Yoga Pose Detection Using Deep Learning Techniques

S. Sankara Narayanan[1]* Devendra Kumar Misra[2] Kartik Arora[3] Harsh Rai[4]

[1]* Department of Information Technology, HMR ITM, GGSIPU New Delhi, 2412shankar@gmail.com
[2] Department of Information Technology, HMR ITM, GGSIPU New Delhi, misra.deven@gmail.com
[3] Department of Information Technology, HMR ITM, GGSIPU New Delhi, kartikarora27.ka@gmail.com
[4] Department of Information Technology, HMR ITM, GGSIPU New Delhi, rai.harsh99@outlook.in

***Abstract* – Yoga aims at development of all-round personality, it is synchronization of mind, body and the spirit. Improper postures / in-correct way of doing Yoga can result in serious damage to one's physique and brain. In ancient time, it used to be performed only under the supervision of an accomplished Guru (teacher). In this busy world with time and place constraints, it is difficult to locate an accomplished Guru. Therefore, it is essential to adopt the right Yoga procedure at the very beginning. Deep Learning models can be trained to detect Yoga postures and be able to provide feedback / corrections if needed. OpenPose was the first real-time multi-person system which brought revolutionary change in the field of pose estimation. In this paper using OpenPose we have created a model which estimates a given human pose. We have compared the results obtained by the model for 2D and 3D points of the image and determined if adding more features to the dataset actually increases the accuracy of the model or not. Furthermore, we have put forward a simple neural network model that efficiently analyses the input image and conveys if the pose performed in the image is correct.**

***Keywords:* - Machine Learning, Deep Learning, OpenPose, PyQtGraph, PyOpenGL, PyQt5, 3D Pose Estimation, Yoga Poses**.

I. INTRODUCTION

The importance of Sports and Exercises in human lives is definitely unquestionable. There are plenty of benefits of doing exercises, not only physical but also mental. So, this has greatly attracted various researchers to be committed in this area. With a rapid growth in the field of Computer Vision and Deep Learning [1], people expect models/systems that provides them with the best opportunities for their career growth. Deep Learning is now being used in USA across various major sports league such as NHL, MLB, NBA, NFL and NASCAR in order to expand their business. A few examples where Deep Learning has played a great role in the Sports industry [2] are Wearable Technology, automated Journalism and incorporating Computer Vision. Yoga is one of the exercises which involves complicated postures. Yoga which was developed in ancient India was initially considered as an old age exercise. But because of its many spiritual, physical and mental benefits, Yoga became popular across all age categories. One of the major problems that Yoga and other exercises face is the correct posture in which they must be performed [3]. Even a slight error in the posture may not only nullify the benefits of the exercise but also may result into injuries or lead to structural deformities. So, this puts forward the requirement of an instructor who guides on how the exercise must be done in order to get the maximum from the exercise. But not everyone has the accessibility to an instructor who can supervise and correct the posture whenever the person decides to exercise [4]. Thus, this demands the requirement of a model that will identify the poses and give an output which will fulfill the requirement of a trainer. So, we created a model that not only tells the name of the exercise but also informs the user if he/she is doing the exercise in a correct way or not. This project focuses on the idea of detecting various Yoga poses and solving the problem of improper posture with the help of an accurate model [5].

So firstly, Pre-trained weights were imported from TensorFlow using which 17 body parts such as left elbow, right elbow etc. were defined using OpenPose. Secondly, a model that identified the body parts using live Camera was created, and a skeletal image was formed using the 17 different points whose values were stored. Further, various correct Yoga postures were trained into the model and their values were stored. Finally, the values of each body part captured by the camera were cross-checked with the original values of the Yoga postures that were trained and a result was obtained which gave the name of the posture and if it was done correctly.

II. DETECTION OF HUMAN BODY USING OPENPOSE

It was the OpenPose (the first real-time multi-person system) which brought revolutionary change in the field of pose estimation. Researchers at Carnegie Mellon University proposed their idea of OpenPose, which could be used as a pose estimation method which could gave impressive results. OpenPose uses a CNN – architecture to identify hand, facial and foot key points of human body using camera/image. OpenPose helps the RGB camera in identifying various human body joints. OpenPose key points include points of the human body joints such as Ankles, Knees, Hips, Wrists, Elbows, Shoulders, Eyes, Nose and Ears [6].

