Project Report

Indian Institute of Information Technology, Kalyani

MACHINE LEARNING (CSC-602 ECE)

Gesture Controlled Robot

Group Members:

- **Shirish Manoj Bobde (812)**
- **Anubhav Ranjan (774)**

Project Details

Project Description

Creating a simulated two-wheel robot controlled by hand gestures using CNN prediction. Challenges include accurate gesture recognition, real-time processing, and model integration with simulation software. Proposed solution involves data collection, CNN model training, simulation setup, real-time execution, and testing. Outcomes aim for a responsive simulation enhancing human-robot interaction understanding in various applications.

Model

Convolutional Neural Network (CNN) model using the Keras library is used to perform gesture recognition. Here's a breakdown of the main components and steps:

Data Preparation:

- The script uses Keras' ImageDataGenerator to preprocess and load image data from directories. It rescales the pixel values to the range [0, 1] and loads the training and validation datasets.

Model Architecture:

- The CNN model consists of several convolutional layers followed by max-pooling layers and dropout layers to prevent overfitting. The architecture progressively extracts features from the input images.
- The final layers are fully connected (dense) layers, which perform classification based on the extracted features.
- The model ends with a softmax activation function in the output layer, which predicts the probability distribution over the different classes (gestures).

Model Training:

- The model is compiled using the Adam optimizer and categorical cross-entropy loss function.
- The script trains the model using the fit() function, specifying the training and validation generators, number of epochs, and batch size.
- It also includes a callback for TensorBoard, which enables visualization of training metrics.

Model Evaluation:

- The script uses TensorBoard to visualize training and validation metrics such as loss and accuracy.

Model Saving:

- After training, the script saves the model's architecture (Gesture_Detection_48x48.json) and its weights (Gesture_Detection_Model48x48.h5) to Google Drive.

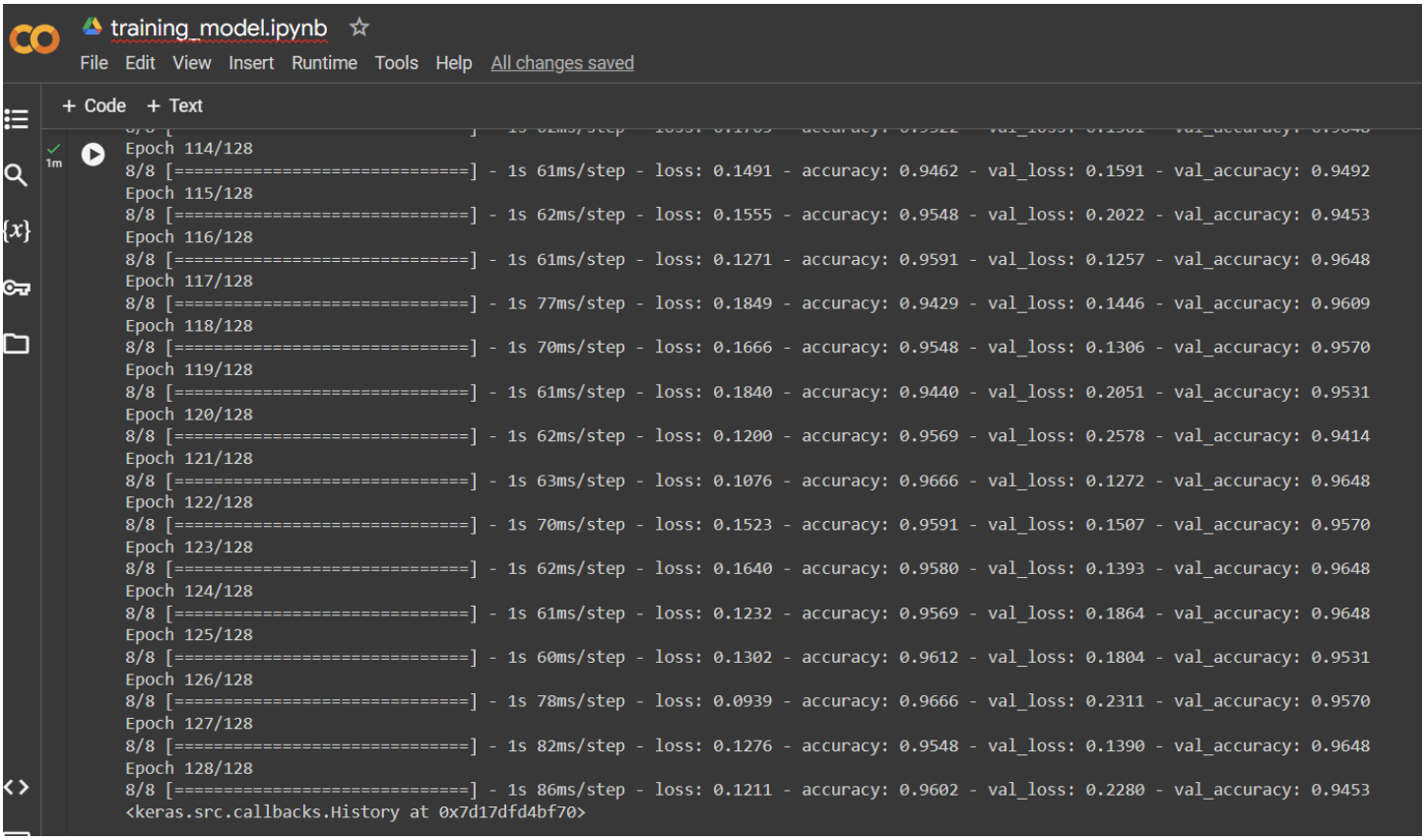
TensorBoard Visualization:

