Human Activity Recognition

Major project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By
HRISHABH TRIPATHI (181298)
TANMAY KUMAR (181352)

UNDER THE SUPERVISION OF

Dr. Pradeep Kumar Gupta

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat,173234, Himachal Pradesh, INDIA

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Dr. Pradeep Kumar Gupta, Associate Prof, Department of CSE** Jaypee University Of Information Technology. We also declare that neither this project nor any portion of this project has been submitted for the granting of any degree or diploma elsewhere.

Supervised by:
Dr. Pradeep Kumar Gupta
Prof
Department of Computer Science & Engineering And Information Technology
Jaypee University Of Information Technology

Submitted by:
Hrishabh Tripathi
(181298)

Tanmay kumar
(181352)
Department of CSE
Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work presented in the project report titled "**HUMAN ACTIVITY RECOGNITION**" in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "**Hrishabh tripathi (181298**)" and "**Tanmay Kumar (181352)**" during the period from August 2021 to December 2021 under the supervision of **Dr. Pradeep Kumar Gupta**, Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat.

Hrishabh Tripathi
(181298)

Tanmay Kumar
(181352)

To the best of my knowledge, the preceding statement is correct.

Dr. Pradeep Kumar Gupta
Professor
Computer Science & Engineering And Information Technology
Jaypee University of Information Technology, Waknaghat, India
Dated:

ACKNOWLEDGMENT

First We convey our heartfelt thanks and gratitude to Almighty God for His divine grace, which has enabled us to successfully complete the project.

We are grateful to and grateful to **Dr. Pradeep Kumar Gupta**, Associate Professor, Department of CSE Jaypee University Of Information Technology, Wakhnaghat. My supervisor's extensive knowledge and genuine interest in the topic of "**Human Activity Recognition**" enabled me to complete this research. His unending tolerance, intellectual direction, constant encouragement, frequent and energetic supervision, constructive criticism, good advice, reading many poor draughts and revising them at all levels allowed us to finish this job.

We would like to offer our heartfelt gratitude **to Dr. Pradeep Kumar Gupta**, Department of CSE, for his generous assistance in completing my project.

We would also like to express our gratitude to everyone who has directly or indirectly assisted us in making this initiative a success. In this unique situation, we would like to thank the many staff members, both teaching and non-teaching, who have created their convenient assistance and helped my project.

Finally, we must express our gratitude for my parents' unwavering support and patience.

Hrishabh Tripathi
(181298)

Tanmay Kumar
(181352)

ABSTRACT

We know that human movement or activity recognition is growing relevance, not only in surveillance and security, but also due to diverse academics' interests in understanding human behavioural or movement patterns.

Previously, efforts to solving the challenge included manually building features from training machine learning models and time series data based on fixed-sized windows. The challenge is that this feature engineering necessitates extensive knowledge of image/video processing. Deep learning methods such as one-dimensional convolutional neural networks, or CNNs, and recurrent neural networks (RNNs) have recently been shown to provide state-of-the-art results on challenging activity recognition tasks with very little or no data feature engineering, instead relying on feature learning on raw data.

Our project focuses on the identification of human activities (HAR) problem, with input from a camera in the form of multiple channels time series data. In this scenario, extracting the functional parts of the job identification is critical yet difficult. Much of the available work is based on its handcrafted feature design heuristic and shallow structural learning features, which can find those dividing factors and accurately separate different functions. We suggest a systematic investigation of the HAR problem in this work.

We present an effective method for extracting a person's silhouette attributes from a video series in this study. Using morphological activities to measure the image silhouette, the suggested technique covers domain termination, edge recognition, circuit filling, and noise removal. To the best of our knowledge, the proposed method of erasing the silhouette requires finishing the background and discovering the first edge of its sort. In the Weizmann data set, we applied our proposed technique (standard). We applied those models on CNN after extracting features.

TABLE OF CONTENTS

INTRODUCTION

The interaction of machines and humans in the modern world has resulted in the development of new computer-based ideas such as the science of discovery, tracking, and, more broadly, identification of human behavior. The site's growth is motivated by the site's numerous and prospective uses in domains such as medical, surveillance, information rooms, and video detection and search. These advancements simplify machine behavior, enhancing human-machine interaction. Future computer vision systems are predicted to be able to discern between diverse movements, allowing them to identify and analyze human movements without being vector spatial structural indicators. In order to achieve this goal, we developed an algorithm based on silhouette vector spatial to detect continuous human activity in this project.

Human action Researchers have paid close attention to recognition. Essentially, research on vision-based techniques has been spurred by interest in gait analysis, gesture analysis, and task perception. Model-based techniques frequently adhere to Johansson's postulate that human perception of activity is influenced by structural information. Rods, cardboard models, volumetric models, and hybrid models that track both edges and regions are used to execute structural approaches to identification. Hidden Markov modelling (HMM) and multidimensional indexing are two further ways based on active recognition modelling. All model-based techniques, however, face the difficulty of comparing model parameters of varying complexity with human imagery.

Non-model-based systems recognize human activity through unstructured means using gross-form motion capabilities. One expression of such motions is the periodicity of human movements, which is frequently employed as a recognition criterion. Polana et al. use periodicity and spatiotemporal magnitude patterns of movement to recognize motions like walking and running.

Deep learning has recently evolved as a learning family of models aimed at creating a high-quality data summit model. In deep learning, a multi-layered in-depth architecture is created for autonomous feature design.

Each level of Deep Structure, in particular, introduces an indirect change to the effect of the preceding level, therefore data in deep learning models is represented as a sequence of characteristics from low to high level. Convolutional neural networks, deep belief networks, and autoencoders are among the most well-known deep learning models. Deep reading patterns can be investigated with or without supervision, depending on how the information on the label is used. Although computer diagnostics, natural language processing, and speech recognition all benefit deep learning models, they have never been properly exploited in the HAR area.

In this paper, we address the HAR problem by improving one deep learning model in particular - convolutional neural networks (CNN). CNN's major feature includes a number of processing units (for example, convolution, pooling, sigmoid / hyperbolic tangent squashing, rectifier, and normalizing). A large number of processing units can produce an accurate representation of the signal's local strength. Then, deep architecture enables the placement of multiple layers of these processing units on the stack, allowing this in-depth learning model to exhibit the intelligence of the symbols on various sizes. As a result, CNN's features are work-dependent rather than manual.

As stated in the next sections, we have integrated the notion of image processing for feature extraction and CNN into the HAR app. The input size is determined by the number of frames, frame height, and frame breadth, and is mapped correspondingly in the convolution network. We developed a new CNN model, which was trained on low resolution Weizmann datasets and tested on the same data set. The results suggest that the proposed technique is a highly competitive HAR algorithm. We are also looking on the effectiveness of CNN.

LITERATURE REVIEW

1. A survey of video datasets for human action and activity recognition Enrique J. Carmona, Jose M. Chaquet, and Antonio Fernández-Caballero

The authors referred to the top Pictorial-based occurrence and movement recognition or detection in this work, which has many applications such as searching video, computer-human interface, or in personal computer's community for visual surveillance. The authors of this work discussed human movement detection utilising varied and numerous datasets. The datasets employed in the system for recognition match with the same input file.

The research work presented by the authors in this paper is a comprehensive description of most of the datasets that are public in nature and are used for detecting human behaviour such as movement and activity based on visual or video, which has indeed allowed researchers to select the most suitable dataset in order to benchmark their algorithms, and the authors attempted to implement that work in this paper. The authors discussed human activity or movement and communicative-based recognition systems that aim to identify the activities and aims of more than one or one agent many observations in series with each of the agents who are of any given environment. The authors also noted how people are becoming more interested in using applications such as visual surveillance, video search, and human-computer interaction, and how this type of system or approach is utilised for behaviour recognition. According to the authors, detecting human movement or activity is the final step in a series of previously obtained tasks such as picture or visual capture, or segmentation of that, tracking of that, recognising that, and classifying that. The authors highlighted several studies that are roughly linked to human activity and action detection and are available, such as human motion or activity capture. There are numerous human action and activity detection datasets available. Authors testified on a total of 66 records when this survey was conducted. All of those records were taken between 2001 and 2012, but it was later discovered that roughly 80% of them were taken after 2005, with only 20% taken between 2001 and 2005.

Later on, roughly an initial cataloguing was done which was gone to the area of various saved actions and its collections like 28 datasets which belonged that class which was currently there were all described and mentioned during this review or report where a heterogeneous human based actions and movements related dedicated groups were there from which Authors referred. The authors referred to around 22 recordings, all of which were dedicated specifically to a specific motion or action. Authors referred to various different kinds and types of styles related to background and whether it was being used of and controlled, many different types of human interactions such as dangerous or any social or any diversity of different kinds of actors involved or different kind of ground related realities or truths, and many more from previous records.

According to the authors, further classifications were presented that took into account behaviors such as professional amateur or common person, the number of views (mono or Multiview), and whether or not the camera was moving. Older datasets with simple hand annotations delivered the authors' specified ground truth, whereas the most recent datasets provided the highest quality ground truth. The authors cited the widespread use of popular XML-based description languages such as Viper and CVML, which sped up the annotation process and allowed for a more detailed description of what happened in each frame, as well as BEHAVE, CAVIAR, and ETISEO, all of which are mentioned by the authors of the Virat dataset. Viewers, for example, have a variety of constantly updated activity records. The extension of MOCAP databases is another relevant topic that the authors of this paper could research. These are particularly appealing for action detection since they provide parametric data that can be used to build a thorough 3D model. The authors emphasised that new datasets encompassing a wide range of records have been developed in recent years, such as the 9 new datasets created between 2010 and 2012, which are among the datasets mentioned in the article as part of the authors' research effort. The authors also mentioned excellent outreach as well as more targeted and engaged action packages.

According to the authors, this study, conducted as part of the research, conceals the lack of a detailed description of crucial public datasets for video-based nudity recognition and tries to aid researchers in selecting the best appropriate dataset for benchmarking algorithms.

The authors compared all data sets on a variety of practical aspects, such as B. actual terrain data, scene shapes, the number of activities and actors, and references to earlier work that used these datasets.

Using the prior article, we discovered many forms of public records available for HAR in various categories:

i)Heterogeneous Action: Activities such as leaping, walking, sprinting, and waving are examples of heterogeneous action.

ii) Specific Action: abandoned goods, everyday activities, crowd behaviour, fall detection, pace, pose, and gestures are examples of specific actions.

iii) Other: This category covers motion capture (MOCAP), infrared imaging, and thermal imaging.

After finding a number of different behaviours from the above categories, we chose the Weizmann activities dataset from 2005. We chose a collection from the Heterogenous Action category and worked on the Weizmann Actions Data set, which was recorded in 2005.

2.

Actions as Space-Time Shapes

Ronen Basri, Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Lena Gorelick

The authors of this work discuss human movement in video sequences, which can be seen as silhouettes of a shifting torso and sticking out limbs in process articulated motion. The human actions were described as third-dimensional shapes induced by silhouettes within the space-time container, according to the authors. The authors of this study used a new technique for reading 2D shapes and generalized it to deal with volumetric space-time movement shapes.

Authors used houses of the response to the Poisson equation to extract space-time functions as well as neighborhood space-time saliency, movement dynamics, form shape, and orientation in their suggested study. These functions are beneficial for movement reputation, detection, and clustering, according to the authors. The authors' technique in this Research study is quick, in which video alignment was no longer necessary and was no longer significant in many scenarios where the historical past is known. Furthermore, the Authors demonstrated the durability of their approach in the face of partial occlusions, non-inflexible deformations, large scale and viewpoint changes, severe abnormalities within the overall performance of a movement, and the occasional first-rate video. Human movement detection is an important component in a variety of computer vision applications, including video surveillance, human-computer interaction, video indexing and browsing, recognition of gestures, sports evaluation, and dance choreography, according to the authors. The authors of this work represented movements as space-time forms and demonstrated that one of the pictures included a plethora of information about the movement done by the authors.

