algorithms

MDPI

Article

Deep Learning Models for Yoga Pose Monitoring

Debabrata Swain [1], Santosh Satapathy [1], Biswaranjan Acharya [2,*], Madhu Shukla [3], Vassilis C. Gerogiannis [4,*], Andreas Kanavos [5,*] and Dimitris Giakovis [6]

[1] Computer Science and Engineering Department, Pandit Deendayal Energy University, Gandhinagar 382007, India
[2] Department of Computer Engineering-AI, Marwadi University, Rajkot 360003, India
[3] Department of Computer Engineering-AI and Big Data Analytics, Marwadi University, Rajkot 360003, India
[4] Department of Digital Systems, University of Thessaly, 41500 Larissa, Greece
[5] Department of Informatics, Ionian University, 49100 Corfu, Greece
[6] Experimental School of Larissa, Ministry of Education, 41334 Larissa, Greece
* Correspondence: biswaranjanacharya2020@gmail.com (B.A.); ; vgerogian@uth.gr (V.C.G.); akanavos@ionio.gr (A.K.)

Abstract: Activity recognition is the process of continuously monitoring a person's activity and movement. Human posture recognition can be utilized to assemble a self-guidance practice framework that permits individuals to accurately learn and rehearse yoga postures without getting help from anyone else. With the use of deep learning algorithms, we propose an approach for the efficient detection and recognition of various yoga poses. The chosen dataset consists of 85 videos with 6 yoga postures performed by 15 participants, where the keypoints of users are extracted using the Mediapipe library. A combination of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) has been employed for yoga pose recognition through real-time monitored videos as a deep learning model. Specifically, the CNN layer is used for the extraction of features from the keypoints and the following LSTM layer understands the occurrence of sequence of frames for predictions to be implemented. In following, the poses are classified as correct or incorrect; if a correct pose is identified, then the system will provide user the corresponding feedback through text/speech. This paper combines machine learning foundations with data structures as the synergy between these two areas can be established in the sense that machine learning techniques and especially deep learning can efficiently recognize data schemas and make them interoperable.

Keywords: yoga pose; machine learning; deep learning; data structures; asanas; C/NN; LSTM; mMedia pipe; pose prediction

Citation: Swain, D.; Satapathy, S.; Acharya, B.; Shukla, M.; Gerogiannis, V.C.; Kanavos, A.; Giakovis, D. Deep Learning Models for Yoga Pose Monitoring. *Algorithms* **2022**, *15*, 403. https://doi.org/10.3390/a15110403

Academic Editors: Alwin Poulose and Melania Susi

Received: 18 September 2022
Accepted: 28 October 2022
Published: 31 October 2022

1. Introduction

Recognizing human activity through computer vision has been an area of prime interest among researchers over several years, owing to its far-reaching applicability [1]. Human activity recognition has the potential to impact various domains such as robotics, human–computer interaction, gaming, video surveillance, biometric verification, chaos detection, sports monitoring, and health tracking among many others [2]. While multiple opportunities for implementing recognition systems have been explored throughout the years, its application in detecting yoga posture is considered a relatively new and under-researched field. Concretely, it holds the immense capacity to promote public health and welfare through the promotion of this ancient Indian practice, besides serving as an important milestone for researchers in the advancement in this sphere of activity recognition [3].

Activity recognition is the problem of predicting the movement of a person, based on sensor data or video sequences. It is a challenging computer vision problem that requires a lot of attention and improvement [1,4]. It has many real-life applications like user interface design, robot learning, surveillance, etc. It also faces many challenges due to differences

in video quality, partial appearance, brightness, proximity, background noise, angle, and viewpoint, etc. Human activity recognition is of concern with yoga pose recognition [3].

Yoga is a form of exercise that is extremely helpful in reducing mental stress and keeping a person physically and mentally fit. Moreover, it improves concentration, focus, calmness, and blood circulation [5]. Yoga originated in ancient India and has now been adopted by countries all over the world. Yoke refers to yuj in Sanskrit, a physical device that was used to join cattle. These devices were enormous, brute, and solid and were yoking long ago for war horses. Yoga was both the method and device to calm down the horses so that you could make them concentrate and perform well in the war [6]. Research shows that people who practice yoga daily have a positive mindset, an improved sense of energy to live a full life, and a good control over-breathing. The classic, Yoga Sutra, written by Patanjali first described yoga philosophy and its practices [7].

Yoga, an ancient discipline that originated in India and was once local to the nation, is now becoming popular worldwide on account of its various physical, mental and spiritual benefits [5]. The increasing significance of yoga in medicine can be attributed to its extraordinary healing effects in a variety of conditions affecting the human body such as respiratory issues, cardiac ailments, musculoskeletal problems, and deep learning uses in healthcare [6–8]. However, a certain gap exists between the new generation and their understanding and awareness of the boons of yoga, leading to a host of health issues associated with today's rapid lifestyle that can be easily curbed by the adoption of yoga as a part of daily routine. One of the major factors leading to the misconceptions surrounding yoga, eventually contributing to people's unwillingness to incorporate it into their lives, is the unavailability of proper guidance. However, the growing innovation in technology brings up the possibility of addressing this inaccessibility to right tutoring through a real-time self-learning aid, capable of detecting various yoga postures by activity recognition tools, allowing it to serve as a convenient means of instruction, necessary to help popularize the form in the desired manner.

Machine learning can be considered the study of algorithms that allows computer programs to automatically improve through experience. Specifically, machine learning algorithms build a model based on training data to make predictions or decisions without being explicitly programmed to do so. In recent years, a recent sub-field of machine learning has developed that is worth highlighting, entitled deep learning. This field addresses the use of different architectures based on artificial neural networks that, through a hierarchy of layers with non-linear processing units, learn high-level abstractions for data. Machine learning as well as deep learning techniques have been extensively employed to explore the causal relations and correlations among big data.

One of the key objects of research in the scientific domains of deep learning and computer vision is the human skill to identify and interpret another person's actions. Activity recognition refers to the vast topic of study that focuses on the detection of the activities and goals of one or more individuals from a sequence of observations of the individuals' actions and surroundings. Human activity recognition is the subset concerning the modeling of the human body to recognize a person's unique motion and behavior through data received from either sensors or vision-based techniques. The movements to be identified may include common activities such as sitting, standing, conversing, or walking, or may encompass highly specific and complicated gestures such as the vivid yoga postures, or specific athletic stances [9]. Accurate classification of human poses still poses a challenge to the scientific community, stimulating further work in this area.

Across the various characterization methods prevalent, a two-fold approach to the categorization problem remains uniform: firstly, the recognition part, and secondly, the localization problem, which indicates the description of the action to be detected and the identification of the portion of real-time movements that contains the object of interest respectively [1,10]. The entire process generally consists of three stages in terms of representation: the low-level core technology, the mid-level behavior detection, and the high-level implementation [2]. In the initial step of the first phase, object segmentation is

conducted on each frame of the video to distinguish and isolate the target object from its environment. The second level involves the retrieval of the attributes of separated objects such as color, shape, outlines, and movements and their subsequent representation into features, which can be broadly classified into four main types: frequency transform, local identifiers, space-time statistics, and body modeling data. The event detection and classification techniques employed to detect various human poses and movements predicated on the features derived from the previous step comprise the third stage of the operation. It concerns the difficulties faced at each of these sub-levels in the wake of a multitude of factors such as poor illumination, changes in video quality, partial visibility, closeness, background disturbance, angle involved, etc., that make human activity recognition a rather arduous task [11].

