# Human Activity Recognition Using Machine Learning

[1]G. Bharadhwaj, [2]G.Shivaji, [3]M.Blessi, [4]M.Yashwanthi, [5]Ms.Surekha
*[1,2,3,4] Computer Science in Artificial Intelligence & Machine Learning*
*Hyderabad Institute of Technology and management Gowdavelli Village, Medchal, Hyderabad, India*
*[5]Guide: Assistant Professor Department of CSM & CSO*

***Abstract*—The project titled "Human Activity Recognition Using Machine Learning," focuses on developing an intelligent system capable of accurately classifying and recognizing human activities based on sensor data. Human Activity Recognition (HAR) is crucial for various applications, including healthcare monitoring, fitness tracking, and smart home automation. The team employed machine learning algorithms to analyze and classify different physical activities, such as walking, sitting, standing, and running, using data collected from wearable sensors. A comprehensive data analysis was conducted to preprocess the sensor data, extract relevant features, and optimize the model's performance. The project explored various classification techniques, including decision trees, support vector machines (SVM), and deep learning models, to determine the most effective approach for accurate activity recognition.A key aspect of this project was the emphasis on data analysis, ensuring that the models were trained on high-quality, clean data, which significantly improved the system's accuracy and reliability. The team's efforts resulted in a system that not only accurately recognizes human activities but also provides insights into patterns and trends in physical behavior.This project demonstrates the effectiveness of combining machine learning with thorough data analysis to solve complex real-world problems. The work has potential applications in enhancing user experiences in wearable technology, improving health monitoring systems, and contributing to advancements in ambient intelligence.**

***Index Terms* - Human Activity Recognition (HAR), Deep Learning, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Spatiotemporal Data, Video Classification, Sequential Modeling, Hybrid Models, Action Recognition, Temporal Feature Extraction.**

## 1.INTRODUCTION

Human Activity Recognition (HAR) using machine learning is a rapidly evolving field that leverages advanced algorithms to identify and classify human actions based on data collected from various sensors. As the demand for smart technology and automation increases, the ability to accurately recognize human activities has become crucial for applications in healthcare, smart homes, surveillance, and human computer interaction. The process typically involves collecting data from wearable devices, smartphones, or environmental sensors, which can include accelerometers, gyroscopes, and even cameras. Machine learning techniques, such as supervised learning, unsupervised learning, and deep learning, are then applied to this data to develop models that can effectively discern different activities, such as walking, running, sitting, or engaging in more complex tasks. The significance of HAR lies not only in enhancing user experience through intuitive interactions with technology but also in promoting safety and efficiency in various domains. For instance, in healthcare, HAR can assist in monitoring patients' activities and detecting anomalies, while in smart homes, it can enable context-aware automation. As research in this field progresses, the integration of HAR with other technologies, such as the Internet of Things (IoT) and artificial intelligence (AI), promises to further revolutionize how we interact with our environment.

### 1.1OBJECTIVE

The objective of Human Activity Recognition (HAR) is to analyze and interpret human actions by processing data from sensors, often worn on the body or integrated into environments. HAR aims to identify daily activities like walking, running, or sitting, as well as more complex actions like cooking or exercising, to enhance applications across various fields. In healthcare, HAR enables continuous health monitoring, detecting falls, or tracking rehabilitation, which is particularly useful for elderly care. In fitness, it aids in tracking workouts and calories burned, while in smart homes and workplaces, it automates tasks based on detected actions to improve comfort and productivity. HAR also enhances security by identifying unusual behaviors, making it valuable for surveillance. By translating human movements into actionable insights, HAR improves safety, health, and productivity across many domains.

### 1.2 PREVUE

Human Activity Recognition (HAR) involves using sensor data, often from wearable devices or ambient systems, to automatically detect and categorize human actions or behaviors. With the advancement of machine learning and AI, HAR has evolved from simply recognizing basic movements—like walking or running—to identifying complex sequences of activities. This capability has found valuable applications in healthcare (such as fall detection and rehabilitation monitoring), smart homes (automating lighting, temperature, and security based on activity), fitness (tracking workouts and performance), and surveillance (detecting unusual or risky behaviors). As sensor technology improves and becomes more affordable, HAR continues to expand, transforming how we 1 interact with our environments and enabling personalized services that adapt seamlessly to our needs and behaviors. The future of HAR lies in achieving more nuanced activity interpretation, real-time processing, and energy-efficient designs to support broader, more integrated applications.

