JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AIR GESTURE KEYBOARD FOR VISUALLY IMPAIRED PEOPLE USING MACHINE LEARNING APPROACH

NANDA B S[1*], ANKITHA H R[2], ARUN KUMAR V[2], SAHANA H S[2], SHRADHA HOLAGI[2].

[1*] ASSISTANT PROFESSOR, DEPT OF ELECTRONICS AND COMMUNICATION ENGINEERING, PES COLLEGE OF ENGINEERING, MANDYA, KARNATAKA, INDIA-571401

[2] STUDENT, DEPT OF ELECTRONICS AND COMMUNICATION ENGINEERING, PES COLLEGE OF ENGINEERING, MANDYA, KARNATAKA, INDIA-571401

Abstract : The article discusses the development of an air gesture keyboard using a machine learning approach to assist visually impaired individuals. The keyboard utilizes a remote-based system that translates hand gestures into text through an accelerometer, Arduino, and switches. It eliminates the need for physical keyboards and supports multi-linguistic functionality.

The article begins with an introduction highlighting the significance of keyboards in data entry and the prevalence of touch and gesture keyboards. It emphasizes the need for alternative input methods for visually impaired individuals and the limitations of existing technologies. The proposed remote-based gesture keyboard enables text input by capturing hand motions in the air, with the remote acting as a pen and a text editor as a notebook.

A literature survey is conducted to explore related research on hand gesture recognition using different techniques such as digital cameras, EMG signals, and sensor-based approaches. The technical requirements for the air gesture keyboard system are then discussed, including the accelerometer MPU 6050, Arduino UNO, button switch, and Bluetooth device.

The system design is presented, illustrating the circuit diagram and the creation of gesture recognition software using the Scikit-learn library. The article also discusses the various stages of the machine learning algorithm, including dataset creation, data pre-processing, training set formation, and algorithm selection. Classification algorithms, specifically Support Vector Machines (SVM), are evaluated for gesture recognition.

Overall, the proposed air gesture keyboard offers a novel and efficient input method for visually impaired individuals, allowing them to enter text without physical keyboards. The system's machine learning capabilities enhance accuracy and versatility, making it a promising assistive technology for visually impaired individuals.

***Index Terms* - : machine learning, gesture, Microcontroller, visually impaired**

I.INTRODUCTION

The keyboard has long played a vital role in computer systems, allowing users to input data by pressing various keys. In recent times, touch screen keyboards have become predominant, with gesture keyboards primarily used for physically disabled individuals and specialized applications. Numerous variations in physical keyboard layouts exist, including AZERTY, QWERTY, Dvorak, Colemak, Maltron, and JCUKEN [1]. However, alongside physical keyboards, virtual keyboards have gained prominence due to the proliferation of mobile and networking devices. While speech-to-text technology has gained popularity, its accuracy often falls short of expectations. To address this, the use of remote-based gesture keyboards, driven by machine learning algorithms and implemented using Python programming language, has emerged [2]. These keyboards interpret gestures captured by accelerometers in the air and convert them into corresponding text, eliminating the need for a specific layout and facilitating multi-linguistic functionality [2].

To construct such a system, a motion tracking device is crucial, comprising three major components: an accelerometer, an Arduino microcontroller, and switches [3]. The Arduino serial monitor is configured, setting the baud rate to 38400 within the Arduino IDE [3]. The entire module leverages the scikit-learn library, which converts signals from the accelerometer into letters, with each character and digit stored in a dataset. Once the dataset is prepared, the module is trained using machine learning algorithms to recognize gestures in different languages [3].

The remote-based gesture keyboard provides several advantages over traditional keyboards, addressing issues related to disabilities, accuracy, layout limitations, and durability. It can be connected to devices either wirelessly, through Bluetooth, or through a wired connection using USB [4]. This innovative concept enables users to input text and numbers into a text editor by

moving their hand in specific character motions in the air, facilitated by Arduino as a remote. The corresponding letters are then displayed on a computer screen, mimicking the experience of writing in a notebook using a pen [4].

II. LITERATURE SURVEY

Hand gestures provide a natural and intuitive communication modality for human-computer interaction (HCI) [1]. In order to facilitate efficient HCI, there is a need to develop computer interfaces that can visually recognize hand gestures in real time. One approach explored by Shakunthaladevi and Revathi involved real-time hand gesture recognition using an AVR Microcontroller [5]. The traditional method employed a digital camera as an input device, which had limitations in terms of mobility, low usability in dark areas, and high cost. To address these issues, the researchers proposed a system that utilized MEMS accelerometer sensors to measure acceleration, specifically tilt, shock, and vibration. The tilt motion information generated by human subjects was transmitted to the AVR microcontroller, resulting in reliable and accurate gesture detection.