The quality of the recovered functions was proven, according to the authors, by the completion of the extremely simple classification scheme applied, such as nearest friends' classification and Euclidian distance. The authors also claimed that in many cases, the information contained in a single space-time dice was adequate to make a reliable classification, which was confirmed in the first classification experiment. According to the authors, trustworthy overall performance in real-life applications can be achieved by integrating data from the entire input sequence, including all of its space-time cubes, as demonstrated by the robustness studies. The Authors' proposed method has various advantages, including the elimination of the need for video alignment. Second, it was extremely linear within the form's range of space-time components. On a Pentium 4, 3.0 GHz, the typical processing time for a 1107050 pre-segmented movie, such as correcting the Poisson equation and extracting functions in MATLAB, was less than 30 seconds, according to the authors. Third, it could deal with low-quality video data, which would cause problems for other techniques that are predominantly focused on depth functions, such as gradients.

The authors also emphasized that when we focus on the most effective space-time form, we often overlook the form's deep truths. This paper technique may be combined with depth-based completely functions in the future to further improve overall performance. It was also possible to expand the number of space-time functions retrieved using the Poisson equation in this proposed technique in order to meet more difficult tasks and human gait reputation. Finally, the proposed technique can be implemented with little or no modification to current 3-D shape illustration and matching.

We downloaded data from this research that comprises of 90 low-resolution video sequences of nine different persons executing ten different everyday motions such as walking, running, skipping, jumping-jack, jumping-forwards, waving, and so on. The data is 180 × 144 pixels at 50 frames per second.

3. **Extraction of Features from Video File Using Different Image Algebraic Point Operations.**

Nachamai M, Pranti Dutta

The authors of this research stated that in the realm of human-computer interaction (HCI), facial point analysis and birth are the most crucial steps that can lead to a stable and effective bracket system such as facial expression identification and emotion bracket. Authors have given an approach or technique to the problem of automatic facial point birth from various films employing multiple image algebraic operations in this work. Those operations dealt with pixel intensity values as a group via a fine concept used in image analysis and metamorphoses. The authors proposed 11 operations in this paper, which include point deduction or subtraction, point addition, point addition, point division, edge detecting, average neighboring filtering, image stretching, log implementation, exponential operation, inverse filtering, and image thresholding, which were implemented and tested on images that were uprooted from three different tone-recorded videos, dubbed video1, video2, and video3. Those videos came in three different formats: avi, mp4, and wmv.

Grayscale and RGB data were used to test this project. The Authors analyzed three parameters to determine the success of each operation: processing time, frames per second (FPS), and sharpness of edges of point and points founded on image slants. The perpetration in this paper was done by the authors in MATLAB R2017a. The authors stated that the trade between man and machine is a truly ironic field in image processing with a vast horizon of investigation. The authors also stated that HCI is becoming more prevalent in fields such as pattern recognition, object discovery, face discovery, face recognition, emotion discovery through face and speech, target recognition, remote seeing, optic character recognition (OCR),3d business based on shape, nonstop fairly shadowing of business, and so on. Authors mentioned that Those fields were complicated due to factors like one to numerous mappings, intricate calculation and recognition problem. Picture processing has numerous steps with approaches to split these challenges down, according to the authors, including image acquisition, pre-processing, point birth, segmentation, representation/ description, recognition, and interpretation.

The authors of this research propose to investigate the performance of three different image algebraic point operations on three different films in order to improve the point birth approach. According to the authors, all procedures performed better in grayscale than RGB when measured in frames per second. Still, according to the authors, point addition was the fastest at 23.558 FPS, while average neighborhood filtering was the slowest at 3.155 FPS.

The authors discovered that in terms of processing time, grayscale outperformed RGB in all of those procedures. The authors discovered that point addition took the least amount of time to process (205.35 s) while average neighborhood filtering took the most time (s). The frames kept in edge discovery procedure had the highest sharpness 7.905 of any operation, whereas image thresholding had the smallest sharpness 0.021 in grayscale, according to the authors. Because the created law did not support RGB picture frames in MATLAB, the authors discovered that each procedure in RGB had three different sharpness values, except for edge finding and image thresholding.

4.**Automatic Motion Tracking of Human in a Surveillance Video**

Murtaza A. Khan, Mohammad A. AlGhamdi, and Sultan H. ALMotiri

The authors present a method for monitoring a person's movement in a sequence of video frames in this article. The method presented by the authors can be used to follow a walking or running individual on surveillance video captured by a single fixed camera. The method in this technique started with removing noise from the acquired images, then segmenting them using frame difference and binary conversion techniques, and then tracking the subject horizontally and vertically using a bounding box based on the occurrence of high intensity values.

The findings of the simulation revealed that the technology may be utilized for real-time tracking of persons in films with a frame rate of 25/30 frames per second, according to the authors. The proposed system by the authors served as part of the Internet of Things (IoT) and communicated the recorded video to the control center for processing in the context of a smart city, according to the authors. Motion tracking, according to the authors, is the act of detecting and tracking a moving object in a series of video frames.

The authors also mentioned that human motion tracking in video was a hot topic in machine vision research. Surveillance, human-computer interaction, and virtual reality were among the applications. The authors of this research proposed a simple method for tracking persons in films automatically. Filtering, segmentation, and monitoring are the three primary aspects of the procedure described in this study. The filter phase, which used an average filter to eliminate noise from images, was the first of the three phases. The Authors assumed that high intensities contributed to motion, so the segmentation phase calculated frame differences and then converted the frame difference images into two-level like black and white images to separate high and low intensity values, and the tracking phase drew the bounding box around the human-like moving object. The Authors had scanned the row and column of each two-layer image in the trace to determine the bounding box's height and breadth, respectively. The Authors' proposed method can be used for real-time monitoring and has applications in security and surveillance.

5. A Novel Approach for Human Silhouette Extraction from Video Data Debotosh Bhattacharjee, Amlan Raychaudhuri, Satyabrata Maity, Amlan Chakrabarti, and Amlan Raychaudhuri

The authors proposed a method for efficiently extracting human silhouettes from video sequences in this research. Background removal, edge detection, area filling, and noise removal were all proposed in this study as ways to estimate an image's silhouette using morphological processes. The Authors' proposed approach for silhouette extraction, which included backdrop removal and edge recognition, was the first of its type.

The authors used their proposed technique on a Weizmann-like standard dataset and then compared the results to the most recent related research work in this model. In this study, the authors discovered that statistical measurements like as precision, recall, and F-measure clearly demonstrated the superiority of their strategy and so validated its uniqueness. The authors stated that the extraction of human silhouettes from video was an essential step toward shape-based analysis for many person-based video applications.

The authors also mentioned several video applications, such as CCTV cameras for indoor surveillance, geriatric monitoring, and office floor surveillance, where the camera is fixed in a specific position, and background information extraction is a crucial step. Our suggested method rapidly removes the backdrop to accurately extract human silhouettes as foreground objects. The authors developed an approach for generating an efficient human silhouette from video sequences in this research.

The authors stated that precise human silhouette synthesis from a video sequence was extremely valuable in a variety of application fields such as gait recognition, human activity recognition, human detection and tracking from movies, and so on. The method suggested by the authors in this study can also be used to obtain silhouettes for other moving objects such as animals, cars, and so on, which will be valuable for object recognition and classification problems. According to the authors, their proposed method produced great results for such video sequences. However, the shortcoming of this system was that the camera had to remain stationary during the movie sequence. The research could be expanded to create an effective human silhouette extraction process that works when a human's body color or garment color is very near to the background color.

This paper describes an efficient method for extracting the human silhouette from a video clip. To measure picture silhouette, this paper approach covers background removal, edge detection, circuit filling, and noise removal utilizing morphological processes. For the first time, a method for removing the silhouette that combines the removal of the background and the acquisition of the edge is presented in this study. We proposed strategies in the standard Weizmann database and compared the outcomes to the most recent related academic work. Comparisons using mathematical measurements such as accuracy, memory, and F-measure clearly demonstrate the superiority of this approach.

6. **Video-Based Human Activity Recognition for Elderly Using Convolutional Neural Network**

Ponniamma M., Vijayaprabakaran K., Sathiyamurthy K.

The authors of this research discussed a typical healthcare application for the elderly to track everyday activities and provide assistance. The authors stated that the system's automatic image processing and classification was difficult in comparison to human vision. The authors also mentioned some difficult challenges for activity recognition from surveillance video, such as the complexity of scene analysis in observations with irregular lighting and low-quality frames. The Authors' solution in this article uses machine learning algorithms to improve activity detection accuracy.

The system has a Convolutional Neural Network (CNN), a machine learning method used for picture classification, according to the authors. The system presented by the authors in this work intends to recognize and support the human activities of the elderly by exploiting entrance surveillance footage. The authors mentioned the RGB image in the dataset that was utilized for training, which required greater processing power to classify. They also mentioned that by employing the CNN network for image classification, they acquired an accuracy of 79.94 in the experimental section, indicating that their model achieves good accuracy for image classification when compared to other pre-trained models.

They claimed that the Human Activity Recognition (HAR) technology is growing popular in the intelligent healthcare setting as artificial intelligence grows. However, according to them, the major purpose of activity detection is to identify human actions from a series of studies of human actions and their ambient state. Artificial intelligence has made significant advances in identifying diseases, evaluating medical imaging, and prescribing medications to patients, among other things. They discussed several machine learning algorithms offered in this study to gain deeper insight into the sensor data produced by the numerous devices that monitor persons in a smart health environment. This research addressed the issue of video-based human activity detection in the intelligent health environment for the elderly. They introduced a deep convolutional neural network in this paper to extract features from successive video frames. These frames were first processed with pre-processing procedures. To improve routing, the deep convolutional network's poor training in the middle layers incorporates shortcuts between network levels utilizing three alternative methodologies. A Deep Convolutional Neural Network with hopping connections from the initial layer to all layers outperforms the other proposed models among these three upgraded deeper models.

The experimental findings revealed that the suggested model outperformed other traditional models such as ResNet, VGG, and AlexNet. They employed 2D representations of video frames to represent people's actions in this work, and the spatial features of the information were only used for recognition.

The study could be expanded further by incorporating temporal aspects and experimenting with different kinds of representation as input.

7.

Human Activity Recognition Based on Silhouette Directionality

Meghna Singh, IEEE Student Member, Anup Basu, IEEE Senior Member, and Mrinal Kr. Mandal, IEEE Senior Member

The authors discussed current advancements in computer vision and pattern recognition that have fueled several programmers targeted at intelligently recognizing human activities in this research. The authors present an algorithm for detecting non-contact human activity in this study. The authors extracted motion information and generated silhouettes (foreground) from the input films using an adaptive background-foreground separation technique. The authors then constructed feature vectors based on the directionality of the silhouette contours (direction vectors) and used the discrete data distribution of the directional vectors in a vector space for clustering and recognition. They also used human motion's dynamic nature to smooth judgments over time and eliminate activity detection errors. Their method is monocular, tolerant of mild view alterations, and applicable to most activities' front and side perspectives. The authors cite trials using short and lengthy video sequences that show strong detection under diverse viewing angles, zoom depths, backdrops, and frame speeds.

They also noted that the dynamic interactions between machines and humans have resulted in the establishment of a new discipline of computer vision that includes the science of identifying, tracking, and generally recognizing human behavior. The authors went on to say that computer vision research is continuously seeking to catch up to human vision research. They also highlighted that researchers' efforts to establish a universal approach for detecting human activity continue to face problems. They presented a novel non-model-based silhouette direction technique for identifying human activity under the premise of limited occlusion in this work. In contrast to most recent work, which deals with template matching of preset static activity poses, the algorithm they proposed captured the static and dynamic like transitional aspects of human activity.

The authors' solution is storage efficient since each action is saved and indexed as an eight-dimensional vector. Furthermore, because they are only concerned with the silhouette's contour, they eliminate the computational overhead of computing the motion for each body part or template match.

The algorithm suggested in this research is translation-independent and can manage changes in viewing angle, scale, backdrop, and clothing. It can also manage limited occlusion. However, the scientists also stated that for people with considerably different body shapes, the algorithm must be taught with a completely separate training set and is no longer compatible with the previous training data.

The experimental findings of this suggested model demonstrated good detection rates with rare misclassifications caused primarily by poor foreground-background separation characteristics. When considering eight activities without temporal smoothing, the CRRs achieved in their studies ranged from 85 percent to 99 percent. A performance evaluation for noisy data, which could come from the loss of pixels in the foreground, was also given. CRR increases with frame rate, and when temporal smoothing is used, 100 percent detection is attained. The algorithm's promise is indicated by its ease of implementation. The authors' proposed method can be used to track numerous identities and detect actions of persons in nursing homes or special care facilities.

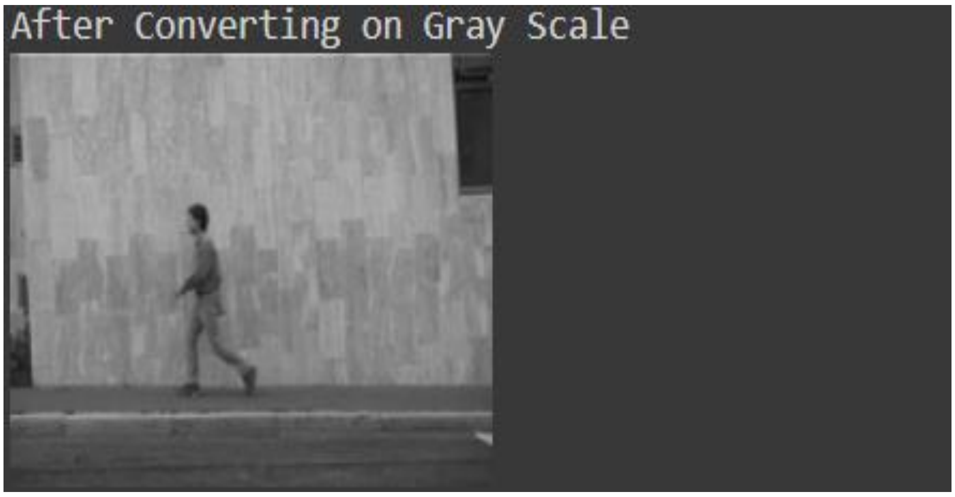
ALGORITHMS

We extracted video frames and converted them to grayscale frames. After subtracting the backdrop frame from the current frame, a new image is formed. The foreground is extracted from the subtracted image using a threshold value equal to the standard deviation of the subtracted image.

Formula Used:

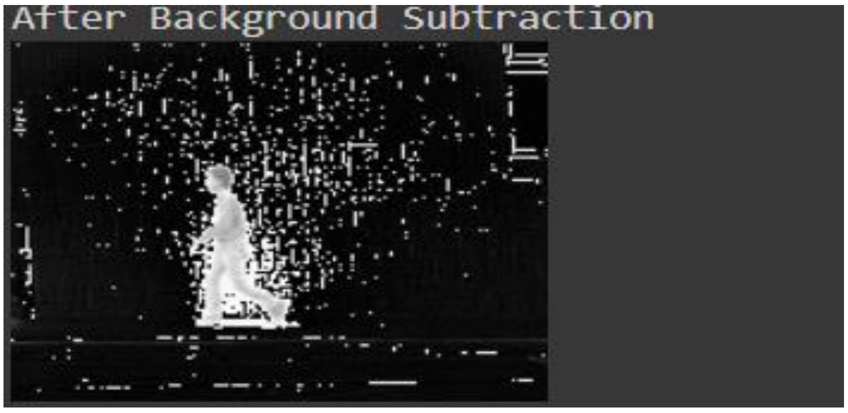
RGB to Grayscale:

$$I(g)=0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$



Subtracted Frame:

$$I = (Currentframe - background)$$



Foreground Detection:

$$I1\ (x, y)\ = 1, if\ |I(x, y)| > \sigma\ else\ 0$$

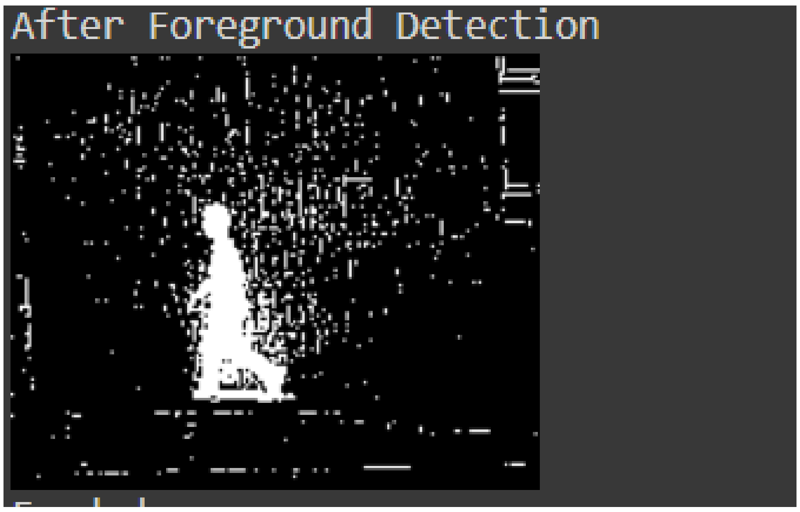
σ is standard deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

N is the population size.

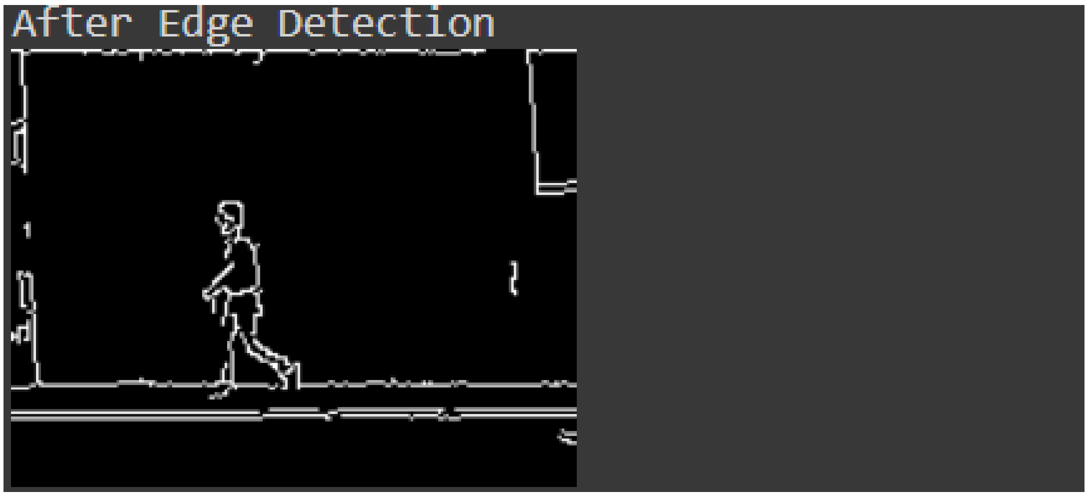
x(i) = each individual value from the population

μ = the population average



Boundary Edge Detection:
For edge detection, we used canny edge detection. As a result, we have a new image that we can refer to as I2.

Canny Edge Detection: This is a technique for extracting useful structural data from various visual items and drastically reducing the amount of data to be processed. It has been extensively tested in a variety of computer vision frameworks. Canny discovered that the requirements for facet detection utility on varied computer vision structures and very similar domains.



Merging of Two image:

$$I3\ (x, y)\ = 1, if\ \ I1(x, y) == 1\ or\ I2(x, y) == 1\ \ else\ 0$$

Morphological Operation:

After integrating two separate bits of information to get the final silhouette image, morphological processes are employed to lessen the effect of undesired noise. Some human silhouette pixels may be identified as background pixels, while others may be considered as foreground pixels. By performing morphological manipulations on the image I3, these misclassifications can be decreased. Fill the bounded sections with foreground color that is surrounded by an edge boundary in the first phase. This step is required if certain foreground pixels within the foreground boundary are regarded as background pixels. The resulting image may have some noise outside of the silhouette zone. The morphological erode procedure is then used to remove noises from outside the human silhouette.

After employing all of the aforementioned procedures, we obtained a new image that may be regarded as a new frame.

Erosion: This operator requires two inputs. The first is an image to be degraded, and the second is a set of coordinate points known as kernels. Kernel calculates the exact effect of erosion on the provided image.

On binary images, erosion can be represented mathematically as follows:

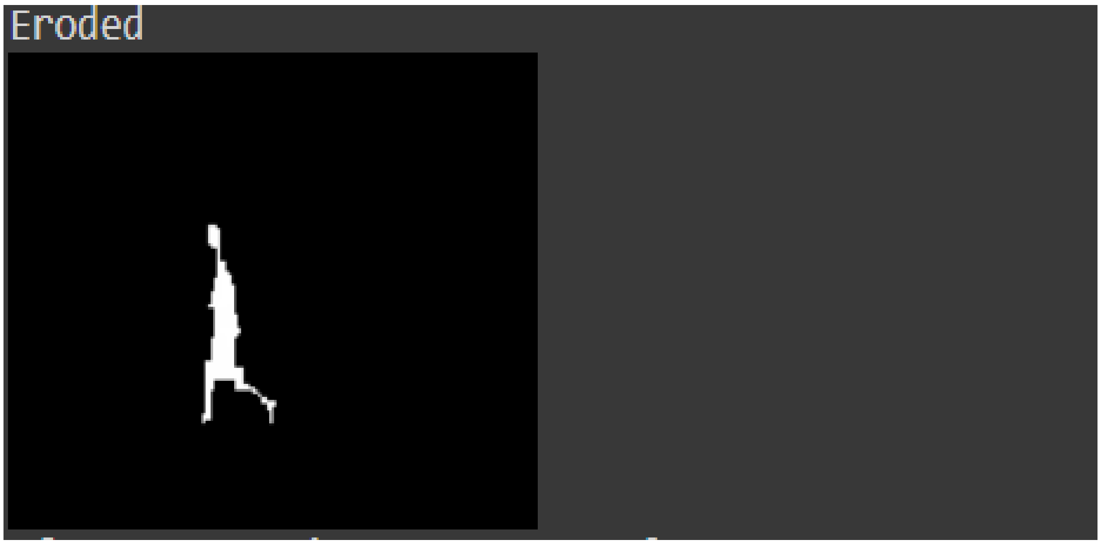
I Assume X is the set of Euclidean coordinates for input images and K is the kernel element coordinate.
ii) Assume K(x) reflects K's translation such that x can be the origin.
iii) The set of all points x such that K(x) is a subset of X is the erosion of X by K.

