RGB Video-Based Human Action Recognition through key points Estimation

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

AP21110010031	Mounish Sai Gowtham M
AP21110010041	Lakshmi Sai Prakash P
AP21110010042	Sai Gopal L
AP21110010168	Khyathi Vardhan R

Under the Guidance of

Dr. M Naveen Kumar Mahamkali

SRM University–AP

Andhra Pradesh – 522 240

Nov, 2023

Certificate

Date: 28-Nov-23

This is to certify that the work present in this Project entitled “**RGB Video-Based Human Action Recognition through key points Estimation**” has been carried out by **Mounish Sai, Sai Gopal, Lakshmi Sai Prakash, Khyathi Vardhan Ravella,** under **Dr. Naveen Kumar M** supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Dr. M Naveen Kumar Mahamkali

Designation,

Affiliation.

Acknowledgement

I would like to take this opportunity to express my sincere gratitude and appreciation to all the individuals and organizations who have contributed to the successful completion of this project report titled "**RGB Video-Based Human Action Recognition through Key Points Estimation**".

I am also thankful to the faculty mentor **Dr. M Naveen kumar** Sir for their constant encouragement and constructive feedback, which have significantly enriched the quality of this project report.

Table Of Contents

Abstract

RGB Video-Based Human Action Recognition through key points Estimation is the name of the project. As a part of it MediaPipe is used for Extracting Joint Positions and Features and Cross validation model is used for prediction of actions. Mounish, Prakash, Gopal, and Khyathi are the members of the project group, which was led by Naveen Kumar Sir. The primary goal of the project's development is to predict the action samples data extracted from the joint locations from the RGB videos that are input from the photos, videos, and webcam feed. The project also determines the accuracy scores of the ml model and the overall accuracy

List of Figures

Introduction

Recognition of human actions is currently a prominent focus within the realm of computer vision research. The significance of human action recognition from RGB videos extends across diverse fields and applications, promising to deepen our comprehension of human behavior and enhance the efficacy of various systems. In recent years, there has been a growing interest in using skeleton data, which captures the joint positions of a human body, to perform human action recognition. This is because skeleton data is less sensitive to variations in appearance and can capture the temporal dynamics of human actions.

This project aims to develop a human action recognition system using pose estimation that utilizes computer vision techniques to extract joint positions and calculate distances between joints from various input sources, including videos, images, and real-time webcam feeds, which are the main inputs for creating the human action recognition system. By integrating the OpenCV and Mediapipe libraries with Python, the system will enable accurate and real-time analysis of human body movements. Our approach is based on recent advances in deep learning and is designed to overcome the challenges associated with recognizing complex and diverse human actions.

By utilizing pose estimation and a strong machine learning model with cross-validation techniques, this study explores the intriguing field of human action recognition. Understanding the spatial arrangement of the human body during various activities begins with pose estimation, which is the process of extracting skeletal information from pictures or movies.An additional layer of intelligence is added to the process through the use of machine learning models that are specifically made for action recognition. With training on annotated datasets containing a variety of human events, these models are able to generalize and predict on data that has never been seen before. We use cross-validation, a validation technique that divides the dataset into distinct subsets, trains the model on various combinations, and assesses its efficacy across a range of scenarios, to improve the performance and dependability of our model.

In addition to pushing the boundaries of human action recognition technology, this study hopes to shed light on the ways that pose estimation and machine learning work together. As we continue to explore the technical details of our methodology, we hope to find novel approaches to the problems associated with precisely and effectively recognizing human behaviors. The results of this study have the potential to completely change the field of applications, from sophisticated virtual reality experiences to intelligent surveillance systems. The significance of this research work is to increase the performance of vision-based action recognition ml models with the data that is obtained by extracting key points from RGB video sequences. It is noted that the extracted key points form a 2D skeleton pose.

1.1 Outline of the Project:

- Setup and Environment Configuration
- Image and Video Processing
- Joint Position Estimation
- Distance Calculation
- Angle Calculation
- Real-Time Pose Estimation
- Feature extraction and Feature matrix
- Training and Testing of ML model
- Output

2. Main Text:

2.1 Method of Treatment:

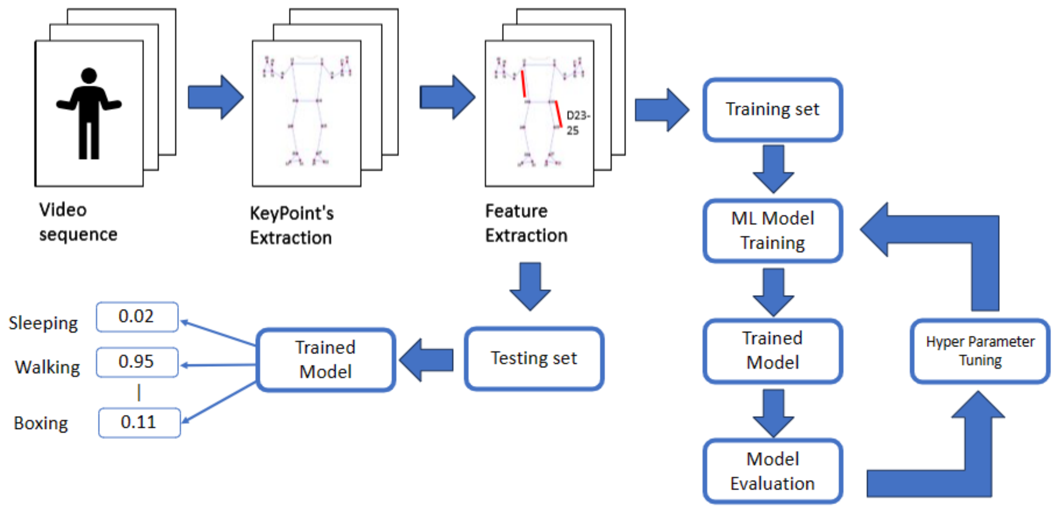


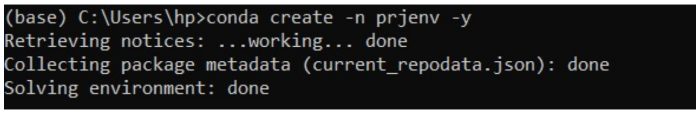
Fig 1. proposed scheme

The proposed approach includes gathering the video sequence , key points extraction and finally feature extraction. which will be explained in a detailed way in the upcoming sections.

2.2 Methodology

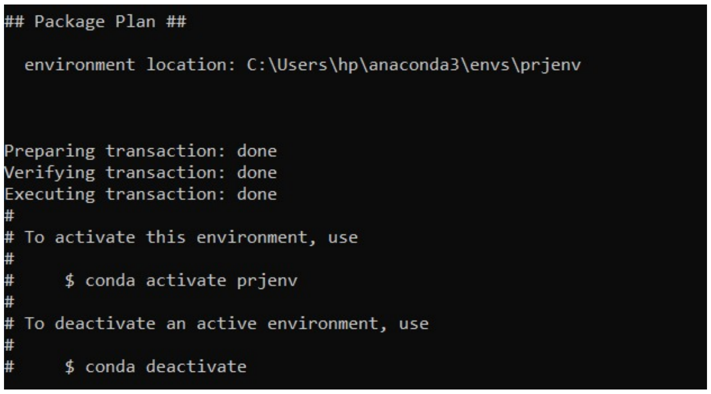
2.2.1 Environmental Setup:

- Initially Anaconda software should be installed on the local system.
- A new environment should be created on the anaconda prompt.



```
(base) C:\Users\hp>conda create -n prjenv -y
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

fig.2 creating a new environment



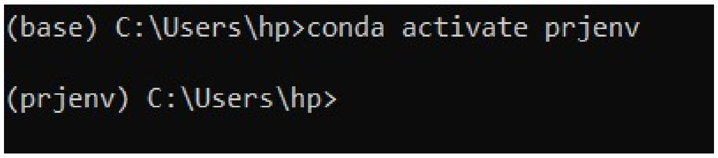
```
## Package Plan ##

  environment location: C:\Users\hp\anaconda3\envs\prjenv



Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate prjenv
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

fig. 3 Activating the environment



```
(base) C:\Users\hp>conda activate prjenv

(prjenv) C:\Users\hp>
```

fig. 4 New environment has been created

- Import the libraries that are necessary for execution of the code.

2.2.2 Libraries/Packages:

- **Pose Estimation Algorithm :**

MediaPipe pose is the Pose Estimation algorithm[1] that has been used in this project. As discussed in [2], It is a part of the MediaPipe library, developed by Google and an open-source framework that provides real-time pose estimation solutions. MediaPipe Pose is designed to detect and track human body key points (joints) in images and videos, enabling applications such as human action recognition, gesture recognition, and more. The library is implemented in C++ and offers APIs for various programming languages, including Python.

Key features of MediaPipe Pose:

- The primary goal of MediaPipe Pose is to detect and track key body points in the human body, such as the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles.
- This pose estimation model, used by MediaPipe Pose is based on a deep learning neural network, which enables it to perform on various devices, including CPUs, GPUs, and mobile devices.
- It is able to handle multi-person scenarios, that is it can detect and track the poses of multiple individuals in a single image or frame.
- MediaPipe Pose supports multiple platforms, making it accessible for both desktop and mobile applications.
- It is an open-source project, which means developers can access the source code, modify it, or contribute to the library's improvement.

- Complex multimodal applications can be built by combining MediaPipe pose with other Mediapipe solutions such as MediaPipe Hands, MediaPipe Holistic, etc.,

command line: pip install mediapipe

- OpenCV :

It is the well known library that is used to process the images and perform computer vision tasks, such as face detection, object detection, and much more. it can also be used for modifying the videos.

command line: pip install opencv-python

- Argparse :

Argparse is a python module that is used to parse the command line arguments.

- Math:

Math is a python library that is used to perform mathematical operations.

- Time:

Time is a python library that is used to perform various functions to work with time-related tasks such as retrieving the current time

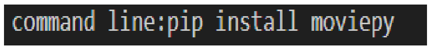
- Numpy:

It is a python library which comes handy while working with arrays.

command line: pip install numpy

- Moviepy:

  Moviepy is a python library for video editing that provides a convenient and accessible interface for basic video manipulation.



```
command line:pip install moviepy
```

<u>To evaluate our proposed methodology</u>

- We are going to use the UT Kinect dataset.
- We are creating our own dataset of recording our own videos.

2.3 Working:

In this section, we will cover the detailed steps of how the project works, from getting the
inputs to creating a feature vector that encapsulates the location information of the video.

2.3.1 Input files:

- The project starts by taking video files or webcam channel as input. OpenCV, an open source computer vision library, is used for this.
- Video files and webcam streams are basically a sequence of frames, where each frame is a 2D matrix of pixels representing the colors of the image.
- The pixel matrix, often called image matrix or pixel board, is the basic information
  structure used to represent the colors of digital images.
- This is a two-dimensional grid of pixels, where each pixel corresponds to a small point image and stores information about its color.

- The pixel matrix is necessary to understand and process the colors of the image.
- Each pixel contains color information which is a combination of red, green and blue
  components (RGB).
- The RGB color model is the most common color representation used in digital images.
- It uses three main color channels: red, green and blue.
- Colors are created by varying the intensity of these three channels.
- The pixel matrix is arranged in a grid with rows and columns.
- Each cell of the matrix corresponds to one pixel.
- The dimensions of the matrix are determined by the image resolution, e.g. 1920x1080 Full HD picture.
- Each pixel of the matrix stores color information in the form of three values: intensity of red (R), green (G) and blue (B).
- In general, each channel is represented by 8 bits, allowing 256 intensity levels (0 up to
  255). This is often referred to as 8-bit color depth.
- Image colors are obtained by combining different intensities R, G and B.
- Each color space has its own way of representing colors in the pixel matrix.

2.3.2 Conversion to RGB format:

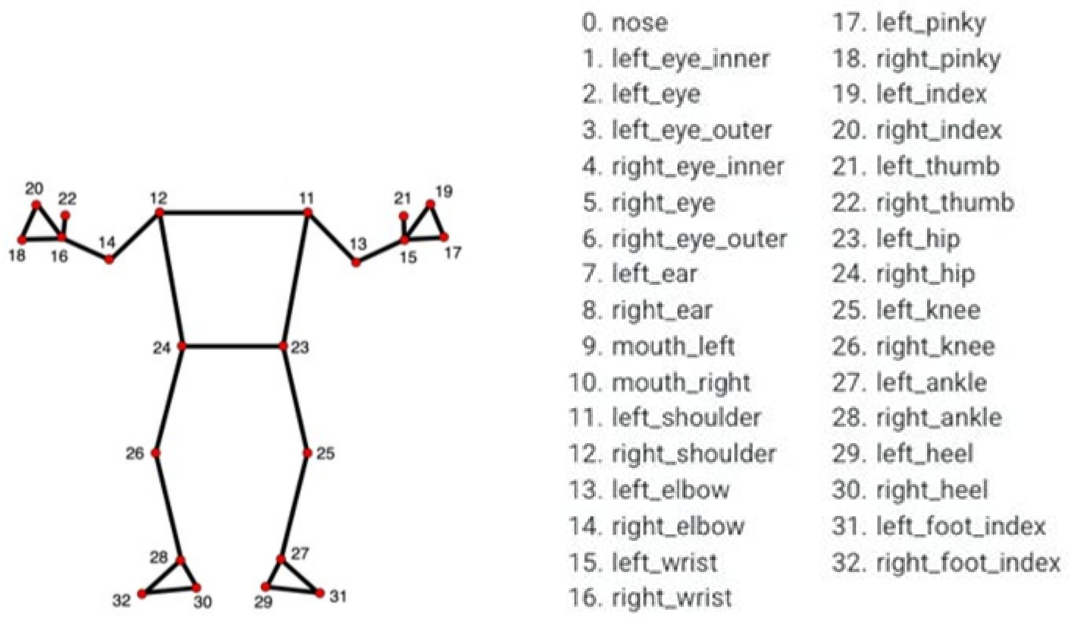
Mediapipe Pose works with images in RGB (red, green, blue) format, while OpenCV reads
video files in BGR (blue, green, red) format by default. Input frames must be converted
from BGR to RGB to be compatible with Mediapipe Pose. This conversion is done using

OpenCV's cvtColor () function with the COLOR_BGR2RGB flag. This simple operation switches the red and blue color channels.

- From [3], The cvtColor () function is an important function provided by the OpenCV library (OpenSource Computer Vision Library) for color space conversion in image processing.
- It allows you to convert an image from one color space to another. A commonly used conversion is from BGR color space to RGB color space.
- This is a key feature when working with digital images, as different algorithms and
  libraries may expect or produce images in different color spaces.
- COLOR_BGR2RGB is one of the many predefined conversion codes (flags) of OpenCV. This specifies that the conversion should be from the BGR color space to the RGB color space.
- In the BGR color space, pixel values are arranged as blue, green, red. Instead, in the RGB color space, pixel values are arranged as red, green, blue.
- This flag basically rearranges the color channels, swapping red and blue to convert the
  image from BGR to RGB.

2.3.3 Identification of landmarks:

Mediapipe Pose is then used to detect human body landmarks or joints in each frame of the video. These landmarks correspond to certain points on the human body, such as shoulders, elbows, wrists, hips, knees and ankles. Facial landmarks (normal features 0-10) are not considered in this project.