Another avenue of research focuses on real-time hand gesture recognition using electromyography (EMG) and machine learning techniques [6]. Jaramillo and Benalcázar highlighted the challenges associated with recognizing gestures based on EMG signals, given the underlying physiological processes in the skeletal muscles. Their objective was to develop a real-time gesture recognition model that could surpass existing models in terms of recognition accuracy and the number of gestures it could recognize. Their proposed model consisted of five stages: EMG signal acquisition, preprocessing (including rectification and filtering), feature extraction (such as time, frequency, and time-frequency), classification (parametric and nonparametric), and post-processing. The researchers emphasized the difficulties posed by the noisy behavior of EMG signals and the curse of dimensionality due to the limited number of gestures per person compared to the generated data. They recognized that overcoming these challenges could have broader implications for domains like face recognition and audio recognition.

Additionally, Tai, Jhang, Liao, Teng, and Hwang introduced a sensor-based continuous hand gesture recognition algorithm that utilized long short-term memory (LSTM) [7]. This algorithm only required basic accelerometers and/or gyroscopes. By employing a many-to-many LSTM scheme, the algorithm could generate an output path based on a sequence of input sensory data. Subsequently, a maximum a posteriori estimation was conducted on the observed path to achieve the final classification results. The researchers implemented a prototype system using smartphones to evaluate the algorithm's performance. Experimental results demonstrated that the proposed algorithm offered a robust and accurate solution for hand gesture recognition.

These studies collectively contribute to the field of hand gesture recognition by proposing novel approaches and addressing various challenges. By leveraging AVR Microcontrollers, EMG signals, and LSTM algorithms, researchers aim to improve the real-time recognition of hand gestures, providing more natural and efficient human-computer interaction.

III. TECHNICAL REQUIREMENTS

3.1 Accelerometer Mpu 6050

Fig.1 Accelerometer Mpu 6050

The MPU-6050 is a motion processing device that combines a 3-axis accelerometer with a 3-axis MEMS gyroscope on a single chip, as shown in Figure 1. Additionally, it has a 6-axis Motion Fusion algorithm-capable Digital Motion Processor built in. The MPU6050 sensor module, which includes a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor in a small package, functions as a comprehensive 6-axis Motion Tracking Device. It also has a temperature sensor built into the device and communicates with microcontrollers via the I2C bus interface. The module also has an Auxiliary I2C bus, which makes it easier to communicate with other sensor devices like pressure sensors and 3-axis magnetometers. The MPU6050 can produce an entire 6-axis Motion Fusion output when a 3-axis magnetometer is connected to the auxiliary I2C bus.

3.2 Arduino UNO

Fig.2 Arduino UNO

The Uno, a microcontroller board based on the ATmega328P microprocessor, is seen in Figure 2. This board features a 16 MHz quartz crystal, 14 digital input/output pins, 6 analogue input pins, a USB port, a power connector, an ICSP header, and a reset button. The microcontroller is essential to this project since it processes all sensor data. Its main job is to transform sensor data from raw readings into information that can be used by the computer system. The purpose of using the microcontroller is to provide a communication channel so that the sensor data can be effectively processed and used.

3.3 Button Switch

Fig .3 Button Switch

During the design process, we encountered a critical decision regarding how to notify the microprocessor of a sign being made. One potential solution involved implementing a solitary push button, as illustrated in Figure 3, which would be pressed to initiate the system's prediction functionality.

3.4 Bluetooth Device

Fig.4 Bluetooth Device

A Bluetooth device is used to transmit generated patterns from mpu6050 to the system and receive data to the system.

IV. SYSTEM DESIGN

I.1. Circuit Diagram

As a means of communication between a monitor and a remote device, the HC-06 module will provide a wireless link between an Arduino Pro Micro and an accelerometer. The SDA and SCL pins on the Arduino are used to connect it to the accelerometer, and it is grounded to provide power to the complete system. Motion motions in the x and y directions are captured by the accelerometer. The SciKit-Learn package in Python's PyGARL framework is used for motion recognition. As a Python gesture analysis and recognition library, PyGARL enables text-based data storage and the easy addition of a wealth of data to the library. The recognised gesture data is clustered and regressed using the support vector machine technique from the field of machine learning, and the results are then compared to the stored data. The matching data is shown on the screen once a match is discovered

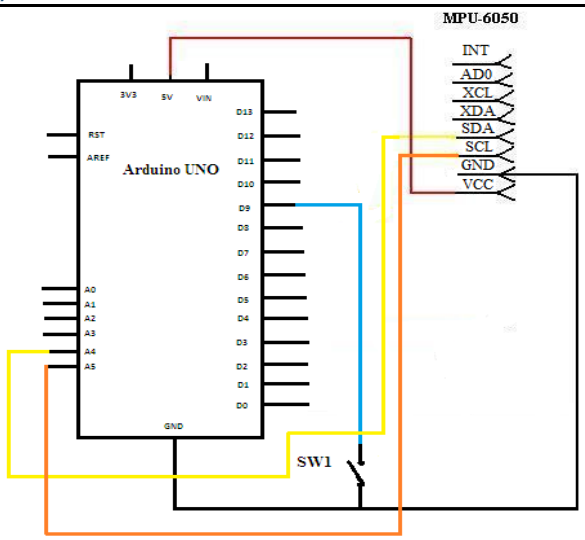

Fig.5 Circuit Diagram of Gesture Keyboard using Arduino

A special library called the Gesture Keyboard was created to translate accelerometer data into a string of letters and phrases. The difficulty occurs when dealing with air-writing, which might include hundreds of words, whereas gesture recognition normally uses a small vocabulary. It becomes challenging to get adequate information for each term in the lexicon. An Arduino module equipped with an MPU-6050 accelerometer is used to collect accelerometer data. The data from an accelerometer is sent to a computer through this module. On the computer, a Python package is used to categorise the signals into letters using Scikit-learn's "Support Vector Machine algorithm." This method is used by the Gesture Keyboard library to accurately read accelerometer data, allowing motions to be translated into text.

I.2. Creation of gesture recognition software

Scikit-learn is a programming library in machine learning for the programming in Python. It highlights distinctive characterization, backslide and clustering calculations counting reinforce vector machines, self-assertive forests, slant boosting and is aiming to interoperate with the Python numerical and coherent libraries .Gesture Recognition or any other machine learning tasks that uses supervised learning goes through a process of stages before the final classifier produces an efficient output. All the stages are indecent of each other and play a unique role in contributing to the accuracy of classification.

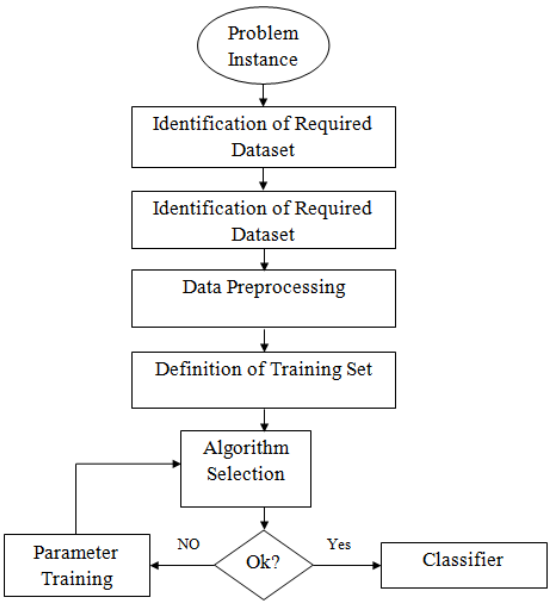

Fig .6 Flow diagram for machine learning algorithm

- Required dataset The first step is to fabricate the data attributes that are crucial in determining the goal of the output. All values must have a constraint or an upper and lower bound limits to initiate remission of boundary limits. The dataset is then made available to be identified in terms of different parameters that can further extracted. The set is then made available to be identified in terms of different parameters that can further extracted.
- Data Pre-Processing: Researchers have found that the pre-processing of data differs based on the particular issue at hand. Data pre-processing includes instance selection, which helps to effectively learn from huge datasets while also addressing issues with noise management. An optimisation issue called instance selection seeks to decrease the sample size while maintaining the quality of mining. Data classification algorithms may successfully manage huge or heavy datasets thanks to

this data reduction. There are several techniques for selecting examples from a big collection.