As indicated in the picture, the kernel is a 3 × 3 square matrix with the center as the origin.

1	1	1
1	1	1
1	1	1

The coordinates are as follows: =
{(-1, -1), (0, -1), (1, -1),
(-1,0), (0,0), (1,0),
(-1,1), (0,1), (1,1)}

To calculate the erosion on the binary input image by the aforementioned kernel element, we must consider each foreground pixel in the given image turn by turn. We superimposed the kernel on top of the given image for each foreground pixel, so that the kernel's origin is at the given pixel coordinates.



Machine learning Model

Construction of Model:

For the data we have, a simple convolutional model is built. The model comprises layers that alternate between convolutional and pooling. The advantages of having such a model are that it may be used to encode the content of an image into a vector with less height and breadth but more depth. This means that the convolutional layers will be used to make the input deeper (raise the depth of the picture, resulting in a stack of numerous feature maps), whilst the pooling layers will be used to minimize the spatial dimensions of the input.

We must configure the following parameters for each convolution layer:
Filters: This is the feature that is utilized as the convolution output layer.

kernel size: This is the size of any window that will be convolved with all of the input data axes to produce a single feature map.

strides: The number of pixels by which our convolution window should be displaced.

padding: It determines what happens at the edges.

activation: The activation function should be employed at that layer; in this case, we are using RELU activation.

Pooling is a pooling process that computes the maximum value for a feature map's patches and utilizes it to produce a down sampled (pooled) feature map.

pool size: The window's size.
strides: The number of pixels the pooling window should shift by.
padding: To control what happens around the edges.

Data Preparation:

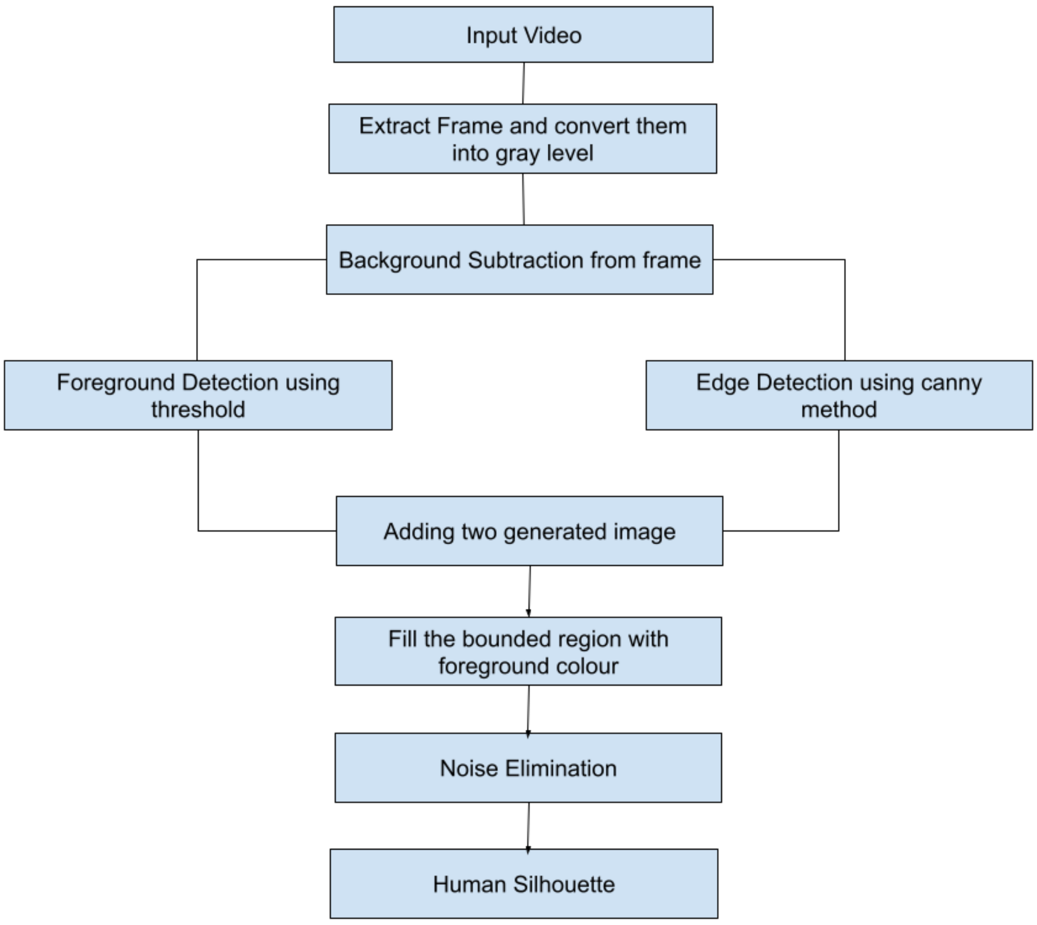
We compiled a library of 90 low-resolution (180 x 144, deinterlaced 50 fps) video sequences of nine different persons performing 10 normal actions such as walking, jumping, waving, and so on. We transformed these 90 data points into 183 low-resolution videos when working with data. Each frame of the movie was processed and decreased by 128 by 128.

Model Training: To avoid the problem of model overfitting, we separated the dataset into two portions for training and testing reasons. 66 percent (120) of the data set was chosen for model training, while the remaining 34 percent (63) was chosen for testing. The training dataset was further subdivided into training data sets and validation data sets. We employed 30 data points for model validation, while the remaining data points were used for model training.

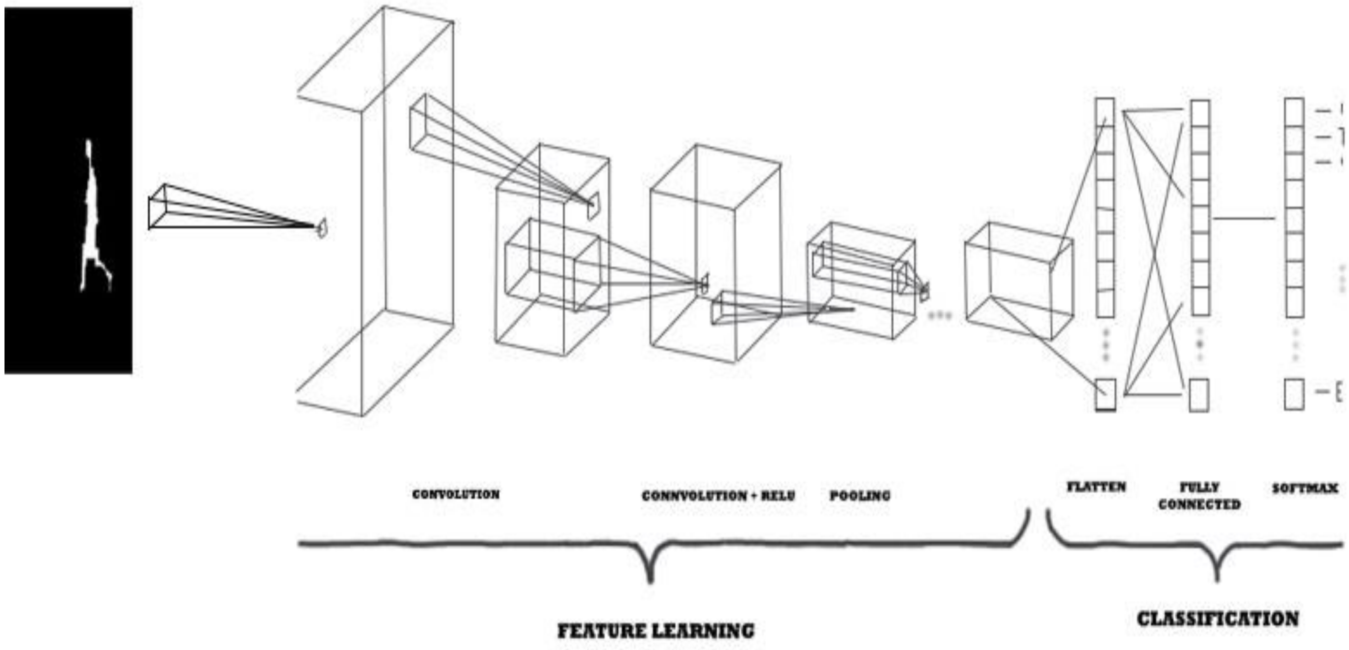
Comparison

Method	**Accuracy**
Bag of 3D points	74.70%
HOJ3D	79.00%
Actionlet Ensemble	82.22%
Depth Motion Maps	88.73%
HON4D	88.89%
Moving Pose	91.70
SNV	93.09%
CNN+SAE	74.6%
Proposed Model	30.00%

Flow Chart of feature extraction:

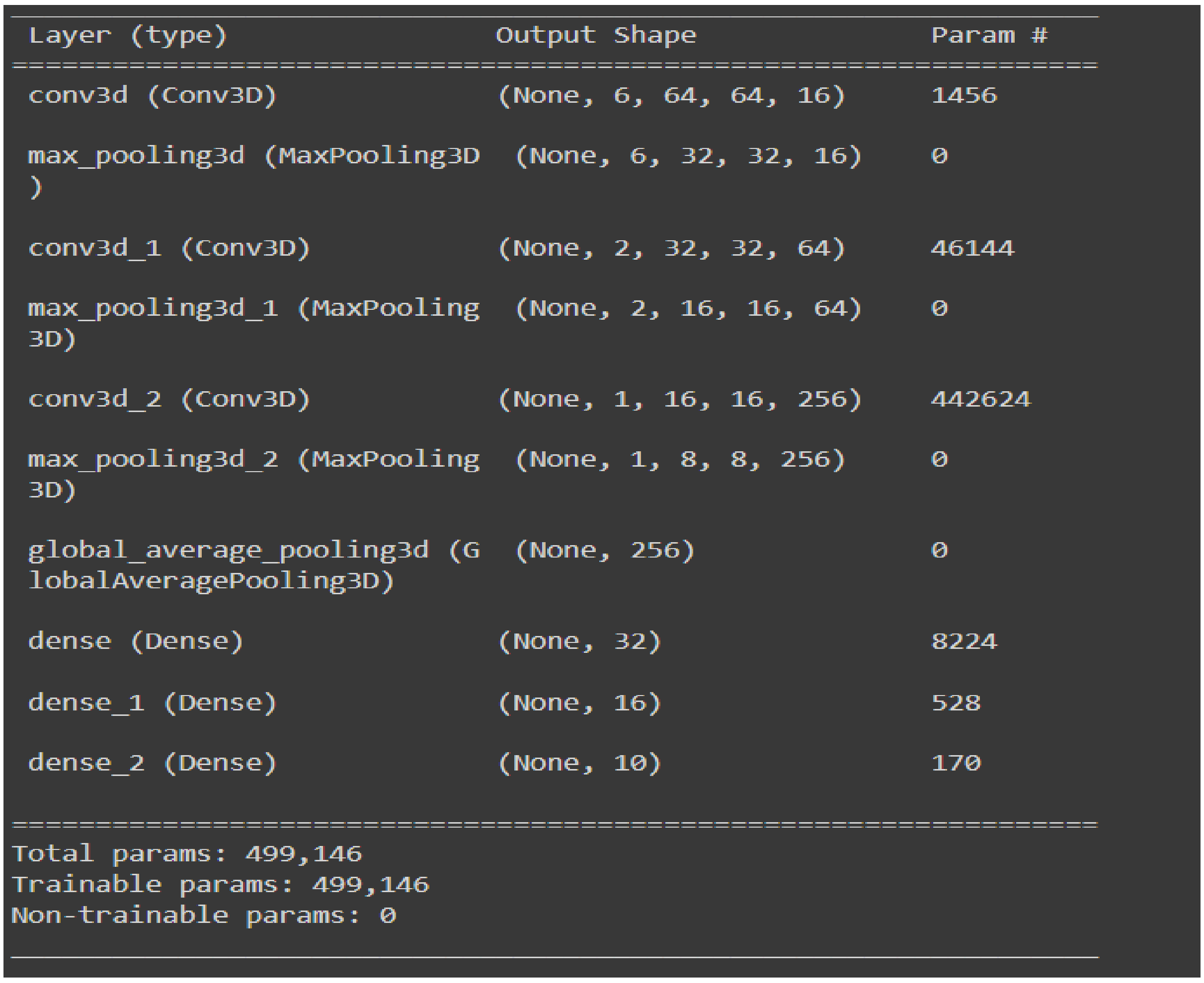

Input Video
Extract Frame and convert them into gray level
Background Subtraction from frame
Foreground Detection using threshold
Edge Detection using canny method
Adding two generated image
Fill the bounded region with foreground colour
Noise Elimination
Human Silhouette

