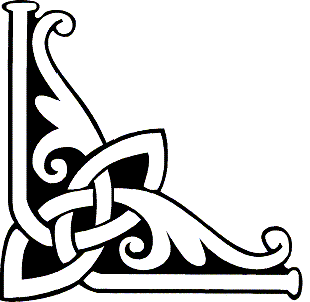
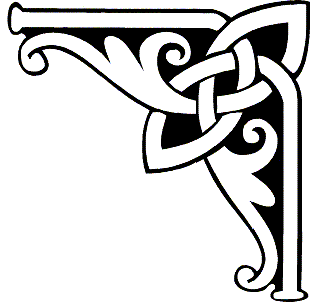
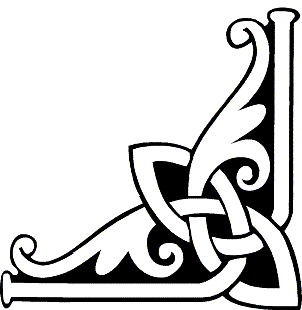
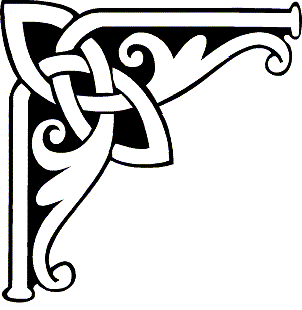
### VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELAGAVI - 590 018, KARNATAKA.



#### MINI PROJECT REPORT

**On**

### “HUMAN ACTIVITY RECOGNITION SYSTEM”

*Submitted in the partial fulfillment of requirements*

*for*

***COMPUTER GRAPHICS AND IMAGE PROCESSING LABORATORY (21CSL66)***

*Submitted by*

#### TASBIHA TAZEEN 4BD21CS165

**THEJASWINI C.R. 4BD21CS168**

### Project Guide: Project Co-Guide:

**Dr. Naveen H M M. Tech., Ph.D Prof. Mohamed Muthahir R M. Tech**

#### Associate Professor Assistant Professor

**Department of CS&E Department of CS&E**

**B.I.E.T., Davanagere B.I.E.T., Davanagere**

### Department of Computer Science and Engineering.

**Bapuji Institute of Engineering & Technology Davangere- 577004**

**2024**

## Bapuji Institute of Engineering and Technology Davangere -577004



### Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **TASBIHA TAZEEN** and **THEJASWINI C.R.** bearing **USN 4BD21CS165** and **USN 4BD21CS168** respectively of **Computer Science and Engineering** department have satisfactorily submitted the Mini project report entitled **“HUMAN ACTIVITY RECOGNITION SYSTEM”**. The report of the project has been approved as it satisfies the academic requirements in respect of project work prescribed for the academic year 2024.

**Project Guide Project Co-Guide:**

**Dr. Naveen H M** M.Tech., Ph.D **Prof. Mohamed Muthahir M. Tech.**

**Associate Professor Assistant Professor**

**Department of CS&E Department of CS&E**

**B.I.E.T., Davangere B.I.E.T., Davangere**

#### Head of Department

**Dr. Nirmala C.R M. Tech., Ph. D**

**Dept. Head & Placement Dean Department of CS&E B.I.E.T., Davangere**

#### Date: Signature of Examiners:

**Place: Davangere (1)**

**(2)**

## ACKNOWLEDGEMENT

Salutations to our beloved and highly esteemed institute, **“BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY”** for having well qualified staff and labs furnished with necessary equipments.

We express our sincere thanks to our guide **Dr. Naveen H M** and co-guide **Prof. Mohamed Muthahir** for giving us constant encouragement, support and valuable guidance throughout the course of the project without whose stable guidance this project would not have been achieved.

We express whole hearted gratitude to **Dr. Nirmala C R** who is our respectable HOD of Computer Science & Engineering Department. We wish to acknowledge her help who made our task easy by providing with his valuable help and encouragement.