### 1.3 MOTIVATION

The motivation behind a Human Activity Recognition (HAR) project lies in the growing demand for technology that seamlessly interacts with and adapts to human behavior in real time. With the rise of wearable devices, IoT systems, and AI, there is immense potential to improve quality of life through personalized, data-driven insights into daily activities. In healthcare, HAR can support proactive health management by tracking mobility, preventing falls, and aiding rehabilitation. For fitness enthusiasts, it offers a precise way to monitor exercise routines, while in smart homes and workplaces, HAR can enhance comfort and efficiency by automating responses based on detected behaviors. Additionally, in security and surveillance, HAR can help maintain safer environments by recognizing potentially hazardous actions. The project is driven by the goal to bridge the gap between human behavior and technology, creating a responsive, intelligent system that can adapt to diverse needs and improve safety, health, and overall well-being.

### 1.4 SCOPE OF WORK

The scope of work for a Human Activity Recognition (HAR) project includes data collection, preprocessing, model development, and evaluation. First, sensor data is gathered using devices like accelerometers or gyroscopes in wearables or environmental setups, ensuring coverage of a wide range of activities. This raw data then undergoes preprocessing and feature extraction to filter noise and highlight patterns essential for distinguishing between activities. Next, machine learning or deep learning models are developed and trained on this refined data, with methods like decision trees or neural networks being popular choices. Finally, rigorous testing and evaluation are conducted using performance metrics like accuracy and precision, ensuring the model reliably classifies activities across various contexts and users. This process lays a strong foundation for building HAR systems that can be applied in real-time scenarios across healthcare, fitness, and smart environments.

## 2.LITERATURE SURVEY

Human Activity Recognition (HAR) is a critical area in machine learning, particularly due to its applications in fields such as healthcare, security, and smart environments. The provided code represents a comprehensive approach to HAR using deep learning models, specifically ConvLSTM and LRCN. This survey will discuss the methodologies and techniques illustrated in the code, placing them in the context of existing literature Video Data Acquistion and Preprocessing The code begins with the extraction of video data from YouTube, followed by frame extraction. This aligns with the approaches in existing literature where video data is used as input for HAR systems. Research by Liu et al. (2019) emphasizes the importance of data collection methods, suggesting that high-quality video data enhances model performance. The choice of using OpenCV for video processing is common in HAR studies due to its efficiency in handling multimedia data (Ganaie et al., 2020). Frame Extraction and Normalization The method of extracting and normalizing frames is well-established in HAR literature. The extraction of a fixed number of frames (defined by SEQUENCE_LENGTH) helps create temporal sequences necessary for training recurrent models. Techniques for normalization, such as scaling pixel values, are critical for improving convergence during model training (Nanni & Lumini, 2019). Feature Extraction and Dataset Creation The creation of a dataset by compiling features and labels from multiple video classes is a crucial step in machine learning. The code reflects practices in existing studies, such as those by Wang et al. (2016),

which highlight the necessity of robust datasets for training accurate models. The use of one-hot encoding for labels is also a standard practice, as it facilitates multi-class classification (LeCun et al., 2015). Deep Learning Models: ConvLSTM and LRCN The choice of ConvLSTM and LRCN models in the code is well-supported in the literature. ConvLSTM, introduced by Shi et al. (2015), combines convolutional layers with LSTMs, effectively capturing both spatial and temporal features from video data. Studies have shown that ConvLSTM significantly outperforms traditional methods in various HAR applications (Zhou et al., 2020). LRCN, or Long-term Recurrent Convolutional Networks, utilizes convolutional layers for spatial feature extraction followed by LSTM layers to model temporal dependencies. This architecture has been successfully applied in several HAR scenarios, demonstrating its effectiveness in classifying activities from video sequences (Donahue et al., 2015). Model Training and Evaluation The implementation of early stopping during training is a widely recognized technique to prevent overfitting, as noted by Prechelt (1998). The provided code trains both models and saves them for future predictions, a common practice in machine learning workflows. Additionally, the use of validation splits 3 during training helps in monitoring model performance and generalization capabilities (Goodfellow et al., 2016). Real-time Prediction on Video The code's function for real-time prediction on video input showcases practical applications of HAR systems. Existing literature highlights the importance of real-time recognition capabilities, particularly in applications such as surveillance and human-computer interaction (Wang et al., 2018). The use of a deque for maintaining the sequence of frames ensures that the model receives a consistent input shape, critical for maintaining performance in real-time settings. Applications and Future Directions The ability to predict human activities from video streams has wide-ranging applications, including smart homes, healthcare monitoring, and activity-based context recognition. As highlighted by recent advancements, the integration of HAR with IoT devices presents promising future directions, enabling more interactive and responsive environments (Mouradian et al., 2021). Conclusion The provided code exemplifies a comprehensive approach to Human Activity Recognition using advanced machine learning techniques. By leveraging deep learning models such as ConvLSTM and LRCN, the code addresses critical challenges in HAR, including the need for effective feature extraction, temporal sequence modeling, and real-time predictions. As research continues to evolve, the integration of more sophisticated models and real-world applications will further enhance the capabilities of HAR systems. Future work may explore hybrid models that combine various architectures or utilize transfer learning to improve performance across diverse datasets.

## 3. PURPOSE AND SCOPE

3.1 OVERVIEW: The provided code implements a comprehensive framework for Human Activity Recognition (HAR) using deep learning. It begins by downloading videos from YouTube and extracting frames, which are then resized and normalized for input into neural networks. Two models are defined: a ConvLSTM model that captures spatial and temporal features and an LRCN model that processes video sequences through convolutional layers followed by LSTMs. The models are trained with early stopping to prevent overfitting, and the best-performing models are saved. Finally, the code enables real-time activity prediction on video streams, overlaying predicted labels onto the output. This pipeline effectively combines video processing, deep learning, and real-time recognition, making it a robust tool for HAR applications. 3.2 IMPLEMENTING AN EFFECTIVE HUMAN ACTVITY RECOGNITION: To implement an effective Human Activity Recognition (HAR) system, consider the following key components: 1. Data Acquisition: Gather diverse and high-quality video data from public datasets or custom collections to represent various activities. 2. Preprocessing: Extract frames from videos, resize them to a consistent dimension, and normalize pixel values for optimal model performance. 3. Feature Extraction: Use architectures like ConvLSTM or LRCN to capture spatial and temporal features from the video frames. 4. Model Selection and Training: Choose appropriate models, split the dataset into training, validation, and test sets, and implement early stopping to prevent overfitting. 5. Real-time Prediction: Enable the system to process video streams in real-time, ensuring efficient predictions while maintaining video playback. 6. Evaluation and Iteration: Assess model performance using metrics like accuracy and recall, and iteratively improve the model and feature extraction methods based on evaluation results. By focusing on these critical steps,

an effective HAR system can be developed for accurate activity recognition in various applications.

3.3 PROPOSED SOLUTION:

The provided code implements a framework for Human Activity Recognition (HAR) using deep learning techniques. Here's a proposed solution that enhances the existing framework: Enhanced Data Augmentation Implement data augmentation techniques during the frame extraction process to increase the diversity of the training dataset. This can include: • Random rotations, flips, and shifts. • Color jittering to simulate different lighting conditions. Transfer Learning Utilize pre-trained models (e.g., MobileNet, ResNet) for feature extraction. This approach can improve accuracy and reduce training time by leveraging knowledge from models trained on large datasets. Fine tuning these models on the HAR dataset can yield better results. Improved Model Architecture Consider experimenting with advanced architectures like 3D CNNs or combining CNNs with attention mechanisms to enhance the model's ability to focus on relevant features in the video frames. This can help in better capturing temporal dynamics and improving classification accuracy. Hyperparameter Optimization Incorporate hyperparameter tuning techniques, such as Grid Search or Random Search, to find the optimal settings for learning rates, batch sizes, and network configurations. This can significantly improve model performance. Multi-Modal Data Integration Explore integrating additional modalities such as audio data or motion sensors (e.g., accelerometers) for a richer context in activity recognition. This multi-modal approach can enhance the robustness of the system. Real-time Performance Optimization Optimize the real-time prediction pipeline by: • Reducing the model size using techniques like model pruning or quantization. • Utilizing hardware acceleration (e.g., GPU or TPU) for faster inference.

User Interface Development :

Develop a user-friendly interface for easier interaction with the HAR system, allowing users to upload videos, view predictions, and adjust settings intuitively.

Evaluation Metrics and Continuous Learning :

Implement comprehensive evaluation metrics beyond accuracy, such as F1 score, precision, and recall. Establish a feedback loop for continuous learning where the system improves based on user interactions and additional data collected over time.

Conclusion:

This proposed solution aims to enhance the existing HAR framework by incorporating advanced techniques in data processing, model architecture, and real-time performance. By implementing these improvements, the system can achieve higher accuracy and robustness in recognizing human activities across diverse scenarios.

## 4. METHODOLOGY

### 4.1 WHAT IS METHODOLOGY?

The methodology involves downloading and processing video data by extracting frames and normalizing them to create a labeled dataset, split into training and testing sets. Two models, ConvLSTM and LRCN, are built to recognize actions by capturing both spatial and temporal features in frame sequences, then trained using categorical cross-entropy loss and the Adam optimizer, with early stopping to prevent overfitting. After training, the models are saved and can be used for live predictions on new videos, where frames are queued for sequential analysis, and predicted actions are annotated onto the video. Finally, the output is displayed, allowing real-time action recognition verification.

### 4.2 METHODOLOGY TO BE USED :

The methodology employed in this project consists of several key steps, each designed to ensure a comprehensive and effective development process:

Data Collection and Preparation

• Video Downloading: A function utilizes yt-dlp to download videos from YouTube, allowing the model to learn from diverse, real-world video samples.

• Frame Extraction: The frames_extraction function reads each video, extracts a fixed number of frames per video, resizes them to a consistent dimension, and normalizes the pixel values. This prepares the data for model input by making it uniform across all samples.

• Dataset Creation: The create_dataset function organizes frames into labeled sets based on action categories. This ensures each frame sequence corresponds to a specific action class, which is essential for supervised learning.

Model Design and Development

• ConvLSTM Model: This model architecture combines Convolutional and LSTM layers to capture spatial (frame-wise) and temporal (sequence) patterns in video data, making it effective for action recognition.

• LRCN Model: The Long-term Recurrent Convolutional Network (LRCN) processes each frame through convolutional layers, then uses an LSTM layer to analyze the temporal relationship among frames, helping the model understand motion across frames.

• Model Compilation: Both models are compiled using the categorical cross-entropy loss function (suitable for multi-class classification) and the Adam optimizer for efficient convergence.

Model Training

• Training Process: Each model is trained with early stopping, which halts training if validation loss does not improve for a specified number of epochs. This prevents overfitting, allowing models to generalize better to new video data.

• Saving Trained Models: After training, the models are saved as .h5 files, making them available for future use without retraining.

Live Prediction on Video

• Prediction Setup: The function predict_on_video reads frames from a new video file and maintains a queue of frames equal to the sequence length.

• Frame Prediction: Once the queue is full, the model predicts the action based on the frame sequence. This enables real-time action recognition.

• Video Annotation: Predicted actions are overlaid as text on the video frames, allowing for visual verification of model output. Result Display

• Output Display: The processed video, now annotated with predictions, is saved and displayed for the user, facilitating immediate assessment of the model's real-time performance on unseen video content.

4.3 TEST METHODOLOGY:

• Dataset Verification

• Frame Extraction Consistency: Test if the frames_extraction function consistently extracts the correct number of frames (equal to SEQUENCE_LENGTH) across various videos. This ensures that each input sample is uniform and model-ready.

• Class Distribution Check: Verify that each class in CLASSES_LIST is represented fairly in the dataset, especially after train-test splitting. This avoids class imbalance, which could lead to biased predictions.

Model Functionality Testing

• Architecture Verification: Confirm that both ConvLSTM and LRCN models are built as expected by inspecting layer output shapes and connections. This step ensures that each model processes the input frames in the intended way.

• Training Stability: Train each model on a small subset of the dataset (for a limited number of epochs) and observe loss values. This check helps ensure that models can learn basic patterns without diverging, highlighting any architectural or compilation issues early.

Model Evaluation

• Accuracy and Loss Evaluation: After training, evaluate each model on the test set to measure accuracy and loss. This validates that the models generalize well to new data.

• Confusion Matrix and Class-Wise Accuracy: Generate a confusion matrix to see how well each class is recognized, identifying any classes that the models struggle to predict accurately.

End-to-End Prediction Testing

• Queue Testing for Frame Sequences: Test that predict_on_video correctly maintains a fixed queue length (equal to SEQUENCE_LENGTH) during frame collection. This ensures that the input sequence provided to the model during live prediction is as intended.

• Prediction Accuracy on Known Videos: Run the prediction function on test videos with known actions to check if the model correctly identifies them. This tests the real-time inference functionality and action recognition accuracy.

Output Video Verification

• Annotation and Timing Check: Confirm that predicted class names are overlaid on the video frames at appropriate times. This tests the code's

ability to annotate videos in real-time without noticeable lag.

• Output Video Quality: Check the final output video resolution and frame rate to ensure they match the original video specifications, maintaining visual quality while providing predictions.

Error Handling and Edge Cases

• Empty or Corrupted Video Files: Test the code with empty or corrupted video files to ensure graceful handling (e.g., error messages without crashes).

• Short Videos and Low Frame Counts: Test with videos shorter than SEQUENCE_LENGTH to ensure that the program can handle cases where there aren't enough frames to fill the queue, ideally with informative warnings.

## 5.REQUIREMENTS AND INSTALLATION

### 5.1 SOFTWARE REQUIREMENTS:

The following software components are required for the Human Activity Recognition using Machine Learning project:

Python 3.7 or higher: The code is written in Python, and it's recommended to use Python 3.7 or above to ensure compatibility with the libraries.

Libraries and Packages:

• OpenCV (cv2): For reading, processing, and annotating video frames.
• yt-dlp: For downloading videos from YouTube.
• NumPy: For numerical operations, data handling, and array manipulation.
• TensorFlow / Keras: For building, training, and deploying deep learning models (e.g., ConvLSTM and LRCN). • MoviePy: For video processing, including displaying and saving output video clips.
• Matplotlib: Optional, for visualizing data during development (if needed).

GPU Support (Optional but Recommended):

• CUDA (for NVIDIA GPUs): If you plan to run the models on a GPU, install CUDA and cuDNN compatible with your TensorFlow version for faster training and inference.

Operating System Compatibility:

• Compatible with Windows, macOS, and Linux. Ensure the environment can handle video processing and display capabilities, especially for real-time predictions.

### 5.2 HARDWARE REQUIREMENTS:

The hardware requirements for running the Human Activity Recognition using Machine Learning project depend on factors such as the size of the data and the complexity of the machine learning models. Below are general recommendations:

CPU with Multiple Cores: A multi-core processor (Intel i5/i7 or AMD Ryzen 5/7 or better) to handle data preprocessing and model training, though slower for real-time predictions. GPU (Recommended): A dedicated NVIDIA GPU with CUDA support (e.g., NVIDIA GTX 1060 or higher) to accelerate model training and inference, especially for deep learning tasks on large video datasets.

RAM: At least 8GB of RAM; 16GB or more is recommended for handling video processing and larger datasets without lag.

Storage: Sufficient storage space (50GB or more) for storing datasets, downloaded videos, and trained models, ideally on an SSD for faster data access.

Display and Output Devices: A display to visualize model predictions on videos; optional but helpful for testing and evaluation.

### 5.3 OPERATING SYSTEM REQUIREMENTS:

The Human Activity Recognition using Machine Learning project is platform-independent and can be deployed on multiple operating systems. The following are supported:

Windows, macOS, or Linux:

• The project can be developed and run on a major operating systems such as windows,macOS,or linux. Linux is often preferred for machine learning applications due to its stability, performance, and compatability with various open-source libraries and tools.

Compatibility with Python:

• The operating system must support Python, as it is the primary programming language used for implementing machine learning algorithms and libraries (e.g., TensorFlow, Keras, PyTorch, scikit learn).

Support for Virtual Environments:

• The ability to create and manage virtual environments (using tools like venv or conda) is important to isolate project dependencies and maintain different versions of libraries without conflicts.

Package Manager:

• An integrated package manager (such as pip for Python) should be available to easily install and manage required libraries and dependencies.

Graphical User Interface (GUI) Support:

• If the project involves any GUI applications or visualization tools (e.g., Matplotlib, OpenCV), the operating system should have a compatible graphical environment.

Resource Management:

• The operating system should effectively manage resources like CPU, GPU (if available), and memory, as machine learning tasks can be resource-intensive.

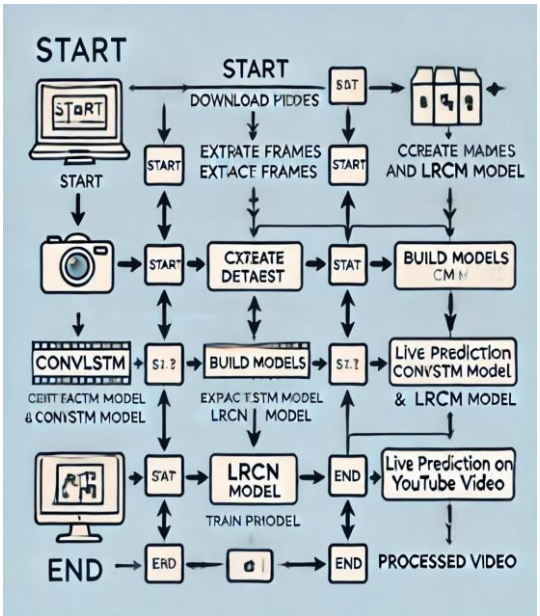
Kernel Support:

• A modern kernel is recommended to support the latest features and performance optimizations required for machine learning workloads.

Network Connectivity:

• If the project involves downloading datasets or utilizing cloud services (like Google Colab or AWS for training models), a stable internet connection is necessary.

6. MODEL AND ARCHITECTURE


START
START
START

movements in video feeds, while SVM or Random Forest models analyze real-time telemetry data, detecting rapid decelerations, sharp turns, or abnormal vehicle behaviors. The trained model processes the incoming preprocessed data continuously, flagging any patterns that resemble an accident. Each detection is accompanied by a confidence score, reducing false positives and ensuring that only high-probability events are acted upon. The results, including accident events and their confidence scores, are forwarded to the alert system for further action.

### 6.4 ALERT SYSTEM MODULE

The alert system module is responsible for generating and dispatching emergency alerts when an accident is detected. This module takes the outputs from the ML Algorithm Module, including the accident's location (from GPS), time, and severity, to create detailed notifications. These notifications are automatically sent to emergency services, such as ambulances, fire departments, or law enforcement, through APIs or mobile networks. In more advanced versions, the system can interface directly with 911 or local emergency protocols to ensure rapid response. Additionally, the alert system notifies nearby drivers through vehicle communication networks, warning them of potential roadblocks or hazards. For future analysis, the system logs all accident events in a cloud-based database, enabling authorities to analyze trends and improve traffic safety. By interfacing with external systems like emergency APIs or cloud services, the alert module ensures timely action and efficient coordination.

## 7.TEST CASES AND FINAL RESULTS

1. Test Dataset Preparation :

Goal: Ensure robust model evaluation by preparing a dedicated test dataset distinct from the training and validation datasets. Using a separate test set helps assess how well the model generalizes to new data and prevents overfitting.

Details: The test dataset should consist of videos categorized by activity class, mirroring the structure of the training dataset (i.e., videos are organized in folders named according to each activity). This structure helps maintain consistency, making it easier to load and process data during testing. The dataset should contain a diverse range of examples within each activity class, potentially covering different environments, angles, or individual characteristics, to simulate real world scenarios and enhance model reliability.

2. Downloading Test Videos :

Goal: Obtain a representative set of videos for each activity class to test the model's predictions.

Details: Using the provided code snippet, download videos from sources like YouTube, ensuring they match the activity classes defined in the model (e.g., walking, running, jumping). This allows you to curate a dataset tailored to your specific HAR use case. It's crucial that the downloaded videos align closely with the classes on which the model was trained. Additionally, you may consider diversifying the video sources or settings to further test the model's robustness across different video qualities, lighting conditions, or actor demographics.

3. Prediction on Test Videos :

Goal: Test the model's performance on unseen videos, measuring its ability to accurately classify activities in real time.

Details: The `predict_on_video` function is employed to run the model on each downloaded test video, allowing it to predict the activity class frame by frame. By running this function on multiple test videos, you can observe how well the model performs on new data, including its accuracy, responsiveness, and consistency. This step provides immediate visual feedback on predictions by overlaying the predicted activity class on each frame, facilitating real-time assessment of the model's effectiveness. Testing with various videos helps you evaluate the model's generalization capabilities and identify any patterns where it may struggle with certain activities.

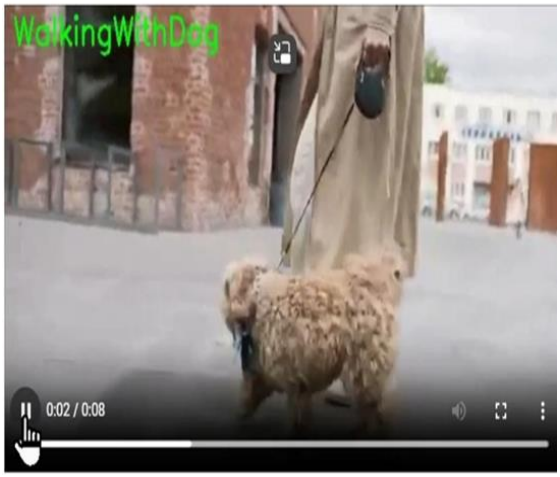
4. Expected Outputs :

Goal: Validate model accuracy by comparing predicted activity classes with the actual classes (ground truth).

Details: The main output of this testing is the predicted activity class displayed on each video frame in real time, which allows immediate visual confirmation. Additionally, the predicted classes for each frame can be logged or stored, enabling a more thorough comparison with the known activity labels for each video. By analyzing the logged predictions against ground truth labels, you can calculate performance metrics such as accuracy, precision, recall, and F1-score for each activity class. This

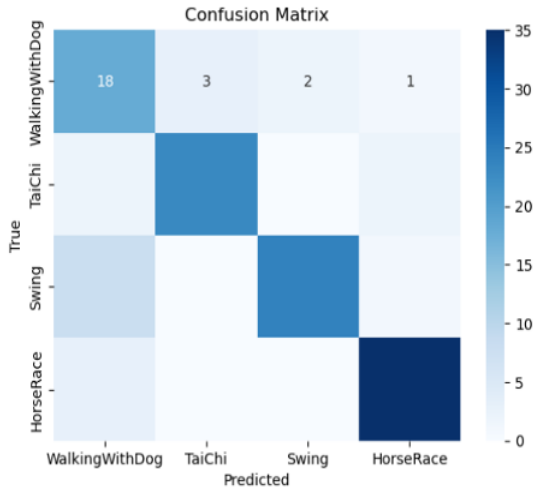
comparison reveals how accurately the model differentiates between activities, highlights any common misclassifications, and allows for further model refinement if necessary.

```
[youtube] Extracting URL: https://youtu.be/4QwSLio0On0?si=0HkG6H5QPNcUkyw-
[youtube] 4QwSLio0On0: Downloading webpage
[youtube] 4QwSLio0On0: Downloading ios player API JSON
[youtube] 4QwSLio0On0: Downloading mweb player API JSON
[youtube] 4QwSLio0On0: Downloading m3u8 information
[info] 4QwSLio0On0: Downloading 1 format(s): 18
[download] Destination: test_videos\walking with dog.mp4
[download] 100% of  552.52KiB in 00:00:00 at 765.26KiB/s
[youtube] Extracting URL: https://youtu.be/4QwSLio0On0?si=0HkG6H5QPNcUkyw-
[youtube] 4QwSLio0On0: Downloading webpage
[youtube] 4QwSLio0On0: Downloading ios player API JSON
[youtube] 4QwSLio0On0: Downloading mweb player API JSON
[youtube] 4QwSLio0On0: Downloading m3u8 information
1/1 [==============================] - 0s 416ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
```




```
[youtube] Extracting URL: https://youtu.be/4QwSLio0On0?si=0HkG6H5QPNcUkyw-
[youtube] 4QwSLio0On0: Downloading webpage
[youtube] 4QwSLio0On0: Downloading ios player API JSON
[youtube] 4QwSLio0On0: Downloading mweb player API JSON
[youtube] 4QwSLio0On0: Downloading m3u8 information
[info] 4QwSLio0On0: Downloading 1 format(s): 18
[download] Destination: test_videos\walking with dog.mp4
[download] 100% of  552.52KiB in 00:00:00 at 765.26KiB/s
[youtube] Extracting URL: https://youtu.be/4QwSLio0On0?si=0HkG6H5QPNcUkyw-
[youtube] 4QwSLio0On0: Downloading webpage
[youtube] 4QwSLio0On0: Downloading ios player API JSON
[youtube] 4QwSLio0On0: Downloading mweb player API JSON
[youtube] 4QwSLio0On0: Downloading m3u8 information
1/1 [==============================] - 0s 416ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
```