Flow chart of Machine Learning Model:



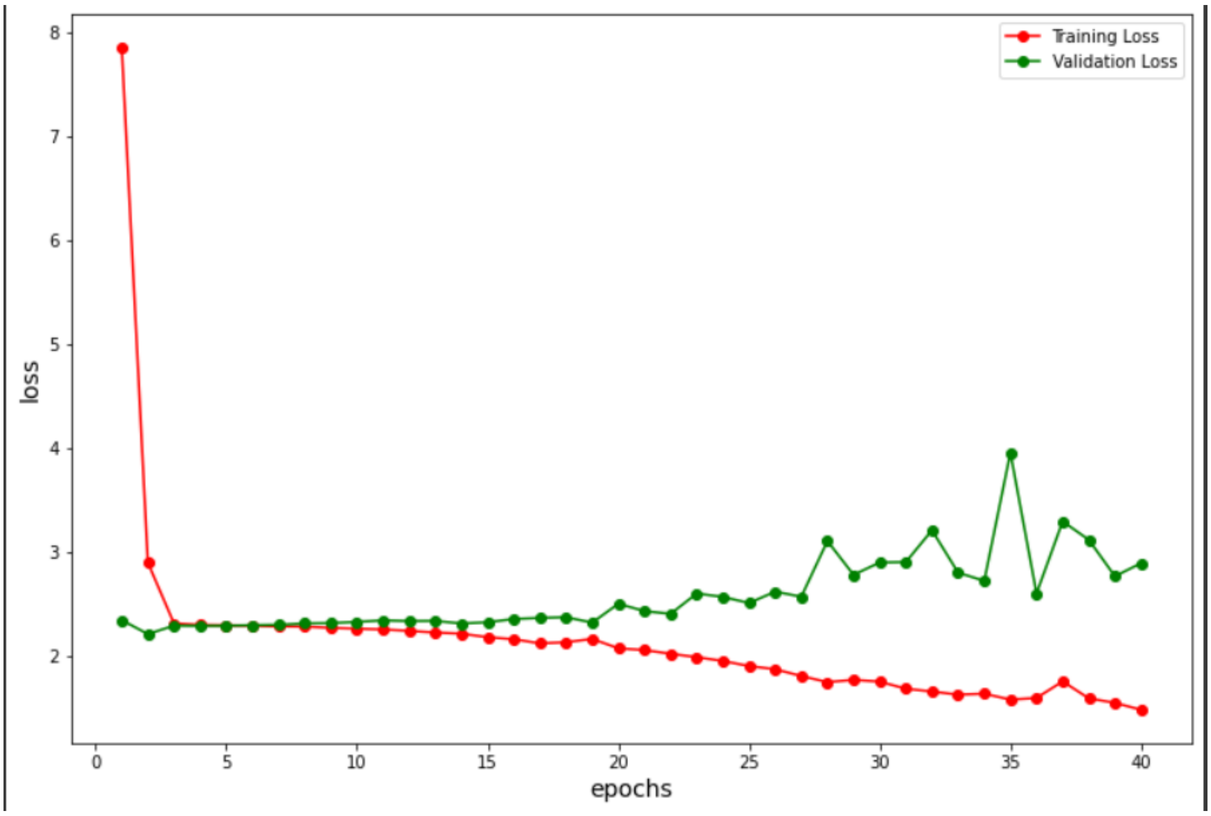
Model Summary:

Model 1:



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 6, 64, 64, 16)     1456

 max_pooling3d (MaxPooling3D  (None, 6, 32, 32, 16)    0
 )

 conv3d_1 (Conv3D)           (None, 2, 32, 32, 64)     46144

 max_pooling3d_1 (MaxPooling  (None, 2, 16, 16, 64)    0
 3D)

 conv3d_2 (Conv3D)           (None, 1, 16, 16, 256)    442624

 max_pooling3d_2 (MaxPooling  (None, 1, 8, 8, 256)     0
 3D)

 global_average_pooling3d (G  (None, 256)              0
 lobalAveragePooling3D)

 dense (Dense)               (None, 32)                8224

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 10)                170

=================================================================
Total params: 499,146
Trainable params: 499,146
Non-trainable params: 0
_________________________________________________________________
```

Result:

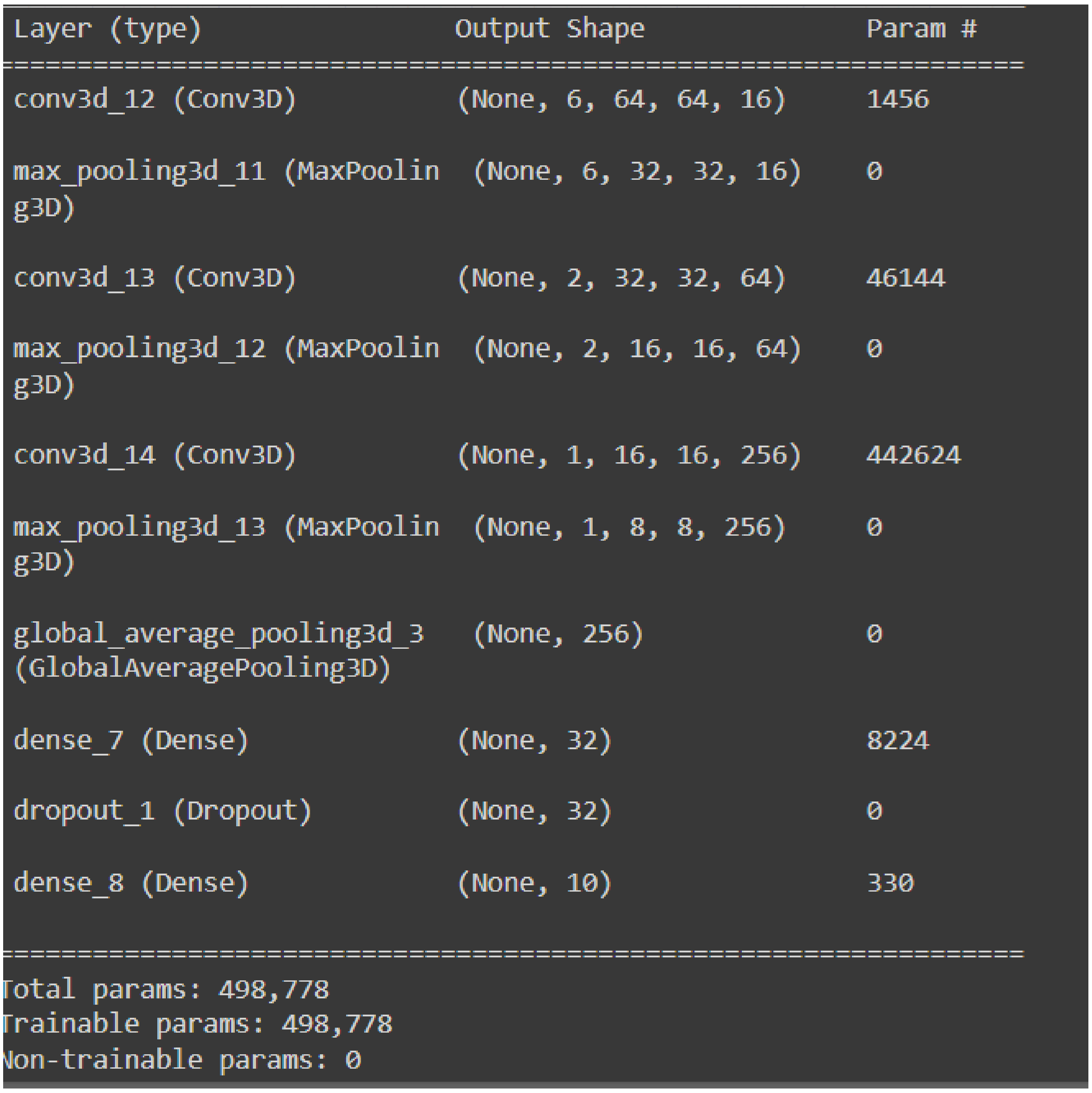


We may conclude from the graph above that the above model is overfitted as the value difference between training loss and Validation grows.

To avoid overfitting, we presented another model:

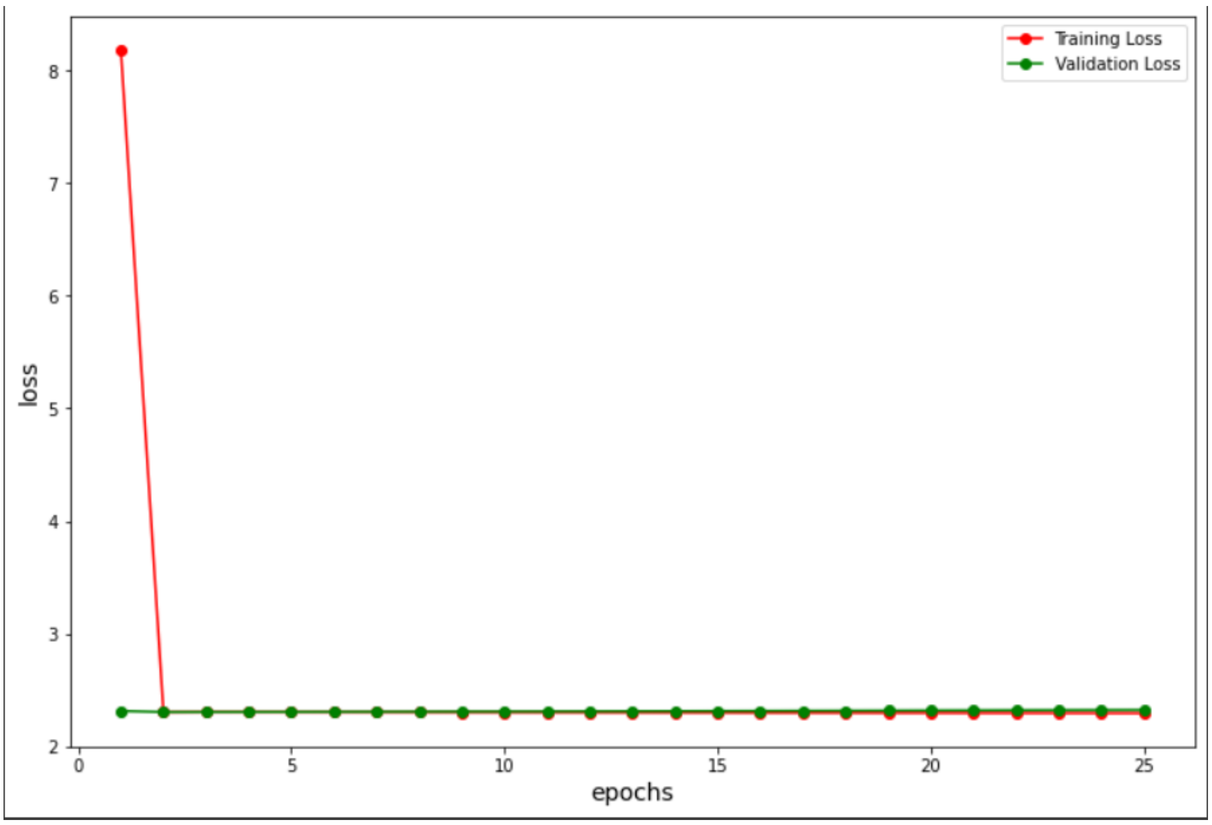
Model 2:

Summary



```
Layer (type)                Output Shape              Param #
=================================================================
conv3d_12 (Conv3D)          (None, 6, 64, 64, 16)     1456

max_pooling3d_11 (MaxPoolin  (None, 6, 32, 32, 16)    0
g3D)

conv3d_13 (Conv3D)          (None, 2, 32, 32, 64)     46144

max_pooling3d_12 (MaxPoolin  (None, 2, 16, 16, 64)    0
g3D)

conv3d_14 (Conv3D)          (None, 1, 16, 16, 256)    442624

max_pooling3d_13 (MaxPoolin  (None, 1, 8, 8, 256)     0
g3D)

global_average_pooling3d_3   (None, 256)              0
(GlobalAveragePooling3D)

dense_7 (Dense)             (None, 32)                8224

dropout_1 (Dropout)         (None, 32)                0

dense_8 (Dense)             (None, 10)                330

=================================================================
Total params: 498,778
Trainable params: 498,778
Non-trainable params: 0
```

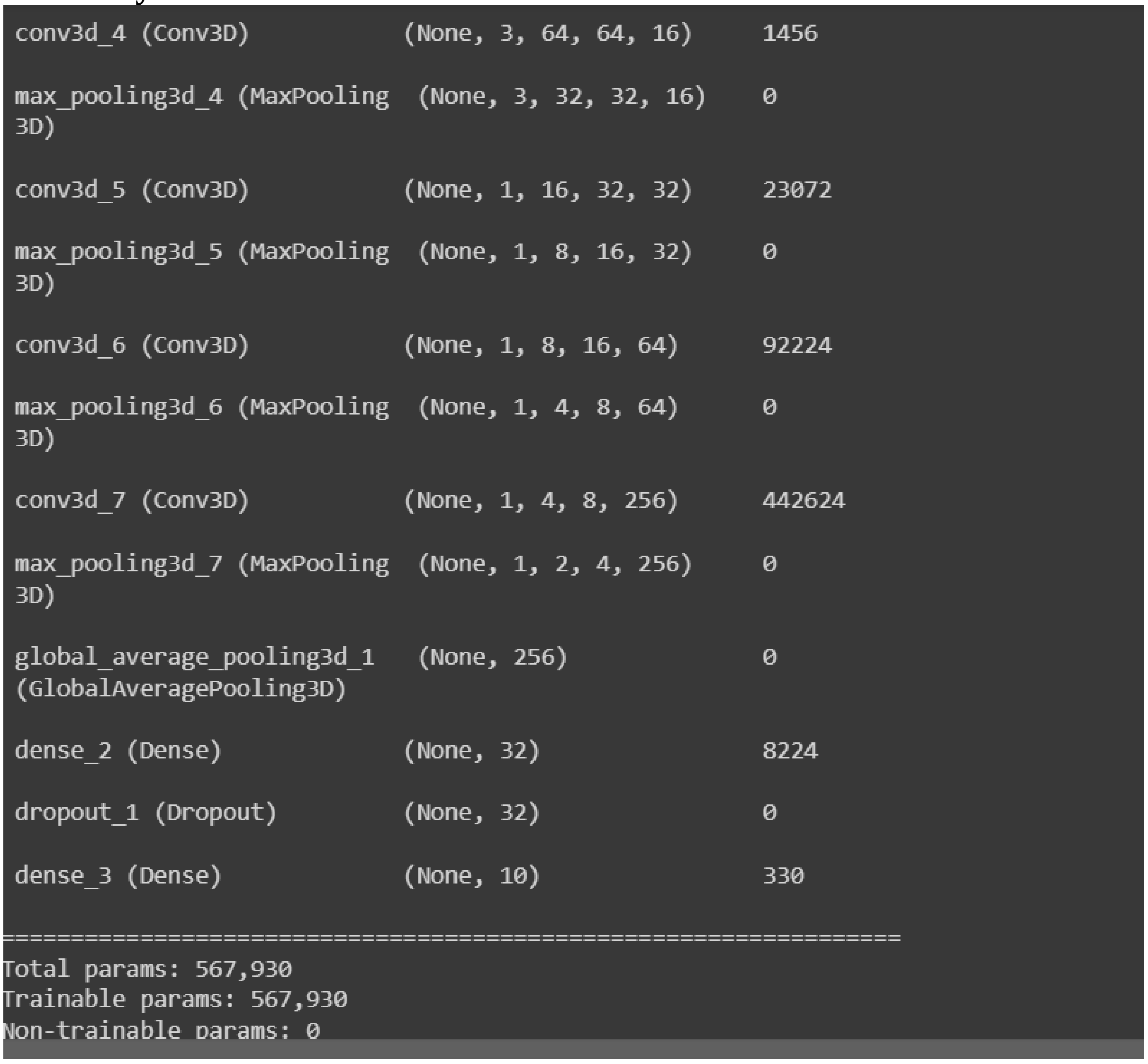
Result:



The accuracy of the above model is 30%.

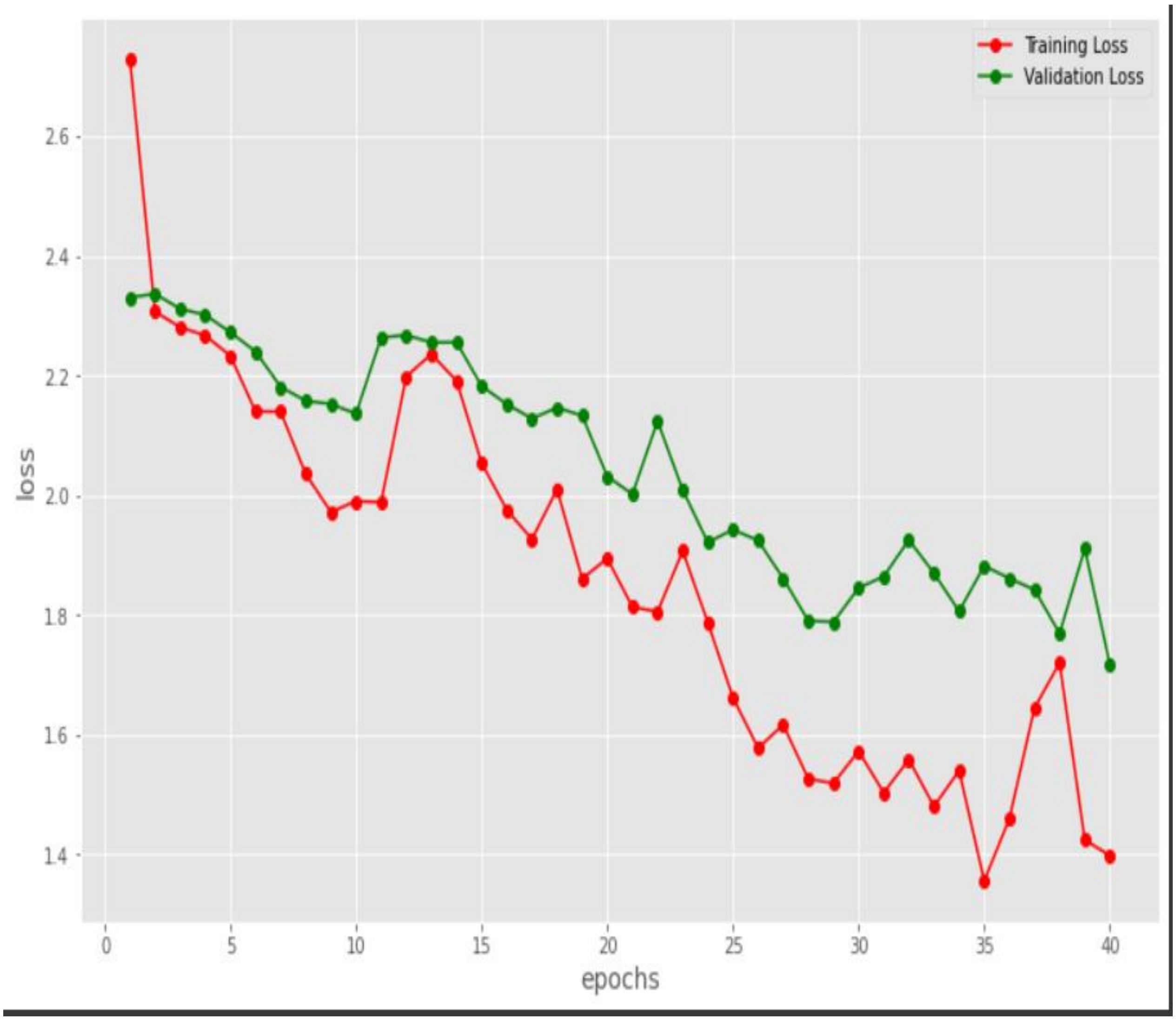
Model 3:

Summary:



```
conv3d_4 (Conv3D)           (None, 3, 64, 64, 16)     1456

max_pooling3d_4 (MaxPooling  (None, 3, 32, 32, 16)    0
3D)

conv3d_5 (Conv3D)           (None, 1, 16, 32, 32)     23072

max_pooling3d_5 (MaxPooling  (None, 1, 8, 16, 32)     0
3D)

conv3d_6 (Conv3D)           (None, 1, 8, 16, 64)      92224

max_pooling3d_6 (MaxPooling  (None, 1, 4, 8, 64)      0
3D)

conv3d_7 (Conv3D)           (None, 1, 4, 8, 256)      442624

max_pooling3d_7 (MaxPooling  (None, 1, 2, 4, 256)     0
3D)

global_average_pooling3d_1   (None, 256)              0
(GlobalAveragePooling3D)

