YogaSiddhi: AI-Powered Pose Analysis using MoveNet for Yoga Refinement

Ashis Kumar Mishra
Asst. Professor
School of Computer Sciences
Odisha University of Technology and Research, Bhubaneswar, Odisha, India

Debashis Sahoo
Research Scholar
School of Computer Sciences
Odisha University of Technology and Research, Bhubaneswar, Odisha, India

Ipsit Shubhankar
Research Scholar
School of Computer Sciences
Odisha University of Technology and Research, Bhubaneswar, Odisha, India

Isha Samal
Research Scholar
School of Computer Sciences
Odisha University of Technology and Research, Bhubaneswar, Odisha, India

ABSTRACT
This study dives into the world of AI (Artificial Intelligence) and its expansion in the sphere of technology, attempting to eliminate manual labor and provide comfort in achieving multiple goals easily. It, in particular, explores the implications of AI in the field of yoga and exercise, and studies the various ways to help people who struggle with doing yoga. The research aims to achieve a real-life yoga trainer experience through AI. Addressing the challenges in currently available AI yoga trainers, the study analyses various available yoga training models and simultaneously looks for an effective way to train the AI model to become a yoga trainer. With a glance into the future implications of the AI yoga trainer, the seminar concludes with the multiple ways to make the trainer as similar to a real-life yoga instructor as possible.

General Terms
Artificial Intelligence (AI), MoveNet, TensorFlow, pose estimation, deep learning.

Keywords
AI yoga trainer, MoveNet, Pose Estimation using MoveNet, yoga app, TensorFlow, deep learning using Python.

1. INTRODUCTION
Yoga has been practiced for decades, and its relevance in sustaining a healthy lifestyle is unrivalled, so much so that it has been incorporated into academic curriculum[1]. With the growing popularity of yoga, it is critical that those who practice it do so correctly in order to achieve the optimum outcomes. However, having an instructor for everyone is challenging, and not everyone can attend yoga sessions led by teachers. With AI producing near-human intelligence outcomes, we can use AI to create a yoga trainer that will not only coach individuals in executing yoga poses correctly but will also give facilities to help users access yoga from anywhere and everywhere.

2. LITERATURE REVIEW
Woodyard, C [1] assessed the conclusions of a few publications about the therapeutic applications of yoga and offered a thorough analysis of the advantages of consistent yoga practice. They concluded that yoga plays a key role in today's environment since it provides an all-encompassing approach to wellness and recovery that may promote psychological, emotional, and physical wellness. They also determined that yoga helps with a variety of health issues, including chronic discomfort, stress, depressive disorders, and anxiety. However, in order to avoid damage and maximize the therapeutic advantages of yoga, they pointed out that it must be practiced correctly. A safe and successful yoga practice may be achieved by practicing with knowledge, adapting postures to match individual requirements and goals, and getting advice from a competent instructor.

Balakrishnan, S. et al., [2] presented an artificial intelligence-based system for real-time position recognition and feedback throughout yoga practice sessions. The data points were extracted from each webcam picture using media pipe and OpenCV. The aforementioned data is subsequently fed into a deep learning model that utilizes a Convolution Neural Network(CNN) to detect pose faults, calculate the error percentage, and provides the user with the necessary feedback via voice assistant or text-based format. However, they note that there are certain drawbacks to employing media pipes, such as the necessity for high-quality photos and the possibility of pose detection mistakes.

The utilization of models based on deep learning, especially OpenPose was proposed by Narayanan, S. S., et al., [3] to recognize yoga poses and, if necessary, give feedback or adjustments. The model was using a collection of photos of various yoga positions and the outcomes were compared for 2D and 3D image points. IT was discovered that adding additional characteristics to the dataset increases the model's accuracy, and that using 3D point data leads to an improved prediction. They do, however, note certain limitations of existing technology, such as the need of a big and diversified dataset to train the model, the necessity for real-time feedback in yoga sessions, and the expense and technical skill necessary to run the equipment.

