"Hand Gesture Recognition with TensorFlow.js"

Group Members:

- Asylbek Oteubaev

- Dinara Ismagilova

Problem Description:

The ability to recognize hand gestures is a crucial part of human-computer interaction, enhancing user experience in various applications such as virtual reality, sign language interpretation, and contactless control systems. Despite the advancement in technology, achieving accurate and real-time hand gesture recognition remains a challenge, especially in browser-based applications.

Background:

Hand gesture recognition systems traditionally rely on hardware sensors and dedicated software. With the rise of machine learning and the advent of libraries like TensorFlow.js, it's now possible to achieve this directly in a web browser, enabling wider accessibility and usage. However, limitations in accuracy, latency, and environmental conditions still pose significant challenges.

Existing Solutions:

Current solutions vary from complex sensor-based systems to software relying on powerful machine learning algorithms. Some notable implementations include:

- Leap Motion Controller: A hardware device that captures hand motion.

- Google's MediaPipe Hands: A machine-learning solution that operates in native applications.

Links to the existing solutions:

- Leap Motion: [Leap Motion Website](https://www.leapmotion.com/)

- MediaPipe Hands: [MediaPipe Hands GitHub Repository](https://github.com/google/mediapipe)

Data Acquisition:

Generated in-house using a web camera to capture various hand gestures under different lighting conditions and backgrounds.

Proposed Solution:

Our solution leverages TensorFlow.js, a powerful library that allows for the training and deployment of machine learning models in a web environment. We aim to develop a system that can recognize a set of predefined hand gestures in real-time, using a web camera and without the need for any specialized hardware.

Technical Stack:

- TensorFlow.js for machine learning model training and inference.

- HTML/CSS/JavaScript for the web interface.

- WebRTC for capturing real-time video feed from the web camera.

- Python for data preprocessing and potentially training a model backend if needed.

- MediaPipe for efficient hand detection and key point extraction from the video feed.

- OpenCV and PIL (Python Imaging Library) for additional image processing tasks.

- Keras Layers like Dropout and Regularizers (e.g., l2) to prevent overfitting and improve model generalization.

Additional Information:

The project will explore the effectiveness of different neural network architectures, focusing on convolutional neural networks (CNNs) for image recognition tasks. We will also evaluate the use of transfer learning to improve model accuracy and reduce the amount of required training data.