dense_2 (Dense)             (None, 32)                8224

dropout_1 (Dropout)         (None, 32)                0

dense_3 (Dense)             (None, 10)                330

=================================================================
Total params: 567,930
Trainable params: 567,930
Non-trainable params: 0
```

Result:


Training Loss
Validation Loss
2.6
2.4
2.2
2.0
1.8
1.6
1.4
loss
0
5
10
15
20
25
30
35
40
epochs

Tool used:

OpenCV: OpenCV is a cross-platform library that can be used to create real-time neural network models. It focuses on image processing, video recording, and analysis, with capabilities like face and object detection.

Colab:

Colab is a cloud-based Jupyter notebook environment that is completely free. Most significantly, it does not require any setup, and the notebooks you create can be changed concurrently by your team members, much like documents in Google Docs. Many common machine learning libraries are supported by Colab and can be quickly loaded into your notebook.

CNN:

CNN is a powerful imaginative, artificial intelligence (AI) system that uses deep learning to execute productive and descriptive tasks, frequently combining picture and video identification, as well as programmers to promote and process natural language.

CNN employs a multi-layer perceptron technology optimized for low processing requirements. CNN layers include an input layer, an output layer, and a hidden layer with several dynamic layers, integration layers, completely integrated layers, and standard layers. The removal of constraints and increased image processing efficiency resulted in a highly efficient, user-friendly training system that restricts image processing and natural language processing.

Application:

I This method can be used for video surveillance.
ii) It can be used to monitor patients.

CONCLUSION AND FUTURE SCOPE

Conclusion

We proposed a new method for extracting features for human activity recognition in this study. To examine multichannel time series data, the suggested method constructs a new deep architecture for the CNN. To capture the distinctive patterns of the human silhouette at multiple time scales, this deep architecture primarily employs convolution and pooling techniques. All found noteworthy patterns are rigorously harmonized across several channels before being mapped into various kinds of human activities. The following are the primary benefits of the proposed method:

I feature extraction is done in a task-dependent and non-handcrafted manner;

ii) extracted features have more discriminative strength in terms of human activity classifications;

iii) Feature extraction and classification are combined in a single model to improve their mutual performance.

iv) With a small adjustment, we can reduce the feature's noise to a minimum. We show in the trials that the suggested CNN technique outperforms other state-of-the-art methods, and we believe that the proposed method can serve as a competitive tool for feature learning and classification for HAR situations.

Future Scope

More and more algorithms: For this problem, only eight machine learning algorithms were tested; try other line approaches and possibly more indirect and integral methods.

Tuning the algorithm: We can experiment with different layer combinations, different types of models, and adjusting the varied form of input with varying number of epochs or kernel size.

Data scalability: The data has already been scaled to [0,1], possibly per participant. Investigate whether additional scaling, such as standardization, can improve performance, particularly on algorithms sensitive to scaling, such as kNN.

REFERENCES

[1] A survey of video datasets for human action and activity recognition
Enrique J. Carmona, Jose M. Chaquet, and Antonio Fernández-Caballero

[2] Actions as Space-Time Shapes
Ronen Basri, Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Lena Gorelick

[3] Extraction of Features from Video File Using Different Image Algebraic Point Operations.
Nachamai M, Pranti Dutta

[4] Automatic Motion Tracking of Human in a Surveillance Video
Murtaza A. Khan, Mohammad A. AlGhamdi, and Sultan H. AL Mutairi

[5] A Novel Approach for Human Silhouette Extraction from Video Data
Debotosh Bhattacharjee, Amlan Raychaudhuri, Satyabrata Maity, Amlan Chakrabarti, and Amlan Raychaudhuri

[6] Video-Based Human Activity Recognition for Elderly Using Convolutional Neural Network
Ponniamma M., Vijayaprabakaran K., Sathiyamurthy K.

[7] Human Activity Recognition Based on Silhouette Directionality
Meghna Singh, IEEE Student Member, Anup Basu, IEEE Senior Member, and Mrinal Kr. Mandal, IEEE Senior Member

Anguita, D. [et al.]. A public domain dataset for human activity recognition using smartphones. A: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. "Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning". Bruges: 2013, p. 437-442.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for AI. Found. Trends Mach. Learn., 2(1):1–127, January 2009.

[Bulling et al., 2014] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys., 46(3):33:1–33:33, 2014.

[Cao et al., 2012] Hong Cao, Minh Nhut Nguyen, Clifton Phua, Shonali Krishnaswamy, and Xiao Li Li. An integrated framework for human activity classific. In ACM International Conference on Ubiquitous Computing, 2012.

[Deng et al., 2013] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael

Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, Yifan Gong, and Alex Acero. Recent advances in deep learning for speech research at microsoft. ICASSP, 2013.

[Deng, 2014] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing, 2014.

[Donahue et al., 2014] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In ICML, 2014.

[Fukushima, 1980] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4):193–202, 1980.

[Hinton and Osindero, 2006] Geoffrey E. Hinton and Simon Osindero. A fast learning algorithm for deep belief nets. Neural Computation, 18(7):1527–1554, 2006.

[Huynh and Schiele, 2005] Tam Huynh and Bernt Schiele. ˆ Analyzing features for activity recognition. In Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies, 2005.

[Ji et al., 2010] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. In ICML, 2010.

[Jia et al., 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In ACM MM, 2014.

[Keogh and Kasetty, 2002] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In SIGKDD, 2002.

```
# -*- coding: utf-8 -*-
"""Copy of Major_Project.ipynb

Automatically generated by Colaboratory.

Original file is located at
   https://colab.research.google.com/drive/1pko34l1MA9l-
iG_snoFKa81t2gmjrxem
"""

import cv2
import os
import statistics
import matplotlib.pyplot as plt

from google.colab.patches import cv2_imshow



import numpy as np
from scipy import stats
from itertools import product

import csv

def getdata():

cap=cv2.VideoCapture("/content/drive/MyDrive/datasets/walk/eli_wa
lk.avi")
  v=[]
  frame_width = int(cap.get(3))
  frame_height = int(cap.get(4))
  size = (frame_width, frame_height)
  kernel=np.array([[0,-1,0],[-1,5,-1],[0,-1,0]])

res=cv2.VideoWriter('/content/drive/MyDrive/vieodata/filename1.avi'
, cv2.VideoWriter_fourcc(*'XVID'),10, size,0)
  while (cap.isOpened(),):
    ret,frame=cap.read()
    if ret==True:
      frame=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
      frame=cv2.filter2D(src=frame,ddepth=-1,kernel=kernel)
      res.write(frame)
      cv2_imshow(frame)
    else:
      break
  cap.release()
  cv2.destroyAllWindows()
  return
getdata()

def getbackgrounddata(file):

cap=cv2.VideoCapture("/content/drive/MyDrive/datasets/background
s/"+file)
  a=0
  while (cap.isOpened()):
    ret,frame=cap.read()
    frame = cv2.medianBlur(frame, 3)
    if ret==True and a==3:
      return frame
    elif ret==True and a!=3:
      a+=1
    else:
      break
  cap.release()
  cv2.destroyAllWindows()
  return

# background=getbackgrounddata()
# cv2_imshow(background)
# background=cv2.cvtColor(background,cv2.COLOR_BGR2GRAY)
# cv2_imshow(background)
```

```
def getdata1(fl,fn,back):

cap=cv2.VideoCapture("/content/drive/MyDrive/datasets/"+fl+"/"+fn)
  v=0
  t=0
  kernel=np.array([[1,1,1],[1,1,1],[1,1,1]])

background=cv2.cvtColor(getbackgrounddata(back),cv2.COLOR_BG
R2GRAY)
  frame_width = int(cap.get(3))
  frame_height = int(cap.get(4))
  size = (frame_width, frame_height)
  res=cv2.VideoWriter('/content/drive/MyDrive/vdata/'+fl+'/'+fn,
cv2.VideoWriter_fourcc(*'XVID'),10, size,0)
  re=cv2.VideoWriter('/content/drive/MyDrive/vdata/'+fl+'/'+'0'+fn,
cv2.VideoWriter_fourcc(*'XVID'),10, size,0)
  while (cap.isOpened(),):
    ret,frame=cap.read()
    if ret==True:
      frame=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
      print("After Converting on Gray Scale")
      cv2_imshow(frame)
      fr=frame.copy()
      ed=cv2.Canny(fr,100,200)
      print("After Edge Detection")
      cv2_imshow(ed)
      #std=np.std(frame)
      #print(std)
      #print("After Implementing Noise")
      fr = cv2.medianBlur(fr, 3)
      #std=(np.std(fr))
      #cv2_imshow(fr)
      f=[]
      foreground=[]
      for i in range(len(frame)):
        r=[]
        for j in range(len(frame[0])):
          r.append(abs(frame[i][j]-background[i][j]))
        foreground.append(r)
      foreground=np.reshape(foreground,(len(frame),len(frame[0])))
      foreground=foreground.astype("uint8")
      print("After Background Subtraction")
      cv2_imshow(foreground)
      std=np.std(foreground)
      for i in range(len(frame)):
        r=[]
        for j in range(len(frame[0])):
          if foreground[i][j]>0:
            if foreground[i][j]>std:
              r.append(255)
            else:
              r.append(0)
          else:
            r.append(0)
        f.append(r)
      f=np.reshape(f,(len(frame),len(frame[0])))
      f=f.astype("uint8")
      print("After Foreground Detection")
      cv2_imshow(f)
      a=f+ed
      erosion = cv2.erode(a,kernel,iterations=2)
      print("Eroded")
      cv2_imshow(erosion)
      if v < 14:
        res.write(erosion)
        v+=1
      else:
        if v<28:
          re.write(erosion)
          v+=1
        else:
          break
    else:
      break
  cap.release()
  cv2.destroyAllWindows()
getdata1("walk","denis_walk.avi","bg_038.avi")

arr = os.listdir("/content/drive/MyDrive/datasets")
print(arr)

print(arr)
arr.remove("IndividualDetails.csv")
arr.remove("Indian States Population and Area.xlsx")
arr.remove("covid_19_india.csv")
arr.remove("classification_masks.mat")
arr.remove("backgrounds")
print(arr)

for i in arr:
  file_arr=os.listdir("/content/drive/MyDrive/datasets/"+i)
  bg=['']
  for j in file_arr:
    if j=="daria_bend.avi" or j=="daria_jack.avi" or
j=="daria_jump.avi" or j=="daria_pjump.avi" or j=="daria_run.avi" or
j=="daria_side.avi" or j=="daria_skip.avi" or j=="daria_walk.avi" or
j=="daria_wave1.avi" or j=="daria_wave2.avi":
      bg[0]="bg_026.avi"
    elif j=="denis_bend.avi" or j=="denis_jack.avi" or
j=="denis_jump.avi" or j=="denis_pjump.avi" or j=="denis_run.avi"
or j=="denis_side.avi" or j=="denis_skip.avi" or j=="denis_walk.avi"
or j=="denis_wave1.avi" or j=="denis_wave2.avi":
      bg[0]="bg_038.avi"
    elif j=="eli_bend.avi" or j=="eli_jack.avi" or j=="eli_jump.avi" or
j=="eli_pjump.avi" or j=="eli_run.avi" or j=="eli_side.avi" or
j=="eli_skip.avi" or j=="eli_walk.avi" or j=="eli_wave1.avi" or
j=="eli_wave2.avi":
      bg[0]="bg_062.avi"
    elif j=="ido_bend.avi" or j=="ido_jack.avi" or j=="ido_jump.avi" or
j=="ido_pjump.avi" or j=="ido_run.avi" or j=="ido_side.avi" or
```