OpenPose shows the results obtained from a real-time camera or static images in the form of the above key points. Then all these key points are connected in an order which gives a skeletal image of the human body. Thus, OpenPose is used in a various place ranging from sports, surveillance, activity detection and Yoga Pose detection. OpenPose initially detects the key points of every person in the image and then assigns parts to each distinct individual. Fig 1 shows the architecture of the OpenPose model.

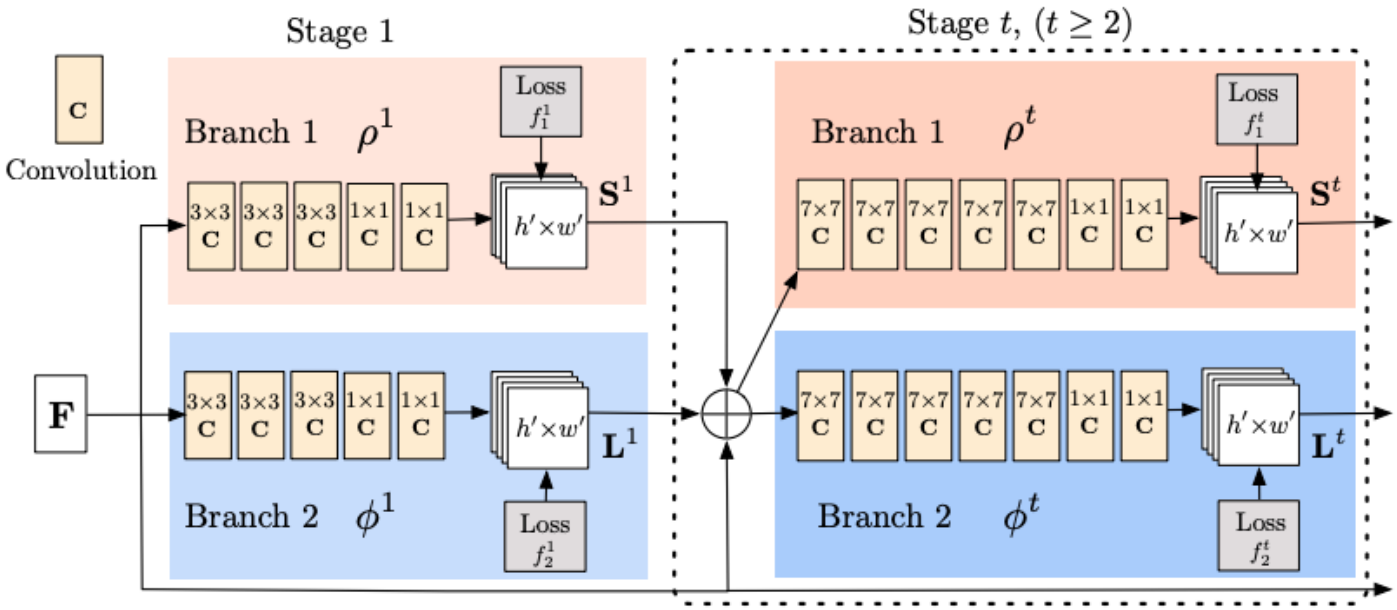

Fig 1. Architecture of OpenPose [7]

Firstly, feature extraction from the image is done using the initial layers of VGG-19 model. Then, these features are advanced on to the two convolutional layers that run parallelly to each other. The set of 18 confidence maps, which represent a particular part of the human pose skeleton are predicted in the 1st branch. Whereas, a set of 38 PAFs (Part Affinity Fields) that indicate the degree of association between parts are predicted in the 2nd branch. Many more stages are used in order to refine the predictions made by each individual branch. Part confidence maps are used to form bipartite graphs between pairs of parts. Then, pruning is done using the PAF values in order to eliminate the weaker links in the above graphs. So, human pose skeletons can be estimated for all the persons in the image using the above steps [8][9].

a) Confidence Maps:

A Confidence Map is a 2D portrayal of the conviction that a specific body part can be situated in some random pixel. Confidence Maps are portrayed by following equation:

$$S = (S_1, S_2, \ldots, S_j) \; where \; Sj \; \epsilon \; \mathrm{R}^{w*h} \quad , j \epsilon 1, , , , J$$

where J = number of body parts locations.

