IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 12, April 2025
Impact Factor: 7.67

Real-Time Face Detection App Using TensorFlow.js and React.js

Shivam Gandhi, Sanyam Grover, Prof. Renu Narwal
Dronacharya College of Engineering, Gurugram, Haryana, India

Abstract: *Real-time face detection has gained immense relevance in areas such as surveillance, authentication, and user experience personalization. This research presents a web-based face detection application built using TensorFlow.js and React.js. Leveraging deep learning models such as BlazeFace and MediaPipe, the system detects faces directly through a browser without backend dependencies. The architecture supports real-time video processing, cross-device compatibility, and responsiveness, making it ideal for scalable, lightweight deployment. The study includes implementation methodology, system workflow, model training references, performance evaluation, challenges, and future prospects.*

Keywords: face detection

I. INTRODUCTION

Face detection is a subset of object detection that identifies and localizes human faces within digital images or video frames. With the rise of web technologies and privacy concerns, there is a need for real-time, browser- based face detection systems that do not rely on server-side computation. TensorFlow.js provides deep learning inference capabilities in the browser, while React.js offers an efficient UI for rendering and control. This project demonstrates how these technologies can be integrated to build a fast, efficient, and privacy-
friendly face detection system.

A. Problem Statement

Traditional face detection systems rely on server-side computation, which may introduce latency, scalability limitations, and privacy risks. Furthermore, such systems require strong internet connectivity and are limited in
responsiveness and user control.

B. Objective

To develop a browser-based, real-time face detection app that uses TensorFlow.js models and React.js frontend to ensure fast, accurate face detection without compromising user data privacy.

II. METHODOLOGY

A. Technology Stack

Frontend: React.js - Model Inference: TensorFlow.js -
Face Detection Models:
BlazeFace
MediaPipe FaceMesh

B. Detection Pipeline

- Video feed captured from webcam using navigator.media Devices.getUserMedia().
- Video frames passed to the TensorFlow.js face detection model.
- Bounding boxes drawn on detected face regions.
- Detection data updated in real time using React state management

DOI: 10.48175/IJARSCT-25905

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 12, April 2025
Impact Factor: 7.67

Model Selection BlazeFace was selected for its lightweight architecture optimized for real-time inference in browsers, achieving 30+ FPS on most devices.

III. PROJECT WORKFLOW

User Interface:

- Users are prompted for camera access.
- Live video feed rendered with face bounding boxes.

Face Detection:

- TensorFlow.js model loaded asynchronously.
- Continuous detection using animation frames.

Feedback System (optional future integration):

- Display detection confidence score.
- Allow snapshot and save with bounding box metadata.

Responsiveness: - Mobile and desktop compatibility via responsive UI in React.

IV. TRAINING AND EVALUATION

The system uses pre-trained BlazeFace model from TensorFlow.js model zoo. Evaluation was conducted by testing the model on varying lighting, distances, and facial orientations.

Metrics Used:

- Frame Rate (FPS)
- Detection Accuracy
- Confidence Score Thresholding

Performance: - Average FPS: ~30 on mid-range devices - Detection Accuracy: ~95% in controlled environments

V. FUTURE DIRECTIONS

- Emotion Detection: Integrate emotion classifiers using face landmarks.
- Face Recognition: Match detected faces against a known dataset.
- Offline PWA Mode: Enable Progressive Web App for offline usage.
- Security: Add encryption for camera feed if stored or transmitted.
- Accessibility: Voice-based alerts for users with visual impairments

VI. EXISTING CHALLENGES

Limited GPU access in browsers may slow performance on low-end devices. - False positives in complex backgrounds. - Privacy concerns even with local detection-requires proper handling of permissions. - Lighting variation affects detection accuracy. - Cross-browser compatibility may cause inconsistencies.

VII. SYSTEM ARCHITECTURE

The application architecture follows a client-side model, eliminating the need for server-side computations. It consists of three primary layers: 1. **UI Layer**: Built using React.js, responsible for rendering the webcam interface, detection overlays, and user controls. 2. **Inference Layer**: TensorFlow.js runs pre-trained models like BlazeFace in the browser to process video frames. 3. **State Management**: React hooks and context manage real-time detection results, user permissions, and performance states.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 12, April 2025
Impact Factor: 7.67

VIII. PSEUDOCODE FOR DETECTION MODULE

Initialize webcam
Load BlazeFace model using TensorFlow.js While (webcam is active):
Capture frame from video
Run inference using model.estimateFaces() For each detected face:
Draw bounding box on canvas
Display detection confidence

IX. USE CASES

1. **Online Proctoring**: Detect and track faces during online exams to monitor candidate presence.
2. **Visitor Logging**: Small-scale surveillance systems for real-time face logging at entry points.
3. **Emotion Analysis**: Future integration with facial expression recognition can enable sentiment-based applications.
4. **Accessibility**: Apps for visually impaired users to get alerts when someone is in front of them.

X. SNAPSHOT/RESULTS

Below are examples of system snapshots during live face detection.

XI. CONCLUSION

The research showcases the feasibility of building a lightweight, real-time face detection system using web technologies. The integration of TensorFlow.js and React.js allows fast detection without server-side support, promoting privacy, scalability, and accessibility. This solution is especially promising for educational, authentication, and monitoring applications in low-resource environments.

ACKNOWLEDGMENT

The author would like to thank Renu Narwal for their invaluable guidance, and Dronacharya College of Engineering , Gurugram, Haryana for providing resources and support throughout the development of this research project.

REFERENCES

[1]. Google MediaPipe: https://google.github.io/mediapipe/
[2]. TensorFlow.js Models: https://www.tensorflow.org/js/models
[3]. S. Zhang et al., "Face detection using deep learning: An improved Faster R-CNN approach", IEEE, 2020
[4]. P. Viola and M. Jones, "Robust Real-time Object Detection," IJCV, 2001.
[5]. "Real-time Face Detection on Web", GitHub Repos, TensorFlow Examples.