IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

AI YOGA ASSISTANT

[1]**Abhishek Chauhan**, [2]**Jaideep Solania**, [3]**Kanika Gola**
[1]Student, [2]Student, [3]Student
[1]Department of Computer Science (Artificial Intelligence),
[1]ABES Institute of Technology, Ghaziabad, India

Abstract : In recent years, yoga has become an integral part of life for many individuals worldwide. Consequently, there is a growing need for the scientific examination of yoga postures. It has been noted that pose detection techniques can be employed to recognize postures and help individuals perform yoga more precisely. However, recognizing postures presents a challenge due to the scarcity of datasets and the difficulty of real-time posture detection. To address this issue, a comprehensive dataset containing at least 5500 images of ten distinct yoga poses has been compiled. A tf-pose estimation algorithm, which draws a skeletal representation of the human body in real-time, is utilized. Joint angles in the human body are extracted from the tf-pose skeleton and used as features for various machine learning models. 80% of the dataset is allocated for training, while 20% is reserved for testing. This dataset has been evaluated using different machine learning classification models, achieving a 99.04% accuracy with a Random Forest Classifier.

IndexTerms **- YOGI - Yoga Gesture Identification dataset, Computer Vision, Machine Learning, Classification, Gesture Recognition.**

I. INTRODUCTION

Yoga, which originated in ancient India, is a comprehensive practice encompassing mental, physical, and spiritual well-being. For many years, yoga and sports have captivated people, but the past decade has seen a significant surge in individuals incorporating yoga into their daily routines, primarily due to its health benefits. Proper execution of yoga, especially maintaining correct postures, is crucial. However, it is often observed that people, lacking adequate guidance or knowledge, attempt yoga independently and risk injury due to incorrect postures. While practicing yoga under a trainer's supervision is ideal, it may not be affordable for everyone. Consequently, many turn to mobile apps to learn yoga poses, yet they remain uncertain if they are performing them correctly. To address these challenges, significant advancements have been made. Computer vision and data science techniques have been employed to develop AI software that functions as a virtual trainer. This software not only highlights the benefits of each pose but also evaluates the accuracy of the user's performance. This innovation allows individuals to practice yoga without needing a personal trainer. Extensive image datasets have been created to facilitate machine learning and deep learning models, encompassing ten yoga poses: Vriksasana, Utkatasana, Virabhadrasana I, Parsva Urdhva Hastasana Baddha Konasana, Standing Bhujangasana, Sukhasana, plank, and Virasana. Features were extracted using computer vision techniques and the tf-pose algorithm. This algorithm creates a skeleton representation of a human body (as illustrated in Figure 4) by identifying all the joints and linking them, resulting in a stick figure known as the body's skeleton. The coordinates and angles formed by the joints are extracted with this algorithm and utilized as features for machine learning models. Various machine learning models were employed to determine the test accuracy, with the Random Forest classifier achieving the highest accuracy among all the models.

II. RELATED WORK

Research has explored the detection and correction of yoga poses. Some investigators have employed the Kinect device to model human postures. This device captures images and is notable for its built-in infrared laser projector, multi-array microphone, and RGB camera, which are used to capture both color and depth images. Additionally, the Kinect device includes a tool that constructs a 3D skeleton of the human body, providing information about joint coordinates. Although effective, the primary drawback of this method is the high cost and lack of user-friendliness of the Kinect device. To address this issue, the tf-pose Algorithm has been utilized. This algorithm generates a skeleton of the human body, providing the necessary information about joint positions. By finding the coordinates of these joints, one can detect body postures. Paula Pullen and William Seffens used the Kinect sensor's visual Gesture Builder feature to capture yoga postures with high accuracy. Similarly, Edwin W. Trejo and Peijiang Yuan used Microsoft Kinect v2, which offers greater accuracy and precision but requires more computational time for complex models. The Kinect camera functions based on three main components: depth, color, and body tracking. Utilizing all the features of the Kinect device, they created a Computer Interaction system aimed at training purposes. An Adaboost Algorithm was employed to identify six common yoga poses. Various researchers have explored different applications of the Kinect device, concluding that it is effective for depth, color, and body tracking. Another method for pose detection involves using a Convolutional Neural Network (CNN). In their approach, the authors utilized a dataset containing six yoga asanas. They implemented a hybrid deep learning model combining CNN and LSTM to recognize yoga poses. The CNN extracts features from

the points obtained by OpenPose, and the LSTM is used to identify the yoga poses. This module achieved a 99.04% accuracy on a single frame.

