VISION BASED DETECTION AND ANALYSIS OF HUMAN ACTIVITIES

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

RAVIPATI ABHIRAM (Reg.No - 39110840)

KONDAMURI RAKESH KRISHNA (Reg.No - 39110521)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RAVIPATI ABHIRAM (Reg.No - 39110840)** and **KONDAMURI RAKESH KRISHNA (Reg.No - 39110521)** who carried out the Project Phase-2 entitled **"VISION BASED DETECTION AND ANALYSIS OF HUMAN ACTIVITIES"** under my supervision from January 2023 to April 2023.

[illegible]

Internal Guide
Dr. A. MARY POSONIA M.E., Ph.D.

[illegible]

Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.,

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CSE

Submitted for Viva voce Examination held on 20/4/2023

[illegible] [illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **Ravipati Abhiram (Reg.No - 39110840),** hereby declare that the Project Phase-2 Report entitled **"VISION BASED DETECTION AND ANALYSIS OF HUMAN ACTIVITIES"** done by me under the guidance of **Dr. A. Mary Posonia, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

Ravi..

DATE: 20/4/2023
PLACE: Chennai **SIGNATURE OF CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Mary Posonia M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Human activity recognition (HAR) is an important research area in the field of computer vision and artificial intelligence. HAR involves identifying and classifying the activities that a person is performing based on sensor data or video data. HAR has numerous applications in various fields, including healthcare, security, and sports analysis. In recent years, deep learning models have shown great promise in HAR tasks. Two popular deep learning models for HAR are Convolutional LSTM (ConvLSTM) and Long-term Recurrent Convolutional Networks (LRCN). ConvLSTM models are an extension of the LSTM model that includes convolutional layers in the recurrent structure. ConvLSTM models are particularly useful for activity recognition tasks since they can capture the temporal dependencies between frames in video data. LRCN models combine a convolutional neural network (CNN) and an LSTM to classify videos. The CNN component extracts features from each frame in the video, while the LSTM component processes the temporal sequence of features to classify the activity. The success of HAR models is heavily dependent on the quality and quantity of data. To train the models, a dataset of labeled videos is required. The dataset is divided into training, validation, and test sets. The models are trained on the training set, and the validation set is used for hyperparameter tuning and model selection. The performance of the models is evaluated on the test set. HAR has numerous applications in healthcare, including monitoring the daily activities of elderly people or patients with chronic conditions. The system can alert caregivers if there is a deviation from normal activity patterns, indicating potential health issues or falls. In the field of security, HAR can be used for surveillance in public areas, such as airports or shopping malls, to detect suspicious behavior or identify individuals engaged in criminal activities. In sports analysis, HAR can be used to analyze the performance of athletes during training or competitions. The system can provide feedback on their technique and suggest improvements based on their activity patterns. Human activity recognition using ConvLSTM and LRCN models is a promising technology with numerous applications. The models are capable of accurately recognizing human activities in video data by capturing the temporal dependencies between frames. As the field of HAR continues to evolve, the models are likely to become more accurate and efficient, further expanding their potential applications

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
HAR	Human Activity Recognition
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
ConvLSTM	Convolutional Long Short Term Memory
LRCN	Long-term Recurrent Convolutional Network

CHAPTER 1
INTRODUCTION

Human activity recognition (HAR) is an emerging research field that has gained increasing attention in recent years due to its numerous applications in various fields, including healthcare, security, sports analysis, and robotics. HAR involves identifying and classifying the activities that a person is performing based on sensor data or video data. HAR has become increasingly important with the growing interest in personalized healthcare, aging in place, and smart homes. HAR can be broadly categorized into two types: sensor-based HAR and vision-based HAR. Sensor-based HAR involves using wearable sensors to capture motion and physiological data to identify activities. Vision-based HAR involves analyzing video data to identify activities. Vision-based HAR has received considerable attention due to the widespread availability of cameras and the rapid development of computer vision algorithms.

One of the main challenges in HAR is dealing with the variability and complexity of human activities. Human activities can vary significantly in terms of their duration, frequency, and context. Moreover, the same activity can be performed in different ways by different people. Another challenge is dealing with the noisy and incomplete sensor data or video data, which can lead to inaccurate activity recognition results. In recent years, deep learning models have shown great promise in HAR tasks. Two popular deep learning models for HAR are Convolutional LSTM (ConvLSTM) and Long-term Recurrent Convolutional Networks (LRCN). ConvLSTM models are an extension of the LSTM model that includes convolutional layers in the recurrent structure. ConvLSTM models are particularly useful for activity recognition tasks since they can capture the temporal dependencies between frames in video data. LRCN models combine a convolutional neural network (CNN) and an LSTM to classify videos. The CNN component extracts features from each frame in the video, while the LSTM component processes the temporal sequence of features to classify the activity.

1.1 BACKGROUND AND MOTIVATION

Human activity recognition using LSTM and CNN is a popular research area in the field of deep learning. LSTM and CNN are two deep learning architectures that can be used together to recognize human activities from sensor data. LSTM is used to recognize time-sequential features, while CNN is used to extract features from signals. This technology has many potential applications, such as in healthcare, sports, and security. The use of deep learning models for human activity recognition has shown great progress in recent years. The combination of LSTM and CNN has been found to be effective in recognizing human activities with high accuracy. This technology has the potential to improve the quality of life for people by providing personalized healthcare and fitness monitoring.

1.2 PROBLEM STATEMENT

The problem statement for human activity recognition using LSTM and CNN is to accurately recognize and classify human activities from sensor data. The traditional pattern recognition methods have limitations in recognizing complex human activities. Therefore, deep learning models such as LSTM and CNN are used to overcome these limitations. The challenge is to design an effective architecture that can extract features from sensor data and recognize time-sequential features with high accuracy. The use of smartphones sensors data for human activity recognition is a popular research area. The goal is to develop a system that can recognize different forms of human activities in real-time with high accuracy. The system should be able to recognize activities such as walking, running, sitting, standing, and other complex activities

1.3 OBJECTIVE AND SCOPE

Objective

The objective of human activity recognition using LSTM and CNN is to develop an accurate and efficient system that can recognize and classify human activities from sensor data. The system should be able to recognize different forms of human activities in real-time with high accuracy. The use of deep learning models such as

LSTM and CNN can help to overcome the limitations of traditional pattern recognition methods. The system should be able to recognize activities such as walking, running, sitting, standing, and other complex activities. The goal is to develop a system that can be used in various applications such as healthcare, sports, and security. The system should be able to provide personalized healthcare and fitness monitoring. The use of smartphones sensors data for human activity recognition is a popular research area.

Scope

The scope of human activity recognition using LSTM and CNN is broad and has many potential applications in various fields. In healthcare, it can be used to monitor patients' activities and provide personalized healthcare. It can also be used in sports to monitor athletes' activities and improve their performance. The system can be used in security to detect suspicious activities and prevent crimes. The use of smartphones sensors data for human activity recognition has made it possible to develop low-cost and portable systems that can be used in various applications. The combination of LSTM and CNN has shown promising results in recognizing human activities with high accuracy. The system can recognize activities such as walking, running, sitting, standing, and other complex activities. The use of deep learning models for human activity recognition has shown great progress in recent years

1.4 APPLICATIONS OF HUMAN ACTIVITY RECOGNITION

HAR has numerous applications in various fields, including healthcare, security, sports analysis, and robotics.

In healthcare, HAR can be used to monitor the daily activities of elderly people or patients with chronic conditions. The system can alert caregivers if there is a deviation from normal activity patterns, indicating potential health issues or falls. HAR can also be used to monitor the physical activity levels of patients with chronic conditions, such as diabetes or cardiovascular disease, to assess their adherence to prescribed exercise regimens.

In security, HAR can be used for surveillance in public areas, such as airports or shopping malls, to detect suspicious behavior or identify individuals engaged in criminal activities. HAR can also be used for crowd monitoring and control during large events, such as concerts or sporting events.

In sports analysis, HAR can be used to analyze the performance of athletes during training or competitions. The system can provide feedback on their technique and suggest improvements based on their activity patterns. HAR can also be used for injury prevention by monitoring the movement patterns of athletes and identifying potential sources of strain or injury.

In robotics, HAR can be used to enable robots to interact with humans more effectively by understanding their activities and intentions. HAR can also be used to enable robots to perform household tasks, such as cleaning or cooking, by recognizing and responding to human activities.

1.5 ORGANIZATION OF THESIS

The organization includes an introduction that provides an overview of the research problem, objectives, and significance. The literature review section discusses the existing research on human activity recognition using LSTM and CNN and the Limitations of the existing system. The methodology section describes the data collection process, pre-processing, and the deep learning models used. The results section presents the findings of the study, including the accuracy of the models and the comparison between ConvLSTM and LRCN. The discussion interprets the results, discusses the limitations of the study, and suggests research directions. Finally, the conclusion section summarizes the main findings and contributions of the study. The thesis also includes an abstract, acknowledgments, and references.

CHAPTER 2

LITERATURE SURVEY

Several studies on human activity recognition have been conducted in the recent years. This section summarizes all of the previous work on facial expression recognition.

2.1 OVERVIEW OF EXISTING RESEARCH AND LITERATURE

Usharani J et al. [1] came up with an idea for a human activity recognition system based on the Android platform. They created an application using the accelerometer data for classification, which supported online training and classification. They used the clustered k-NN approach to enhance the performance, accuracy, and execution time of the k-NN classifier with limited resources on the Android platform. They also concluded that the classification times were also dependent on the device models and capabilities.

In Davide Anguita et al.'s [2] paper, they introduced the improvised Support Vector Machine algorithm, which works with fixed point arithmetic to produce an energy-efficient model for the classification of human activities using a smartphone. They aimed to use the presented novel technology for various intelligence applications and smart environments for faster processing with the least possible use of system resources to save the consumption of energy along with maintaining comparable results with other generally used classification techniques.

The paper titled "Service Direct: Platform that Incorporates Service Providers and Consumers Directly" by Mary Posonia A et al. [3] published in the International Journal of Engineering and Advanced Technology (IJEAT) in 2019, discusses a platform that enables direct interaction between service providers and consumers. It begins by highlighting the challenges faced by consumers in finding service providers and the difficulties faced by service providers in reaching their target audience. The authors propose a solution in the form of a platform called "Service Direct" that directly connects service providers and consumers. The proposed platform allows consumers to search for service providers based on their requirements, such as location and service type, and directly interact with the service providers without any intermediaries. The platform also offers service providers the ability to create their profiles, list their services, and interact with potential consumers directly.

To understand people’s behavior in different places such as homes, clinics, etc., Md Zia Uddin et al. [4] proposed a body-sensor-based activity recognition system using deep Neural Stretchered Learning based on Long Short-Term Memory (LSTM). For better clustering of features from all the activities, Kernel-based Discriminant Analysis (KDA) was applied, which will maximize inter-class scattering and minimize intra-class scattering of the samples. The proposed model successfully achieved a recall of 99%, which was further compared to the existing deep learning models such as the RNN, Convolutional Neural Network (CNN), and Deep Belief Network (DBN).

Meysam, Vakili, et al. [5] proposed a real-time HAR model for online prediction of human physical movements based on the smartphone inertial sensors. A total of 20 different activities were selected, and six incremental learning algorithms were used to check the performance of the system, then all of them were also compared with the state-of-the-art HAR algorithms such as Decision Trees (DTs), AdaBoost, etc. Incremental k-NN and Incremental Naive Bayesian have given the best accuracy of 95%.

In Jirapond Muangprathub et al.’s [6] paper, they introduced a novel elderly person tracking system using a machine learning algorithm. In this work, they used the k-NN model with a k value of 5, which was able to achieve the best accuracy of 96.40% in detecting the real-time activity of elderly people. Furthermore, they created a system that displays information in a spatial format for an elderly person, and in case of an emergency, they can use a messaging device to request any help.

The paper titled "A Joint Optimization Approach for Security and Insurance Management on the Cloud" by Joshila Grace L.K et al.[7] published in Lecture Notes in Networks and Systems in 2021, proposes a joint optimization approach for security and insurance management on cloud computing platforms. The paper begins by highlighting the need for robust security measures and insurance policies in cloud computing to protect against potential threats such as data breaches and cyber-attacks. The authors propose a solution that integrates both security and insurance management into a single optimization problem. The proposed joint optimization approach involves modeling the security and insurance management as a two-stage stochastic optimization problem. In the first stage, the authors optimize the security measures to minimize the risk of potential threats, while in the second stage, they optimize the insurance policies to minimize the potential losses from these threats.

Baoding Zhou et al. [8] proposed a CNN for indoor human activity recognition. A total of nine different activities were recognized based on accelerometers, magnetometers, gyroscopes, and barometers collected by smartphones. The proposed method was able to achieve an excellent accuracy of 98%.

Abdulmajid Murad et al. [9] proposed a deep LSTM network for recognizing six different activities based on smartphone data. The network was able to achieve an accuracy of 96.70% on the UCI-HAD dataset.

The paper titled "Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network" by S. O. Eyobu and D. S. Han [10], published in Sensors in 2018 proposes a deep learning-based approach for human activity recognition using wearable IMU sensors. The authors focus on the problem of feature representation and data augmentation for improving the performance of the deep LSTM neural network model in classifying human activities.

The paper by Rueda et al. [11] proposes a system for human activity recognition using convolutional neural networks (CNNs) with body-worn sensors. The system consists of two stages: feature extraction and activity classification. CNNs are used for both stages, allowing the system to automatically learn discriminative features and classify activities. The authors evaluate their system on a public dataset containing sensor data from daily activities of 30 subjects. The proposed system achieves high accuracy rates of up to 98% for recognizing activities such as walking, standing, sitting, and lying down.

The paper by Kim et al. [12] (2019) presents a deep learning-based approach for recognizing the accompanying status of smartphone users using multimodal data. The authors proposed a system that combines accelerometer data, Bluetooth, and Wi-Fi signals to detect the accompanying status of users, which includes walking alone, walking with others, and not walking. The proposed system was evaluated using a dataset collected from 50 participants over a period of 5 days. The results showed that the system achieved an accuracy of 94.2% in detecting the accompanying status of users.

The paper titled "An Efficient Algorithm for Traffic Congestion Control" by Mary Posonia A. et al. [13] proposes an algorithm for controlling traffic congestion. The algorithm is designed to be efficient and effective in reducing traffic congestion in urban areas. The study aims to address the increasing problem of traffic congestion, which results in delays, increased fuel consumption, and increased pollution. The proposed algorithm is based on a dynamic traffic control model that takes into account real-time traffic flow data, road network topology, and the demand for transportation services. The algorithm is designed to optimize traffic flow and reduce congestion by adjusting traffic signal timings in real time.

The paper by F. Chollet [14] titled "Layer wrappers" describes the use of layer wrappers in Keras, a popular deep learning framework. Layer wrappers are used to modify the behavior of a layer in a neural network, such as adding regularization or modifying the input/output shapes. The "TimeDistributed" layer wrapper is particularly useful for handling sequences of data, where the same layer is applied to each time step of the sequence. This wrapper allows for efficient processing of temporal data in neural networks, such as in natural language processing or video analysis. The paper provides examples of how to use layer wrappers in Keras and discusses their practical applications in deep learning.

S. Hochreiter and J. Schmidhuber's [15] paper "Long short-term memory" published in Neural Computation in 1997 is a seminal work in the field of deep learning. The paper proposes a novel architecture for recurrent neural networks (RNNs) called Long Short-Term Memory (LSTM), which aims to address the vanishing gradient problem of traditional RNNs.

The paper titled "Unsupervised Learning of Human Activities from Long-Term Videos" by J. Wang, X. Zhang, Y. Wu, and Y. Wang [16], published in IEEE Transactions on Pattern Analysis and Machine Intelligence in 2022, proposes an unsupervised method for learning human activities from long-term videos. The proposed method, called LTVA (Long-Term Video Analysis), learns to segment videos into temporal regions, and extract features that are discriminative for different human activities.

The paper "Self-Supervised Learning of Human Activities from Temporal Segments in Videos" by Yao et al. (2021) [17] proposes a self-supervised method for learning human activities from unannotated video data. The method extracts temporal segments from video data and learns to recognize them through a contrastive learning framework. The method uses a two-stream architecture to separately model spatial and temporal features of the video data. The spatial features are extracted using a convolutional neural network (CNN) and the temporal features are extracted using a Long Short-Term Memory (LSTM) network. The learned features are then used for activity recognition through a classification model. The method is evaluated on several benchmarks and outperforms previous unsupervised methods for human activity recognition. The authors also demonstrate the effectiveness of their method in transferring knowledge to other related tasks, such as action detection and human-object interaction recognition.

2.2 INFERENCES FROM LITERATURE SURVEY

After going through the previous works we inferred some points. In Human activity recognition system based on the Android platform, they used the clustered approach to enhance the performance accuracy, and execution time but the classification times are dependent on the device models and capabilities. Real-time HAR model based on the smartphone inertial sensors used six different machine learning algorithms but they used inertial sensors. Novel elderly person tracking system using a machine learning algorithm. In this work, they used the k-NN model with a k value of 5, which was able to achieve the best accuracy of 96.40% in detecting the real-time activity of elderly people. Davide Anguita et al.'s paper, they introduced the improvised Support Vector Machine algorithm, which works with fixed point arithmetic to produce an energy-efficient model for the classification of human activities using a smartphone. Proposed model successfully achieved a recall of 99%, which was further compared to the existing deep learning models such as the RNN, Convolutional Neural Network (CNN), and Deep Belief Network (DBN). A deep

LSTM network for recognizing six different activities based on smartphone data. The network was able to achieve an accuracy of 96.70% on the UCI-HAD dataset.

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

While sensor-based human activity recognition (HAR) has shown great promise in recent years, there are still several open problems in the existing system that need to be addressed. Some of these open problems include:

- **Data Collection -** Collecting high-quality labelled data is essential for developing accurate HAR models. However, collecting such data is a time-consuming and expensive process. Moreover, the variability and complexity of human activities make it challenging to collect representative datasets that can cover a wide range of activities and scenarios.
- **Sensor Placement -** The placement of sensors on the body can significantly impact the accuracy of HAR models. The optimal sensor placement depends on the type of activity being performed, the sensor technology used, and the characteristics of the user. However, there is currently no standardized sensor placement protocol for HAR.
- **Cross-Subject Generalization -** The effectiveness of HAR models is typically evaluated using a leave-one-subject-out cross-validation approach, where the model is trained on data from one subject and tested on data from another subject. However, the performance of the model may deteriorate when tested on data from subjects that were not included in the training set. This is known as the cross-subject generalization problem and is a significant challenge in sensor-based HAR.
- **Real-Time Implementation -** Real-time implementation of sensor-based HAR is critical for many applications, such as fall detection or gait analysis. However, most existing sensor-based HAR systems have significant latency and computational requirements, making them unsuitable for real-time applications.
- **User Privacy -** Sensor-based HAR involves collecting sensitive data about individuals, such as their activity patterns and physiological data. Ensuring

user privacy and data security is essential for the widespread adoption of sensor-based HAR systems.

Addressing these open problems is critical for the widespread adoption of sensor-based HAR systems. New research is needed to develop solutions to these challenges, including improved data collection methods, standardized sensor placement protocols, and more efficient and interpretable models. By overcoming these challenges, sensor-based HAR has the potential to revolutionize healthcare, security, and other industries.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES OF THE PROJECT

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately, this condition does not prevail in practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill- conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are:

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

3.1.1 Economic Feasibility

- **Computing resources -** Training CNN-LSTM models for activity recognition requires high computing resources, such as GPUs and high memory capacity, which may increase the cost of implementation.
- **Maintenance and Upkeep -** Regular maintenance of sensors and computing resources may add to the overall cost of implementation.
- **Cost of Development -** Developing and implementing the CNN-LSTM model requires skilled professionals, and the cost may vary based on the size and complexity of the project.

3.1.2 Technical Feasibility

- **Availability of data -** The availability of labeled data covering a diverse range of human activities and environmental conditions is essential for the success of the CNN-LSTM model.

- **Model architecture -** The design of the CNN-LSTM model plays a critical role in the performance of activity recognition. Hyperparameter tuning and feature selection are also important factors to consider.

3.1.3 Social Feasibility:

- **Privacy Concerns -** Collecting data on individuals activities may raise privacy concerns. Proper consent and transparency must be ensured during the data collection and usage process.
- **Social Acceptance -** Acceptance of the technology and willingness to use it in public settings must be considered.
- **Social Benefits -** Human activity recognition can have significant benefits, such as improving healthcare, sports training, and rehabilitation programs, which can improve people's quality of life.

3.2 REQUIREMENTS SPECIFICATION

3.2.1 Hardware Requirements

• RAM	: 4GB or above 4GB
• Processor of Frequency	: 1.5GHz or above
• Processor	: Intel Pentium 4 or higher

3.2.2 Software Requirements

• Operating System	: Windows 8 and above
• Languages	: Python
• Tools used	: Visual Studio Code, Jupyter Notebook

3.2.3 Python

Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

Python Features

- **Easy-to-learn** - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- **Easy-to-read -** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain -** Python's source code is relatively simple to maintain.
- **A broad standard library -** The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode -** Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- **Portable -** Python can operate on a broad range of hardware devices and has the same interface across them all.
- **Extendable -** The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- **Databases -** Python provides interfaces to all major commercial databases.
- **GUI Programming -** Python can construct and port GUI programmes to numerous system calls, libraries, and windows systems, including Windows MFC, Macintosh, and Unix's X Window system.
- **Scalable -** Python provides a better structure and support for large programs than shell scripting.

3.2.4 Deep Learning

Deep learning is a type of machine learning that uses algorithms meant to function in a manner similar to the human brain. While the original goal for AI was broadly to make machines able to do things that would otherwise require human intelligence, the idea has been refined in the decades since. François Chollet, AI researcher at Google and creator of the machine learning software library Keras, says: “Intelligence is not a skill in itself, it’s not about what you can do, but how well and how efficiently you can learn new things.