Intensive studies continue to be conducted within this field, with an aim to improve upon the existing results achieved. Lately, methods based on deep learning have gained popularity due to their use of representation learning algorithms that can instinctively extract suitable features from the input data supplied by sensors without the need for human interference and unique underlying patterns [12,13]. In this particular paper, we propose a deep learning model for recognizing and correcting yoga poses in real-time, with the motive to address the lack of availability of proper training and to make such education more affordable and suitable for people. This is implemented by attempting to exploit the opportunity of self at-home training through an application that can accurately differentiate between the poses practiced by expert trainers from those of beginners, resulting in reliable recommendations to learners concerning necessary corrections. With the use of CNN and LSTM models, an intelligent yoga pose monitoring system has been devised that takes as input the videos of sample subjects, in following converts them into a sequence of keypoints, which are the coordinates of key object points on a person, performs then the detection task, and finally, provides essential feedback to the user based on similarity in the form of text/speech [9]. The forthcoming sections will elucidate the precise methods employed and the results achieved, before highlighting the scope for further improvement.

In this work, a system that can recognize and correct yoga poses in real-time using various deep learning techniques is proposed. By using such a system, a whole yoga class atmosphere can be created at the user's home where the yoga pose is detected and automatically corrected by the system. Finding a yoga class that is affordable and aligned with a user's schedule is difficult. This paper tries to fill that gap of on-demand yoga classes with feedback on the yoga routine in real-time. The main idea is to create a deep learning model that can correctly classify a user's yoga pose by training it on a dataset of yoga videos.

The rest of the paper is organized as follows. The background material is introduced in Section 2. Section 3 overviews the basic concepts and methods used in this paper along with the dataset used. Section 4 presents the research results, and finally, Section 5 depicts conclusions and draws directions for future work.

2. Related Work

Different methods and techniques for a variety of applications have been proposed and employed in the field of real-time human monitoring. The proposed research aims to take into account the existing literature and further contribute towards an enhancement in the currently achievable results in the field of computer vision for human pose estimation [14]. Furthermore, the effectiveness of LSTM neural network and its important contribution for specific tasks was proved [15,16]. An attempt has been developed to devise a deep learning based system for efficiently detecting yoga poses and practically works as a substitute for a trainer by giving accurate feedback to the user.

A comparative study between machine learning and deep learning techniques for yoga pose estimation was carried out in [3] involving both technologies and analyzed the performance of the framework by using Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Convolutional Neural Network along with Long Short-Term

Memory (LSTM). The research concluded that the performance of hybrid CNN-LSTM model resulted in the least number of misclassifications. A system where the skeleton of the user is initially detected using an algorithm called tf-pose estimation, was introduced in [17]. In following, six different machine learning models, namely, Decision Tree, Random Forest, Logistic Regression, Naive Bayes, SVM and KNN were tested and compared. The used dataset contained 10 different yoga poses with around 5500 images and the optimum performance was obtained by the Random Forest classifier with an accuracy of 99.04%.

A model named ExNet, which is a multilayer Convolutional Neural Network, was proposed in [18]. The authors used an image dataset containing 2000 images of human exercise poses divided into five classes labeled as push up, pull up, cycling, swiss ball hamstring curl, walking. The model used Adam optimizer and automatic Learning Rate reduction method. After 50 epochs, ExNET got 82.68% accuracy on classifying 2D human exercise pose from the dataset. However, the model needs better hyper-parameter tuning because it faces the problem of overfitting. The authors in [19] managed to accurately design a real-time deep learning model using OpenPose to extract the keypoints from the videos supplied using a dataset comprising of six asanas, i.e., Trikonasana, Padmasana, Vrikshasana, Bhujangasana, Tadasana and Shavasana. While system was capable of an accuracy of 99.04% with the input taken framewise, the detection on the basis of a survey of the preset 45 edges yielded a performance measure of 99.38%. The work was further tested on twelve individuals and proved to be highly effective with an accuracy of 98.92%.

The authors in [20] captured video information using Microsoft Kinect. Kinect has a skeletal pursuit tool that may acknowledge twenty joints of a person's body. The chosen ten joint points were used for calculation and the reference structure for each yoga position was engineered by gathering info of the joint points from human posture. In following, the cosine similarity of vectors was calculated by finding angles between all the vectors connecting any two joint points. Specifically, if the calculated deviation is more than a set threshold value, the pose is classified as incorrect. A system capable of detecting poses associated with four kinds of normal human activity and identifying human action in both the second and third dimension, was presented in [21]. The research was conducted on data derived from four different sources, i.e., Human 3.6M, NTU, Penn Action and MPII using a Multitask Convolutional Neural Network [22].

In another employed method, when a pose is made by the user in front of his camera, Mediapipe library implements the geometric analysis based on the frames obtained from the camera [23]. Geometric analysis produces the output based on the angles between various joints. The calculated angles are then compared with the accurate angles that are saved in the database for the specific yoga pose. If the difference between these two angles is more than the set threshold, then the corresponding feedback is provided to the user, either in the form of text or through an audio device. One of the most recent works exhibits an LSTM-CNN based system for classification [24]. Specifically, the classification task was improved as the proposed method reduced the execution time by values ranging from 30% to 42%.

A smart fitness trainer on the concepts of pose detection and estimation using deep learning methods, was utilized in [25], where a regional multi-person pose estimation framework has been employed. In addition, AlphaPose has been used for identifying the keypoints, which are the human joints within every frame. The figure obtained on connecting them is then compared with the ideal poses, resulting in suggestions to the user on how to correctly practice the exercises. Another model to correct posture while exercising was proposed in [26], where the training of the model was achieved with use of OpenPose and 18 keypoints were defined for accurately detecting the differences in poses. The dataset used contained more than 100 videos of both correctly and incorrectly performed exercises. The task was conducted on 4 different exercises, namely front raise, standing shoulder press, bicep curl and shoulder shrug.

Finally, another deep learning technique for training the model to recognize yoga poses, was applied in [27]. After extracting features through Keras multi-purpose pose estimation,

the classification process was utilized into one of the six poses: corpse pose, tree pose, mountain pose, triangle pose, lotus pose and cobra pose. This process was implemented according to the angles between twelve keypoints using Multilayer Perceptron, resulting in an accuracy of 99.58%.

3. Proposed System

The proposed system takes video sequence frames as input in real time. The output would be the predicted yoga pose along with the corresponding feedback in terms of both angle and pose correction. The system consists of three main phases, namely, keypoints extraction, pose prediction as well as pose correction. The keypoints extraction phase aims at detecting and extracting the location of important keypoints based on user's position [14]. The pose prediction phase defines the model architecture and classifies whether the pose is correct or not. The final phase is pose correction where user is further given feedback for correction of the pose and depicts also the similarity percentage compared to the actual pose. Figure 1 illustrates the proposed system architecture along with the above three phases.

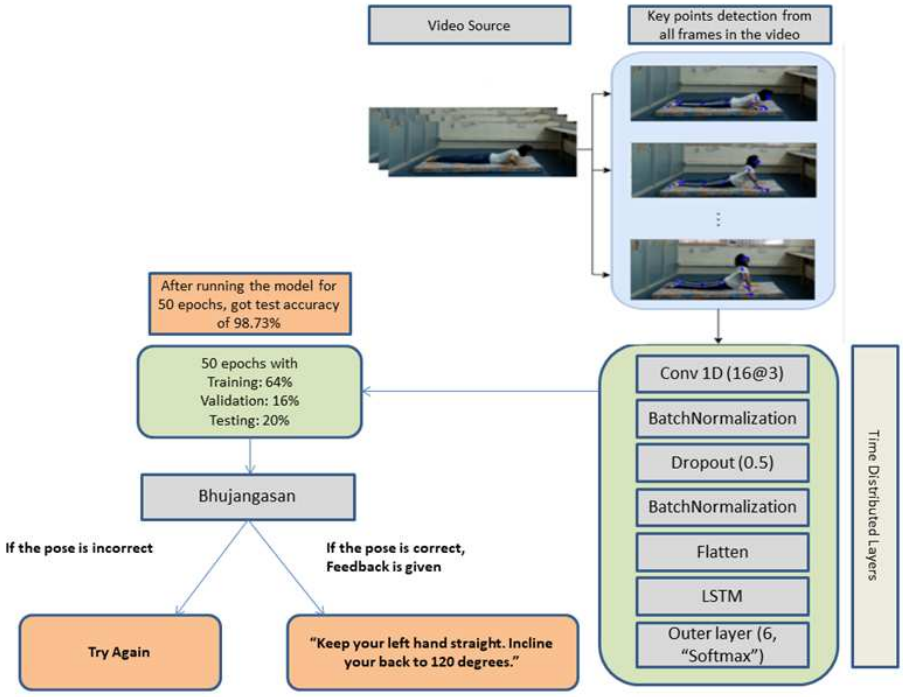


Figure 1. Flowchart of Proposed System.

3.1. Keypoints Extraction

The first phase consists the extraction of keypoints from all the frames of the video and in following stores it in JSON format. Keypoints comprise of different points in a person that are vital in the formation of a yoga pose; examples include shoulders, elbows, wrist, knees, etc. We used the MediaPipe library for keypoints extraction, which is a cross-platform library developed by Google that provides amazing ready-to-use ML solutions for computer vision tasks.

This phase utilizes a highly optimized pre-trained CNN model used for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation masks on the whole body from RGB video frames. Mediapipe library generates 3 coordinates (X, Y and Z), where Z indicates the depth of a 2D coordinate [10]. Figure 2 presents the 33 keypoints provided by MediaPipe library, whereas Figure 3 illustrates the result after extracting the keypoints with use of the MediaPipe library.

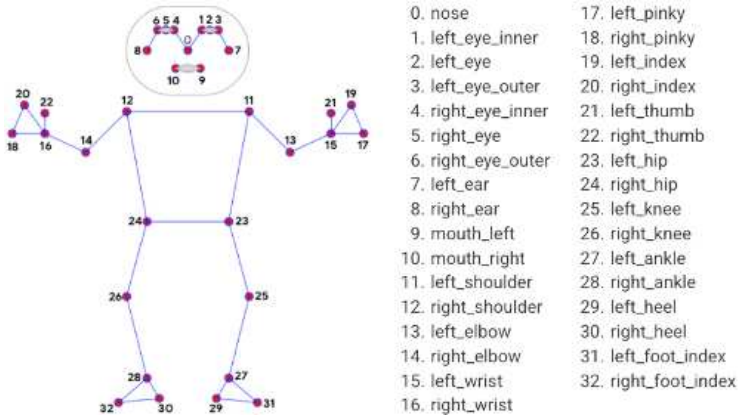


Figure 2. MediaPipe Keypoints.

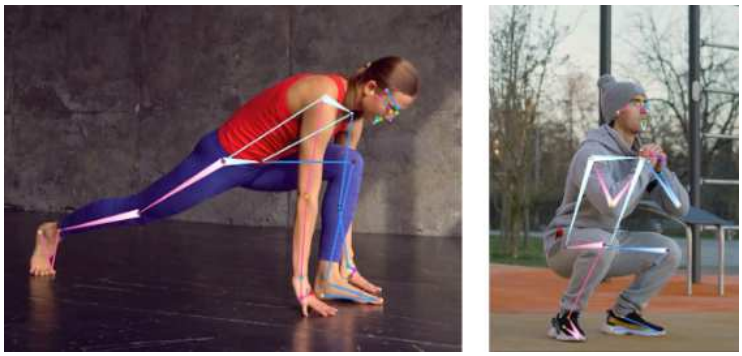


Figure 3. MediaPipe Keypoints on Actual Humans.

After converting the videos to JSON format, the split process into train, validation and test dataset is taking place. The split ratio used is 64:16:20 and each test case consists of a sequence of 45 frames with an overlap size of 36 frames, containing the coordinates of all the 33 keypoints. Thus, the input shape of a single test case can be given as $(45, 33, 2)$. The total count of training, validation and testing samples is 7063, 1832 and 2202.

This phase utilizes machine learning techniques to effectively design data structures in the context of the specific application. The contribution lies in the successful system development as well as in the employment of the evaluating operations.

3.2. Pose Prediction

The second phase concerns the creation of a deep learning model that aims at correctly classifying any real-time video to one of the six poses given in the dataset. Here, a hybrid model is incorporated, which is a mixture of CNN and LSTM, where CNN is used for feature extraction [13]. Specifically, CNN is a multilayered Artificial Neural Network (ANN) that is specifically created to work on images and is used for tasks like object recognition and image classification [28]. On the other hand, LSTM is useful for understanding the sequence of frames occurring in a particular yoga pose. Concretely, LSTM is a type of RNN that is equipped for learning and recollecting extremely long-term dependencies over long successions of input data [29].

In this work, a TimeDistributed layer along with CNN, which is particularly useful when working with video frames or time series data [30], is employed. Moreover, a Softmax layer, which uses a normalised exponentiation function, is used to find the probability of each yoga pose. The pose with the highest probability is predicted as the output.

To be more specific, The first layer used in the pose prediction is a CNN layer with 16 filters and a window size equals to 3. The activation function used in this layer is ReLU, which is a piece-wise linear function that gives an output equal to 0 when the input

provided is less than 0, else it gives as output the given input [31,32]. Equation (1) presents the activation function of *ReLU*:

$$ReLU(x) = max(0, x) \tag{1}$$

where $x \in R$.

The second layer used is a Batch Normalization layer that solves the problem of internal covariate shift and makes easier the flow of data through different layers [33].

The next layer used is a Dropout layer, which is a regularization technique for preventing overfitting at the rate of 0.5 [34]. Afterwards, another Batch Normalization layer is used. The output obtained from this layer is then passed to a Flatten layer that converts the data to a one-dimensional array. The next layer utilized is an LSTM layer where its size is 20 units with forget bias set to True in order to return the output of every node. The output of LSTM is generally generated through a series of Gated operations, like the Forget gate, the Input gate and the Output Gate. The mathematical equations for an LSTM are presented in the following Equations (2)–(7).

The state information of current and previous cell are stored in C. The sigmoid function output in the forget gate indicates which information to retain or forget. The output mainly depends upon the previous state output vector h_{r-1} and current state input vector x_r. If there is difference, then σ function does not allow the information to retain. Moreover, the forget bias gives a shift to the output either in the direction of 1 (retain) or 0 (forget); the model outperforms with forget bias value equal to 0.5. In the input gate, the σ function decides which value to update and tanh function adds new values for the state. Regarding the output layer, the σ function determines the value to be selected as output h_r.

$$l_r = \sigma(W_l \cdot [h_{r-1}, x_r] + b_l) \tag{2}$$

$$m_r = \sigma(W_m \cdot [h_{r-1}, x_r] + b_m) \tag{3}$$

$$\hat{C}_r = tanh(W_c \cdot [h_{r-1}, x_r] + b_c) \tag{4}$$

$$C_r = l_r * C_{r-1} + i_r * \hat{C}_r \tag{5}$$

$$o_r = \sigma(W_o \cdot [h_{r-1}, x_r] + b_o) \tag{6}$$

$$h_r = o_r * tanh(C_r) \tag{7}$$

where

- l_r, m_r, o_r and h_r are the outputs of forget gate, input gate, output gate and current state, respectively;
- W_f, W_m and W_o are the weights of forget gate, input gate and output gate, respectively;
- x_r is the current state input;
- b_l, b_m and b_o are the bias of forget gate, input gate and output gate, respectively;
- m_r is the output of input gate;
- $\hat{C}_r$ and C_r are the current and previous cell state.

The final layer used is a Dense layer that uses Softmax as the activation function, which assigns probabilities of different poses based on the current given input [35]. The mathematical equation of Softmax activation function is presented in Equation (8).

$$\sigma(\underline{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{8}$$

where σ is the softmax, $\underline{z}$ is the input vector, e^{z_i} is the standard exponential function for input vector, K is the number of classes in the multi-class classifier and e^{z_j} is the standard exponential function for output vector.

The output obtained after this layer is polled on 45 frames to get the final prediction. The Adam optimizer [36] is employed with learning rate equal to 0.0001; this optimizer

helps the model to fast converge by the addition of momentum term and scaling term as proved in Equation (9). It combines the idea of momentum and RMSprop optimizer and helps to avoid the exponential decay of learning rate issue.

$$\theta_{new} = \theta_{old} - \eta \cdot \hat{m} \cdot \phi\sqrt{\hat{s} + \epsilon} \tag{9}$$

where θ_{new} and θ_{old} are the new and old weight value, respectively, η is the learning rate, $\hat{m}$ is the momentum term, $\hat{s}$ is the scaling term, ϵ is the smoothing term to avoid zero division error and ϕ is the element wise division operation.

The loss function used is categorical cross-entropy which is very popular for multi-class classification tasks [37]. Equation (10) depicts the mathematical equation used in categorical cross-entropy loss function.

$$E_{CC} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}(p_{ic}log(y_{ic})) \tag{10}$$

where E_{CC} is the categorical cross-entropy, N is the number of pairs available in training set, C is the number of categories, p_{ic} is a binary indicator function that detects whether the i^{th} training pattern belongs to c^{th} category and y_{ic} is a predicted probability distribution for i^{th} observation belonging to class c.

The metric used to gauge the performance of the model is accuracy [38], where the particular model was trained for a total of 50 epochs. Initially, the growth was exponential and became steady after a few epochs. Specifically, after each epoch, we checked whether the accuracy has been improved and became better than the best accuracy obtained. If it is better than the best accuracy, then the best accuracy is replaced by the current accuracy. All the parameters used in the model have been perfectly tuned using hyper-parameter tuning in order to obtain the most optimized results [39,40].

3.3. Pose Correction

After the predicted pose is classified to be correct, with respect to the chosen pose, the user is provided with the appropriate feedback and in following, the similarity percentage (using cosine similarity) is calculated to be presented to the user.

For all the six yoga poses present in the dataset, which will be introduced below, important and critical angles have been identified and rules have been formulated for each pose. For each rule, a threshold is set, which constitutes the maximum deviation allowed for the user from the standard pose. If the user exceeds this threshold value, a corresponding feedback is given accordingly in the form of text and speech. The angle between two keypoints can be found out by calculating the tangent inverse of the slope with positive X-axis. Equation (11) depicts the formula for finding the angle, given the two coordinates of the keypoints:

$$\Theta = tan^{-1}(\frac{y_2 - y_1}{x_1 - x_2}) \tag{11}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of two keypoints.

Feedback is initially obtained in the form of text, which is then converted into speech using Pyttsx3 library [41]; it is a text-to-speech converter and works offline as well.

Cosine similarity, which is a measure that compares two vectors by calculating the cosine of angles between them [42], is also reported to the user. The value of this metric varies from -1 to $+1$. If the similarity score lies between -1 and 0 then it is multiplied with -1 to transform it to positive side. Then, the similarity is calculated according to the score in the range 0 to $+1$. The mathematical formula of cosine similarity is given in Equation (12):

$$cos\theta = \frac{A \cdot B}{||A| \cdot |B||} \tag{12}$$

where, A and B are two vectors in a multidimensional space.

In this work, cosine similarity is calculated between keypoints of the user's pose and the standard pose. This way, this measure reveals the degree of closeness to the actual pose. Since the distance of different users can vary based on their position from the camera, all keypoints are initially normalized for determining a similar scale for all of them.

3.4. Dataset

The dataset comprises a total of 85 videos of 1080p HD quality, recorded by 15 different individuals. It contains videos of 6 different yoga poses, namely Bhujangasana, Padmasana, Shavasana, Tadasana, Trikonasana and Vrikshasana as illustrated in Figure 4. The videos were recorded using an NVIDIA TITAN X GPU and Intel Xeon processor with 32 GB RAM with a frame rate equals to 30 frames per second. The users tried to perform different asanas with many variations. The characteristics of the corresponding dataset are presented in Table 1.

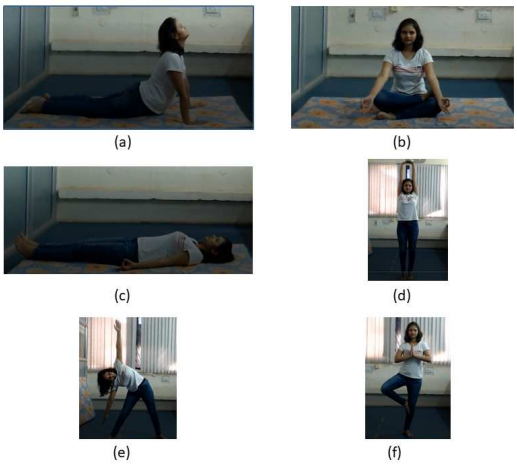


Figure 4. Yoga Poses: (**a**) Bhujangasana, (**b**) Padmasana, (**c**) Shavasana, (**d**) Tadasana, (**e**) Trikonasana, (**f**) Vrikshasana.

Table 1. Dataset.

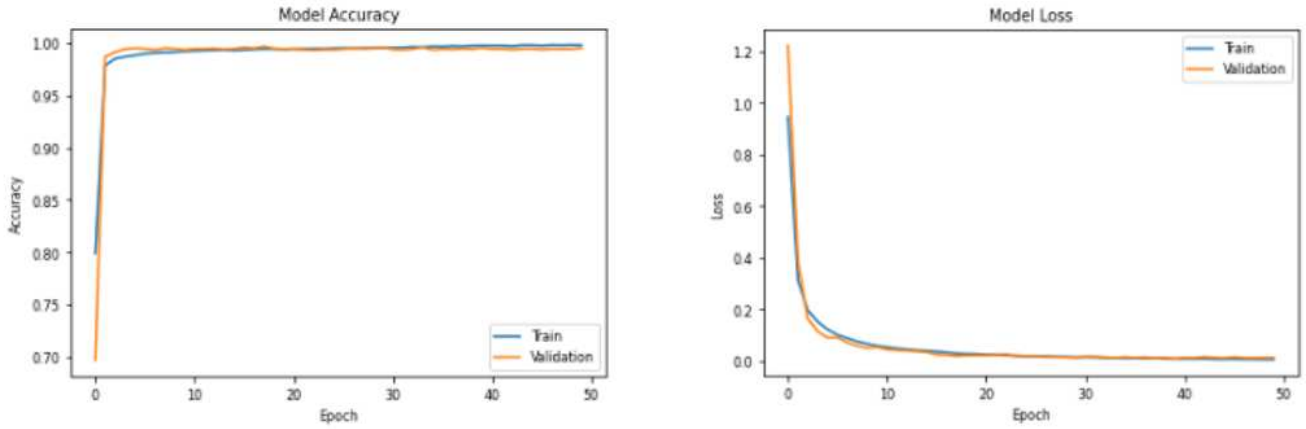
No.	Asana Name	#Persons	#Videos
1	Bhujangasana	15	16
2	Padmasana	14	14
3	Shavasana	15	15
4	Tadasana	15	15
5	Trikonasana	13	13
6	Vrikshasana	12	12

4. Results

The prediction of yoga pose for a sequence of 45 frames is implemented using polling, where the mode of all yoga pose predictions is considered as the final prediction.

4.1. Model Accuracy and Model Loss

After running the model for 50 epochs, the training accuracy obtained is 99.49%, whereas validation accuracy is 99.70%. Figure 5 illustrates the graphical representation of both model accuracy and model loss.



(**a**) (**b**)

Figure 5. (**a**) Model Accuracy, (**b**) Model Loss.

4.2. Confusion Matrix

A confusion matrix is considered an important performance measure that is used for calculation of other analytics like recall, precision, specificity, sensitivity, etc. It is also used for classification problems in order to summarize prediction results [43]. Furthermore, it consists of four main values, which are true positive, true negative, false positive and false negative [44].

The test accuracy obtained is 99.53% and Figure 6 presents the confusion matrix between the predicted yoga asanas and the actual asanas. The darker the blue colour is on the diagonal, the higher is the correlation between the predicted and true values. Since most of the diagonal values are close to 1, the matrix indicates high correlation and this corresponds to a fairly accurate model.

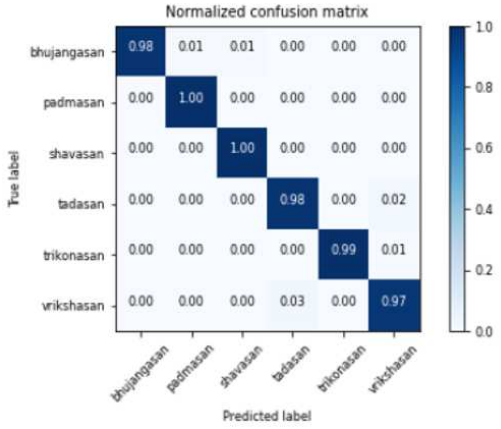


Figure 6. Confusion Matrix.

4.3. Precision

Precision is a metric used to evaluate the performance of a model. Concretely, it is defined as the fraction of samples classified correctly as positive given all the samples that are indeed positive [21]. Precision obtained from the model is 0.9866. Its mathematical formula is given in Equation (13):

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

where TP are the true positives and FP the false positives.

4.4. Recall

Recall is yet another metric used to evaluate the performance of a model. It is defined as the fraction of samples classified correctly as positive given all the samples that are predicted positive [45]. Recall obtained from the model is 0.9869. Its mathematical formula is given in Equation (14):

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

where TP are the true positives and FN the false negatives.

4.5. Receiver Operating Characteristic (ROC) Curve

Furthermore, we employ ROC curve, which is a graphical representation that reveals the performance of a model by plotting false positive rate against true positive rate [46]. A model is said to be ideal when the area under the ROC curve tends to 1. Figure 7 illustrates the ROC curve of all the asanas used for our experiments. The obtained area under the curve is 0.99, a value that can be considered as exceptionally high.

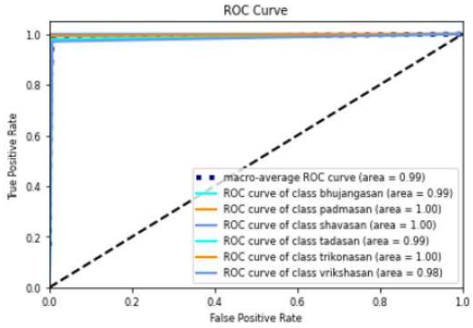


Figure 7. ROC Curve.

4.6. Comparative Analysis of Activation Functions used

The activation functions are essential to any deep learning model in helping the artificially designed framework to autonomously learn complex patterns from the data supplied to it. Also known as the transfer function, it is generally located at the end, with an assigned task of carrying out a number of non-linear operations on the input, before finalizing it as the output and sending it to the successive layer of neurons.

Four different activation functions were employed and tested in this study, namely, ReLu, Sigmoid, Softmax and Tanh, in alphabetic order. As can be depicted in Figure 8, ReLu proved to be the most accurate followed by Sigmoid, Softmax and Tanh.

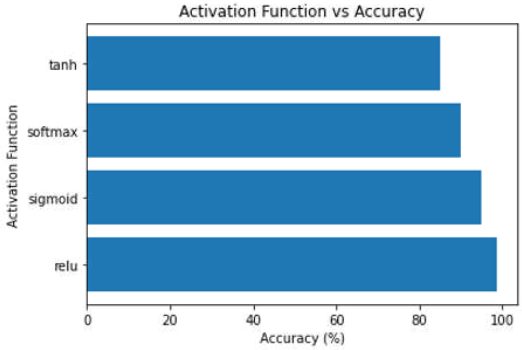


Figure 8. Activation Functions used against their corresponding Accuracy.

4.7. Comparative Analysis of Optimizers Used

The optimizers in a neural network play an important role in reducing the model loss and simultaneously improve accuracy. These are the algorithms that are responsible for

modifying certain aspects of the network, such as weights and learning rate, in order to provide optimal results.

Three different optimizers were employed and tested in this study, namely, Adam, RMSprop and Stochastic Gradient Descent (SGD), in alphabetic order. As can be depicted in Figure 9, Adam proved to be the most accurate followed by RMSprop and SGD. Specifically, the obtained accuracy values for Adam, RMSprop and SGD are 98.72%, 94.66% and 89.16%, respectively.

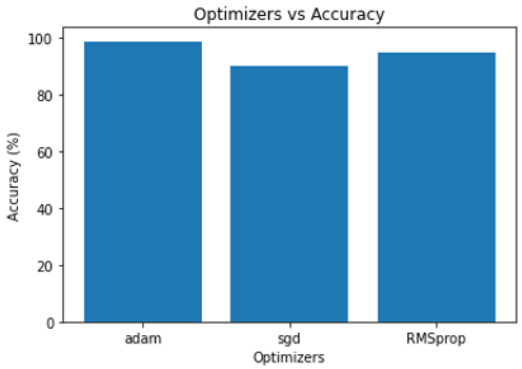


Figure 9. Optimizers used and their corresponding Accuracy.

4.8. Impact of Number of CNN Layers on Model Performance

The number of hidden layers in a CNN model can significantly affect its classification accuracy. While an increased number of layers is generally attributed to an increased overall accuracy, this is not a universal observation and often depends on the complexity of the task involved. In the absence of a sufficiently large training set, adding additional layers can provide a considerable boost to a highly large and complicated network that is prone to overfitting, and thereby to reduce the accuracy obtained on the test data.

On the other hand, the addition of extra layers can also contribute to a greater execution time, as demonstrated through the line graph in Figure 10. For this particular model, the best results were obtained when using a single convolutional layer.

The time needed by the model due to inclusion of layers from 1 to 5 is 1100 ms, 2500 ms, 4600 ms, 9100 ms and 19,900 ms, respectively. This proves that the addition of layers increased the models processing time in terms of prediction.

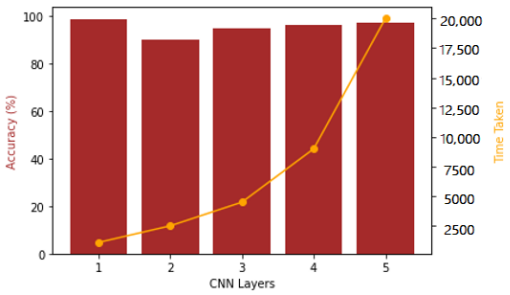


Figure 10. Accuracy associated with varied number of CNN Layers.

4.9. Impact of Number of CNN Filters on Model Performance

A strong correlation can be identified among the number of CNN filters used and the corresponding accuracy as can be viewed in Figure 11. Every filter layer is responsible for extracting a certain set of features from the input data. As patterns continue to get more complicated, a growing number of filters are required for accurately extracting a number of different combinations of prominent features from the data received. Specifically, for at least 8 CNN filters, the accuracy has value equal to 100%.

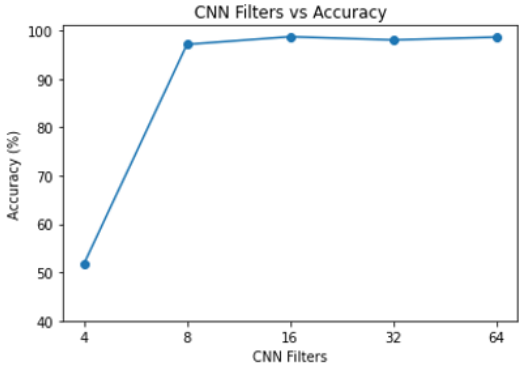


Figure 11. Accuracy associated with varied number of CNN Filters.

4.10. Impact of Learning Rate on Model Performance

The learning rate refers to the extent of adjustment allowed in the model in response to the error observed every time the model weights are altered. Deciding upon an optimal learning rate can be tricky, since neither of the extremes is desirable. While a lower learning rate may help in building a more accurate model, thus it can result in an extremely slow training process. Similarly, a very high value can significantly reduce the learning time, but the cost of accuracy could be further increased.

Figure 12 presents the dependence of accuracy in terms of learning rate. Specifically, the learning rate takes values in the range between 0.0010 and 0.1 whereas the accuracy is at least 97%.

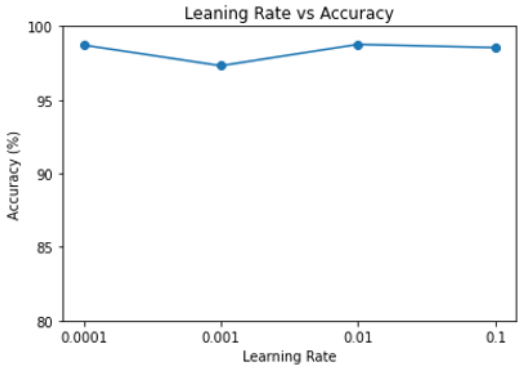


Figure 12. Impact of Learning Rate on Accuracy.

4.11. Impact of Dropout Rate on Model Performance

At every stage of the training process, a certain number of neurons are dropped out from each layer. This fraction of neurons, whose values are to be nulled, increases the dropout rate. More to the point, dropout is necessary to prevent overfitting in neural networks and is also used as a means of dealing with complex co-adaptions caused by neurons with similar connection weights during training.

Figure 13 presents the ideal dropout rate obtained for this model. Specifically, the dropout rate takes values in the range between 0.30 and 0.70 whereas the accuracy is at least 97%.

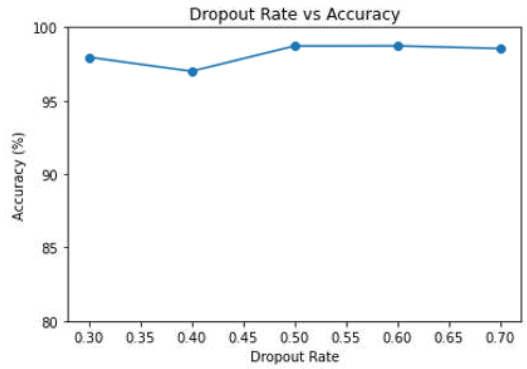


Figure 13. Impact of Dropout Rate on Accuracy.

4.12. Overall Result

After the combination of the three aforementioned phases, a reliable front-end was created for monitoring the user in real-time for pose prediction and correction. Figure 14 illustrates the overall view of the front-end while Figure 15 presents the result provided to user after pose prediction and correction.

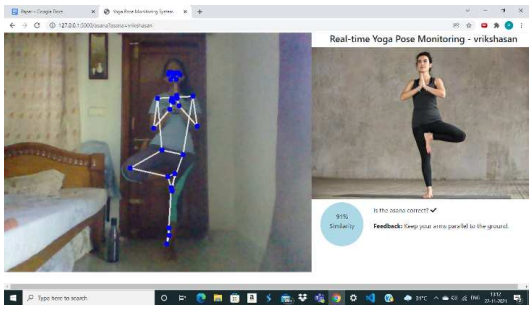


Figure 14. Front-end.

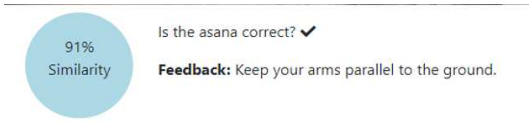


Figure 15. Prediction and Correction.

4.13. Discussion

In the proposed work, an advancement has been implemented comparing to other works with similar architectures [19,27]. Specifically, Table 2 depicts the basic differences of our proposed work against the above.

The use of more key points in the proposed work results in better and more reliable quality of prediction. In addition, the Batch Normalization (BN) layer is used after the Dropout layer for faster convergence; this introduction of BN layer ensures that the input pixels have similar distribution and helps in making the training faster and better. Moreover, Adadelta optimizer used in [27] is slower and inefficient as it considers stepwise convergence.

Table 2. Differences with other works.

Feature	[19]	[27]	Proposed Work
Library used for extracting Keypoints	OpenPose	-	Mediapipe
Number of Keypoints	18	13	33
Epochs for Convergence	100	1000	50
Optimizer	-	Adadelta	Adam
Layers	Dropout and Flattening	LSTM and CNN	Dropout and Batch Normalization
Feedback after Yoga Pose Identification	no	Finding the Angle between the joints	Finding the Similarity with the correct Yoga Pose

5. Conclusions and Future Work

In this paper, an efficient system for real-time yoga monitoring was proposed. It initially identifies keypoints of a particular user with use of the MediaPipe library, where important coordinates are recorded and stored in JSON format. We then create in real-time a sequence of 45 frames, and pass it to the model. In following, the model, when utilizing a mixture of CNN and LSTM, finds the useful features using CNN and observes the occurrence of frames sequence employing LSTM. The Softmax layer in the end finds the probability of each yoga asana for the current frame sequence and outputs the highest probability asana. More to the point, the output of each frame is polled on 45 frames where mode is calculated and provided to the user as output.

The system achieves an excellent accuracy of 99.53% on the test dataset. If the pose is classified as correct, further feedback is given to the user based on a set threshold. Specifically, the threshold is set in such a way that it does not make the system too strict and also ensures accurate pose and angles made by the user at the same time. Finally, similarity percentage is shown to the user when compared with the standard pose.

Regarding future work, variations and combinations of the proposed set of models presented in this work are worth trying, in order to study whether it is possible to further improve the accuracy. Furthermore, the existing classifiers could be tested in larger datasets to verify the high levels of accuracy achieved in sentiment detection. In addition to the larger volume of the dataset, it is important to add more numerical features than those contained in the set used for the purposes of this work. Moreover, the inefficiencies of single models can be resolved by applying several combination techniques, which will lead to more accurate results as in [47].

Author Contributions: Data curation, D.S., S.S., B.A., M.S., V.C.G. and A.K.; Methodology, D.S., S.S., B.A., M.S., V.C.G. and A.K.; Writing—original draft, D.S., S.S., B.A., M.S., V.C.G. and A.K.; Writing—review & editing, D.S., S.S., B.A., M.S., V.C.G., A.K. and D.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vrigkas, M.; Nikou, C.; Kakadiaris, I.A. A Review of Human Activity Recognition Methods. *Front. Robot. AI* **2015**, *2*, 28. [CrossRef]
2. Ke, S.; Hoang, L.U.T.; Lee, Y.; Hwang, J.; Yoo, J.; Choi, K. A Review on Video-Based Human Activity Recognition. *Computers* **2013**, *2*, 88–131. [CrossRef]
3. Kothari, S. Yoga Pose Classification Using Deep Learning. Ph.D. Thesis, San Jose State University, San Jose, CA, USA, 2020.
4. Acharya, B.R.; Gantayat, P.K. Recognition of Human Unusual Activity in Surveillance Videos. *Int. J. Res. Sci. Innov. (IJRSI)* **2015**, *2*, 18–23.
5. Stephens, I. Medical Yoga Therapy. *Children* **2017**, *4*, 12. [CrossRef]

6. Newcombe, S. The Development of Modern Yoga: A Survey of the Field. *Relig. Compass* **2009**, *3*, 986–1002. [CrossRef]
7. Woodyard, C. Exploring the Therapeutic Effects of Yoga and its Ability to Increase Quality of Life. *Int. J. Yoga* **2011**, *4*, 49. [CrossRef]
8. Dash, S.; Acharya, B.R.; Mittal, M.; Abraham, A.; Kelemen, A. *Deep Learning Techniques for Biomedical and Health Informatics*; Springer: Berlin/Heidelberg, Germany, 2020.
9. Brownlee, J. Deep Learning Models for Human Activity Recognition. 2018. Available online: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1480070&dswid=-5372 (accessed on 10 September 2022).
10. Zhang, F.; Bazarevsky, V.; Vakunov, A.; Tkachenka, A.; Sung, G.; Chang, C.; Grundmann, M. MediaPipe Hands: On-device Real-time Hand Tracking. *arXiv* **2020**, arXiv:2006.10214.
11. Alzahrani, M.; Kammoun, S. Human Activity Recognition: Challenges and Process Stages. *Int. J. Innov. Res. Comput. Commun. Eng. (IJIRCCE)* **2016**, *5*, 1111–1118.
12. Gupta, S. Deep Learning Based Human Activity Recognition (HAR) Using Wearable Sensor Data. *Int. J. Inf. Manag. Data Insights* **2021**, *1*, 100046. [CrossRef]
13. Jogin, M.; Madhulika, M.S.; Divya, G.D.; Meghana, R.K.; Apoorva, S. Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning. In Proceedings of the 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 18–19 May 2018; pp. 2319–2323.
14. Josyula, R.; Ostadabbas, S. A Review on Human Pose Estimation. *arXiv* **2021**, arXiv:2110.06877. Available online: https://arxiv.org/pdf/2110.06877.pdf (accessed on 10 September 2022).
15. Kanavos, A.; Kounelis, F.; Iliadis, L.; Makris, C. Deep Learning Models for Forecasting Aviation Demand Time Series. *Neural Comput. Appl.* **2021**, *33*, 16329–16343. [CrossRef]
16. Lyras, A.; Vernikou, S.; Kanavos, A.; Sioutas, S.; Mylonas, P. Modeling Credibility in Social Big Data using LSTM Neural Networks. In Proceedings of the 17th International Conference on Web Information Systems and Technologies (WEBIST), Online, 26–28 October 2021; pp. 599–606.
17. Agrawal, Y.; Shah, Y.; Sharma, A. Implementation of Machine Learning Technique for Identification of Yoga Poses. In Proceedings of the 9th IEEE International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 10–12 April 2020; pp. 40–43.
18. Haque, S.; Rabby, A.S.A.; Laboni, M.A.; Neehal, N.; Hossain, S.A. ExNET: Deep Neural Network for Exercise Pose Detection. In Proceedings of the 2nd International Conference on Recent Trends in Image Processing and Pattern Recognition (RTIP2R), Solapur, India, 21–22 December 2018; Communications in Computer and Information Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 1035, pp. 186–193.
19. Yadav, S.K.; Singh, A.; Gupta, A.; Raheja, J.L. Real-time Yoga Recognition using Deep Learning. *Neural Comput. Appl.* **2019**, *31*, 9349–9361. [CrossRef]
20. Kadbhane, S.G.; Datir, K.D.; Jagdale, T.S.; Dhongade, S.S.; Jagtap, G.R. Yoga Posture Recognition. *Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE)* **2021**, *10*, 143–147.
21. Buckland, M.K.; Gey, F.C. The Relationship between Recall and Precision. *J. Am. Soc. Inf. Sci. (JASIS)* **1994**, *45*, 12–19. [CrossRef]
22. Luvizon, D.C.; Picard, D.; Tabia, H. 2D/3D Pose Estimation and Action Recognition Using Multitask Deep Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 5137–5146.
23. Anilkumar, A.; Athulya, K.; Sajan, S.; Sreeja, K. Pose Estimated Yoga Monitoring System. In Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems (ICICNIS), Kottayam, India, 28–29 June 2021.
24. Savvopoulos, A.; Kanavos, A.; Mylonas, P.; Sioutas, S. LSTM Accelerator for Convolutional Object Identification. *Algorithms* **2018**, *11*, 157. [CrossRef]
25. Zou, J.; Li, B.; Wang, L.; Li, Y.; Li, X.; Lei, R.; Sun, S. Intelligent Fitness Trainer System Based on Human Pose Estimation. In Proceedings of the International Conference On Signal And Information Processing, Networking And Computers (ICSINC), Yulin, China, 29 November–1 December 2018; pp. 593–599.
26. Chen, S.; Yang, R.R. Pose Trainer: Correcting Exercise Posture using Pose Estimation. *arXiv* **2020**, arXiv:2006.11718.
27. Thoutam, V.A.; Srivastava, A.; Badal, T.; Mishra, V.K.; Sinha, G.R.; Sakalle, A.; Bhardwaj, H.; Raj, M. Yoga Pose Estimation and Feedback Generation Using Deep Learning. *Comput. Intell. Neurosci.* 2022. Available online: https://www.hindawi.com/journals/cin/2022/4311350/ (accessed on 10 September 2022).
28. Al-Saffar, A.A.M.; Tao, H.; Talab, M.A. Review of Deep Convolution Neural Network in Image Classification. In Proceedings of the IEEE International Conference on Radar Antenna, Microwave, Electronics and Telecommunications (ICRAMET), Jakarta, Indonesia, 23–24 October 2017; pp. 26–31.
29. Shiranthika, C.; Premakumara, N.; Chiu, H.L.; Samani, H.; Shyalika, C.; Yang, C.Y. Human Activity Recognition Using CNN & LSTM. In Proceedings of the 5th IEEE International Conference on Information Technology Research (ICITR), Moratuwa, Sri Lanka, 2–4 December 2020; pp. 1–6.
30. Ullah, A.; Ahmad, J.; Muhammad, K.; Sajjad, M.; Baik, S.W. Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features. *IEEE Access* **2018**, *6*, 1155–1166. [CrossRef]
31. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). *arXiv* **2018**, arXiv:1803.08375.

32. Swain, D.; Pani, S.K.; Swain, D. Diagnosis of Coronary Artery Disease Using 1-D Convolutional Neural Network. 2019. Available online: https://www.ijrte.org/wp-content/uploads/papers/v8i2/B2693078219.pdf (accessed on 10 September 2022).
33. Thakkar, V.; Tewary, S.; Chakraborty, C. Batch Normalization in Convolutional Neural Networks—A Comparative Study with CIFAR-10 Data. In Proceedings of the 5th IEEE International Conference on Emerging Applications of Information Technology (EAIT), Kolkata, India, 12–13 January 2018; pp. 1–5.
34. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
35. Szandala, T. Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In *Bio-Inspired Neurocomputing*; Springer: Singapore, 2021.
36. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
37. Rusiecki, A. Trimmed Categorical Cross-entropy for Deep Learning with Label Noise. *Electron. Lett.* **2019**, *55*, 319–320. [CrossRef]
38. Fatourechi, M.; Ward, R.K.; Mason, S.G.; Huggins, J.E.; Schlögl, A.; Birch, G.E. Comparison of Evaluation Metrics in Classification Applications with Imbalanced Datasets. In Proceedings of the 7th International Conference on Machine Learning and Applications (ICMLA), San Diego, CA, USA, 11–15 December 2008; pp. 777–782.
39. Diaz, G.I.; Fokoue-Nkoutche, A.; Nannicini, G.; Samulowitz, H. An Effective Algorithm for Hyperparameter Optimization of Neural Networks. *IBM J. Res. Dev.* **2017**, *61*, 1–11. [CrossRef]
40. Swain, D.; Pani, S.K.; Swain, D. An Efficient System for the Prediction of Coronary Artery Disease using Dense Neural Network with Hyper Parameter Tuning. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2019**, *8*, 689–695.
41. Subhash, S.; Siddesh, S.; Srivatsa, P.N.; Ullas, A.; Santhosh, B. Developing a Graphical User Interface for an Artificial Intelligence-Based Voice Assistant. *Int. J. Organ. Collect. Intell. (IJOCI)* **2021**, *11*, 49–67. [CrossRef]
42. Lahitani, A.R.; Permanasari, A.E.; Setiawan, N.A. Cosine Similarity to Determine Similarity Measure: Study Case in Online Essay Assessment. In Proceedings of the 4th IEEE International Conference on Cyber and IT Service Management, Bandung, Indonesia, 26–27 April 2016; pp. 1–6.
43. Hasnain, M.; Pasha, M.F.; Ghani, I.; Imran, M.; Alzahrani, M.Y.; Budiarto, R. Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking. *IEEE Access* **2020**, *8*, 90847–90861. [CrossRef]
44. Swain, D.; Ballal, P.; Dolase, V.; Dash, B.; Santhappan, J. An Efficient Heart Disease Prediction System Using Machine Learning. *Mach. Learn. Inf. Process.* **2020**, *1101*, 39–50.
45. Bradley, A.P. The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognit.* **1997**, *30*, 1145–1159. [CrossRef]
46. Swain, D.; Pani, S.K.; Swain, D. A Metaphoric Investigation on Prediction of Heart Disease using Machine Learning. In Proceedings of the IEEE International Conference on Advanced Computation and Telecommunication (ICACAT), Bhopal, India, 28–29 December 2018; pp. 1–6.
47. Vernikou, S.; Lyras, A.; Kanavos, A. Multiclass Sentiment Analysis on COVID-19 related Tweets using Deep Learning Models. *Neural Comput. Appl.* **2022**, *34*, 19615–19627. [CrossRef]