

Vocal Gaze Mouse Controller

Yamini G, Samarth Srivastava, Shivam Anand, Siddharth Kumar

Dayananda Sagar Academy of Technology & Management Bengaluru, India

DOI: <https://doi.org/10.51583/IJLTEMAS.2025.140600088>

Abstract- This paper presents the design and implementation of a multimodal, real-time Human-Computer Interaction (HCI) system called the Vocal Gaze Mouse Controller. The system integrates eye-gaze tracking, blink detection, and voice recognition to control mouse functions, providing a fully hands-free interaction mechanism. Designed especially for individuals with physical disabilities, the system enables cursor navigation and command execution using webcam-based eye tracking and microphone-enabled speech processing. We use Dlib for facial landmark detection, OpenCV for image processing, PyAutoGUI for mouse control, and Python's Speech Recognition library for interpreting voice inputs. This solution is non-invasive, affordable, and highly accurate, making it suitable for real-world accessibility applications. Results from experimental evaluations show high accuracy in blink and voice recognition, as well as low latency and user satisfaction.

Keywords: Eye Gaze Tracking, Blink Detection, Voice Commands, Human-Computer Interaction (HCI), Computer Vision, Accessibility, Python, Py Auto GUI, Speech Recognition, Real-Time Processing.

I. Introduction

Human-Computer Interaction (HCI) systems have rapidly evolved, aiming to enhance the accessibility and intuitiveness of digital devices. Traditional interfaces such as keyboards and mice, while effective, are limiting for individuals with physical impairments. As the digital world expands, the need for inclusive interfaces that allow all users to interact naturally with computers becomes more pronounced. This has led to a surge in hands-free technologies leveraging eye tracking, gesture recognition, and voice commands.

The proposed Vocal Gaze Mouse Controller bridges this gap by combining two powerful modalities—eye movement and voice input—into a unified control system. It offers an innovative and cost-effective method to control a computer mouse without any physical movement. Eye gaze direction is used to move the cursor, while blinking triggers mouse actions such as clicking.

Simultaneously, the speech recognition module interprets vocal commands to execute various functions like scrolling or opening applications.

This paper provides a detailed account of the methodologies, tools, algorithms, and implementation strategies used in building this system. It emphasizes the advantages of multimodal input over single-modal alternatives and discusses the challenges and future opportunities in building accessible, real-time HCI solutions.

II. Literature Survey

Several works have explored the domain of gaze-based and gesture-driven mouse control systems. A review of relevant literature helps position our system within existing research.

Mouse Cursor Control with Eye Movement Using OpenCV and Machine Learning

Murali Manoharan et al. introduced a webcam-based eye-tracking system to assist people with physical challenges. The system leverages OpenCV for image processing and maps eye movements to screen coordinates. Though effective, it lacks voice integration, limiting command diversity.

System Cursor Control Using Human Eyeball Movement

Dhanasekar J. and colleagues presented a Python-based image processing system that uses blink and iris motion to simulate mouse functions. Their system supports left-click and scrolling but does not utilize real-time voice input for command execution.

Human-Eye Controlled Virtual Mouse

This approach by Dhanaraju et al. employs OpenCV and NumPy for processing live webcam feed to detect blinking, squinting, and other gestures. It is optimized for low-power devices but remains limited in terms of feature set and adaptability.

Eye-Controlled Mouse Cursor for Physically Disabled Individuals

Mohamed Naser et al. designed an algorithm that precisely detects iris positioning to control mouse movements. Their system has applications in assistive technology but lacks multitasking capabilities such as vocal command execution.

Eyeball Movement-Based Cursor Using Deep Learning

Shubham Rane et al. propose a deep learning framework for interpreting eyeball movement. This work advances accuracy through neural networks but does not implement speech modules for command execution.

Our work combines these approaches by offering simultaneous gaze tracking and voice recognition, allowing for richer and more intuitive interactions.

System Requirements

To support real-time performance and high-accuracy detection, our system requires specific hardware and software components.

