Work-out pose estimation using Deeplearning

# ABSTRACT

Pose estimation is a supervised Machine learning task for estimating the human's different types of body poses or classifying and identifying the joints of the human body from images, video clips, or real-time video. To get an incredible output for this purpose, a deep learning model can be the best fit. In this paper, we describe a deep learning model to identify all bone joints of a human body then we use some training sets to train our model to classify the workout pose of a human body.

# Introduction

Regular physical activity has been shown to reduce morbidity and mortality by decreasing heart disease, diabetes, high blood pressure, colon cancer, feelings of depression/anxiety, and weight while building and maintaining healthy bones, muscles, and joints.[1] It's documented that regular physical activity is essential for healthy aging [2][3]. The American College of Sports Medicine (ACSM) and the American Heart Association (AHA) defines protocols, guidelines, and recommendations regarding the exact type and intensity of elderly exercise regimens [2][3][4][5].

Online workout platform-assisted solutions for elderly people. Pose estimation is one of the most interesting machine learning areas since it is used in different fields, including activity recognition, animation, gaming, augmented reality, etc. An online workout platform for older people /patient where the deep learning model classifies their pose and give the accuracy of their pose might be an effective way for their exercise.

Nowadays, pose estimation has achieved a huge gain in performance by using deep learning. Some existing libraries exist like OpenCV, meidaPipe, and TensorFlow to detect human body parts. In this paper, we use the TensorFlow library to find out body part coordinates (17 landmarks) from images. The work has been done before for pose estimation; most of those are in 2D with fewer yoga poses. The problem is to get better accuracy for exercise and different poses because of background or surroundings, visibility, clothing variations, etc. Therefore, for every image or video, we take 17 different coordinates values (X, Y ) and additionally 17 values for the ground truth of each pose.

To make a classifier using deep learning, we first use the TensorFlow lite model to extract the coordinate of the human joint. Afterward, we build a classifier model with six different types of exercise using CNN. The evaluation of this classification system will be done by using classification scores and a confusion matrix. The model makes predictions of six different exercises from images, and we can examine is the prediction is correct or not.

Pose estimation is a supervised machine learning task for estimating the human's different types of body poses or classifying and identifying the joints of the human body from images, video clips, or real-time video. To detect a human's pose automatically from real-time video or image is a difficult task with good accuracy. To build an exercise instructor platform that will help old people meet online and do exercises related to heart disease accurately on their own, we need an artificial model that can properly classify different types of exercises. To get an incredible output for this purpose, a deep learning model can be the best fit. In this paper, we describe a deep learning model to identify all bone joints of a human body, and then I use some training sets to train our model to classify the workout pose of a human body.

Pose estimation is one of the most interesting machine learning areas since it is used in different fields, including activity recognition, animation, gaming, augmented reality, etc. An online workout platform for older people /patient where the deep learning model classifies their pose and give the accuracy of their pose might be an effective way for their exercise. So, for AI trainer application, realistic gaming fields, intern of health and medical system pose estimation is very important and an interesting field for research.

A screenshot of a computer

Description automatically generated with low confidence

**Materials and methods**

First, we need to capture a set of coordinates for each joint and then connect those with the edge to estimate the pose. The model first identifies the body part localization as input and outputs a low-resolution per-pixel heatmap. This heatmap shows the probability of a joint occurring at each spatial location in the image.[6]

There are three different approaches to modeling the human body: Skeleton-based, Contour based, and Volume-based model. We used a Contour type of approach. Model will:

1. Detect the pose of a single person (**3ft ~ 6ft)**

2. Detect the pose of the person who is closest to the image center and ignore the other people who are in the image frame.

3. The model predicts **17 human key points** of the full body.

MoveNet Lighting is smaller, faster, and can run in real-time on browsers and modern smartphones. Therefore, the MoveNet Lighting version has been used for this project to estimate the keypoint of the human body. For this project, we are using MoveNet, which is the state-of-the-art pose estimation model that can detect these 17 key points:

1. Nose
2. Left eye
3. Right eye
4. Left ear
5. Right ear
6. Left shoulder
7. Right shoulder
8. Left elbow
9. Right elbow
10. Left wrist
11. Right wrist
12. Left hip
13. Right hip
14. Left Knee
15. Right knee
16. Left ankle
17. Right ankle

In order to train our model, we detect the body joints of the human body from the image dataset. The moveNet lite model detects the landmark data(x and y) and grounds truth labels into a CSV file. Save this value into a CSV file for six different exercise /yoga classes. Then we convert these values into a feature vector. Next, we use these vector values to train our neural network based on the pose classifier.

Convolutional neural networks consist of multiple layers of artificial neurons and are widely used for image classification. In the case of key points, CNN extracts the feature (x, y) value and the ground truth value from the CSV file. Based on the filter size, The convolutional filter slides to the next set of input. After the convolution, an activation function Rectified Linear Unit (ReLU6) is generally applied to add nonlinearity in the CNN since the real-world data is mostly nonlinear and the convolution operation is linear.[paper] Tanh and sigmoid are other activation functions, but ReLU is mostly used because of its better performance.[reference paper]

The Keras model takes the detected landmark coordinates to predict the pose class.

The loss function used for compiling the model is categorical cross - entropy which is also called softmax loss. This is used as it allows measuring the performance of the output of the densely connected layer with softmax activation. This loss function is used for multi class classification, and as we have multiple yoga pose classes, it makes sense to use categorical cross entropy. Eventually, we use adam optimizer adam optimizer with an initial learning rate of 0.0001 to manage the learning rate. 100 epochs are used to train our model.