III. DATASET COLLECTION

At present, sourcing a comprehensive and efficient dataset of yoga poses online presents a formidable task. The YOGI dataset encompasses an array of standing and sitting poses, employing the entirety of the body to depict each yoga posture. With a diverse range of hand and leg positions, it poses a challenge for posture detection algorithms to operate with optimal efficiency. Recognizing the aforementioned issue, the YOGI dataset comprises 10 yoga postures, meticulously captured utilizing the burst feature of a DSLR camera. These images boast exceptional precision and accuracy. Within this dataset, each of the 10 yoga poses is represented by a class containing approximately 400 to 900 images. In total, the compiled Color Image dataset encompasses 5459 images. Four example yoga poses are illustrated in given figure 3.1.

Fig 3.1: Images of the YOGI dataset

3.1 The procedure followed in collecting YOGI dataset

Gathering data manually from a sizeable dataset demands significant dedication, focus, and accuracy. Here's the outlined procedure:

· A controlled environment devoid of direct sunlight was utilized to ensure image clarity, eliminating any potential for reflection or glare.
· The camera was securely mounted on a tripod and positioned appropriately to center the individual executing yoga poses, maintaining a consistent distance of approximately 4 to 5 meters between the camera and the subject.
· A plain white background was employed to accentuate and differentiate the yoga poses performed by the individual.
· Multiple images of each pose were captured from various angles and directions, capturing the nuances of every pose to compile a comprehensive real-time dataset.
· Each pose was photographed in continuous mode, capturing 25 images in a single sequence.

IV. EXPERIMENTS AND RESULTS

After collecting the YOGI dataset the following steps were followed as shown in Fig 4.1.

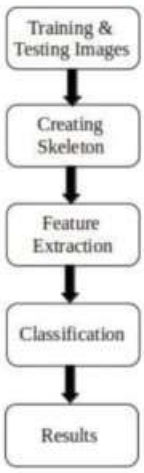

Fig 4.1: Flowchart of the procedure followed on the dataset

4.1 Creating Skeleton

Initially, the luminosity of each image underwent augmentation, with the enhancement parameter standardized at 2.0 for consistency. Subsequently, the images were adjusted to a resolution of 500x500 to optimize compatibility with the pose estimation algorithm, ensuring precise and reliable results. The tf-pose-estimation algorithm is employed to generate a skeletal representation of individuals executing yoga postures, delineating each bodily joint and linking them into a skeletal framework, illustrated in Fig 4.2. Remarkably, the algorithm exhibits real-time efficacy, delivering highly accurate outcomes.

Fig 4.2: Skeleton made using Tf-pose estimation

4.2 Feature Extraction

In the subsequent phase, the tf-pose-estimation algorithm is employed to derive the coordinates of the joints, as illustrated in Fig 4.3. These coordinates serve as the basis for computing 12 distinct angles, crucial for the detection and adjustment of yoga poses. The formula for angle calculation is provided below.

Here's an equation:

To find the distance between two points
a = (x1- x2)2+ (y2- y1)2 (2)
Where, (x1,y1) is the coordinate of point p1, and (x2,y2) is the coordinate of p2

$a^2=b^2+c^2-2bc\cos A(1)$
Where,
a = Distance between point p1 and p2
b = Distance between point p2 and p3
c = Distance between point p1 and p3

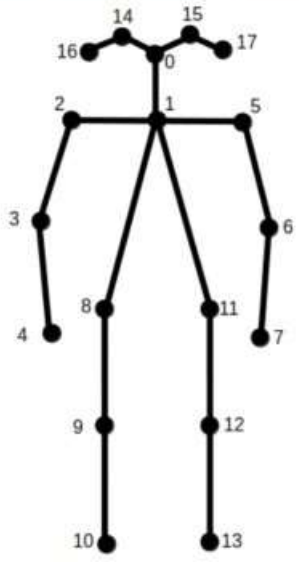

Fig 4.3: Stick Diagram using tf-pose Algorithm

4.3 Classification

In the concluding phase, attributes are saved in a CSV document and appropriately tagged. Subsequently, the data is divided into training and testing sets at an 80:20 ratio, illustrated in Table-I. Six machine learning classification models—Logistic Regression, Random Forest, SVM, Decision Tree, Naive Bayes, and KNN—with various parameters were employed as benchmarks for the dataset. Post final assessment, 25 accuracy outcomes were obtained, detailed in Table 4.2.