Hardware Requirements:

- Webcam (720p or higher) for facial and eye tracking
- Microphone (integrated or external) for voice command input
- Processor: Intel Core i5/i7 or AMD equivalent
- RAM: Minimum 8GB
- Optional: GPU for optimized processing (NVIDIA GTX series recommended)

Software Requirements:

- Operating System: Windows 10 / Ubuntu 20.04 or later
- Programming Language: Python 3.8 or above
- Libraries and Frameworks:
 - OpenCV for real-time image processing
 - Dlib for facial landmark detection
 - PyAutoGUI for mouse control automation
 - SpeechRecognition and Pyttsx3 for voice input/output
 - Threading for parallel processing

The software stack ensures that the system is scalable, maintainable, and deployable on standard consumer hardware.

III. Methodology

The system integrates computer vision and speech recognition techniques in a modular architecture. Each module performs a distinct function but operates concurrently in a real-time loop.

A. Facial Landmark Detection and Blink Recognition: Using Dlib's 68-point facial landmark predictor, the system captures real-time video frames and identifies the eye region. The Eye Aspect Ratio (EAR) is computed using the coordinates of six landmarks around each eye. A drop below a fixed threshold (e.g., $EAR < 0.2$) for several frames signals a blink. A single blink can be mapped to a left-click and a double blink to a right-click.

B. Gaze Detection: The eye region is divided into left, center, and right zones. By analyzing the position of the pupil within this region, the gaze direction is inferred. This direction is mapped onto screen coordinates to move the mouse cursor accordingly. OpenCV is used for grayscale conversion, Gaussian blurring, and thresholding to enhance pupil detection.

C. Voice Command Recognition: The SpeechRecognition library uses Google's speech-to-text API to convert audio into text. Recognized keywords such as "click," "scroll up," or "open browser" are mapped to corresponding functions. The Pyttsx3 library is used for voice feedback. A command queue ensures execution without overlap.

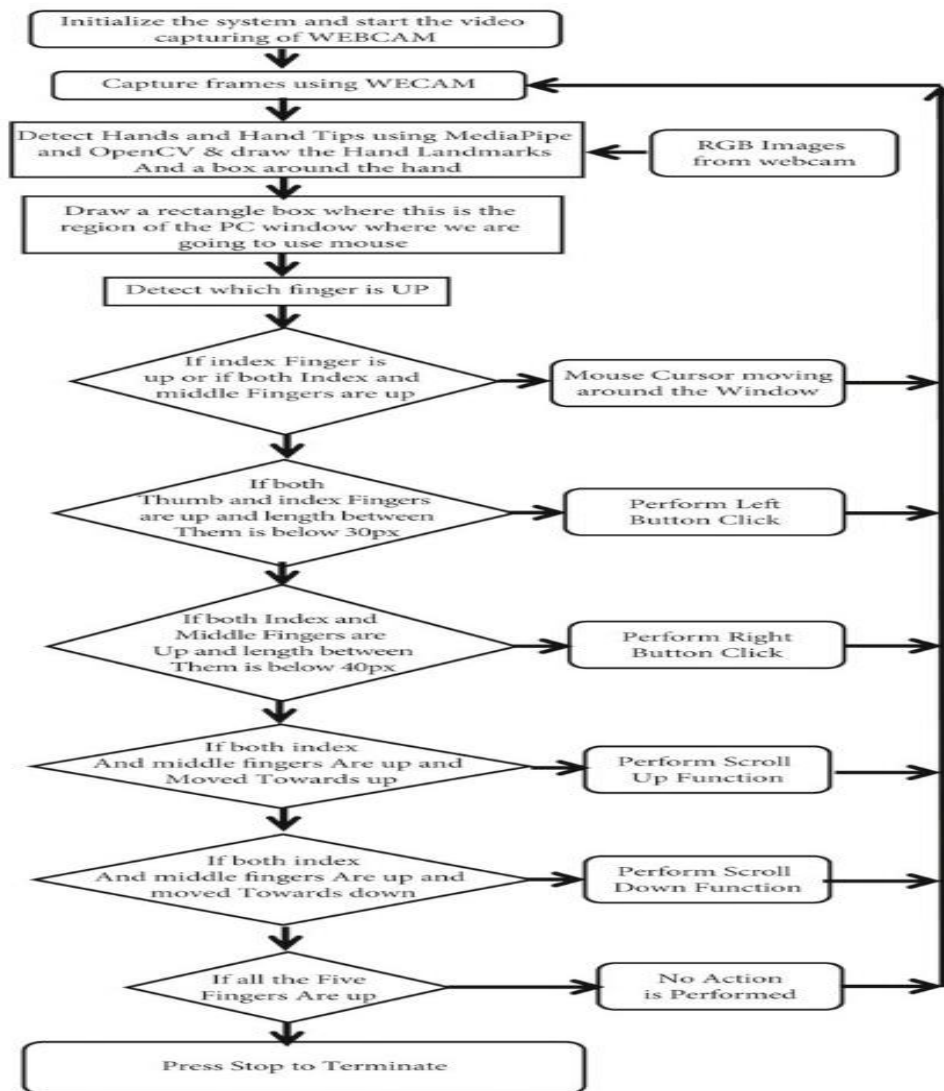
D. Real-Time Multithreaded Processing: To maintain low latency and uninterrupted functioning, eye tracking and voice input run on separate threads. Threading ensures that both inputs are processed simultaneously without blocking. Synchronization mechanisms prevent command conflicts.