We also express our whole hearted gratitude to our principal, **Dr. Aravind H B,** for his moral support and encouragement.

We would like to extend my gratitude to all staff of **Department of Computer Science and Engineering** for the help and support rendered to me. I have benefited a lot from the feedback, suggestions given by them.

We would like to extend my gratitude to all my family members and friends especially for their advice and moral support.

#### TASBIHA TAZEEN (4BD21CS165)

**THEJASWINI C.R. (4BD21CS168)**

|  |  |  |
| --- | --- | --- |
|  | **CONTENTS** |  |
| **CHAPTERS** |  | **PAGE NO’S** |
| **1. INTRODUCTION** |  | **01** |
| 1.1 Introduction to Python |  | 01 |
| 1.1.1 History of Python |  | 01 |
| 1.1.2 Features of Python |  | 02 |
| 1.2 Introduction to OpenCV |  | 04 |
| 1.2.1 Features of OpenCV |  | 04 |
| 1.2.2 Applications of OpenCV |  | 05 |
| 1.2.3 OpenCV Packages |  | 05 |
| **2. SYSTEM REQUIREMENTS** |  | **07** |
| 2.1 Software Requirements |  | 07 |
| 2.2 Hardware Requirements |  | 07 |
| **3. SYSTEM DESIGN** |  | **08** |
| 3.1 Methodology |  | 08 |
| **4. IMPLEMENTATION** |  | **11** |
| 4.1 Installation steps |  | 11 |
| 4.2 User Interface |  | 11 |
| 4.3 Module Implementation |  | 12 |
| **5. RESULTS** |  | **14** |
| **CONCLUSION** |  | **16** |
| **REFERENCES** |  | **17** |

# ABSTRACT

This project involves developing a human activity recognition system using a pre-trained deep learning model (ResNet-34) and OpenCV. The system captures frames from a video file or webcam, processes them into a format suitable for the model, and predicts the ongoing human activity by leveraging a deque to maintain a sequence of frames. The recognized activity is displayed on the video feed in real-time using matplotlib for visualization, enabling seamless integration and real-time activity monitoring. This approach facilitates practical applications in surveillance, health monitoring, and human-computer interaction.

# CHAPTER 1: INTRODUCTION

### Introduction to Python

**Python Definition**

Python is an interpreted, high-level, general-purpose (designed to be used for writing software in the widest variety of application domains) programming language.

### History

Python is a high-level programming language that was conceived in the late 1980s and first released in 1991 by Guido van Rossum. Here’s a brief history of how Python came to be and evolved over the years:

**Origin:** Python’s development began in December 1989 when Guido van Rossum, then working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands, started work on a successor to the ABC programming language.

**First Release:** The first public release, Python 0.9.0, came in February 1991. It included features like exception handling, functions, and the core data types of list, dict, str, and more.

#### Python 2 Era:

**Python 2.x:** Following the initial release, Python 2.0 was released in 2000, introducing features like garbage collection and support for Unicode. The Python 2 series continued to evolve with incremental updates, with Python 2.7 being the final release of the 2.x branch, supported until January 1, 2020.

#### Python 3 Transition:

**Python 3.x:** Python 3.0, released in 2008, marked a significant overhaul aimed at addressing fundamental design flaws and inconsistencies in the language. It introduced backward in compatible changes to improve consistency, simplicity, and future maintainability.

**Adoption Challenges:** The transition from Python 2 to Python 3 faced challenges due to compatibility issues, which led to a period where both versions coexisted in production environments.

#### Recent Developments:

**Python 3.x Series:** Since Python 3.0, the language has continued to evolve with regular updates, improving performance, adding new features, and refining existing ones. Notable releases include Python 3.5 (2015), Python 3.6 (2016), Python 3.7 (2018), Python 3.8 (2019), Python 3.9 (2020),

and Python 3.10 (2021).