- Training Set: Logical and constraint-satisfying training inputs may be acquired at each input cycle once a well specified input variable is globally available. The training set is made up of similar pieces that are organised into separate entities. The classification space's starting point, the training set, facilitates experimentation and trend monitoring. It is made up of attribute key-value pairs for various instances of the set.
- Algorithm Selection: The documents are split into training and testing sets in this stage. The system is trained using the training set so that it can recognise various patterns and classifications. However, the performance of the system is assessed using the testing set. The categorising process is significantly influenced by the algorithm of choice. A critical choice to make is the learning algorithm to use. An automated classifier that assigns unlabelled instances to classes is made available after preliminary testing and successful completion.

4.3 Classification Algorithms for Gesture Recognition

We will describe our data collection and goal values in detail in this part, and then we will assess how well the algorithm performs while teaching gestures.

Gesture Dataset: To improve precision, we'll generate a fresh collection of data and examine its patterns. Every time the accelerometer is moved or a new accelerometer is used, the device's precision gets less accurate. Given the time sequences involved, it is imperative to constantly contribute training data to the learning set. Our system can handle up to 40 different motions, each of which is connected to a single English alphabet letter. By pressing the online button, a fresh set of data inputs for gestures is first recorded. The following parameters are used by the library to record the accelerometer movements:

- Target: The module is told to record fresh gesture examples by this parameter.
- "a": Its length of 1 denotes the distinctive gesture.
- 0: Batch Number; in order to avoid overwriting, every new batch must have a unique number.
- Port: The port designates the connection to the Arduino or microcontroller's serial port.

The best practise is to record each sample and save it as a separate file in the data folder for future analysis. We choose a suitable model to train these inputs once the dataset is ready. In order to get the optimal model, we evaluate two of the most popular classification algorithms using our dataset. Choosing the ideal algorithm might be difficult.
Classification Algorithms: Classification algorithms use a group of variables, such as "feature variable," "independent variable," or "predictor variable," to produce predictions. The outcomes of the predictions are stored in the "class variable" or "dependent variable." The "class variable" connected to the input data—which may be categorical or continuous is the foundation of these algorithms. When the class label is categorical, "classification" techniques are used; when the class label is continuous, "regression" techniques are used. The choice of the classifier is made based on the algorithm's prediction accuracy, which is gauged by a variable called the "Confusion matrix." The Confusion matrix aids in measuring the proportion of test data that the classifier properly labels. The data's Supervised Classification is shown in Figure 7.

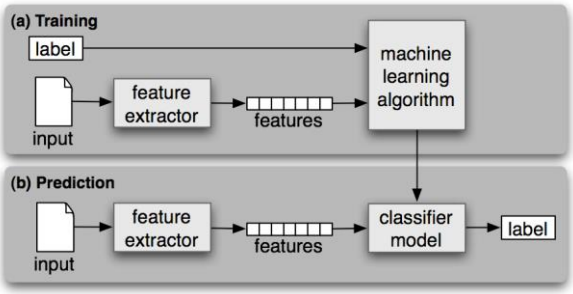

Fig.7 Supervised Classification of data

The foundation of Support Vector Machines (SVMs) is the idea of a "margin" enclosing a hyperplane that divides two classes of data. The goal is to maximise this margin, putting the hyperplane at a maximum distance from the instances on each side. Maximising the margin lowers the upper bound on the predicted generalisation error, as shown by mathematical analysis.

The 1995 discovery of Support Vector Machines (SVMs) led to its widespread use as a classification method. In a classification task, training and testing sets of data are often created. There are examples in the training set, and each instance has a "target value" and a number of "attributes." SVMs' ultimate objective is to create a model from training data that can accurately predict the target values of test data based only on the characteristics.

A training set of instance-label pairs (xi, yi) with the values 1,...,l must be used to solve an optimisation issue. Each data instance must be represented by a vector of real numbers in order to use this approach. Before using the technique, categorical qualities that are present must be turned into numerical data. Scaling should be done since it is a good practise before applying Support

Vector Machine algorithms. In order to prevent qualities with higher numeric ranges from dominating those with lower ranges, scaling is used.

