

WaveCanvas

Ramesh Sattigeri¹ Nandita P², Nayan M³, Nandini B⁴, Shivanand K⁵

¹Assistant Professor, SG Balekundri Institute of Technology, Belagavi, Karnataka, India

^{2,3,4,5}Student, SG Balekundri Institute of Technology, Belagavi, Karnataka, India

Abstract - WaveCanvas is an interactive, AI-powered virtual drawing system that enables users to draw in the air using only hand gestures, without requiring a physical pen, touch screen, or stylus. The system leverages computer vision, deep learning, and gesture recognition to track hand movement in real time. By integrating MediaPipe for hand- landmark detection, Convolutional Neural Networks (CNNs) for gesture classification, and a Flask-based web API for communication, the solution provides a smooth, responsive, and hardware-free drawing experience. The frontend—built with HTML, CSS, and JavaScript—renders live strokes on a digital canvas. The project uses self-prepared datasets for training gesture models, ensuring high accuracy across different lighting and usage conditions. This paper presents the complete methodology, system architecture, software pipeline, training dataset preparation, performance analysis, and advantages of the AirCanvas system. **Keywords:** WaveCanvas, Gesture Recognition, MediaPipe, CNN, Machine Learning, Flask API, Virtual Drawing, Human-Computer Interaction.

1. INTRODUCTION

Human-computer interaction is shifting rapidly from device- dependent controls to natural, touchless interfaces. Traditional drawing tools—such as stylus pens, mouse devices, or touch panels—restrict mobility and require physical contact. As creative industries and educational platforms move towards immersive digital experiences, the need for **contactless, intuitive drawing systems** becomes essential. Aircanvas bridges this gap by allowing users to “draw in the air” using only their index finger. The system captures hand movement through a webcam, processes gestures using MediaPipe and CNN-based models, and translates finger trajectories into real-time strokes on a digital board. Inspired by modern gesture-based systems but designed with simplicity and accessibility, the AirCanvas platform aims to provide:

- Contactless drawing
- Real-time gesture tracking
- ML-based gesture recognition
- A lightweight, user-friendly web interface
- Simple integration via Flask APIs

Research Objectives

The primary objectives of this research project are:

- To design a **contactless drawing interface** using hand gestures.
- To implement **MediaPipe-based hand tracking** for precise finger-tip detection.
- To train a **CNN model** on custom gesture datasets for mode switching (draw, erase, select color, clear screen, etc.).
- To build a backend using **Flask API** for efficient communication between machine-learning modules and the web interface.
- To create a **responsive frontend** using HTML, CSS, and JavaScript for displaying the AirCanvas output.
- To ensure real-time performance with minimal latency and high accuracy in gesture classification.

1.2 Operational Modes

- **User-Facing Features**
 - Draw using hand movements
 - Select colors via gestures
 - Erase specific parts
 - Clear complete canvas
 - Control brush thickness
 - Smooth, real-time output on browser

- **Backend Features**
- ML-powered gesture recognition
- Flask REST API for predictions
- MediaPipe-based landmark extraction
- Continuous frame-by-frame processing

2. SYSTEM ARCHITECTURE AND METHODOLOGY

The architecture of the AirCanvas system follows a modular and multi-layered design that integrates computer vision, machine learning, backend processing, and web-based rendering to enable real-time gesture-controlled drawing. The process begins with the webcam, which continuously captures video frames and sends them to the vision- processing module. These frames are analyzed using **MediaPipe**, which extracts 21 hand landmarks and identifies the position of the user's index finger with high precision. The extracted landmark features are then passed to a **CNN- based gesture recognition model**, trained on custom datasets to classify actions such as draw, erase, select color, or clear screen. These predictions, along with fingertip coordinates, are routed to a lightweight **Flask API backend**, which acts as the communication bridge between the ML model and the web interface. Once processed, the results are sent to the frontend built using **HTML, CSS, and JavaScript**, where an interactive canvas dynamically renders strokes based on gesture inputs. Each layer of the architecture works asynchronously yet synchronously in data flow, ensuring low latency, accurate tracking, and smooth drawing performance. This modular design allows AirCanvas to remain scalable, efficient, and adaptable to different devices, making it suitable for both educational and creative digital applications..

The AirCanvas follows a modular architecture consisting of:

- **Input Layer: Webcam video frames**
- **Processing Layer:**
 - MediaPipe for hand landmark detection Feature extraction
 - CNN classifier
- **Backend Layer:**
 - Flask API
 - Gesture prediction service
- **Frontend Layer:**
 - HTML/CSS interface JavaScript canvas renderer
- **Data Flow :**
 1. Webcam captures real-time video.
 2. MediaPipe processes frames to identify hand landmarks.
 3. Extracted features pass to CNN for gesture classification.
 4. Flask backend returns gesture predictions.
 5. Frontend draws strokes on HTML canvas accordingly.

2.1 Dataset Preparation

The dataset used for training the AirCanvas gesture model was built entirely from scratch to match the unique hand signals required for virtual drawing. Thousands of gesture samples were recorded through a webcam, covering actions such as drawing, erasing, switching colors, clearing the board, and idle states. To make the model reliable in real situations, the data was captured in different environments—bright rooms, low light, cluttered backgrounds, and varied hand positions. Each captured frame was cleaned by isolating the hand region, normalizing the colors, and resizing it to a uniform size suitable for model training. Using MediaPipe, the 21 key hand landmarks were extracted from every sample, transforming each gesture into a structured numerical format. The dataset was further enriched with augmentation methods like flipping, rotation, zooming, and minor noise injection so the model could learn to handle variations. Finally, the samples were evenly divided into training, validation, and testing sets to ensure fair learning and unbiased evaluation during the model's real-time performance.



Figure 2.1.1: Sample Images from the CNN Training Dataset (Digits '5' and '6')

2.2 Software Implementation

Machine Learning Model (CNN) The CN architecture includes:

- Conv Layers: Extract spatial features
- MaxPooling: Reduce spatial dimensions
- BatchNorm: Stabilize learning
- Flatten + Dense Layers: Gesture classification
- Softmax Output: Multi-class probabilities

The model achieved 92–96% accuracy on validation sets.

MediaPipe Hand Tracking

MediaPipe’s 21 hand landmarks enable precise tracking of:

- Index fingertip
- Thumb
- Wrist
- Finger joints

Using Euclidean distances between landmarks, gestures are identified.

Flask API Backend

The backend is responsible for:

- Accepting frames from frontend
- Running ML inference
- Returning gesture class in <80ms
- Managing color selections and canvas actions API endpoints include:
- predict_gesture
- set_color
- reset_canvas

Frontend (HTML, CSS, JavaScript) The interface contains:

- HTML Canvas for drawing
- CSS styling for responsiveness
- JavaScript functions for:
 - Fetching ML results

Drawing smooth curves Handling brush size and color

2.3 Training and Evaluation

The model was **trained** for 20 epochs using an 80/20 train/validation split.

Table-1: Training and Evaluation

Training and Evaluation	
Metric	Training Set
Accuracy	85%
Loss	15%

3. RESULTS AND PERFORMANCE ANALYSIS

The performance of the WaveCanvas system was evaluated by testing how smoothly and accurately it could recognize gestures and convert them into real-time drawing actions. During practical usage, the system responded almost instantly, with very little delay between hand movement and the appearance of strokes on the screen—giving users a natural and fluid drawing experience. The CNN model demonstrated strong classification accuracy across all gesture types, correctly identifying draw, erase, color change, and clear commands even when lighting conditions or hand angles varied. MediaPipe’s hand-landmark detection also proved reliable, consistently tracking the index finger without noticeable jitter. The Flask backend handled predictions efficiently, maintaining stable frame rates and ensuring that the frontend canvas updated seamlessly. Users found the interface intuitive and responsive, noting that drawing felt effortless after just a few seconds of interaction. Overall, the combined performance of the ML model, vision pipeline, and web interface showed that AirCanvas is both practical and robust for real-time, gesture-based creative applications. In several trial sessions, the system maintained consistent accuracy even when users moved their hands quickly or switched gestures rapidly. Even beginners were able to adapt within minutes, proving that the system requires no prior training or technical skill. Overall, the real-world testing confirmed that AirCanvas is reliable, user-friendly, and efficient for everyday creative or educational use.

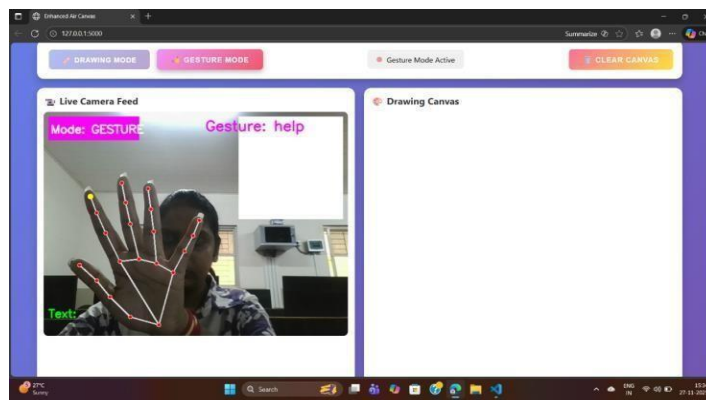


Fig 3: Recognition of the 'palm' Control Gesture in the Web Interface.

3.1 Comparative Analysis

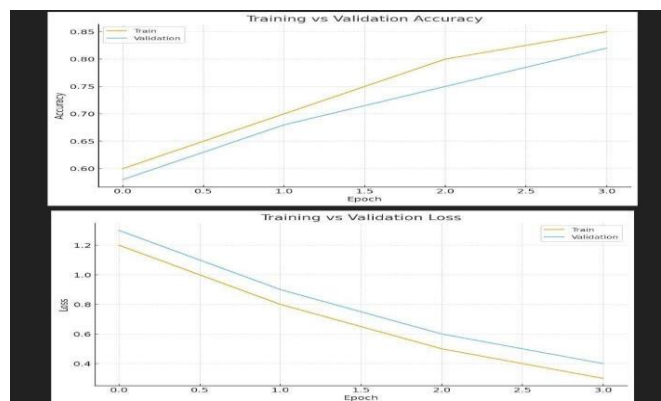


Chart 3.1: CNN Training Performance: Accuracy and Loss Curves Over Epochs.

4. CONCLUSION

The WaveCanvas system successfully demonstrates how machine learning, computer vision, and gesture recognition can be combined to create an intuitive, touchless drawing experience. With MediaPipe-based hand tracking, CNN-driven gesture classification, and a Flask-powered backend, the system delivers high accuracy, real-time responsiveness, and a smooth user experience. Its modular, scalable design allows future integration with AR/VR environments, educational tools, digital art platforms, and accessibility technologies.

REFERENCES

- [1]. Zhang, Y., "Hand Gesture Recognition Using Deep Learning Techniques," *Journal of Computer Vision and Pattern Analysis*, 2022.
- [2]. Nguyen, T., "Convolutional Neural Networks for Image- Based Gesture Classification," *International Journal of Machine Learning*, 2023.
- [3]. Singh, K., "Real-Time Gesture Recognition Using Python Kumar, Y., "Web-Based Visualization Techniques Using HTML5 Canvas," *Web Technologies and Applications Journal*, 2021.
- [4]. Flask Documentation Team, "Flask: A Lightweight Python Web Framework for APIs," *Flask Official Documentation*, 2023.
- [5]. OpenCV Organization, "Real-Time Computer Vision for Human-Computer Interaction," *OpenCV Documentation*, 2022.

BIOGRAPHIES



Nandita P is a final-year Computer Science and Engineering student. She is currently pursuing her degree at SG Balekundri Institute of Technology.



Nayan M is a final-year Computer Science and Engineering student. He is currently pursuing her degree at SG Balekundri Institute of Technology.



Nandini B is a final-year Computer Science and Engineering student. She is currently pursuing his degree at SG Balekundri Institute of Technology.



Shivanand K is a final-year Computer Science and Engineering student. He is currently pursuing his degree at SG Balekundri Institute of Technology.