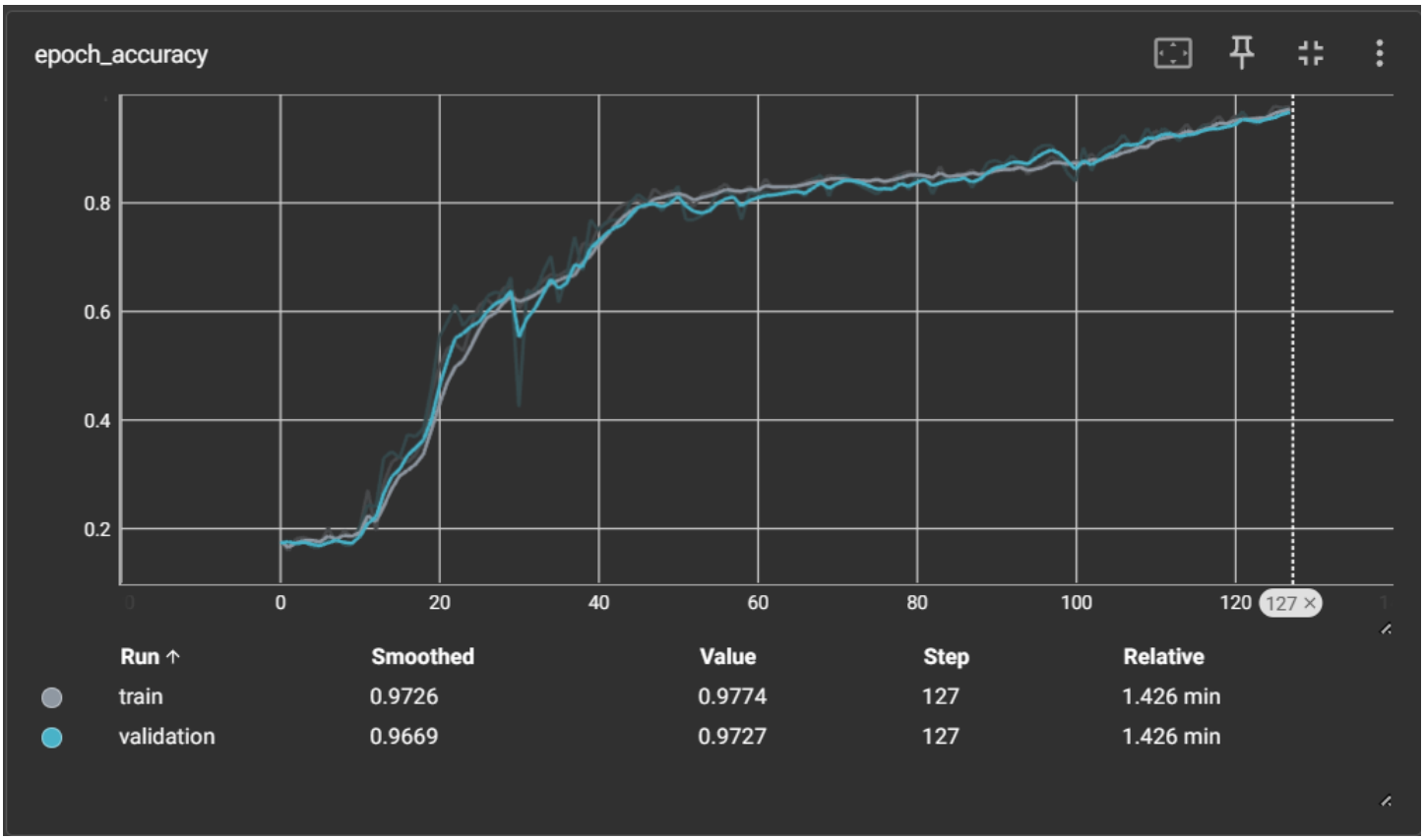
- The script loads the TensorBoard extension to enable visualization within the Colab environment.
- It launches TensorBoard to visualize the training logs stored in the specified directory.

