JSS MAHAVIDYAPEETHA

JSS Science and Technology University

"Human Action Recognition in Videos"

A technical project report submitted in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

BY

Kartik Nagaraj Nayak.	01JST18CS047
Rishitha S Ramesh.	01JST18CS106
Shriya Mittapalli.	01JST18CS133
Vanshika V.	01JST18CS162

Under the Guidance of

Dr. MANIMALA S

Assistant Professor
Department of Computer Science & Engineering
JSS STU Mysore

2021-22

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JSS MAHAVIDYAPEETHA

JSS Science and Technology University

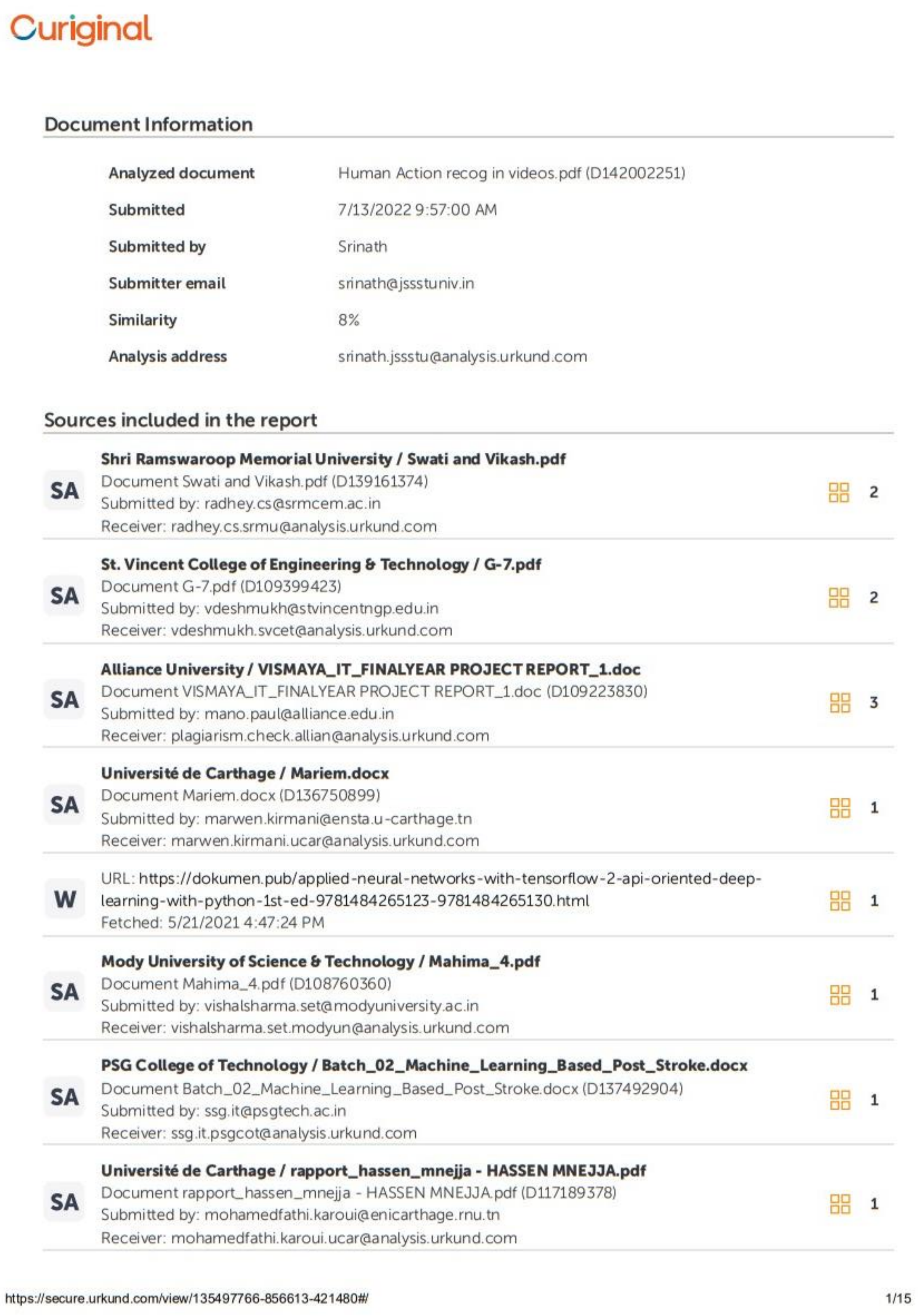
CERTIFICATE

This is to certify that the work entitled "**Human Action recognition in Videos**" is a Bonafide work carried out by Kartik Nagaraj Nayak (01JST18CS047), Rishitha S Ramesh (01JST18CS106), Shriya Mittapalli (01JST18CS133), Vanshika V (01JST18CS162) in partial fulfilment of the award of the degree of Bachelor of Engineering in Computer Science and Engineering for the award of Bachelor of Engineering by JSS Science and Technology University, Mysuru, during the year 2021-2022. The project report has been approved as it satisfies the academic requirements in respect to project work prescribed for the Bachelor of Engineering degree in Computer Science and Engineering.

Under the guidance of	**Head of the Department**
Dr. Manimala S.	**Dr. Srinath S.**
Assistant Professor	Assoc. Prof. and HOD
Dept. of CS &E	Dept. of CS &E
JSS STU Mysore - 06	JSS STU Mysore - 06

Name of Examiner	Signature with Date
1.	....
2.	....
3.	....

Certificate of Plagiarism Check



Curiginal

Document Information

Analyzed document	Human Action recog in videos.pdf (D142002251)
Submitted	7/13/2022 9:57:00 AM
Submitted by	Srinath
Submitter email	srinath@jssstuniv.in
Similarity	8%
Analysis address	srinath.jssstu@analysis.urkund.com

Sources included in the report

SA	**Shri Ramswaroop Memorial University / Swati and Vikash.pdf** Document Swati and Vikash.pdf (D139161374) Submitted by: radhey.cs@srmcem.ac.in Receiver: radhey.cs.srmu@analysis.urkund.com	2
SA	**St. Vincent College of Engineering & Technology / G-7.pdf** Document G-7.pdf (D109399423) Submitted by: vdeshmukh@stvincentngp.edu.in Receiver: vdeshmukh.svcet@analysis.urkund.com	2
SA	**Alliance University / VISMAYA_IT_FINALYEAR PROJECT REPORT_1.doc** Document VISMAYA_IT_FINALYEAR PROJECT REPORT_1.doc (D109223830) Submitted by: mano.paul@alliance.edu.in Receiver: plagiarism.check.allian@analysis.urkund.com	3
SA	**Université de Carthage / Mariem.docx** Document Mariem.docx (D136750899) Submitted by: marwen.kirmani@ensta.u-carthage.tn Receiver: marwen.kirmani.ucar@analysis.urkund.com	1
W	URL: https://dokumen.pub/applied-neural-networks-with-tensorflow-2-api-oriented-deep-learning-with-python-1st-ed-9781484265123-9781484265130.html Fetched: 5/21/2021 4:47:24 PM	1
SA	**Mody University of Science & Technology / Mahima_4.pdf** Document Mahima_4.pdf (D108760360) Submitted by: vishalsharma.set@modyuniversity.ac.in Receiver: vishalsharma.set.modyun@analysis.urkund.com	1
SA	**PSG College of Technology / Batch_02_Machine_Learning_Based_Post_Stroke.docx** Document Batch_02_Machine_Learning_Based_Post_Stroke.docx (D137492904) Submitted by: ssg.it@psgtech.ac.in Receiver: ssg.it.psgcot@analysis.urkund.com	1
SA	**Université de Carthage / rapport_hassen_mnejja - HASSEN MNEJJA.pdf** Document rapport_hassen_mnejja - HASSEN MNEJJA.pdf (D117189378) Submitted by: mohamedfathi.karoui@enicarthage.rnu.tn Receiver: mohamedfathi.karoui.ucar@analysis.urkund.com	1

https://secure.urkund.com/view/135497766-856613-421480#/ 1/15

DECLARATION

We do hereby declare that the project titled **"Human Action Recognition in Videos"** is carried out by the by Kartik Nagaraj Nayak (01JST18CS047), Rishitha S Ramesh (01JST18CS106), Shriya Mittapalli (01JST18CS133), Vanshika V (01JST18CS162), under the guidance of **Dr. Manimala S**, **Assistant Professor,** Department of Computer Science and Engineering, JSS Science and Technology University, Mysuru, in partial fulfilment of requirement for the award of Bachelor of Engineering by JSS Science and Technology University, Mysore, during the year 2021-2022.

We also declare that we have not submitted this dissertation to any other university for the award of any degree or diploma courses.

Date: 16 July 2022
Place: Mysore

Kartik Nagaraj Nayak (01JST18CS047)
Rishitha S Ramesh (01JST18CS106)
Shriya Mittapalli (01JST18CS133)
Vanshika V (01JST18CS162)

ACKNOWLEDGEMENT

We express our gratitude to Dr. S.B. Kivade, Honourable Principal of JSS Science and Technology University, for providing a wonderful educational atmosphere and for his encouragement during our college years. We would like to express our deepest gratitude to our HOD, Dr. Srinath S for allowing us to pursue this project. We would like to express our gratitude to Dr. Manimala S, our project guide, for her helpful direction and passionate assistance, as well as for giving us with support and constructive recommendations for the project's improvement, without which this project would not have been feasible. We thank our lecturers' prompt assistance and gracious collaboration, as well as that of other department staff members, with whom we have worked closely throughout our project work and without whose help this project would not have been a success. Finally, we'd like to express our gratitude to our friends for their countless excellent recommendations. We also want to express our gratitude to everyone who has helped to make this learning opportunity possible at every stage of the project.

Thank you,

Kartik Nagaraj Nayak

Rishitha S Ramesh

Shriya Mittapalli

Vanshika V

ABSTRACT

The technique of discovering and understanding human activities is known as human action recognition. The major goal of this procedure is to recognize people in films that may be utilized for a variety of applications. It can be used for security, surveillance, and other operations. Our video classifier is based on the UCF101 dataset. The dataset contains films of various acts, such as playing the guitar, punching, bicycling, and so on. This dataset is widely used in the construction of action recognizers, a kind of video classification application. We introduce a class of end-to-end trainable recurrent convolutional architectures that are ideal for optical comprehension tasks and illustrate how helpful these models are for action detection. In contrast to past models that assumed a fixed visual representation or conducted simple temporal averaging for sequential processing, recurrent convolutional models develop compositional representations in space and time. To understand temporal dynamics and convolutional perceptual representations, our recurrent sequence models may be trained together. We'll evaluate two models: LRCN and ConvLSTM, to see which one performs the best.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Human Action Recognition in Videos is the ability to recognize human actions, such as walking, running and jumping. It can be used for a variety of applications including surveillance, security and detection of crime. The video data is usually captured by an on-board camera or from a networked camera system.

The recognition process works on the basis of learning models that are trained using labelled videos with known actions (e.g., walking). These models are then applied to new video sequences which are similar enough to be classified correctly.

Human action recognition is the process of extracting visual features from a video and using them to recognize human actions. It is an important part of computer vision and machine learning, as it allows computers to identify people in videos. The human action recognition system is a computer vision algorithm that recognizes the actions of humans in videos. In other words, it can recognize what people are doing in video clips and extract information such as their facial expressions, gestures, body postures and movements.

It was developed by researchers at Microsoft Research Asia (MSRA) in collaboration with researchers from the University of California San Diego (UCSD), Stanford University, and National Taiwan University. The research team used this technology to develop an interactive game called “Human Action Recognition Challenge” which allows users to play against each other using images of real-life scenes captured by Kinect.

1.1 Problem Statement

Human Action Recognition (HAR), is the problem of predicting what a person is doing based on a trace of their movement using sensors. Activity detection is a major problem in computer vision, i.e. to detect the activity of humans. Task of tracking and understanding what is happening in video is very challenging. Given a video in which a person is performing an action, develop an automatic system to recognize that action.

1.2 Aim and Objectives

To build an Artificial Intelligence based human action recognition system in videos using ConvLSTM and LRCN, and compare the results.

1. Learn visual features from video frames.
2. Detect specific motions/ Recognize human actions.
3. Classify human action from video sequences.
4. Boost the understanding of the ongoing event.
5. Compare the performance of two models.

1.3 Applications

The main idea behind the application of Human Action Recognition is the use of deep neural networks for recognizing objects or events in images or videos. Deep neural networks are trained by feeding them millions upon millions of examples from which they learn how to identify certain features that make up an object or event.

1. Identifying users' activity levels that helps in predicting their energy consumption and promoting health and fitness.
2. Detecting a fall and the movements of the user after the fall.
3. Automatic behaviour modification of the mobile device based on user activity.
4. Enhanced localization based on context discovered utilizing data from activity.
5. Personal Biometric Signature.

1.4 Existing Solution

Method	Title of the Paper	Year	Limitations
2D CNN	Human Activity Recognition using Accelerometer and Gyroscope Data from Smartphones	2020	Limited activity recognition, do not consider the impact of carrying cell phones in different locations.
Single Frame CNN	Large-scale Video Classification with Convolutional Neural Networks	2014	Neglects the temporal relationships between frame sequences and fails to employ a pre-trained model to extract features and use them in other deep learning models.
CNN+RNN	Video Classification using Machine Learning	2020	Transfer learning of a pre-trained Deep Learning model to our system loses association with spatial and temporal data.

Table 1.1: EXISTING SOLUTION METHODS

1.5 Proposed Solution

A video model should ideally be able to handle input sequences of different lengths and offer outputs of different lengths, such as the creation of full-length phrase descriptions that go beyond traditional one-versus-all prediction tasks. Hence, we propose Long-term Recurrent Convolutional Networks (LRCNs) and Convolutional LSTM (ConvLSTM), a class of architectures for visual recognition which combines convolutional layers and long-range temporal recursion and is end-to-end trainable.

A class of models known as LRCN (Figure 1. 1) is deep in both spatial and temporal and robust enough to be used for a range of vision tasks involving sequential inputs and outputs. This approach consistently demonstrates that by learning sequential dynamics with a deep sequence model, we can enhance existing techniques that accept a fixed visual representation of the input and merely learn the dynamics of the output sequence, as well as those that learn a deep hierarchy of parameters in the visual domain.