**Community and Ecosystem Growth:** Python’s popularity has grown significantly over the years, supported by a vibrant community contributing libraries, frameworks, and tools across various domains such as web development, data science, machine learning, and more.

### Features of python

#### Easy to Learn and Use:

Python’s syntax is designed to be intuitive and readable, making it easy for beginners to grasp. It emphasizes code readability and allows developers to express concepts in fewer lines of code compared to other languages.

#### Expressive Language:

Python enables developers to write clear and logical code for both small and large-scale projects. It supports multiple programming paradigms including procedural, object-oriented, and functional programming.

#### Interpreted and Interactive:

Python is an interpreted language, meaning code is executed line by line, which allows for rapid prototyping and debugging. It also supports an interactive mode where you can test code snippets and get immediate feedback.

#### Large Standard Library:

Python comes with a broad range of modules and libraries, known as the Python Standard Library, which provides ready-to-use tools for tasks such as file I/O, string manipulation, operating system interfaces, and more. This reduces the need for external libraries for many common tasks.

#### Third-Party Libraries and Frameworks:

In addition to the standard library, Python has a rich ecosystem of third-party libraries and frameworks that extend its functionality. Examples include Django and Flask for web development, NumPy and Pandas for data analysis, TensorFlow and PyTorch for machine learning, and many more.

#### Open Source and Community-driven:

Python is developed under an open-source license, fostering a supportive and active community. This community contributes to the language’s growth by creating libraries, frameworks, and tools, and providing support through forums, conferences, and online resources.

#### Portability:

Python runs on various platforms including Windows, macOS, Linux, and Unix. This portability makes it suitable for developing applications that can run on different systems without modification.

#### Integration Capabilities:

Python can integrate with other languages such as C, C++, and Java via modules and libraries, allowing developers to leverage existing code and tools.

#### Scalability:

While Python is often used for small to medium-sized projects, it is also capable of scaling up to handle complex applications. This scalability is supported by its robust standard library, thirdparty frameworks, and tools for profiling and optimizing code.

#### Versatility:

Python is used in various domains including web development, scientific computing, data analysis, artificial intelligence, machine learning, automation, scripting, and more. Its versatility and ease of use make it a preferred choice for diverse applications

### Introduction to OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source library for computer vision and image processing. Originally developed by Intel, it is now maintained by the OpenCV community. It provides a comprehensive set of tools and functions for handling and manipulating images and performing various computer vision tasks.

* + 1. **Features of OpenCV:**
       1. **Cross-platform:** OpenCV is compatible with various operating systems including Windows, Linux, macOS, Android, and iOS. This makes it highly versatile for developing applications across different platforms.
       2. **Extensive Functionality:** OpenCV offers a vast array of functions and algorithms for performing tasks such as image and video processing, feature detection, object recognition, machine learning, camera calibration, and more.
       3. **Support for Multiple Languages:** While OpenCV is primarily implemented in C++, it also provides bindings for Python, Java, and MATLAB, making it accessible to developers using different programming languages.
       4. **Efficiency:** OpenCV is optimized for real-time applications and is known for its high computational efficiency. It leverages hardware acceleration (like SIMD instructions) and supports multi-threading to maximize performance.
       5. **Community and Development:** OpenCV is maintained by a large community of developers and researchers worldwide. Contributions from the community ensure continuous improvement, bug fixes, and the addition of new features.
       6. **Modules and Libraries:** OpenCV is modularized into different libraries and modules, each focusing on specific tasks or functionalities. Some key modules include:
          - **Core:** Basic data structures, matrix operations, and fundamental functions.
          - **Imgproc:** Image processing operations such as filtering, edge detection, and morphological operations.
          - **Video:** Video analysis, optical flow, background subtraction.
          - **Features2D:** Feature detection (e.g., keypoints) and description algorithms (e.g., SIFT, SURF, ORB). o **Objdetect:** Object detection using Haar cascades, HOG (Histogram of Oriented Gradients), etc.
          - **Machine Learning:** Algorithms for machine learning tasks like clustering, classification, and regression. o **Calib3d:** Camera calibration and 3D reconstruction.