The confidence map for each person k and each body part j is defined by:

$$S^*_{j,k}(p) = \exp(-\frac{\left|\left|p - x_{j,k}\right|\right|_2^2}{\sigma^2}$$

b) Part Affinity Fields:

Part Affinity is a bunch of 2D vector handle that encodes area and direction of appendages of various individuals in the image. It encodes the information as pairwise associations between body parts.

$$L = (L_1, L_2 \ldots, L_c) where \; L_c \boldsymbol{\epsilon} \; \boldsymbol{R^{w*h*c}}, \boldsymbol{c \epsilon 1} \ldots . \boldsymbol{C}$$

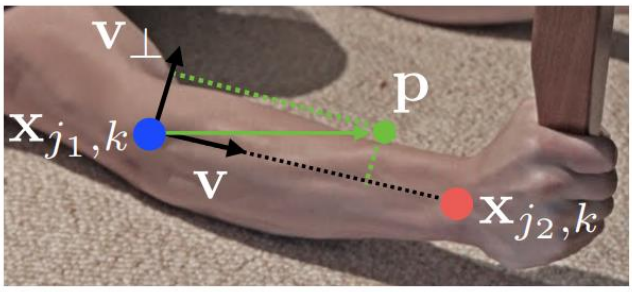

Fig 2. Part Affinity Fields [7][8]

- The L* unit is usually 0 but if the point p is situated on the limb it becomes the unit vector.

$$L^*_{c,k}(p) = \begin{cases} v & if \; p \; on \; limb \; c,k \\ 0 & otherwise \end{cases}$$

- The predicted Part affinity field, Lc along the line segment is to quantify the confidence for two candidate part locations dj1 and dj2:

$$E = \int_{u=0}^{u=1} L_c(p(u)).\frac{d_{j_2}-d_{j_1}}{\left\|d_{j_2}-d_{j_1}\right\|_2} du,$$

- We need to maximize the Total E for multi person:

$$\max_{z_c} E_c = \max_{z_c} \sum_{m\epsilon D_{j_1}} \sum_{n\epsilon D_{j_2}} E_{mn}.z_{j_1 j_2}^{mn}$$

c) Loss Function:

The loss between the predicted confidence maps and Part Affinity fields to the ground truth maps and fields is calculated using the L2-loss function.

$$f_L^t = \sum_{c=1}^{C} \sum_p W(p) \cdot \left\|L_c^t(p) - L_c^*(p)\right\|_2^2 ,$$

$$f_S^t = \sum_{j=1}^{J} \sum_p W(p) \cdot \left\|S_j^t(p) - S_j^*(p)\right\|_2^2 ,$$

Where, Lc* = the ground truth part affinity fields,
Sj*= the ground truth part confidence map,
W = binary mask with W(p) = 0.

$$f = \sum_{t=1}^{T_p} f_L^t + \sum_{t=T_p+1}^{T_p+T_c} f_S^t$$

III. PLOTTING OF THE SKELETON IMAGE

As we probably are aware, the strength of Python is in exploratory Data Science and Visualization utilizing different devices, for example, Pandas, NumPy, Sklearn for its information examination and Matplotlib for plotting. Thus, to approach all these Python devices straightforwardly from inside the application, PyQt is utilized. PyQt permits us to construct complex information driven applications and intelligent dashboards. Despite the fact that it is possible to insert the matplotlib plots in PyQt, the experience doesn't feel native. Along these lines, instead of matplotlib, PyQt is utilized for straightforward and profoundly intuitive plots. PyQtGraph is based on top of PyQ5 local QGraphicsScene giving better drawing execution, especially for live information. It additionally gives incredible intelligence and the capacity to effortlessly alter plots with Qt illustrations gadgets [10].

PyQtGraph is an illustrations and UI library for Python which gives the different usefulness usually needed in planning and science applications. Its essential objective is to give quick, intelligent illustrations for showing information (plots, video, and so forth) [11].

Most of the OpenGL API is covered by a large python pckage called PyOpenGL which presents a more Pythonic interface, but it won't save from having to learn the details of OpenGL. It abstracts the API called glColor in place of glColor3b, glColor3d, glColor3f, glColor3i, glColor3s, etc [12].
So, packages such as PyQtGraph and PyOpenGL are used to plot the skeletal image of a human body (Fig 2).

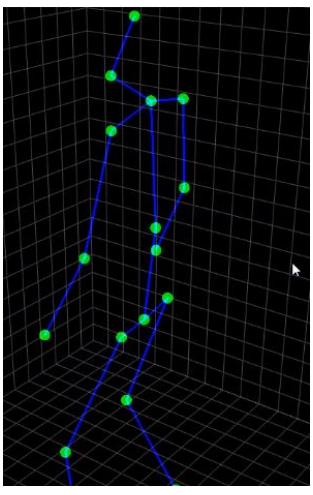

Fig 3. 3D Plot using a real-time camera

IV. DATASET AND DATA PREPROCESSING

Initially, we only had the data in the form of images of different yoga poses.

Fig 4: Yoga Pose

So, in order to train the model, we performed Data Preprocessing and generated data in the form of 3D points which were stored in a CSV file.

Since, we only had images of 3 Yoga Poses. We found out the points of all the images of the 3 Yoga Poses and stored it in a list.

```
data1= []
labels = []
for img in tree:
    file="../training_set/tree/"+img
    print(file)
    data1.append(run(file))
```

```
  [ 44  60  89]
  [ 43  61  90]]

 [[ 90 100 110]
  [ 91 101 111]
  [ 88 100 112]
  ...
  [ 35  49  78]
  [ 33  49  78]
  [ 35  51  80]]] in 0.3709 seconds.
```

Then, the list which was of 4D shape was converted into a np array. So, we converted the array into 2D shape as we only require 2D i.e., size of dataset and the co-ordinates of all the body parts across all the three dimensions.

```
c= np.reshape(c,(94,51))
```

Finally, we convert the np array into a dataframe using the pandas library. and further the dataframe is converted to a CSV file.

```
df=pd.DataFrame(data=c,columns=column_values)
df.to_csv('data.csv')
```

The CSV of 2D points contains the data having 34 points i.e., 17 body points for 2 different dimensions (X-Y Axis).

X_RKnee	X_RAnkle	...	Y_RKnee	Y_RAnkle

The CSV of 3D points was further created which contained the data having 51 points i.e., 17 body points for 3 different dimensions (XYZ- Axis).

X_RHip	X_RKnee	X_RAnkle	...	Z_RHip	Z_RKnee	Z_RAnkle

This means that the dataset of 3D points has more features as compared to the 2D points which describes each image in a more accurate manner.

- **STANDARIDIZATION SCALING:**

Standardization replaces the values by their Z-scores, it brings all of the data into a standard normal distribution which has mean (μ) zero and standard deviation one (σ) [13].

$$Standardisation: x = \frac{x - mean(x)}{sd(x)}$$

Standardization in python is implemented with the help of sklearn.preprocessing package .

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
```

- **CONVERTING THE DATA INTO CATEGORICAL VALUES**

The data in the dataset was converted into categorical values using the Keras API and stored into y_binary array (size of data X number of categories).

```
from keras.utils import to_categorical
y_binary = to_categorical(Y)
```

```
Using TensorFlow backend.
```

The variable y_binary now contains the data in categorical form.

V. NEURAL NETWORK MODEL

A Sequential model is fitting for a simple collection of layers where each layer has precisely one input node and one output node. The successive API permits you to make models layer-by-layer for most issues. On the other hand, the useful API permits you to make models that have significantly greater adaptability as you can undoubtedly characterize models where layers interface with something beyond the past and next layers. A sequential model is easy to work on, so we used sequential model in our Simple Neural Network. The quantity of hidden layers is made dependent on the accessible dataset. As per the dataset accessible to us, we made a model that comprised of 2 Hidden Layers and 1 Output Layer. Further, you need to check as indicated by the dataset, of what initiation work and what enhancers are to be utilized.

- **RELU ACTIVATION FUNCTION**

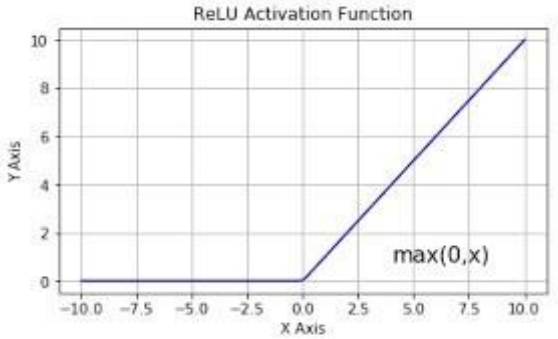

Fig 5: ReLu Activation Function

The Rectified straight unit (ReLu) initiation work has found to be the most broadly utilized activation function for Deep Learning applications with cutting edge results. It as a rule accomplishes better execution and speculation in Deep Learning contrasted with the

sigmoid activation function. The primary thought behind the ReLu actuation work is to play out an edge activity to every input component where the non-positive values are set to zero.

$$f(x) = \max(0, x) = \begin{cases} x_i & if\ x_i > 0 \\ 0 & if\ x_i < 0 \end{cases}$$

- **SOFTMAX ACTIVATION FUNCTION**

A target class having the highest probability is obtained by using the SoftMax function for expectation in multi-class models in which probabilities of each class in gathering of various classes is returned. The determined probabilities are then useful in deciding the objective class for the given data sources. The SoftMax function delivers an output which is a scope of qualities somewhere in the range of 0 and 1, with the amount of the probabilities been equivalent to 1.
The SoftMax function is calculated using the equation:

$$f(x) = \frac{\exp(x_i)}{\sum_j \exp(x_i)}$$

In our model, we utilized the ReLu initiation function for the hidden layers and the SoftMax Activation function for our Output Layer as they gave a superior train precision and great validation score. Finally, while compiling the model, we used the Adam Optimization algorithm as it is considered to be the best for training a model in less time and in an efficient manner. If you have a sparse data use an optimizer that have a dynamic learning rate.

- **ADAM OPTIMIZATION ALGORITHM**

Adam (Adaptive Moment Estimation) works with momentums of first and second order. The instinct behind the Adam is that we would prefer not to roll so quick since we can hop over the base, we need to diminish the velocity a tad for a cautious pursuit. Adam keeps an exponentially decaying average of past gradients **M(t)** in addition to storing an exponentially decaying average of past squared gradients like **AdaDelta**.

M(t) the first moment which is the *Mean* and **V(t)** is the *uncentered variance* of the gradients. [14].

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

The value for β1 is 0.9, 0.999 for β2, and (10 x exp (-8)) for '**ϵ**'.
So, all the decision of what activation function, optimizer, the number of hidden layers depends on the available dataset.

VI. COMPARISON BETWEEN ACCURACY OF 2D AND 3D POINTS

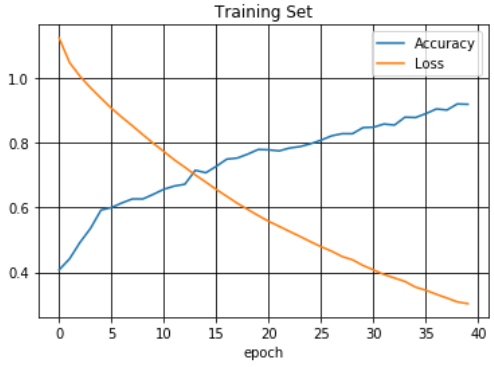

Fig 6: Training set Accuracy and Loss of the Model using 2D points

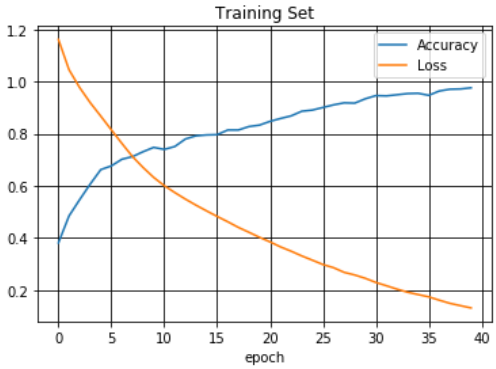

Fig 7: Training set Accuracy and Loss of the Model using 3D Points

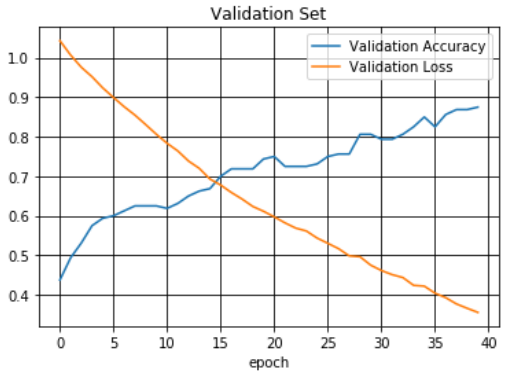

Fig 8: Validation set Accuracy and Loss of the Model using 2D points

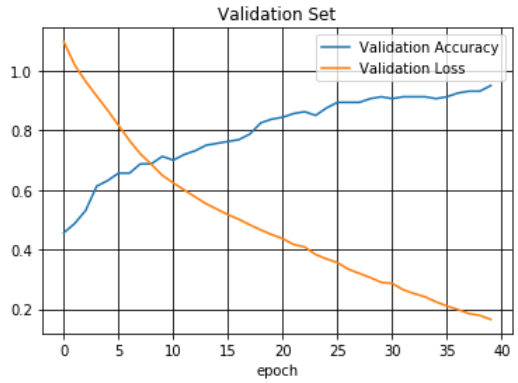

Fig 9: Validation set Accuracy and Loss of the Model using 3D points

From Fig 6, Fig 7, Fig 8 and Fig 9 it can be observed that there is significant improvement in the accuracy of the Model. So, it can be determined that the accuracy of a model can be improved by adding more features to the data.

VII. EVALUATION AND RESULTS

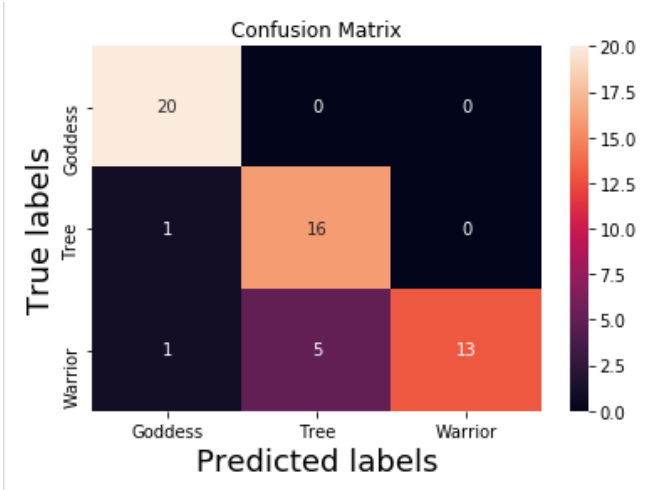

Fig 10: Confusion Matrix using 2D points

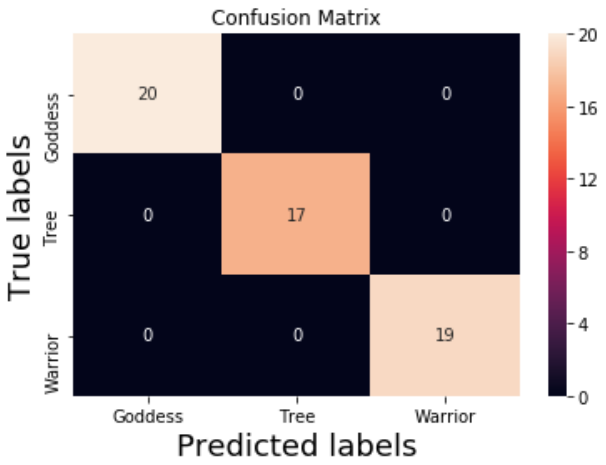

Fig 11: Confusion Matrix using 3D points

From Fig 10 and Fig 11, it can be seen that when 2D points data is used to train the model, it predicted with an accuracy of 85.8%. Whereas, when trained using the 3D points data, the model predicted with an impeccable accuracy i.e., 100%. So, it can be seen that addition of more features into the dataset improves the model significantly.

VIII. CONCLUSION

An away from in the field of Human Pose Estimation is yet to be created. Human Pose Estimation is a confounded point in the field of Computer vision as it includes formation of a human skeleton figure based on an all-around characterized structure of the human body. Exercises unquestionably play an important role in human life. But, performing exercises in an improper manner may lead to injuries, which demands proper training and assistance during the course of action. But, expecting an instructor at all times while performing an exercise is a bit too much to ask for. This demands for a Yoga Self Instructing Model which accurately informs the user if the exercise is performed in a correct manner. So, while creating a model for the dataset consisting of Yoga Poses, we observed that addition of more features into the dataset improves the overall accuracy of the model. Also, the use of OpenPose, PyQt and a Neural Network model on the dataset containing the 3D values is seen to be highly effective than the 2D values and classifies all the 3 yoga poses perfectly.

IX. FUTURE WORK

The proposed model currently identifies only 3 yoga asanas. As there are huge number of yoga asanas, it is a huge challenge to create a pose estimation model that accurately identifies all the poses. In case of overlapping between humans or body parts, the OpenPose library will face challenge in detecting accurately. So, a 3D approach was made which has a scope of improvisation. Using this approach,

Human Pose Estimation can be used in various fields such as Sports, Medical Management, security etc. There is still a lot of scope for research in this incredible field of Pose Estimation. In order to improve the model further, depth camera can be used to detect the Human body/ image. The depth camera will be able to identify multiple bodies, which may solve the problem of multi-person pose estimation. Using the above ideas, we will create an AI Trainer, which will act as a substitute for a trainer. This AI trainer will not only recognize the Yoga pose but will also rectify the incorrect posture(if any).

X. REFERENCES

[1] Joshi, Amit, Khosravy, Mahdi, Gupta, Neeraj (Eds.), Machine Learning for Predictive Analysis", Springer Science and Business Media LLC, 2021

[2] Abawajy, J.H., Choo, K.-K.R., Islam, R., Xu, Z., Atiquzzaman, M. (Eds.), International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019", Springer Science and Business Media LLC, 2020

[3] https://medium.com/datadriveninvestor/how-deep-learning-and-ai-is-changing-the-sports-industry-2657759a4cfb.

[4] S. Yadav, A. Singh, A. Gupta, and J. Raheja, “Real-time yoga recognition using deep learning”, Neural Comput. and Appl., May 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04232-7.

[5] Agrawal, Y., Shah, Y., & Sharma, A. (2020). Implementation of Machine Learning Technique for Identification of Yoga Poses.

[6] https://www.kdnuggets.com/2019/06/human-pose-estimation-deep-learning.html.

[7] https://medium.com/analytics-vidhya/understanding-openpose-with-code-reference-part-1-b515ba0bbc73.

[8] Zhe Cao, Student Member, IEEE, Gines Hidalgo, Student Member, IEEE, Tomas Simon, Shih-En Wei, and Yaser Sheikh OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.

[9] https://www.geeksforgeeks.org/openpose-human-pose-estimation-method/.

[10] https://www.learnpyqt.com/tutorials/plotting-pyqtgraph/.

[11] https://www.geeksforgeeks.org/introduction-to-pyqtgraph-module-in-python/.

[12] https://wiki.python.org/moin/PyOpenGL#:~:text=PyOpenGL%20is%20a%20large%20Python,learn%20the%20details%20of%20OpenGL.

[13] https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.

[14] https://medium.com/ai%C2%B3-theory-practice-business/a-beginners-guide-to-numpy-with-sigmoid-relu-and-softmax-activation-functions-25b840a9a272.