```
j=="ido_skip.avi" or j=="ido_walk.avi" or j=="ido_wave1.avi" or j=="ido_wave2.avi":
      bg[0]="bg_062.avi"
    elif j=="ira_bend.avi" or j=="ira_jack.avi" or j=="ira_jump.avi" or j=="ira_pjump.avi" or j=="ira_run.avi" or j=="ira_side.avi" or j=="ira_skip.avi" or j=="ira_walk.avi" or j=="ira_wave1.avi" or j=="ira_wave2.avi":
      bg[0]="bg_062.avi"
    elif j=="shahar_bend.avi" or j=="shahar_jack.avi" or j=="shahar_jump.avi" or j=="shahar_pjump.avi" or j=="shahar_side.avi" or j=="shahar_skip.avi" or j=="shahar_walk.avi" or j=="shahar_wave1.avi" or j=="shahar_wave2.avi":
      bg[0]="bg_062.avi"
    elif j=="lena_jack.avi":
      bg[0]="lena_bg_jack.avi"
    elif j=="moshe_run.avi":
      bg[0]="moshe_bg_run.avi"
    elif j=="shahar_run.avi":
      bg[0]="shahar_bg_run.avi"
    else:
      bg[0]="bg_038.avi"
    #print(i,j)
    getdata1(i,j,bg[0])

import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler
plt.style.use('ggplot')

from sklearn.datasets import load_files
import os

from sklearn.model_selection import train_test_split
```

```
print(os.getcwd())
raw_data=load_files(os.getcwd()+
r'/drive/MyDrive/vdata',shuffle=True)
target=raw_data['target']

len(raw_data)

files = raw_data['filenames']
train_files, test_files, train_targets, test_targets = train_test_split(files,
target, test_size=60, random_state=196)

len(train_files)

valid_files = train_files[90:]
valid_targets = train_targets[90:]

train_files = train_files[:90]
train_targets = train_targets[:90]

print(raw_data['target_names'])

for label in zip(range(10), raw_data['target_names']):
    print(label)

for pair in zip(train_files[:5], train_targets[:5]):
    print(pair)

pip install sk-video

import numpy as np
from skvideo.io import FFmpegReader, ffprobe
from skvideo.utils import rgb2gray
from PIL import Image
from tqdm import tqdm
from keras.preprocessing import image

def _read_video(path):
  cap=FFmpegReader(filename=path)
  list_of_frames=[]
  fps=int(cap.inputfps)
  target_size=(128,128)
  required_fps=fps
  for index,frame in enumerate(cap.nextFrame()):
    capture_frame=True
    if required_fps !=None:
      is_valid=range(required_fps)
      capture_frame=(index % fps) in is_valid
    if capture_frame:
      temp_image=image.array_to_img(frame)

frame=image.img_to_array(temp_image.resize(target_size,Image.AN
TIALIAS)).astype('uint8')
      list_of_frames.append(frame)
  temp_video=np.stack(list_of_frames)
  cap.close()
  temp_video=rgb2gray(temp_video)
  return np.expand_dims(temp_video, axis=0)

def read_videos(paths):
  normalise_pixel=(0,1)
  list_of_videos=[_read_video(path) for path in tqdm(paths)]
  tensor=np.vstack(list_of_videos)
  if len(normalise_pixel)==2 and len(normalise_pixel)== tuple:
    base=normalise_pixel[0]
    r=normalise_pixel[1]-base
    mini=np.min(tensor,axis=(1,2,3),keepdims=True)
    maxi=np.max(tensor,axis=(1,2,3),keepdims=True)
    return ((tensor.astype('float32')-mini)/(maxi-mini))*r+base
  return tensor

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
```

```
x_train=read_videos(train_files)
y_train=to_categorical(train_targets,num_classes=10)
print("Shape of Training data: ", x_train.shape)
print("Shape of Training Targets: ", y_train.shape)

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import matplotlib.pyplot as plt
import skvideo.io
# %matplotlib inline


# The path of a sample video in the training data
sample_files = train_files[:1]

# An object of the class 'Videos'
sample = skvideo.io.vread(sample_files[0]);

print('\nShape of the sample data:', sample.shape)

# Displaying a frame from the sample video
plt.imshow(sample[10])

x_valid=read_videos(valid_files)
y_valid=to_categorical(valid_targets,num_classes=10)
print("Shape of validation data:",x_valid.shape)
print("Shape of validation Labels:",y_valid.shape)

x_test = read_videos(test_files)
y_test = to_categorical(test_targets, num_classes=10)
print('Shape of testing data:', x_test.shape)
print('Shape of testing labels:', y_test.shape)

y_train.shape[1:]

from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from keras.models import Sequential
from keras.layers import Dense

from sklearn.metrics import f1_score
from sklearn.model_selection import cross_validate
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score

from keras.models import Sequential
from keras.layers import Conv3D, MaxPooling3D,
GlobalAveragePooling3D
from keras.layers.core import Dense, Dropout

from keras.models import Sequential
from keras.layers import Conv3D, MaxPooling3D,
GlobalAveragePooling3D
from keras.layers.core import Dense, Dropout
model = Sequential()
model.add(Conv3D(filters=16, kernel_size=(10, 3, 3), strides=(5, 2, 2),
padding='same', activation='relu',input_shape=(14,128,128,1)))
model.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model.add(Conv3D(filters=64, kernel_size=(5, 3, 3), strides=(3, 1, 1),
padding='same', activation='relu'))
model.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model.add(Conv3D(filters=256, kernel_size=(3, 3, 3), strides=(3, 1, 1),
padding='same', activation='relu'))
```

```
model.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model.add(GlobalAveragePooling3D())
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(10,activation="softmax"))
model.summary()

from keras.callbacks import ModelCheckpoint
from keras import optimizers

model.compile(loss='categorical_crossentropy', optimizer="adam",
metrics=['accuracy'])
checkpoint = ModelCheckpoint(filepath='Model.weights.best.hdf5',
save_best_only=True, verbose=1)
hist = model.fit(x_train, y_train, batch_size=10, epochs=40,
validation_data=(x_valid, y_valid),verbose=2,callbacks=[checkpoint])
print(hist)

model.load_weights('Model.weights.best.hdf5')
(loss,accuracy)=model.evaluate(x_test,y_test,batch_size=10,verbose=
0)
print("Accuracy: ",accuracy*100)

plt.figure(figsize=(12, 8))

loss = hist.history['loss']                    # Loss on the training data
val_loss = hist.history['val_loss']                 # Loss on the validation
data
epochs = range(1, 41)

plt.plot(epochs, loss, 'ro-', label='Training Loss')
plt.plot(epochs, val_loss, 'go-', label = 'Validation Loss')
plt.xlabel('epochs', fontsize=14)
plt.ylabel('loss', fontsize=14)
plt.legend()
```

```
model2 = Sequential()
model2.add(Conv3D(filters=16, kernel_size=(10, 3, 3), strides=(5, 2,
2), padding='same', activation='relu',input_shape=(29,128,128,1)))
model2.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model2.add(Conv3D(filters=64, kernel_size=(5, 3, 3), strides=(3, 1, 1),
padding='same', activation='relu'))
model2.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model2.add(Conv3D(filters=256, kernel_size=(3, 3, 3), strides=(3, 1,
1), padding='same', activation='relu'))
model2.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model2.add(GlobalAveragePooling3D())
model2.add(Dense(32, activation='relu'))
#model2.add(Dense(16, activation='relu'))
model2.add(Dropout(0.5))
model2.add(Dense(10,activation="softmax"))
model2.summary()

model2.compile(loss='categorical_crossentropy', optimizer="adam",
metrics=['accuracy'])
checkpoint2 = ModelCheckpoint(filepath='Model2.weights.best.hdf5',
save_best_only=True, verbose=1)
hist = model2.fit(x_train, y_train, batch_size=15, epochs=40,
validation_data=(x_valid,
y_valid),verbose=2,callbacks=[checkpoint2])
print(hist)

model2.load_weights('Model2.weights.best.hdf5')
(loss2,accuracy2)=model2.evaluate(x_test,y_test,batch_size=15,verbo
se=0)
print("Accuracy: ",accuracy2*100)

plt.figure(figsize=(12, 8))
```

```
loss2 = hist.history['loss']                    # Loss on the training data
val_loss2 = hist.history['val_loss']                # Loss on the validation
data
epochs = range(1, 26)

plt.plot(epochs, loss2, 'ro-', label='Training Loss')
plt.plot(epochs, val_loss2, 'go-', label = 'Validation Loss')
plt.xlabel('epochs', fontsize=14)
plt.ylabel('loss', fontsize=14)
plt.legend()

model3 = Sequential()
model3.add(Conv3D(filters=16, kernel_size=(10, 3, 3), strides=(5, 2,
2), padding='same', activation='relu',input_shape=(29,128,128,1)))
model3.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model3.add(Conv3D(filters=64, kernel_size=(5, 3, 3), strides=(3, 1, 1),
padding='same', activation='relu'))
model3.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model3.add(Conv3D(filters=256, kernel_size=(3, 3, 3), strides=(3, 1,
1), padding='same', activation='relu'))
model3.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model3.add(GlobalAveragePooling3D())
model3.add(Dense(64, activation='relu'))
model3.add(Dense(32, activation='relu'))
#model3.add(Dense(16, activation='relu'))
model3.add(Dropout(0.5))
model3.add(Dense(10,activation="softmax"))
model3.summary()

model3.compile(loss='categorical_crossentropy', optimizer="adam",
metrics=['accuracy'])
checkpoint2 = ModelCheckpoint(filepath='Model3.weights.best.hdf5',
save_best_only=True, verbose=1)
```

```
hist = model3.fit(x_train, y_train, batch_size=10, epochs=25, validation_data=(x_valid,
y_valid),verbose=2,callbacks=[checkpoint2])
print(hist)

model3.load_weights('Model3.weights.best.hdf5')
(loss2,accuracy3)=model3.evaluate(x_test,y_test,batch_size=10,verbo
se=0)
print("Accuracy: ",accuracy3*100)

from google.colab import drive
drive.mount('/content/drive')




import cv2
import os
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow



model10 = Sequential()
model10.add(Conv3D(filters=16, kernel_size=(10, 3, 3), strides=(5, 2,
2), padding='same', activation='relu',input_shape=(14,128,128,1)))
model10.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model10.add(Conv3D(filters=32, kernel_size=(5, 3, 3), strides=(3, 2,
1), padding='same', activation='relu'))
model10.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
model10.add(Conv3D(filters=64, kernel_size=(5, 3, 3), strides=(3, 1,
1), padding='same', activation='relu'))
model10.add(MaxPooling3D(pool_size=2, strides=(1, 2, 2),
padding='same'))
```

```
model10.add(Conv3D(filters=256, kernel_size=(3, 3, 3), strides=(3, 1,
1), padding='same', activation='relu'))
model10.add(MaxPooling3D(pool_size=2,     strides=(1,     2,     2),
padding='same'))
model10.add(GlobalAveragePooling3D())
model10.add(Dense(32, activation='relu'))
#model2.add(Dense(16, activation='relu'))
model10.add(Dropout(0.5))
model10.add(Dense(10,activation="softmax"))
model10.summary()

model10.compile(loss='categorical_crossentropy',  optimizer="adam",
metrics=['accuracy'])
checkpoint2                                                          =
ModelCheckpoint(filepath='Model10.weights.best.hdf5',
save_best_only=True, verbose=1)
hist  =  model10.fit(x_train,  y_train,  batch_size=10,  epochs=40,
validation_data=(x_valid,
y_valid),verbose=2,callbacks=[checkpoint2])
print(hist)

model10.load_weights('Model10.weights.best.hdf5')
(loss10,accuracy10)=model10.evaluate(x_test,y_test,batch_size=10,v
erbose=0)
print("Accuracy: ",accuracy10*100)

model10.load_weights('Model10.weights.best.hdf5')
(loss10,accuracy10)=model10.evaluate(x_test,y_test,batch_size=15,v
erbose=0)
print("Accuracy: ",accuracy10*100)

plt.figure(figsize=(12, 8))

loss2 = hist.history['loss']                    # Loss on the training data
val_loss2 = hist.history['val_loss']                # Loss on the validation
data
epochs = range(1, 41)
```

```
plt.plot(epochs, loss2, 'ro-', label='Training Loss')
plt.plot(epochs, val_loss2, 'go-', label = 'Validation Loss')
plt.xlabel('epochs', fontsize=14)
plt.ylabel('loss', fontsize=14)
plt.legend()
```