(...) | : | Functional interface to the Average layer. |
| concatenate(...) | : | Functional interface to the Concatenate layer. |
| deserialize(...) | : | Instantiates a layer from a config dictionary. |
| dot(...) | : | Functional interface to the Dot layer. |
| maximum(...) | : | Functional interface to the Maximum layer that computes |
| minimum(...) | : | Functional interface to the Minimum layer. |
| multiply(...) | : | Functional interface to the Multiply layer. |
| subtract(...) | : | Functional interface to the Subtract layer |

## 1. **Numpy:**

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created the NumPy package by incorporating the features of Numarray into the Numeric package. There are many contributors to this open source project.

**Operations using NumPy:**

Using NumPy, a developer can perform the following operations –

-Mathematical and logical operations on arrays.

-Fourier transforms and routines for shape manipulation.

-Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

## 7.2 Sample Code:

## 1. Image Extraction From Video:

### image_extractor.py

```
import cv2

def extract(video,dest_dir,filename):

      import os

      if not (os.path.exists(video) and os.path.exists(dest_dir)):

       raise Exception('path does not exists')

      import subprocess as sp

mimetype = sp.check_output(["file","-b","--mime-type",video]).decode('utf-
8')

      if not mimetype.startswith('video'):

            raise Exception('video is taken as input '+video)

            retina = cv2.VideoCapture(video)

            count = 0

             havemore,frame = retina.read()

       while havemore:

       count+=1

       loc = os.path.join(dest_dir,filename+"-
"+str(count).rjust(4,"0")+".jpeg")

       if count%6 == 0:

            cv2.imwrite(loc,frame)

       havemore,frame = retina.read()

        return count
```

```
  def ofoneclass(parent,dest_dir,classname):
        import os
        if not (os.path.exists(parent) and os.path.exists(dest_dir)):
           raise Exception('path does not exists '+classname)
        video_cnt = 1
        dirs = os.listdir(parent)
        for videop in dirs:
           vfullpath = os.path.join(parent,videop)
           filename = classname+str(video_cnt).rjust(2,"0")
           count = extract(vfullpath,dest_dir,filename)
           video_cnt+=1
           print("[INFO] "+vfullpath+"(%d) is completed"%(count))
def catch_class(data_dir,dest_dir):
        import os
        if not (os.path.exists(data_dir) and os.path.exists(dest_dir)):
           raise Exception('path does not exists')
        dirs = os.listdir(data_dir)
        for d in dirs:
           #if d.startswith("."):continue
           dest_dir_d = os.path.join(dest_dir,d)
           if os.path.isfile(dest_dir_d) or os.path.exists(dest_dir_d) :
                 continue
           os.mkdir(dest_dir_d)
```

```
            print("[INFO] Directories Created")
            for d in dirs:
               #if d.startswith("."):continue
               dest_dir_d = os.path.join(dest_dir,d)
               #print(dest_dir_d)
               if os.path.isfile(dest_dir_d) :
                     continue
               src_dir_d = os.path.join(data_dir,d)
               ofoneclass(src_dir_d,dest_dir_d,d)
               print("[INFO] "+d+" completed")
catch_class("./data","./ims")
```

## 2. Image Data Preparation:

```
!python3 extract.py
IMAGE_SIZE = [224, 224]
BATCH_SIZE = 32
IMG_DIR = "data/image_data"
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping, ReduceLROnPlateau,
ModelCheckpoint
train_datagen = ImageDataGenerator(
   rotation_range=40,width_shift_range=0.2,height_shift_range=0.2,
   rescale=1/255,shear_range=0.2,zoom_range=0.2,fill_mode='nearest',
   horizontal_flip = True,validation_split=0.2)
```

```
test_datagen = ImageDataGenerator(rescale=1/255)

train_generator = train_datagen.flow_from_directory(

   IMG_DIR,target_size=IMAGE_SIZE,batch_size=BATCH_SIZE,

   class_mode='categorical',subset='training')

validation_generator = train_datagen.flow_from_directory(

   IMG_DIR,target_size=IMAGE_SIZE,batch_size=BATCH_SIZE,

   class_mode='categorical',subset='validation')
```