### Applications of OpenCV:

* **Computer Vision:** OpenCV is widely used for tasks such as face detection and recognition, object tracking, gesture recognition, motion analysis, and augmented reality.
* **Image Processing:** It provides numerous functions for basic to advanced image processing tasks like image filtering, transformation, enhancement, and segmentation.
* **Machine Learning:** OpenCV integrates with machine learning libraries like TensorFlow and PyTorch for tasks such as image classification, object detection, and image generation.
* **Robotics:** OpenCV plays a crucial role in robotics applications, including autonomous navigation, object manipulation, and environment mapping.
* **Medical Imaging:** It is utilized for tasks such as medical image analysis, pathology detection, and diagnostic assistance.
* **Industrial Applications:** OpenCV is applied in industrial automation for quality control, defect detection, object sorting, and process monitoring.

### OpenCV Packages

OpenCV (Open Source Computer Vision Library) provides a wide range of modules and packages that cater to different aspects of computer vision, image processing, and machine learning tasks. These packages encapsulate various functionalities and algorithms, making OpenCV a versatile tool for developing vision- based applications. Here are some key packages/modules provided by OpenCV:

* + - 1. **Core (cv2 module):**
         * **cv2.imread(), cv2.imwrite():** Functions for reading and writing images.

**cv2.VideoCapture():** Interface for capturing video from cameras or files.

* + - * + **cv2.VideoCapture() and cv2.VideoWriter():** Classes for handling video streams and writing videos.

#### ImageProcessing Imgproc (cv2 module):

* + - * + **cv2.cvtColor():** Convert between color spaces (e.g., BGR to grayscale).
        + **cv2.GaussianBlur(), cv2.medianBlur():** Apply Gaussian or median blurring to images.
        + **cv2.Canny():** Perform Canny edge detection. o **cv2.threshold():** Apply thresholding to an image. o **cv2.resize(), cv2.rotate(), cv2.flip():** Resize, rotate, and flip images.
        + **cv2.warpAffine(), cv2.warpPerspective():** Perform affine and perspective transformations.
        + **cv2.matchTemplate():** Template matching for finding instances of a template image within a larger image.

#### Feature Detection and Description Features2D (cv2 module):

* + - * + **cv2.SIFT\_create(), cv2.SURF\_create(), cv2.ORB\_create():** Create instances of feature detection algorithms like SIFT, SURF, and ORB.
        + **cv2.drawKeypoints(), cv2.drawMatches():** Draw key points or matches between two images.

#### Python Bindings and Utilities Python (cv2 module):

* + - * + **cv2.waitKey(), cv2.destroyAllWindows():** Manage user interface events like key presses and window destruction.
        + **cv2.imwrite(), cv2.imshow():** Display and save images.

# CHAPTER 2 :SYSTEM REQUIREMENTS

### Software Requirements

* + 1. Operating System : Microsoft Windows 11
    2. Compiler used: PyCharm
    3. Language used: Python

### Hardware Requirements

* + 1. Main Processor: intel core i5
    2. Processor Speed: 3 GHz or more
    3. RAM Size: 4GB

# CHAPTER 3 : SYSTEM DESIGN

### 3.1 Methodology:

Load Class Labels and Models

Open Camera and Capture Frames

Display Frame and Set Callback

­­­­­­­

Process Frame and Check Buffer

Buffer Full: Predict,Annotate,Display Activity

Fig 3: Methodology diagram for Human Activity Recognition System

The methodology for this project involves several key steps:

1. **Initialize Parameters:**

* Create a “Parameters” class to define and store constants and important paths, such as the path to the pre-trained model and the classes file.

1. **Load Class Labels**

* Read the class labels from a text file containing the names of activities that the model can recognize.

1. **Load the Pre-trained Model**

* Use OpenCV's “cv2.dnn.readNet” to load the pre-trained ResNet-34 model for human activity recognition from an ONNX file.