3.2.5 Artificial Neural Network

Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions. The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

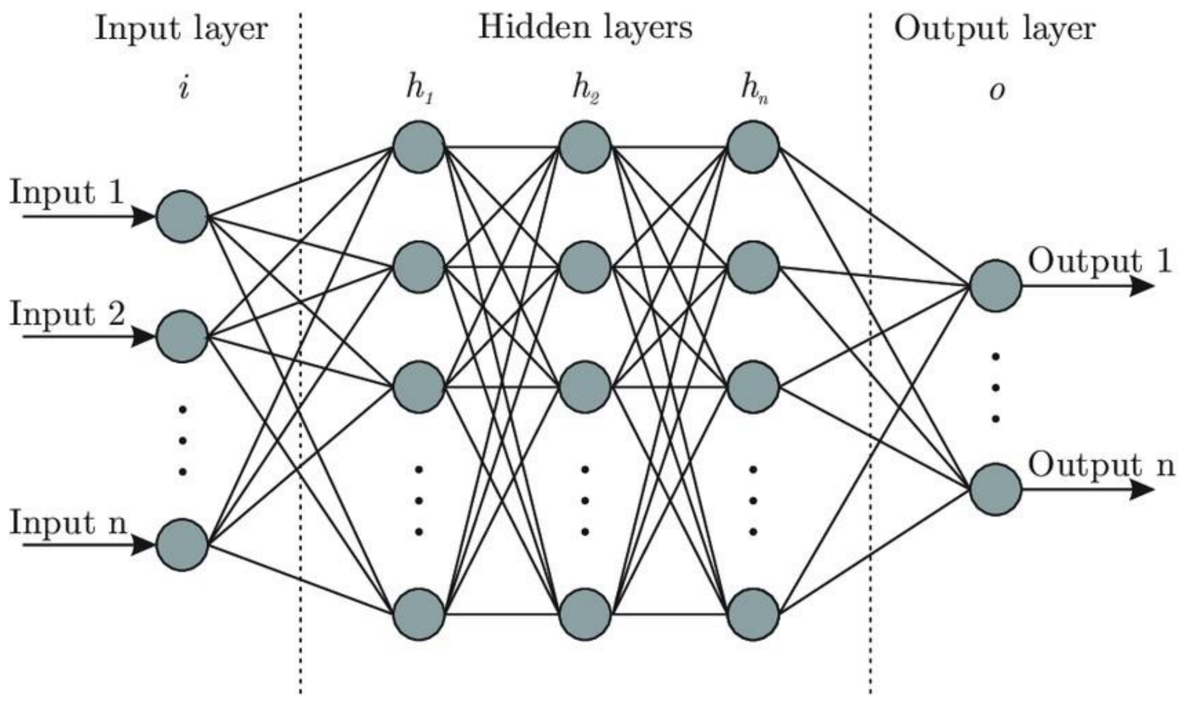


Fig. 3.1: Structure of Artificial Neural Network

3.2.6 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field. In this post, you will get insight into LSTMs using the words of research scientists that developed the methods and applied them to new and important problems. There are few that are better at clearly and precisely articulating both the promise of LSTMs and how they work than the experts that developed them.

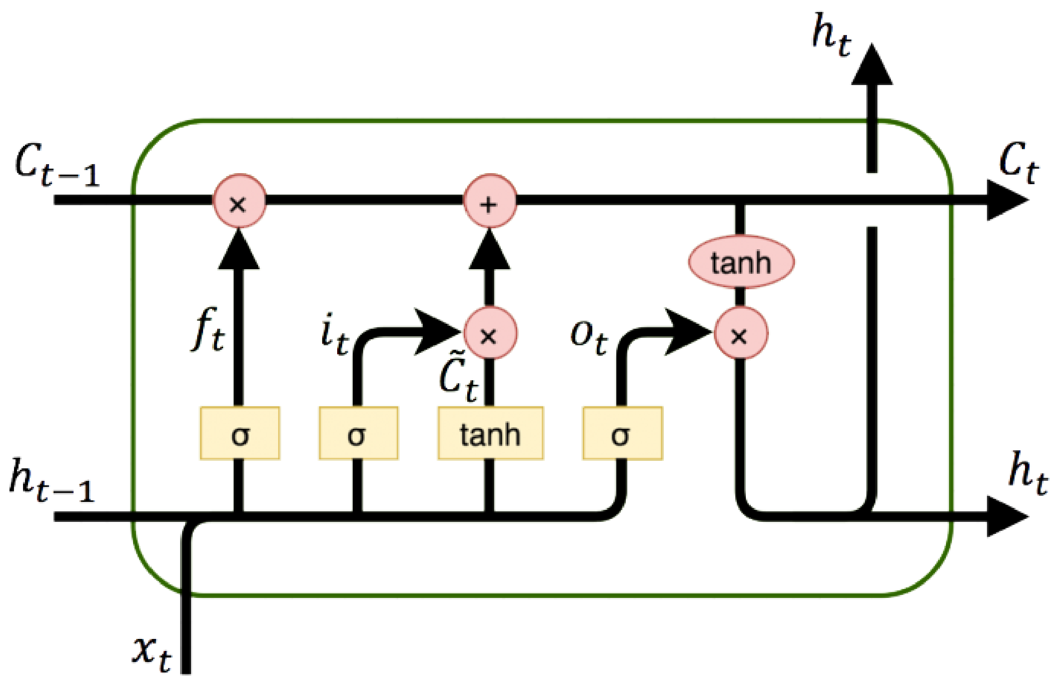


Fig. 3.2: Structure of Long Short-Term Memory

3.2.7 Convolutional Neural Network

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any preprocessing. A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces. The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

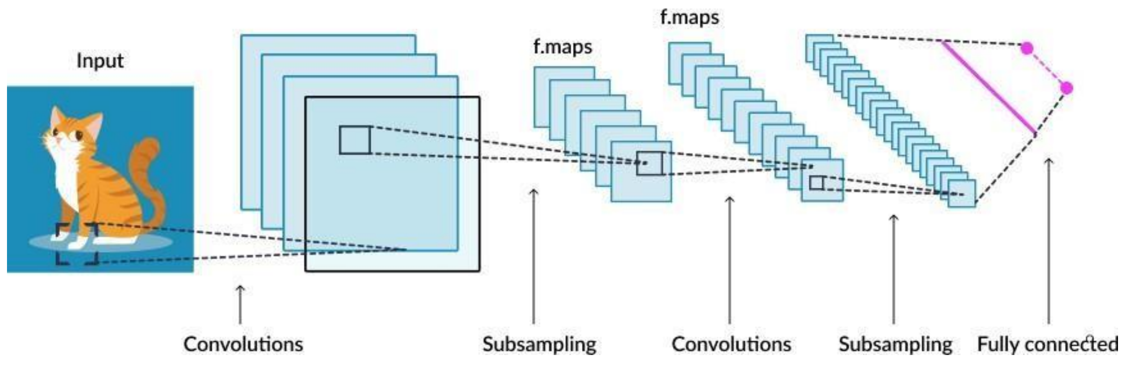


Fig: 3.3: Structure of Convolutional Neural Network

3.2.8 ConvLSTM

ConvLSTM is a type of recurrent neural network (RNN) that combines the convolutional neural network (CNN) and LSTM (long short-term memory) architectures. It is a powerful model that can be used for various tasks, including video prediction, image processing, and natural language processing. The ConvLSTM network is designed to learn spatial and temporal dependencies in sequential data. It consists of convolutional layers, which extract features from the input data, and LSTM layers, which capture the temporal dependencies in the data. The combination of these layers enables the ConvLSTM network to learn both spatial and temporal patterns in the input data.

The ConvLSTM architecture is similar to the standard LSTM architecture, with the addition of convolutional layers before the LSTM layers. The convolutional layers help in reducing the spatial dimensions of the input data and extracting relevant features. The output of the convolutional layers is then passed to the LSTM layers, which capture the temporal dependencies in the data. The ConvLSTM architecture has shown significant improvements in various tasks such as video prediction, where it can learn long-term dependencies in sequential data. It is also useful in image processing tasks, where it can capture the spatial dependencies in the data.

In human activity recognition, ConvLSTM models have shown promising results in accurately classifying physical activities by capturing both spatial and temporal features in the sensor data.

3.2.9 LRCN

LRCN (Long-term Recurrent Convolutional Networks) is a type of neural network architecture that combines the strengths of CNN (Convolutional Neural Networks) and LSTM (Long Short-Term Memory) models. The CNN part of the LRCN architecture is used to extract spatial features from the input data, such as images or video frames, while the LSTM part is used to capture the temporal dependencies between these features. The CNN layers are typically used as a feature extractor, while the LSTM layers are responsible for learning the temporal patterns in the extracted features.

In human activity recognition, LRCN models have shown promising results in accurately classifying physical activities by capturing both spatial and temporal features in the sensor data. The CNN layers can learn useful visual features from raw sensor data, while the LSTM layers can capture the temporal dependencies between these features, allowing the model to accurately recognize complex activity sequences.

Overall, LRCN models have been shown to be effective in various tasks such as video analysis, image captioning, and speech recognition, where they can learn both spatial and temporal patterns in the data.

3.2.10 Tensorflow

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

3.2.11 Keras

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

Advantages of Keras

Keras has several advantages that make it a popular choice for deep learning.

- **Simplicity -** Keras is very easy and simple to use. It is a user-friendly API with easy-to-learn and code features.
- **Backend support -** Keras does not operate with low-level computations. So, it supports the use of backends.
- **Pre-trained models -** Keras provides numerous pre-trained models.
- **Fast experimentation -** Keras is built to simplify the tasks of users.
- **Great community and calibre documentation -** Keras has a large supportive community.

3.3 SYSTEM USE CASE

3.3.1 System Description

The system is designed to recognize human activities using videos as input. The system utilizes two deep learning models, Convolutional LSTM (ConvLSTM) and Long-term Recurrent Convolutional Networks (LRCN), to extract and classify features from videos, respectively.

3.3.2 System Use Case Scenario

- User uploads a video containing a human activity to the system.
- The system preprocesses the video by dividing it into frames and resizing each frame to a standard size.
- The ConvLSTM model is used to extract features from the video frames. The ConvLSTM model takes in a sequence of video frames as input and outputs a sequence of feature maps that capture the temporal dependencies between frames.
- The LRCN model is used to classify the human activity based on the extracted features. The LRCN model takes in the sequence of feature maps as input and outputs the predicted activity label.
- The system displays the predicted activity label to the user.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

After analyzing the drawbacks of previous works, we proposed a new system. The proposed system detects human activity without the use of additional sensors. It only depends on the camera feed. In this proposed system, Data of human activity is taken over fixed time interval for determining the activity and fed to the model for detection. The output shows the activity being performed by the human.

Advantages of Proposed System

No initial setup is required before implementation. In the existing system, sensors are used to detect the human activity. Here we don't need any sensors. Hence, Sensor cost is eliminated. We need not to depend on Sensors. Easy to enhance and add activities to current model without any additional hardware requirements.

4.1 DESCRIPTION OF RESEARCH APPROACH AND METHODS

There are many approaches to do the video classification. The selected process is combination of CNN and LSTM. CNN will extract spatial features at a given time step in the input sequence. After that LSTM will be used to identify relations between frames.

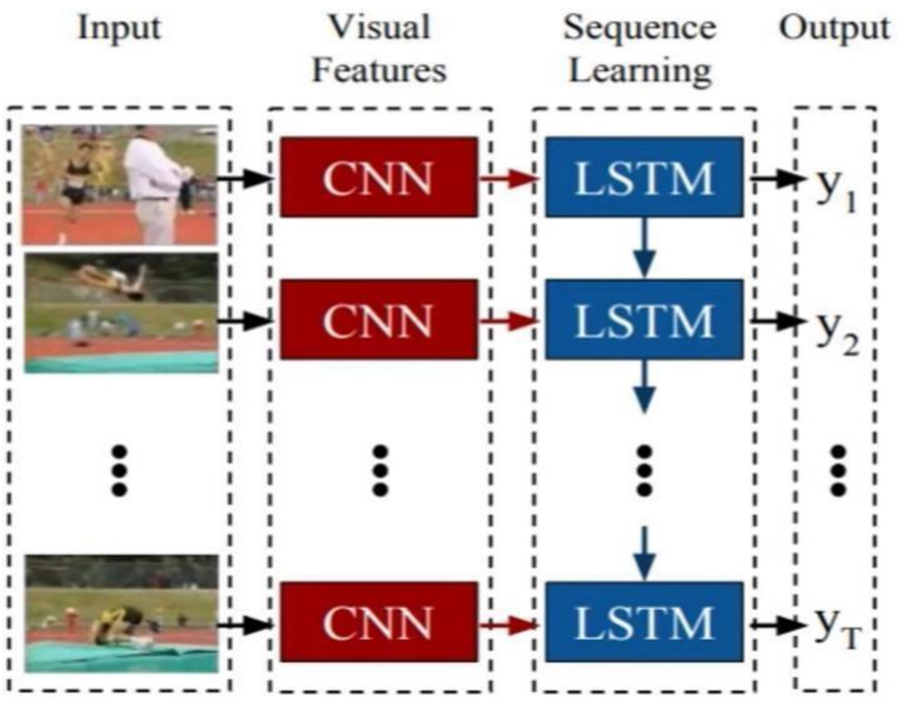


Fig: 4.1: Methodology of Human Activity Recognition

4.1.1 Data Collection and Pre-processing

The first step in HAR using ConvLSTM and LRCN models is to collect the data from the sensors. The sensors used in the system typically include accelerometers, gyroscopes, and magnetometers. The data collected from these sensors is typically noisy and requires pre-processing to remove noise and artifacts. The data is then segmented into windows of fixed length, typically ranging from 1-10 seconds. The length of the window depends on the activity being recognized and the sampling frequency of the sensor.

4.1.2 Feature Extraction

Once the data has been pre-processed and segmented, the next step is to extract features from the sensor data. ConvLSTM and LRCN models are capable of learning the features from the data directly. However, feature extraction can help improve the accuracy of the models. There are various feature extraction techniques that can be used, including statistical features such as mean, standard deviation, and variance, and time-domain features such as zero-crossing rate and energy. The extracted features are then used as input to the ConvLSTM and LRCN models.

4.1.3 ConvLSTM and LRCN Models

ConvLSTM and LRCN models are two types of deep learning models that have been shown to be effective in HAR. ConvLSTM is a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The CNNs are used to extract features from the input data, while the LSTM networks are used to capture the temporal dependencies in the data. LRCN is a combination of CNNs and Recurrent Neural Networks (RNNs), where the CNNs are used to extract features and the RNNs are used to capture the temporal dependencies.

4.1.4 Training and Testing

Once the models have been developed, the next step is to train them on the labelled dataset of physical activities. The labelled dataset typically includes a set of physical activities such as walking, running, and cycling. The dataset is divided into training and testing sets, with a larger portion of the dataset used for training the models. The models are trained using backpropagation and stochastic gradient descent. Once the models have been trained, they are tested on the testing set to evaluate their accuracy.

4.1.5 Evaluation Metrics

The accuracy of the models is evaluated using various evaluation metrics such as accuracy, precision, recall, and F1-score. The accuracy of the models is the proportion of correctly classified physical activities. The precision is the ratio of correctly classified physical activities to the total number of physical activities classified as positive. The recall is the ratio of correctly classified physical activities to the total number of actual positive physical activities. The F1-score is the harmonic mean of precision and recall.

4.2 ARCHITECTURE OF PROPOSED SYSTEM

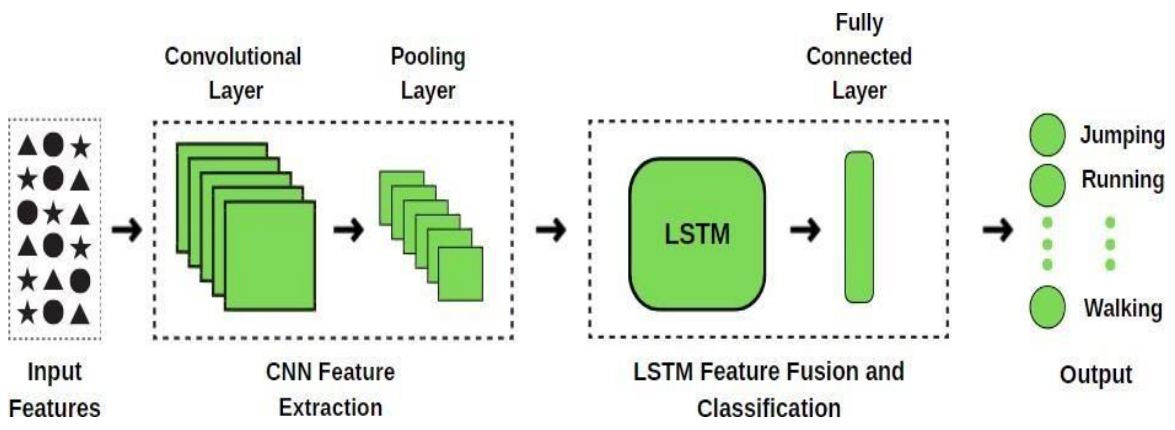


Fig. 4.2: Architecture Diagram

The architecture of a human activity recognition system using CNN and LSTM with ConvLSTM and LRCN models consists of several layers that perform feature extraction, classification, and prediction tasks.

The input to the system is a time series of data captured by a camera. The raw data is first pre-processed to remove noise, filter out unwanted frequencies, and normalize the data. The pre-processed data is then fed into the feature extraction layer, which uses a combination of convolutional and pooling layers to extract meaningful features from the input data.

The output of the feature extraction layer is a sequence of feature maps, which are then fed into the classification layer. The classification layer consists of one or more LSTM layers, which learn to model the temporal dependencies in the input data and perform classification of the activities based on the extracted features.

In the ConvLSTM model, the input data is fed into a ConvLSTM layer, which combines the functionality of convolutional and LSTM layers. The ConvLSTM layer learns to capture both spatial and temporal features in the input data and performs classification based on these features.

In the LRCN model, the input data is fed into a CNN layer, which extracts spatial features from the input data. The output of the CNN layer is then fed into an LSTM layer, which learns to model the temporal dependencies in the input data and performs classification based on the extracted features.

The output of the classification layer is a probability distribution over the set of possible activities, which is then used to predict the most likely activity at each time step.

The architecture of the human activity recognition system using CNN and LSTM with ConvLSTM and LRCN models is designed to be scalable and adaptable to different types of sensor data and activity recognition tasks. The system can be trained using supervised learning techniques, such as backpropagation and gradient descent, to optimize the model parameters and improve the classification performance.

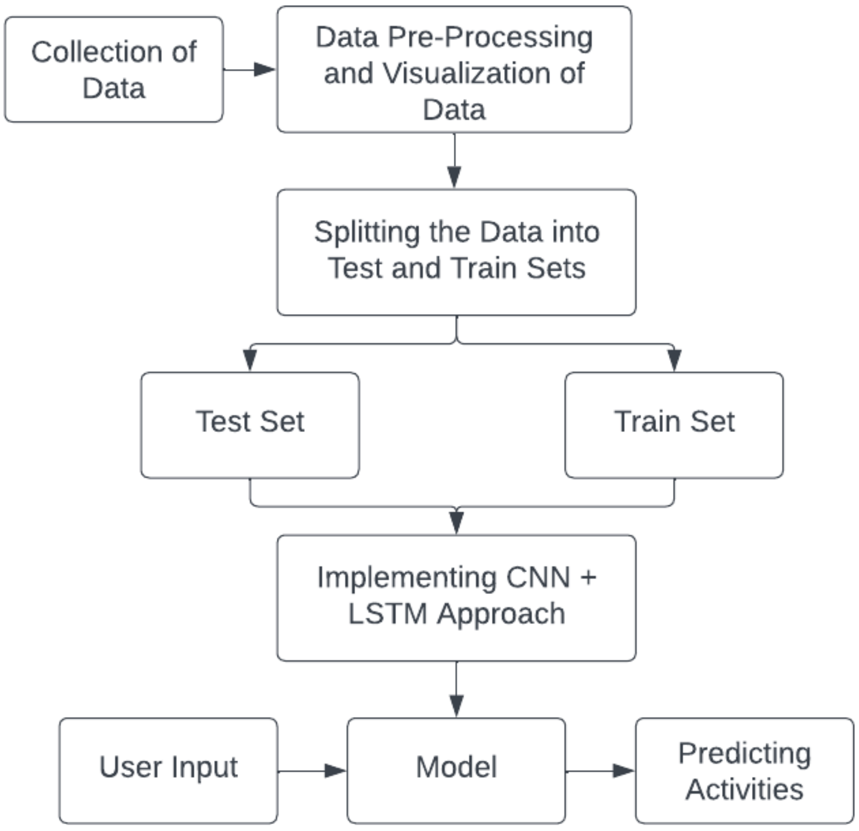

Collection of Data
Data Pre-Processing and Visualization of Data
Splitting the Data into Test and Train Sets
Test Set
Train Set
Implementing CNN + LSTM Approach
User Input
Model
Predicting Activities

Fig. 4.3: Flow Diagram

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED SYSTEM

The software for implementation and testing of the proposed human activity recognition system using CNN and LSTM algorithms can include several components, such as:

- **Data collection and pre-processing -** This component involves collecting data from mobile cameras, security cameras, etc., and pre-processing the data to ensure that it is in the appropriate format for the model. We have taken UCF50 as our dataset which contains 50 different categories of actions. This component can be implemented using Python libraries, such as Pandas, NumPy, and Scikit-Learn.

- **Model development and training -** This component involves developing and training the CNN and LSTM models to recognize human activities based on the preprocessed data. This component can be implemented using deep learning frameworks, such as TensorFlow or PyTorch.
- **Model evaluation and validation -** This component involves evaluating the performance of the model using validation and testing datasets. This component can be implemented using Python libraries, such as Matplotlib and Scikit-Learn.
- **Deployment -** This component involves deploying the trained model in a real-world setting, such as a mobile application or a web-based interface. This component can be implemented using software development frameworks, such as Flask or Django.
- **Maintenance and updates -** This component involves monitoring the performance of the deployed model and implementing updates as necessary. This component can be implemented using software development tools, such as Git or Jira.

The testing plan for the proposed human activity recognition system can include several stages, such as:

- **Unit testing -** This involves testing the individual components of the system, such as the data collection and pre-processing, model development and training, and model evaluation and validation, to ensure that they are working as expected.

- **Integration testing -** This involves testing the integration of the different components of the system to ensure that they are working together as expected.

- **Performance testing -** This involves testing the performance of the system, such as the accuracy, precision, and recall of the model, to ensure that it meets the required performance metrics.

- **Acceptance testing -** This involves testing the system in a real-world setting with end-users to ensure that it is meeting their needs and requirements.

Overall, the software for implementation and testing plan of the proposed human activity recognition system using CNN and LSTM algorithms should be designed to ensure that the system is accurate, efficient, and reliable, and can be deployed and maintained effectively.

4.4 PROJECT MANAGEMENT PLAN

- **Project Planning and Management Module -** This module will include the overall planning and management of the project, including setting project goals and timelines, creating and assigning tasks, tracking progress, and communicating with team members.
- **Data Collection and Pre-processing Module -** This module will involve collecting and preprocessing sensor data from various sources, such as wearable devices, smartphones, or cameras. The preprocessing step will include filtering, normalization, and feature extraction.
- **ConvLSTM Model Module -** This module will implement the ConvLSTM model for human activity recognition. It will include designing the model architecture, training the model on the preprocessed data, and evaluating the performance of the model on a validation dataset.
- **LRCN Model Module -** This module will implement the LRCN model for human activity recognition. It will include designing the model architecture, training the model on the preprocessed data, and evaluating the performance of the model on a validation dataset.
- **Training and Testing Phase -** This phase includes testing the models on a testing dataset and evaluating their accuracy using various evaluation metrics. The testing phase is expected to take two weeks.

The project management plan with modules for human activity recognition will allow for a well-organized and structured approach to developing the system, ensuring that each component is carefully designed, implemented, and tested before integrating them into the final product. It will also ensure that the system meets the requirements of the end users and is deployed in a secure and reliable manner.