## 2. CNN Preparation:

```
from keras.models import Sequential

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

prior = keras.applications.VGG16(include_top=False,

   weights='imagenet',input_shape=IMAGE_SIZE + [3])

model = Sequential()

model.add(prior)

model.add(MaxPooling2D(pool_size=(7, 7)))

model.add(Flatten())

model.add(Dense(512, activation="relu"))

model.add(Dropout(rate=0.2))

model.add(Dense(len(train_generator.class_indices), activation='softmax',
name='Output_Layer'))

model.layers[0].trainable = False
```

## 3. Train CNN:

```
model.compile(optimizer="adam",loss='categorical_crossentropy',metrics=['a
ccuracy'])

history = model.fit_generator(
```

```
      train_generator,
      steps_per_epoch=len(train_generator.filenames) // BATCH_SIZE,
      epochs=20,
   validation_data=validation_generator,
   validation_steps=len(train_generator.filenames) // BATCH_SIZE,
   callbacks=[checkpoint],
   use_multiprocessing = True,workers=16
)
```

## 4. Feature Extractor Class Code:

```
import numpy as np
from keras.models import load_model,Model
from sklearn.preprocessing import LabelEncoder
import cv2
import os
from collections import deque
from sklearn.preprocessing import OneHotEncoder
class FeatureExtractor():
    def __init__(self,nframes,cnn,imdimension):
       #self.qu = deque([])
       self.cnn = load_model(cnn)
       self.model = Model(
             inputs = self.cnn.input,
             outputs = self.cnn.get_layer('max_pooling2d_2').output
       )
       self.X = list()
       self.y = list()
       self.n = nframes
       self.actual = None
```

```
        self.imsize = imdimension
    def add_features(self,video,actual_class):
        temp_list = deque()
       v = cv2.VideoCapture(video)
        havemore,frame = v.read()
        while havemore:
              frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
              frame = cv2.resize(frame, self.imsize).astype("float32")
              features =
self.model.predict(np.expand_dims(frame,axis=0))[0].reshape((512))
              assert len(features)!=512
              if len(temp_list) == self.n:
                    self.X.append(np.array(list(temp_list)))
                    self.y.append(self.actual.transform([actual_class]))
                    temp_list.popleft()
              else:
                    temp_list.append(features)
              havemore,frame = v.read()
     def ofoneclass(self,videos_dir,classname):
        if not os.path.exists(videos_dir):
              raise Exception('path does not exists '+videos_dir)
        class_dirs = os.listdir(videos_dir)
        #print(dirs,parent)
        for videop in class_dirs:
              video_fullpath = os.path.join(videos_dir,videop)
              self.add_features(video_fullpath,classname)
              print("[INFO]",video_fullpath,"completed")
    def load_from_directory(self,src_dir):
```

```
        ls = os.listdir(src_dir)
        self.actual = LabelEncoder().fit(ls)
        for folder in ls:
              dir_rel_path = os.path.join(src_dir,folder)
              self.ofoneclass(dir_rel_path,folder)
              print("[INFO]",dir_rel_path,"completed")
     def get_features(self,frame):
        return
self.model.predict(np.expand_dims(frame,axis=0))[0].reshape((512))
    def load_dataset(self):
        y = OneHotEncoder(sparse=False).fit_transform(self.y)
        return self.X,y
     def genrate_features(self,video):
        if not os.path.exists(video):
              raise ValueError('video does not exist')
        temp_list = deque()
        v = cv2.VideoCapture(video)
        havemore,frame = v.read()
        while havemore:
              frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)


        frame = cv2.resize(frame, self.imsize).astype("float32")
              features =
self.model.predict(np.expand_dims(frame,axis=0))[0].reshape((512))
              if len(temp_list) == self.n:
                    #self.X.append(list(temp_list))
                    #self.y.append(self.actual.transform([actual_class]))
                    yield np.array(temp_list)
```

```
                temp_list.popleft()
            else:
                temp_list.append(features)
            havemore,frame = v.read()
    def save(self):
       np.save(self.X,'x.npy')
       np.save(self.y,'y.npy')
```

## 5. Feature Extraction:

```
N_FRAMES = 10
from feature_extractor import FeatureExtractor
f = FeatureExtractor(N_FRAMES,'modal.hdf5')
f.load_from_directory('./data/video_data')
X, y = f.load_dataset()
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False)
y = y.reshape(len(y), 1)
y = encoder.fit_transform(y)
import numpy as np
np.save("features/lstm_x.npy",X)
np.save("features/lstm_y.npy",y)
```

## 6. Preparing Data For Training Bi-LSTM:

```
import numpy as np
X, y = np.load('lstm_x.npy'),np.load('lstm_y.npy')
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
```

## 7. Preparing The Model Bidirectional LSTM:

```
from keras.layers import Bidirectional,LSTM,Dense,Dropout,Input

from keras.models import Sequential

model = Sequential()

model.add(Bidirectional(LSTM(2048, return_sequences=False,

                input_shape=X_train.shape[1:],#(N_FRAMES,512),

                dropout=0.5)))

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(4, activation='softmax'))
```

## 8. Training The Model Bidirectional LSTM:

```
model.compile(optimizer='adam',metrics=['accuracy'],loss="categorical_cros
sentropy")

history=model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=5
)
```

## 9. Displaying Training Statistics:

```
from matplotlib import pyplot as plt

print(history.history.keys())

from matplotlib import pyplot as plt

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

plt.plot(history.history['val_acc'])
```

```
plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'])#, loc='upper left')

plt.show()

# summarize history for loss

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

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

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'])#, loc='upper left')

plt.show()
```

## 10. Saving Bi-LSTM Model:

```
model.save('bi-lstm.hdf5')

!cp bi-lstm.hdf5 drive/My\ Drive/Colab\ Notebooks/
```

## 11.Evaluating Bi-LSTM Model:

```
loss,accuracy = model.evaluate(X_train,y_train)

print("accuracy on train dataset %.3f\nLoss %.3f"%(accuracy*100,loss*100))
```

# CHAPTER-8

# TESTING

## 8. Testing:

## Testing Introduction:

## System Testing:

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

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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 or System Test.Entered 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 test 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 test in which the software

under test is treated as a black box .You cannot “see” 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 life cycle, 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.

# CHAPTER-9

# RESULTS

## 9. Results:

## 1. Command to run script:

```
$ python3.7 classifier.py -i 'mix_video0001-0174.mp4' -o 'output.mp4'
-n 10 -b './models/bi-lstm.hdf5' -c './models/model.hdf5' -l 'Golf Swi
nging' 'Skate Boarding' 'Runnig' 'Horse Riding'
```

**Fig:13. Showing the Command how to run Classifier Script**

## 2. CNN Architecture:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
Output_Layer (Dense)         (None, 4)                 2052
=================================================================
Total params: 14,979,396
Trainable params: 264,708
Non-trainable params: 14,714,688
_________________________________________________________________
```

**Fig:14. Picture Showing the Layers used in Model CNN**

## 3. Training CNN:

```
[ ] 9/9 [==============================] - 257s 29s/step - loss: 0.5347 - acc: 0.8307 - val_loss: 0.8821 - val_acc: 0.6150
    Epoch 5/20
    9/9 [==============================] - 257s 29s/step - loss: 0.5032 - acc: 0.8315 - val_loss: 0.6234 - val_acc: 0.7183
    Epoch 6/20
    9/9 [==============================] - 271s 30s/step - loss: 0.3936 - acc: 0.8785 - val_loss: 0.5793 - val_acc: 0.7559
    Epoch 7/20
    9/9 [==============================] - 243s 27s/step - loss: 0.3092 - acc: 0.9218 - val_loss: 0.6161 - val_acc: 0.7746
    Epoch 8/20
    9/9 [==============================] - 271s 30s/step - loss: 0.2927 - acc: 0.9097 - val_loss: 0.4265 - val_acc: 0.8216
    Epoch 9/20
    9/9 [==============================] - 243s 27s/step - loss: 0.3475 - acc: 0.8709 - val_loss: 0.5179 - val_acc: 0.7746
    Epoch 10/20
    9/9 [==============================] - 272s 30s/step - loss: 0.2897 - acc: 0.9097 - val_loss: 0.3886 - val_acc: 0.8826
    Epoch 11/20
    9/9 [==============================] - 250s 28s/step - loss: 0.2087 - acc: 0.9614 - val_loss: 0.4922 - val_acc: 0.7840
    Epoch 12/20
    9/9 [==============================] - 256s 28s/step - loss: 0.1510 - acc: 0.9789 - val_loss: 0.3422 - val_acc: 0.8732
    Epoch 13/20
    9/9 [==============================] - 271s 30s/step - loss: 0.1748 - acc: 0.9653 - val_loss: 0.3012 - val_acc: 0.8920
    Epoch 14/20
    9/9 [==============================] - 242s 27s/step - loss: 0.1385 - acc: 0.9573 - val_loss: 0.4545 - val_acc: 0.8216
    Epoch 15/20
    9/9 [==============================] - 256s 28s/step - loss: 0.1549 - acc: 0.9509 - val_loss: 0.2359 - val_acc: 0.9249
    Epoch 16/20
    9/9 [==============================] - 271s 30s/step - loss: 0.1521 - acc: 0.9514 - val_loss: 0.3015 - val_acc: 0.8920
    Epoch 17/20
    9/9 [==============================] - 243s 27s/step - loss: 0.0955 - acc: 0.9822 - val_loss: 0.3408 - val_acc: 0.8545
    Epoch 18/20
    9/9 [==============================] - 270s 30s/step - loss: 0.1103 - acc: 0.9757 - val_loss: 0.2531 - val_acc: 0.8920
    Epoch 19/20
    9/9 [==============================] - 256s 28s/step - loss: 0.1083 - acc: 0.9789 - val_loss: 0.2885 - val_acc: 0.9014
    Epoch 20/20
    9/9 [==============================] - 257s 29s/step - loss: 0.0895 - acc: 0.9895 - val_loss: 0.2249 - val_acc: 0.9155
```

**Fig:15. Picture Showing the Training Log of CNN Model**

## 4. Feature Extraction:

```
[ ] f.load_from_directory('./data/video_data')
```

```
[INFO] ./data/video_data/Golf Swinging/7616-7_70270.avi completed
[INFO] ./data/video_data/Golf Swinging/RF1-13678_70045.avi completed
[INFO] ./data/video_data/Golf Swinging/RF1-13209_70050.avi completed
[INFO] ./data/video_data/Golf Swinging/RF1-13588_70046.avi completed
[INFO] ./data/video_data/Golf Swinging/RF1-13428_70288.avi completed
[INFO] ./data/video_data/Golf Swinging/3283-8_701201.avi completed
[INFO] ./data/video_data/Golf Swinging/7603-4_70159.avi completed
[INFO] ./data/video_data/Golf Swinging/7608-12_70275.avi completed
[INFO] ./data/video_data/Golf Swinging/RF1-13157_70040.avi completed
[INFO] ./data/video_data/Golf Swinging completed
[INFO] ./data/video_data/SkateBoarding/947-70005.avi completed
[INFO] ./data/video_data/SkateBoarding/1058-22003.avi completed
[INFO] ./data/video_data/SkateBoarding/860-37150.avi completed
[INFO] ./data/video_data/SkateBoarding/711-66044.avi completed
[INFO] ./data/video_data/SkateBoarding/708-77009.avi completed
[INFO] ./data/video_data/SkateBoarding/761-39000.avi completed
[INFO] ./data/video_data/SkateBoarding/708-75070.avi completed
[INFO] ./data/video_data/SkateBoarding/860-2729.avi completed
[INFO] ./data/video_data/SkateBoarding/947-58108.avi completed
[INFO] ./data/video_data/SkateBoarding completed
[INFO] ./data/video_data/Running/5117-8_70157.avi completed
[INFO] ./data/video_data/Running/5238-17_701581.avi completed
[INFO] ./data/video_data/Running/5238-17_700000.avi completed
[INFO] ./data/video_data/Running/6065-8_70110.avi completed
[INFO] ./data/video_data/Running/5238-17_701141.avi completed
[INFO] ./data/video_data/Running/2670-5_70111.avi completed
[INFO] ./data/video_data/Running/7850-5_70090.avi completed
[INFO] ./data/video_data/Running/5238-17_700950.avi completed
[INFO] ./data/video_data/Running/3687-17_70245.avi completed
[INFO] ./data/video_data/Running completed
[INFO] ./data/video_data/Riding Horse/6018-29_70000.avi completed
```

**Fig:16. Picture Showing the output of Feature Extraction**

## 5. Bi-LSTM Architecture:

```
[ ] model.summary()

Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_5 (Bidirection (None, 4096)              41959424
_________________________________________________________________
dense_9 (Dense)              (None, 512)               2097664
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 4)                 2052
=================================================================
Total params: 44,059,140
Trainable params: 44,059,140
Non-trainable params: 0
_________________________________________________________________
```

**Fig:17. Picture showing the Layers of Model Bi-LSTM**

## 6. Training Bi-LSTM:

```
history = model.fit(
          X_train,
          y_train,
          validation_data=(X_test, y_test),
          epochs=5)

Train on 643 samples, validate on 317 samples
Epoch 1/5
643/643 [==============================] - 97s 150ms/step - loss: 0.5821 - acc: 0.7543 - val_loss: 0.0683 - val_acc: 0.9653
Epoch 2/5
643/643 [==============================] - 91s 142ms/step - loss: 0.1223 - acc: 0.9518 - val_loss: 0.1046 - val_acc: 0.9590
Epoch 3/5
643/643 [==============================] - 90s 141ms/step - loss: 0.0932 - acc: 0.9673 - val_loss: 0.0130 - val_acc: 1.0000
Epoch 4/5
643/643 [==============================] - 92s 143ms/step - loss: 0.0351 - acc: 0.9907 - val_loss: 0.0038 - val_acc: 1.0000
Epoch 5/5
643/643 [==============================] - 92s 143ms/step - loss: 0.0221 - acc: 0.9953 - val_loss: 1.2138e-04 - val_acc: 1.0000
```

**Fig:18. Picture showing Training Log of Bi-LSTM model**

## 7. Save Bi-LSTM:

```
[ ] model.save('bi-lstm.hdf5')

[ ] !cp bi-lstm.hdf5 drive/My\ Drive/Colab\ Notebooks/

[ ] loss,accuracy = model.evaluate(X_train,y_train)

643/643 [==============================] - 15s 24ms/step

[ ] print("accuracy on train dataset %.3f\nLoss %.3f"%(accuracy*100,loss*100))

accuracy on train dataset 100.000
Loss 0.006
```

**Fig:19.Saving of Bi-LSTM model**

## 8. Bi-LSTM Training Accuracy Graph:

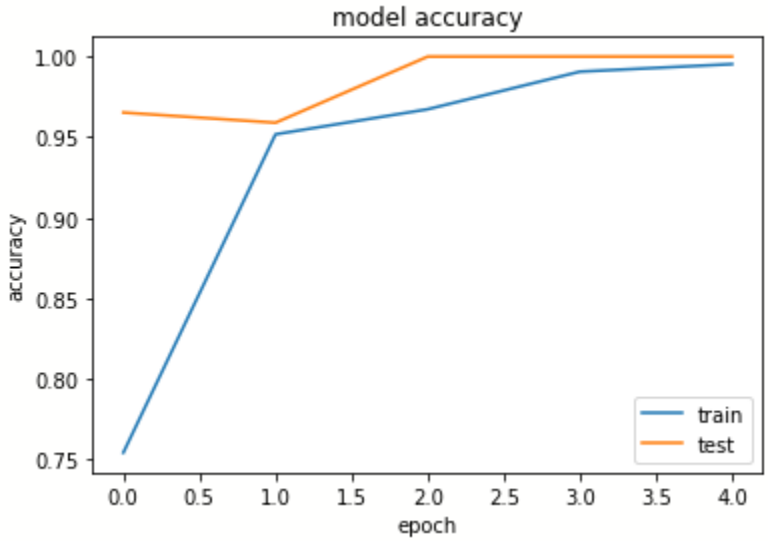


**Fig:20. Data Accuracy Graph of Bidirectional-LSTM**

## 9. Bi-LSTM Training Accuracy Loss Graph:

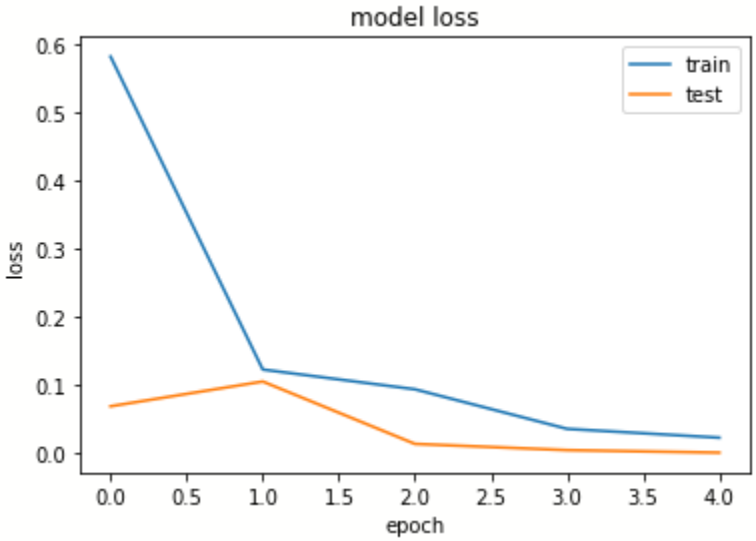


**Fig:21. Training Loss Graph of Bidirectional LSTM**

## 10. Input Video:

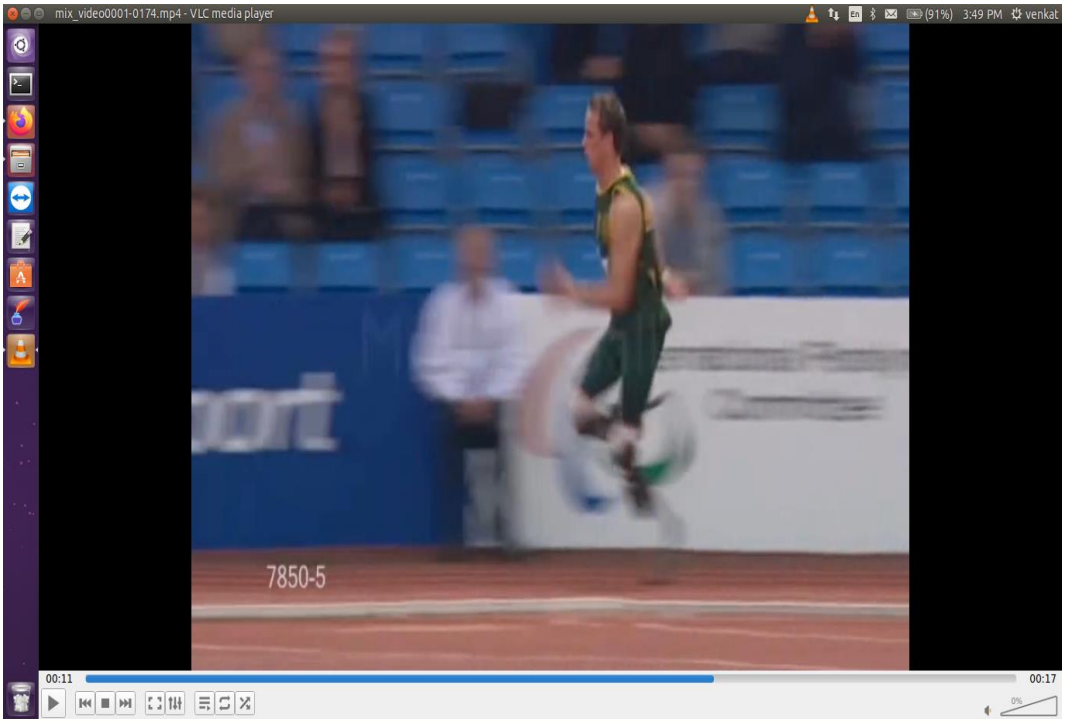


**Fig:22. Pictures Some of Input Videos**

## 11. Video Output:

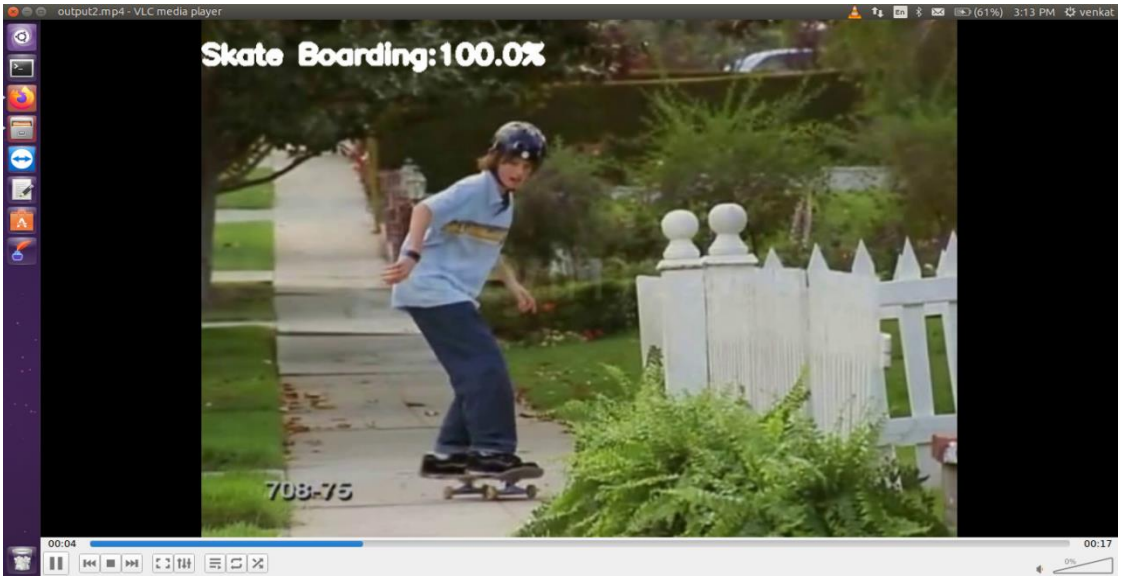


**Fig:23. Pictures of Some of Classified Output Videos**

# CHAPTER-10

# CONCLUSION

## 10. Conclusion:

Action recognition in video sequences is a challenging problem in computer vision. In the context of videos, an action is represented as a sequence of frames, which computers can easily understand by analyzing the contents of multiple frames in sequence. We recognize human actions similar to our observations of actions in real life. In this project only images will be considered as scenes of activities, these images will be used in video classification. these will be designed by using N number of scenes considered as datasets these can be understood as an activity, such activities will be fed into machine learning model using transfer learning for greater accuracy and with the help of this machine learning model the video classification is possible, hence this video classification can be used as activity recognition. We use Bi-LSTM to consider the sequence of information of frames of video in automatic understanding of actions.

# CHAPTER-11

# FUTURE ENHANCEMENTS

## 11. Feature Enhancement:

As technology changes the new requirements are expected by the user,to enhance the functionality of the product may require new versions to be introduced. Although the system is complete and working efficiently, new changes which enhance the system functionality can be added without any major changes to the entire system. In Future if we use Posenet in the place of CNN then Multiple Human Actions will be recognized very Accurately and fastly. Also More Activities can be detected by embedding object detection in our project which is trained for detecting mostly the human, so we can use the output from object detector as input to the activity detector.

# CHAPTER-12

# REFERENCES

# 12. References:

1.https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=81 21994

2.https://towardsdatascience.com/introduction-to-video-classification-6c6acbc57356

3.Video-Classification-CNN-and-LSTM-/train_CNN_RNN.pyatmastersagarvegad/Video-Classification-CNN-and-LSTM-