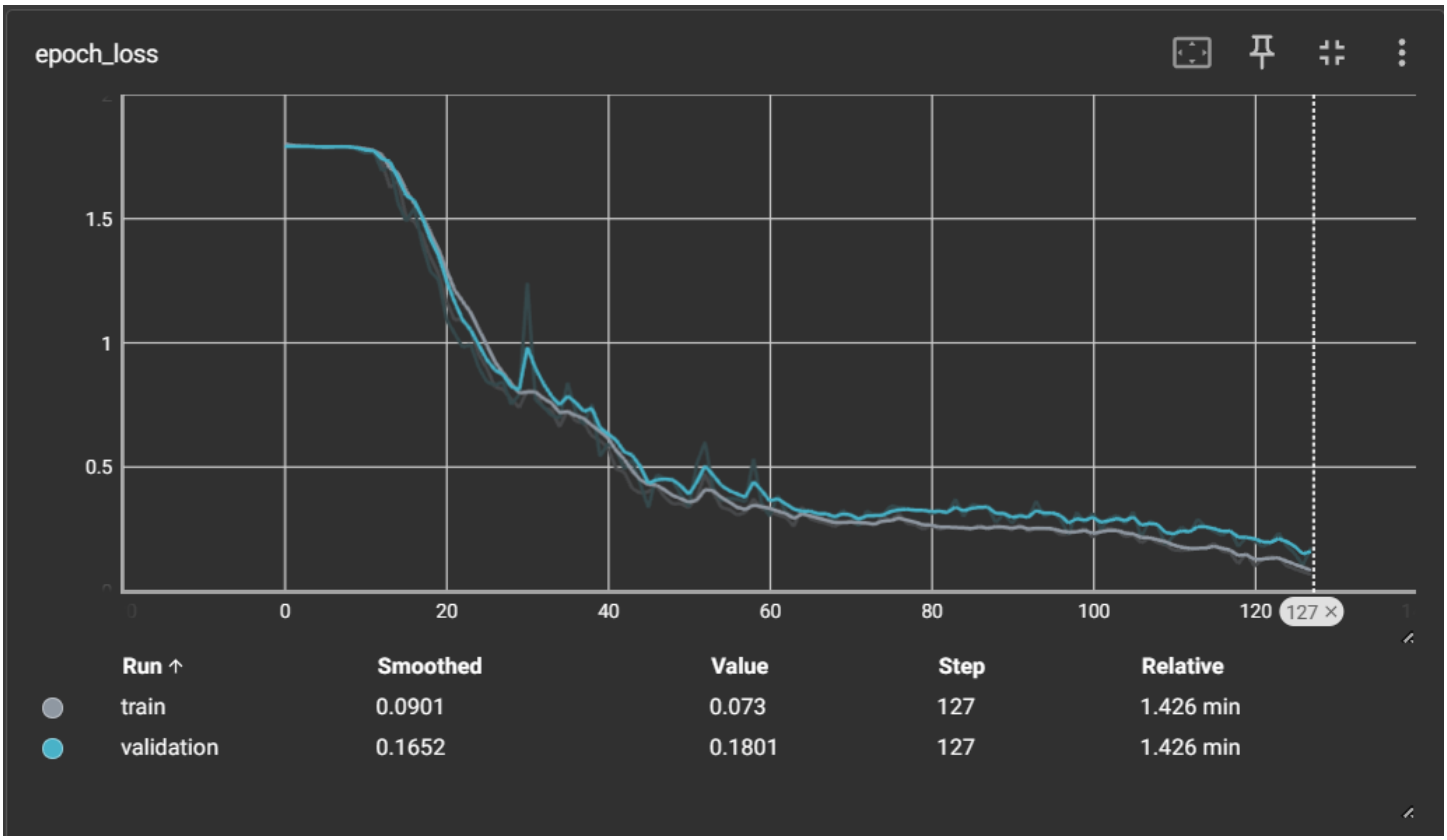
Overall, this script trains a CNN model for gesture recognition using image data, evaluates its performance, and saves it for future use.

Output

Model Training:



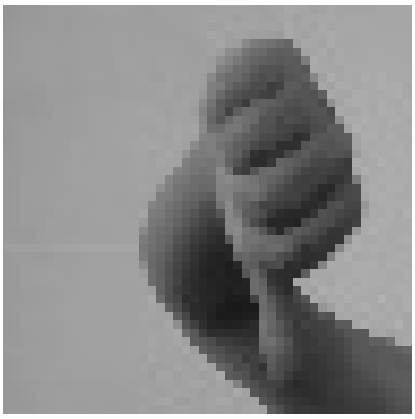




Classes:

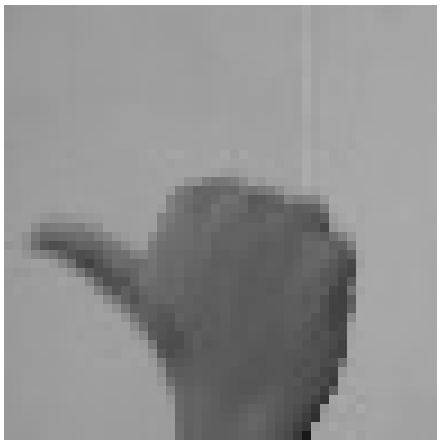
Forward

Backward



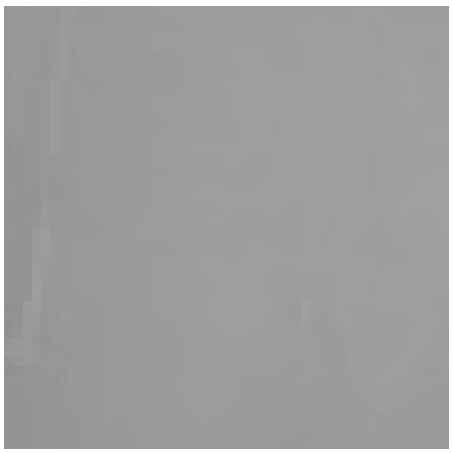
Left

Right



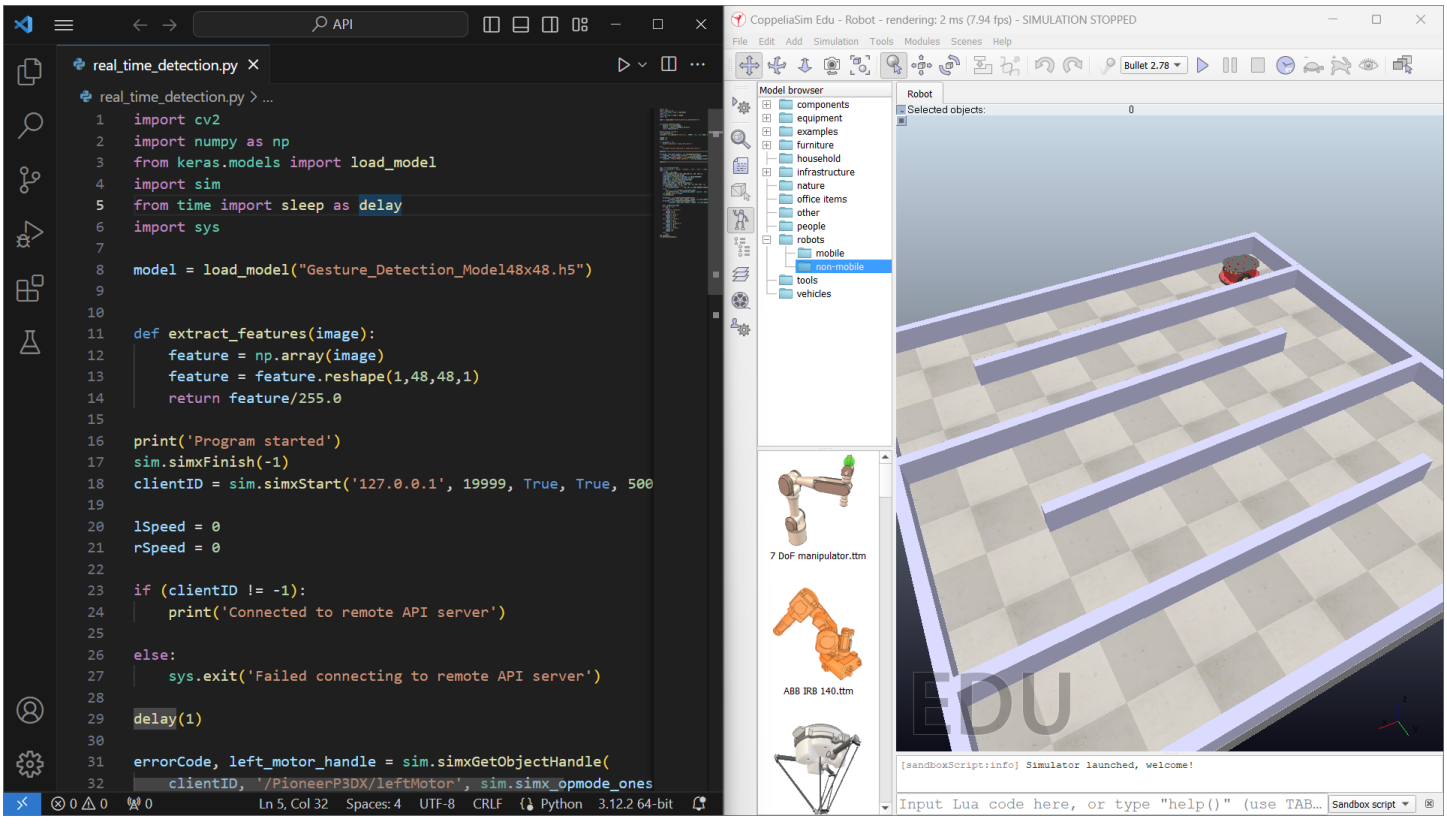
Brake

Null

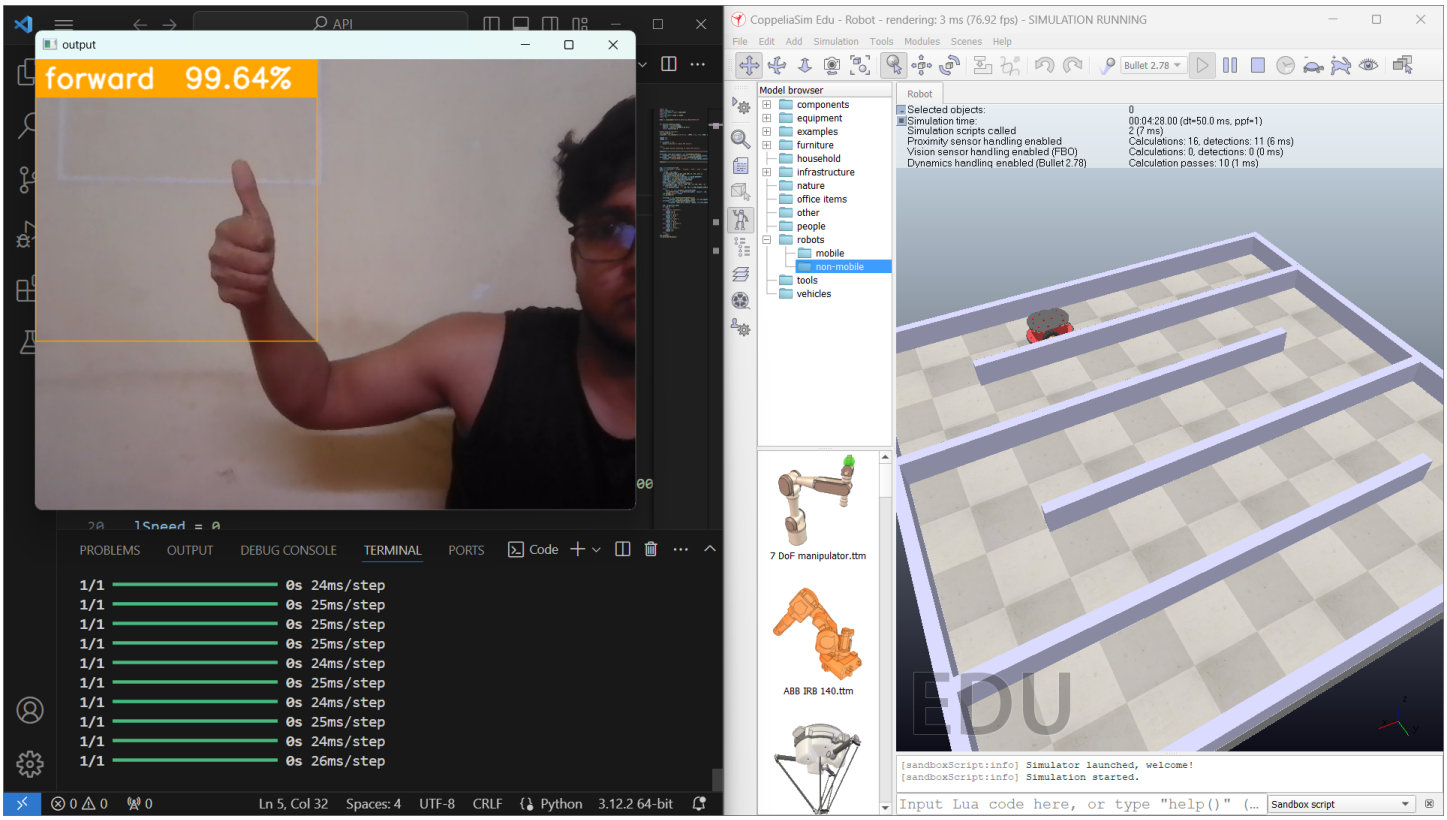


Real Time Robot Control:

Before



After



Discussion

The development process outlined presents a comprehensive approach to implementing gesture-controlled navigation for a two-wheel vehicle, leveraging machine learning and computer vision technologies. By breaking down the process into distinct stages, from data collection to model training, simulation setup, and control implementation, the project ensures a systematic and efficient workflow.

Data collection and preprocessing lay the groundwork for training a robust machine learning model capable of accurately recognizing gestures. Through scripts like "collectdata.py" and "split.py," developers can organize and partition gesture data, essential for training and validating the model's performance.

By harnessing the computational power of Google Colab, you maximize efficiency and scalability, enabling seamless model training and validation. The resulting model, encapsulated in files like "Gesture_Detection_48x48.json" and "Gesture_Detection_Model48x48.h5," serves as the cornerstone for gesture recognition.

The integration of CoppeliaSim simulation software facilitates realistic testing and validation of the gesture-controlled system. By designing the vehicle within CoppeliaSim and establishing communication through APIs and "remoteApi.dll," simulated real-world scenarios and refine the system's functionality in a controlled environment.

The culmination of these efforts is realized in scripts like "real_time_detection.py," where the trained machine learning model is deployed in real-time to interpret gestures from camera input and translate them into actionable commands for the two-wheel vehicle. Through meticulous testing and validation, developers can ensure the system's responsiveness, accuracy, and robustness in diverse scenarios.

Overall, the project exemplifies the synergistic integration of machine learning, computer vision, and simulation technologies to achieve gesture-controlled navigation for a two-wheel vehicle. By following a structured development process and leveraging cutting-edge tools and methodologies, developers can create intuitive and immersive human-machine interaction experiences.

Github Link: https://github.com/CodeWizard812/Gesture-Controlled-Robot.git