1. **Setup Video Capture**

* Initialize video capture using OpenCV's “cv2.VideoCapture” to read from a specified video file or webcam.

1. **Frame Buffer Initialization**

* Use a double-ended queue (deque) to store a sequence of captured frames. The deque has a fixed size defined by “SAMPLE\_DURATION” to hold the required number of frames for model input.

1. **Frame Capture and Preprocessing**

* Continuously capture frames from the video source.
* Resize each frame to a consistent size.
* Append each captured frame to the deque.
* Ensure that the deque is fully populated before processing.

1. **Blob Construction**

* Convert the sequence of frames in the deque to a blob using “cv2.dnn.blobFromImages”, which involves resizing, normalizing, and reordering the frames to match the model's input requirements.

1. **Model Prediction**

* Set the constructed blob as the input to the neural network.
* Perform a forward pass through the model to get predictions.
* Determine the activity label with the highest probability from the model's output.

1. **Display Results**

* Annotate the captured frame with the predicted activity label.
* Display the annotated frame using matplotlib for visualization.

1. **Keyboard Input Handling**

* Implement a method to check for keyboard input using “plt.waitforbuttonpress” to allow the user to exit the loop by pressing the 'q' key.

1. **Resource Cleanup**

* Release the video capture object and close any open windows once the loop is exited.

This structured approach ensures real-time human activity recognition with efficient processing and visualization, facilitating applications in various domains.

# CHAPTER 4 : IMPLEMENTATION

### Installation Steps:

To install and run the project:

#### Prerequisites:

* **Python Environment**: Ensure you have Python installed.
* **Project Directory**: Download or clone the project directory (e.g., human-activity-recognition).

#### Steps:

* + - **Download or Clone**: Download or clone your project directory “human-activity-recognition”.
    - **Install Required Libraries**: Install necessary libraries if not already installed:

**pip install opencv-python numpy matplotlib**

* + - **Ensure Files**: Make sure the “resnet-34\_kinetics.onnx” and “action\_recognition\_kinetics.txt” files are in the correct directory.

#### Execution:

* + - **Run Script**: Execute the script to start human activity recognition:

**python recognise\_human\_activity.py**

### User Interface:

The user interface (UI) of Human Activity Recognition project consists of:

#### Main Window (Human Activity Recognition):

* **Functionality:**
  + Displays the real-time video feed from the specified video file or the default camera **(cv2.VideoCapture(param.VIDEO\_PATH if param.VIDEO\_PATH else 0)).**
  + Users interact by viewing the live feed with the predicted human activity labels overlayed on the video frames.

#### Display Logic:

* Implemented using matplotlib for displaying frames with ax.imshow.
* Annotations for detected activities are added using OpenCV's cv2.rectangle and cv2.putText functions.

**4.2.2 Execution Window (Console Output):**

* **Functionality:**
* Logs the system status messages such as loading the model and accessing the video stream
* Outputs information about the detected activities for each frame.

### Module Implementation:

#### 4.3.1 Data Handling and Preprocessing:

* **Data Loading:** Uses OpenCV and Python's file handling to load the class labels for human activity recognition from a text file (action\_recognition\_kinetics.txt).
* **Preprocessing:** Extracts frames from the video stream and preprocesses them by resizing and normalizing to construct a blob suitable for input to the pre-trained ResNet-34 model.

#### 4.3.2 Machine Learning Model:

* **Model Selection:** Utilizes a pre-trained ResNet-34 model for human activity recognition, loaded using OpenCV's cv2.dnn.readNet function.
* **Model Inference:** Performs forward pass through the ResNet-34 model to predict the human activity based on the preprocessed sequence of frames.

#### 4.3.3 Real-Time Video Processing:

* **Video Capture:** Utilizes OpenCV's cv2.VideoCapture to access the video feed from a specified video file or the default camera.
* **Frame Buffer:** Maintains a sequence of frames using a deque to ensure the model receives a continuous stream of images for accurate activity recognition.

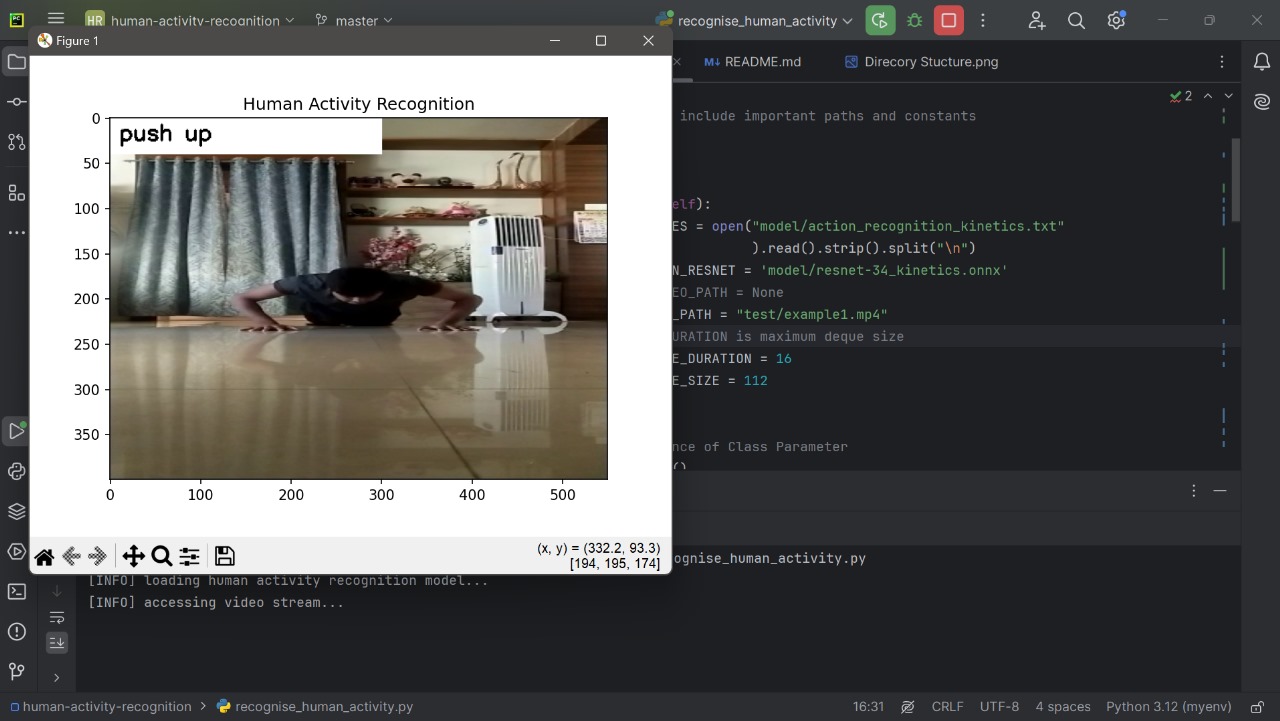
#### 4.4.4 Activity Recognition and Display:

* **Activity Prediction:**
  + Constructs an image blob from the deque of frames and sets it as input to the model.
  + Performs a forward pass through the model to predict the current activity.
  + Annotates the frames with the predicted activity label using OpenCV's cv2.rectangle and cv2.putText.

**4.4.5 Display:**

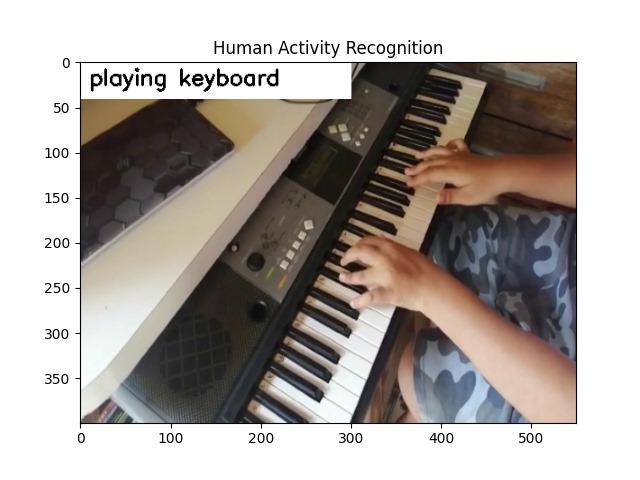
* Displays the annotated video feed using matplotlib's imshow function to render frames with the predicted activity labels.
* Provides a real-time user interface for viewing the video feed with activity annotations.

# CHAPTER 5 : RESULTS



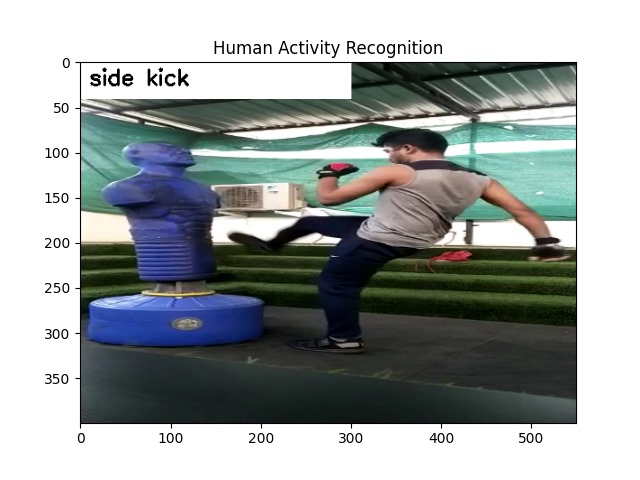
**Fig 5.1:** The image shows a computer program running in a development environment, displaying a video feed with the label "push up" indicating real-time human activity recognition.

The screen captures a person performing push-ups, with the activity accurately identified and annotated on the video frame.



**Fig 5.2:** The image displays a video frame where a person is playing a keyboard, with the activity recognition system correctly identifying and labeling the action as "playing keyboard."

The labeled activity is shown at the top of the frame in a graphical user interface.



**Fig 5.3:** The image captures a video frame where a person is performing a side kick on a training dummy, with the human activity recognition system correctly identifying and labeling the action as "side kick."

The labeled activity is displayed at the top of the frame in a graphical interface.

## CONCLUSION

The human activity recognition project effectively harnesses the power of a pre-trained ResNet-34 model along with OpenCV to accurately detect and display real-time human activities from video feeds. This implementation showcases the practical utility of deep learning and computer vision in various domains, including surveillance for enhanced security, health monitoring for tracking physical activities and detecting anomalies, and human-computer interaction for creating more intuitive and responsive systems. By continuously capturing and processing video frames, the system identifies activities such as running, walking, and jumping, providing instant feedback and visual annotations. This project not only demonstrates the potential of integrating advanced neural networks with real-time video processing but also opens avenues for future enhancements and applications in diverse fields requiring accurate activity recognition and monitoring.

## REFERENCES

1. **"Programming Computer Vision with Python" by Jan Erik Solem:** This book covers practical computer vision techniques using Python and OpenCV, including image processing and machine learning applications.
2. **"OpenCV 4 for Secret Agents" by Joseph Howse:** Focuses on practical applications of OpenCV for real-world projects, including image and video processing tasks.
3. **“ResNet-34 Architecture” by He , K, Zhang , X ,Ren , S ,&Sun, J:** Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
4. **“Video Analytics Using OpenCV and Python Shells” by Kushal Vyas**: This book focuses on the application of OpenCV in real-time video analysis using Python. It covers various aspects of video analytics such as motion detection, object tracking, and activity recognition, providing readers with the skills needed to develop sophisticated video surveillance systems.