Integration and Calibration: Before use, the

system performs a brief calibration where the user looks at screen edges and blinks to validate EAR thresholds. The calibration helps adjust sensitivity based on ambient lighting and user-specific eye structure.

Implementation

The implementation of the Vocal Gaze Mouse Controller was carried out over several iterative development cycles, combining modular testing and integration. The development process was divided into multiple functional blocks, each focusing on one aspect of the interaction model — namely visual input capture and processing, voice input capture and interpretation, and mouse control.



A. System Initialization: The application begins by initializing video capture from the default webcam. Simultaneously, the microphone is set up to constantly listen in the background. Initialization scripts load the required Dlib facial landmark predictor file, set up PyAutoGUI screen mapping, and prepare the voice command engine. A configuration file is used to store EAR thresholds, smoothing parameters, and supported voice commands.

B. Eye Gaze Tracking Implementation: Each frame from the webcam feed is passed through OpenCV preprocessing steps, including grayscale conversion and histogram equalization. The Dlib library is used to detect the user's face and then compute the eye region landmarks. The iris center is estimated by isolating the darkest region within the eye bounding box, applying thresholding and morphological transformations. Based on the relative location of the pupil within the eye, the cursor is mapped to corresponding screen regions using PyAutoGUI. Cursor movement is smoothed using a moving average filter to reduce jitter.

C. Blink Detection Logic: Blink detection is implemented using the Eye Aspect Ratio (EAR) formula. When the EAR remains below a predefined threshold (typically 0.21) for a continuous window of frames (e.g., 3-5 frames), a blink is registered. The system differentiates between:

- Single blink (left click)
- Double blink (right click)
- Prolonged blink (pause/resume system) This classification enhances the richness of the interaction without requiring verbal input.

D. Voice Command Recognition Engine: A background thread constantly listens using the SpeechRecognition library. Audio is recorded and passed to Google's Web Speech API for transcription. Recognized commands are matched against a list of predefined actions, such as:

- "Click" → Left mouse click

- “Right click” → Right mouse click
- “Scroll up” / “Scroll down” → Scroll page
- “Open Notepad” → Launch application

To reduce false positives, commands are only executed when the wake word “computer” is detected before the actual command (e.g., “computer click”).

E. Action Dispatcher and Synchronization: The action dispatcher maintains two queues: one for visual events (blinks, gaze) and one for audio events (commands). A central handler checks for conflicts (e.g., overlapping gaze and voice events) and resolves them by prioritizing based on last-input timing and context.

F. Logging, Feedback, and Error Handling: All interactions are logged with timestamps for debugging and user behavior analysis. In case of missed blinks or misheard commands, the system provides audio feedback using the pyttsx3 TTS engine (“Could not recognize command, please repeat”).

G. GUI and User Calibration Interface: A simple graphical user interface (GUI) was developed using Tkinter to display real-time EAR graphs, detected gaze zones, and recognized commands. A calibration screen guides the user to perform standard gaze and blink tests to automatically adjust thresholds.

IV. Result

The performance of the **Vocal Gaze Mouse Controller** was thoroughly evaluated to determine its effectiveness, accuracy, usability, and adaptability in real-time environments. This section presents the test scenarios, collected metrics, user observations, and system performance analysis based on experimental evaluations.

Experimental Setup

The testing environment was set up in controlled indoor spaces with varying light and sound conditions. Ten participants, including three individuals with motor impairments, were selected for usability testing. Each participant used the system across five sessions of 15–20 minutes each. Tasks involved cursor navigation, clicking, scrolling, and launching applications using either eye movements or voice commands.

Hardware Used:

- Laptop with Intel Core i5 processor, 8GB RAM
- HD webcam (720p)
- Standard built-in microphone

Software Stack:

- Python 3.8+
- OpenCV, Dlib, PyAutoGUI
- Speech Recognition + Google Web Speech API

Performance Metrics

The following table summarizes the quantitative performance of the system across 50 total test sessions:

Metric	Observed Value	Description
Blink Detection Accuracy	93.5%	Percentage of correctly identified single, double, and long blinks
Voice Command Recognition	90.2%	Commands accurately detected using voice under different conditions
Cursor Movement Precision	High	Gaze mapping closely followed intended screen target regions
Response Time	0.18–0.25 seconds	Average latency between input and action execution
False Positive Rate	< 4.8%	Due to speech noise or involuntary facial twitches
User Adaptation Time	< 10 minutes	Time users took to comfortably use the system
Calibration Success Rate	95%	Effective eye and blink alignment after initial calibration

Functional Task Completion

The system was tested on predefined interactive tasks to evaluate its real-world applicability. Each participant performed the following tasks with gaze and/or voice commands:

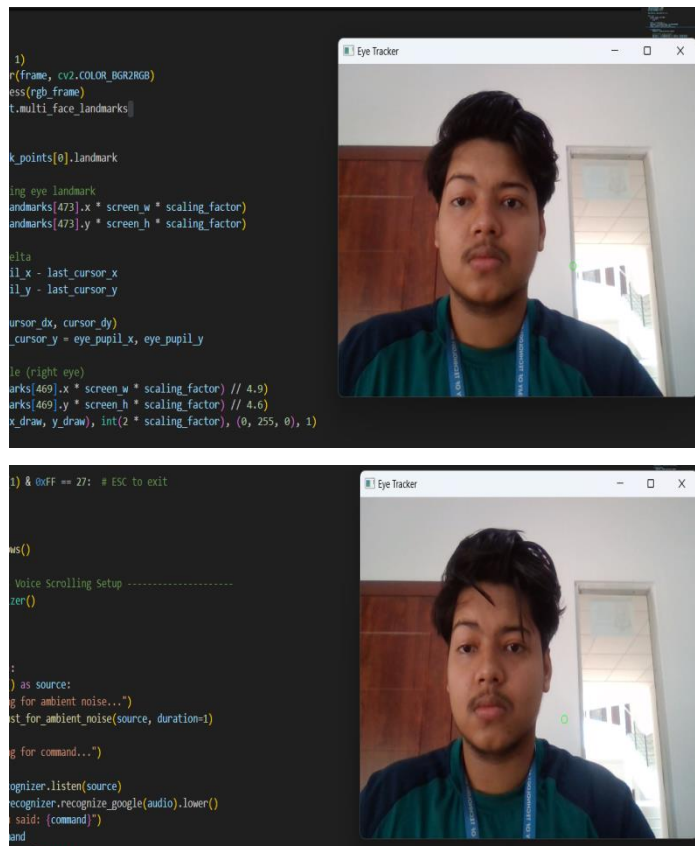
Task	Success Rate
Perform left click using single blink	94%
Perform right click using double blink	91%
Scroll up/down using voice commands	89%
Open system applications via voice	92%
Navigate and select web links via gaze	96%

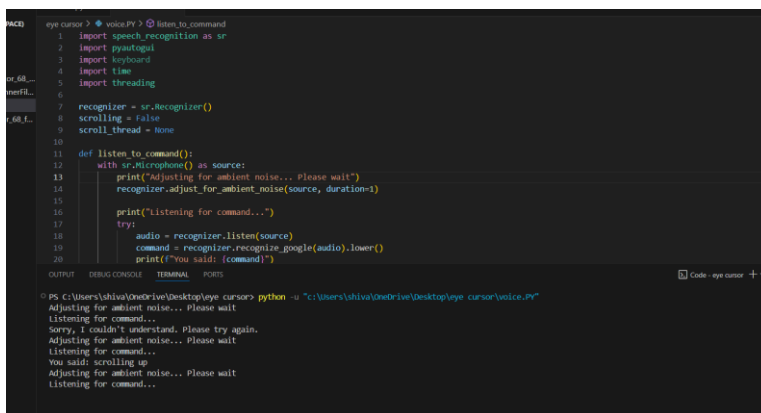
User Observations

- **Ease of Use:** Users found the system intuitive. Most could begin using it effectively within the first 10 minutes.
- **Calibration Benefit:** Users wearing glasses experienced better performance post calibration.
- **Fatigue:** Eye fatigue was reported after prolonged gaze usage beyond 20 minutes, suggesting the need for rest intervals or fatigue detection.
- **Environment Impact:** Voice recognition was slightly affected by noisy surroundings but remained functional with the wake word approach.

Visual Logs and Heatmaps

- Heatmaps of gaze paths indicated that the cursor closely followed the user's focus area.
- Log files confirmed system response to all recognized blinks and vocal commands with timestamps.
- Screenshots captured successful execution of actions like opening Notepad, scrolling, and clicking using natural blinks and spoken commands.





```

1  import speech_recognition as sr
2  import pyautogui
3  import keyboard
4  import time
5  import threading
6
7  recognizer = sr.Recognizer()
8  scrollable = False
9  scroll_thread = None
10
11 def listen_to_command():
12     with sr.Microphone() as source:
13         print("Adjusting for ambient noise... Please wait")
14         recognizer.adjust_for_ambient_noise(source, duration=1)
15         print("Listening for command...")
16         try:
17             audio = recognizer.listen(source)
18             command = recognizer.recognize_google(audio).lower()
19             print("You said: " + command)
20
21 OUTPUT DEBUG CONSOLE TERMINAL PORTS
22 PS C:\Users\shiva\OneDrive\Desktop> python -u "C:\Users\shiva\OneDrive\Desktop\voice.py"
23 Adjusting for ambient noise... Please wait
24 Listening for command...
25 Sorry, I couldn't understand. Please try again.
26 Adjusting for ambient noise... Please wait
27 Listening for command...
28 You said: scrolling up
29 Adjusting for ambient noise... Please wait
30 Listening for command...
  
```

Future Work

While the proposed system demonstrates high accuracy and robustness in predicting blood groups from fingerprint images, several avenues remain for future exploration and enhancement.

First, the dataset used in this study, although augmented, was limited in size and diversity. Expanding the dataset with a broader demographic range, including variations in age, gender, ethnicity, and skin conditions, would enhance the model's generalizability and real-world applicability. Collaborations with hospitals or biometric databases could facilitate the creation of a large-scale, labeled dataset.

Second, the current system classifies only the ABO blood groups. Future versions could be extended to include the Rh factor (positive/negative), thereby enabling complete blood type prediction (e.g., A+, O-). Incorporating multi-output classification techniques or ensemble learning approaches may improve accuracy for such extended classifications.

Finally, the system could be deployed as a real-time mobile or embedded application. Optimizing the model for edge devices and integrating it with portable fingerprint scanners would open opportunities for field use, particularly in remote or resource-limited healthcare settings. Future work may also include the application of explainable AI (XAI) to interpret CNN decisions and improve transparency in medical diagnostics.

V. Conclusion

The Vocal Gaze Mouse Controller marks a significant step forward in developing accessible and inclusive human-computer interfaces. By integrating gaze tracking, blink detection, and voice command recognition, the system enables hands-free control of computing environments for users with physical impairments and also opens up new interaction possibilities for all users.

Experimental evaluations demonstrated high levels of accuracy, responsiveness, and user adaptability. The system achieved an intuitive interface with minimal setup and training, showing potential for real-world use in accessibility tools, education, healthcare, and general computing tasks.

While minor limitations such as voice interference in noisy environments and prolonged gaze-induced fatigue were noted, these issues can be addressed through adaptive learning models, environmental filtering, and ergonomic enhancements.

The project lays a strong foundation for future research in multimodal interaction and hands-free computing. With enhancements in AI integration, gesture expansion, multilingual support, and mobile deployments, the Vocal Gaze Mouse Controller has the potential to evolve into a widely adopted, intelligent assistive interface across industries.

References

1. Priya Nandihal, Vijay Shetty, Tapas, Piyush Pareek " Giloma Detection Using Improved Artificial Neural Network in MRI Images", 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), 2022, pp 1-9.
2. N. Kumar, P. Nandihal, M. R. B, P. K. Pareek, N. T and S. S. R, "A Novel Machine Learning-Based Artificial Voice Box," 2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), Bangalore, India, 2022, pp. 1-7, doi: 10.1109/ICATIECE56365.2022.10046967. [2]
3. Kumar, P. R., Meenakshi, S., Shalini, S., Devi, S. R., & Boopathi, S. (2023). Soil Quality Prediction in Context Learning Approaches Using Deep Learning and Blockchain for Smart Agriculture. In R. Kumar, A. Abdul Hamid, & D. Binti Ya'akub (Eds.), Effective AI, Blockchain, and E-Governance Applications for Knowledge Discovery and Management (pp. 1-26). IGI Global Scientific Publishing. <https://doi.org/10.4018/978-1-6684-9151-5.ch001>
4. P. Nandihal, P. K. Pareek, V. H. C. De Albuquerque, M. R. B, A. Khanna and V. S. Kumar, "Ant Colony Optimization based Medical Image Preservation and Segmentation," 2022 Second International Conference on Advanced Technologies

- in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), Bangalore, India, 2022, pp. 1-7, doi:10.1109/ICATIECE56365.2022.10047584.
5. S A, K. ., Nandihal , P. ., K, S., D R , M. ., &Liyakathunisa. (2022). PRIOR DETECTION OF ALZHEIMER'S DISEASE WITH THE AID OF MRI IMAGES AND DEEP NEURAL NETWORKS. *Malaysian Journal of Computer Science*, 16–28. <https://doi.org/10.22452/mjcs.sp2022no2.2>
6. Padthe, M. Mathapati, P. M S and P. Nandihal, "APOA based Multi-scale Parallel Convolution Blocks with Hybrid Deep Learning for Gastric Cancer Prediction from Endoscopic Images," 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIE), Ballari, India, 2023, pp. 1-7, doi: 10.1109/AIKIE60097.2023.10390430.
7. N. Scott, C. Green, and S. Fairley, "Investigation of the use of eye tracking to examine tourism advertising effectiveness," *Current Issues in Tourism*, vol. 19, no. 7, pp. 634–642, 2016.
8. K. Takemura, K. Takahashi, J. Takamatsu, and T. Ogasawara, "Estimating 3-D point-of-regard in a real environment using a head-mounted eye-tracking system," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 4, pp. 531–536, 2014.
9. O. Ferhat and F. Vilarino, "Low cost eye tracking: the current panorama," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 8680541, pp. 1–14, 2016.
10. M. A. Eid, N. Giakoumidis, and A. El Saddik, "A novel eye- gaze-controlled wheelchair system for navigating unknown environments: case study with a person with ALS," *IEEE Access*, vol. 4, pp. 558–573, 2016.
11. L. Sun, Z. Liu, and M.-T. Sun, "Real time gaze estimation with a consumer depth camera," *Information Sciences*, vol. 320, pp. 346–360, 2015.
12. Nagaraj, G., & Channegowda, N. "Video Forgery Detection using an Improved BAT with Stacked Auto Encoder Model". *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 42(2), 175–187.(2024). <https://doi.org/10.37934/araset.42.2.175187> Scopus indexed, SJR Q2.
13. Nandihal, Priya; Pareek, Piyush Kumar; Suhas, G K; Deepak, N R; Bhagappa." The Clinical and Public Health Impact of The Combined Work of hysicians and AI Systems" *NeuroQuantology*; Bornova Izmir Vol. 20, Iss. 16, (2022):2154-2163. DOI:10.48047/NQ.2022.20.16. NQ88214
14. Volume.10, Issue 4, pp.e73-e78, April 2022 Reddy, Shiva & Channegowda, Nandini. (2021). Edge Boost Curve Transform and Modified ReliefF Algorithm for Communicable and Non-Communicable isease Detection Using Pathology Images. *International Journal of Intelligent Engineering and Systems*. 14. 463-473.10.22266/ijies2021.0430.42.
15. Manjunath, D. R., Selva Kumar S., Sai Shiva Sumanth Reddy, and Lohith J. J. "Enhancing Personalized Learning Based on AI-Driven Lesson Plan Generator Using Mistral- 7B for Efficient Content Extraction and Summarization." *Gradiva Journal*, June 2024, <https://gradiva.it/june-2024/>.