4.5 FINANCIAL REPORT ON ESTIMATED COSTING

Deep Learning techniques such as LSTM and CNN has greatly improved the accuracy of HAR systems. However, developing a HAR system using LSTM and CNN can be a complex and resource-intensive process that requires careful planning and budgeting. In this financial report, we will provide an estimate of the cost of developing a HAR system using LSTM and CNN

- **Cost Estimate -** Hardware and Software Costs: The hardware and software cost for developing a HAR system using LSTM and CNN can be significant. This includes the cost of purchasing and maintaining high-performance GPUs, cloud computing services, and specialized software such as TensorFlow or Keras.
- **Data Acquisition and Pre-processing Costs -** Collecting, annotating, and preprocessing the data required for training and testing the LSTM and CNN models can be a significant expense. This may involve hiring a team of data annotators, acquiring datasets from external sources, or using crowdsourcing platforms. Additionally, the cost of storing and managing large datasets can be significant.
- **Model Development Costs -** Developing and optimizing the LSTM and CNN models can be a time-consuming and resource-intensive process. This may involve hiring machine learning experts or outsourcing the development work to a third-party provider. Additionally, the cost of testing and validating the models can be significant, as this requires access to large datasets and specialized software tools.
- **Deployment and Maintenance Costs -** Once the models have been developed, they need to be deployed and integrated into the target system. This may involve additional costs for server infrastructure, API development, and ongoing maintenance and support. The cost of ongoing maintenance and support can be significant, especially if the system is deployed in a complex environment with multiple users or if the models need to be updated frequently.

Developing a HAR system using LSTM and CNN can be a complex and resource-intensive process that requires careful planning and budgeting. The cost of developing a HAR system can vary greatly depending on the size and complexity of the system, the data sources and size, the programming languages and tools used, and the expertise of the developers. Therefore, it is important to carefully assess the requirements of the project and work with experienced developers and data scientists to ensure the success of the project.

4.6 SOFTWARE TO OPERATIONS PLAN

Human activity recognition (HAR) is an important technology that can be used in various domains such as healthcare, sports, and security. Developing a HAR system involves various stages such as data collection, model development, testing, and deployment. Once the system is developed and tested, it needs to be transitioned to operations to ensure its long-term sustainability and effectiveness. This requires a well-planned project transition/ software to operations plan. In this document, we will outline the project transition/ software to operations plan for our HAR system.

- **Monitoring -** The HAR system should be monitored for performance, errors, and other issues on a regular basis. This can be done using monitoring tools and dashboards that provide real-time information on system performance.
- **Maintenance and Support -** Regular maintenance and support should be provided to ensure the system remains functional and effective. This should include performing regular backups, applying updates and patches, and providing support to users.
- **Security -** The HAR system should be regularly audited for security vulnerabilities and measures should be taken to address any issues that are identified. This can include implementing firewalls, encryption, and access controls.
- **Upgrades and Enhancements -** The HAR system should be periodically upgraded and enhanced to ensure that it remains up-to-date with the latest technology and requirements. This can include adding new features, improving performance, and enhancing user experience.

- **Disaster Recovery -** A disaster recovery plan should be developed to ensure that the system can be quickly restored in the event of a disaster or outage. This should include regular backups and a plan for restoring the system in the event of a catastrophic failure.

Developing HAR system is a complex process that requires careful planning and execution. Once the system is developed and tested, it needs to be transitioned to operations to ensure its long-term sustainability and effectiveness. This requires a well-planned project transition/ software to operations plan. The plan should include establishing a project transition team, defining transition requirements, developing a deployment plan, developing a maintenance and

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

To develop and deploy a human activity recognition system using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), you can follow the steps below:

- **Data Collection and Pre-processing -** Collect data from various sources and pre-process it to make it suitable for machine learning. You can use various techniques for data pre-processing, such as normalization, data augmentation, and feature extraction.
- **Model Architecture Design -** Design the architecture of the CNN-LSTM model. This architecture should include convolutional layers for feature extraction from the input data, LSTM layers for modeling time-series data, and fully connected layers for classification.
- **Model Training -** Train the CNN-LSTM model using the preprocessed data. You can use various optimization algorithms, such as Adam, RMSprop, or SGD, to minimize the loss function.
- **Model Evaluation -** Evaluate the performance of the trained model using various metrics, such as accuracy, precision, recall, and F1 score. You can also use visualization techniques, such as confusion matrices or ROC curves, to analyze the performance of the model.
- **Deployment -** Deploy the model in a production environment. You can use various frameworks, such as TensorFlow or PyTorch, to create a deployable model. You can also use cloud platforms, such as AWS or Azure, to deploy the model in a scalable and cost-effective way.
- **Continuous Improvement -** Continuously improve the model by retraining it on new data and fine-tuning the hyperparameters. You can also use techniques such as transfer learning to improve the model's performance.

Overall, the development and deployment setup for human activity recognition using CNN-LSTM involves several steps, including data collection and preprocessing, model architecture design, model training, model evaluation, deployment, and continuous improvement. With proper planning and execution, you can develop a robust and accurate human activity recognition system that can be deployed in a real-world environment.

5.2 ALGORITHMS

Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) are two types of neural network architectures used in machine learning for various applications, including image recognition, natural language processing, and time series analysis.

5.2.1 Convolutional Neural Network (CNN)

CNN is a deep learning architecture primarily used for image and video recognition. CNNs use a series of convolutional and pooling layers to extract and learn features from input images, followed by fully connected layers for classification. The convolutional layer applies a filter or kernel to the input image, sliding it across the image and computing a dot product at each position. The pooling layer then reduces the spatial size of the image by aggregating the output of the previous layer. This process of convolution and pooling is repeated multiple times to extract higher-level features from the input image. CNNs are highly effective for image recognition tasks and have achieved state-of-the-art performance on various benchmark datasets.

5.2.2 Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) that is designed to model sequences of data, such as speech, text, and time series. Unlike traditional RNNs, which suffer from the vanishing gradient problem when modelling long sequences, LSTM uses a gated memory mechanism to store and retrieve information over time. Each LSTM cell has three gates - input gate, forget gate, and output gate - that control the flow of information into, out of, and within the cell. The input gate decides which information to update, the forget gate decides which information to discard, and the output gate decides which information to output. The LSTM architecture allows for the modelling of long-term dependencies in the input data and has been widely used for various applications, including speech recognition, language translation, and time series analysis.

Overall, CNN and LSTM are two powerful neural network architectures that have been used in various machine learning applications. While CNN is mainly used for image and video recognition, LSTM is used for modelling sequences of data. Both architectures have achieved state-of-the-art performance on various benchmark datasets and continue to be widely used and researched in the machine learning community.

5.3 TESTING

When using CNN and LSTM algorithms for human activity recognition, the following testing techniques can be used to evaluate their performance:

- **Hold-Out Testing -** In this technique, the dataset is divided into two parts: a training set and a testing set. The model is trained on the training set and then evaluated on the testing set to determine its accuracy.
- **Cross-Validation -** As described earlier, this technique can also be used for CNN and LSTM algorithms to evaluate their performance.
- **Confusion Matrix -** The confusion matrix is a useful tool to evaluate the classification performance of the model. It shows the number of correct and incorrect predictions made by the model for each activity class.
- **F1 Score -** The F1 score is a measure of the model's accuracy, calculated as the harmonic mean of precision and recall. It is a useful metric for evaluating the overall performance of the model.

In addition to these techniques, other measures can be used to evaluate the performance of the model, such as accuracy, precision, recall, and the area under the ROC curve. These measures can be used to compare the performance of different CNN and LSTM models and to select the best model for a given application.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 PRESENTATION OF FINDINGS

We have used UCF50 Dataset which contains 50 different types of action categories. Two different types of algorithms (CNN and LSTM) are used for Training and Testing the Models. We splitted the data into Training and Testing sets in the ratio 80:20, which is 80% for the training set and rest is for testing set. ConvLSTM and LRCN are two models used for training. We achieved 77% accuracy for ConvLSTM Model and 87% accuracy for LRCN Model.

6.2 DATA ANALYSIS AND INTERPRETATION

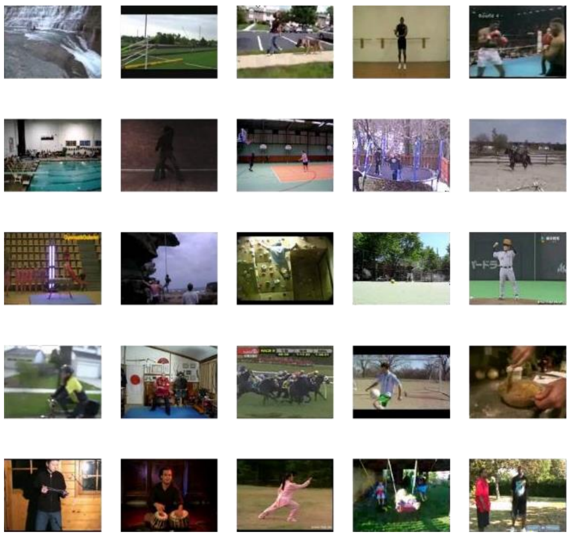
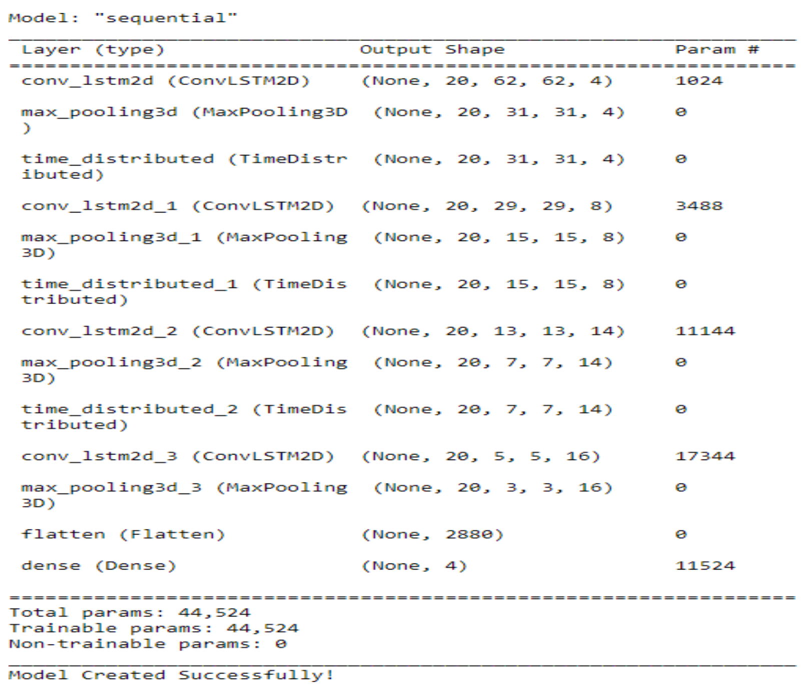


Fig. 6.2.1: Dataset Used

We have used UCF50 Dataset, which contains 50 different types of action categories



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d (ConvLSTM2D)    (None, 20, 62, 62, 4)     1024

 max_pooling3d (MaxPooling3D  (None, 20, 31, 31, 4)    0
 )

 time_distributed (TimeDistr  (None, 20, 31, 31, 4)    0
 ibuted)

 conv_lstm2d_1 (ConvLSTM2D)  (None, 20, 29, 29, 8)     3488

 max_pooling3d_1 (MaxPooling  (None, 20, 15, 15, 8)    0
 3D)

 time_distributed_1 (TimeDis  (None, 20, 15, 15, 8)    0
 tributed)

 conv_lstm2d_2 (ConvLSTM2D)  (None, 20, 13, 13, 14)    11144

 max_pooling3d_2 (MaxPooling  (None, 20, 7, 7, 14)     0
 3D)

 time_distributed_2 (TimeDis  (None, 20, 7, 7, 14)     0
 tributed)

 conv_lstm2d_3 (ConvLSTM2D)  (None, 20, 5, 5, 16)      17344

 max_pooling3d_3 (MaxPooling  (None, 20, 3, 3, 16)     0
 3D)

 flatten (Flatten)           (None, 2880)              0

 dense (Dense)               (None, 4)                 11524

=================================================================
Total params: 44,524
Trainable params: 44,524
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

Fig. 6.2.2: Creation of ConvLSTM Model

Here we have our LRCN Model's Loss and Accuracy Curves from our Training and Testing Steps.

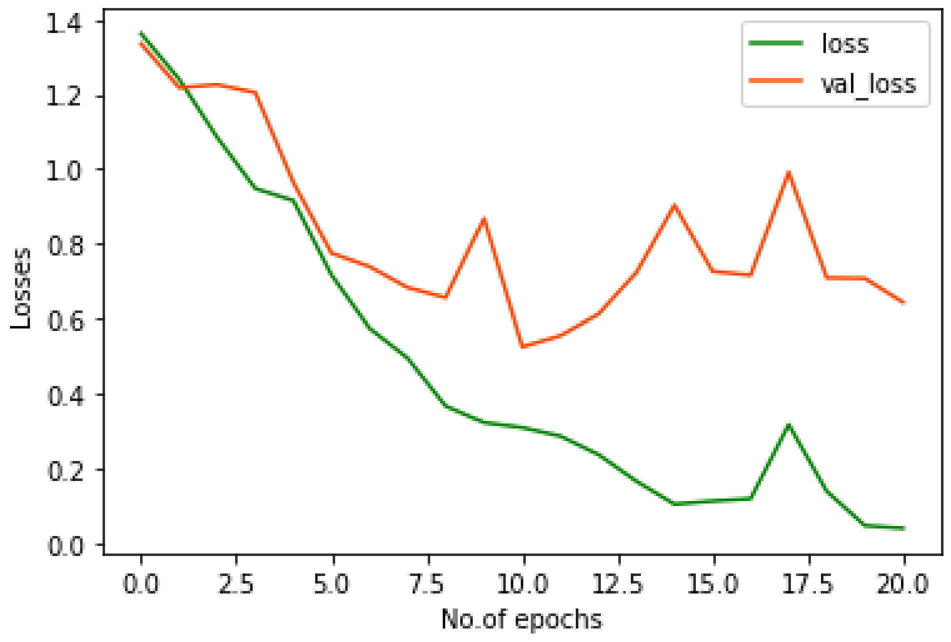


Fig. 6.2.3: Total Loss vs Total Validation Loss of ConvLSTM Model

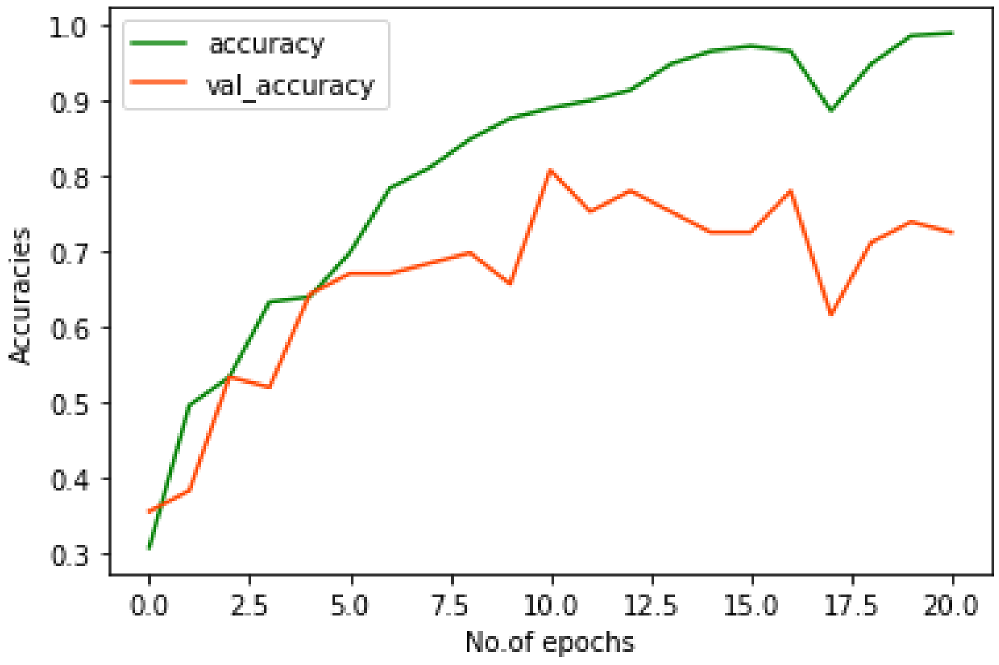
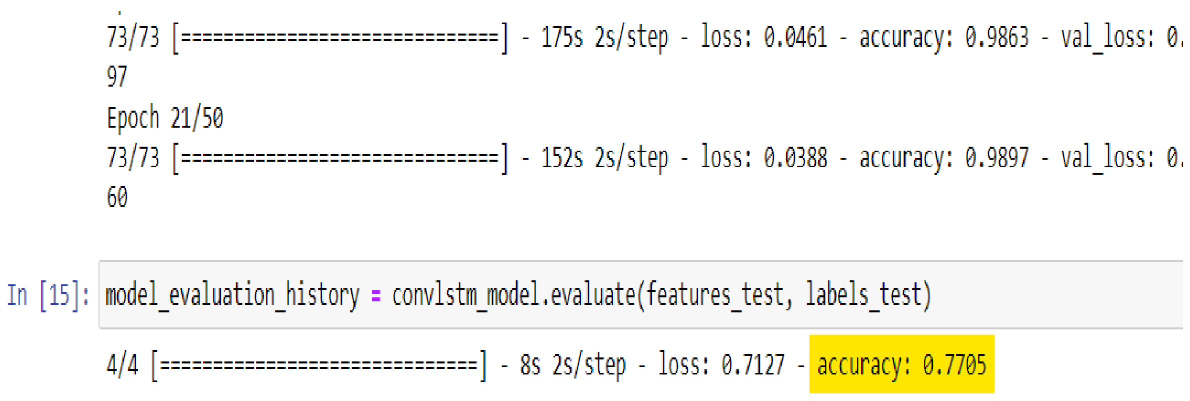


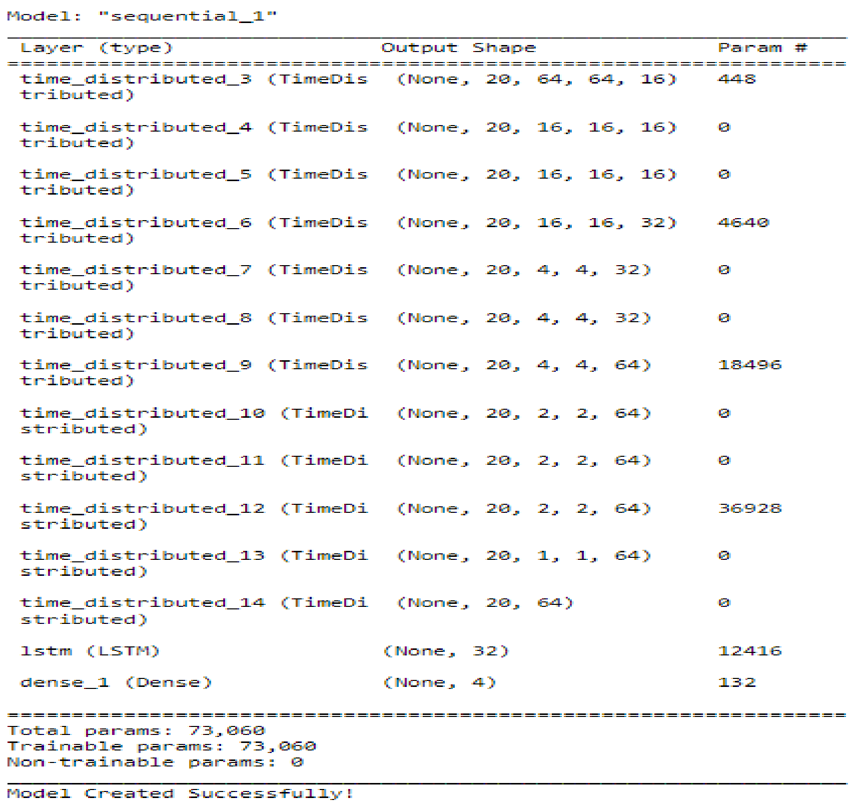
Fig. 6.2.4: Total Accuracy vs Total Validation Accuracy of ConvLSTM Model



```
73/73 [==============================] - 175s 2s/step - loss: 0.0461 - accuracy: 0.9863 - val_loss: 0.
97
Epoch 21/50
73/73 [==============================] - 152s 2s/step - loss: 0.0388 - accuracy: 0.9897 - val_loss: 0.
60

In [15]: model_evaluation_history = convlstm_model.evaluate(features_test, labels_test)

4/4 [==============================] - 8s 2s/step - loss: 0.7127 - accuracy: 0.7705
```

Fig: 6.2.5: Accuracy of ConvLSTM Model

We achieved an accuracy of 77% for the ConvLSTM Model after Training and Testing.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_3 (TimeDis  (None, 20, 64, 64, 16)   448
 tributed)

 time_distributed_4 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_5 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 16, 16, 32)   4640
 tributed)

 time_distributed_7 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_9 (TimeDis  (None, 20, 4, 4, 64)     18496
 tributed)

 time_distributed_10 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time_distributed_11 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time_distributed_12 (TimeDi  (None, 20, 2, 2, 64)     36928
 stributed)

 time_distributed_13 (TimeDi  (None, 20, 1, 1, 64)     0
 stributed)

 time_distributed_14 (TimeDi  (None, 20, 64)           0
 stributed)

 lstm (LSTM)                 (None, 32)                12416

 dense_1 (Dense)             (None, 4)                 132

=================================================================
Total params: 73,060
Trainable params: 73,060
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

Fig. 6.2.6: Creation of LRCN Model

The above picture shows the creation of LRCN Model.

Here we have our LRCN Model's Loss and Accuracy Curves from our Training and Testing Steps.

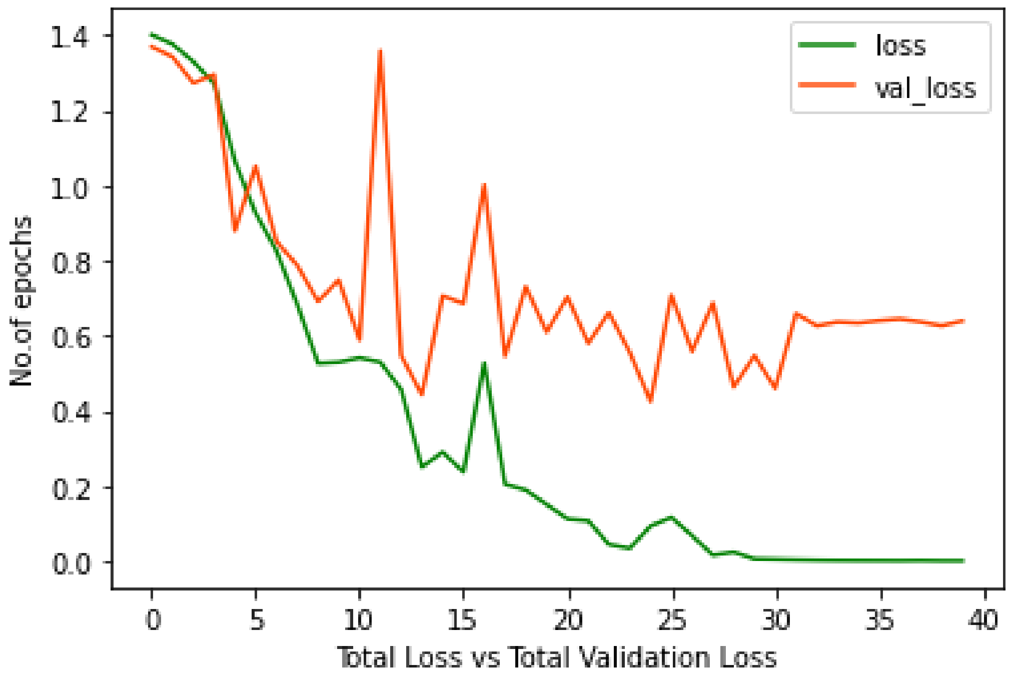


Fig. 6.2.7: Total Loss vs Total Validation Loss of LRCN Model

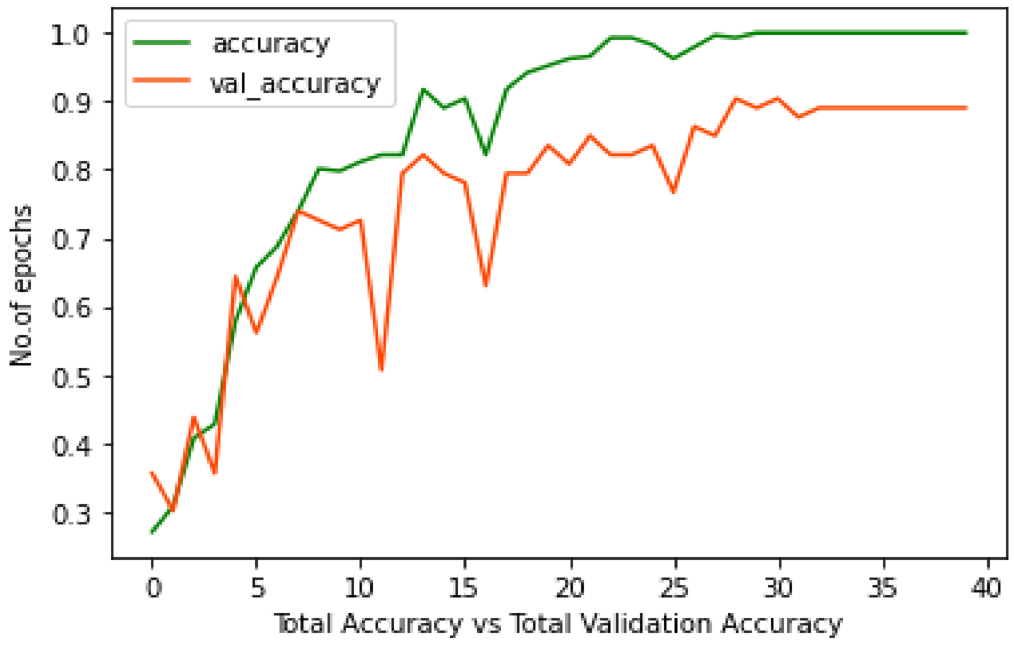
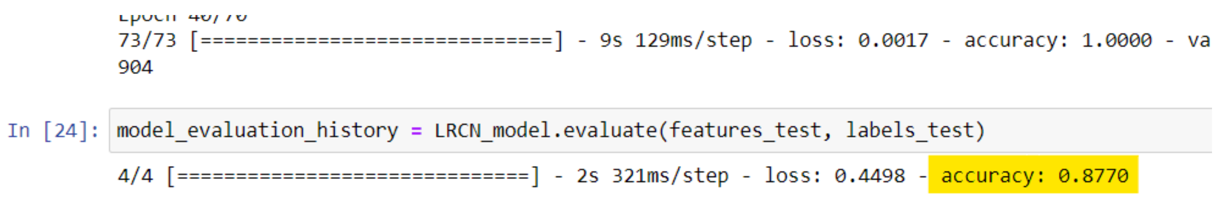


Fig. 6.2.8: Total Accuracy vs Total Validation Accuracy of LRCN Model



```
Epoch 40/70
73/73 [==============================] - 9s 129ms/step - loss: 0.0017 - accuracy: 1.0000 - va
904

In [24]: model_evaluation_history = LRCN_model.evaluate(features_test, labels_test)

         4/4 [==============================] - 2s 321ms/step - loss: 0.4498 - accuracy: 0.8770
```

Fig. 6.2.9: Accuracy of LRCN Model

After training and testing the LRCN Model we got an accuracy of 87%.

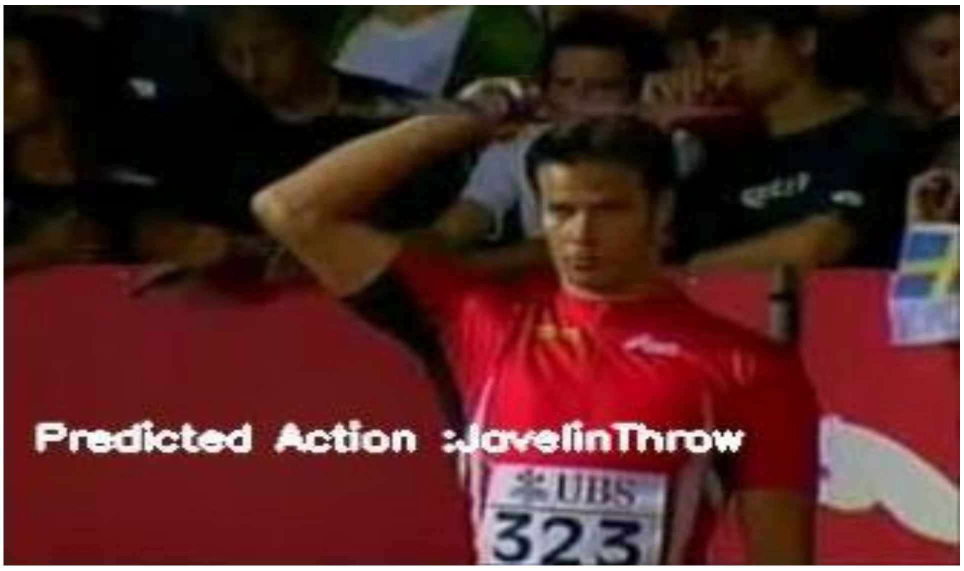


Fig. 6.2.10: Picture showing the predicted action as Javelin Throw

The above picture shows the predicted action of human is Javelin Throw.

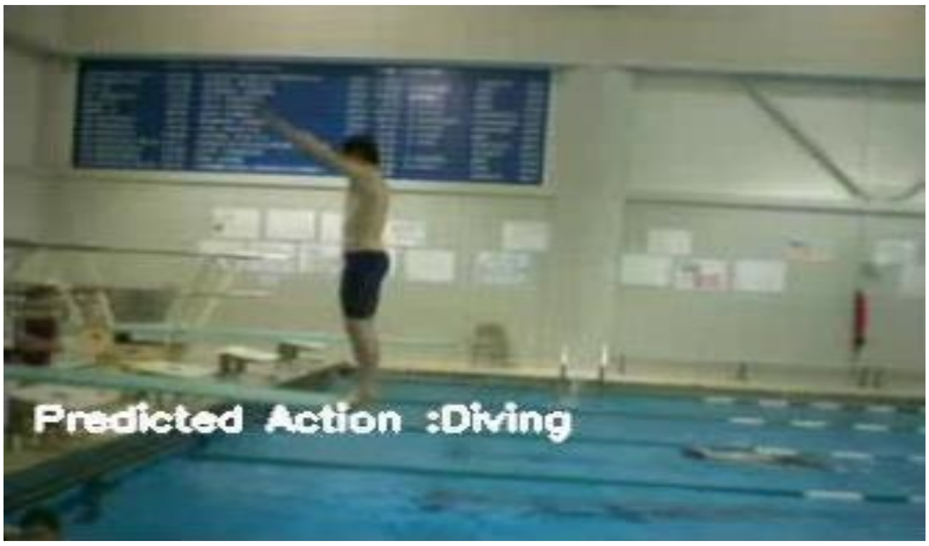


Fig. 6.2.11: Picture showing the predicted action as Diving

The above picture shows the predicted action of human is Diving.

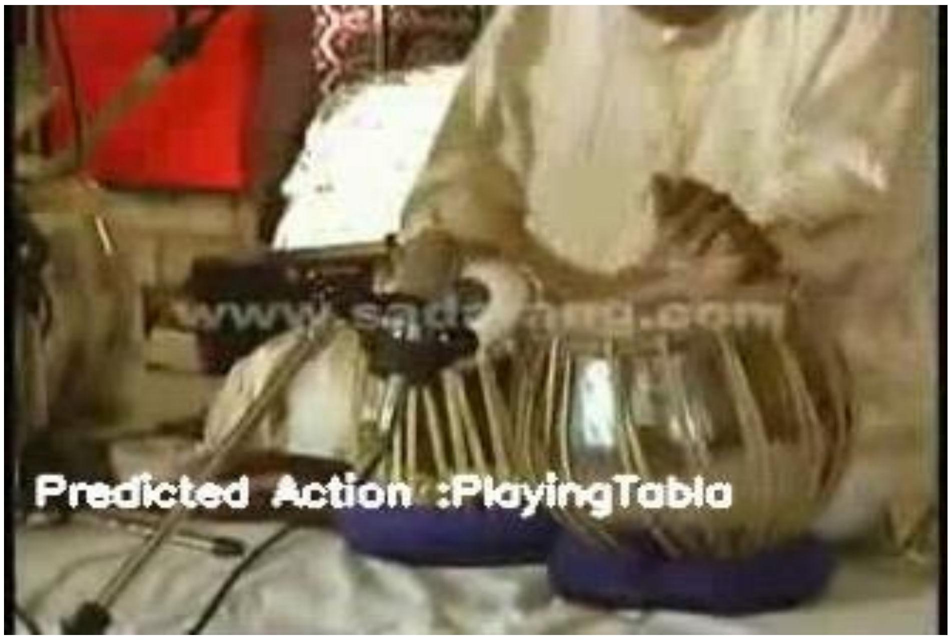


Fig. 6.2.12: Picture showing the predicted action as Playing Tabla

The above picture shows the predicted action of human is Playing Tabla.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, human activity recognition using CNN and LSTM algorithms has shown promising results in recent years. CNNs are effective in extracting spatial features from images, while LSTMs can capture the temporal dynamics of sequential data. By combining these two architectures, we can achieve a more accurate and robust model for human activity recognition.

Testing techniques such as cross-validation, confusion matrix, F1 score, and ROC curve are commonly used to evaluate the performance of the model. These techniques help to identify potential issues with the model and to improve its accuracy and reliability.

The success of human activity recognition using CNN and LSTM algorithms has led to numerous applications in healthcare, sports, and security, among others. It has the potential to enhance the quality of life for individuals and contribute to the development of more efficient and intelligent systems in various domains.

7.2 RECOMMENDATIONS FOR FUTURE RESEARCH

There are several potential areas of future work for human activity recognition using CNN and LSTM algorithms. Some of these include:

- **Improved architectures -** We can explore new CNN and LSTM architectures that can improve the accuracy and efficiency of human activity recognition. This could involve experimenting with deeper networks, attention mechanisms, and other techniques.
- **Multi-modal data fusion -** Human activity recognition can benefit from combining information from different sources, such as audio, video, and sensor data. Future work can explore multi-modal data fusion techniques to enhance the accuracy and robustness of the system.
- **Transfer learning -** Transfer learning is a technique that involves using pre-trained models to improve the performance of a new task. Future work can investigate the use of transfer learning in human activity recognition to reduce

the need for large labelled datasets and improve the generalization performance of the model.

- **Real-time recognition -** Many applications of human activity recognition require real-time processing of data. Future work can focus on developing models that can perform real-time recognition of human activities with low latency and high accuracy.
- **Privacy and security -** As human activity recognition systems become more prevalent, there is a need to address privacy and security concerns. Future work can explore methods to ensure that these systems protect user privacy and prevent malicious attacks.

Overall, there are several exciting directions for future work in human activity recognition using CNN and LSTM algorithms, which can further improve the accuracy, efficiency, and applicability of these systems.

7.3 RESEARCH ISSUES

There are several research issues that need to be addressed in human activity recognition using CNN and LSTM algorithms. Some of these include:

- One of the major challenges in human activity recognition is the lack of large, diverse, and annotated datasets. This can limit the accuracy and generalization performance of the model. Future work can focus on developing new datasets or leveraging transfer learning techniques to address this issue.
- While CNN and LSTM algorithms are effective in recognizing human activities, they often lack interpretability. It can be challenging to understand why a particular activity was recognized or to identify the key features that contribute to the classification decision. Future work can focus on developing methods to improve the interpretability of the model.
- Human activity recognition systems must be designed to work in real-world settings, where there may be variations in lighting, background noise, and other factors. Future work can investigate methods to improve the robustness of the system and ensure that it can operate effectively in real-world
- As human activity recognition systems become more prevalent, there is a need to address ethical considerations, such as privacy, bias, and fairness.

Future work can explore methods to mitigate these concerns and ensure that the systems are developed and deployed responsibly.

Overall, addressing these research issues can improve the accuracy, efficiency, and applicability of human activity recognition using CNN and LSTM algorithms, and ensure that these systems can be deployed effectively and responsibly.

7.4 IMPLEMENTATION ISSUES

There are several implementation issues that should be considered when developing a human activity recognition system using CNN and LSTM algorithms. Some of these include:

- CNN and LSTM algorithms require significant computational resources to train and run the model. Implementation should consider the hardware requirements, including the processing power, memory, and storage, to ensure that the system can handle the workload efficiently.
- The accuracy of the model depends on the quality of the data used for training and testing. Implementation should consider data pre-processing steps, including data cleaning, normalization, and feature extraction, to ensure that the data is in the appropriate format for the model.
- Training the model can be time-consuming and resource-intensive. Implementation should consider strategies to optimize the training process, such as using pre-trained models or parallel processing techniques.
- Once the model is trained, it needs to be deployed in a real-world setting. Implementation should consider the deployment environment, including the hardware and software requirements, and any potential integration issues with other systems.
- Human activity recognition systems require ongoing maintenance and updates to ensure that they remain accurate and reliable. Implementation should consider strategies for monitoring the system performance and implementing updates to the model or the data as necessary.

Overall, addressing these implementation issues can help to ensure that the human activity recognition system using CNN and LSTM algorithms is developed and deployed effectively and can provide accurate and reliable results.

REFERENCES

[1] Usharani, J.; Saktivel, U. Human Activity Recognition using Android Smartphone. In Proceedings of the International Conference on Innovations in Computing & Networking ICICN16, Bengaluru, Karnataka, 2016.

[2] Anguita, D.; Ghio, A.; Oneto, L.; Parra-Llanas, X.; Reyes-Ortiz, J. Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic. J. Univers. Comput. Sci. 2013, 19, 1295–1314.

[3] Mary Posonia A, S. Vigneshwari, Albert Mayan J, Jamunarani D, "Service Direct: Platform that Incorporates Service Providers and Consumers Directly", International Journal of Engineering and Advanced Technology (IJEAT), Vol.8, No.6, pp. 3301-3304,2019.

[4] Uddin, Z.; Soylu, A. Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning. Sci. Rep. 2021, 11, 16455.

[5] Vakili, M.; Rezaei, M. Incremental Learning Techniques for Online Human Activity Recognition. arXiv 2021, arXiv:2109.09435.

[6] Muangprathub, J.; Sriwichian, A.; Wanichsombat, A.; Kajornkasirat, S.; Nillaor, P.; Boonjing, V. A Novel Elderly Tracking System Using Machine Learning to Classify Signals from Mobile and Wearable Sensors. Int. J. Environ. Res. Public Health 2021, 18, 12652.

[7] Joshila Grace L.K, Vigneshwari S, SathyaBama Krishna R, Ankayarkanni B, Mary Posonia A, "A Joint Optimization Approach for Security and Insurance Management on the Cloud", Lecture Notes in Networks and Systems, Vol. 430, pp. 405–413.

[8] Zhou, B.; Yang, J.; Li, Q. Smartphone-Based Activity Recognition for Indoor Localization Using a Convolutional Neural Network. Sensors 2019, 19, 621.

[9] Murad, A.; Pyun, J.-Y. Deep Recurrent Neural Networks for Human Activity Recognition. Sensors 2017, 17, 2556.

[10] S. O. Eyobu and D. S. Han, “Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network,” Sensors, 2018, 18, 2892.

[11] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst and M. Hompel, “Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors,” Informatics, 5(2), 26, May 2018.

[12] K. Kim, S. Choi, M. Chae, H. Park, J. Lee, and J. Park, “A Deep Learning Based Approach to Recognizing Accompanying Status of Smartphone Users Using Multimodal Data,” Journal of Intelligence and Information Systems, vol. 25, no. 1, pp. 163–177, Mar. 2019.

[13] Mary Posonia A, Ankayarkanni B, Usha Nandhini D, Albert Mayan J, Nagarajan G (2022), "An Efficient Algorithm for Traffic Congestion Control", Advances in Intelligent Computing and Communication, Lecture Notes in Networks and Systems, Vol 430, Springer.

[14] F. Chollet, “Layer wrappers,” Keras Documentation, 2015, Online, Available: https://keras.io/layers/wrappers/#timedistributed, Accessed: Dec. 1, 2019.

[15] S. Hochreiter and J. Schmidhuber. “Long short-term memory,” Neural computation, 9(8):1735–1780, 1997.

[16] Wang, J., Zhang, X., Wu, Y., & Wang, Y. (2022). Unsupervised Learning of Human Activities from Long-Term Videos. IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2022.3182538.

[17] Yao, J., Zhang, L., Lu, J., & Xu, Y. (2021). Self-Supervised Learning of Human Activities from Temporal Segments in Videos. IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2021.3111372.

APPENDIX

A. SOURCE CODE

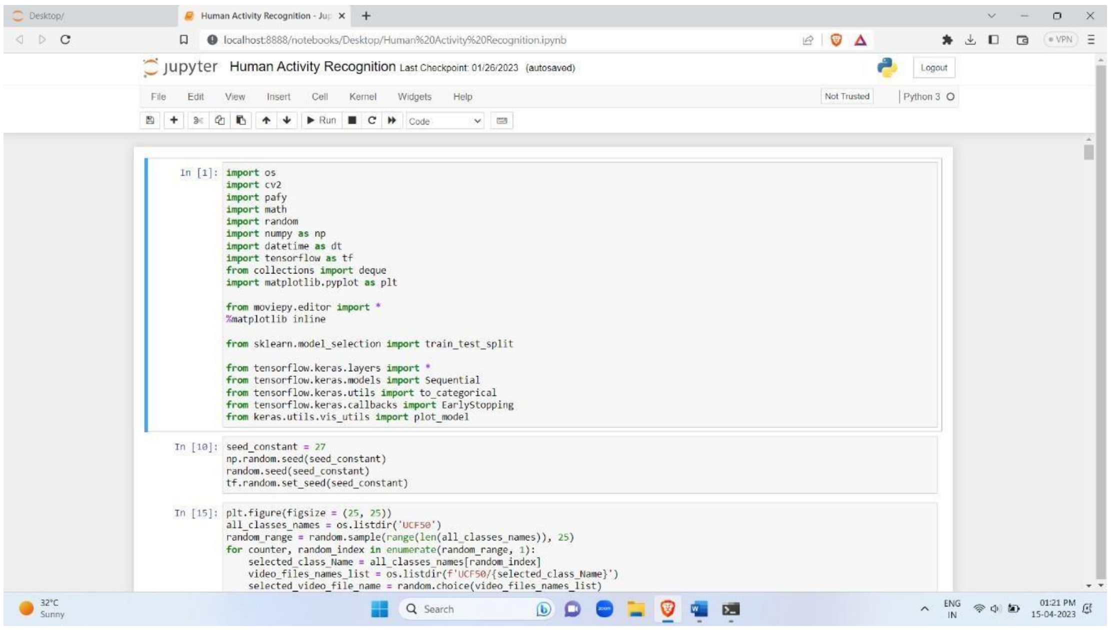
```
import os
import cv2
import pafy
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt
from moviepy.editor import *
%matplotlib inline
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from keras.utils.vis_utils import plot_model
seed_constant =  27
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed(seed_constant)
plt.figure(figsize = (25, 25))
all_classes_names = os.listdir('UCF50')
random_range = random.sample(range(len(all_classes_names)), 25)
for counter, random_index in enumerate(random_range, 1):
    selected_class_Name = all_classes_names[random_index]
    video_files_names_list = os.listdir(f'UCF50/{selected_class_Name}')
    selected_video_file_name = random.choice(video_files_names_list)
    video_reader =
cv2.VideoCapture(f'UCF50/{selected_class_Name}/{selected_video_file_name
}')
    _, bgr_frame = video_reader.read()
    video_reader.release()
    rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)
    plt.subplot(5, 5, counter);plt.imshow(rgb_frame);plt.axis('off')
IMAGE_HEIGHT , IMAGE_WIDTH = 64, 64
SEQUENCE_LENGTH = 20
DATASET_DIR = "UCF50"
CLASSES_LIST = ["WalkingWithDog", "TaiChi", "Swing", "HorseRace"]
def frames_extraction(video_path):
    frames_list = []
    video_reader = cv2.VideoCapture(video_path)
    video_frames_count =
int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    skip_frames_window =
max(int(video_frames_count/SEQUENCE_LENGTH), 1)
    for frame_counter in range(SEQUENCE_LENGTH):
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter *
skip_frames_window)
        success, frame = video_reader.read()
        if not success:
            break
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255
        frames_list.append(normalized_frame)
    video_reader.release()
    return frames_list
def create_dataset():
    features =  []
    labels = []
    video_files_paths = []
    for class_index, class_name in enumerate(CLASSES_LIST):
        print(f'Extracting Data of Class: {class_name}')
        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))
        for file_name in files_list:
            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)
            frames = frames_extraction(video_file_path)
            if len(frames) == SEQUENCE_LENGTH:
                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)
    features = np.asarray(features)
    labels = np.array(labels)
    return features, labels, video_files_paths
features, labels, video_files_paths = create_dataset()
one_hot_encoded_labels = to_categorical(labels)
features_train, features_test, labels_train, labels_test =
train_test_split(features, one_hot_encoded_labels, test_size = 0.25, shuffle =
True, random_state = seed_constant)
def create_convlstm_model():
    model = Sequential()
    model.add(ConvLSTM2D(filters = 4, kernel_size = (3, 3), activation =
'tanh',data_format = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True, input_shape =
(SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',
data_format='channels_last'))
    model.add(TimeDistributed(Dropout(0.2)))
    model.add(ConvLSTM2D(filters = 8, kernel_size = (3, 3), activation = 'tanh',
data_format = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',
data_format='channels_last'))
    model.add(TimeDistributed(Dropout(0.2)))
```

```
   model.add(ConvLSTM2D(filters = 14, kernel_size = (3, 3), activation = 'tanh',
data_format = "channels_last",
                   recurrent_dropout=0.2, return_sequences=True))

   model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',
data_format='channels_last'))
   model.add(TimeDistributed(Dropout(0.2)))

   model.add(ConvLSTM2D(filters = 16, kernel_size = (3, 3), activation = 'tanh',
data_format = "channels_last",
                   recurrent_dropout=0.2, return_sequences=True))

   model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same',
data_format='channels_last'))

   model.add(Flatten())

   model.add(Dense(len(CLASSES_LIST), activation = "softmax"))



   model.summary()

   return model
convlstm_model = create_convlstm_model()

print("Model Created Successfully!")
plot_model(convlstm_model, to_file = 'convlstm_model_structure_plot.png',
show_shapes = True, show_layer_names = True)
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 10,
mode = 'min', restore_best_weights = True)
convlstm_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam',
metrics = ["accuracy"])
```

```
convlstm_model_training_history = convlstm_model.fit(x = features_train, y =
labels_train, epochs = 50, batch_size = 4,shuffle = True, validation_split = 0.2,
callbacks = [early_stopping_callback])
model_evaluation_history = convlstm_model.evaluate(features_test,
labels_test)
model_evaluation_loss, model_evaluation_accuracy =
model_evaluation_history
date_time_format = '%Y_%m_%d__%H_%M_%S'
current_date_time_dt = dt.datetime.now()
current_date_time_string = dt.datetime.strftime(current_date_time_dt,
date_time_format)
model_file_name =
f'convlstm_model___Date_Time_{current_date_time_string}___Loss_{model_e
valuation_loss}___Accuracy_{model_evaluation_accuracy}.h5'
convlstm_model.save(model_file_name)
def plot_metric(model_training_history, metric_name_1, metric_name_2,
x_label, y_label):
    metric_value_1 = model_training_history.history[metric_name_1]
    metric_value_2 = model_training_history.history[metric_name_2]
    epochs = range(len(metric_value_1))
    plt.plot(epochs, metric_value_1, 'green', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'orangered', label = metric_name_2)
    plt.xlabel(str(x_label))
    plt.ylabel(str(y_label))
    plt.legend()
plot_metric(convlstm_model_training_history, 'loss', 'val_loss', 'No.of epochs',
'Losses')
plot_metric(convlstm_model_training_history, 'accuracy', 'val_accuracy', 'No.of
epochs', 'Accuracies')
def create_LRCN_model():
    model = Sequential()
    model.add(TimeDistributed(Conv2D(16, (3, 3), padding='same',activation =
'relu'),
                              input_shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT,
IMAGE_WIDTH, 3)))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.25)))

    model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same',activation =
'relu')))
    model.add(TimeDistributed(MaxPooling2D((4, 4))))
    model.add(TimeDistributed(Dropout(0.25)))
    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation =
'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2))))
    model.add(TimeDistributed(Dropout(0.25)))
    model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation =
'relu')))
    model.add(TimeDistributed(MaxPooling2D((2, 2)))
    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(32))
    model.add(Dense(len(CLASSES_LIST), activation = 'softmax'))
    model.summary()
    return model
LRCN_model = create_LRCN_model()
print("Model Created Successfully!")
plot_model(LRCN_model, to_file = 'LRCN_model_structure_plot.png',
show_shapes = True, show_layer_names = True)
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 15,
mode = 'min', restore_best_weights = True)
LRCN_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam',
metrics = ["accuracy"])
```

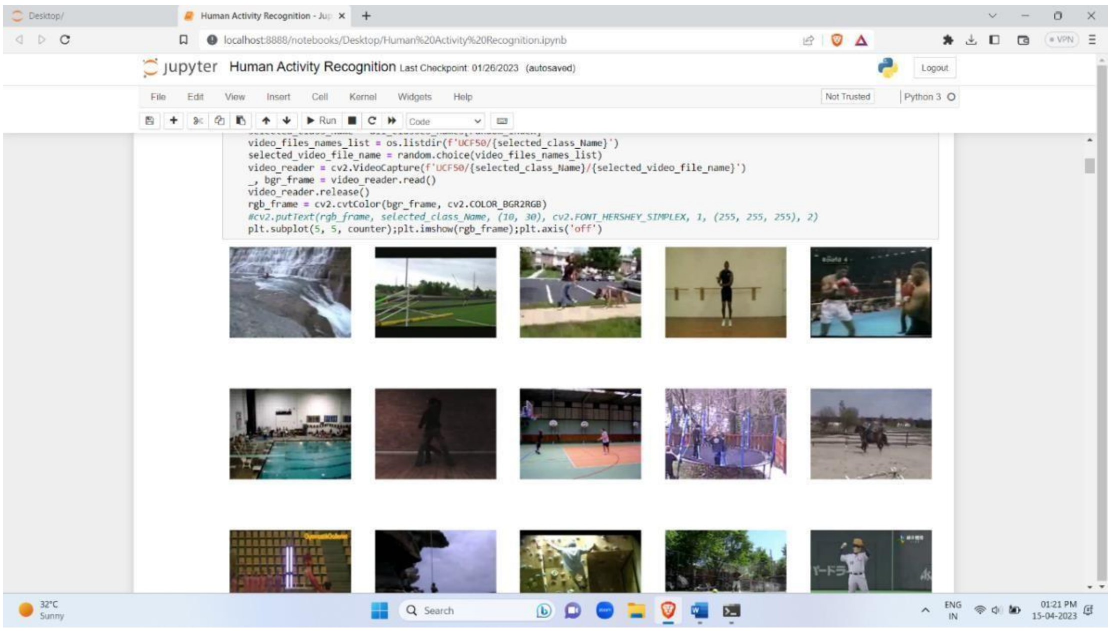
```
LRCN_model_training_history = LRCN_model.fit(x = features_train, y =
labels_train, epochs = 70, batch_size = 4 , shuffle = True, validation_split = 0.2,
callbacks = [early_stopping_callback])
model_evaluation_history = LRCN_model.evaluate(features_test, labels_test)
model_evaluation_loss, model_evaluation_accuracy =
model_evaluation_history
date_time_format = '%Y_%m_%d__%H_%M_%S'
current_date_time_dt = dt.datetime.now()
current_date_time_string = dt.datetime.strftime(current_date_time_dt,
date_time_format)
model_file_name =
f'LRCN_model___Date_Time_{current_date_time_string}___Loss_{model_eval
uation_loss}___Accuracy_{model_evaluation_accuracy}.h5'
LRCN_model.save(model_file_name)
plot_metric(LRCN_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total
Validation Loss','No.of epochs')
plot_metric(LRCN_model_training_history, 'accuracy', 'val_accuracy', 'Total
Accuracy vs Total Validation Accuracy','No.of epochs')
def download_youtube_videos(youtube_video_url, output_directory):
    video = pafy.new(youtube_video_url)
    title = video.title
    video_best = video.getbest()
    output_file_path = f'{output_directory}/{title}.mp4'
    video_best.download(filepath = output_file_path, quiet = True)
    return title
test_videos_directory = 'test_videos'
os.makedirs(test_videos_directory, exist_ok = True)
video_title =
download_youtube_videos('https://www.youtube.com/watch?v=8u0qjmHIOcE',
test_videos_directory)
input_video_file_path = f'{test_videos_directory}/{video_title}.mp4'
def predict_on_video(video_file_path, output_file_path,
SEQUENCE_LENGTH):
    video_reader = cv2.VideoCapture(video_file_path)
    original_video_width =
int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height =
int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
    video_writer = cv2.VideoWriter(output_file_path, cv2.VideoWriter_fourcc('M',
'P', '4', 'V'),
                                   video_reader.get(cv2.CAP_PROP_FPS),
(original_video_width, original_video_height))
    frames_queue = deque(maxlen = SEQUENCE_LENGTH)
    predicted_class_name = ''
    while video_reader.isOpened():
        ok, frame = video_reader.read()
        if not ok:
            break
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255
        frames_queue.append(normalized_frame)
        if len(frames_queue) == SEQUENCE_LENGTH:
            predicted_labels_probabilities =
LRCN_model.predict(np.expand_dims(frames_queue, axis = 0))[0]
            predicted_label = np.argmax(predicted_labels_probabilities)
            predicted_class_name = CLASSES_LIST[predicted_label]
        cv2.putText(frame, predicted_class_name, (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        video_writer.write(frame)
    video_reader.release()
    video_writer.release()
output_video_file_path = f'{test_videos_directory}/{video_title}-Output-
SeqLen{SEQUENCE_LENGTH}.mp4'
predict_on_video(input_video_file_path, output_video_file_path,
SEQUENCE_LENGTH)
VideoFileClip(output_video_file_path, audio=False,
target_resolution=(300,None)).ipython_display()
```

B. SCREENSHOTS



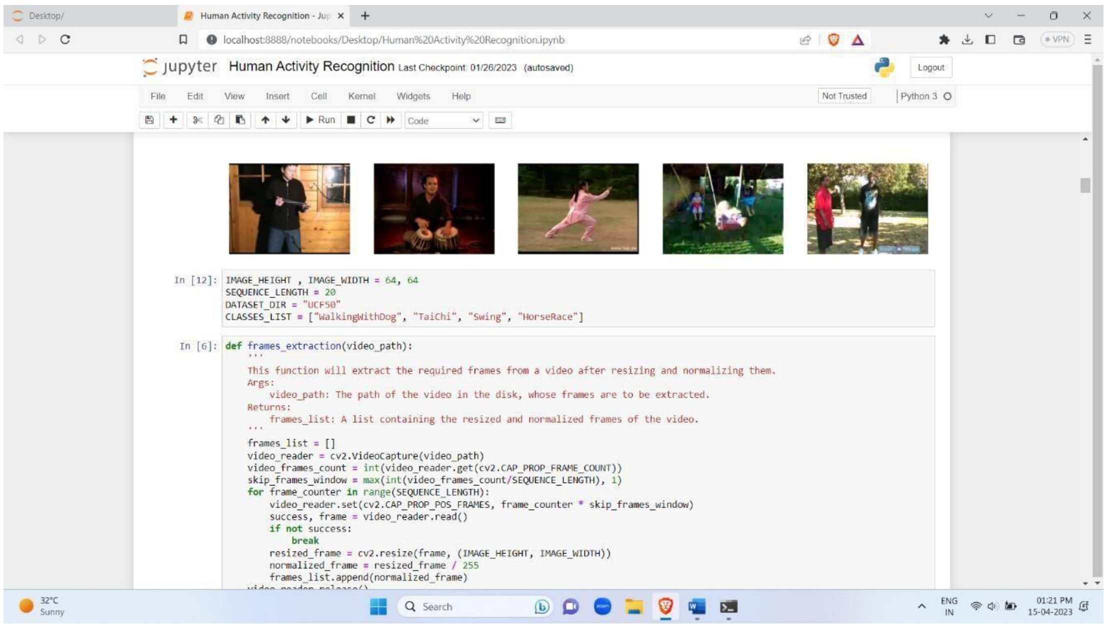
```
In [1]: import os
        import cv2
        import pafy
        import math
        import random
        import numpy as np
        import datetime as dt
        import tensorflow as tf
        from collections import deque
        import matplotlib.pyplot as plt

        from moviepy.editor import *
        %matplotlib inline

        from sklearn.model_selection import train_test_split

        from tensorflow.keras.layers import *
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.utils import to_categorical
        from tensorflow.keras.callbacks import EarlyStopping
        from keras.utils.vis_utils import plot_model

In [10]: seed_constant = 27
         np.random.seed(seed_constant)
         random.seed(seed_constant)
         tf.random.set_seed(seed_constant)

In [15]: plt.figure(figsize = (25, 25))
         all_classes_names = os.listdir('UCF50')
         random_range = random.sample(range(len(all_classes_names)), 25)
         for counter, random_index in enumerate(random_range, 1):
             selected_class_Name = all_classes_names[random_index]
             video_files_names_list = os.listdir(f'UCF50/{selected_class_Name}')
             selected_video_file_name = random.choice(video_files_names_list)
```

Fig. B.1: Importing Required Libraries



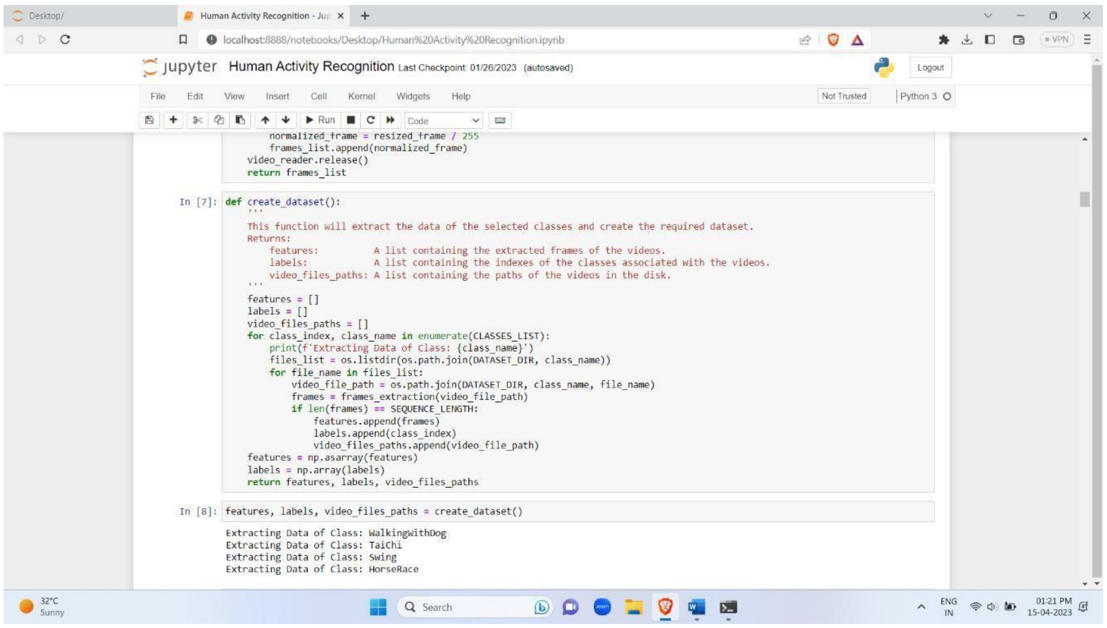
```
    video_files_names_list = os.listdir(f'UCF50/{selected_class_Name}')
    selected_video_file_name = random.choice(video_files_names_list)
    video_reader = cv2.VideoCapture(f'UCF50/{selected_class_Name}/{selected_video_file_name}')
    _, bgr_frame = video_reader.read()
    video_reader.release()
    rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)
    #cv2.putText(rgb_frame, selected_class_Name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    plt.subplot(5, 5, counter);plt.imshow(rgb_frame);plt.axis('off')
```

Fig. B.2: Pre-processing the data



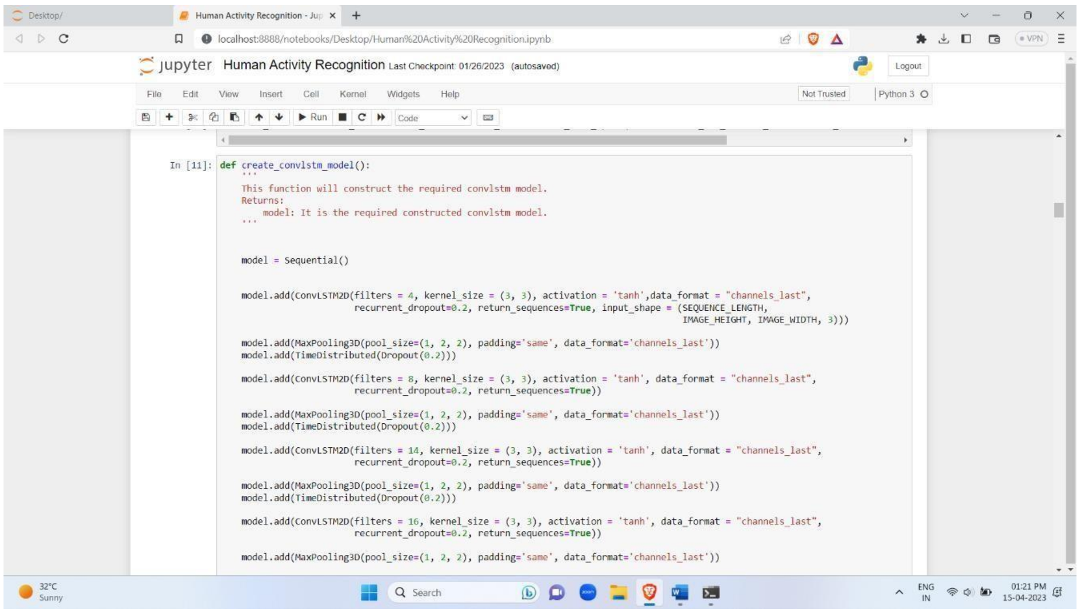
```
In [12]: IMAGE_HEIGHT , IMAGE_WIDTH = 64, 64
         SEQUENCE_LENGTH = 20
         DATASET_DIR = "UCF50"
         CLASSES_LIST = ["WalkingWithDog", "TaiChi", "Swing", "HorseRace"]

In [6]:  def frames_extraction(video_path):
             '''
             This function will extract the required frames from a video after resizing and normalizing them.
             Args:
                 video_path: The path of the video in the disk, whose frames are to be extracted.
             Returns:
                 frames_list: A list containing the resized and normalized frames of the video.
             '''
             frames_list = []
             video_reader = cv2.VideoCapture(video_path)
             video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
             skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)
             for frame_counter in range(SEQUENCE_LENGTH):
                 video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)
                 success, frame = video_reader.read()
                 if not success:
                     break
                 resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
                 normalized_frame = resized_frame / 255
                 frames_list.append(normalized_frame)
```

Fig. B.3: Extracting the Frames



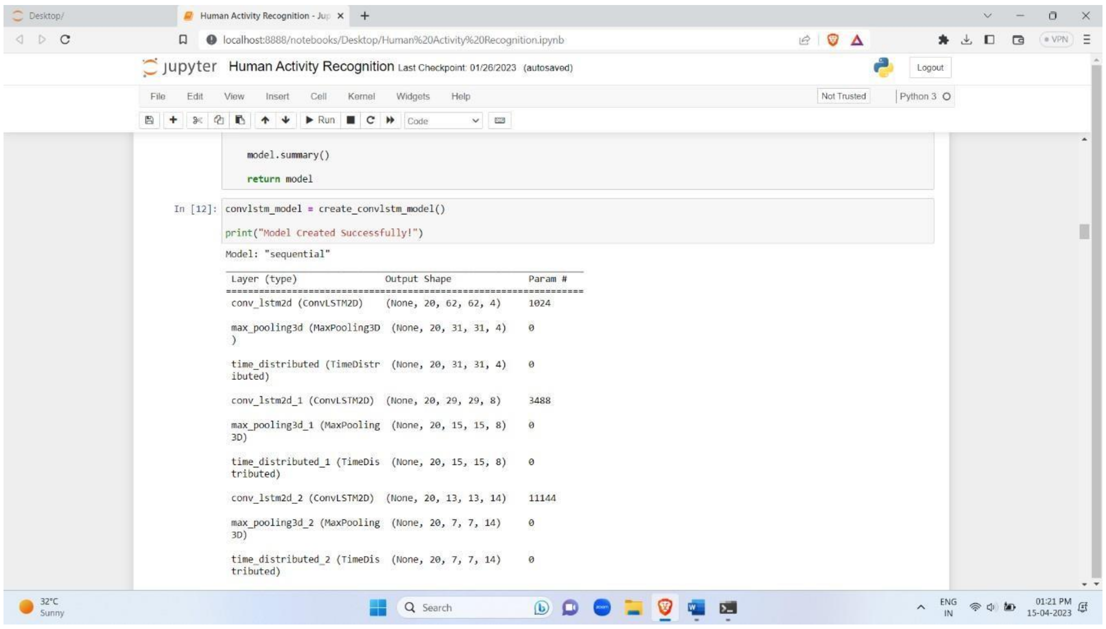
```
                 normalized_frame = resized_frame / 255
                 frames_list.append(normalized_frame)
             video_reader.release()
             return frames_list

In [7]:  def create_dataset():
             '''
             This function will extract the data of the selected classes and create the required dataset.
             Returns:
                 features:         A list containing the extracted frames of the videos.
                 labels:           A list containing the indexes of the classes associated with the videos.
                 video_files_paths: A list containing the paths of the videos in the disk.
             '''
             features = []
             labels = []
             video_files_paths = []
             for class_index, class_name in enumerate(CLASSES_LIST):
                 print(f'Extracting Data of Class: {class_name}')
                 files_list = os.listdir(os.path.join(DATASET_DIR, class_name))
                 for file_name in files_list:
                     video_file_path = os.path.join(DATASET_DIR, class_name, file_name)
                     frames = frames_extraction(video_file_path)
                     if len(frames) == SEQUENCE_LENGTH:
                         features.append(frames)
                         labels.append(class_index)
                         video_files_paths.append(video_file_path)
             features = np.asarray(features)
             labels = np.array(labels)
             return features, labels, video_files_paths

In [8]:  features, labels, video_files_paths = create_dataset()

         Extracting Data of Class: WalkingWithDog
         Extracting Data of Class: TaiChi
         Extracting Data of Class: Swing
         Extracting Data of Class: HorseRace
```

Fig. B.4: Creating dataset with required frames



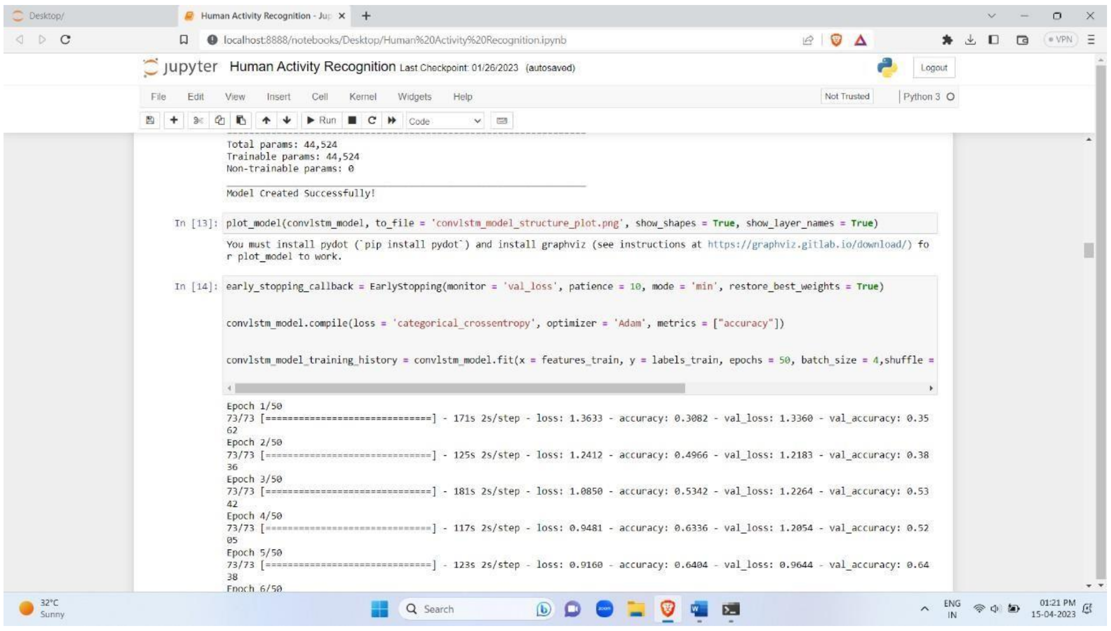
```
In [11]: def create_convlstm_model():
             '''
             This function will construct the required convlstm model.
             Returns:
                 model: It is the required constructed convlstm model.
             '''


             model = Sequential()


             model.add(ConvLSTM2D(filters = 4, kernel_size = (3, 3), activation = 'tanh',data_format = "channels_last",
                                  recurrent_dropout=0.2, return_sequences=True, input_shape = (SEQUENCE_LENGTH,
                                                                                               IMAGE_HEIGHT, IMAGE_WIDTH, 3)))

             model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
             model.add(TimeDistributed(Dropout(0.2)))

             model.add(ConvLSTM2D(filters = 8, kernel_size = (3, 3), activation = 'tanh', data_format = "channels_last",
                                  recurrent_dropout=0.2, return_sequences=True))

             model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
             model.add(TimeDistributed(Dropout(0.2)))

             model.add(ConvLSTM2D(filters = 14, kernel_size = (3, 3), activation = 'tanh', data_format = "channels_last",
                                  recurrent_dropout=0.2, return_sequences=True))

             model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
             model.add(TimeDistributed(Dropout(0.2)))

             model.add(ConvLSTM2D(filters = 16, kernel_size = (3, 3), activation = 'tanh', data_format = "channels_last",
                                  recurrent_dropout=0.2, return_sequences=True))

             model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
```

Fig. B.5: Creating ConvLSTM Model



```
             model.summary()

             return model

In [12]: convlstm_model = create_convlstm_model()

         print("Model Created Successfully!")

         Model: "sequential"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          conv_lstm2d (ConvLSTM2D)    (None, 20, 62, 62, 4)     1024

          max_pooling3d (MaxPooling3D  (None, 20, 31, 31, 4)    0
          )

          time_distributed (TimeDistr  (None, 20, 31, 31, 4)    0
          ibuted)

          conv_lstm2d_1 (ConvLSTM2D)  (None, 20, 29, 29, 8)     3488

          max_pooling3d_1 (MaxPooling  (None, 20, 15, 15, 8)    0
          3D)

          time_distributed_1 (TimeDis  (None, 20, 15, 15, 8)    0
          tributed)

          conv_lstm2d_2 (ConvLSTM2D)  (None, 20, 13, 13, 14)    11144

          max_pooling3d_2 (MaxPooling  (None, 20, 7, 7, 14)     0
          3D)

          time_distributed_2 (TimeDis  (None, 20, 7, 7, 14)     0
          tributed)
```

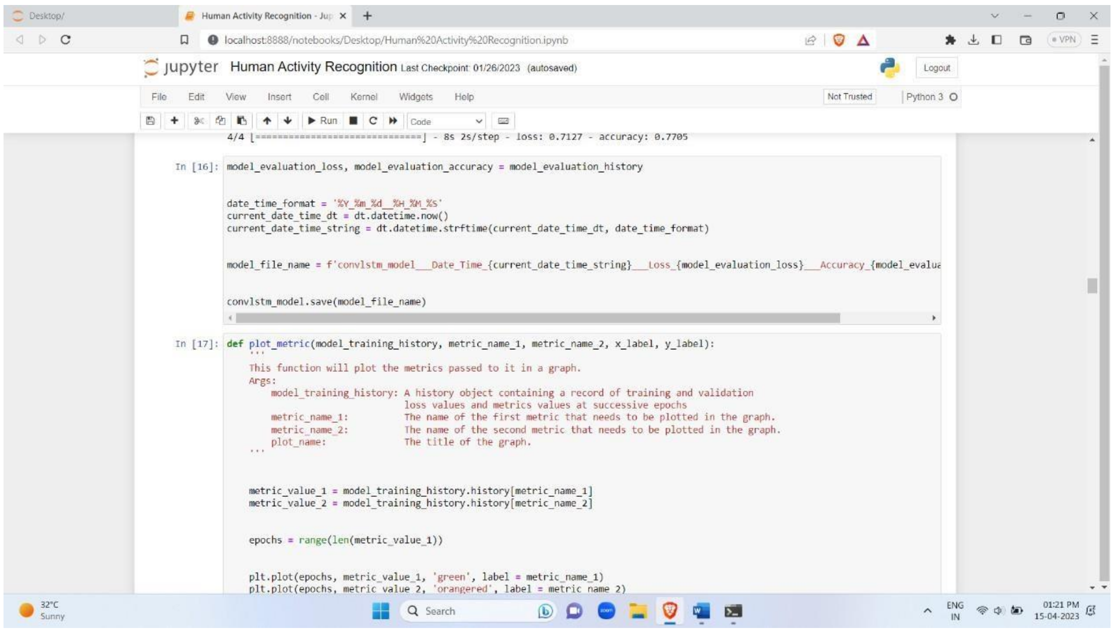
Fig. B.6: Successfully created ConvLSTM Model



localhost:8888/notebooks/Desktop/Human%20Activity%20Recognition.ipynb
jupyter Human Activity Recognition Last Checkpoint: 01/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
Total params: 44,524
Trainable params: 44,524
Non-trainable params: 0

Model Created Successfully!

In [13]: plot_model(convlstm_model, to_file = 'convlstm_model_structure_plot.png', show_shapes = True, show_layer_names = True)

You must install pydot (`pip install pydot`) and install graphviz (see instructions at https://graphviz.gitlab.io/download/) fo
r plot_model to work.

In [14]: early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 10, mode = 'min', restore_best_weights = True)


convlstm_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])


convlstm_model_training_history = convlstm_model.fit(x = features_train, y = labels_train, epochs = 50, batch_size = 4,shuffle =

Epoch 1/50
73/73 [==============================] - 171s 2s/step - loss: 1.3633 - accuracy: 0.3082 - val_loss: 1.3360 - val_accuracy: 0.35
62
Epoch 2/50
73/73 [==============================] - 125s 2s/step - loss: 1.2412 - accuracy: 0.4966 - val_loss: 1.2183 - val_accuracy: 0.38
36
Epoch 3/50
73/73 [==============================] - 181s 2s/step - loss: 1.0850 - accuracy: 0.5342 - val_loss: 1.2264 - val_accuracy: 0.53
42
Epoch 4/50
73/73 [==============================] - 117s 2s/step - loss: 0.9481 - accuracy: 0.6336 - val_loss: 1.2054 - val_accuracy: 0.52
05
Epoch 5/50
73/73 [==============================] - 123s 2s/step - loss: 0.9160 - accuracy: 0.6404 - val_loss: 0.9644 - val_accuracy: 0.64
38
Epoch 6/50
```

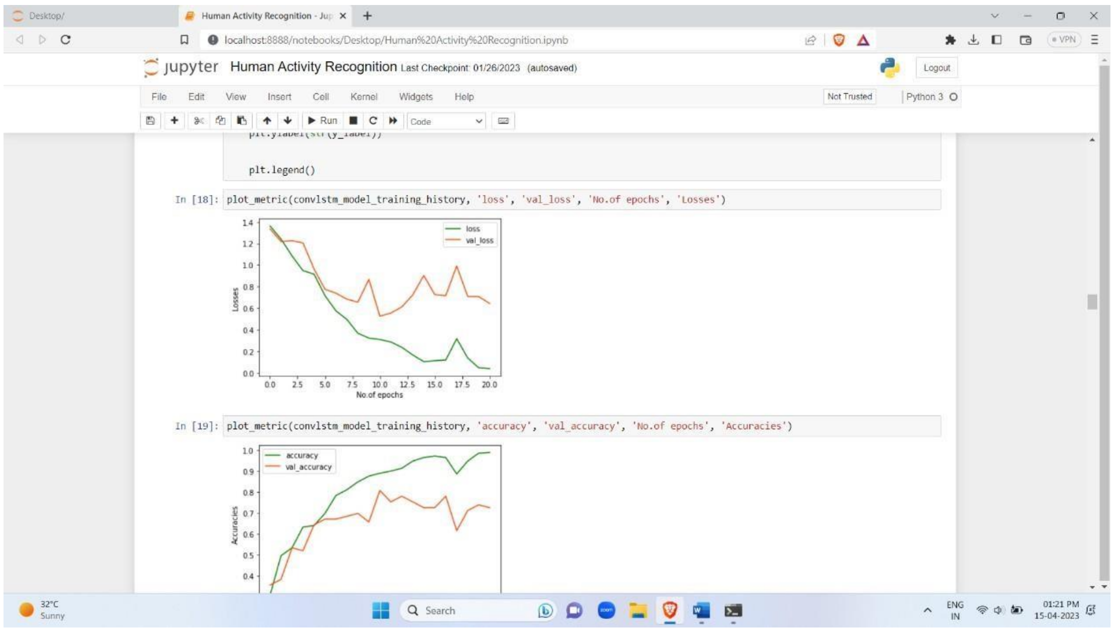
Fig. B.7: Plotting the Model



localhost:8888/notebooks/Desktop/Human%20Activity%20Recognition.ipynb
jupyter Human Activity Recognition Last Checkpoint: 01/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
4/4 [==============================] - 8s 2s/step - loss: 0.7127 - accuracy: 0.7705

In [16]: model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history


date_time_format = '%Y_%m_%d__%H_%M_%S'
current_date_time_dt = dt.datetime.now()
current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_format)


model_file_name = f'convlstm_model___Date_Time_{current_date_time_string}___Loss_{model_evaluation_loss}___Accuracy_{model_evalua


convlstm_model.save(model_file_name)

In [17]: def plot_metric(model_training_history, metric_name_1, metric_name_2, x_label, y_label):
    '''
    This function will plot the metrics passed to it in a graph.
    Args:
        model_training_history: A history object containing a record of training and validation
                                loss values and metrics values at successive epochs
        metric_name_1:          The name of the first metric that needs to be plotted in the graph.
        metric_name_2:          The name of the second metric that needs to be plotted in the graph.
        plot_name:              The title of the graph.
    '''


    metric_value_1 = model_training_history.history[metric_name_1]
    metric_value_2 = model_training_history.history[metric_name_2]


    epochs = range(len(metric_value_1))


    plt.plot(epochs, metric_value_1, 'green', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'orangered', label = metric_name_2)
```

Fig. B.8: Evaluating the Model


```
    plt.legend()

In [18]: plot_metric(convlstm_model_training_history, 'loss', 'val_loss', 'No.of epochs', 'Losses')
```

```
In [19]: plot_metric(convlstm_model_training_history, 'accuracy', 'val_accuracy', 'No.of epochs', 'Accuracies')
```

Fig. B.9: Plotting Loss and Accuracy Curves of ConvLSTM Model

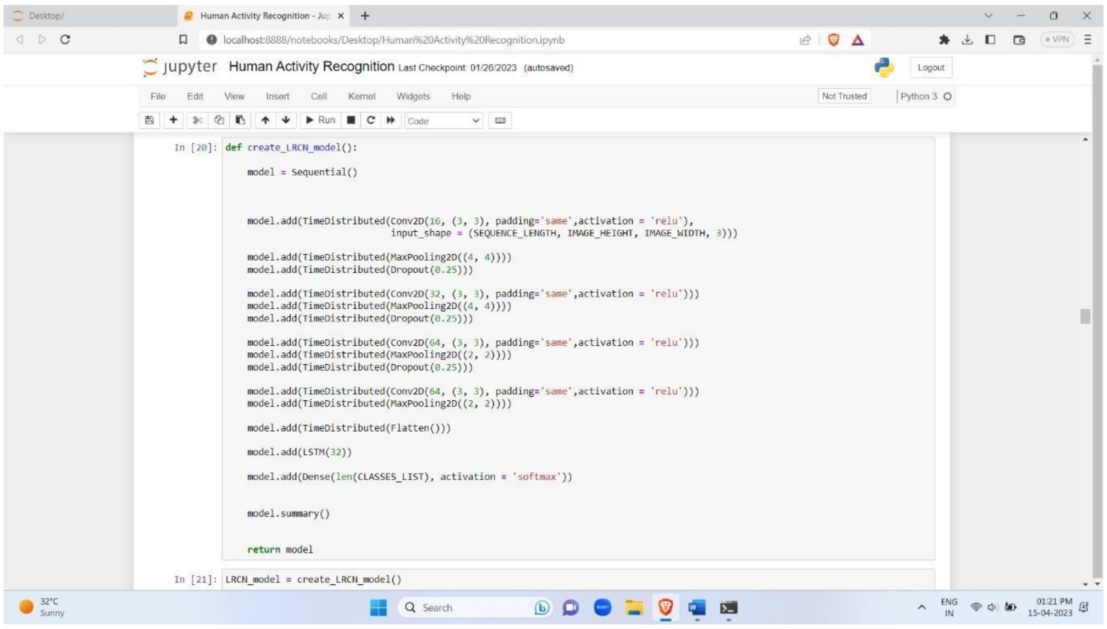
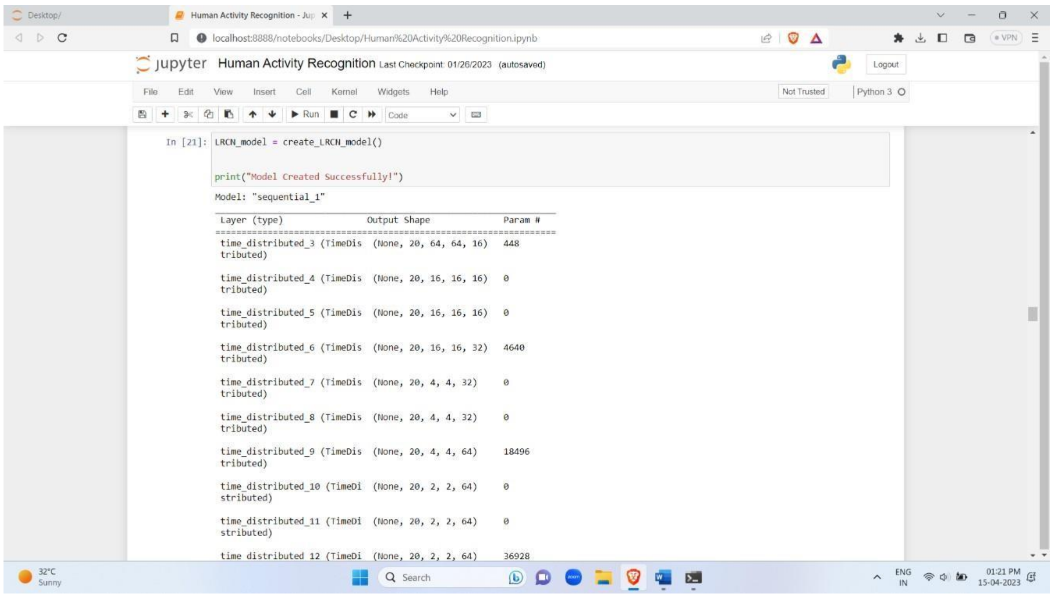
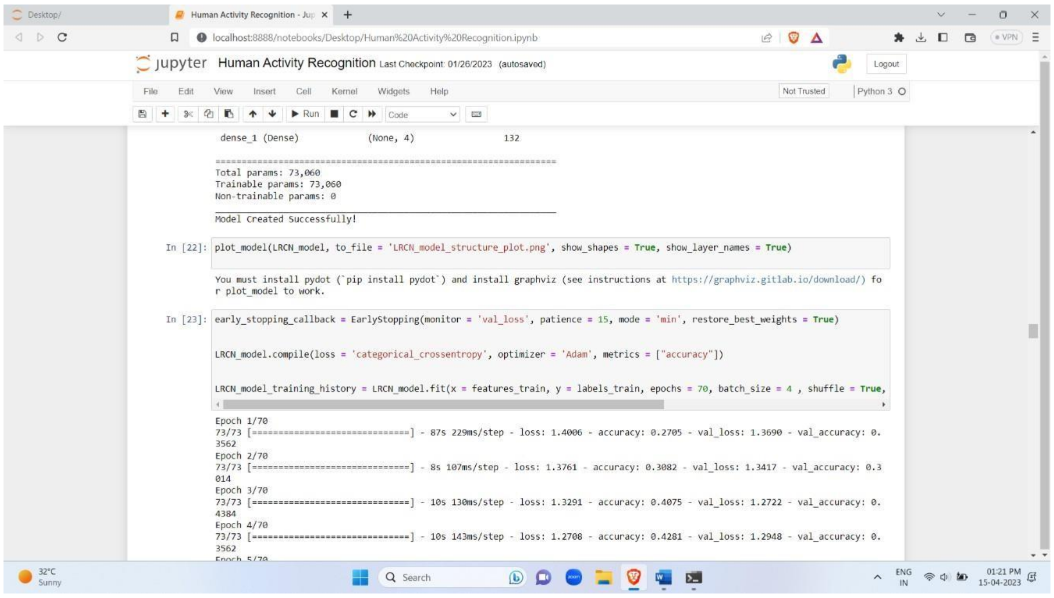

```
In [20]: def create_LRCN_model():

             model = Sequential()


             model.add(TimeDistributed(Conv2D(16, (3, 3), padding='same',activation = 'relu'),
                                       input_shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))

             model.add(TimeDistributed(MaxPooling2D((4, 4))))
             model.add(TimeDistributed(Dropout(0.25)))

             model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same',activation = 'relu')))
             model.add(TimeDistributed(MaxPooling2D((4, 4))))
             model.add(TimeDistributed(Dropout(0.25)))

             model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
             model.add(TimeDistributed(MaxPooling2D((2, 2))))
             model.add(TimeDistributed(Dropout(0.25)))

             model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same',activation = 'relu')))
             model.add(TimeDistributed(MaxPooling2D((2, 2))))

             model.add(TimeDistributed(Flatten()))

             model.add(LSTM(32))

             model.add(Dense(len(CLASSES_LIST), activation = 'softmax'))


             model.summary()


             return model

In [21]: LRCN_model = create_LRCN_model()
```

Fig. B.10: Creating LRCN Model



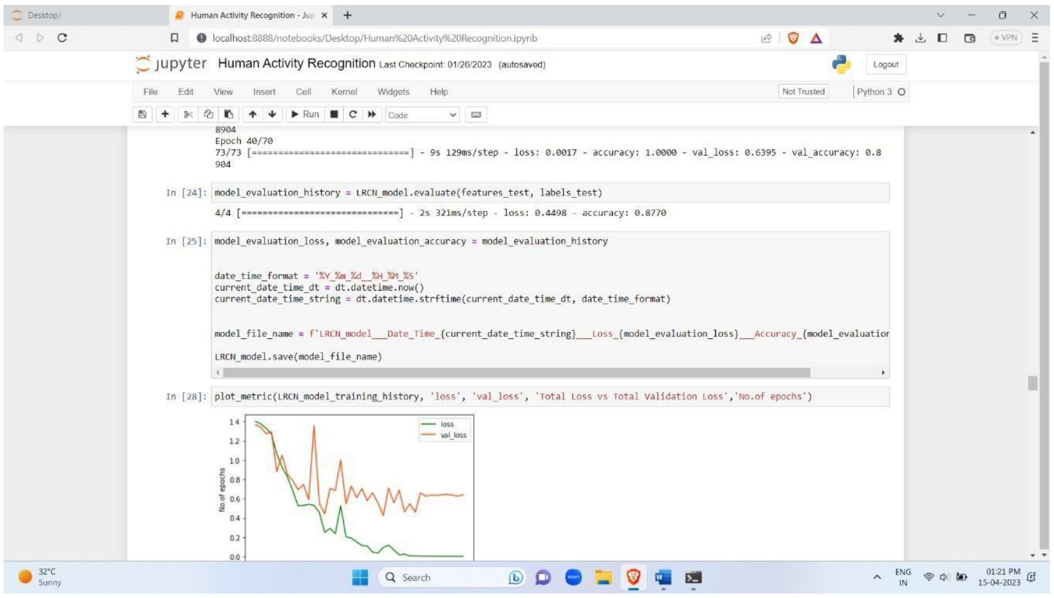
```
In [21]: LRCN_model = create_LRCN_model()


         print("Model Created Successfully!")

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_3 (TimeDis  (None, 20, 64, 64, 16)   448
 tributed)

 time_distributed_4 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_5 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 16, 16, 32)   4640
 tributed)

 time_distributed_7 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_9 (TimeDis  (None, 20, 4, 4, 64)     18496
 tributed)

 time_distributed_10 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time_distributed_11 (TimeDi  (None, 20, 2, 2, 64)     0
 stributed)

 time distributed 12 (TimeDi  (None, 20, 2, 2, 64)     36928
```

Fig. B.11: LRCN Model Created Successfully



```
 dense_1 (Dense)             (None, 4)                 132

=================================================================
Total params: 73,060
Trainable params: 73,060
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!

In [22]: plot_model(LRCN_model, to_file = 'LRCN_model_structure_plot.png', show_shapes = True, show_layer_names = True)

You must install pydot (`pip install pydot`) and install graphviz (see instructions at https://graphviz.gitlab.io/download/) fo
r plot_model to work.

In [23]: early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 15, mode = 'min', restore_best_weights = True)


         LRCN_model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])


         LRCN_model_training_history = LRCN_model.fit(x = features_train, y = labels_train, epochs = 70, batch_size = 4 , shuffle = True,

Epoch 1/70
73/73 [==============================] - 87s 229ms/step - loss: 1.4006 - accuracy: 0.2705 - val_loss: 1.3690 - val_accuracy: 0.
3562
Epoch 2/70
73/73 [==============================] - 8s 107ms/step - loss: 1.3761 - accuracy: 0.3082 - val_loss: 1.3417 - val_accuracy: 0.3
014
Epoch 3/70
73/73 [==============================] - 10s 130ms/step - loss: 1.3291 - accuracy: 0.4075 - val_loss: 1.2722 - val_accuracy: 0.
4384
Epoch 4/70
73/73 [==============================] - 10s 143ms/step - loss: 1.2708 - accuracy: 0.4281 - val_loss: 1.2948 - val_accuracy: 0.
3562
```

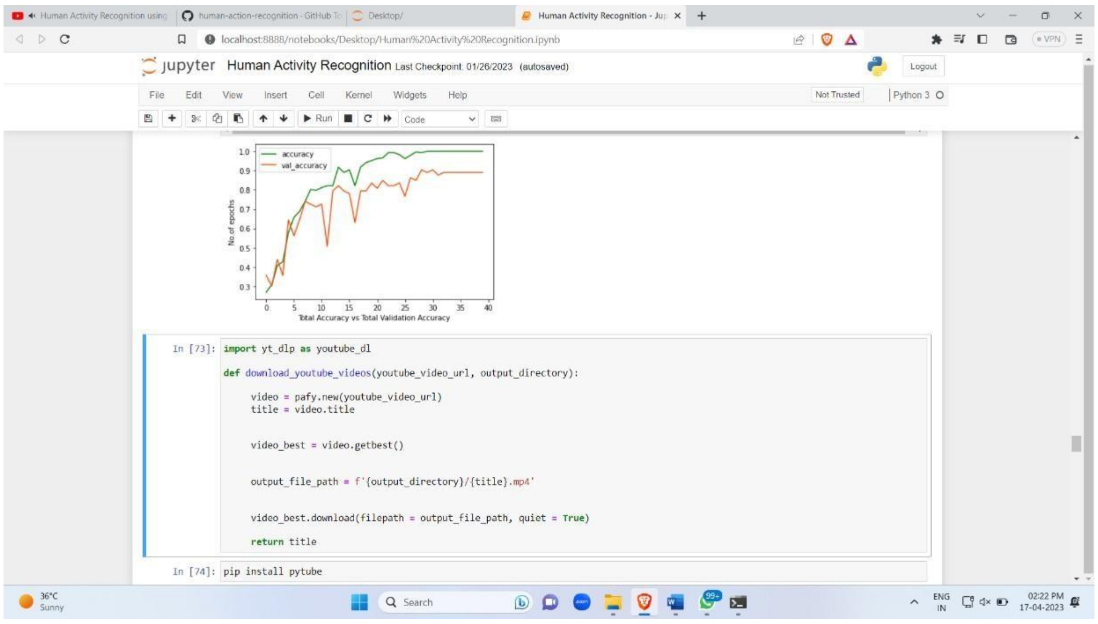
Fig. B.12: Plotting the Model



```
8904
Epoch 40/70
73/73 [==============================] - 9s 129ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.6395 - val_accuracy: 0.8
904

In [24]: model_evaluation_history = LRCN_model.evaluate(features_test, labels_test)

         4/4 [==============================] - 2s 321ms/step - loss: 0.4498 - accuracy: 0.8770

In [25]: model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history


         date_time_format = '%Y_%m_%d__%H_%M_%S'
         current_date_time_dt = dt.datetime.now()
         current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_format)


         model_file_name = f'LRCN_model___Date_Time_{current_date_time_string}___Loss_{model_evaluation_loss}___Accuracy_{model_evaluatior

         LRCN_model.save(model_file_name)

In [28]: plot_metric(LRCN_model_training_history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss','No.of epochs')
```

Human Activity Recognition
Last Checkpoint: 01/26/2023 (autosaved)
loss
val_loss
No.of epochs

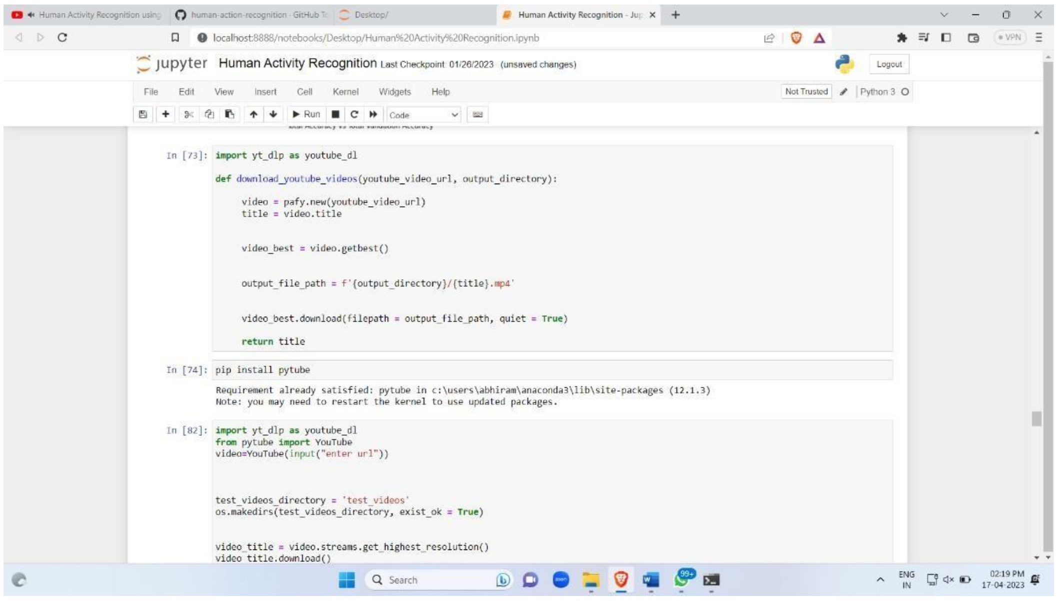
Fig. B.13: Evaluating the Model



```
In [73]: import yt_dlp as youtube_dl

         def download_youtube_videos(youtube_video_url, output_directory):

             video = pafy.new(youtube_video_url)
             title = video.title


             video_best = video.getbest()


             output_file_path = f'{output_directory}/{title}.mp4'


             video_best.download(filepath = output_file_path, quiet = True)

             return title

In [74]: pip install pytube
```

Human Activity Recognition
Last Checkpoint: 01/26/2023 (autosaved)
accuracy
val_accuracy
No.of epochs
Total Accuracy vs Total Validation Accuracy

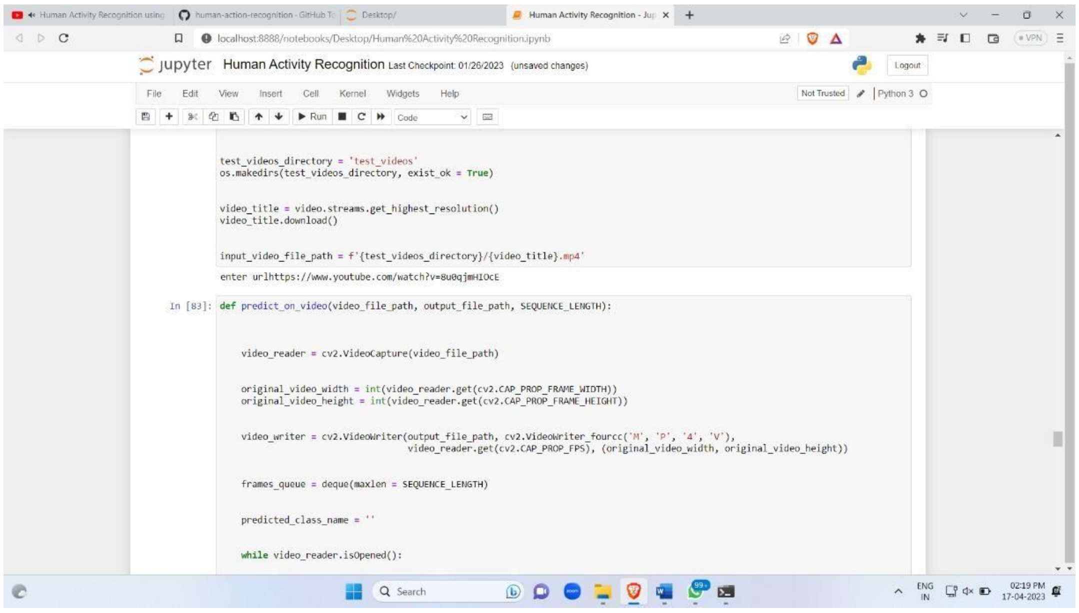
Fig. B.14: Plotting Loss and Accuracy Curves of LRCN Model



jupyter Human Activity Recognition Last Checkpoint: 01/26/2023 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Logout

```
In [73]: import yt_dlp as youtube_dl

         def download_youtube_videos(youtube_video_url, output_directory):

             video = pafy.new(youtube_video_url)
             title = video.title


             video_best = video.getbest()


             output_file_path = f'{output_directory}/{title}.mp4'


             video_best.download(filepath = output_file_path, quiet = True)

             return title

In [74]: pip install pytube

         Requirement already satisfied: pytube in c:\users\abhiram\anaconda3\lib\site-packages (12.1.3)
         Note: you may need to restart the kernel to use updated packages.

In [82]: import yt_dlp as youtube_dl
         from pytube import YouTube
         video=YouTube(input("enter url"))



         test_videos_directory = 'test_videos'
         os.makedirs(test_videos_directory, exist_ok = True)


         video_title = video.streams.get_highest_resolution()
         video_title.download()
```

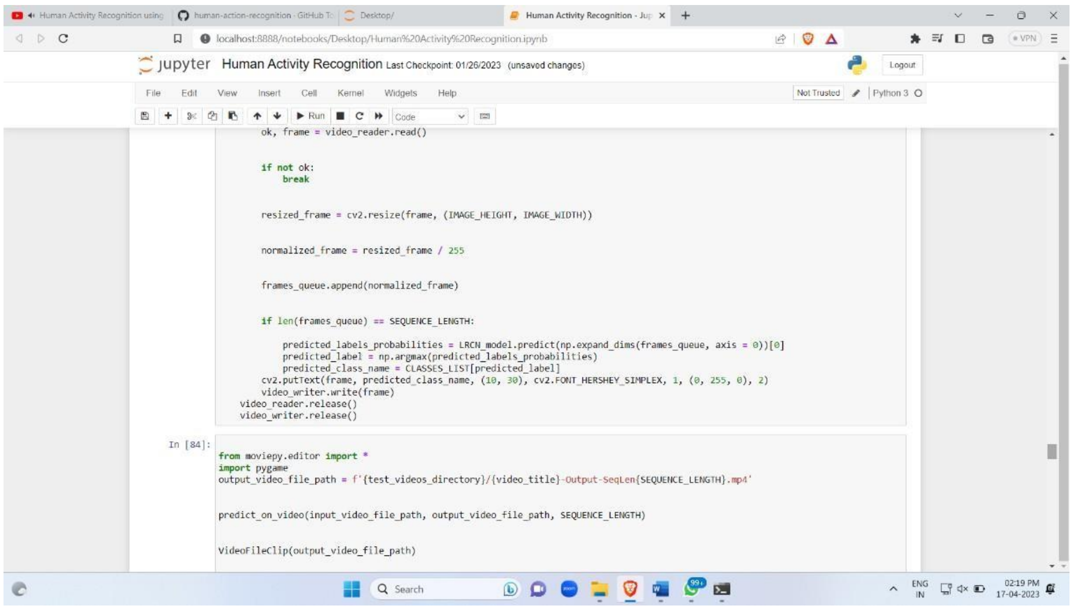
Fig. B.15: Function to download video from YouTube



jupyter Human Activity Recognition Last Checkpoint: 01/26/2023 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Logout

```
         test_videos_directory = 'test_videos'
         os.makedirs(test_videos_directory, exist_ok = True)


         video_title = video.streams.get_highest_resolution()
         video_title.download()


         input_video_file_path = f'{test_videos_directory}/{video_title}.mp4'

         enter urlhttps://www.youtube.com/watch?v=8u0qjmHIOcE

In [83]: def predict_on_video(video_file_path, output_file_path, SEQUENCE_LENGTH):



             video_reader = cv2.VideoCapture(video_file_path)


             original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
             original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))


             video_writer = cv2.VideoWriter(output_file_path, cv2.VideoWriter_fourcc('M', 'P', '4', 'V'),
                                            video_reader.get(cv2.CAP_PROP_FPS), (original_video_width, original_video_height))


             frames_queue = deque(maxlen = SEQUENCE_LENGTH)


             predicted_class_name = ''


             while video_reader.isOpened():
```

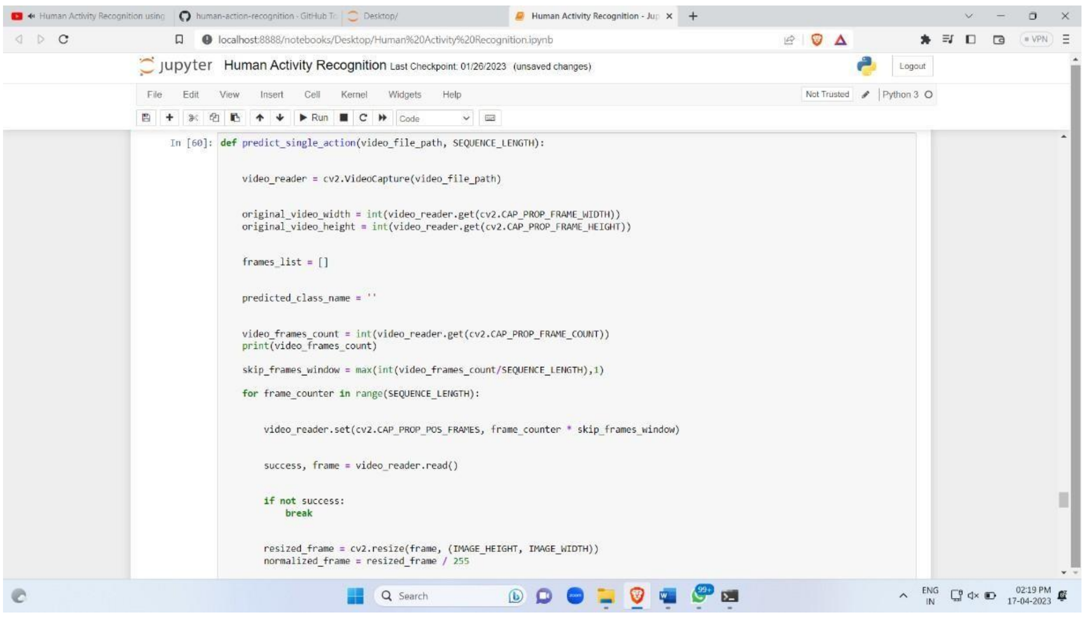
Fig. B.16: Predict on Videos Function



```
        ok, frame = video_reader.read()


        if not ok:
            break


        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))


        normalized_frame = resized_frame / 255


        frames_queue.append(normalized_frame)


        if len(frames_queue) == SEQUENCE_LENGTH:

            predicted_labels_probabilities = LRCN_model.predict(np.expand_dims(frames_queue, axis = 0))[0]
            predicted_label = np.argmax(predicted_labels_probabilities)
            predicted_class_name = CLASSES_LIST[predicted_label]
        cv2.putText(frame, predicted_class_name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        video_writer.write(frame)
    video_reader.release()
    video_writer.release()
```

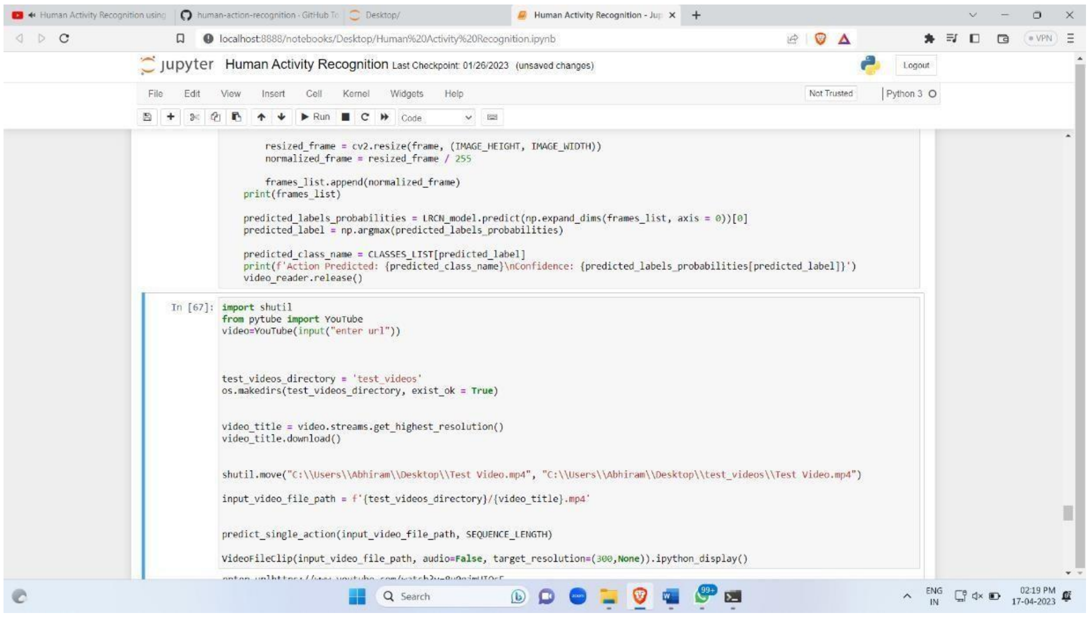
```
In [84]:
from moviepy.editor import *
import pygame
output_video_file_path = f'{test_videos_directory}/{video_title}-Output-SeqLen{SEQUENCE_LENGTH}.mp4'


predict_on_video(input_video_file_path, output_video_file_path, SEQUENCE_LENGTH)


VideoFileClip(output_video_file_path)
```

Fig. B.17: Predict on Single Video



```
In [60]: def predict_single_action(video_file_path, SEQUENCE_LENGTH):


    video_reader = cv2.VideoCapture(video_file_path)


    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))


    frames_list = []


    predicted_class_name = ''


    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    print(video_frames_count)

    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH),1)

    for frame_counter in range(SEQUENCE_LENGTH):


        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)


        success, frame = video_reader.read()


        if not success:
            break


        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255
```

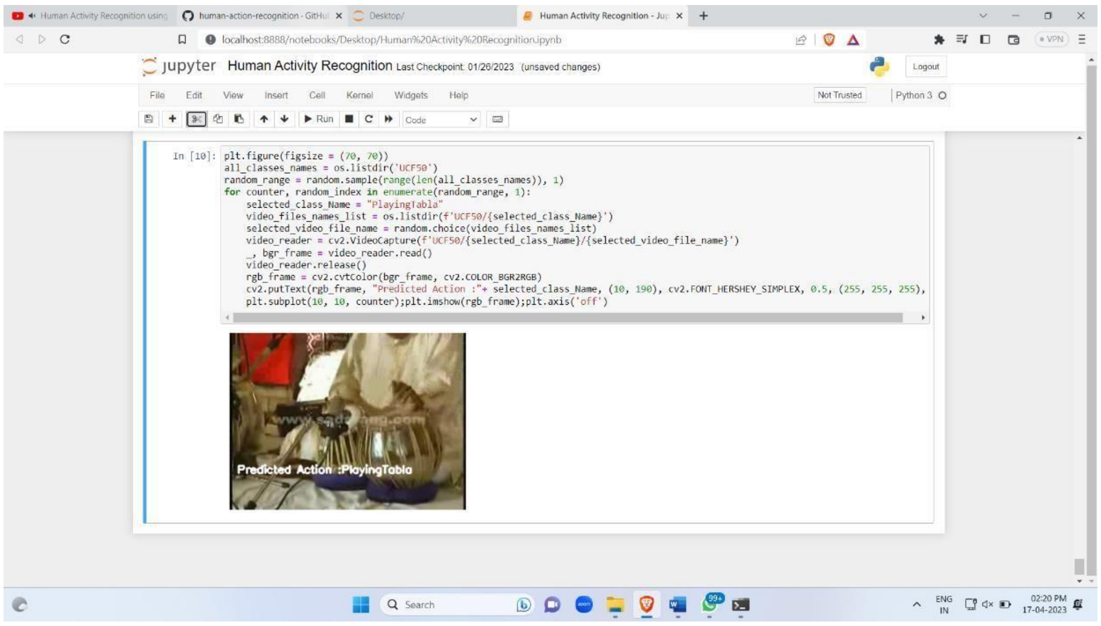
Fig. B.18: Predict Single Action



```
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255

        frames_list.append(normalized_frame)
    print(frames_list)

    predicted_labels_probabilities = LRCN_model.predict(np.expand_dims(frames_list, axis = 0))[0]
    predicted_label = np.argmax(predicted_labels_probabilities)

    predicted_class_name = CLASSES_LIST[predicted_label]
    print(f'Action Predicted: {predicted_class_name}\nConfidence: {predicted_labels_probabilities[predicted_label]}')
    video_reader.release()
```

```
In [67]: import shutil
         from pytube import YouTube
         video=YouTube(input("enter url"))



         test_videos_directory = 'test_videos'
         os.makedirs(test_videos_directory, exist_ok = True)


         video_title = video.streams.get_highest_resolution()
         video_title.download()


         shutil.move("C:\\Users\\Abhiram\\Desktop\\Test Video.mp4", "C:\\Users\\Abhiram\\Desktop\\test_videos\\Test Video.mp4")

         input_video_file_path = f'{test_videos_directory}/{video_title}.mp4'


         predict_single_action(input_video_file_path, SEQUENCE_LENGTH)

         VideoFileClip(input_video_file_path, audio=False, target_resolution=(300,None)).ipython_display()
```

Fig. B.19: Testing on Videos



```
In [10]: plt.figure(figsize = (70, 70))
         all_classes_names = os.listdir('UCF50')
         random_range = random.sample(range(len(all_classes_names)), 1)
         for counter, random_index in enumerate(random_range, 1):
             selected_class_Name = "PlayingTabla"
             video_files_names_list = os.listdir(f'UCF50/{selected_class_Name}')
             selected_video_file_name = random.choice(video_files_names_list)
             video_reader = cv2.VideoCapture(f'UCF50/{selected_class_Name}/{selected_video_file_name}')
             _, bgr_frame = video_reader.read()
             video_reader.release()
             rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)
             cv2.putText(rgb_frame, "Predicted Action :"+ selected_class_Name, (10, 190), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255),
             plt.subplot(10, 10, counter);plt.imshow(rgb_frame);plt.axis('off')
```

Fig. B.20: Picture showing predicted output

C. RESEARCH PAPER

Vision based detection and analysis of human activities

Abhiram Ravipati
Department of Computer Science and Engineering
Sathyabama Institute Of Science and Technology
Chennai, India
abhiramravipati9@gmail.com

Rakesh Krishna Kondamuri
Department of Computer Science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India
rakeshkondamuri97@gmail.com

Mary Posonia A
Department of Computer Science and Engineering
Sathyabama Institute Of Science and Technology
Chennai, India
maryposonia.cse@sathyabama.ac.in

Albert Mayan J
Department of Computer Science and Engineering
Sathyabama Institute Of Science and Technology
Chennai, India
albert.cse@sathyabama.ac.in

Abstract*— These days, it's not uncommon to see video cameras installed to keep an eye on pedestrians and motorists alike in a variety of public spaces. The proliferation of camera footage necessitates the creation of some means of deducing the nature of the activity captured on film. Due to the widespread availability of acquisition devices like handsets & camcorders, HAR may be used in a wide variety of contexts. The proliferation of electronic gadgets and software has been matched by a revolution in data extraction made possible by breakthroughs in AI. Difficulties such backdrop congestion, partly blockage, variations in size, perspective, illumination, & look, make it difficult to recognise human actions in video frame or still photographs. Multimodal recognition is necessary in several fields, such as cctv, machine-human contact, & robots for characterising human behaviour. Here, we survey the most current and cutting-edge findings from the study of how to categorise human actions. In this paper, we propose a taxonomy for studying human activity and explore the benefits and drawbacks of this diverse set of approaches. Disabled people's daily routines, including resting, moving, traveling down & up the stairs, speaking, & laying, have been frequently tracked using cellphones. Human motion detection often makes use of well-established ML & DL techniques include CNN, & LSTM Network.

***Keywords*—: Machine Learning, Human-Machine, Convolution Neural Network, Long short-term memory, Human Activity Recognition.**

I. INTRODUCTION

Accurately identifying & classifying human actions from wearable sensors is the goal of HAR, a topic of research. This is often accomplished by analysing information gleaned from portable tech like wearables, cellphones, or other monitors that record a user's movement, posture, or gestures. To improve areas as diverse as monitoring devices, sports evaluation metrics, and more, HAR seeks to create technologies that can accurately identify and comprehend a wide variety of human motion and activity in real time.

HAR is a method that may be used with either security cameras or regular cameras to identify a wide range of human activities. Researchers in the fields of healthcare, elderly services, or life welfare services have been more interested in HAR in recent years because to its potential to enable the automated monitoring and comprehension of patients' or residents' behaviours in "smart" settings like hospitals and residences. A HAR setup in a smart device, for illustration, may keep track of what its inhabitants get up to on a weekly, quarterly, & annual basis just by seeing them. The physicians may look into the resident' routines and habits and provide recommendations based on that information. A HAR system is able to detect falls and other abnormalities in older people's activity levels with relative ease. Activity recognition's foundational approach is made up of activity feature extraction, modelling, and recognition methods. The complexity of video-based HAR stems from the fact that, unlike arm movements or sign communications, motion detection of the complete body must be taken into account. Therefore, it is crucial to have a whole body model in order to accurately depict human motion. Despite the fact that video-based HAR technologies have attracted the attention of many academics owing to their potential use, reliable detection of human actions remains a significant difficulty.

Recognizing human actions is an intricate time series linear classifier called HAR. To accurately generate attributes from the original data to suit a ml model, prior approaches have relied on in-depth domain understanding and methodologies from noise removal. By applying machine learning characteristics from the raw data, dl techniques like CNN & RNN have recently shown competent and even produce state-of-the-art outcomes. This article will introduce you to the person activity identification issue and the state-of-the-art dl approaches that have been developed to solve it. Vision-based approaches towards person identification involve using image or video data to identify individuals. These methods typically involve using computer vision techniques to extract features from images or video frames that can be used to distinguish between individuals. These techniques have a wide range of applications in security, access control, and forensic investigations.

Some of the characteristics of human behavior in this classification are:

Motion patterns: Motion patterns refer to the way humans move their bodies when performing different activities. Different activities have unique motion patterns that can be captured using sensors. For example, walking has a distinct motion pattern compared to running or cycling. Posture and orientation: Posture and orientation refer to the position and orientation of different body parts during an activity. For example, standing has a different posture and orientation compared to sitting or lying down.

In essence, the task of categorising activities is a time series issue. One application of supervised ml is time-series categorization. It may be used for predicting & unloading sensor data, as well as predicting future values based on historical data using statistical methods. Neural networks are the current gold standard for wearable sensors. The most popular methods for this job are a subset of neural network models called CNN Models as well as a subset of RNN Models called RNN Models.

A subset of ml & computer vision, HAR seeks to autonomously identify and categorise human activities. With several potential uses in areas including health, sports, entertainment, & defense, it is quickly rising in importance as a r&d focus. Inertial sensors, magnetometers, & gyros are just some of the sensors that are often used in HAR system to track and record user motion and behaviour. Many handsets, watches, and fitness bands all include built-in sensors that allow for continuous monitoring of user activities and instantaneous data collection.

In the medical profession, for example, HAR may be used to track patients' activity levels and identify when they need assistance getting about. It has applications in both games & sporting, where it could be employed to monitor and enhance player efficiency, and in the latter it can be harnessed to allow players to direct virtual avatars with their own bodies. In the field of security, it may be used to monitor for and react to suspicious activity.

As a whole, HAR is making great strides forward, and the pace of technological progress suggests that even more advanced and precise algorithms will be produced in the not-too-distant ahead.

II. LITERATURE REVIEW

Over the last several years, researchers have examined a variety of approaches to the problem of identifying human actions. The literature on emotion recognition is reviewed in this area.

Using the Android operating system, UsharaniJ[1] proposed a HAR system. By analysing data from accelerometers, they were able to develop a programme that allowed for learning & categorization to take place in real time through the cloud. To improve the k-NN classifier's efficiency, reliability, overall processing time on the Android version while using less resources, they adopted the clustering k-NN strategy. They also determined that different device types and abilities resulted in different categorization times.

In their [2] article, DavideAnguita. developed the adapted SVM technique, which uses fixed point computing to generate an energy-efficient framework for the categorization of human actions using a phone. The displayed novel technique was developed with the intention of being used in a wide range of intellect applications & advanced surroundings to expedite computation while minimising the use of system resources, thereby reducing energy, while still achieving results competitive with those of other commonly employed classification methods.

MdZiaUddin[3] suggested a body-sensor-based behavior detection method using deep NSL based on LSTM to comprehend people's behaviour in various settings. KDA was used to improve feature aggregation across all tasks; this method increases inter-class scattered while decreasing intra-class scattering in the data. The suggested model outperformed other dl models like the RNN, CNN, &

Bayesian Belief Networks in terms of recall, with an impressive 98%.

For continuous estimation of human biological motions using cellphone motion detectors, Meyasam, Vakiili[4] presented a true HAR framework. Six gradual learning methods were employed to evaluate the system's efficacy over a set of twenty tasks; these results were then compared to those obtained using state-of-the-art HAR techniques like DTs, AdaBoost, and etc. The highest achieved accuracy was 95%, achieved with Sequential k-NN & Additive Naïve bayes Probabilistic.

A innovative older person surveillance technology based on a ml algorithm was presented in a publication by Jirapond Muangprathub[5]. The authors found that the k-NN algorithm with a k value of five provided the highest accuracy (96.45%) while attempting to identify the real-time activities of the aged. Moreover, they developed a system that provides geospatial data presentation for the elderly, and in emergency situations, the person may utilise a messaging instrument to ask for assistance.

For the purpose of identifying human activities within buildings, BaodingZhou[6] presented a convolutional neural network. Based on data from cellphones' motion sensors, gyros, accelerometers, & leading indicators, nine distinct activities were identified. Reliability of 98% was achieved using the suggested strategy.

Six separate behaviors may be recognised using data from smartphones, and AbdulmajidMurad[7] suggested using a deep Lstm to do so. When tested on the UCI datasets, the system performed at an impressive 96.70% correctness.

For activity recognition, it is necessary to provide people with sensors. Imprecise and unreliable biological sensing technologies is now available. The sensors must be installed in advance of HAR. Due to the high price of sensors and the even higher price at which we find ourselves in the event of a damaged sensor, using sensors is inefficient.

There are a few disadvantages too. Accuracy: Current systems have low accuracy and may generate false-positives /false-negatives despite technological and ml algorithmic advancements. This is especially true of elaborate actions that defy easy description. The accuracy of HAR systems is very dependent on the quality of the sensor used. For this reason, not all gadgets or wearables will include high-quality sensors. However, the battery life of wearables and cellphones used for HAR is often short, which might restrict their use and efficacy. Since this might compromise the accuracy of the data, it's a big issue for prolonged usage. Concerns have been raised concerning the gathering and handling of personal information for HAR, which raises issues of security and confidentiality of data. Protecting sensitive information requires stringent security and safety safeguards. Due to their high price tag, many people are priced out of using current HAR systems. In addition, some consumers may find it prohibitive to invest in such systems due to the high upfront cost of wearables, monitors, as well as other gear. The inability to easily compare and integrate data from various sources due to a general lack of standards and compatibility across HAR systems.

III. PROPOSED SYSTEM

After examining the shortcomings of prior efforts, we developed a novel method. The suggested system can identify human behavior with no extra sensors. Only the live footage from the cameras matters. For the purpose of detecting human activity, the suggested system collects data over a predetermined time period. Output reveals what a person is doing.

There is no preliminary configuration needed prior to rollout. Sensors are employed to monitor human activities in the current system. In this case, sensor equipment is unnecessary. Because of this, there is no longer a need to spend money on sensors. In other words, we can get by without using sensors. Adding new features and capabilities to the existing model is simple and requires no more hardware.

CNN –

CNN (Convolutional Neural Network) models are often used in human activity recognition, including in LRCN (Long-term Recurrent Convolutional Networks) models, for several reasons:

1) Posture and orientation: Posture and orientation refer to the position and orientation of different body parts during an activity. For example, standing has a different posture and orientation compared to sitting or lying down.
2) Robustness to variations: CNN models are designed to be robust to variations in input data, such as changes in lighting or orientation. In human activity recognition, variations in video data due to changes in lighting or position can affect the accuracy of the model. By using a CNN model to extract spatial features, the LRCN model can be more robust to these variations.
3) End-to-end training: LRCN models are trained end-to-end, meaning that both the CNN and LSTM (Long Short-Term Memory) layers are trained simultaneously. This allows the model to learn features that are optimized for the task of human activity recognition. By using a CNN model as part of the LRCN model, the model can

learn spatial features that are optimized for the task, leading to better performance.

The use of CNN model in human activity recognition is important because it allows for the extraction of spatial features that are important for identifying different activities. The CNN model also helps to make the LRCN model more robust to variations in input data and allows for end-to-end training, leading to better performance.

LSTM –

LSTM (Long Short-Term Memory) models are often used in LRCN (Long-term Recurrent Convolutional Networks) models for human activity recognition, for several reasons:

1) Temporal sequence modeling: LSTM models are designed to model sequences of data, which makes them well-suited for human activity recognition tasks where the temporal aspect of the data is important. By using an LSTM layer in the LRCN model, the model can learn to recognize patterns in the sequences of spatial features extracted by the CNN layer.

2) Memory retention: LSTMs are designed to remember past inputs, which is important for human activity recognition where the context of the activity is important. By retaining memory of previous inputs, the LSTM layer can better understand the context of the current input and improve the accuracy of activity recognition.

LSTM allows for the modeling of temporal sequences of spatial features, which are important for identifying different activities. LSTMs are also robust to variations in input data and can retain memory of past inputs, leading to better performance in human activity recognition tasks.

Data Pre-processing –

We'll proceed to preprocess the dataset. To simplify computations, we will first read the video files from the dataset and resize the video frames to a fixed width and height. We will next normalise the data to the range
[0-1] by dividing the pixel values with 255, which speeds up convergence while training the network. We'll implement a function that takes a video's path as an argument and outputs a list of the video's resized and normalised frames. Although not every frame will be added to the list because we only require a sequence length of evenly distributed frames, the function will interpret the video frame by frame.

Feature Extraction –

In Human Activity Recognition (HAR) using LRCN, feature extraction is a critical process to capture relevant information from raw input data, such as videos readings. The LRCN model combines Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM) layers to extract spatial and temporal features from the input data, respectively.

The feature extraction process can be divided into two parts: spatial feature extraction using CNN and temporal sequence modeling using LSTM. In the first part, the input video frames are passed through multiple convolutional layers that extract relevant spatial features from the frames. The output of the convolutional layers is a 3D tensor that contains information about the spatial features of the frames.

In the second part, the 3D tensor output from the CNN layers is fed to the LSTM layers for temporal sequence modeling. The LSTM layers can capture the temporal dependencies between the frames in the video and learn to represent the sequential patterns in the data. The LSTM layers are designed to process sequential data, such as videos, and are able to remember long-term dependencies in the input sequence.

The output of the LSTM layers is a fixed-length vector that represents the temporal features of the input video sequence. This vector can be used as input to a classifier that predicts the activity label. The classifier can be trained using supervised learning with labeled training data, where the input features are the fixed-length vector obtained from the LSTM layers, and the output is the activity label.

Overall, the feature extraction process in HAR using LRCN involves using CNN layers to extract spatial features from the input video frames and LSTM layers to capture the temporal dependencies between the frames and extract temporal features from the video sequence. The combination of these two types of features allows the LRCN model to effectively recognize human activities from video data.

Implementation -

To analyse and forecast on pictures, a CNN or ConvNet is a form of deep NN developed particularly for this task.

It uses kernels (called filtration) to analyse a picture & produce image features (which depict if a particular feature is visible at a spatial position or not), with number of features and map sizes increasing and decreasing, respectively, as we progress greater depth into the system via lumping processes.

An LSTM will take into account all of the preceding inputs before producing an output, making it ideal for use with a data series. RNN, of which LSTMs are a subtype, are notoriously ineffective at handling long-term connections in an original signal due to a phenomenon known as the disappearing gradients issue.

An LSTM unit is able to retain context for lengthy input vectors since it was designed to get around the diminishing slope problem.

Because of this, an LSTM is better equipped to tackle issues related to time series, such as timing prediction, voice control, translation software, and musical composition. However, for the time being, we will just investigate how LSTMs may help improve action detection models.

Let's go on to the method we'll use in this guide's walkthrough of creating an Activity Recognition system. In order to conduct Classification Process while taking use of the Geographic element of the movies, we shall utilise a CNN + LSTM model.

Numerous methods exist for classifying videos. The chosen method is a hybrid of the cnn and the lstm. At a certain instant in the input sequence, CNN will draw out relevant spatial characteristics. The next step is for LSTM to determine the connections between each frame.

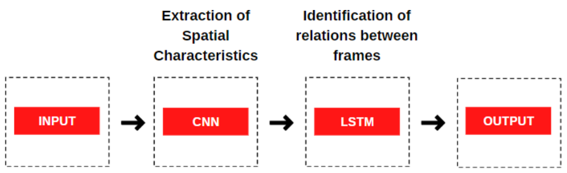

Extraction of Spatial Characteristics
Identification of relations between frames
INPUT
CNN
LSTM
OUTPUT

Fig.1. CNN+LSTM process model

The above diagram depicts the suggested system design. The footage was recorded straight from the camera. The captured footage is then given to the model that has been pre-trained. Human behaviour is recognised by the framework. The result is the performed action.

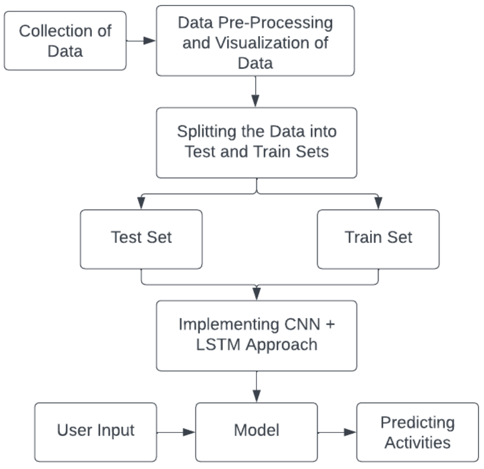

Collection of Data
Data Pre-Processing and Visualization of Data
Splitting the Data into Test and Train Sets
Test Set
Train Set
Implementing CNN + LSTM Approach
User Input
Model
Predicting Activities

Fig.2. Flow Diagram

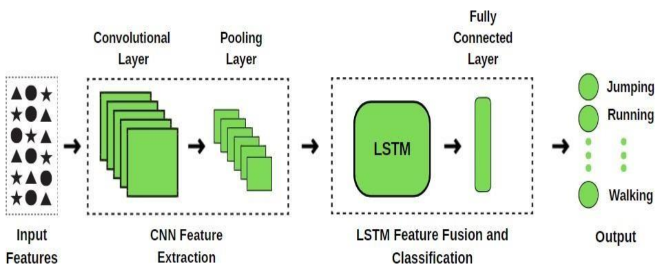


Fig.3. System Architecture

LRCN Model –

LRCN is a model that integrates CNN and LSTM layers into a single model. The CNN layers are responsible for extracting spatial features from the frames, while the extracted spatial features are then passed to LSTM layer(s) at each time-step for temporal sequence modeling. By doing so, the model is able to learn spatiotemporal features through end-to-end training, leading to a strong and reliable model.

IV. RESULTS & DISCUSSION

In order to verify that the CNN-LSTM system will function as expected, we undertook a series of tests to properly analyse its performance.

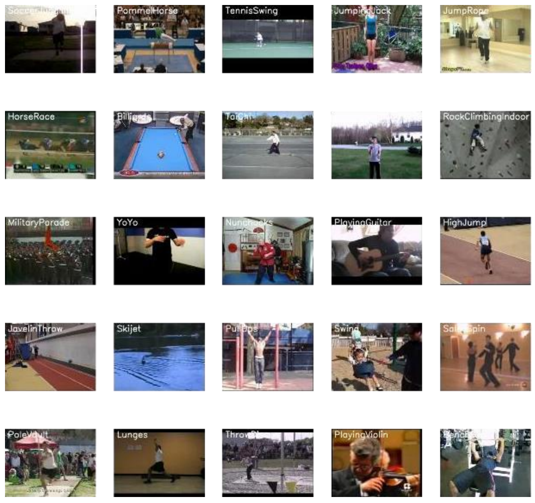


Fig.4. Dataset

Here we have Model's Loss and Accuracy curves from our training and validation steps

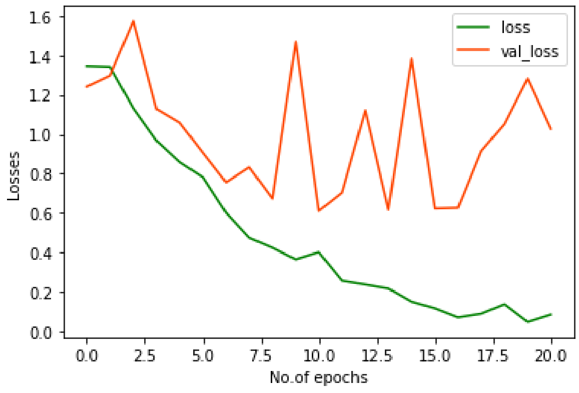


Fig.5. Total Loss vs Total Validation Loss

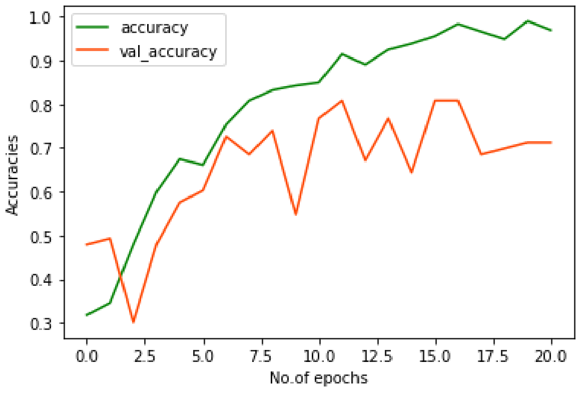


Fig.6. Total Accuracy vs Total Validation Accuracy

V. CONCLUSION & FUTURE WORK

Here we suggested a convolutional neural network (CNN) and long short-term memory (LSTM) method for human activity recognition. The above method hopes to enhance activity detection performance by making use of the former method's strong feature removal and the latter method's predicting future time sequence & categorization. Comparing the effectiveness of this CNN-LSTM network to that of other dl strategies that use raw sensor readings as input revealed that the latter is superior on both fronts. We tested our model on UCF Dataset, and we found that it performed well. We used LRCN model since it is robust and provides us an high accuracy. It achieved an accuracy of 85.25 %. The duration of time it required to execute the various systems and factor is determined was another parameter that was not assessed in this article but was evident in the trials. When contrasted to our method, the other algorithms took much longer to complete.

Future research will concentrate on expanding this model's capabilities and doing thorough evaluations with a variety of additional factors, such as activation functions, number of iterations, batch normalization, and others. In the future, we want to evaluate this model on other datasets so that we may use it for more complicated tasks in order to better meet the challenges of deep learning and HAR. We will also compare the outcomes of our method to the current state of the art on the UCI sample as well as other publically accessible datasets.

REFERENCES

[1] Usharani, J.; Saktivel, U. Human Activity Recognition using Android Smartphone. In Proceedings of the International Conference on Innovations in Computing & Networking ICICN16, Bengaluru, Karnataka, 2016.

[2] Anguita, D.; Ghio, A.; Oneto, L.; Parra-Llanas, X.; Reyes-Ortiz, J. Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic. J. Univers. Comput. Sci. 2013, 19, 1295–1314.

[3] Mary Posonia A, S. Vigneshwari, Albert Mayan J, Jamunarani D, "Service Direct : Platform that Incorporates Service Providers and Consumers Directly",International Journal of Engineering and Advanced Technology (IJEAT) , Vol.8 ,No.6,pp.3301-3304,2019.

[4] Uddin, Z.; Soylu, A. Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning. Sci. Rep. 2021, 11, 16455.

[5] Vakili, M.; Rezaei, M. Incremental Learning Techniques for Online Human Activity Recognition. arXiv 2021, arXiv:2109.09435.

[6] Muangprathub, J.; Sriwichian, A.; Wanichsombat, A.; Kajornkasirat, S.; Nillaor, P.; Boonjing, V. A Novel Elderly Tracking System Using Machine Learning to Classify Signals from Mobile and Wearable Sensors. Int. J. Environ. Res. Public Health 2021, 18, 12652.

[7]Joshila Grace L.K,Vigneshwari S, SathyaBama Krishna R, Ankayarkanni B, Mary Posonia A,"A Joint Optimization Approach for Security and Insurance Management on the Cloud", Lecture Notes in Networks and Systems,Vol.430, pp. 405–413.

[8]Zhou, B.; Yang, J.; Li, Q. Smartphone-Based Activity Recognition for Indoor Localization Using a Convolutional Neural Network. Sensors 2019, 19, 621.

[9] Murad, A.; Pyun, J.-Y. Deep Recurrent Neural Networks for Human Activity Recognition. Sensors 2017, 17, 2556.

[10] S. O. Eyobu and D. S. Han, “Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network,” Sensors, 2018, 18, 2892.

[11] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst and M. Hompel, “Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors,” Informatics, 5(2), 26, May 2018.

[12] K. Kim, S. Choi, M. Chae, H. Park, J. Lee, and J. Park, “A Deep Learning Based Approach to Recognizing Accompanying Status of Smartphone Users Using Multimodal Data,” Journal of Intelligence and Information Systems, vol. 25, no. 1, pp. 163–177, Mar. 2019.

[13] Mary Posonia A, Ankayarkanni B, Usha Nandhini D, Albert Mayan J, Nagarajan G (2022), "An Efficient Algorithm for Traffic Congestion Control", Advances in Intelligent Computing and Communication, Lecture Notes in Networks and Systems, Vol 430, Springer.

[14] F. Chollet, “Layer wrappers,” Keras Documentation, 2015, Online, Available: https://keras.io/layers/wrappers/#timedistributed, Accessed: Dec. 1, 2019.

[15] S. Hochreiter and J. Schmidhuber. “Long short-term memory,” Neural computation, 9(8):1735–1780, 1997.

[16] Wang, J., Zhang, X., Wu, Y., & Wang, Y. (2022). Unsupervised Learning of Human Activities from Long-Term Videos. IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2022.3182538.

[17] Yao, J., Zhang, L., Lu, J., & Xu, Y. (2021). Self-Supervised Learning of Human Activities from Temporal Segments in Videos. IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2021.3111372.