Table 4.1: Description of Dataset

Category	*Number of Example*
Training set	4367
Test set	1092

Table 4.2: Benchmark of YOGI Dataset

Classifier	*Description*	*Accuracy*
Logistic Regression	IterationNumber: 1000, Solver: Newton-cg	0,8215
	IterationNumber: 1500, Solver: Newton-cg	0,8302
	IterationNumber: 2000, Solver: Newton-cg	0,8379
	IterationNumber: 2500, Solver: Newton-cg	0,8316
Random Forest	n estimator: 30, MaxDepth: 7	0,9926
	n estimator: 30, MaxDepth: 10	0,9972
	n estimator: 30, MaxDepth: None	0,9990
SVM	Kernal Function: Linear, Loss Function: Hinge	0,8791
	Kernal Function: Polynomial, Loss Function: Hinge	0,9358
	Kernal Function: Radial Basis Function, Loss Function: Hinge	0,9871
Decision Tree	MinSampleLeaf: 1, MinSampleSplit: 2	0,9771
	MinSampleLeaf: 2, MinSampleSplit: 2	0,9670
	MinSampleLeaf: 3, MinSampleSplit: 2	0,9679
	MinSampleLeaf: 1, MinSampleSplit: 3	0,9752
	MinSampleLeaf: 2, MinSampleSplit: 3	0,9670
	MinSampleLeaf: 3, MinSampleSplit: 3	0,9761
	Naive Bayes Distribution: Normal	0,7475
KNN	Neighbours: 3, DistanceWeight: Equal, Distance: Euclidean	0,9899
	Neighbours: 3, DistanceWeight: Inverse, Distance: Euclidean	0,9826
	Neighbours: 5, DistanceWeight: Equal, Distance: Euclidean	0,9853
	Neighbours: 5, DistanceWeight: Inverse, Distance: Euclidean	0,9901
	Neighbours: 7, DistanceWeight: Equal, Distance: Euclidean	0,9826
	Neighbours: 7, DistanceWeight: Inverse, Distance: Euclidean	0,9890
	Neighbours: 9, DistanceWeight: Equal, Distance: Euclidean	0,9725

V. ALGORITHMS

The one-dimensional array, once flattened, is passed to the initial fully connected layer. The first fully connected layer starts with weights of size 1000x198 and a bias of -1000. Subsequently, the succeeding layer has weights sized 700x1000 with a bias of -700. Following this progression, the outcomes of this operation, applied to a 1x700 image, are forwarded to the subsequent continuous process. This process involves weights of 500x700 and a bias of 500, resulting in a disparity of 82. The entire cross-outer layer employs the ReLu initialization function, while the final connection layer also adopts ReLu initialization. The output layer, however, employs the softmax initialization function. To optimize the loss model, this model utilizes the Adam optimizer [17]. It incorporates ReLu activation function, denoted as R(z) where R(z) = z if z > 0, otherwise 0, and softmax activation for multiclass classification.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Adaptive Moment(Adam) Estimation algorithm is used for optimization.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

VI. SYSTEMARCHITECTURE

The Yoga AI Trainer consists of three main components: the Meditation Coach, Exercise Trainer, and Yoga Assistant, all seamlessly integrated within the Flask web application. Leveraging Mediapipe, the system captures 33 key points of the human pose for analysis. Users have the flexibility to either select the specific model they require assistance with or allow the system to make a recommendation. Client frames are transmitted to the server for processing, where the pre-CNN model is initialized upon the initial request. Within the Yoga Assistant module, a CNN algorithm is deployed using Keras, employing categorical cross-entropy for loss calculation and the adam optimizer for loss minimization.

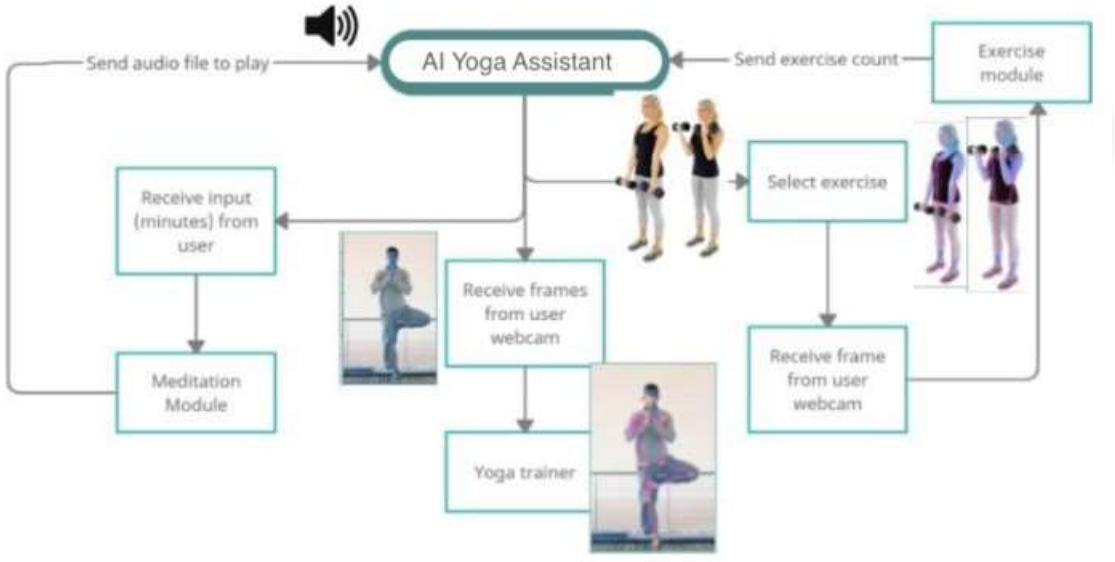

Fig 6.1: System Architecture of AI Yoga Assistant

6.1 Module Design Specification

The Yoga AI Trainer encompasses three core components: the Yoga Assistant, Exercise Trainer, and Meditation Coach, all seamlessly integrated within a web-based platform.

6.1.1 Yoga Assistant Module

· Users have the option to specify the model they require assistance with or allow the system to infer based on their needs. Upon initiation, frames from the client are transmitted to the server, triggering the loading of the pre-CNN model.
· Leveraging the Mediapipe Blazepose framework, the module extracts the relevant area from the input image and overlays it with graphical annotations.
· The primary subject is transformed into a structured array resembling dimensions of 33x2, subsequently fed into the CNN model, yielding an output comprising 82 distinct values.
· The module furnishes the user with either the precise value corresponding to the selected pose or, in the absence of a specific selection, the pose associated with the highest computed value. Training of the CNN model was facilitated using the Yoga-82 dataset.
· Data management within the module involves parsing the Yoga-82 dataset, comprising image URLs for 82 unique asanas. Through a systematic process, all acquired images are cataloged within a CSV file, serving as image references. Subsequently, these images are processed to generate training and testing datasets by converting them into 33x2 arrays.

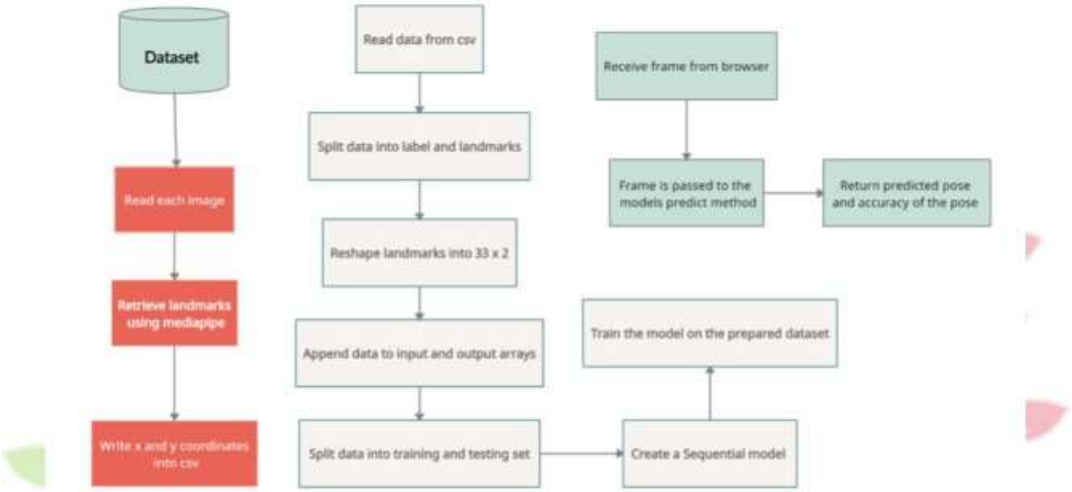

Fig 6.2: Yoga Assistant Architecture

The CNN model underwent training with the dataset, yielding a commendable accuracy of 98.9%.

6.1.2 Exercise Trainer Module

· Users are prompted to choose their desired exercise, upon which the web page, tailored to the selected routine, is deployed to the client, furnishing a visual guide on execution steps.
· Initiating the session triggers the transmission of webcam frames from the user to the server.
· Employing the Mediapipe Blazepose model, the input image's key-points are captured and depicted.
· Pertaining to the chosen exercise, angles and distances between joints are computed [e.g., shoulder, elbow, wrist for bicep curls].
· Angle calculations undergo scrutiny against predefined thresholds; surpassing them prompts phase adjustments, along with feedback and image updates sent back to the user.
· Client-server communication occurs every 500 milliseconds, maintaining real-time angle and distance calculations based on the chosen exercise [e.g., shoulder, elbow, wrist for bicep curls].

· Evaluated angles are subject to threshold checks; if exceeded, counts are incremented, triggering phase modifications.

Customers are reimbursed with replacement costs and images upon phase adjustments.

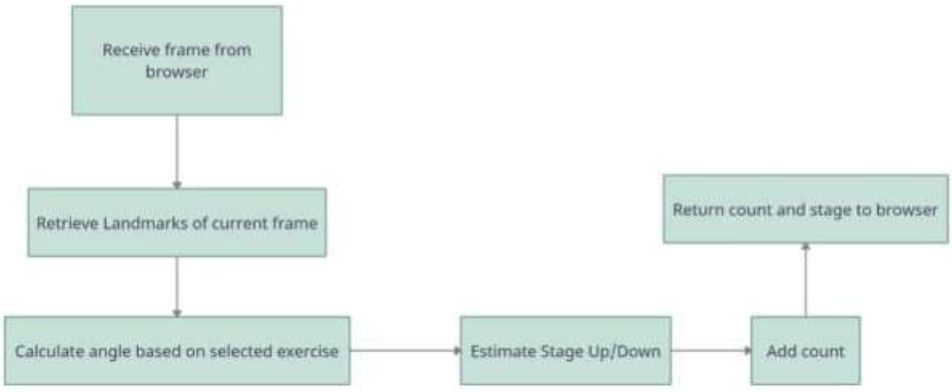

Fig 6.3: Exercise Trainer Module

6.1.3 Meditation Module

· Time input from the client is received by the meditation module, prompting the creation of a thread set to operate for the designated duration.

· Subsequently, the audio file "From Your Eyes.mp3" is transcribed. Following a brief pause of 2 seconds, audio files "breathIn.mp3" and "breathe out.mp3" are converted into 3-second audio segments.

· Continuously, the user monitors the ongoing audio playback, switching to any newly introduced audio files. Upon expiration of the allocated time, the thread halts, and the corresponding audio file is removed.

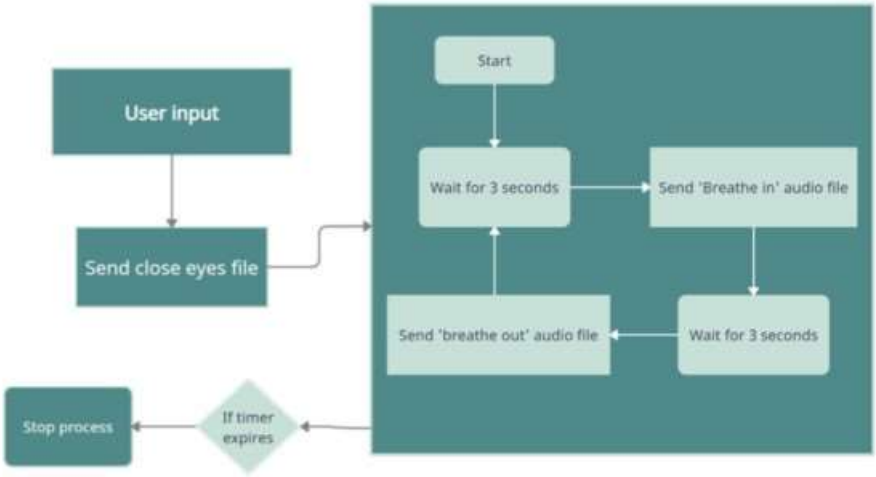

Fig 6.4: Meditation Coach Architecture

VII. CONCLUSION

This paper proposes a system for classifying ten yoga poses, leveraging a dataset comprising six machine learning classification models. Detection of yoga poses relies on angles derived from Skeleton joints via the TF pose estimation algorithm, achieving an overall accuracy of 94.28% across all models. Data preprocessing and model training were conducted using Google Colab and the Ubuntu 18.04.4 LTS terminal. Future plans involve expanding the YOGI dataset to encompass additional yoga poses and integrating deep learning modules for enhanced performance, along with the inclusion of an audio guidance system. Additionally, a CNN model enables yoga trainers to accurately assess 82 yoga poses, constructed within a Sequential model architecture, achieving a test accuracy of 98.9%. Training and testing utilize the yoga-82 dataset, comprising an average of 347 images per asana, divided into an 80:20 ratio for training and testing, respectively. Evaluation metrics include categorical accuracy and loss, with accuracy calculated by dividing the total correct predictions by the total test dataset size, which represents 20% of the yoga-82 dataset.

REFERENCES

[1] Yoga Pose Detection using deep learning techniques Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2023.

[2] AI-Based Yoga Pose Estimation for android application uploaded by International Journal of Innovative Science and Research Technology Pune- 411041, India Issue 9, September 2022.

[3] Yoga Pose Perfection using Deep Learning: An Algorithm to Estimate the Error in Yogic Poses. Journal of Student Research, 10(3), 2021.

[4] Yoga-Guru Real-Time yoga pose correction system using deep learning methods 2021 International Conference on Communication Information and Computing Technology (ICCICT), June 2021.

[5] Real-time Yoga recognition using deep learning Neural Computing and Applications volume 31, pages 9349–9361 (2020).

[6] Development of a yoga posture coaching system using an interactive display based on transfer learning © The Author(s), under exclusive license to Springer Science+Business Media, LLC, part of Springer Nature 2022.

[7] Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, March 2021 Journal of Big Data 8(1) DOI: 10.1186/s40537-021-00444-8.

[8] Yoga Pose Estimation and Feedback Generation Using Deep Learning, Computational Intelligence and Neuroscience Volume 2022 Publisher-Hindawi Limited London, United Kingdom, 2022.

[9] Agrawal, Y., Shah, Y., & Sharma, A. (2020). Implementation of Machine Learning Technique for Identification of Yoga Poses. 2023 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT). doi:10.1109/csnt48778.2020.911

[10] Development of a yoga posture coaching system using an interactive display based on transfer learning © The Author(s), under exclusive license to Springer Science+Business Media, LLC, part of Springer Nature 2021.

[11] Chhaihuoy Long, Eunhye Jo, and Yunyoung Nam, (2022), "Development of a yoga posture coaching system using an interactive display based on transfer learning" - J Supercomputer 78, 5269–5284

[12] Nagalakshmi Vallabhaneni and Dr. P. Prabhavathy, (2021), "The Analysis of the Impact of Yoga on Healthcare and Conventional Strategies for Human Pose Recognition", Turkish Journal of Computer and Mathematics Education(TURCOMAT).,vol. 12 no. 6, pp. 1772-1783 DOI: 10.17762/turcomat.vl2i

[13] Wu, Y., Lin et al.(2021). "A Computer Vision-Based Yoga Pose Grading Approach Using Contrastive Skeleton Feature Representations." Healthcare (Basel), 10(1):36

[14] Deepak Kumar, and Anurag Sinha, (2020), "Yoga Pose Detection and Classification Using Deep Learning" International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Volume 6 Issue 6 Page: 160-184

[15] Girija Gireesh Chiddarwar,et al. (2020), "AI-Based Yoga Pose Estimation for Android Application" International Journal of Innovative Science and Research Technology, Volume 5 - Issue 9

[16] Camillo Lugaresi, et al. (2019), "MediaPipe: A Framework for Perceiving and Processing Reality", Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR).