Aim

To implement a real-time hand pose detection application using TensorFlow.js and React.

Objective

Develop a React application that utilizes TensorFlow.js to detect hand poses through webcam input.

Visualize hand landmarks and finger joints on the canvas overlaying the webcam feed.

Summary

This project involves creating a React application that integrates TensorFlow.js to perform real-time hand pose detection using a webcam. The application captures video input from the webcam, processes the frames to detect hand landmarks, and renders these landmarks on a canvas. The handpose model from TensorFlow.js is used for detecting hand landmarks, and custom drawing utilities are employed to visualize the detected hand poses with distinct colors and sizes.

Tools and Libraries Used

React: For building the user interface.

TensorFlow.js: For running the hand pose detection model.

@tensorflow-models/handpose: Pre-trained model for hand pose detection.

Webcam: For capturing video input.

Canvas API: For drawing hand landmarks on the canvas.

Procedure

Initialize React App: Set up a new React application using npm init react-app hand-pose.

Import Dependencies: In app.js, import necessary libraries including TensorFlow.js and the handpose model.

Set Up Webcam and Canvas: Use the react-webcam library to capture video and a HTML canvas to draw landmarks.

Load Handpose Model: Asynchronously load the handpose model from TensorFlow.js.

Detect Hands: Implement a detect function that uses the model to estimate hand poses from video frames.

Draw Landmarks: Utilize the Canvas API to draw landmarks and finger joints on the canvas, with predefined styles for visualization.

Highlights

Real-Time Detection: The application detects and updates hand poses in real-time with a frame interval of 100ms.

Custom Visualization: Implements a custom drawing function (drawHand) to visualize hand landmarks with different colors and sizes, enhancing the user experience.

Webcam Integration: Seamlessly integrates webcam input with TensorFlow.js for live hand pose detection.

Conclusion

The real-time hand pose detection application effectively demonstrates the capabilities of TensorFlow.js in a React environment. By combining webcam input with hand pose detection and custom visualization, the project provides a practical example of how machine learning models can be integrated into web applications to create interactive and visually informative tools.