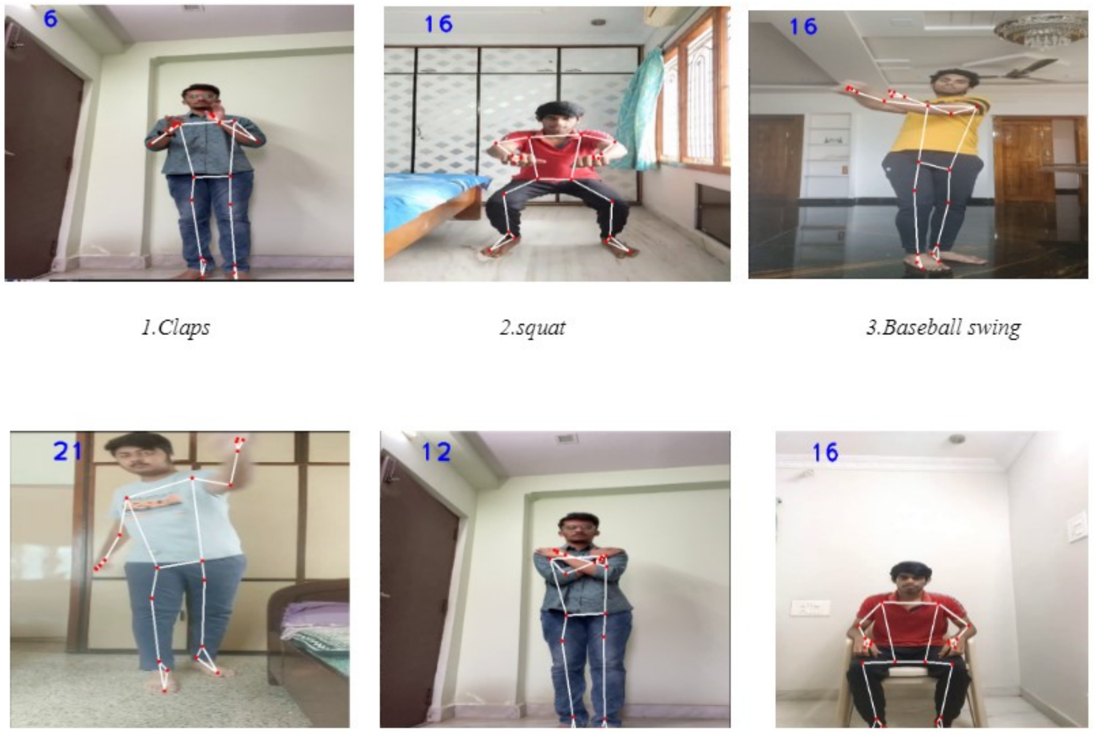
0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig 5. skeleton structure representation

In this project the evaluation process consists of two fundamental features, Distances and angles[4]. Both Distances and angle measurements play a vital role. We explain the significance and application of these features in a detailed manner in the upcoming sections.

2.3.4 Drawing connections:

Using the detected landmarks, connections are made between certain pairs of joints and the skeletal structure of the human body is effectively created in each frame. This step helps to visualize the results of the pose estimation in the video frames, which makes the pose easier to understand and interpret.



1.Claps *2.squat* *3.Baseball swing*

4.Throw *5.Arm Cross* *6.Sit to Stand*

Fig 6. overlaid landmarks representation

2.3.5 Distance matrix from the coordinate points:

The coordinates (x, y) of each connection point are extracted from the observed landmarks. These coordinates represent the spatial location of each joint in the frame and provide precise information about the location of the person shown in the video.

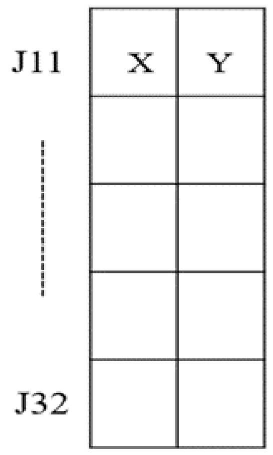


Fig 7. Image representation of coordinates

2.3.6 Angle matrix from coordinate points:

Angle features are measurements of the angle between different joints in the body. They can be used to track the movement of the body and to identify different individuals' actions.
To extract angle features from the dataset, the following steps are typically followed:

The skeleton of the subject is extracted from each frame.
The angle between each pair of joints in the skeleton is calculated.
The angle features are then extracted from the sequence of frames.

2.3.7 Distance calculation:

For each frame of the video, the distance between all connection pairs is calculated. The distance between two joints (x1, y1) and (x2, y2) is calculated using the Euclidean distance formula: "distance = sqrt ((x2 - x1)^2 (y2 - y1)^2)". This step creates a matrix where each element represents the distance between two joints of the given frame.

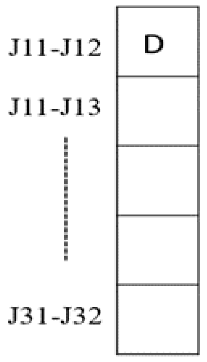


Fig 8. Image representation of the distance of a single frame

2.3.8 Structure of the feature matrix:

All distance matrices obtained for each frame are combined into a single matrix. The size of this matrix is 231 x M, where M is the number of video frames. The number 231 means the total number of unique common combinations (21 joints and 2 combinations per common pair, no diagonal).

2.3.9 Feature vector:

To obtain a more compact representation of all pose information in the video, the feature matrix is transformed into a 1D matrix known as a feature vector. This transformation is achieved by concatenating the rows of the matrix into a single vector. The resulting feature vector as mentioned in [5] is a short but comprehensive summary of the pose information of the video, making it suitable for further analysis and processing.

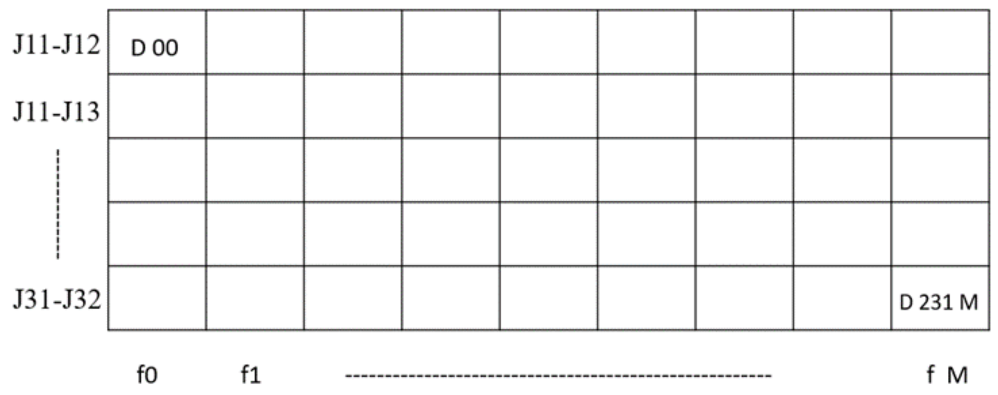


Fig9. Image representation of feature matrix

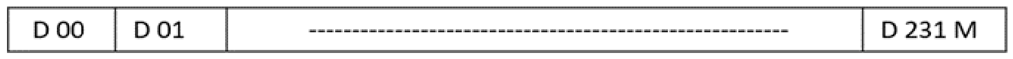
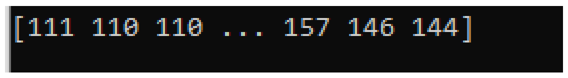


Fig10. Image representation of feature vector.

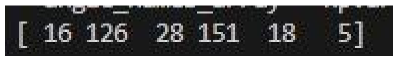
The Figure 11 represents the outcome where the distances between all the joints of every frame is stored in a 1D array which is a feature vector.



```
[111 110 110 ... 157 146 144]
```

Fig 11. feature vector

To extract angle features from the dataset, you can follow the same steps as described for the distance feature. However, you may need to modify the feature matrix depending on the size and format of the data and the specific angle features that are required to extract.



```
[ 16 126  28 151  18   5]
```

Fig 12. Angle vector

Feature Matrix

As mentioned in [6], Feature Matrix, a commonly used term in machine learning and data science which refers to a matrix where each row represents an individual observation, which in our projects represents the feature vector of each action sample. The feature matrix is a fundamental concept in representation of datasets for machine learning algorithms. If m is the rows representing an individual observation and n is the columns representing features of that observation then the Feature matrix having the dimensions m x n consists of elements where Each element X ij in the matrix represents the value of the j-th feature for the i-th instance.

As mentioned in [7], An index matrix, in the context of machine learning, is a matrix that stores indices or identifiers for each instance. Each row of the index matrix corresponds to a row in the feature matrix, and it may contain unique identifiers or indices for those instances.

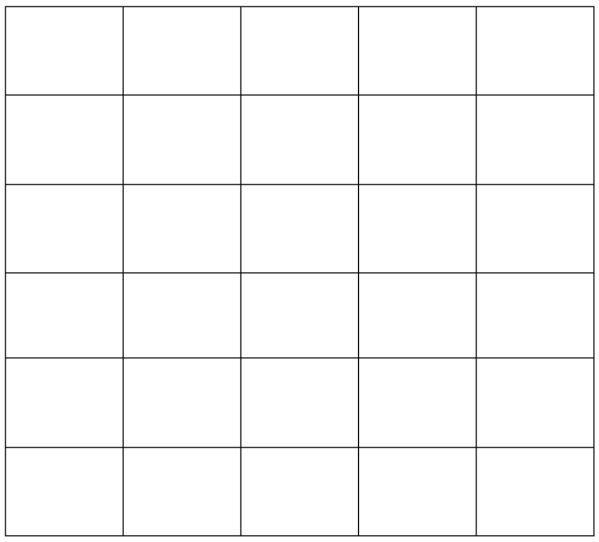


Fig.13 Feature Matrix

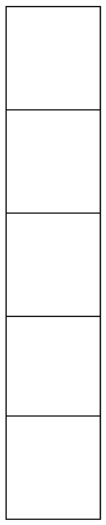


Fig.14 Index Matrix

The feature vector of one frame in the action sample consists of a feature vector of 231 units data, now this feature vector in a single row consists of data of 40 such frames(as we considered 40 fixed number of frames using the minimum frame count). That is the feature vector of one action sample consists of 9240 units of data. All these feature vectors of the action samples are concatenated in a vertical fashion to form the feature matrix, which acts as the input for the machine learning model. An Index array which consists of the indexes of the respected action samples which serves as a mapping between the feature matrix and the corresponding actions is used along with the feature matrix to help the machine learning model identify the actions samples with ease.

Machine learning Model

As mentioned in [8], A machine learning model is a mathematical representation or algorithm that is trained on data to make predictions or decisions. In other words, a machine learning model learns patterns and relationships from data, and once trained, it can be used to make predictions or classifications on new, unseen data.

The process of creating a machine learning model involves two main steps: training and testing.
Figure 3 represents the working of the machine learning model. Hyper parameter tuning is the process of determining which collection of hyperparameters is optimal for a machine learning model. Hyperparameters are model setup parameters that are chosen before training. They have the potential to greatly affect the model's performance. Hyperparameters are external configurations which guide the learning process, as opposed to model parameters, which are learned during training. To maximize a machine learning model's performance and get the best outcomes, hyperparameter adjustment is essential.

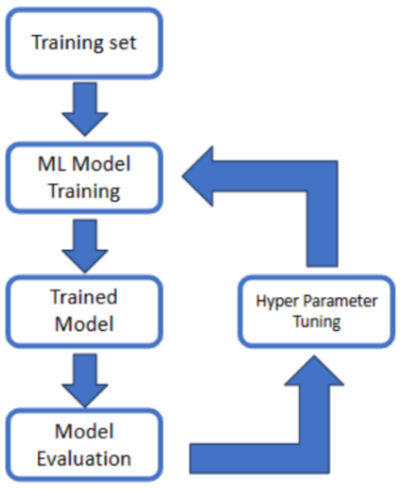


Fig.15 Working of ML model

1. Training a Machine Learning Model:

- Data Collection: Compile a dataset containing instances of the input data and the appropriate output (or target variable) that correspond to them.

- Feature extraction: Determine which features in the input data to extract and identify. The traits or qualities that the model will employ in its prediction process are known as features.
- Model Selection: Select a model architecture or machine learning technique based on its applicability to the given job. This decision is based on the properties of the data and the type of task (classification, regression, etc.).
- Training: Provide the selected model with the training data. The model modifies its internal parameters during training in order to reduce the discrepancy between the target values and its predictions. Gradient descent and other optimization techniques are frequently used in this procedure.

2. Testing a Machine Learning Model :

- Testing Data: To determine the model's performance and capacity for generalization, it must be tested with fresh, untested data once it has been trained.
- Examining or testing: The testing or validation dataset is a different dataset used to assess the model's performance on non-previously encountered data. Examples with known target values can be found in this dataset.
- Measures: Use task-relevant evaluation criteria to measure the model's performance. For example, common metrics for classification problems include accuracy, precision, recall, and F1 score; for regression problems, mean squared error is frequently employed.
- Adjustment (if necessary): The model's accuracy or ability to generalize to new data may require changes, such as feature set modification or hyperparameter tweaking, based on how well it performs on the testing dataset.

The goal of training and testing is to ensure that the model can make accurate predictions on new, unseen data and is not just memorizing the training examples. The division of data into training and testing sets helps assess how well the model generalizes to new data, providing an indication of its real-world performance.

Used ML model:

A statistical method called cross-validation is used to evaluate a predictive model's performance on a separate dataset.

It is frequently used in machine learning to assess a model's performance and address overfitting and underfitting-related problems.

Cross-validation helps in selecting the best model by comparing the average performance of different models.
It also aids in tuning hyperparameters to improve the model's performance.

Steps involved in cross-validation:

1.Dataset Splitting:

1)The dataset is divided into two subsets: the training set and the testing set.
2)The training set is used to train the model, while the testing set is reserved for evaluating its performance.

2.K-Fold Cross-Validation:

As mentioned in [9], The training set is further divided into K subsets, often referred to as "folds."
The value of K is usually chosen based on the size of the dataset.
The model is trained K times, each time using K-1 folds as the training data and the remaining fold as the validation data.
This results in K trained and evaluated models.

3.Accuracy Calculation:

The accuracy is calculated for each of the K models on their respective validation sets.

4.Average Performance:

The average performance across all K folds is calculated. This average performance is often considered a more reliable estimate of
the model's generalization performance compared to using a single train-test split.

Datasets :

1. UT Kinect Dataset

As mentioned in [10], The UT kinect dataset was collected as a part of research work for action recognition from depth sequences. It consists of 10 action types: *w*alk, sit down, stand up, pick up, carry, throw, push, pull, wave hands and clap hands performed by 10 subjects twice. As this dataset only contains the frames(i.e images) we used the python module moviepy, which is a Python library for video editing and manipulation. It provides a convenient and easy-to-use interface for working with video files. In our project moviepy is used to create a video by concatenating a sequence of images and then write the resulting video to a file.

2. Real Time Dataset

We've created our own dataset inspired by the UT Kinect Dataset. Our aim is to build a dataset like UT Kinect. This will make our dataset a reliable tool for training and testing machine learning models. It consists of 27 action types : walk, push, arm curl, standup, throw, basketball shoot, tennis serve, draw x, lunge, draw a triangle, swipe left, swipe right, jog, jump, claps, boxing, bowling, knock, sit down, squat, running, circle clockwise, circle anticlockwise etc.
Each subject repeated four actions four times, yielding a total of 108 action samples.

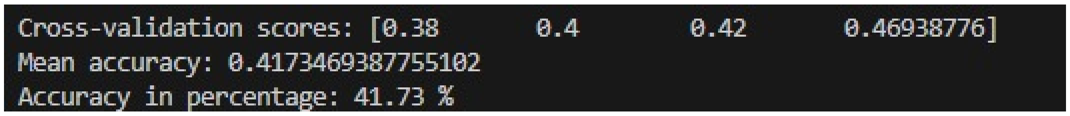
Results

The final outcome of the project is the accuracy of the designed machine learning model to predict the action samples given in the testing set by comparing the with the data given in the training set. To train this ml model, the data is divided into four folds and given to the ml model. In each iteration one of the folds of the data is considered as the testing set and the other three folds of the data as training set. In each iteration these testing set data is predicted from the training set and the accuracy of the machine learning model is noted as the cross validation scores. As we contain four folds in our project, four cross validation scores are noted and the final accuracy of the machine learning model is calculated by averaging the cross validation scores. This final accuracy of the machine learning model represents the final output and this ml model can be further used to predict the more real time actions by training the model beforehand.

1.Distance feature

1.1 Distance feature on UT Kinect Dataset

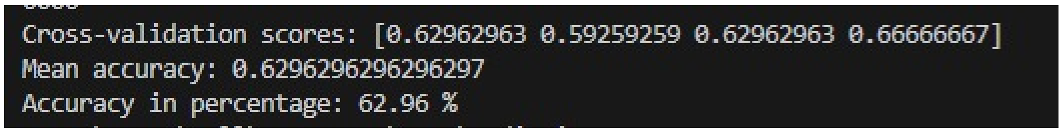
The UT Kinect dataset, which includes sequences of at least 24 frames, is a useful resource for training and evaluating machine learning models, particularly in the field of human activity recognition. The use of at least 24 frames per sequence highlights the significance of temporal dynamics in capturing meaningful patterns of human movement. Machine learning models can use this dataset to accurately learn and predict activities. Model evaluation entails assessing accuracy, precision, recall, and other metrics on a dataset that has been divided into training and testing sets. The selection of a minimum of 24 frames per sequence emphasizes the importance of capturing sequential actions in tasks such as gesture recognition and activity analysis.



```
Cross-validation scores: [0.38       0.4        0.42       0.46938776]
Mean accuracy: 0.4173469387755102
Accuracy in percentage: 41.73 %
```

1.2 Distance feature on Own Dataset
We set a minimum threshold of 40 frames per sequence in our custom dataset for training and evaluating machine learning models, particularly in the domain of human activity recognition. The decision to impose this minimum frame requirement stems

from the realization that a greater number of frames allows the model to capture more efficient temporal dynamics inherent in human movements. Each sequence of at least 40 frames provides a more detailed representation of actions and activities, enhancing the model's ability to detect patterns. This emphasis on increasing the number of frames in our dataset is intended to foster more accurate and robust machine learning models. During the model evaluation process, which typically entails dividing the dataset into training and testing subsets.

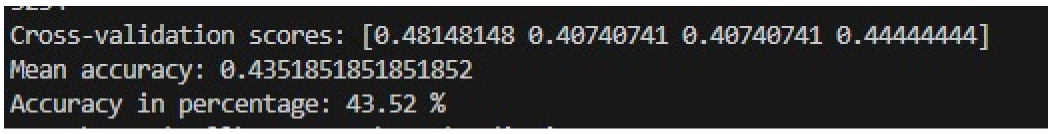


```
Cross-validation scores: [0.62962963 0.59259259 0.62962963 0.66666667]
Mean accuracy: 0.6296296296296297
Accuracy in percentage: 62.96 %
```

1.3 Interleaving Distance Feature on Own Dataset

Interleaving frames is a technique often used in the preprocessing of sequential data, including video or image sequences to enhance the information available for machine learning models. In the context of our dataset, interleaving frames involves strategically combining adjacent frames to create a more comprehensive representation of temporal data. Rather than relying solely on adjacent frames, this method interleaves sequences of frames to capture nuanced transitions and dependencies between actions. This approach not only reduces the dimensionality of the input data but also helps the model better grasp the temporal context and relationships between frames.

In our specific dataset, we used a novel approach to optimize the temporal information available for machine learning model training and evaluation by interleaving frames. Interleaving involves strategically combining adjacent frames to create a more condensed yet informative representation of temporal dynamics with a predefined minimum of 40 frames per sequence. This method allows our dataset to capture nuanced transitions and dependencies between actions, resulting in a more complete understanding of human activities. We hope to improve the efficiency and performance of machine learning models on our dataset by incorporating this interleaving technique.

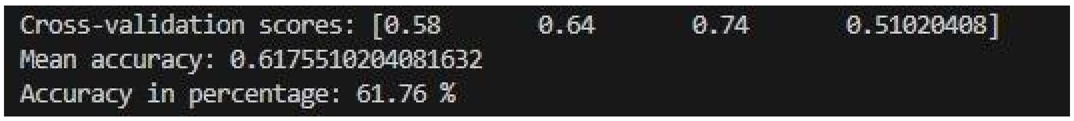


```
Cross-validation scores: [0.48148148 0.40740741 0.40740741 0.44444444]
Mean accuracy: 0.4351851851851852
Accuracy in percentage: 43.52 %
```

2. Angle feature

2.1 Angle feature on UT Kinect Dataset

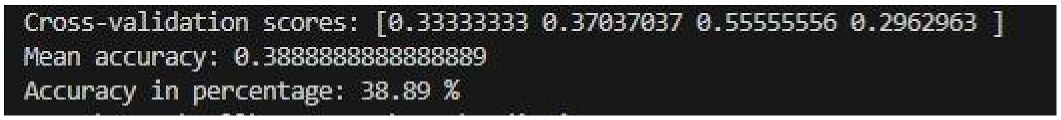
With a minimum of 24 frames per sequence in the UT Kinect Dataset, angle calculation becomes a critical step in extracting valuable spatial information from the provided data. The machine learning model can recognize specific postures or gestures by calculating angles between joints or limbs across these frames. With a limited number of frames, it's crucial to extract key angles as they define human movements within the dataset's timeframe, enabling accurate recognition of activities. This angular information is required for training models to accurately recognize and classify various activities.



```
Cross-validation scores: [0.58       0.64       0.74       0.51020408]
Mean accuracy: 0.6175510204081632
Accuracy in percentage: 61.76 %
```

2.2 Angle feature on Own Dataset

In our dataset with at least 40 frames per sequence, calculating angles becomes crucial for understanding detailed joint movements. The increased frame count allows a more nuanced analysis of skeletal angles and limb motions, enhancing the model's capability to capture intricate human actions. By tracking changes over this extended duration, our approach contributes to a more accurate and generalized representation of various activities, providing a robust foundation for effective machine learning model training. This broader timeframe improves the model's accuracy and ability to recognize different activities.



```
Cross-validation scores: [0.33333333 0.37037037 0.55555556 0.2962963 ]
Mean accuracy: 0.3888888888888889
Accuracy in percentage: 38.89 %
```

Conclusion

Our project involves action recognition using machine learning, with a focus on datasets, feature matrices, and machine learning model development. The combination of feature matrices and index matrices facilitates the mapping between feature vectors and corresponding actions. Training and testing processes are critical for ensuring the model's ability to generalize to new data. Cross validation process enhances the model's robustness, ensuring it generalizes well to diverse, unseen data and contributes to optimal model configuration. The use of both pre-existing datasets like UT Kinect and a newly created real-time dataset enhances the diversity and reliability of the training and testing data. The project's comprehensive approach, including dataset creation, feature representation, model training, and evaluation strategies, positions it well for developing effective machine learning models for action recognition. The use of cross-validation reflects a commitment to robust model assessment and improvement.

References

posenet - https://github.com/rwightman/posenet-python
openpose - https://viso.ai/deep-learning/openpose/
UT Kinect dataset - https://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html
Cross validation - https://scikit-learn.org/stable/modules/cross_validation.html

1. Cao, Z., Hidalgo, G., Simon, T., Wei, S., & Sheikh, Y. (2018). OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*, 172-186.

2. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C., Yong, M.G., Lee, J., Chang, W., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A Framework for Building Perception Pipelines. *ArXiv, abs/1906.08172*.

3. Lakhwani, Kamlesh & Murarka, P & Narendra, Mr. (2015). Color Space Transformation for Visual Enhancement of noisy color Image. IET Image Processing.

4. Merhej, C., Beal, R., Matthews, T., & Ramchurn, S. (2021). What Happened Next? Using Deep Learning to Value Defensive Actions in Football Event-Data. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 2863-2873).

5. Liyan Luo, An Autonomous Star Identification Algorithm Based on One-Dimensional Vector Pattern for Star Sensors, Sensors 2015, 15(7), 16412-16429; doi:10.3390/s150716412.

6. Forina, D., Cerra, D., & Cittaro, J. (2014). Feature extraction based on dependency measures. IEEE Transactions on Knowledge and Data Engineering, 26(2), 425-439.

7. Liu, X., & Zhao, Q. (2019). Index matrix-based image classification. IEEE Transactions on Image Processing, 28(10), 4836-4848.

8. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

9. Yates, Luke A., Aandahl, Zach, Richards, Shane A., and Brook, Barry W.. 2023. “ Cross Validation for Model Selection: A Review with Examples from Ecology.” *Ecological Monographs* 93(1): e1557.

10. F Gu, K. Khoshelham, S. Valaee, J. Shang and R. Zhang, "Locomotion Activity Recognition Using Stacked Denoising Autoencoders," in IEEE Internet of Things Journal, vol. 5, no. 3, pp. 2085-2093, June 2018, doi: 10.1109/JIOT.2018.2823084.