Li, M., et al., [4] describe a completely automated marker less motion capture technique for numerous interacting individuals that automatically detects the number of people and calculates the bone's length for each participant. This 3D hypothesis clustering approach combines information obtained from photometric appearance, the Multiview geometry, as well as bone length to match 2D joints for the same individual across multiple views, from which 3D joints may be inferred. With integration of Multiview appearance evidence, geometry constraints, and bone length restrictions, the technique generates credible 3D skeletons from the ground up. This method was examined and contrasted to state-of-the-art approaches on many benchmark datasets, examined and contrasted to state-of-the-art approaches on many benchmark

datasets, comprising Panoptic, Shelf, and Campus and was noted for producing significantly lower estimation errors.

XNect technology, which enables real-time multi-person three-dimensional human posture estimate with a single RGB camera was introduced by Mehta, D., et al. [5]. XNect is intended to estimate the 3D posture of numerous persons in a scenario, even when occlusions and interactions are present. This employs a CNN with a first core network which divides into two distinct branches for 2D pose estimation as well as 3D pose encoding. But XNect's accuracy is not yet equivalent to that of multi-view capture methods, and it may still fail owing to inaccurate 2D posture estimations or part associations, in addition to instances in which the neck is hidden for effective human recognition.

The research model as proposed by M. U. Islam, et al., [6] for posture identification makes use of Microsoft Kinect to identify human joint points in real time. The system can identify twenty human joints and extract the coordinates of these joint locations for the purpose of recognizing yoga poses by acquiring colour, depth, and skeleton information. It is designed to recognize and evaluate yoga postures with high accuracy by focusing on specified poses and using particular joint sites for accurate recognition, including the wrists. One drawback as mentioned, though, is that it could have trouble identifying complex as well as dynamic yoga postures that call for rapid motions or adjustments to body alignment. Furthermore, the system's capacity to adjust to individual differences in body proportions or motions may be limited by the dependence on certain joint locations for pose identification, which might have an impact on the pose recognition process's overall resilience and accuracy.

This study conducted by B. Jo and S. Kim, [7] compares the MoveNet models, PoseNet and OpenPose, to estimate pose on mobile devices. They compared the models' characteristics and effectiveness in the same setting. In all comparative trials, the effectiveness of the various Human Pose Estimation libraries was evaluated using the percentage of detected joints (PDJ). According to the results, MoveNet Thunder appears to be the quickest model, whilst OpenPose remains the slowest. While the speeds of MoveNet Lightning & PoseNet are comparable. OpenPose & MoveNet Thunder outperform PoseNet & MoveNet Lightning in terms of accuracy. According to them, the selection of model is determined by the application's unique needs, which include speed and accuracy.

3. OBSERVATION

There are different methods for creating a trainer, and comparing and analyzing them is critical for progressing in the study. The study is to investigate several approaches and procedures involved in developing the trainer, as well as to identify the optimal option at each stage of the process. Compiling them can lead to the optimal solution.[6]

3.1 MoveNet vs Mediapipe

The first step in creating a trainer is to create a model that can analyze real-time images and determine whether or not it is correctly mimicking a yoga stance[2]. Two common frameworks provide this functionality of real-time picture analysis: the mediapipe library and the MoveNet library. Both work with real-time photos, however MoveNet has an advantage over Mediapipe due to its simplicity and speedier results[7].

3.2 MoveNet Thunder vs MoveNet Lightning

MoveNet comes in two models: thunder and lightning. Both function with precision at a real-time speed of 30 frames per second. While Lightning aids in the development of speedier applications, Thunder is designed for applications that prioritize high accuracy. As a result, Thunder is popular in applications such as yoga trainer.

3.3 TensorFlow vs OpenCV

Both the TensorFlow and OpenCV libraries work on image preprocessing and offer Machine Learning features. However, the presence of MoveNet in the TensorFlow hub acts as a determining factor when choosing a TensorFlow library.