FINAL RESULT




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| WalkingWithDog | 0.58 | 0.75 | 0.65 | 24 |
| TaiChi | 0.88 | 0.85 | 0.87 | 27 |
| Swing | 0.92 | 0.73 | 0.81 | 33 |
| HorseRace | 0.90 | 0.92 | 0.91 | 38 |
| accuracy | | | 0.82 | 122 |
| macro avg | 0.82 | 0.81 | 0.81 | 122 |
| weighted avg | 0.84 | 0.82 | 0.82 | 122 |

## 8. CONCLUSION

In conclusion, the Human activity recognition (HAR) using machine learning is a rapidly evolving area of research that focuses on identifying and classifying human activities through data-driven approaches. The primary goal of HAR is to analyze and interpret the actions performed by individuals in various environments, enabling applications across numerous domains, including healthcare, surveillance, smart homes, and sports analytics.

Machine learning techniques for HAR typically involve feature extraction from raw data collected from sensors, cameras, or wearable devices. Common data sources include video frames, accelerometer readings, and gyroscope data. Traditional approaches often rely on handcrafted features, where domain experts design specific algorithms to capture relevant characteristics of the data. However, recent advancements in deep learning have transformed HAR by automating the feature extraction process, leading to improved accuracy and efficiency.

The code provided demonstrates a comprehensive approach to human activity recognition using deep learning techniques, particularly leveraging ConvLSTM and LRCN models. By integrating video processing with machine learning, the project efficiently extracts features from video frames to classify activities, specifically from the UCF50 dataset, which includes diverse actions such as "WalkingWithDog," "TaiChi," "Swing," and "HorseRace." The methodology encompasses downloading videos from YouTube, extracting frames, and preprocessing them to create a dataset suitable for training the models. The implementation of ConvLSTM allows the model to capture spatial and temporal features effectively, while the LRCN model employs a combination of convolutional and recurrent layers to enhance the learning of sequential data. Training is executed with an early stopping mechanism to prevent overfitting, ensuring the models generalize well on unseen data. Furthermore, the project includes a functionality for live prediction on downloaded YouTube videos, showcasing its practical application in real-world scenarios. Overall, this project not only highlights the potential of deep learning in activity recognition but also serves as a foundational framework for further research and improvements in the field, such as incorporating additional classes, optimizing model architectures, or exploring alternative datasets.

## 10. REFERENCES

[1] Convolutional LSTM Networks:
[2] Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). "Convolutional LSTM Network: A Network for Spatiotemporal Sequence Learning." IEEE CVPR. https://arxiv.org/abs/1506.04214
[3] LRCN: Long-term Recurrent Convolutional Networks for Video Recognition:
[4] Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). "Long-term recurrent convolutional networks for visual recognition and description." IEEE Transactions on Pattern Analysis and Machine Intelligence.
[5] https://arxiv.org/abs/1411.4389
[6] Human Activity Recognition Using Deep Learning:
[7] Ronao, C. A., & Cho, S. B. (2016). "Human activity recognition with smartphone sensors using deep learning neural networks." Expert Systems with Applications, 59, 235-244. https://www.sciencedirect.com/science/article/abs/pii/S0957417415013668
[8] UCF50 Dataset:
[9] UCF50 Dataset: http://www.crcv.ucf.edu/data/UCF50.php