# VIRTUAL YOGA TEACHER WITH AI MODEL FOR YOGA POSTURE CORRECTION FOR GOOD HEALTH

*Swapna Yenishetti[1]; Dr.Ganesh Karajkhede[2;] Lakshmi Panat [3]*

[1,2,3]Centre For Development of Advanced Computing

## ABSTRACT

*With the advent of preventive and promotive health in the world, driven by the WHO's Sustainable Development Goal for Healthcare to ensure healthy lives and promote well-being for all ages it is a necessity to develop Artificial Intelligence driven tools for Yoga. Yoga is the art and science of healthy living and has a history spanning thousands of years. Hence there is genuine need of AI model for Yoga posture correction, to reap the benefits of practice of Yoga. Developing AI models for yoga posture correction has the potential to enhance the experience for yoga followers and acting as a Virtual Yoga teacher. In this study, we are focusing on identification of 10 important yoga poses performed by the user with the help of Pre - trained Move Net model and transfer learning. Using such deep learning algorithms, an individual can understand the gap between current and ideal way of performing the specific yoga asana, thus enabling correction in the yoga pose. This process is to be completed in real-time and needs to be interactive necessitating the usage of react.js library. The system analyzes the difference between the actual and ideal yoga pose and landmarks the image of human body while performing the yoga pose with required correction.*

**Keywords** – MoveNet, Transfer learning, Yoga Posture Detection, Tensorflow Object Detection API, Deep Neural Network

## 1. INTRODUCTION

The history of yoga is spanning thousands of years and evolving through various cultural, religious, and philosophical contexts. Archaeological evidence suggests that yoga practices may have existed as far back as the Indus Valley Civilization (around 3300–1300 BCE). It is one of the most appreciated scientific contribution of India to world.

The global yoga industry has experienced significant growth in recent years, driven by increasing awareness of health-wellness and popularity of mind-body practices. According to market research, the yoga market was valued at over $37 billion in 2020 and is projected to continue growing in the coming years. The global yoga market size is forecast to reach $66.2 billion by 2027 and is expected to have a compound annual growth rate of 9.6% from 2021 to 2027.

Practicing yoga in wrong way can lead to various negative effects on the body. It may include muscle injury, joint pain, back pain, nerve compression, pressure on internal organs leading to injury, breathing problems, psychological effects etc. To avoid these negative effects, it's crucial to practice yoga under the guidance of a qualified instructor. For beginner or attempting advanced poses focusing on proper alignment and breathing for a safe and effective yoga practice.

Therefore, a need for a 'Yoga Pose Detection' algorithm arises, that can check and correct yoga poses, while also eliminating the need of human instructors. Taking inspiration from the psychology of transfer of learning, Transfer Learning is a section of Machine Learning that focuses on gaining knowledge from one problem, and using the knowledge in another similar problem. This has proven to be a significant milestone in Reinforcement Learning. In Yoga, since we don't have to deal with motion, Yoga poses have been proven to be an excellent example of image classification. There are different pre-trained models that can be used for Image Classification based on Transfer Learning. The first approach used in this project was ResNet34. Then other approaches such as VGG, EfficientNet, Image Net were also explored. Although each of these approaches had their individual benefits, they also came with their own drawbacks. After evaluation and comparison of different pre-trained models, MoveNet was finalized and used in the project. Similarly we have prepared and finalized dataset. After exploring existing datasets of images, and benefits of the poses, we prepared a custom data set consisting of following ten poses: Bhujangasana, Garudasana, Halasana, Natarajasana, Sukhasana, Makarasana, Savasana, Simhasana, Tadasana. We intend to broaden the data set in the near future. By integrating following features, the application aims to be a catalyst for positive environmental, social, and economic change, empowering individuals to embrace sustainable living practices in their everyday lives.

1. The user interface is designed to be intuitive, engaging, and visually appealing, ensuring ease of navigation and accessibility for all users.
2. Features clear and concise information on sustainable practices, making it easy for users to adopt and incorporate these practices into their daily routines.
3. The application leverages data analytics to provide personalized recommendations and insights based on user behavior and preferences.
4. Utilizes data-driven approaches to track and measure the user's environmental and social impact, providing feedback and rewards for positive contributions.
5. Offers a wealth of educational resources such as articles, videos, and interactive modules to raise awareness and educate users about sustainable yoga practices.

## 2. DATASETS

Yoga-82 is a new Dataset for Fine-grained Classification of Human Poses. It is one of the largest data set which is used to train the yoga classification techniques. The data set contains a three-level hierarchy including body positions, variations in body positions, and the actual pose names. We present the classification accuracy of the state-of-the-art convolutional neural network architectures on Yoga-82. We also present several hierarchical variants of Dense Net in order to utilize the hierarchical labels. On similar lines, we have also created custom data set to train the model for most common poses used by the public to demonstrate the capability of the model.

## 3. LOGIC

Human pose estimation is a Computer Vision technique used to predict a person's body parts or joints position. This can be done by defining the human body joints like wrist, shoulder, knees, eyes, ears, ankles, arms, also called key points in images and videos. Then, when a picture or video comes in as input to the pose estimator model, it identifies the coordinates of those detected body parts as output and a confidence score indicating continuity of the estimations. The MoveNet model is based on 3D estimation. The operation takes place in a phase-wise manner like; first, the RGB image is fed to the convolutional network as input, then the pose model is applied to detect the poses, key points, pose confidence score and key point confidence score from the model outputs.

## 4. LANDMARKS

The landmarks of a human body used in this project are nose, ears, eyes, shoulders, elbows, wrists, hips, knees, and ankles. This gives us 17 landmarks, which will be used to detect various poses by the model.

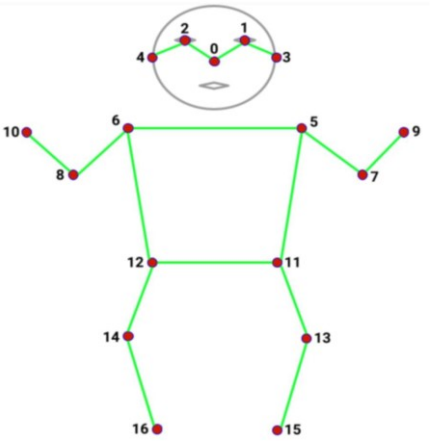



**Figure 1– Landmarks**

## 5. OBJECTIVE AND METHODOLOGY

**Objective**: Development of DNN based detection model using transfer learning technique to detect the yoga pose, capture landmarks on real-time image or video to detect the posture and integrate it in application to alert user for correctness. The application shall also have encyclopedia on yogasana's with relevant information.

**Methodology**: For examination of different yoga poses, pre-trained model MoveNet is used, which detects 17 landmarks on each images and creates csv file and prepared train and test data. These manually created csv files are fed to train the model for classification of poses. We have developed react.js front-end application and used tensorflow.js for importing the model. At first it shall detect the pose and capture 17 landmarks from input image or video for comparison with master landmarks. If current posture reaches the accuracy of threefold above 70% probability then, the colour turns to the green and also it counts the time for holding the posture correctly and notifies us by playing sound.

## 6. LITERATURE SURVEY

Due to the increase of stress in the modern lifestyle, yoga has become popular throughout the world. Although considered to be a great way to bring physical, mental, and spiritual harmony, it can come with its own set of problems, if not done properly. This gave rise to the need for Yoga Pose Detection and Classification, to omit the necessity of a human instructor. But, for an AI agent to behave like a human, it is necessary to have a strong model to detect and classify various yoga poses. OpenPose was the pioneer project in the field of Landmark detection using Image Processing. Using 18 landmarks detected on the human body, it was able to achieve 78% accuracy in detecting the yoga poses. PifPaf uses a Part Intensity Field (PIF) and a Part Association Field (PAF) for body part limitation, and relationship of body parts to shape full human stances, respectively. This technique is dependent on the base up

approach for 2-D multi-individual human posture assessment. This could not achieve accuracy beyond 76.4%. Convolutional Neural Networks did achieve an accuracy beyond 80%. It achieved an accuracy of 82.84%, by using various layers of neurons along with techniques like Batch Normalization, and Dropouts. This accuracy did not prove to be sufficient for effective training of the agent. But, it did become the base of many state-of-the-art learning techniques. EfficientNet is the first technique to use CNN along with the Scaling method. EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. It also transfers well to other problems, and is shown to achieve accuracy on CIFAR-100 (91.7%) and Flowers (98.8%), among other datasets. But, when this learning was used on Yoga Poses, it showed an accuracy of 85%. Our next approach was to explore Deep Residual Learning. Since deeper neural networks are more difficult to train, residual learning framework makes it easier to train the model. This makes the network easier to optimize, while also gaining accuracy, because of the increased number of hidden layers. This approach provides an accuracy of 70% on Yoga Dataset.

| **PoseNet** | **MoveNet** |
|---|---|
| PoseNet is profound learning structure that detects human postures by distinguishing joint areas in a human body. But, even with this structure, accuracy kept fluctuating between 0.5 and 0.9, depending on the pose | Using MoveNet, the model detects 17 landmarks on the image of the human body. After detecting the landmarks, the model estimates the pose using the distance of each landmark from the centre point of the image. An average of each individual landmark score is taken, and considered to be the score of that image |
| PoseNet is an older generation pose estimation model released in 2017. It is trained on a standard COCO dataset and provides a single pose and multiple pose estimation variants. | MoveNet is the latest generation pose estimation model released in 2021.which is an ultra-fast and accurate model that detects 17 key-points of a body. The model is offered on TF Hub with two variants, known as Lightning and Thunder. Lightning is intended for latency-critical applications, while Thunder is intended for applications that require high accuracy. |
| The single pose variant can detect only one person in an image/video and the multi pose variant can detect multiple persons in an image/video. Both variants have their own set of parameters and methodology. Single pose estimation is simpler and faster but required to have a single person in an image/video otherwise key points from multiple persons will likely be estimated as being part of a single subject | Both variants run faster than real time (30+ FPS) on most modern desktops, laptops, and phones, which proves crucial for live fitness, health, and wellness applications. MoveNet is a bottom-up estimation model, using heat-maps to accurately localize human key-points. |
| PoseNet again has two variants in terms of model architecture that is MobileNet v1 architecture and ResNet50 architecture. The MobileNetV1 architecture model is smaller and faster but has lower accuracy. The ResNet50 variant is larger and slower but it's more accurate. Both MobileNetV1 and ResNet50 variants support single pose and multi-person pose estimation. The model returns the coordinates of the 17 key points along with a confidence score. | This is a project of TensorFlow, which provides two variants, Thunder and Lightning. Since our project requires high accuracy, we used Thunder in this project. For latency-critical applications, Lightning is considered to be a better option. Using this model, the model is capable of achieving an accuracy between 0.87 and 0.90. |

**Table 1– PoseNet and MoveNet**

## 7. METHODOLOGY/TECHNIQUES

MoveNet architecture consists of two components: a feature extractor and a set of prediction heads. The prediction scheme loosely follows CenterNet, with notable changes that improve both speed and accuracy. All models are trained using the TensorFlow Object Detection API.

The feature extractor in MoveNet is MobileNetV2 with an attached feature pyramid network (FPN), which allows for a high resolution (output stride 4), semantically rich feature map output. There are four prediction heads attached to the feature extractor, responsible for densely predicting a:

**Person center heat-map**: predicts the geometric center of person

**Landmark regression field**: predicts full set of key-points for a person, used for grouping key-points into instances
**Person regression field**: predicts the location of all key-points, independent of person instances

**2D per-key point offset field**: predicts local offsets from each output feature map pixel to the precise sub-pixel location of each key point

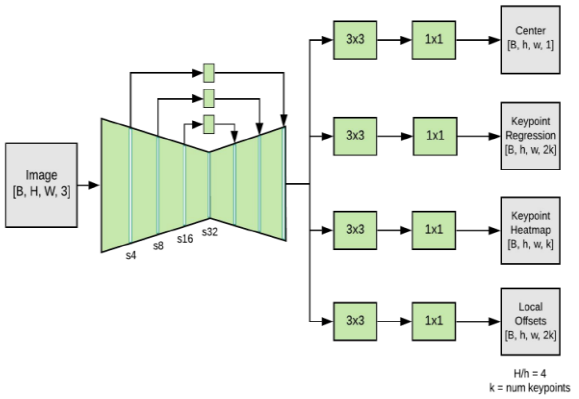



**Figure 2– MoveNet Architecture**

Although these predictions are computed in parallel, one can gain insight into the model's operation by considering the following sequence of operations:

**Step 1**: The person center heat-map is used to identify the centers of all individuals in the frame, defined as the arithmetic mean of all key-points belonging to a person. The location with the highest score (weighted by the inverse-distance from the frame center) is selected.

**Step 2**: An initial set of key-points for the person is produced by slicing the key point regression output from the pixel corresponding to the object center.

**Step 3**: Each pixel in the key point heat-map is multiplied by a weight which is inversely proportional to the distance from the corresponding regressed key point. This ensures that we do not accept key-points from background people, since they typically will not be in the proximity of regressed key-points, and hence will have low resulting scores.

**Step 4**: The final set of key point predictions are selected by retrieving the coordinates of the maximum heat-map values in each key point channel. The local 2D offset predictions are then added to these coordinates to give refined estimates. See the figure below which illustrates these four steps. Since this is a center-out prediction – which must operate over different scales

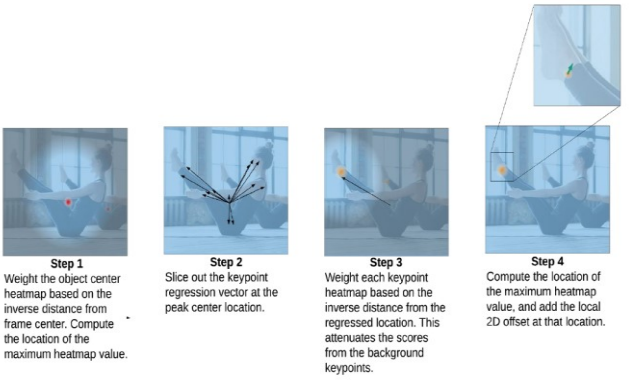



**Figure 3– sequence of MoveNet Model Operations**

## 8. DATASET PREPARATION

We started collecting various images of different Yoga Poses from the Internet. After collecting a sufficient amount of images, we categorized data into 10 different Yoga Poses. For the sake of ease, we decided to manually split our data into training and testing datasets. We classified the collected images into sub-folders based on Poses. Then, keeping in mind a ratio of 3:7 between testing and training datasets, we split the entire data into two parts. Each part have sub-folders corresponding to each data set.

## 9. DATA AUGMENTATION

Data augmentation is a set of techniques to artificially increase the amount of data by generating new data points from existing data. This includes making small changes to data or using deep learning models to generate new data points. Data augmentation is useful to improve performance and outcomes of machine learning models by forming new and different examples to train datasets. If the data set in a machine learning model is rich and sufficient, the model performs better and more accurately.

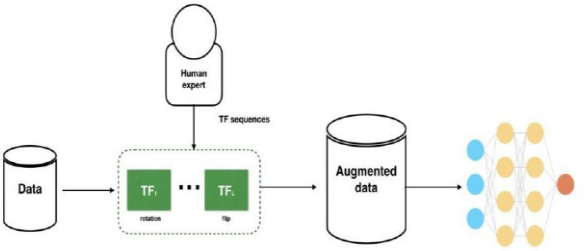



**Figure 4– Data Augmentation**

In this project, we mainly focused on cropping the images such that the landmarks necessary for pose detection are easy to visualize and analyze. We used a separate class known as Pre-processor for Data Augmentation and imported MoveNet pre-trained model for predicting the landmarks. The landmarks of the images are written to a

CSV file for each individual image. Each CSV file is then combined into a single file, for each folder (pose) given. A function named 'class_names' is used to return the names of the classes (poses). A final function named 'all_landmarks_as_dataframe' is used to combine all CSV files corresponding to poses, into a single data frame. The training data set CSV is split into Validation and Training datasets, with the ratio of 3:17, to train the model.

```
Preprocess the TRAIN dataset

[ ]  1 images_in_train_folder = "/content/yoga10_new/yoga_poses/train/"
     2 images_out_train_folder = 'poses_images_out_train'
     3 csvs_out_train_path = 'train_data.csv'
     4
     5 preprocessor = MoveNet_Preprocessor(
     6     images_in_folder=images_in_train_folder,
     7     images_out_folder=images_out_train_folder,
     8     csvs_out_path=csvs_out_train_path,
     9 )
    10
    11 preprocessor.preprocess(per_pose_class_limit=None)
```

**Figure 5– Pre-process code snippet**

## 10. MODEL DESCRIPTION

After data augmentation in CSV format, need to convert the data into tensors for the model. We first computed the various centre points relevant to the pose, using a function 'get_center_point'. Our next step was to get a normalized pose size. This is achieved by calculating distances of landmarks from various centre points, and then taking an average. Finally, the average is normalized to determine the pose size. The landmarks are normalized depending on the pose size. Since each landmark is denoted by coordinates and score, the Input size initially is 51 (17 * 3) for our model. We embed the input size, by expanding first, normalizing landmarks based on coordinates, and then flatten it to a size of 34. Due to this, our final input size for the model is 34 (17 * 2). We introduced 3 hidden layers to our model. Our first hidden layer consists of 128 neurons. In the second and third layer, we go on reducing the number of neurons by half the size of the previous layer. Thus the second and third layers have 64 and 32 neurons, respectively. At each layer, we are using ReLU as an activation function. Along with the layers, we have added Batch Normalization after each hidden layer. This is done with the intention of normalizing the outputs of each hidden layer, and tackling the problem of Vanishing Gradients. This approach ensures that we can avoid the case of over-fitting for various Yoga Pose Datasets

```
[ ]  1 # Define the model
     2 inputs = tf.keras.Input(shape=(51))
     3 embedding = landmarks_to_embedding(inputs)
     4
     5 layer = keras.layers.Dense(128, activation=tf.nn.relu6)(embedding)
     6 layer = keras.layers.BatchNormalization()(layer)
     7 layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
     8 layer = keras.layers.BatchNormalization()(layer)
     9 layer = keras.layers.Dense(32, activation=tf.nn.relu6)(layer)
    10 layer = keras.layers.BatchNormalization()(layer)
    11 outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)
    12
    13 model = keras.Model(inputs, outputs)
    14 model.summary()
```

**Figure 6– Model Description**

## 11. CONFUSION MATRIX AND RESULT VISUALIZATION

After training and compilation, this model gives a validation accuracy of 0.87. The same model, when fit on the Testing data set, gives an accuracy of 0.86. We also calculated and plotted a Confusion Matrix to visualize the correct and incorrect predictions in the Testing Dataset.

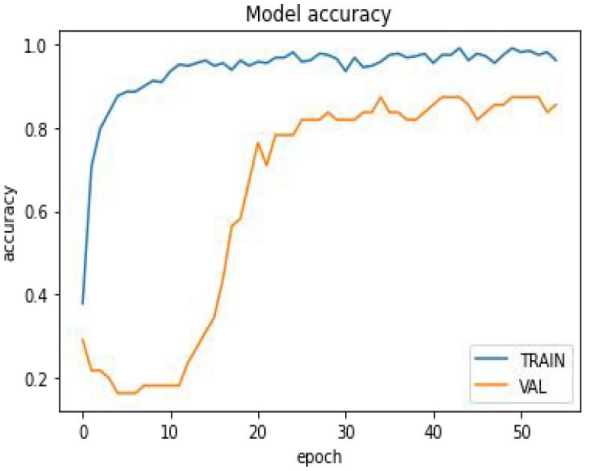



**Figure 7– Accuracy**

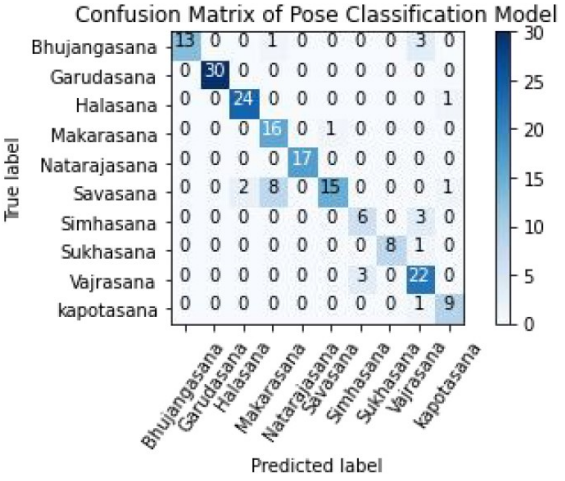



**Figure 6– Confusion Matrix**

As an extra measure to visualize the correct and incorrect predictions, we added separate codes to investigate the correct and incorrect predictions. The separation is done by comparing y_true and y_predicted labels. For the images, where the labels don't match, they are shown as incorrect predictions. The remaining images are shown as correct predictions.

## 12. TENSORFLOW.JS IMPLEMENTATION

### 12.1 Requirements:

The conversion procedure requires a Python environment, we need to keep an isolated one using pipenv or virtualenv.

Importing a Keras model into TensorFlow.js is a two-step process. First, convert an existing Keras model to TF.js Layers format, and then load it into TensorFlow.js.

*12.1.1* Step 1:

Convert an existing Keras model to TF.js Layers format. Use the Python API to export directly to TF.js Layers format. As we have a Keras trained model in Python, we are exporting it directly to theTensorFlow.js Layers format as follows:

```python
# Python

import tensorflowjs as tfjs

def train(...):
    model = keras.models.Sequential()   # for example
    ...
    model.compile(...)
    model.fit(...)
    tfjs.converters.save_keras_model(model, tfjs_target_dir)
```

**Figure 7– Step-1**

12.1.2 Step 2:

Load the model into TensorFlow.js by providing the URL to the model.json file:

```javascript
// JavaScript

import * as tf from '@tensorflow/tfjs';

const model = await tf.loadLayersModel('https://foo.bar/tfjs_artifacts/model.json');
```

**Figure 8– Step-2**

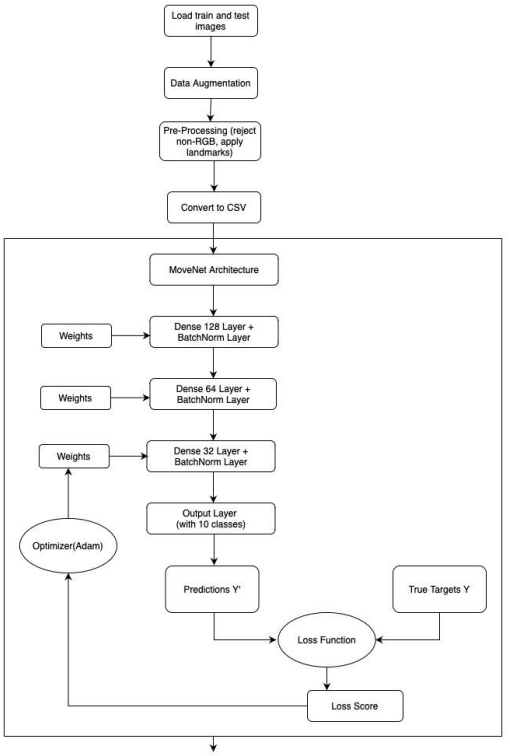
Now the model is ready for inference, evaluation, or re-training. For instance, the loaded model can be immediately used to make a prediction: We have taken this approach, so that pre-trained model can easily be hosted on any Cloud Storage Platform.

```javascript
// JavaScript

const example = tf.fromPixels(webcamElement);  // for example
const prediction = model.predict(example);
```

This approach allows all of these files to be cached by the browser (and perhaps by additional caching servers on the internet), because the model.json and the weight shards are each smaller than the typical cache file size limit. Thus a model is likely to load more quickly on subsequent occasions.


```javascript
// JavaScript

const example = tf.fromPixels(webcamElement);  // for example
const prediction = model.predict(example);
```


**Figure 9– Use Model**



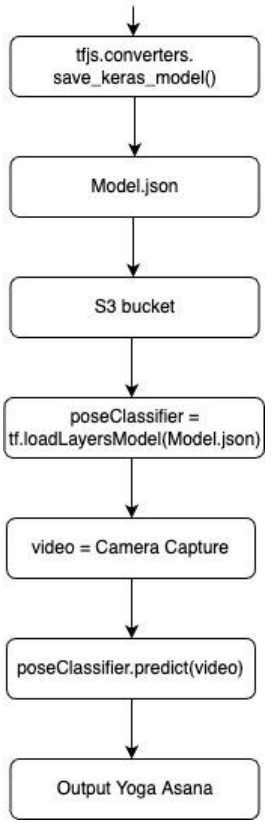



**Figure 10– Flow Chart of Application**

## 13. RESULTS & CONCLUSION

As it can be seen in the screenshots below, at first our model is detecting all the 17 landmarks on the body and draw connections between them in white colour. Further, if our current posture reaches the accuracy of threefold above 70% probability then, the colour turns to the green and also it counts the time for holding the posture correctly and notifies us by playing sound.

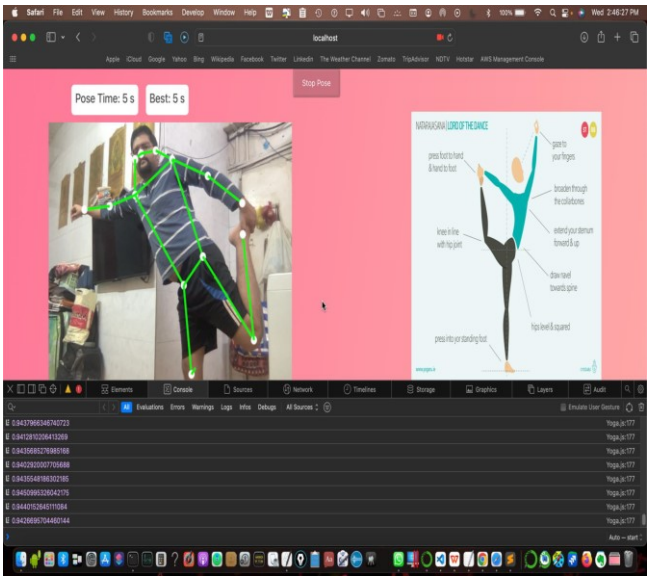



**Figure 11– Screenshot of Application**

We have exploited these different transfer learning models (TL-VGG16, TLMobileNetV2, TL-InceptionV3, and TL-Resnet50, TLInception-ResnetV2, TLEfficientNetB0, TL-MoveNet Thunder) as shown in figure12. As we can see the accuracies all other models are lower than Movenet Thunder for classification task. Hence we can conclude that the MoveNet Thunder is optimal model for the yoga correcting system, based on evaluation metrics. As a result, the TL-MoveNet model was selected as the optimal model, showing validation accuracy of 87%, precision 0.87, recall of 0.86, and validation loss of 0.4958.

| SR. NO. | TECHNOLOGY EMPLOYED | VAL ACCURACY |
|---|---|---|
| 1 | PyTorch | 59% |
| 2 | TensorFlow without Pre-trained model | 56% |
| 3 | VGG16 | 78% |
| 4 | ResNet50 | 48% |
| 5 | InceptionV3 | 70% |
| 6 | Inception-ResnetV2 | 73% |
| 7 | MobileNetV2 | 81% |
| 8 | EfficientNetV2 B3 | 85% |
| 9 | MoveNet Thunder (Our Current Project) | 87% |

**Figure 12–Transfer learning models with accuracy**

Possible benefits of developing AI model for yoga posture correction are

1. Real-time Feedback: With an AI model, practitioners can receive immediate feedback on their posture as they perform yoga poses. This real-time feedback allows them to make adjustments and corrections on the spot, leading to more effective and safer practice sessions.
2. Personalized Feedback: While attending a yoga class, it can be challenging for instructors to provide individualized feedback to each student due to time constraints and class size. AI model could offer personalized feedback tailored to each practitioner's unique needs and abilities.
3. Continuous Improvement: AI models can be trained on large datasets of yoga poses, allowing them to learn from a wide range of examples and continuously improve over time. This means that the accuracy and effectiveness of the posture correction AI can increase as more data is collected and the model is refined.
4. Supplementary Learning Tool: Even for individuals attending yoga classes with instructors, AI model can serve as a supplementary learning tool, providing additional support and guidance outside of class hours.
5. Consistency: Human instructors may vary in their teaching styles and levels of expertise, leading to inconsistencies in the feedback provided to students. AI model can provide consistent and objective feedback, ensuring that practitioners receive accurate guidance regardless of who is teaching them.
6. Accessibility: Not everyone has access to a yoga instructor or classes, especially in remote or

underserved areas. AI model could provide guidance and feedback to individuals practicing yoga on their own, making the practice more accessible to a wider range of people.

Overall, AI model for yoga posture correction has the potential to enhance the practice experience for yoga enthusiasts, regardless of their level of experience or access to traditional instruction.This application serves as a holistic platform for individuals striving towards sustainable living, providing the tools, resources, and community support needed to make impactful and lasting changes.

## 14. FUTURE DIRECTIONS

Future directions on developing AI model for yoga posture correction are

- **Customizable Routines**: Allow users to create personalized yoga routines based on their fitness levels, goals, and preferences.
- **Adaptive Learning**: Use machine learning to adapt and suggest poses or routines based on user performance and progress.
- **Integration of NLP**: Use natural language processing for voice commands, allowing users to interact hands-free with the virtual instructor
- **Sensor Data**: Integrate data from wearables (like smartwatches) to monitor heart rate, calories burned, and stress levels.
- **Comprehensive Health Dashboard**: Integrate nutrition, sleep, and stress management tools to offer a holistic health overview.
- **Wellness Insights**: Provide insights and recommendations based on user data to enhance overall well-being.
- **Challenges and Rewards**: Introduce gamified elements like challenges, badges, and leaderboards to motivate users.
- **Progress Tracking**: Offer detailed analytics and reports on progress, flexibility improvements, and time spent practicing.
- **Research Emerging Technologies**: Keep up with advances in AI, machine learning, and health tech to incorporate the latest features.
- **Explore VR and AR**: Investigate the potential of virtual reality and augmented reality for immersive yoga experiences.
- **Iterative Development**: Use feedback to continuously improve the application, add features, and fix issues.

## REFERENCES


[1] https://ieeexplore.ieee.org/document/9310832

[2] https://www.researchgate.net/publication/350931097_Infinity_Yoga_Tutor_Yoga_Posture_Detection_and_Correction_System

[3] https://tfhub.dev/google/movenet/singlepose/thunder/4

[4] https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1932&context=etd_projects

[5] https://www.tensorflow.org/js/tutorials/conversion/import_keras

[6] https://keras.io/about/

[7] https://blog.logrocket.com/tensorflow-js-an-intro-and-analysis-with-usecases8e1f9a973183/#:~:text=js%20is%20that%20it%20allows,the%20Python%20version%20of%20TensorFlow.

[8] https://www.tensorflow.org/hub/tutorials/movenet

[9] https://www.freecodecamp.org/news/reactjs-basics-dom-componentsdeclarativeviews/#:~:

text=The%20DOM%20(Document%20Object%20Model,%2C%20attributes%2C%20and%20so%20on.

[10] https://www.jsr.org/hs/index.php/path/article/view/2140

[11] .https://www.npmjs.com/package/web-vitals

[12] .https://www.npmjs.com/package/@tensorflow-models/pose-detection