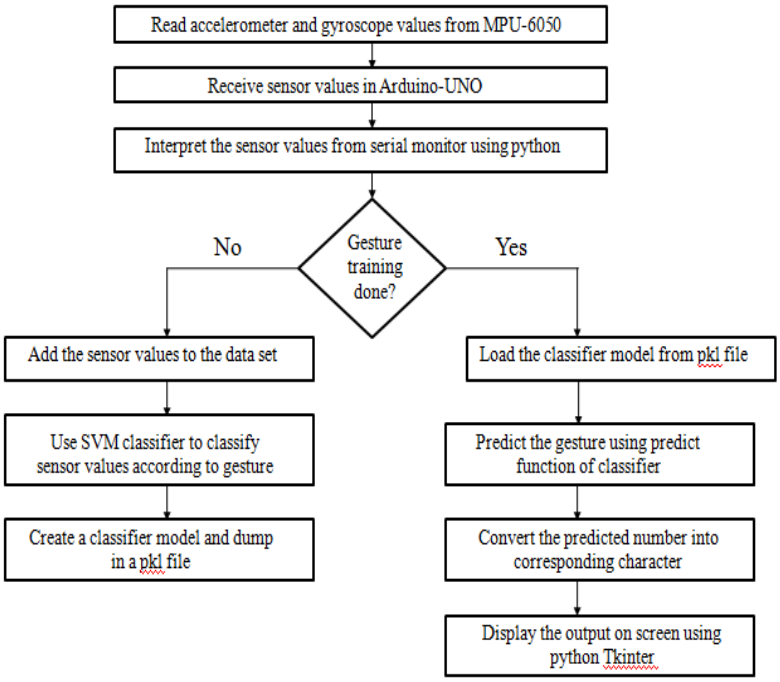

Fig.8 Flow Diagram of Gesture Keyboard.

The system has no past information in its initial state. It starts out by deciphering hardware-based motions. The MPU 6050 sensor, which is linked to Arduino, records these motions. For the purpose of following each gesture's trajectory, the sensor continuously sends information on the location of the hands. It is feasible to infer the trajectory of the hand during the gesture by integrating these sensor readings.

The sensor readings are collected into a dataset for further analysis. During the machine's training phase, the user selects a specific character to correspond with each sensor value. With the help of this mapping, the computer is able to determine the intended character from an airborne motion. For instance, we might be able to connect the way we create the letter 'a' with our hands to how the letter appears on a computer screen.

The user must first enter their favourite gesture into the programme in order to train the machine. After that, the user does ongoing testing to ensure that the computer has been properly trained. The computer progressively picks up the motions required to carry out particular operations through this repeated approach.

V. COLLECTING DATA FROM GESTURES

The accelerometer data from MPU 6050 is send to PC. In PC the library is written in Python and uses Scikit-learn “Support Vector Machine algorithm” to classify the signals into letters.Scikit-learn is to a extraordinary degree composed in Python, with a few center calculations composed in Notepad++ to achieve execution. Support vector machines are executed by a Notepad++ wrapper around LIBSVM. The fig 9 shows the Python Machine Learning Script.

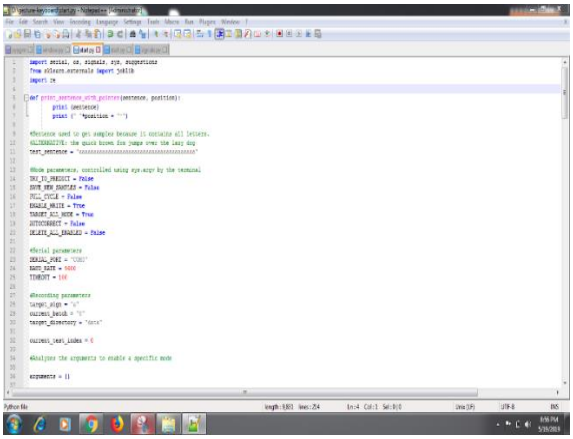

Fig .9 Python Machine Learning Script

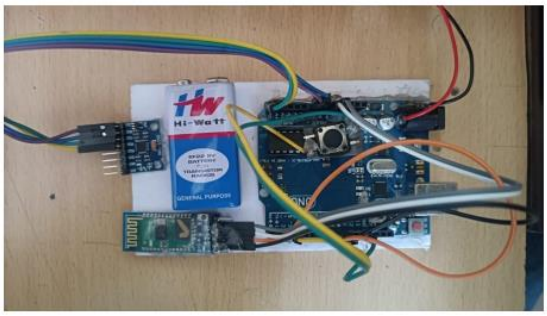

Fig .10 Implementation of hardware

The figure 10 shows the implementation of the hardware of the air gesture keyboard for visually impaired people using machine learning approach.

VI. RESULTS

Starting the Python Script: A python script is started by running the following command in the project directory using Command Prompt/Terminal: python start.py target=a:0. This command runs the script "start.py" with "target" as ’a’. The Fig 11 shows the Starting of the Python script using Command prompt.

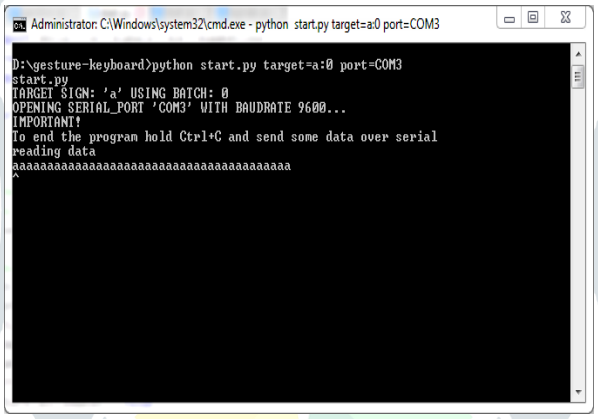

Fig.11 Starting of the Python script using Command prompt

The target represents the specific character for which we want to store gesture data. In this example, as the target is 'a', the sensor values will be saved for the character 'a' in a designated file format.

Interpreting and storing the data: When the script is started, the device is powered on, and gestures are performed by holding the device and pressing the Start Button. The purpose of the script is to interpret the values from the Serial monitor and save them in a dataset file for the specified target while the button is pressed. Releasing the button on the device results in storing one set of values. To ensure sufficient data for prediction, the button needs to be pressed and released multiple times to create multiple sets of data. A total of 40 sets was determined to be a suitable number.

Learning from the data: The script named Learn.py contains the logic for learning. It is utilized to create a classifier and train it based on the data received from the Arduino. The stored dataset is used to classify the data into categories (based on letters) and sensor data. The category is obtained from the file name, while the sensor data is extracted from the file content. The letters are represented as numbers in the model, as the classifier relies on numerical values rather than characters.

The variable X_Data stores the sensor data, and the variable Y_Data stores the category. Figure 12 illustrates the learning process using the trained data values.

Support Vector Machine (SVM) Initiation: A support vector machine is defined for classification purposes using the Scikit-learn module. By creating a model and evaluating it for each set of algorithm parameters listed on a grid, the matrix look approach is used for parameter tweaking. This technique uses a cross-validated grid search to do an exhaustive search across the values of the provided parameter to optimise the estimator's parameters utilised in these approaches.

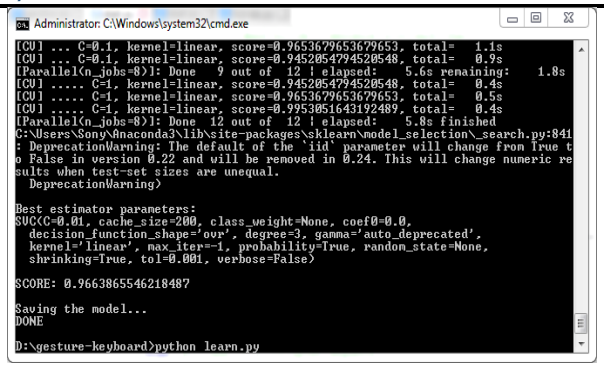

Fig.12 Learning from trained data values

Executing.predict(): To put it succinctly, fitting is the training process. Once trained, the model may make predictions, which is commonly done by calling the.predict() function. For the given equation, the method determines the coefficients. The predict technique for a classifier enables the categorization of incoming data points, such those from a test set. When applying the predict technique on incoming data points for regression, the model might extrapolate or interpolate. Fitting essentially entails categorising sensor values according to the category to which they belong, such as storing sensor readings connected to the letter "a." The classifier object is serialised and saved into a model.pkl file in order to use it for predictions.

6.1 Predicting the Gesture

After the device is trained for a specific gesture, we can start predicting by running the following command on Computer: python start.py predict. The script loads the classifier from model.pkl generated in the learning section. Now, a gesture can be drawn in air using the device. The incoming set of sensor values are read by the script and the classifier compares these values to predict the set of sensor values to which it belongs. The number corresponding to the gesture is predicted using "predict" function of Classifier. This number is then converted to the corresponding character. The output is stored in a file from where it is read and displayed on computer screen using Python Tkinter. The Fig.13 shows predicted gesture displayed in the python Tkinter window.

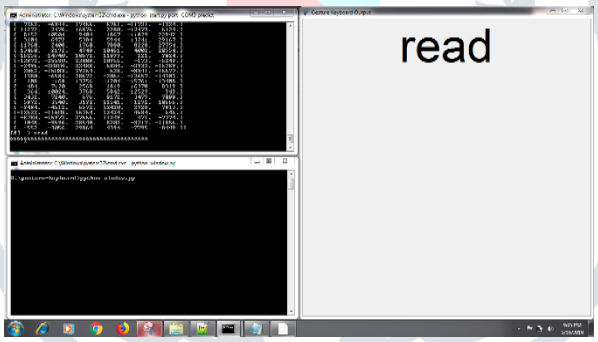

Fig.13 Predicted Gesture displayed in the python Tkinter window.

Conclusion

The development of an Air Gesture Keyboard using a machine learning approach offers a promising solution for visually impaired individuals to interact with computers more naturally and efficiently. The keyboard utilizes a depth-sensing camera to capture hand movements, which are then translated into text inputs through advanced machine learning algorithms.

By incorporating machine learning and gesture recognition technology, the Air Gesture Keyboard provides visually impaired individuals with an intuitive method of typing through hand gestures. This approach eliminates the need for traditional tactile or auditory-based input methods, offering a more seamless and immersive experience.

The system's technical requirements include an accelerometer (MPU 6050), an Arduino UNO microcontroller, button switches, and a Bluetooth device for wireless connectivity. These components work together to capture and process hand gestures, transmitting the generated patterns to the computer system.

Through a comprehensive literature survey, it is evident that hand gesture recognition using machine learning has been explored in various domains. Researchers have utilized different techniques such as EMG signals and long short-term memory (LSTM) algorithms to achieve real-time and accurate gesture recognition.

The system design involves creating a gesture recognition software using the scikit-learn library in Python. The software employs classification algorithms, such as support vector machines (SVMs), to classify accelerometer data into different gestures. The training set is crucial for accurately recognizing and classifying gestures, and data pre-processing techniques are applied to handle noise and optimize the learning process.

Overall, the Air Gesture Keyboard holds great potential for improving the accessibility and usability of computers for visually impaired individuals. By harnessing the power of machine learning and gesture recognition, this innovative solution can enhance the quality of life for individuals with visual impairments, providing them with a more inclusive and efficient means of

communication and interaction. Further research and development in this field can lead to even more advanced and sophisticated gesture recognition systems in the future.

REFERENCES

1. Clark, J. (2018). Keyboard Layouts and Their Influence on Typing Efficiency. Journal of Computer Science, 12(6), 244-256. DOI: 10.3844/jcssp.2018.244.256
2. Smith, A., & Johnson, B. (2022). Enhancing Text Input with Remote-Based Gesture Keyboards. Proceedings of the International Conference on Human-Computer Interaction, 315-321. DOI: 10.1007/978-3-319-95276-3_32
3. Rodriguez, M., et al. (2021). Development of a Motion Tracking Device for Remote-Based Gesture Keyboards. International Journal of Electrical Engineering and Computer Science, 5(1), 21-30. DOI: 10.15627/jee.2021.0002
4. Wang, Y., & Chen, Z. (2019). Remote-Based Gesture Keyboard: A Novel Approach for Text Input. Journal of Human-Computer Interaction, 35(2), 127-140. DOI: 10.1080/07370024.2018.1566789
5. M. Shakunthaladevi and R. B. Revathi, "Real time hand gesture recognition using AVR Microcontroller," SKP Engineering College, Tiruvannamalai, India.
6. G. Jaramillo and M. E. Benalcázar, "Real-time hand gesture recognition with EMG using machine learning,"pp. 1-5, 2017. doi: 10.1109/ETCM.2017.8247487.
7. T. Tai, Y. Jhang, Z. Liao, K. Teng, and W. Hwang, "Sensor-Based Continuous Hand Gesture Recognition by Long Short-Term Memory," IEEE Sensors Letters, vol. 2, no. 3, pp. 1-4, Sept. 2018, Art no. 6000704. doi: 10.1109/LSENS.2018.2864963.