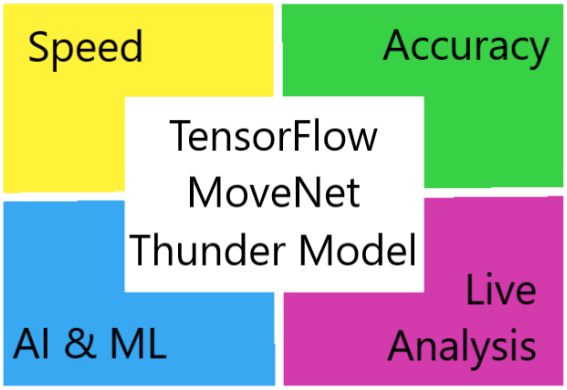

Figure 1: TensorFlow MoveNet Thunder Model

4. PROPOSED METHOD

Introducing TensorFlow MoveNet, a precise and quick model for identifying keypoints in the human body, is used in the construction of the model. Two datasets Active, an internal Google dataset and COCO were used to train MoveNet. It recognizes 17 important bodily parts. This model is available on TF Hub in two models: Thunder and Lightning. Both operate at a real-time speed of 30 frames per second with accuracy. These versions function well on contemporary laptops, smartphones, and desktop computers. The health and fitness industries employ this strategy more often.

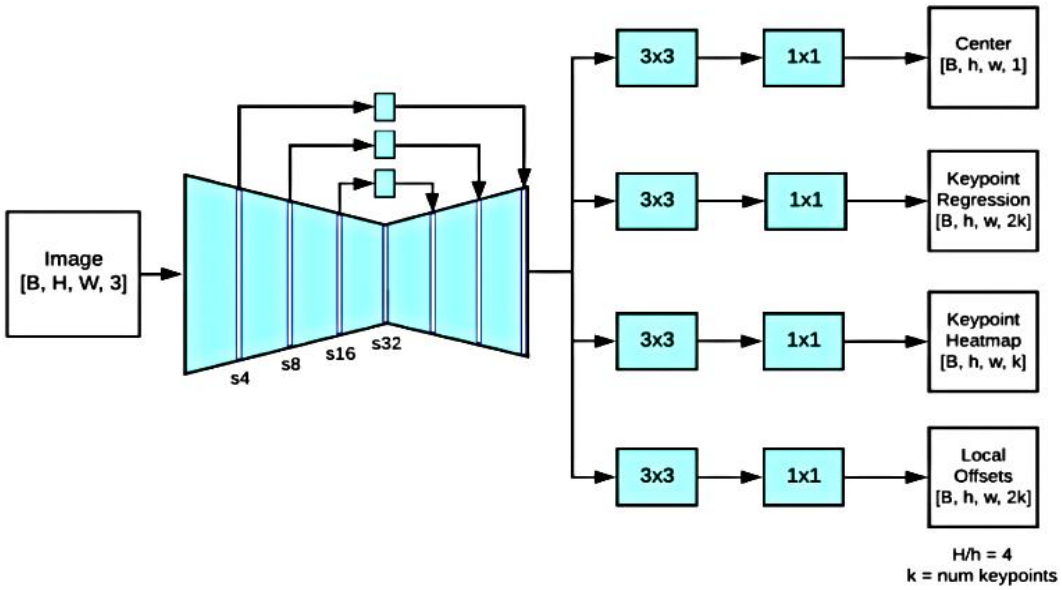

Figure 2: Architecture of TensorFlow MoveNet

Here, the input tensor dimensions are denoted by [B, H, W, 3], where the batch size, or B, is the quantity of photographs processed concurrently, H is an image's height, W is an image's width, and 3 represents the number of channels in the input pictures, typically red, green, and blue[7]. Scales ranging from s4 to s32 represent different feature map resolutions, supporting low-resolution(s32) for broad context like pose estimation and high-resolution(s4) for specific details like object recognition. 3x3 convolutional layer collects larger-scale features while a 1x1 retrieves finer-scale details.

TensorFlow MoveNet employs a bottom-up methodology. The design consists of many prediction heads and a feature extractor. Even with some major modifications to improve speed and accuracy, prediction techniques still mostly employ CenterNet. All models are trained using TensorFlow Object Detection API. MoveNet produces semantically rich, high-resolution feature maps (output stride 4) using its feature pyramid network (FPN) in combination with the feature extractor, MobileNetV2[7]. The four prediction heads of the feature extractor are:

4.1 Individual Center Heatmap

By averaging related keypoints, it forecasts the center of an individual instance. based on the inverse distance from the frame center, determines the spot with the greatest score.

4.2 Keypoint Regression Field

Essential for grouping, it forecasts full-person keypoints. Initial keypoints derived from the object center pixel; center-out prediction across scales may restrict accuracy.

4.3 Person Keypoint Heatmap

Accurately forecasts every keypoint, regardless of example. In order to minimize scores for remote background keypoints, each heatmap pixel is inversely weighted by distance from the regression keypoint.

4.4 2D Per-Keypoint Offset Field

Finding the coordinates in each keypoint channel with the maximum values allowed for the final forecasts. Overall estimations are refined using local 2D offset forecasts.

We followed MoveNet Thunder model for better speed as well as high accuracy that can detect 17 keypoints for real-time human pose detection and yoga pose classification. We can refer to figure 4 and table 1 to get a proper insight of keypoints detected by our posed model.

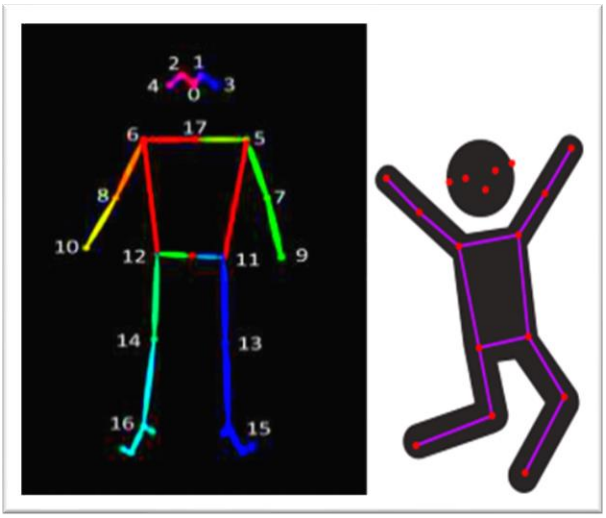

Figure 3: 17 Keypoints using TensorFlow Movenet

Table 1: 17 Keypoints names of human body using MoveNet

Index	Body Keypoint	Index	Body Keypoint
0	Nose	9	Left Wrist
1	Left Eye	10	Right Wrist
2	Right Eye	11	Left Heap
3	Left Ear	12	Right Heap
4	Right Ear	13	Left Knee
5	Left Shoulder	14	Right Knee
6	Right Shoulder	15	Left Ankle
7	Left Elbow	16	Right Ankle
8	Right Elbow	-	

5. IMPLEMENTATION

The implementation starts with a two-phase solution that includes a powerful "Training Pipeline" for model building.

Next follows the incorporation of the learned model into an interactive "Web Application" that can be used in real-world scenarios.

5.1 Training pipeline

The entire model development process is included in this step. It includes gathering data, preparing it for analysis, preprocessing and training the model. For the model to be able to extract features and patterns from the given dataset, this step is essential. By using learnt patterns, the training pipeline makes sure the model is prepared to produce precise predictions or classifications[3].

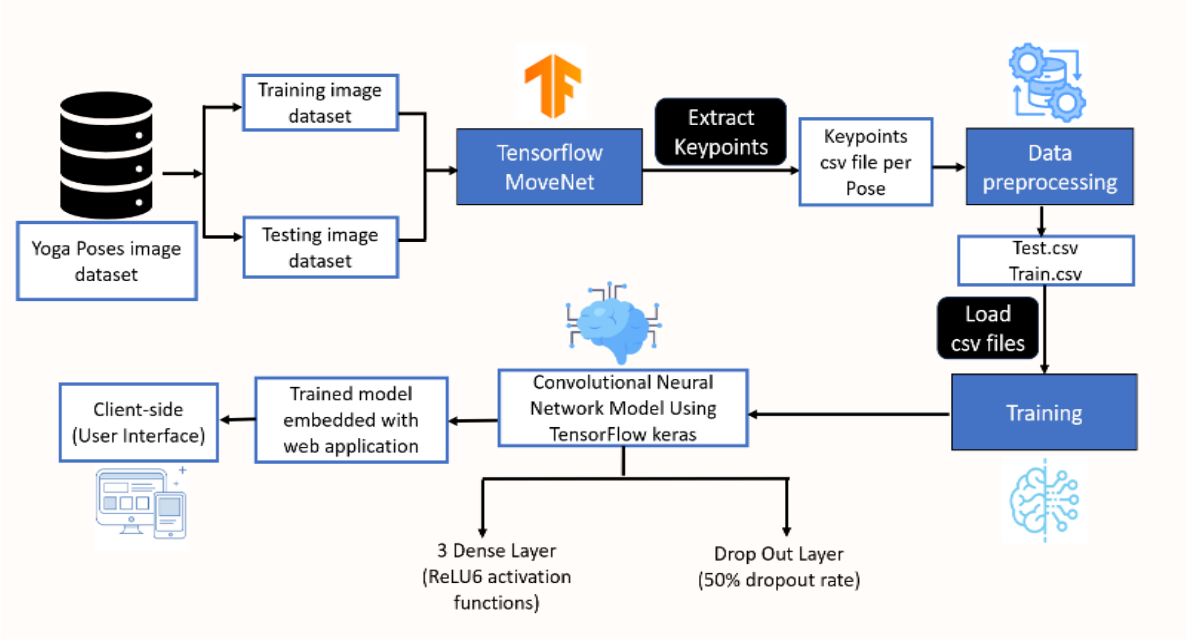

Figure 4: Implementation Flow diagram of YogaSiddhi

5.1.1 Images data collection

A labelled dataset of 8 different yoga positions that were tagged is collected. To enable efficient model training and assessment, this dataset was carefully divided into subsets for testing and training. With more than 2300 images in the enormous collection, a wide-ranging and in-depth analysis of the yoga positions is guaranteed. All the images are carefully selected and organized into specific folders so that the dataset has structure and clarity for the purpose of preprocessing and training the model.

5.1.2 Data Preprocessing

Definitions for posture estimation data types and structures, including people, categories, keypoints, points, rectangles, and body sections are provided. Then, important crop areas for inference using the MoveNet TFLite model are established and proceeded for data preprocessing.

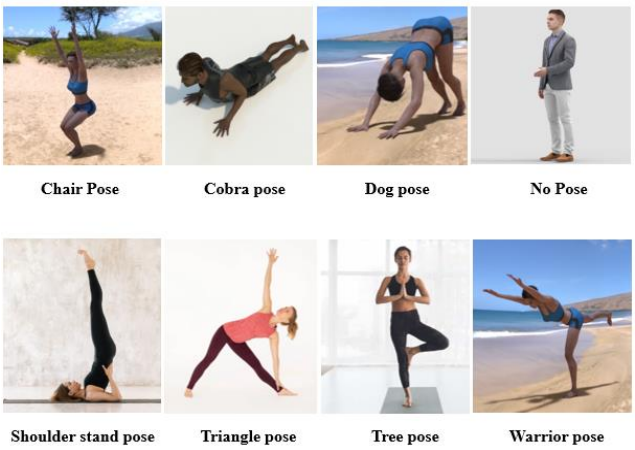

Figure 5: Yoga Poses used for classification

Keypoints on the pictures are predicted by preprocessing the image data, which then stores the keypoints in a CSV file for use in the classification job later. To begin preprocessing, the path to the folder holding the images is retrieved. Next, the location of the CSV file containing the keypoints that have been preprocessed is specified. The images in the designated folder are then preprocessed, and the keypoints are saved to a CSV file. Finally, it creates a single CSV file from the combination of all per-class CSV files and returns it.

5.1.3 Model Training

Model Training involves several steps to be followed. Initially

the CSV files containing the training and testing data is loaded. Next, the data is preprocessed by scaling the landmarks to a consistent pose size and normalizing their translation. Then Generates a Keras model using the architecture as follows:

- 34 inputs total x, y, and scores for each of the 17 body parts make up the input layer.
- 128-unit dense layer with ReLU6 activation
- 50% dropout rate dropout layer
- 64-unit dense layer with ReLU6 activation
- 50% dropout rate dropout layer
- Output layer with SoftMax activation.

It uses the Adam optimizer and the categorical crossentropy loss function to compile the model. A batch size of 16 is taken to train the model on the training data for 200 epochs, verifying against the validation data. Finally, test data is evaluated and save model to Tensorflow.js directory in JSON format.

5.2 Web Application

The trained model is applied in real-world circumstances through the creation of a web application in the second stage. The popular JavaScript library React is utilized in the development of the application's front end. Because of React's adaptable user interface, users may interact with the trained model and utilize its insights with ease. The web-based interface of the implemented solution enhances its usability and practicality by ensuring accessibility and usefulness on a variety of devices. We have introduced Two mode as Manual yoga pose selection and Automatic yoga pose detection in real-time operation.

Node.js is used for server-side components, including API endpoints and AI model Integration. The website can update Current pose, pose time, and best time for a pose due to modern technologies like MoveNet Thunder, React Cam and are integrated using node.js that improve the overall speed and user experience of the system.

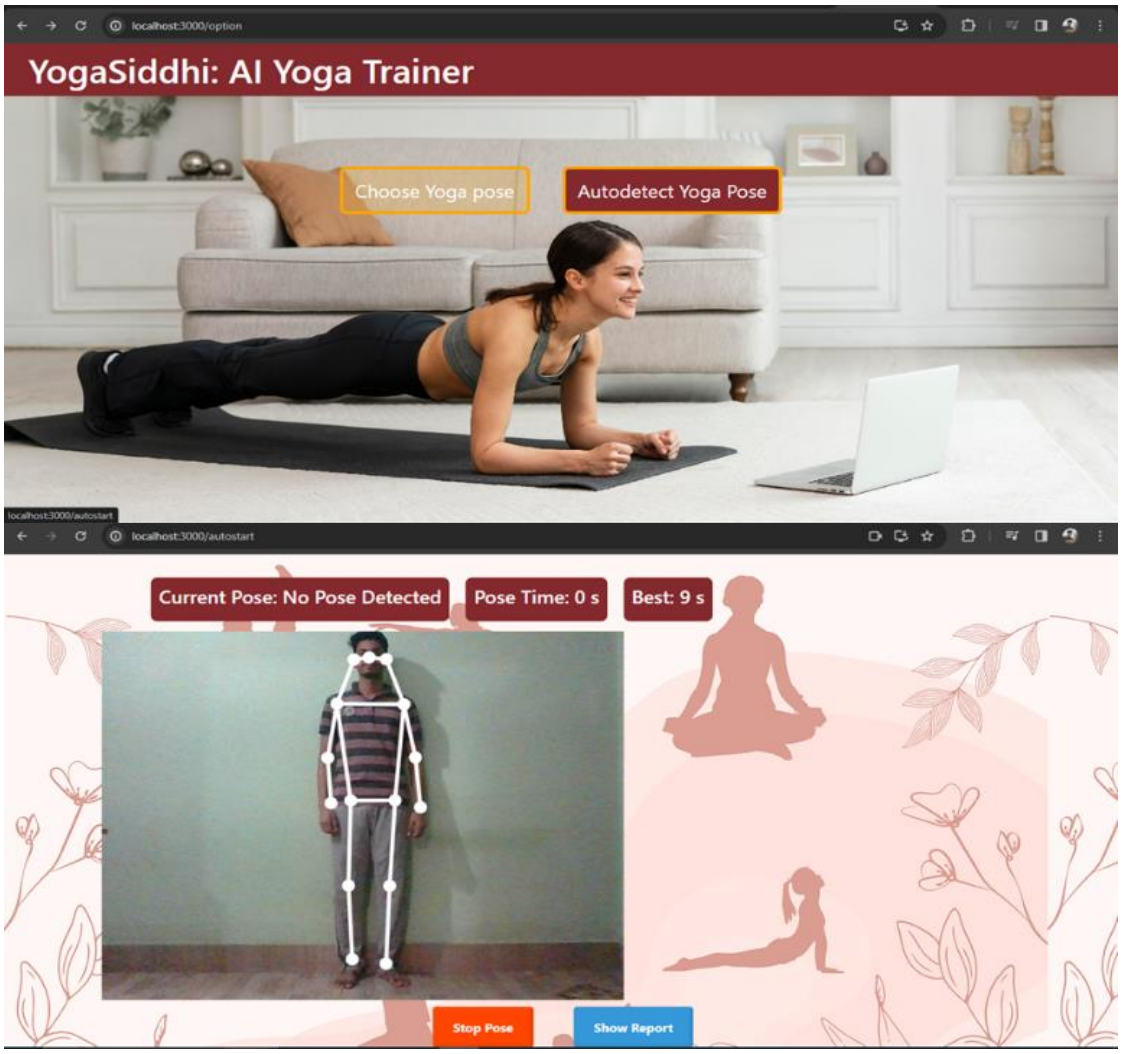

Figure 6: Website with Manual and Automatic modes

6. RESULT ANALYSIS

With the building of the AI trainer, it is important to analyze the results and performance of the model based on basic performance standards.

6.1 Web Application Performance Result

The website renders the trained MoveNet Thunder model smoothly, enabling instantaneous keypoint representation of the whole human body without perceptible delay. A skeleton view that constantly adapts to the motions of the body is formed by the complex connections between the keypoints. To ensure

synchronous motion, this adaptation is accomplished by using the center point as a reference from which vectors are generated. When the user hits one of the eight preset positions with an accuracy higher than 97%, the skeleton view turns green instead of white. Concurrently, an auditory feedback countdown starts, furnishing users with a systematic time system. The interface tracks and updates the best time attained for a particular posture in addition to providing users with updates on their current pose and the amount of time that has passed.

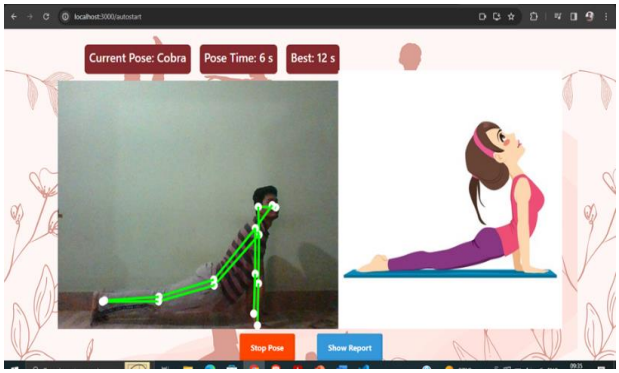

Figure 7: Demo of WebApp with Cobra pose

The website provides a thorough report on the performance of each posture, enabling users to examine their development in greater depth. Users may customize their experience to suit their tastes with the two modes for real-time pose detection manual and automated. Utilizing the MoveNet Thunder model for position prediction and React Cam and integration using Node.js for real-time user feedback, the system maintains a lightweight design that guarantees quick and effective use.

Figure 8: Report on yoga session completion

The website is notable for its capacity to provide quick and precise real-time posture detection in addition to an easy-to-use design that supports a wide range of user types.

6.2 Evaluation result

Yoga position recognition is accomplished using Convolutional Neural Networks. Chart 1 displays the model accuracy for the training and validation data, whereas Chart 2 displays the obtained model loss.

Final LOSS: 0.008070270530879498 Final ACCURACY: 0.9965792298316956

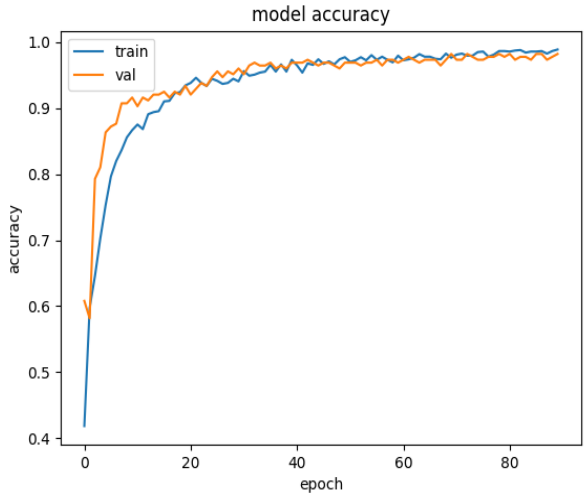

Chart 1: Accuracy of the proposed trained model

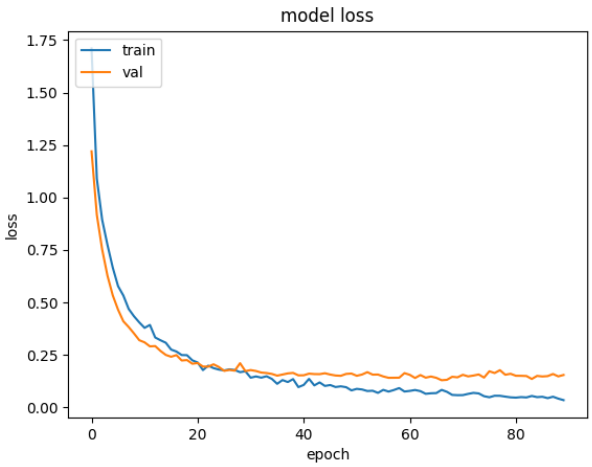

Chart 2: Loss of the proposed trained model

It is important to draw attention to the remarkable results that TensorFlow MoveNet and Convolutional Neural Network[5] achieves in the yoga pose identification challenge. The model's ability to minimize mistakes is demonstrated by the exceptionally low loss value of 0.008070270530879498, which is the result of training. Moreover, the obtained accuracy of 0.9965792298316956 exhibits a remarkable degree of precision, confirming the model's ability to accurately identify and categorize yoga poses.

To sum up, the model is a reliable solution for yoga position detection because of its exceptional precision and little loss, which highlight its effectiveness in producing better outcomes.

7. CONCLSION & FUTURE SCOPE

It is notable to highlight the importance of using AI in the field of Yoga Science, as it not only helps in accessing coaching from anywhere and everywhere, but also helps in keeping a track of a person's progress[4]. The MoveNet and TensorFlow libraries surely help in identifying keypoints that helps in training the AI model in an effective way. Using them not only helps reduce time in training the model, but also help in providing real-time results, making it even more preferrable.

Although the MoveNet and TensorFlow libraries are better choices for building AI yoga trainers, not much research has been done on them, and with proper exploration, they can surely change the way AI models are trained in the future. The dataset for training the model can be further increased to achieve even more accuracy. The user interface can be loaded with multiple other features to track progresses and make them even more user friendly, and can be launched on multiple platforms, in the form of applications, softwares and websites, to make it easily accessible for users. In conclusion, with proper usage, we can unlock the true potential of AI in revolutionizing the future generation.

8. REFERENCES

[1] Woodyard, C. (2011). “Exploring the therapeutic effects of yoga and its ability to increase quality of life” International Journal of Yoga.

[2] Balakrishnan, S., Shriya, S., Srajana, Vaishnavi N., & Dr. Kavitha C. (Year). "The AI Yoga Trainer Using Artificial Intelligence and Machine Learning." International Journal of Computer Research and Technology (IJCRT).

[3] S. Sankara Narayanan, D. K. Misra, K. Arora, H. Rai. "Yoga Pose Detection Using Deep Learning Techniques." In Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021.

[4] M. Li, Z. Zhou, and X. Liu. "3D Hypothesis Clustering for Cross-View Matching in Multiperson Motion Capture." Computational Visual Media, vol. 6, no. 2, June 2020, pp. 147– 156.

[5] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu. "XNect: Real-time Multi-person 3D Motion Capture with a Single RGB Camera." ACM Transactions on Graphics, July 2020, Volume 39, Issue 4.

[6] M. U. Islam, H. Mahmud, F. B. Ashraf, I. Hossain and M. K. Hasan, "Yoga posture recognition by detecting human joint points in real time using microsoft kinect," 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, Bangladesh, 2017, pp. 668-673, doi: 10.1109/R10-HTC.2017.8289047.

[7] B. Jo, S. Kim. "Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices." Technology and Science, Volume 39, Issue 1, Pages 119-124.