 Describe the model you will train or use: inputs and outputs, model architecture (feel free to add a figure), optimizer and learning rate, batch size, number of training epochs, regularization strategies (e.g. dropout, early stopping), evaluation metrics and any other relevant information**that would help a researcher to replicate what you did**. Check your related papers to see how they describe the models they use, for inspiration

**Dataset**

The dataset collected from Kaggle datasets is a publicly available and open-source collection. The datasets consist of a variety of pose images. Our paper mainly focuses on cardiac rehabilitation exercise; therefore, we choose six yoga poses that can fit heart disease patients. The yoga poses are Tadasana (Mountain pose) and Vrikshasana (Tree pose). The total number of the image is 2523.

Images have been taken in indoor and outdoor environments at different angles and distances from the camera. Individual images have been performed with many variations to build a robust pose recognition model. Three different image files are taken to build the dataset, namely, jpg, png, and bmp. The size of the image dataset is 525 MB. The image shows the variation of different ages, people, and gender.

# Data Preprocessing.

To extract key points for pose estimation, Keras real-time multiperson pose estimation is utilized [7, 8]. This pose estimation is run on every video, frames are extracted for every 2 seconds, and pose is calculated for 5 consecutive frames of each video, which results in 350 instances for 70 videos. Every pose outputs an array of 18 key points where every point consists of *x* and *y* coordinates. Figure [5](https://www.hindawi.com/journals/cin/2022/4311350/fig5/) shows key points extracted from a frame by the pose estimation code.

**Figure 5**

Extracted key points from a frame by the pose estimation method [7].

The research work has used 320 instances for training. While detecting poses for a person, many key points are being detected with different confidence levels. Keras pose estimation works in such a way where it includes the first key point detected without taking into consideration confidence intervals. In this paper, a few modifications were done to the Keras pose estimation to consider key points of highest confidence levels. With these *x* and *y* coordinates, the study extracted features like angles between body joints and with the ground so that models will be trained to achieve good accuracy. Utmost priority is given to these instances so that there will be no abnormality data given as input. Figure [6](https://www.hindawi.com/journals/cin/2022/4311350/fig6/) depicts pose estimation on all 6 yoga poses.

**Figure 6**

Every extracted point is treated as a vector-connecting origin. In body points, nose, ears, and eyes features are not considered as they are not important features, and the features whose confidence score is less than 0.3 are also not considered in order to consider the joints that are accurately visible. So, the number of vectors present is 13. In total, the feature set has 12 joints without nose, ears, and eyes. The 12 joints are neck to the right shoulder, right shoulder to the right elbow, right elbow to the right wrist, neck to the left shoulder, left shoulder to the left elbow, left elbow to the left wrist, neck to the right hip, right hip to the right knee, right knee to the right ankle, neck to the left hip, left hip to the left knee, and left knee to the left ankle. From these 13 vectors, 12 joints can be obtained by subtracting vectors. Suppose body point neck and right shoulder are , respectively. Then, their vectors are  for the neck and  for the right shoulder. To get a vector for the joint neck and right shoulder, subtract the neck vector from the shoulder vector, which is  as shown in Figure [7](https://www.hindawi.com/journals/cin/2022/4311350/fig7/). But, −1 should be multiplied with  because origin in images is present at the top left corner, which is different from the bottom left corner. So, the vector for the joint is . In this way, 12 vectors for 12 joints are obtained and the angles they are making with the *x*-axis need to be calculated. Suppose, the angle made by a vector with the *x*-axis is theta, then  for the vector  is . With this method, 12 angles for 12 different vectors for 12 joints are obtained. So, the feature set has 12 columns.

# Model Performance and Results

The model is built-in Anaconda environment 3.9.7 using Python libraries like cv2, TensorFlow - Keras, NumPy, Pandas, and Scikit Learn on a system Ryzen 5 with 8GB RAM.

Figure Model Layer

Table

Description automatically generated

Graphical user interface, table

Description automatically generated

**The training loss assesses the error on the training data of the model. It means how a model fits on the training set.** On the other hand,**validation loss assesses the error on the validation set of the model** where the validation set is a part of the dataset to validate the model's performance.

In terms of training accuracy means how accurate our model can predict on the training dataset. On the other hand, validation accuracy means the performance of our model on new data set or on validation data set. The figure:2 curves show that our model performance on the validation data set is more accurate, almost 99%. The Train accuracy of our model is 0.9527. The validation loss of our model: 0.0646. The validation accuracy is 0.9919, and the Test accuracy is 0.9474. However, the model accuracy curve illustrates an increase in the training accuracy and a decrease in the validation accuracy, which means some underfitting.

Chart

Description automatically generated

Figure: 2

Table

Description automatically generated

Figure: 3

Chart, scatter chart

Description automatically generated

The confusion matrix further represents that except for warrior2, the model predicts other poses with reasonable accuracy. The model misclassified 11 warrior2 poses as warrior1 and 5 warrior1 as tree pose.

Chart

Description automatically generatedA picture containing text

Description automatically generated

Discussion & Future work

The model can classify only six yoga poses for a single person, which may upgrade multiperson pose estimation. A pose estimation model that can predict all yoga exercise-related cardiac rehabilitation is a challenging task. The model's accuracy depends upon the quality of pose estimation of the tensorflow moveNet lite model. Calculating the angle of every body part joint from the coordinate value might give a good accuracy for a complicated yoga pose. However, there is still a massive amount of work that we can continue to examine. Future work will focus on a real-time exercise classifier and calculate the angle of the body part to improve the model's accuracy.