Virtual Yoga Trainer with PoseNet API

Harshraj Mahesh (harshraj.mahesh@sjsu.edu)
Jojo Joseph (jojo.joseph@sjsu.edu)
Shivangi Jain (shivangi.jain@sjsu.edu)
Vanditt Sama (Vanditt.sama@sjsu.edu)

Computer Engineering Department, San Jose State University
San Jose, CA 95192 USA

***Abstract*— This paper covers our findings and learnings while building a Web-app and methodologies associated with it. This project leverages PoseNet API released by Google Creative Labs for Pose detection which requires a combination of computer vision and machine learning techniques to detect human poses in images or video. This paper discusses the development and training of the machine learning model for yoga poses and different components of an application on which the model is deployed. The application is built upon React.js library to maintain consistency and modularity between various application components and utilizes bootstrap for UI elements. Further discussed is connection with Firebase API for authentication and process of storing application/user data into Cloud Firestore.**

***Keywords*— Posenet, yoga-pose, workout, progress-tracker, TensorFlow, React.js, Authentication, Web-app, Machine Learning (ML), Firebase, Application programming Interface (API), Dataset**

I. Introduction

This is the final report submission of our team project completed during Fall-2019 semester in class of Enterprise software platforms (CMPE-272) at San Jose State University as graduate students.

Our mentor and instructor, Prof. Rakesh Ranjan, Innovation Leader, Director, Emerging Technologies at IBM Data & AI, directed our efforts in this project for selection of technology stack, market research, design thinking process, development and deployment of Virtual Yoga Trainer.

The core functionality uses a front camera or webcam on a device to detect users pose and matches it to a machine learning model trained using TensorFlow to determine correctness of a yoga pose. The app built around this idea enhances this functionality by providing several yoga poses along with it's advantages and images to guide users during a yoga workout session. Our aim is to provide a complete guided and automated workout routine for the users along with a progress tracker.

II. Preparing Machine Learning Model

A. Front-End

Pose estimation is a technique of detecting positions of body parts and joints in an image or a real time video and combining the data to predict the posture of the user. The main advantage of this API is that it does not require any additional or bespoke hardware such as infrared camera, X-Ray or Ultrasonic imagery. It is because the image processing takes place locally on a web browser, which is why it is also able to perform real-time analysis while also keeping the user data private.

Fig 2.1

The above figure (2.1) depicts how PoseNet represents the identified key parts and joints in a body. For each point, the output returned is the X and Y coordinates of the image that has been detected as a key point and a measure of how confidence the API is about this estimation. The following block is an example of high level output generated by the API.


```
{
  "score": 0.32371445304906,
  "keypoints": [
    { // nose
      "position": {
        "x": 301.42237830162,
        "y": 177.69162777066
      },
      "score": 0.99799561500549
    },
    { // left eye
      "position": {
        "x": 326.05302262306,
        "y": 122.9596464932
      },
      "score": 0.99766051769257
    },
    { // right eye
      "position": {
        "x": 258.72196650505,
        "y": 127.51624706388
      },
      "score": 0.99926537275314
    },
    ...
  ]
}
```


Fig 2.2

To use, Once the library is importing using <npm install @tensorflow-models/posenet>, a simple HTML code snippet will be able to integrate the API into the web browser.

```
<html>
  <body>
    <!-- Load TensorFlow.js -->
    <script src="https://unpkg.com/@tensorflow/tfjs"></script>
    <!-- Load Posenet -->
    <script src="https://unpkg.com/@tensorflow-models/posenet">
    </script>
    <script type="text/javascript">
      posenet.load().then(function(net) {
        // posenet model loaded
      });
    </script>
  </body>
</html>
```

B. Training the model using TensorFlow

To achieve pose matching and return data from fig 2.2 is returned in the form of image, Weighted matching technique has been utilized. First, a cosine similarity is calculated between the PoseNet data from the training image and the received image.

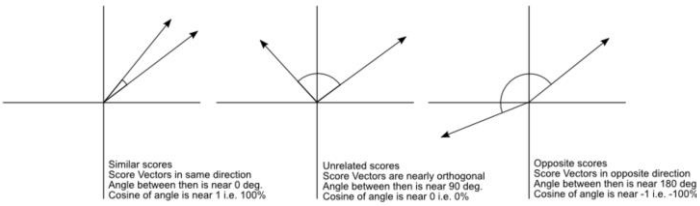

Fig 2.3

In the above figure, if the score vectors are similar, the model can assume that the key points between two images are similar. Following is the high level representation of the comparison operation.

$$D(Fxy, Gxy) = \sqrt{2 * (1 - cosineSimilarity(Fxy, Gxy))}$$

But, the confidence scores for a few key point might be low, this happens because the image set is 2-dimensional and the user could be in an irregular posture.

By not giving equal weight to the data with low confidence, the accuracy can be greatly improved. Google researchers George Papandreou and Tyler Zhu have derived a formula which incorporates this idea:

$$D(F,G) = \underbrace{\frac{1}{\sum_{k=1}^{17} Fc_k}}_{\text{Summation 1}} \times \underbrace{\sum_{k=1}^{17} Fc_k ||Fxy_k - Gxy_k||}_{\text{Summation 2}}$$

In the formula above, F and G are two vectors from the PoseNet data. Fck is the confidence score of the k^{th} keypoint of F. F_{xy} and G_{xy} represent the x and y positions of the k^{th} keypoint for each vector.

III. Developing a React Alppication

Content to be added.

A. Front-End

Content to be added.

B. User Authentication

Content to be added.

C. Google Firebase

Content to be added.

IV. Results

Content to be added.

V. Conclusions

Content to be added.

Acknowledgment

Content to be added.

References

[1] https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5

[2] https://medium.com/tensorflow/move-mirror-an-ai-experiment-with-pose-estimation-in-the-browser-using-tensorflow-js-2f7b769f9b23

[3] https://ai.google/research/pubs/pub47173

[4] https://fribbels.github.io/vptree/writeup