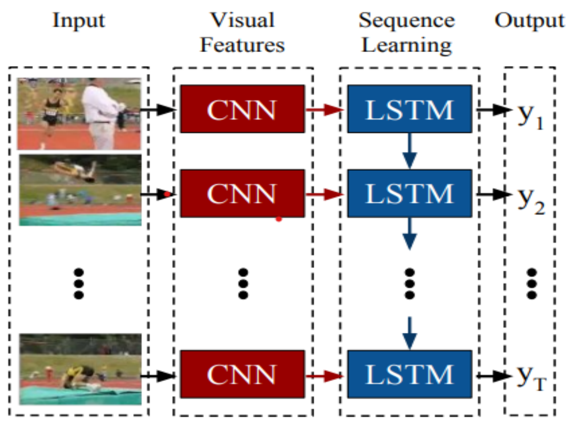


Figure 1.1: LRCN Model

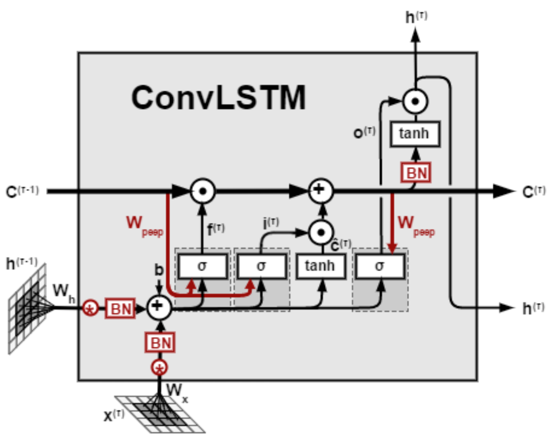


Figure 1.2: ConvLSTM Model

In order to go beyond the tasks requiring static input and predictions, deep sequence modelling methods, such as LRCN and ConvLSTM are increasingly central to vision systems for problems with sequential structure. These methods are a good solution for perceptual issues involving time-varying visual input or sequential outputs since they integrate easily into current visual recognition pipelines and need little to no hand-designed features or input pre-processing.

This mechanism offers a wide variety of action recognitions and considers a vast dataset which is diverse. End-to-end fine tuning of LSTM models is simple. Since LSTMs are not limited to inputs or outputs of a given length, they permit efficient modelling for sequential data with variable durations, such as text or video, and considers spatial and temporal features in the video frames. Model will be trained to predict considering spatial and temporal features simultaneously and hence will perform better.

A recurrent neural network for spatiotemporal prediction called ConvLSTM (Figure 1.2) uses convolutional structures in both the input-to-state and state-to-state transitions. By using the inputs and previous states of its local neighbours, the ConvLSTM predicts the future state of each cell in the grid.

1.6 Gantt Chart

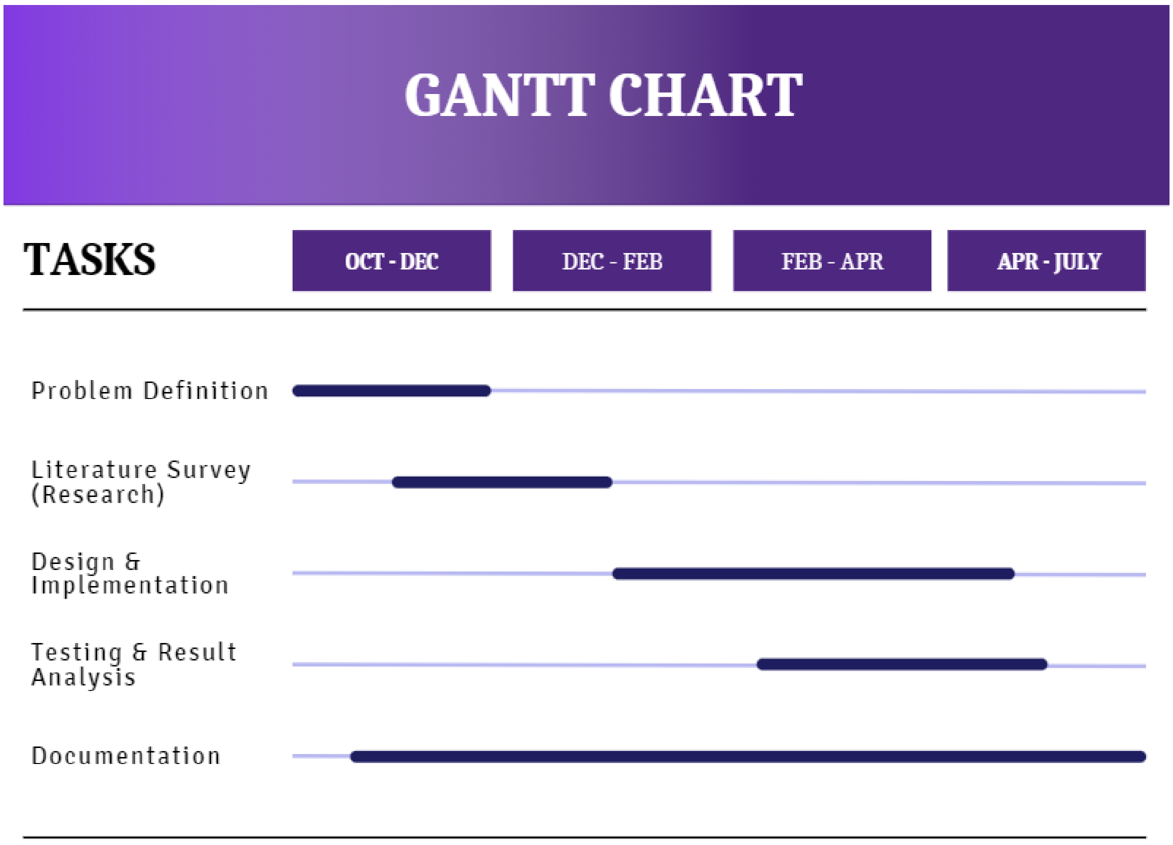


Figure 1.3: Gantt Chart

CHAPTER 2
LITERATURE REVIEW

A key driving factor in video recognition research has been the development of image recognition algorithms, which have been repeatedly changed and enlarged to cope with video data. For example, [1]'s technique involves recognising sparse spatiotemporal interest locations, which are subsequently represented using local spatiotemporal features: Histogram of Oriented Gradients [3] and Histogram of Optical Flow. The traits are then stored in a Bag of Features representation, which is pooled over many spatiotemporal grids and combined with an SVM classifier. Dense sampling of local characteristics beats sparse interest locations, according to a subsequent study [2].

State-of-the-art shallow video representations [5] employ dense point trajectories instead of calculating local video characteristics over spatio-temporal cuboids. The method, which was initially described in [7], entails changing local descriptor support areas to follow dense trajectories estimated using optical flow. The Motion Boundary Histogram (MBH) [4], a gradient-based feature computed independently on the horizontal and vertical components of optical flow, has the highest performance in the trajectory-based pipeline. Two recent breakthroughs in trajectory-based hand-crafted representations (in [5]) are the compensation of global (camera) motion [5, 9] and the application of the Fisher vector encoding [8].

Attempts to design a deep learning model for video recognition have also been numerous. Because the majority of these experiments employ a stack of consecutive video frames as input, the model must learn spatio-temporal motion-dependent properties implicitly in the initial layers, which might be problematic. [6] presented an HMAX architecture with predetermined spatiotemporal filters in the first layer for video recognition. It was then merged [9] with an HMAX model to create spatial and temporal identification channels.

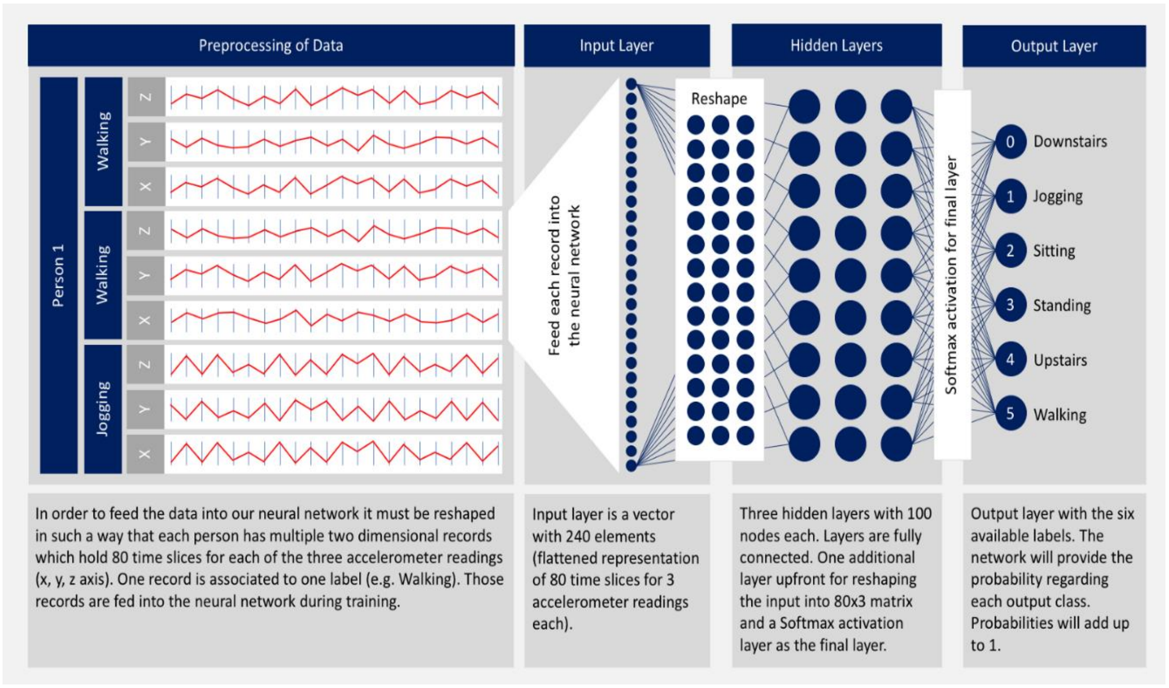
The temporal stream ConvNet works with multiple frame dense optical flow, which is commonly computed by solving for a displacement field in an energy minimization framework. A typical approach [1] for expressing energy based on intensity and gradient constancy assumptions, as well as smoothness of the displacement field. There have been efforts in [10] to use sensor data from smartphones to detect human activity. These uses the spatial movements of a person to detect his actions. Deep neural networks are also being used.

Many attempts have been done over a long period of time to identify human action, and the field continues to evolve as computer vision improves. Soon we will have a technology that accurately predicts human behaviours and actions.

2.1 Human Activity Recognition using Accelerometer and Gyroscopic Data from Smartphones

The review paper titled “Human Activity Recognition using Accelerometer and Gyroscope Data from Smartphones” by Khimraj, Praveen Kumar Shukla, Ankit Vijayvargiya, and Rajesh Kumar revolves around recognizing human activities. It uses a 2D CNN Model to predict the actions.

Acceleration signals obtained from accelerometers have been used for HAR, due to their robustness against occlusion, viewpoint, lighting, and background variations, etc. A tri-axial accelerometer, in particular, may provide an estimate of acceleration along the x, y, and z axes, which can be used to analyse human activities.



In order to feed the data into our neural network it must be reshaped in such a way that each person has multiple two dimensional records which hold 80 time slices for each of the three accelerometer readings (x, y, z axis). One record is associated to one label (e.g. Walking). Those records are fed into the neural network during training.

Input layer is a vector with 240 elements (flattened representation of 80 time slices for 3 accelerometer readings each).

Three hidden layers with 100 nodes each. Layers are fully connected. One additional layer upfront for reshaping the input into 80x3 matrix and a Softmax activation layer as the final layer.

Output layer with the six available labels. The network will provide the probability regarding each output class. Probabilities will add up to 1.

Figure 2.1: 2D CNN for Accelerometer Data

Although each person's body size and proportions are unique, most people perform actions in qualitatively similar ways, hence the acceleration signal often does not show apparent intra-class variances for the same movement.

HAR using acceleration signal achieves a high accuracy, and thus has been adopted for remote monitoring systems, while taking care of privacy issues. Acceleration-based HAR has frequently employed deep learning networks. However, the subject needs to carry wearable sensors that are often cumbersome and disturbing. In addition, the position of the sensors on the human body can also affect the HAR performance. This approach does not consider the characteristic features of the surroundings which plays an important role in action recognition.

This work is significant because the activity recognition model permits us to gain useful knowledge about the habits of millions of users passively—just by having them carry cell phones in their pockets. It has a wide range of applications, including automatic customization of the mobile device’s behaviour based upon a user’s activity (e.g., sending calls directly to voicemail if a user is jogging) and generating a daily/weekly activity profile to determine if a user (an obese child) is performing a healthy amount of exercise.

2.2 Action Recognition Using Single Frame CNN

The review paper titled “Large-scale Video Classification with Convolutional Neural Networks” by Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar and Li Fei-Fei revolves around developing a deep learning model that uses Single Frame Convolution Neural Networks to recognize human actions in the video.

Using an image classifier on each frame of the video and classifying the actions in each frame separately is the simplest and most fundamental method of categorising activities in a video. We will run an image classification model on every single frame of the video and then average all the individual probabilities to get the final probabilities vector.

Despite the method's effectiveness, it is important to note that it disregards the temporal relationships between the frames in the sequence. The lack of consideration for the temporal component of the data renders this method ineffective.

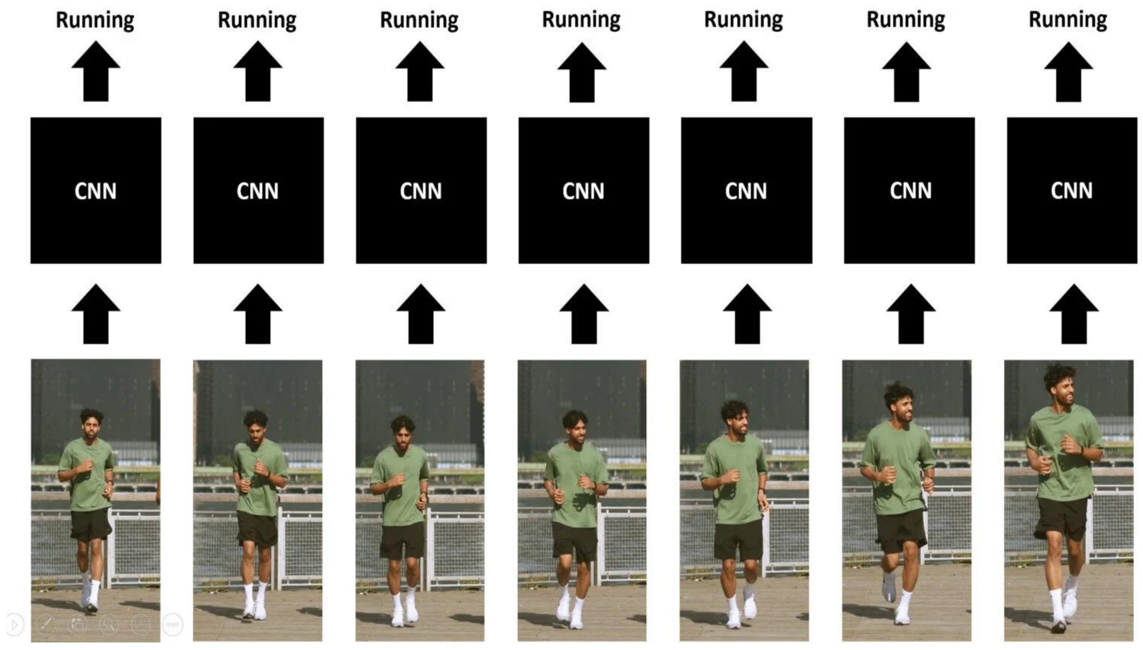


Figure 2.2: Single Frame CNN

2.3 Action recognition using CNN-RNN Architecture

The state of the art of deep learning activity recognition suggests that the best way to tackle this problem is through a model with a Convolutional Neural Network, at the beginning, to extract the features of the video frames, followed by a Recurrent Neural Network that can model frame sequences. This approach involves feeding the CNN, which can extract visual attributes, video frames. All these features are then fed into a RNN

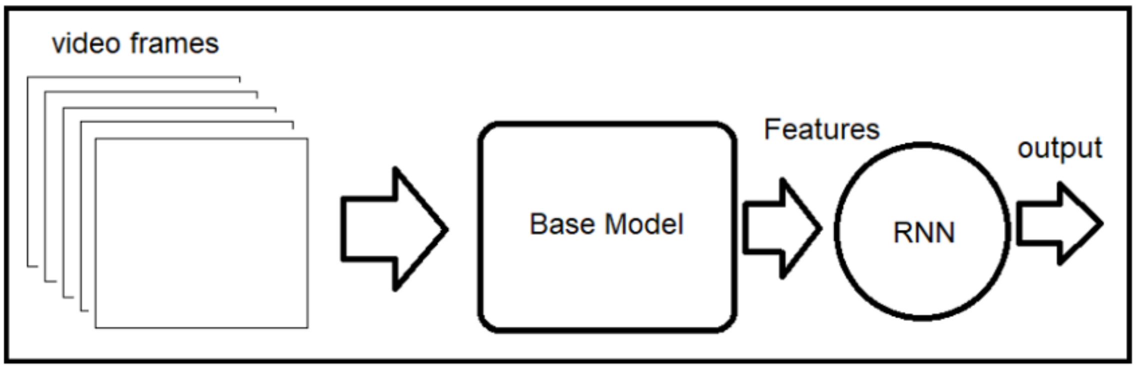


Figure 2.3: CNN-RNN Architecture

For this project, as a base model, we will use the extraction feature part of a pre-trained model called transfer learning which focuses on storing knowledge and applying it to a specific but related question. The model we will be using is Inception v3 because it has excellent classification accuracy and low cost of computation. To extract high-level characteristics, a CNN model is given the video's pictures. The RNN layer's output is connected to a fully connected layer to produce the classification output once the features have been provided to it.

All these methods have their own limitations. With each successive method we try to overcome the limitation of previous methods. So we propose ConvLSTM and LRCM models which tries to integrate both spatial and time data.

CHAPTER 3
SYSTEM REQUIREMENTS AND ANALYSIS

The hardware and software components of a computer system that are necessary for installing and using software effectively are referred to as system requirements. The suggested system requirements, if fulfilled, will improve the product's usability. The recommended system requirements must be satisfied for the software to run on a system at all.

A system's or a component's function is defined by a functional requirement, where a function is defined as the behaviour between outputs and inputs. Functional requirements specify what a system should be able to do through computations, technical details, data manipulation and processing, and other specialised functions.

Non-functional requirements, which place restrictions on the design or execution, complement functional needs. An evaluation of a system's performance that uses criteria rather than particular actions is known as a non-functional requirement. Quality criteria and quality characteristics are other names for non-functional requirements.

3.1 Product Perspective

In this project, we have proposed a highly robust and efficient mechanism for human action classification. The proposed system has been emphasised on developing an efficient scheme that can accomplish activity recognition. Recognizing human activity is important for interpersonal interactions and human-to-human communication. And an ability to interpret the movements of the human body via sensors and to determine human action. Given a video in which a person is performing an action, developing an automatic system to recognize that action.

The major components of this system are:

1.Pre-processing
2.Feature Extraction
3.Feature Representation
4.Action Classification

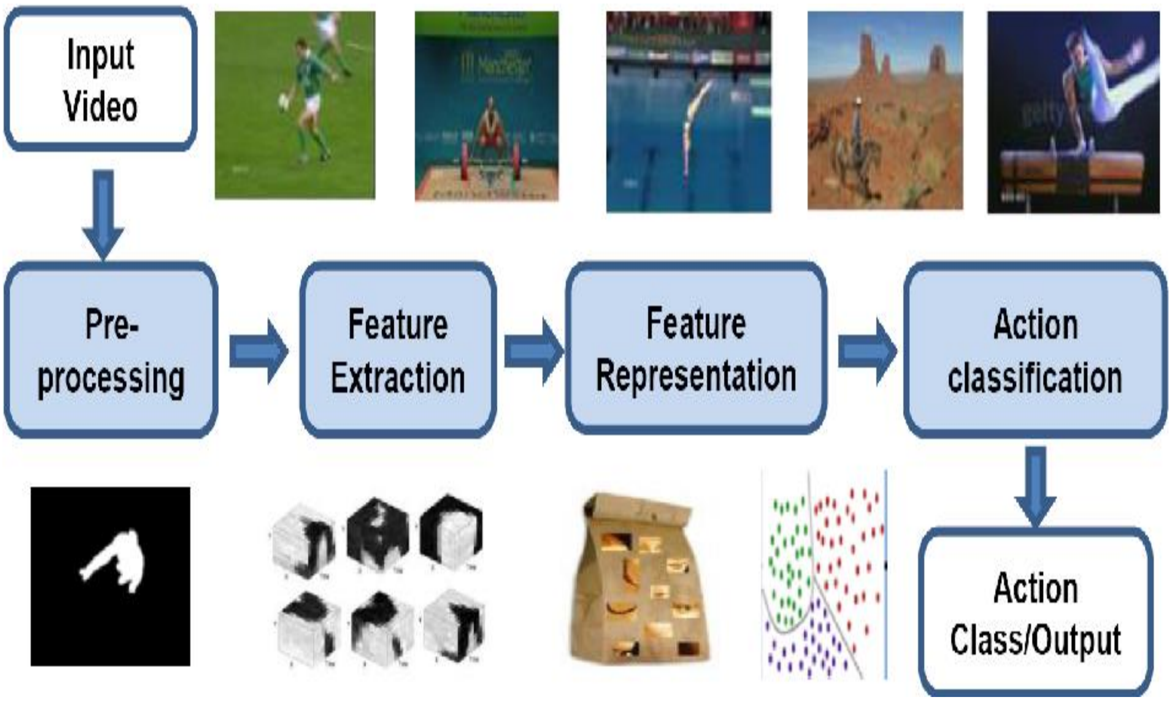


Figure 3.1: Major Components

3.2 Product Functions

1. **Pre-Processing**

   An approach of transforming raw data into something that can be used by a machine learning model. It is the first and most crucial step in the process of creating a machine learning model. Additionally, in working with data, it is necessary to clean it up and format it. Action Recognition System tests a few pre-processing processes before choosing one of them to continue the action recognition process.

2. **Feature Extraction**

   Choose which traits will be used for learning. We identify the action accurately using the spatial and temporal information seen in the video.

3. **Feature Representation**

   A group of methods that enable a system to automatically extract from raw data the representations required for feature categorization. Using thick layers, the features are fully linked after being represented as feature maps.

4. **Action Classification**

   Classification of human activities from video. The method uses information directly from the video sequence. It exploits spatial information using CNN and temporal information using LSTM networks.

3.3 Operating Environment

The server will be running on any device capable of running machine learning algorithms irrespective of the OS. The Python environment is used to train and test the model. Users can use the model with any device with an active internet connection and compatible browser to access the website.

3.4 Design and Implementation Constraints

Model must be developed using Tensor flow and Keras libraries available in python. Mechanism to download YouTube videos and live capturing of videos must be provided. We need to create a strong model that can handle wide ranges of camera movements, item position and appearance, scale, perspective, cluttered backdrop, lighting conditions, etc.

3.5 Assumptions and Dependencies

Assumptions:
The basic assumption is that the video falls into one of the classes the model is trained for. The video contains only one actor performing the action.

Dependencies:

- OpenCV
- Numpy
- Tensorflow
- Keras
- PyTube
- SkLearn

3.6 User Interface Requirements

The software developed is accessible to the end-user as a website, which is a very intuitive and simple interface to the user and it follows all the website accessibility guidelines.

The user inputs data via the website. The input can be provided in the following way:

- Link to the video and download.

3.7 Hardware Requirements

To develop this software, the user needs to possess a hardware device with below listed configuration.

- CPU, 1.00GHz and above.
- RAM, 8GB and above.
- Good GPU for Video Processing.
- Camera to capture real time video.

3.8 Software Requirements

The system requires the following software support to effectively develop the model.

- Any OS.
- Python support.
- Jupyter Notebook/Google Colab.
- Browser: Google Chrome (v96.0.4664.110 and above), Firefox (v95.0 and above), Safari (v15.0 and above), Microsoft Edge (v96.0.1054.62 and above).
- Active Internet Connection.

3.9 Functional Requirements

1. Collection and Pre-processing of Dataset

This involves collection of video dataset which contains labelled videos of different classes of human actions. After collection of datasets, they have to be processed to facilitate training. This function has high priority. Benefit - 8/9, cost - 7/9, risk - 2/9.

REQ-1: Collection of dataset from the various datasets available on the internet. Different datasets available have different classes of actions. We select a dataset that has a collection of realistic action films with a wide range of camera movements, object size, object scale, perspective, cluttered backdrop, lighting conditions, etc.

REQ-2: The collected videos may have different resolutions, frame count etc. For ease of training we collect a specific number of frames from each video distributed throughout the video and resize them to fixed size.

REQ-3: After bringing all the frames to the same format, they are normalised and are converted to arrays so that they become useful features for training. And later they are saved in a file for future training.

2. Training the model

This involves setting hyper parameters and determining parameters through training the model over datasets. The priority of this function is high. Benefit - 9/9, cost - 8/9, risk - 5/9.

REQ-1: Build the LRCN model with CNN and LSTM and define the model structure for optimal accuracy and minimizing the cost function.

REQ-2: Tuning the hyper parameters to get best result and using algorithms, and activation functions like softmax, relu, sigmoid.

3. Testing the model

This involves testing the model over testing datasets. The priority of this function is moderate. Benefit - 7/9, cost - 5/9, risk - 3/9.

REQ-1: Collecting test videos from YouTube. Later, extracting required features in desired structure.

REQ-2: Testing the processed test data in with the model generated.

REQ-3: The model should be able to predict the action being performed in the video.

4. Other User Interface Related Functions

This includes functions for actions like capturing real time video, download/display video, etc. The priority of this function is low. Benefit - 3/9, cost - 5/9, risk - 2/9.

REQ-1: The system should have functions to download YouTube videos and convert them to required format.

REQ-2: The system should be able to allow functions to display the video.

3.10 Non-Functional Requirements

1. Performance Requirements

We aim to classify the video in real time. Because of that the classification action should take place within seconds. We also aim at training the model with high performance.

2. Software Quality Attributes

Functionality: The project is designed to classify the action being performed in video.

Reliability: The application is believed to work in all computers which support videos and python modules.

Flexibility: The model will undergo extensive training on a variety of settings, including changes in camera motion, item stance and appearance, size, perspective, cluttered backgrounds, lighting conditions, etc.

Portability: The system works on major OS and also on cloud based platforms.
Availability: As long as the application is installed in a computer, the application will be available to the user.

Usability: The interface will be designed to make users easy to operate on.

Maintainability: Once a correct model is trained, maintaining it is easy and feasible.

Adaptability: The model can classify action on any given video. It can adapt to various object appearances, camera motions, illuminations, viewpoints, etc.

Correctness: The correctness of the model can be measured using accuracy that we obtain from testing the dataset.

Reusability: The trained model can be reused in various applications where action recognition is required.

Testability: The trained model can be tested with various videos from YouTube, gifs, real time dynamic videos.

3.11 Domain Requirements

This project requires good knowledge of

- Deep Learning.
- Python.
- Working on Jupyter Notebook/Google Colab.
- Keras library.
- OpenCV.
- Numpy.

Deep Learning concepts needed to implement this project are

- Multi-layered Perceptron.
- Convolution Neural Networks.
- LSTM.
- Long-term Recurrent Convolutional Networks.

CHAPTER 4
TOOLS AND TECHNOLOGY

Using suitable equipment, we make any implementations less complicated than imposing the whole thing from scratch. This bankruptcy offers a quick perception into the equipment and technology which might be used for the venture work.

1. Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its built-in high-level data structures combined with dynamic typing and dynamic binding make it very attractive for rapid application development, as well as for use as a scripting or assembly language to link existing components together. Python's syntax is easy-to-learn, emphasizes readability and therefore reduces program maintenance costs. The Python interpreter and extensive standard library are available in source or binary form for free for all major platforms and are freely redistributable The source-level debugger allows checking local and global variables, evaluating arbitrary expressions, setting breakpoints, stepping through code line by line, and so on. The debugger is written in Python itself, a testament to Python's introspective power. The implementation is done in Python. Python is an interpreted, interactive, and high-level object-oriented scripting language. Python is designed to be highly readable. It often uses English words while other languages use punctuation and has fewer syntactic constructions than other languages.

2. Numpy: A Python library used for working with arrays. It also has functions for running with inside the area of linear algebra, Fourier transform, and matrices. Numpy is a Python library for large-scale numerical computations. It includes the core functionality of Python's ndarray module, but also provides many additional features such as: Faster matrix operations (e.g., transpose) by using efficient algorithms based on Data Parallelism and Multiprocessing. New functions to work with sparse matrices (e.g., reshape). A high-performance linear algebra submodule that implements fast vectorized routines for common linear algebra operations like dot products, matrix multiplication, etc. This makes it possible to use numpy in applications where speed is crucial

3. OpenCV: OpenCV is a huge open source library for computer vision, machine learning and image processing and now plays a major role in real-time operation, which is very important in today's systems. It can be used to process images and videos to identify objects, faces or even a person's handwriting.

When integrated with various libraries such as NumPy,
python is able to handle the OpenCV array structure for parsing. To identify an image pattern and its various elements, we use a vector space and perform mathematical operations on these elements. The library has more than 2500 optimized algorithms that include a comprehensive set of classical and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movement, track moving objects, extract 3D object models, create 3D point clouds from stereo cameras, stitch images together to create high-resolution images. image of the entire scene, search for similar images in the image database, remove red-eye from flash images, track eye movements, recognize scenery and create markers to overlay it with augmented reality, etc.

4. Tensorflow: Tensorflow is a cease-to-cease open supply platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that shall we researchers push the present day in ML and developers easily build and deploy ML powered applications. Tensorflow is a machine learning framework developed by Google, and its open source. It's not just a library of pre-built models, but also an API that allows you to build your own models. Tensorflow is a machine learning framework that allows you to build and deploy machine learning models. Tensorflow has been designed for both high performance and ease of use by programmers, researchers, engineers, data scientists. It can be used for training deep neural networks as well as performing inference on graphs of model weights.

5. Keras: Built on top of Tensorflow 2, Keras is an industry-energy framework that may scale to massive clusters of GPUs or a whole TPU pod. Keras follows first-rate practices for decreasing cognitive load: it gives consistent &, easy APIs, it minimizes the variety of customer actions required for common use cases, and it presents clear & actionable mistake messages. It additionally has good sized documentation and develop.er guides.

6. MatPlotLib: Matplotlib is a complete library for creating static, animated, and interactive visualizations in Python. Matplotlib makes smooth matters smooth and tough matters possible.

7. SkLearn: Open source, commercially usable library built on NumPy, SciPy, and matplotlib provides simple and efficient tools for predictive data analysis. SkLearn is a Python library for machine learning. It provides an easy-to-use interface to the most popular deep learning libraries, including Tensorflow, Theano and CNTK. SkLearn is a Python library for machine learning. It provides: A set of high-level machine learning models, including linear and nonlinear regression, classification, clustering and dimensionality reduction. These are implemented as numpy arrays or tensors (numpy/ndarray) with optional Cython wrappers. An API that allows you to easily construct new models from scratch using the same intuitive syntax used by scikit-learn's Model Builder. A collection of useful tools for data exploration and visualization: plotting functions such as scatter plots, bar charts and histograms; summary statistics such as mean and standard deviation.

8. PyTube: pytube is a lightweight, Pythonic, dependency-free, library (and command-line utility) for downloading YouTube Videos.

9. Neural networks: A computer system that is modelled after the human brain and nervous system. A neural network is a model that mimics the way neurons in the brain work. It consists of multiple layers of nodes, where each layer has its own set of nodes and connections between them. The output from one node feeds into another node, which then passes on information to yet another node, and so on until it reaches an output node. Each input node receives data from other nodes in the network and passes it on to their connected outputs. The number of inputs depends on the number of layers (called hidden layers) you have in your network.

10. CNN: A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm of rules that may absorb input image, assign importance (learnable weights and biases) to diverse aspects/objects within the image and be ability to distinguish one from the other.

11. RNN: A recurrent neural network (RNN) is a category of artificial neural networks wherein connections among nodes form a directed or undirected graph along a temporal sequence. This permits it to show off temporal dynamic behaviour. Derived from feedforward neural networks, RNNs can use their inner state (memory) to process variable length sequences of inputs

12. LSTM: Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture [1] used with inside the area of deep learning. Unlike standard feedforward neural networks, LSTM has remark connections. It can process not only single data points (such as images), however additionally complete sequences of data (such as speech or video).

13. LRCN: It is an aggregate of CNN and RNN, cease-to-cease trainable and appropriate for massive-scale visual understanding tasks such as video description, activity recognition and image captioning. The recurrent convolutional models are deeper in that they research compositional representations in area and time.

14. Jupyter Notebook: Jupyter Notebook is an original web application for creating and sharing computational documents. It offers a simple, efficient document-centric environment. Jupyter Notebook is an open source web application that you can use to create and share documents containing live code, equations, visualizations, and text. Jupyter comes with the IPython core, which allows you to write programs in Python, but there are currently over 100 other cores that can also be used.

Create a copy: You can create duplicate copies of your Jupyter notebook, which will be useful for trying different experiments with your Data Science Projects.

Save As: Used to save the notebook with a specific path. Rename: You can use this command to rename a notebook, or simply double-click the title header instead to perform the same action.

Save and Checkpoint: As the name suggests, it saves your progress and checks the file accordingly. The keyboard shortcut ctrl + s performs an equivalent action.

Checkpoint Rollback: If you ever mess up while experimenting with Jupyter Notebook, this is a handy command to roll back to a previous save and reset your mistakes.

Print Preview: Used to get only a portion of the Jupyter notebook code for clean printing.

Download as: Allows you to download the Jupyter notebook in various formats including HTML, IPython, .py format, etc., among many other options.

Close and Stop: Terminates your currently running Jupyter notebook session and returns you to your local host.

15. Flask: Flask is classified as a micro-framework because it does not require any special tools or libraries. it is not an abstraction layer for a database, form validation, or other component where pre-existing third-party libraries provide common functionality.
However, Flask supports extensions that add functionality to an application as if it were implemented in Flask itself. There are extensions for object-relational mappers, form validation, upload processing, various open authentication technologies, and a number of other common tools related to the framework.

Flask uses the Jinja templating engine to dynamically create HTML pages using familiar Python concepts such as variables, loops, lists, and so on. You will use these templates as part of this project. Despite its origins as a joke, the Flask framework has become very popular as another Django project with its monolithic structure and dependencies. Flask's success generated a lot of other flyers and pull requests. The Pallets Project is now a community-driven organization that maintains Flask and other related Python libraries such as Lektor, Jinja, and many others.

16. HTML: HTML is used in web pages. It can be helped by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or local storage and convert the documents into multimedia web pages. HTML semantically describes the structure of a web page and originally contained guidelines for the appearance of a document.

HTML constructs can be used to insert images and other objects such as interactive forms into a rendered page. HTML provides a means to create structured documents by marking the structural semantics of text, such as headings, paragraphs, lists, links, citations, and other items. HTML elements are delimited by tags written using curly braces. Tags like <img /> and <input /> directly put content on the page. Other tags, such as <p>, surround and provide information about the text of the document and may contain other tags as sub-elements. Browsers do not display HTML tags, but use them to interpret page content.

CSS inclusions define the appearance and layout of content. The World Wide Web Consortium (W3C), the former maintainer of HTML and the current maintainer of CSS standards, has supported the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in cooperation with JavaScript.

17. Bootstrap: Bootstrap is a free and open-source CSS framework focused on responsive, mobile front-end web development, featuring HTML, CSS, JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. As of April 2022, Bootstrap is the eleventh most followed project on GitHub with over 156,000 stars. Bootstrap is an HTML, CSS, and JS library that aims to simplify the development of informative websites (as opposed to web applications).

The primary purpose of adding to a web project is to apply Bootstrap's colour, size, font, and layout choices to that project. So the primary factor is whether the responsible developers find these options to their liking. Once it is added to a project, Bootstrap provides a basic style definition for all the HTML elements. The result is a uniform appearance of prose, tables and form elements across web browsers. In addition, developers can use the CSS classes defined in Bootstrap to further customize the appearance of their content. Bootstrap prepared, for example, light and dark tables, page headers, bolder quotes and highlighted text.

Bootstrap also comes with several JavaScript components that do not require additional libraries such as jQuery. They provide additional user interface elements such as dialog boxes, help, progress bars, navigation dropdowns, and carousels. Each Bootstrap component consists of HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including, for example, the function of automatic completion of input fields. The most distinctive part of Bootstrap is its layouts, as they affect the entire website.

The basic layout component is called a "container" because it contains all the other elements on the page. Developers can choose between a fixed-width container and a fluid container. While the latter always fills the web page width, the former uses one of five predefined fixed widths depending on the screen size displaying the page: Less than 576 pixels and Greater than 1200 pixels. Once the container is in place, the other Bootstrap layout components implement the CSS Flexbox layout by defining rows and columns. The precompiled version of Bootstrap is available in the form of one CSS file and three JavaScript files that can be easily added to any project. However, Bootstrap's raw form allows developers to implement additional customizations and size optimizations. This raw form is modular, meaning a developer can remove unnecessary components, apply a theme, and modify uncompelled Sass files.

CHAPTER 5
SYSTEM DESIGN

System design gives the overall architecture of the solution to the problem. This chapter provides the design of the solution of the problem.

5.1 Architecture

Detailed explanation of CNN, LSTM and ConvLSTM models are given in this Section.

5.1.1. CNN

CNNs are a subset of Deep Neural Networks that are capable of identifying and categorising characteristics in photographs. Convolution in the context of CNN refers to the mathematical operation of multiplying two images, which can be represented as matrices, to provide an output from which features can be extracted from the image. It has many convolution layers with filters, activation functions based on which the image is classified. The process is shown in the figure below.

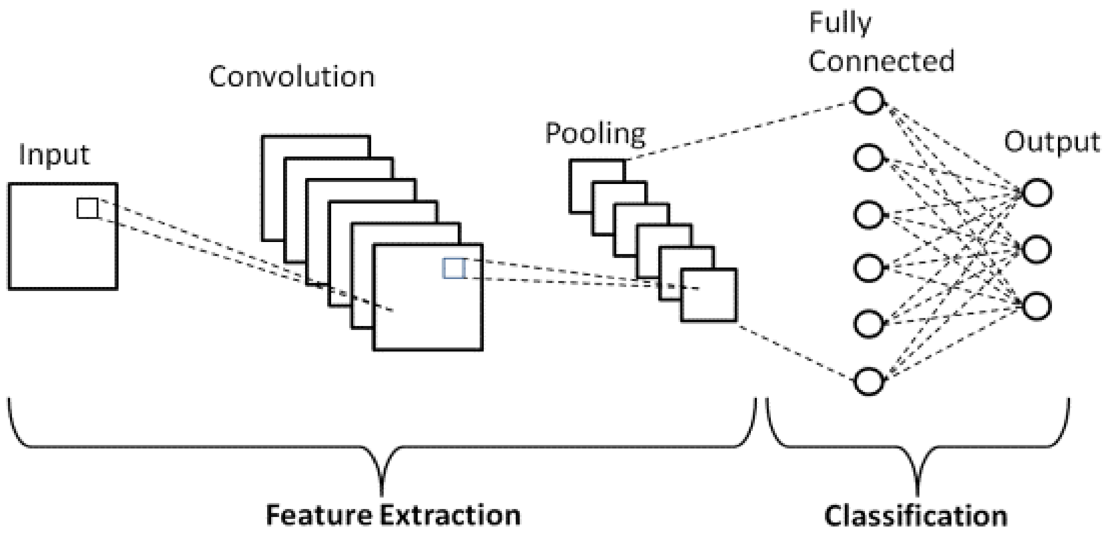


Figure 5.1: CNN Architecture

1. Convolutional Layer: This layer is the first layer that is used to extract the numerous features from the input images. Convolution is a mathematical process that is carried out at this layer between the input image and a filter of a specific size, MxM. The dot product is taken between the filter and the input image's components with regard to the filter's size by sliding the filter over the input image (MxM). The result is known as the Feature map, and it provides details about the image, including its corners and edges. This feature map is later supplied to further layers to teach them additional features from the input image.

2. Pooling Layer: A Pooling Layer often comes after a Convolutional Layer. This layer's main goal is to lower the convolved feature map's size in order to save on computational expenses. This is done independently on each feature map and by reducing the links between layers. There are various sorts of pooling operations, depending on the mechanism utilized. Typically, the Pooling Layer acts as a link between the FC Layer and the Convolutional Layer.

3. Fully Connected Layer: The Fully Connected (FC) layer connects the neurons between two layers by including weights and biases in addition to the neurons. These layers make up the final few layers of a CNN architecture and are often positioned before the output layer. This flattens the input picture from the layers underneath and provides it to the FC layer. The flattened vector then travels through a few additional FC levels, where the operations on the mathematical functions often happen. The categorization procedure starts to take place at this point.

4. Dropout: Normally, overfitting in the training dataset might result from all features being connected to the FC layer. When a certain model performs so well on training data that it has a negative effect on the model's performance when applied to new data, this is known as overfitting. A dropout layer, which involves removing a few neurons from the neural network during training to lower the size of the model, is used to solve this issue.

5. Activation Functions: The activation function is one of the most crucial elements of the CNN model. They are employed to discover and approximation any type of continuous and complex link between network variables. In layman's terms, it determines which model information should shoot ahead and which should not at the network's end.

5.1.2 LSTM

Long Short-Term Memory is the higher version of Recurrent Neural Network (RNN) architecture, designed to remove the vanishing gradient problem when the training model is undisturbed. It has different types of parameters like learning rate, input and output biases, thus eliminating the need for fine adjustments. LSTMs reduced the complexity involved to update each weight to O(1). The below figure describes the model.

LSTMs use the notion of gates to simplify and effectively perform calculations involving both Long Term Memory (LTM) and Short Term Memory (STM).

1. Forget Gate: When LTM goes to the forget gate, it forgets unhelpful knowledge.
2. Learn Gate: The current input (event) and STM are integrated in order to apply the pertinent knowledge we have recently gained from STM to the present input.
3. Remember Gate: In the Remember Gate, which functions as an updated LTM, LTM data that we haven't forgotten, STM data, and Event data are joined.
4. Use Gate: This gate also makes use of the LTM, STM, and Event to forecast the outcome of the currently occurring event, which serves as an updated STM.

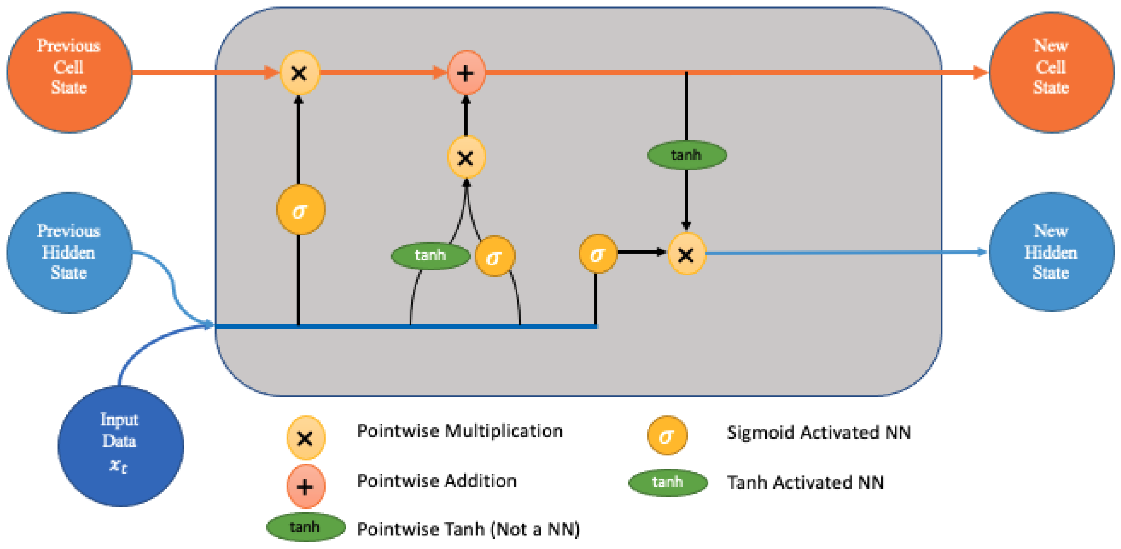


Figure 5.2: LSTM Architecture

5.1.3. CONV-LSTM

This model combines time series and computer vision by adding convolutional recurrent cells in a LSTM layer. The application is to predict the next-frame based on previous frames. ConvLSTM overcomes the problem involved in spatiotemporal data handling. Here, the image is sent through the convolution layers, and the output is a set of features flattened to a 1D array. When this method is applied to all other images in the time set, a set of characteristics that change over time is obtained, which is the LSTM layer input. The method is illustrated in the following figure.

ConvLSTM layer:

1. **ConvLSTM layer input:** A set of time-series data, or a 3D tensor with shape (samples, time steps, features), is used as the LSTM cell input. The input to the Convolution layer is a set of a 4D tensor with form of images (samples, channels, rows, cols). A collection of photos collected over time as a 5D tensor with shape (samples, rows, columns, time steps, and channels).

2. **ConvLSTM layer output:** The return sequences attribute affects the LSTM cell's output. The output is a sequence over time when True is set (one output for each input). The output is the last value of the sequence, which is a 2D tensor with shape, when return sequences is set to False (the default) (samples, features). A collection of photos as a 4D tensor with shape is what the convolution layer produces (samples, filters, rows, cols).

3. **filters:** How many output filters are included in the convolution.

4. **kernel size:** Determining the convolution window's height and width.

5. **padding:** Choose between "valid" and "same." When the channel comes first ("channels first") or last ("channels last"), the images are formatted accordingly.

6. **data_format:** If a channel appears first ("channels first") or last ("channels last"), the format of the images reflects that.

7. **activation:** The function of activation. The linear function a(x) = x is the default. The activation function to utilise for the recurrent step is 8. Hard sigmoid (hard sigmoid) is the default.

8. **recurrent_activation:** Utilizing a non - linear activation for the recurrent step Hard sigmoid (hard_sigmoid) is the default.

9. **return_sequences:** It decides whether to merely return the final output in the output series (False) or the whole output sequence (True). The default value is False.

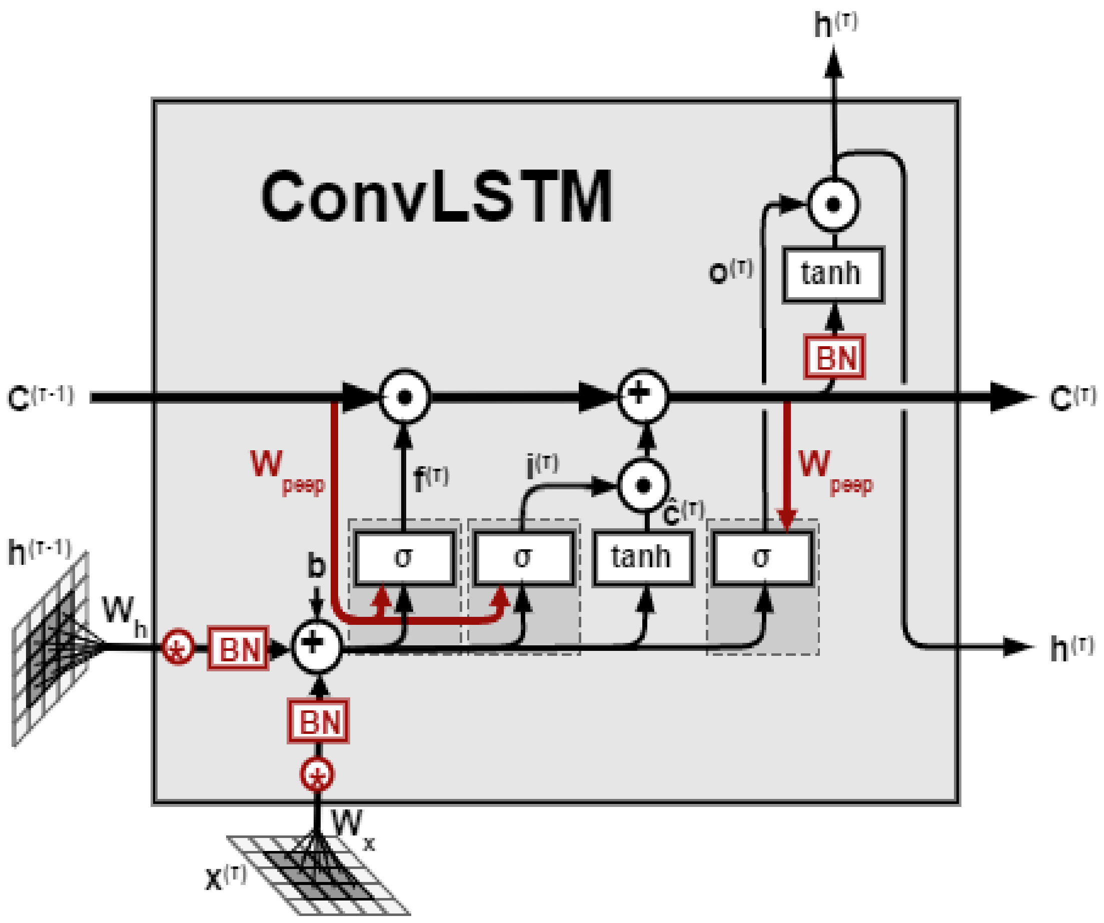


Figure 5.3: ConvLSTM Architecture

5.2 Flow Diagrams

Data Pre-processing

The UCF 101 Dataset is used to collect the data. It gives us access to the original dataset used by the Centre for Research in Computer Vision at the University of Central Florida for the detection of human action in videos. The quantity and quality of data collected directly influence how accurate the desired system will be. Therefore, in order to execute Data Pre-processing, where data preparation is done, we must clean the raw data. Raw data cannot be used to create models since it may contain missing values, inconsistent values, duplicate instances, etc.

Frame Extraction

Key frame extraction is an important method that uses a set of summary key frames to represent video sequences in order to implement video content. In this phase, we get the disc location of the video whose frames need to be retrieved, resize the video's frames, and then normalise the resized frames.

Creation of a dataset

In this step, data from the chosen classes are extracted, a dataset is created, and features, labels (class indices associated with the videos), and disc paths for the movies are generated. The data mentioned above is used to create our dataset.

Data Splitting: Train and Test

Training dataset and testing dataset are created from the dataset. For training, a training dataset is utilised. A test dataset is utilised.

Training Model

In the first case, as shown in figure (a) the training dataset is fed into two learning algorithms, CNN and LSTM consecutively to build an LRCN model. Data is processed with CNN model whose outputs are fed into a stack of recurrent sequence models (LSTMs), which finally produce a variable-length prediction. The representation may scale to arbitrarily long sequences since the CNN and LSTM weights are shared across time. In the second case, as shown in figure (b) the training dataset is fed into the

Convolutional-Long short term memory model as shown in the above flowchart. But instead of creating the learning algorithm in two steps like in the LRCN Model, we perform one step where we construct our ConvLSTM model directly.

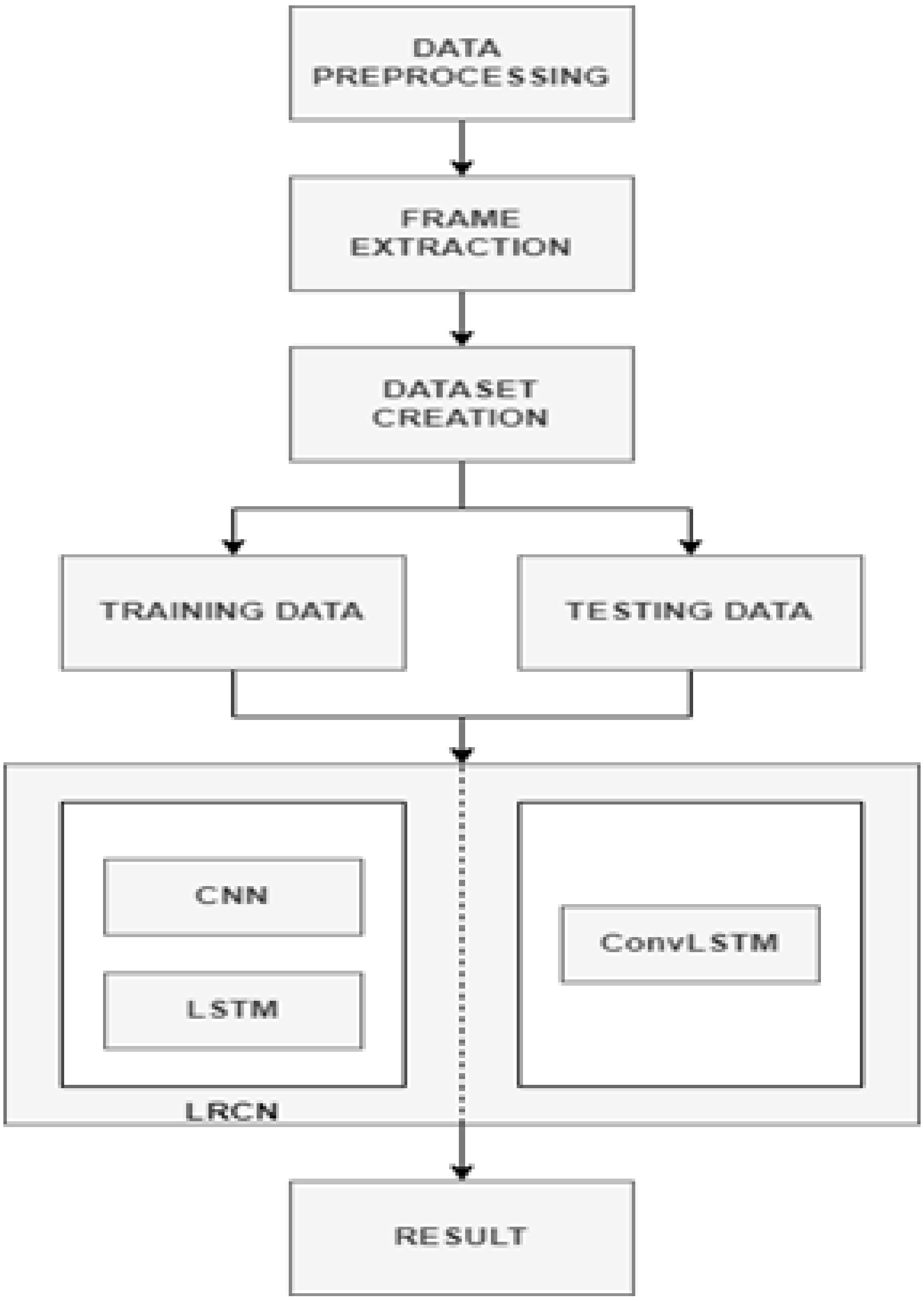


Figure 5.4: Flow Chart

Result

On the new dataset, we train the ConvLSTM and LRCN models, and we compare their performance by categorising random YouTube videos.

CHAPTER 6
SYSTEM IMPLEMENTATION

Visions and plans are put into action at this period. This is the logical conclusion, after evaluating, deciding, visioning, planning, applying of a project. The system implementation elaborates on how the system and the algorithms were implemented.

To create the classification model, we make use of the UCF101 action dataset. We compare two methods: LRCN and ConvLSTM to understand which model performs better. We need to create a dataset before training the model. This includes taking each video and extracting frames from it. This pre-processing task includes the following step:

- Select 20 classes from UCF101 dataset.
- Extract equally spaced frames from each video.
- Resize each frame to 64x64 height and width.
- Normalize the resized frame.
- Convert labels to one-hot-encoded vectors.
- Split data into Train and Test.

We use OpenCV to capture each frame and then resize it to a 64x64 image. Then we normalize the resized frame by dividing it with 255 so that each pixel value then lies between 0 and 1. We use Keras's to_categorical method to convert labels into one-hot-encoded vectors. The data was then divided into training and test sets. After performing these steps, we build our LRCN and ConvLSTM model.

6.1 LRCN Model

Long-term Recurrent Neural Network (LRCN) is a recurrent neural network that has been trained for a long time. This model uses the Long Short Term Memory (LSTM) cell to remember the past events and then predict future values of the target variable. The LSTM unit can remember up to 10,000 samples in its memory pool. It takes a lot of time for this type of unit to learn new data but it is very good at predicting future values from existing data points. The LRCN works on a time-step basis and its input layer consists of an array of vectors (one for each time step). The output layer consists of another vector that contains the probability distribution over all possible outputs at this point in time based on these inputs.

We use Keras Sequential model to define our architecture. In LRCN, we will have convolution layers which are then connected to a LSTM. We use time distributed layers so that we can apply the same Conv2D layers to each of the frames in a sample. Because time distributions apply the same instance of Conv2D to each frame, the same set of weights are used at each frame.

We have four Conv2D layers with relu activation and with 16, 32, 64 and 64 filters respectively. The computational cost of each of these layers is decreased by max pooling, while overfitting is avoided through dropout. Convolution layer output is flattened before being sent into the LSTM layer. Finally, we group the videos based on their content using a thick layer with softmax activation. The structure of model is shown below:

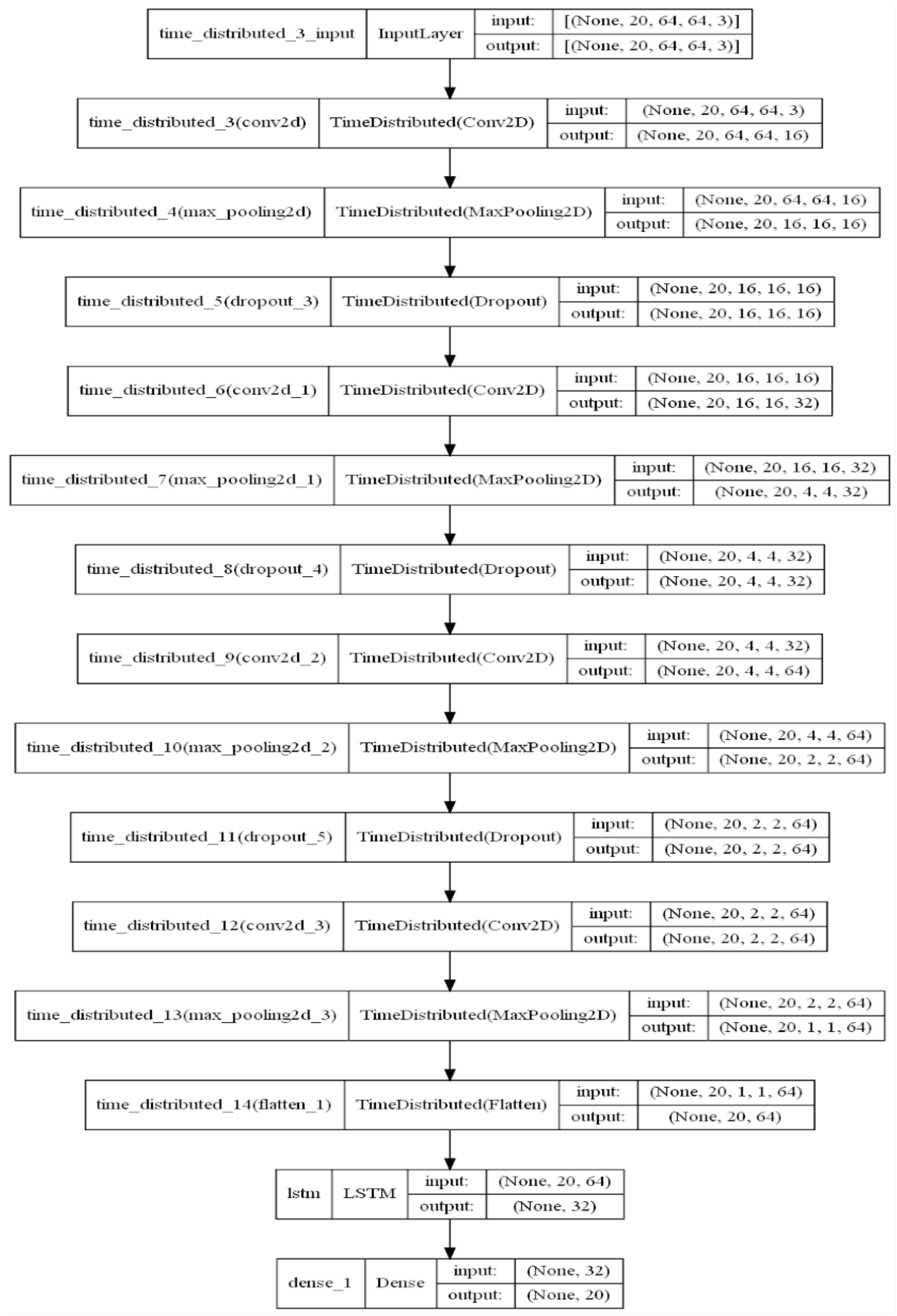

time_distributed_3_input | InputLayer | input: | [(None, 20, 64, 64, 3)] | output: | [(None, 20, 64, 64, 3)]
time_distributed_3(conv2d) | TimeDistributed(Conv2D) | input: | (None, 20, 64, 64, 3) | output: | (None, 20, 64, 64, 16)
time_distributed_4(max_pooling2d) | TimeDistributed(MaxPooling2D) | input: | (None, 20, 64, 64, 16) | output: | (None, 20, 16, 16, 16)
time_distributed_5(dropout_3) | TimeDistributed(Dropout) | input: | (None, 20, 16, 16, 16) | output: | (None, 20, 16, 16, 16)
time_distributed_6(conv2d_1) | TimeDistributed(Conv2D) | input: | (None, 20, 16, 16, 16) | output: | (None, 20, 16, 16, 32)
time_distributed_7(max_pooling2d_1) | TimeDistributed(MaxPooling2D) | input: | (None, 20, 16, 16, 32) | output: | (None, 20, 4, 4, 32)
time_distributed_8(dropout_4) | TimeDistributed(Dropout) | input: | (None, 20, 4, 4, 32) | output: | (None, 20, 4, 4, 32)
time_distributed_9(conv2d_2) | TimeDistributed(Conv2D) | input: | (None, 20, 4, 4, 32) | output: | (None, 20, 4, 4, 64)
time_distributed_10(max_pooling2d_2) | TimeDistributed(MaxPooling2D) | input: | (None, 20, 4, 4, 64) | output: | (None, 20, 2, 2, 64)
time_distributed_11(dropout_5) | TimeDistributed(Dropout) | input: | (None, 20, 2, 2, 64) | output: | (None, 20, 2, 2, 64)
time_distributed_12(conv2d_3) | TimeDistributed(Conv2D) | input: | (None, 20, 2, 2, 64) | output: | (None, 20, 2, 2, 64)
time_distributed_13(max_pooling2d_3) | TimeDistributed(MaxPooling2D) | input: | (None, 20, 2, 2, 64) | output: | (None, 20, 1, 1, 64)
time_distributed_14(flatten_1) | TimeDistributed(Flatten) | input: | (None, 20, 1, 1, 64) | output: | (None, 20, 64)
lstm | LSTM | input: | (None, 20, 64) | output: | (None, 32)
dense_1 | Dense | input: | (None, 32) | output: | (None, 20)

Figure 6.1: LRCN Model Structure

6.2 CONV-LSTM Model

ConvLSTM is a convolutional network that can be used for sequential data. It is based on the LSTM model, which has been widely applied to sequence modelling tasks in many fields, such as speech and text processing, machine translation, and time series forecasting. The ConvLSTM model inherits the advantages of LSTMs such as memory capacity (up to 128-bit), high accuracy of prediction (on average 99% or higher), and ability to learn from multiple examples simultaneously. In addition, it also has much faster training speed than other models with similar performance. It is trained to predict the next output in an input sentence given the previous input and its context.

The ConvLSTM model was first introduced in 2016 and it can be used to generate sequences with LSTM-like behaviour. This model consists of two parts: Conv layer: It takes an input sequence as a single unit and then generates one output sequence from it. In other words, this part learns how to take multiple inputs and produce one output at each time step. Long short-term memory (LSTM): cell whose state encodes the past history of inputs.

We use Keras Sequential model to define the architecture. ConvLSTM is just like LSTM but the internal matrix multiplications are replaced with convolutional operations. As a result, the data that flows through ConvLSTM keeps input dimensions instead of being just a 1D array. The model is shown below:

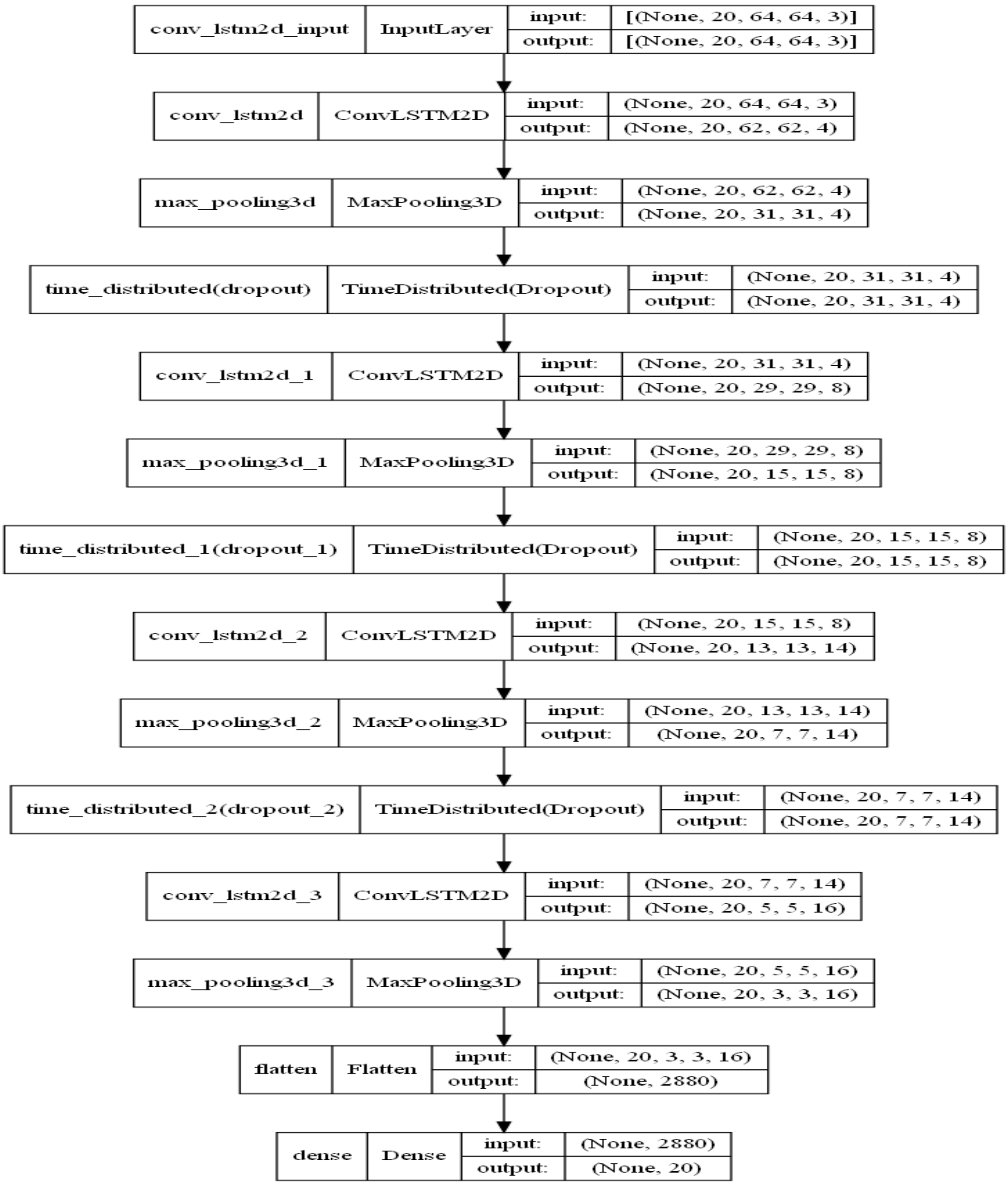

conv_lstm2d_input | InputLayer | input: [(None, 20, 64, 64, 3)] | output: [(None, 20, 64, 64, 3)]
conv_lstm2d | ConvLSTM2D | input: (None, 20, 64, 64, 3) | output: (None, 20, 62, 62, 4)
max_pooling3d | MaxPooling3D | input: (None, 20, 62, 62, 4) | output: (None, 20, 31, 31, 4)
time_distributed(dropout) | TimeDistributed(Dropout) | input: (None, 20, 31, 31, 4) | output: (None, 20, 31, 31, 4)
conv_lstm2d_1 | ConvLSTM2D | input: (None, 20, 31, 31, 4) | output: (None, 20, 29, 29, 8)
max_pooling3d_1 | MaxPooling3D | input: (None, 20, 29, 29, 8) | output: (None, 20, 15, 15, 8)
time_distributed_1(dropout_1) | TimeDistributed(Dropout) | input: (None, 20, 15, 15, 8) | output: (None, 20, 15, 15, 8)
conv_lstm2d_2 | ConvLSTM2D | input: (None, 20, 15, 15, 8) | output: (None, 20, 13, 13, 14)
max_pooling3d_2 | MaxPooling3D | input: (None, 20, 13, 13, 14) | output: (None, 20, 7, 7, 14)
time_distributed_2(dropout_2) | TimeDistributed(Dropout) | input: (None, 20, 7, 7, 14) | output: (None, 20, 7, 7, 14)
conv_lstm2d_3 | ConvLSTM2D | input: (None, 20, 7, 7, 14) | output: (None, 20, 5, 5, 16)
max_pooling3d_3 | MaxPooling3D | input: (None, 20, 5, 5, 16) | output: (None, 20, 3, 3, 16)
flatten | Flatten | input: (None, 20, 3, 3, 16) | output: (None, 2880)
dense | Dense | input: (None, 2880) | output: (None, 20)

Figure 6.2: ConvLSTM Model Structure

We have four ConvLSTM2D layers with 4, 8, 14 and 16 filters respectively. The computational cost of each of these layers is decreased by max pooling, while overfitting is avoided through dropout. When the output is flattened and fed into a dense network using softmax activation, the video is categorised into one of the 20 categories.

CHAPTER 7
SYSTEM TESTING AND RESULT ANALYSIS

System testing and result analysis of ML model is the process of running a machine learning algorithm on a dataset to evaluate its performance. This process is also called as end-to-end testing. It involves training, evaluation, and reporting of the results from an ML system. The final output can be used for decision making or any other purpose that requires accurate prediction by using data analytics tools. System testing is used to test the model performance on a real data. It includes both manual and automated tests. The results of this testing are then analysed by result analysis. This process helps us understand how well our model performs on different types of data, what kind of errors it has and how much time it takes to run the model.

To compare our model, we utilize the Test dataset. We also put our models to the test by watching random YouTube videos. From the video, equal-spaced frames are extracted. Each frame is scaled and normalized before being input into the prediction model. The output of both models is examined and reported in Tables 7.1 and 7.0, respectively, and the accuracies of both models are compared as shown in figure 7.1.

ConvLSTM Model					
Sequence Length	Training		Validation		Trainable Parameters
	Loss	Accuracy	Loss	Accuracy	
20	0.0985	0.9677	1.4445	0.6969	90,620
25	0.1379	0.9551	1.1463	0.6890	105,020
30	0.0938	0.9716	1.1798	0.6654	119,420

Table 7.1: LRCN Model Results

`

LRCN Model					
Sequence Length	Training		Testing		Trainable Parameters
	Loss	Accuracy	Loss	Accuracy	
20	0.0512	0.9852	0.7467	0.8307	73.588
25	0.0712	0.9809	0.7242	0.8189	73,588
30	0.0884	0.9716	0.7689	0.8228	73,588

Table 7.2: ConvLSTM Model Results

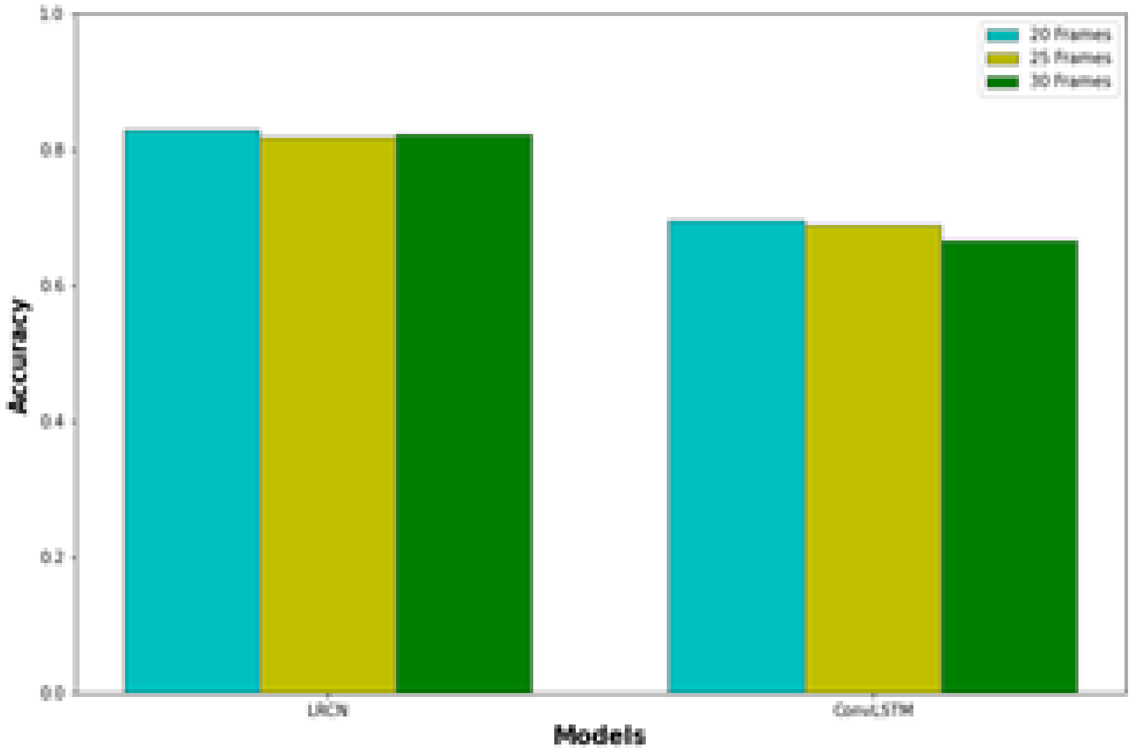
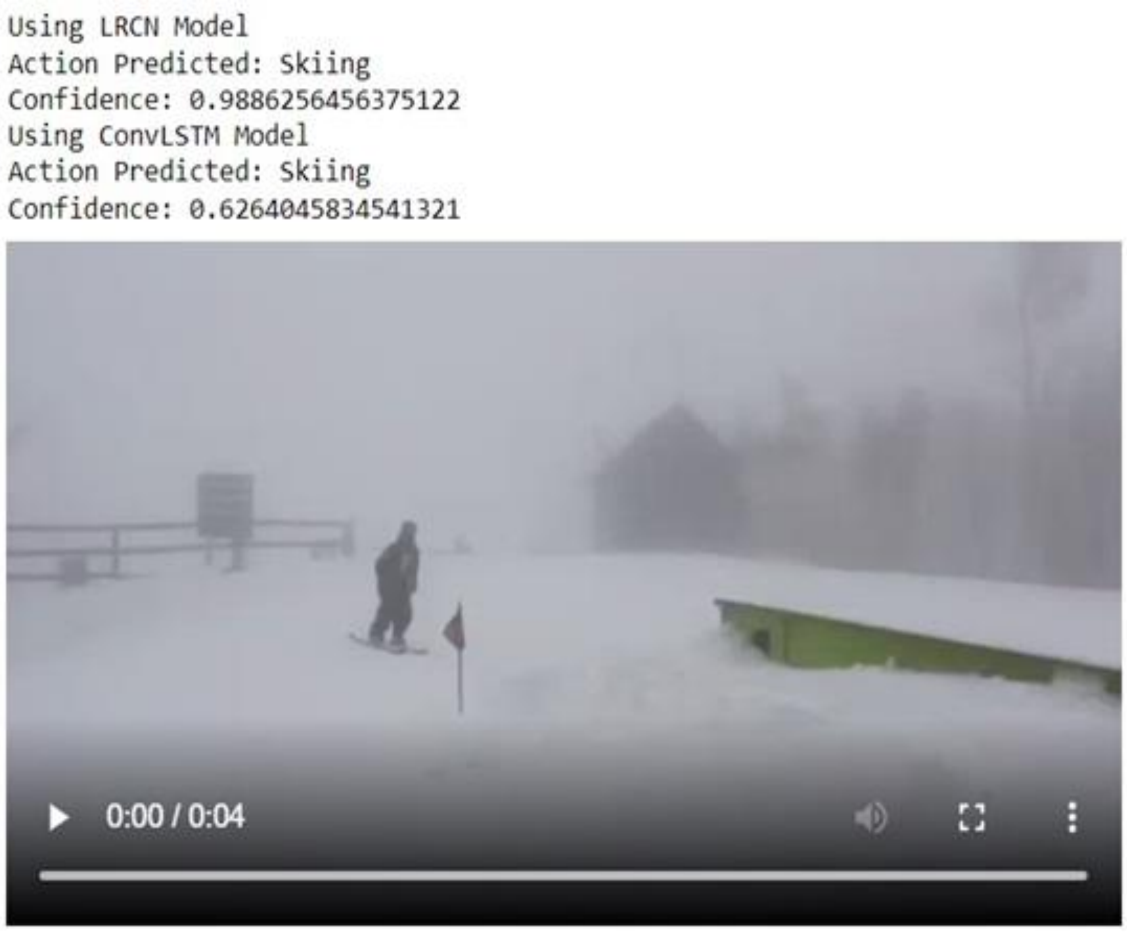


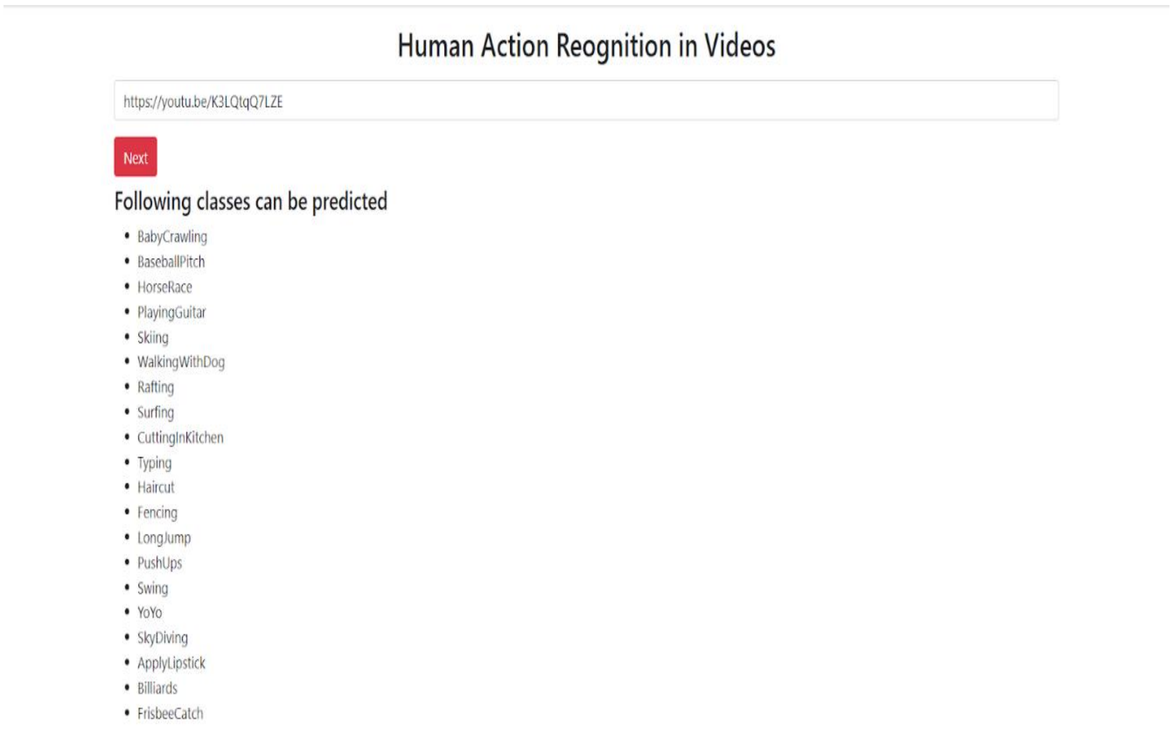
Figure 7.1: Accuracy of models for different frames

Upon giving a random Skiing (One of the training Class) video from YouTube to both the models, LRCN model predicts the class with more confidence than ConvLSTM model as shown in Fig. By carefully analysing these results we can conclude that LRCN is a better model for Human Action Recognition



```
Using LRCN Model
Action Predicted: Skiing
Confidence: 0.9886256456375122
Using ConvLSTM Model
Action Predicted: Skiing
Confidence: 0.6264045834541321
```

Figure 7.2: Prediction using both models



Human Action Reognition in Videos

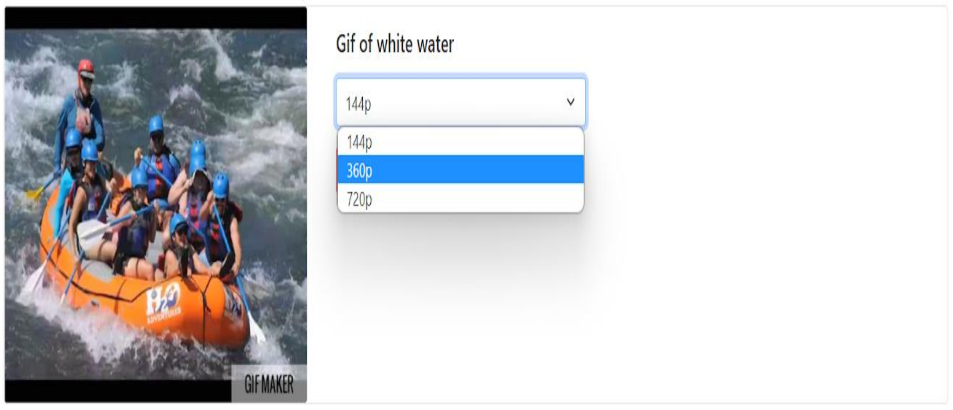
https://youtu.be/K3LQtqQ7LZE

Next

Following classes can be predicted

- BabyCrawling
- BaseballPitch
- HorseRace
- PlayingGuitar
- Skiing
- WalkingWithDog
- Rafting
- Surfing
- CuttingInKitchen
- Typing
- Haircut
- Fencing
- LongJump
- PushUps
- Swing
- YoYo
- SkyDiving
- ApplyLipstick
- Billiards
- FrisbeeCatch

Figure 7.3: Website Homepage

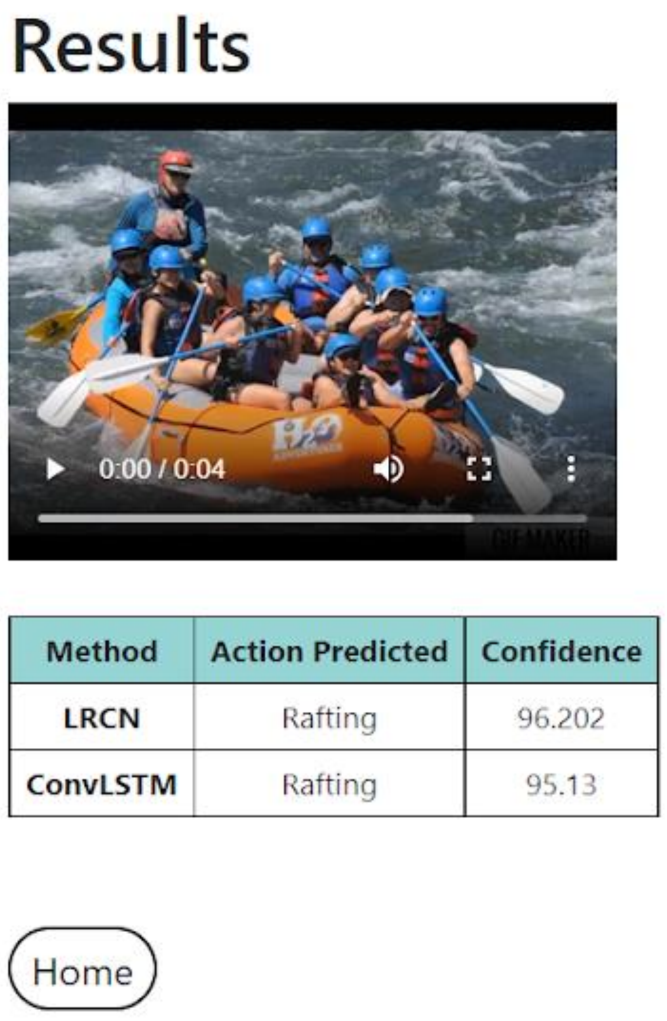


Gif of white water

144p

144p
360p
720p

Figure 7.4: Downloading Video



Results

Method	Action Predicted	Confidence
LRCN	Rafting	96.202
ConvLSTM	Rafting	95.13

Figure 7.5: Result Page

Figure 7.3 shows the websites homepage which lists all the class of action that our model predicts. We provide the link of the video and download the same as shown in Figure 7.4, which then is subjected to prediction. The prediction of our models is shown in our result page as in Figure 7.5.

CHAPTER 8
CONCLUSION AND FUTURE WORK

Action recognition in videos can be achieved in numerous ways. We compared two popular deep learning techniques in this paper: The Long-Term Recurrent Convolutional Network (LRCN) and the Convolutional LSTM (ConvLSTM), a family of models that is both spatially and temporally deep and versatile enough to be used in a variety of vision tasks with sequential inputs and outputs. After carefully analysing the results, our finding shows that the LRCN model outperforms the ConvLSTM model and also is easily trainable.

In future work, we aim to experiment with real time data in lieu of feeding already existing videos to the model. We hope to incorporate broader categories in the dataset to obtain more powerful and generic features and investigate approaches that explicitly reason about camera motion. We can attempt to increase the system accuracy by taking advantage of the diversity datasets such as Activity net, as it offers to obtain an even more robust activity recognition system. A better model could be implemented by improving the fine tuning process, varying the number of layers, number of neurons, learning rate, etc.

APPENDIX A

Project team details

Project Title	Human Action Recognition in Videos		
USN	**Team Members**	**Email ID**	**Mobile number**
01JST18CS047	Kartik Nagaraj Nayak	kartiknnayak2000@gmail.com	7022773055
01JST18CS106	Rishitha S Ramesh	rishitha8112000@gmail.com	8660774599
01JST18CS133	Shriya Mittapalli	mittapallishriya@gmail.com	8618630171
01JST18CS162	Vanshika V	vanshikav.2000@gmail.com	7619554350

Table A1: Team Details

Figure A1
Team Photo with all team members and guide placed from the left as Vanshika V, Shriya Mittapalli, Dr. Manimala S, Rishitha S Ramesh and Kartik Nagaraj Nayak.

APPENDIX B
COs, POs and PSOs

Mapping for the project work (CS83P)

Course Outcomes:

CO1: Formulate the problem definition, conduct literature review and apply requirements analysis.

CO2: Develop and implement algorithms for solving the problem formulated.

CO3: Comprehend, present and defend the results of exhaustive testing and explain the major findings.

Program Outcomes:

PO1: Apply knowledge of computing, mathematics, science, and foundational engineering concepts to solve the computer engineering problems.

PO2: Identify, formulate and analyze complex engineering problems.

PO3: Plan, implement and evaluate a computer-based system to meet desired societal needs such as economic, environmental, political, healthcare and safety within realistic constraints.

PO4: Incorporate research methods to design and conduct experiments to investigate real-time problems, to analyze, interpret and provide feasible conclusion.

PO5: Propose innovative ideas and solutions using modern tools.

PO6: Apply computing knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO7: Analyze the local and global impact of computing on individuals and organizations for sustainable development.

PO8: Adopt ethical principles and uphold the responsibilities and norms of computer engineering practice.

PO9: Work effectively as an individual and as a member or leader in diverse teams and in multidisciplinary domains.

PO10: Effectively communicate and comprehend.

PO11: Demonstrate and apply engineering knowledge and management principles to manage projects in multidisciplinary environments.

PO12: Recognize contemporary issues and adapt to technological changes for lifelong learning.

Program Specific Outcomes:

PSO1: Problem Solving Skills: Ability to apply standard practices and mathematical methodologies to solve computational tasks, model real world problems in the areas of database systems, system software, web technologies and Networking solutions with an appropriate knowledge of Data structures and Algorithms.
PSO2: Knowledge of Computer Systems: An understanding of the structure and working of the computer systems with performance study of various computing architectures.
PSO3: Successful Career and Entrepreneurship: The ability to get acquaintance with the state-of-the-art software technologies leading to entrepreneurship and higher studies.
PSO4: Computing and Research Ability: Ability to use knowledge in various domains to identify research gaps and to provide solution to new ideas leading to innovations.

Justification for CO-PO and PSO mapping

In our project we have come across many of these Program Outcomes and Program Specific Outcomes at various stages. The first CO is related to problem definition, literature survey and requirement analysis. Planning the project such that it meets the needs of society, by considering all the constraints is very relevant for this. Investigating real time problems and incorporating our findings in literature survey plays an important role as well. Understanding the constraints of the environment in which the system will be used plays a crucial role in deciding the requirements and using latest technology to make our implementation better is of high relevance. For CO1 we should have a clear knowledge and proper understanding of the problem statement. We must also have knowledge of the various computer technologies so as to know which ones are related to the topic chosen and can be used to solve the problem. Feasibility study must be done in order to figure out if the solution to our problem is available and can be done with the knowledge and technologies available to us. Good communication and reading skills are required while doing the literature survey. We should also make sure that the result of our project is beneficial for the society. A plan of execution must be made and followed and the work must be equally divided among the members of the group.

The second CO, design and implementation highly depends on the way we apply already acquired knowledge about computing, mathematics, etc., the innovation we bring in to our implementation, make our implementation adaptable to technological changes that might happen in future and the way we look at the problem and apply our knowledge. The ability to analyze global and local impact of the system, ability to uphold the ethical principles of engineering practices, the way in which we communicate and comprehend the concepts, the way in which the project is handled in multidisciplinary environments hold great relevance in defending our work and explaining major findings during our project. For CO2 based on the problem statement and definition we should able to select the apt technologies to work on the project so as to derive efficient outcomes. Develop algorithms to approach the problem statement and accordingly come up with modules. For this we need to have problem solving skills, knowledge of computing, mathematics, science, and foundational engineering concepts to solve the computer engineering problems and incorporate research methods to design the modules. During this phase also it is very important to divide the work equally among the team members and there should be equal participation and good team co-ordination.

For CO3 thorough testing has to be done for chosen data. Several different test cases must be applied and the accuracy/correctness must be checked. Based on the calculations performed and the results obtained, efficiency must be calculated and we must ensure that the chosen method of arriving at the solution must be more effective than the already existing methods. At the end of the project completion we must have thorough understanding about all technologies used.

Table of Mapping of CO, PO and PSO:

SUBJECT	CODE	CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
Project Work	CS83P	CO1	2	3	2	3	3	1	1	3	2	2	1	2	1	1	2	3
		CO2	3	2	3	3	3	1	1	3	2	2	2	3	3	2	3	3
		CO3	2	1	1	2	2	2	1	3	2	3	3	3	2	1	2	3

Table B1: CO, PO, PSO Mappings

Note:

Scale

0 – Not Applicable

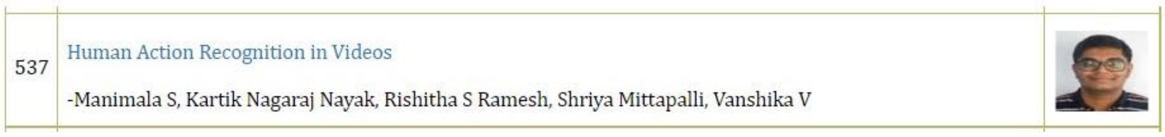
1 – Low relevance Scale

2 – Medium relevance Scale

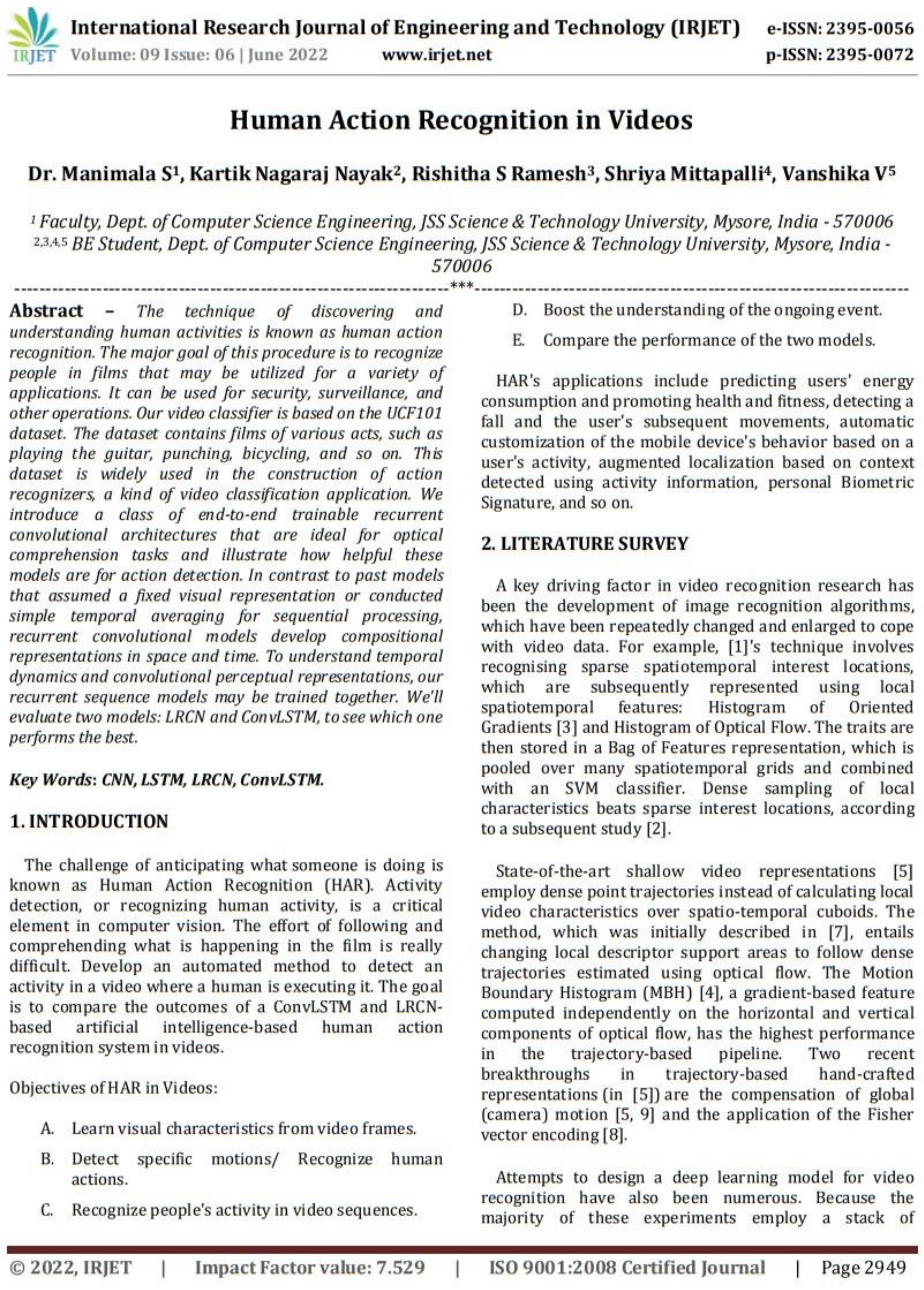
3 – High relevance

APPENDIX C – Publication details

Our project titled **Human Action Recognition in Videos** was made into a technical paper and has been submitted to **International Research Journal of Engineering and Technology (IRJET)** on BE Thesis in Computer Science. The technical paper has also been published in the same under Volume 9 Issue 6, June 2022.



537	Human Action Recognition in Videos -Manimala S, Kartik Nagaraj Nayak, Rishitha S Ramesh, Shriya Mittapalli, Vanshika V	



International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 06 | June 2022 www.irjet.net p-ISSN: 2395-0072

Human Action Recognition in Videos

Dr. Manimala S[1], Kartik Nagaraj Nayak[2], Rishitha S Ramesh[3], Shriya Mittapalli[4], Vanshika V[5]

[1] *Faculty, Dept. of Computer Science Engineering, JSS Science & Technology University, Mysore, India - 570006*
[2,3,4,5] *BE Student, Dept. of Computer Science Engineering, JSS Science & Technology University, Mysore, India - 570006*

---***---

Abstract – *The technique of discovering and understanding human activities is known as human action recognition. The major goal of this procedure is to recognize people in films that may be utilized for a variety of applications. It can be used for security, surveillance, and other operations. Our video classifier is based on the UCF101 dataset. The dataset contains films of various acts, such as playing the guitar, punching, bicycling, and so on. This dataset is widely used in the construction of action recognizers, a kind of video classification application. We introduce a class of end-to-end trainable recurrent convolutional architectures that are ideal for optical comprehension tasks and illustrate how helpful these models are for action detection. In contrast to past models that assumed a fixed visual representation or conducted simple temporal averaging for sequential processing, recurrent convolutional models develop compositional representations in space and time. To understand temporal dynamics and convolutional perceptual representations, our recurrent sequence models may be trained together. We'll evaluate two models: LRCN and ConvLSTM, to see which one performs the best.*

Key Words*: *CNN, LSTM, LRCN, ConvLSTM.

1. INTRODUCTION

The challenge of anticipating what someone is doing is known as Human Action Recognition (HAR). Activity detection, or recognizing human activity, is a critical element in computer vision. The effort of following and comprehending what is happening in the film is really difficult. Develop an automated method to detect an activity in a video where a human is executing it. The goal is to compare the outcomes of a ConvLSTM and LRCN-based artificial intelligence-based human action recognition system in videos.

Objectives of HAR in Videos:

A. Learn visual characteristics from video frames.
B. Detect specific motions/ Recognize human actions.
C. Recognize people's activity in video sequences.
D. Boost the understanding of the ongoing event.
E. Compare the performance of the two models.

HAR's applications include predicting users' energy consumption and promoting health and fitness, detecting a fall and the user's subsequent movements, automatic customization of the mobile device's behavior based on a user's activity, augmented localization based on context detected using activity information, personal Biometric Signature, and so on.

2. LITERATURE SURVEY

A key driving factor in video recognition research has been the development of image recognition algorithms, which have been repeatedly changed and enlarged to cope with video data. For example, [1]'s technique involves recognising sparse spatiotemporal interest locations, which are subsequently represented using local spatiotemporal features: Histogram of Oriented Gradients [3] and Histogram of Optical Flow. The traits are then stored in a Bag of Features representation, which is pooled over many spatiotemporal grids and combined with an SVM classifier. Dense sampling of local characteristics beats sparse interest locations, according to a subsequent study [2].

State-of-the-art shallow video representations [5] employ dense point trajectories instead of calculating local video characteristics over spatio-temporal cuboids. The method, which was initially described in [7], entails changing local descriptor support areas to follow dense trajectories estimated using optical flow. The Motion Boundary Histogram (MBH) [4], a gradient-based feature computed independently on the horizontal and vertical components of optical flow, has the highest performance in the trajectory-based pipeline. Two recent breakthroughs in trajectory-based hand-crafted representations (in [5]) are the compensation of global (camera) motion [5, 9] and the application of the Fisher vector encoding [8].

Attempts to design a deep learning model for video recognition have also been numerous. Because the majority of these experiments employ a stack of

REFERENCES

[1] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. “Learning realistic human actions from movies”. In Proc. CVPR, 2008.

[2] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features ¨for action recognition. In Proc. BMVC., pages 1–11, 2009.

[3] N. Dalal and B Triggs. Histogram of Oriented Gradients for Human Detection. In Proc. CVPR, volume 2, pages 886– 893, 2005.

[4] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In Proc. ECCV, pages 428–441, 2006.

[5] H. Wang and C. Schmid. Action recognition with improved trajectories. In Proc. ICCV, pages 3551– 3558, 2013.

[6] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In Proc. ICCV, pages 1–8, 2007.

[7] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In ¨ Proc. CVPR, pages 3169–3176, 2011.

[8] F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In Proc. ECCV, 2010.

[9] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In Proc. ICCV, pages 2556–2563, 2011.

[10] Human Activity Recognition using Accelerometer and Gyroscope Data from Smartphones. Khimraj, Praveen Kumar Shukla, Ankit Vijayvargiya, Rajesh Kumar.

[11] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, Li Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks.

[12] Shaunak Deshpande, Ankur Kumar, Abhishek Vastrad, Prof. Pankaj Kunekar. Video Classification using Machine Learning.

[13] Long-term Recurrent Convolutional Networks for Visual Recognition and Description Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell