Japhne *et al. Journal of Big Data* (2024) 11:101
https://doi.org/10.1186/s40537-024-00915-8

Journal of Big Data

RESEARCH Open Access

Fitcam: detecting and counting repetitive exercises with deep learning

Ferdinandz Japhne[1], Kevin Janada[1], Agustinus Theodorus[1] and Andry Chowanda[1*]

*Correspondence:
achowanda@binus.edu

[1] Computer Science Department, School of Computer Science, Bina Nusantara University, 11480 Jakarta, Indonesia

Abstract

Physical fitness is one of the most important traits a person could have for health longevity. Conducting regular exercise is fundamental to maintaining physical fitness, but with the caveat of occurring injury if not done properly. Several algorithms exists to automatically monitor and evaluate exercise using the user's pose. However, it is not an easy task to accurately monitor and evaluate exercise poses automatically. Moreover, there are limited number of datasets exists in this area. In our work, we attempt to construct a neural network model that could be used to evaluate exercise poses based on key points extracted from exercise video frames. First, we collected several images consists of different exercise poses. We utilize the the OpenPose library to extract key points from exercise video datasets and LSTM neural network to learn exercise patterns. The result of our experiment has shown that the methods used are quite effective for exercise types of push-up, sit-up, squat, and plank. The neural-network model achieved more than 90% accuracy for the four exercise types.

Keywords: Human activity recognition, Pose estimation, Long short term memory

Introduction

Physical fitness is one of the most important traits a person could have for health longevity. There are a variety of physical exercises that can help us sustain ourselves; for example, aerobics have been proven to reduce stress and improve vascular health [1]. Physical exercise has also been associated with an increase in happiness with the positive side-effect of improving physical health [2], but unfortunately, there is also a risk of injury associated with any type of exercise; Gray and Finch identified that one of the categories that cause injuries in fitness centres is mainly divided into three categories: overexertion, strenuous, and unnatural movement [3]. Therefore, to avoid injury, it is necessary to know how to exercise properly. Hiring personal trainers is one of the solutions to ensure that injuries do not occur while exercising. Nevertheless, there are caveats to hiring a personal trainer. To start, there are several good criteria such as four years of experience in the field, credible certification from a respected organization, understanding of nutritional science, having the ability to work with different clients, having strategies that suit client behavior, having a positive and supportive leadership style, and last but not least good communication skills [4].

While personal trainers can help prevent injuries in fitness centers by planning exercises that have low to moderate risks for their clients [5], the number of personal training sessions has reduced during the pandemic, and personal trainers are reporting a loss of clients [6]. A report by Glofox showed that because of the Covid-19 lockdowns, fitness centers are increasing the frequency of online gym classes. The surge in online gym classes is an organic response from the fitness industry to survive the lockdowns [7, 8], but online classes have unique limitations. For example, physical educators have a hard time overcoming the monotony and limited environmental conditions experienced by the students. Furthermore, the lack of experience teaching physical education from online platforms also impacts the classes making them less effective [9]. Ultimately, online physical education classes limit the number of interactions a client has with their trainer.

Lockdowns have increased the difficulty of day-to-day exercise because of the lack of motivation and the limited opportunities to find alternatives to gyms [8]. According to a study by Bravata et al. [6], almost 40% of Norwegian personal trainers who participated in the study have found difficulty in the continuance of conducting physical activities and exercising during lockdowns. Thus, even though restrictions have subsided, there still needs to be a way to conduct online classes. A study about gamification and self-monitoring concluded that it might be used as a source of motivation for physical activities [9]. Self-monitoring accurately enables self-reflection and increases the reactivity necessary to respond and cue behavior change [10]. However, to self-monitor effectively, there needs to be a monitoring device that can track movements [11, 12].

Pose estimation techniques use various data capturing mechanisms such as wearable inertial sensors and visual sensors. There are several benefits and disadvantages each type of sensor has compared to the other. For example, visual sensors can collect from every part of the body at once but have the disadvantage of being limited by occlusion [13]. In contrast, wearable inertial sensors are ubiquitous but cannot collect data if the sensor is not located on the active part of the body [14]. High-end 3D marker-based motion capture systems such as 3DMoCap can overcome this limitation but have limited accessibility making them impractical for everyday use. A study comparing deep-learning-based pose estimation systems and marker-based motion capturing systems found that deep-learning-based systems, in some segments, have comparable performance to marker-based systems, but marker-based systems still have the edge and are the better choice out of the two.

Therefore, we present a vision-based neural network model to self-monitor exercises accurately and automate exercise tracking. For our specific use-case, we would need a pose estimation technique to gather Spatio-temporal data, and we settled for Open Pose, a 2D skeleton-based pose estimation model provided by Dua et al. [15]. Open Pose has been used in a multitude of pose classification studies as part of the feature extraction method.

To develop our neural network, we have recorded data of ourselves doing four different exercises, including sit-ups, push-ups, planks, and squats. In total, we gathered 18,778 videos and 55,008 images. Each video will be processed using Open Pose to retrieve the series of key points needed to evaluate each exercise repetition, while each image will be processed to retrieve key points to detect and classify exercise based on the user's pose.

The process of our model is as follows:

1 Detect the user that raises their right hand (accuracy 90%).
2 Detect exercise type based on the user pose (accuracy 90%).
3 Track the repetitions done by the user (accuracy 90%).
4 Display the tracking results.

This paper contributes to:

1 Collecting many datasets in push-ups, sit-ups, squats, and planks (video, images, features extracted using Open Pose).
2 More accurate human activity recognition model by using the combination of spatiotemporal features extraction using Open Pose, LSTM for sequence classification, and MLP for human activity recognition.

Related work

Human Action Recognition (HAR) is a field of study concerned with recognizing an action performed by a person based on sensor data. HAR detects human activities such as sitting down, running, raising the right hand, and other activities based on the features retrieved from our postures. Pose estimation has become a focal point in approaches for HAR. Modern pose estimation approaches are adequate for real-world applications with use-cases ranging from usage for sports and education to biomechanics and medication [16].

Collecting pose estimation data can be done using a variety of inputs such as wearable inertial sensors and visual sensors, with each type of sensor having its own set of challenges. For example, wearable sensors would be effective for specific types of exercise, but sensors attached to less ideal positions will affect the tracking results [14]. There is also a possibility that magnetic disturbances or measurement noise can affect the input from low-end inertial sensors [18]. High-end marker-based motion capturing systems can overcome these shortcomings, but these sensors' lack of availability makes them impractical for everyday use. Limitations in using wearable inertial sensors would also be the lack of publicly available data sets to compare performance between solutions, mainly because each solution has differing algorithms for the analysis of raw IMU data [17].

Alternatively, visual sensors have different limitations compared to wearable sensors. State-of-the-art solutions have overcome the difficulties of computer vision tasks such as object detection because of deep neural networks [13]. However, processing input from visual sensors is still difficult for embedded solutions because it requires a large number of computational resources required to do so [18]. Furthermore, visual sensors do not offer the same amount of ubiquity compared to wearable sensors because visual sensors are limited by occlusion. Visual sensors need to be pointed directly at the target object, making it difficult to classify exercises such as running. However, visual sensors have an advantage in utilizing modern deep-learning-based pose estimation techniques, making it possible to recognize human activity by analyzing multiple body parts at once. In some segments, modern deep-learning-based pose estimation is comparable to 3D

motion capturing systems, making visual sensors a viable option for stationary exercise monitoring solutions. Some studies have also attempted to combine the two, yielding better results [19].

Our current use case requires us to involve a sequential classifier of monocular images. Thus we would need to consider various types of human body models when considering pose estimation techniques, mainly between skeleton-based models [15] and 3D shaped models [20]. The skeleton-based model is a tree-like structure that connects adjacent joints to create key points, while the 3D models create a 3D model representing the target object [16]. For example, skeleton-based approaches have been used to analyze athletes and create quantitative assessments of their performance, while 3D models approaches are used to create a 3D reconstruction of an ongoing soccer game by analyzing the depth map of the players using convolutional neural networks (CNN).

In the study by Dua et al. [15], a multi-stage CNN is used to estimate the pose of the human body in real-time and succeeded in estimating the poses of many people in a crowded environment using a multi-stage CNN. The approach is made using the bottom-up method, where estimation is done by detecting body parts first and then combining them to form a single body. The results show that the number of people in the video does not impact the performance or frame rate resulting in [15] being one of the best state-of-the-art pose estimation techniques. Open Pose, a 2D skeletal pose estimation model [21], is based on [15]. Another pose estimation study was carried out by Angelini et al. [22] that improved upon the shortcomings of the previous attempt by [15] using Long Short Term Memory (LSTM). This study combines the use of recurrent neural networks (RNN), CNN, and LSTM. The results of utilizing LSTMs significantly increased the performance and efficiency of the model. In addition, LSTMs could utilize temporal information better, thus beating previous state-of-the-art pose estimation models.

Modern pose estimation algorithms that can perform real-time enabled the increase in human activity recognition research. For example, a study by another researcher created a highly generalized model to classify similar human gestures and actions in real-time using Open Pose, and the model can adapt to proximity and viewpoint changes with a high accuracy rate of 99.04%.

Additionally, another study done by another researcher trained a residual bidirectional LSTM model to classify human gestures (including raising left hand, trunk back), action (including running, jumping, sitting), and behavior (including drinking, sleeping, and typing) using data from the Public Domain UCI Dataset and the Opportunity dataset collected using wearable inertial sensors. The model managed to achieve high accuracy of 93.6%, but the authors state that the model performs better with different parameters for different datasets. Nevertheless, the two studies have similarities in creating a highly generalized model capable of adapting to data.

Another study conducted by another researcher used OpenPose to classify skeletal structures. The research is to compare ballet poses using various methods. The feature extraction methods used include Histogram of Gradients (HOG), SIFT, SURF, and Open Pose. The classification methods applied are K-Nearest Neighbors, Naive Bayes, Decision Trees, Random Forest, Gradient Boosting Tree, Support Vector Machine, Artificial Neural Network, and Recurrent Neural Networks. The final result achieves an accuracy

Fig. 1 Sample images dataset

Table 1 Multiclass pose datasets

Exercise class	Image counts	Video counts
Plank	552	6245
Push Up	542	5587
Sit Up	588	4157
Squat	522	2789

of 99.375%. The accuracy is achieved by a combination of Open Pose and Random Forest.

Similar research was also conducted by another researcher. This study aims to classify four poses, namely standing, sitting, lying down, and dangerous sitting poses. The classification method used is multilayer perceptron and managed to achieve the best results with an F-Score of 92.5% for one of the cases. Subsequently, a study by Rushil et al. [14] aims to create a model that can classify sports movements capable of calculating the repetitions performed with people.

Dataset

The datasets used in this research are videos consisting of four exercises (push up, plank, sit up, and squat) from different angles and three different subjects, and videos of three different subjects raising their right hand from different angles. Figure 1 illustrates the example of the images collected and Table 1 demonstrates the detail of the dataset collected.

Right hand up neural network dataset

Right-hand-up video datasets are pre-processed into images per frame and grouped into true and false labels. The true label means that the subject is currently raising their right hand, while false means otherwise. Each frame has been labelled, processed into keypoints data using the OpenPose library, and fed through the neural network.

Multi-class pose network dataset

Exercise video datasets are pre-processed into images per frame and grouped into multiclass labels of each exercise. Each frame has been labeled, processed into keypoints data using the OpenPose library, and fed through the neural network.

Starting pose network dataset

Exercise video datasets are pre-processed into images per frame and grouped into multi-class labels of each exercise. Each frame has been labeled, processed into keypoints data using the OpenPose library, and fed through the neural network.

Exercise pose evaluation dataset

Exercise video datasets are split into single repetitions and labeled based on each exercise class. Next, each repetition is processed into arrays of key points, forming a pattern for each exercise repetition. These pre-processed datasets are then fed to LSTM RNN to study keypoints patterns of each exercise class. The main features used in this research for pose tracking are key points. Keypoints are representations of human pose in the form of joints and skeletons. For example, 14 joints consist of nose, neck, left and right shoulder, left and right elbow, left and right wrists, waist, left and right eyes, left and right ears for a single set of key points. Pre-processed keypoints collection will be labeled based on each exercise group.

Proposed methodology

Data retrieval

The author decided at the outset to use four types of exercise in this experiment. The types of exercises are planks, sit-ups, push-ups, and squats. For the plank, the subject holds the position for 30 s times four sets. For push-ups, the target does 15 reps times four sets. For sit-ups, 15 repetitions times four sets, and for squats, 15 repetitions times four sets. Figure 2 shows the process of the video recording for the dataset. The data collection process starts recording a video where the author does 1 set of exercise movements per video. The video was recorded using a smartphone camera with 1280x720 pixels and 30 frames per second (FPS). The camera is about 2 ms away from the subject. The angle of the video is taken so that all or at least most of the body parts are visible. This is done so that the resulting data have as much information as possible about the positions of the limbs. Thus, the hope is that the patterns of exercise movements can be

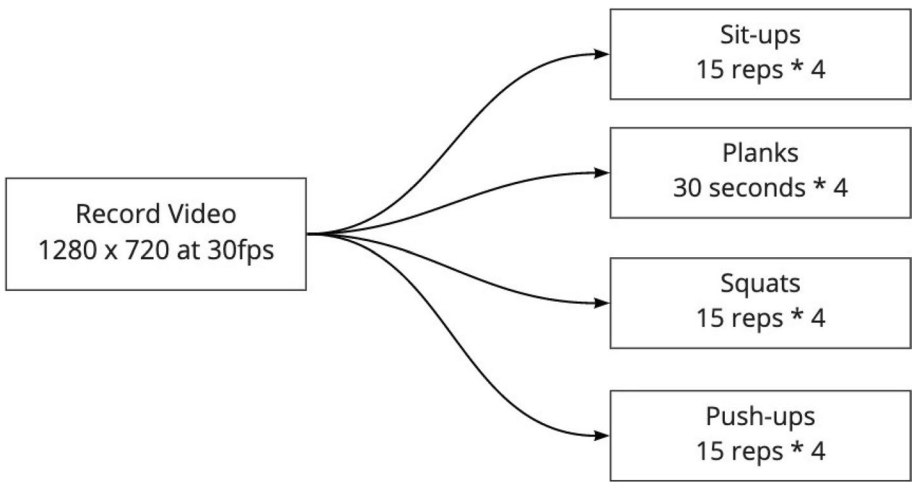

Fig. 2 Data collection

more visible in the data so that our neural network model can learn from the data more easily.

Apart from exercise movement videos, the author also recorded videos where the subject raises his right hand. This was necessary because we needed to train a model to detect a subject holding his right hand up. This detection process will become a trigger that tells the system to classify the exercise by looking at the starting pose. This process will be explained in more detail in the model development chapter.

Preprocessing data

Before the data can be used, the data needs to be pre-processed. First, the resolution of each video is reduced to 720x404 pixels to reduce the size of the data. Furthermore, FPS is also standardized to 24 FPS to have the same number of FPS across all videos. By doing this, the raw video that was 30-50 megabytes in size becomes 1-3 megabytes. Next, each exercise video is cut into shorter videos of 1 exercise movement repetition. This cutting process is done manually by looking at the subject's initial position and ending position in the video. Then, the author determines the duration of the repetition video for each type of exercise. For example, for push-ups, the author determines the duration to be 1 s per repetition, while for sit-ups and squats, the author determines the duration for 2 s per repetition. The duration of the videos is determined based on the observation that sit-ups and squats have an average duration of 2 s while push-ups have 1. Finally, for the plank type exercise, because it is different from other types of exercise that aim to count the number of repetitions, the plank is an exercise to maintain the position for a set amount of time. So the author decided to process the video of 1 plank repetition into a duration of 1 s.

The author carried out a mirror process for all exercise repetition videos to add variation to the data, so the amount of data is doubled. The videos will be used as the data source for the LSTM-based exercise evaluation model. The videos are further processed by cutting each frame into images which will be used as the data source for our binary classification model. The images are organized into three main folders: the exercise starting pose, right hand up, and exercise pose. Inside the exercise starts, the pose folder is images of the subjects in the starting pose of each exercise. The images are separated into four folders for each type of exercise, namely planks, push-ups, sit-ups, and squats. In each exercise folder, the data is further divided into two folders named positive and negative. The positive folder contains data containing images of subjects in the starting position of the related exercise. In contrast, the negative folder contains images where the subject is not in the starting position of the exercise.

In the right-hand up folder, the data is divided into a positive folder and a negative folder. The exercise poses folder contains video data that have been processed previously. In this folder, the videos are separated into four folders according to the type of exercise. The organized data will now be further processed to extract the key points of the subject's body joints. This process is done by using several libraries. First, an open Pose is used to detect the position of key points. The position of the obtained key points is then normalized to be relative to the bounding box of the target. This is done so that the position of the key points does not depend on the subject's position relative to the image box. For example, in 2 different images, wherein image A, the position of the subject

is at the top left of the image, and an image B, the subject is at the bottom right of the image, but the pose of the subject is the same, the keypoint values in both images will be the same. After that, the keypoint numbers are further normalized using the MinMax Scaler from the sci-kit learn library. They result in the keypoint numbers having a value between 0 and 1.

The image extraction process will produce data that is the spatial representation of the subject's joints in the image. In code, this spatial representation is an array with 14 pairs of keypoint position coordinates in the picture. The body parts stored are the neck, right and left shoulders, right and left elbows, right and left wrists, waist, right and left thighs, right and left knees, and right and left ankles. For the right hand up image data, the number of key points taken is 13 pairs where the body parts stored are the nose, neck, right and left shoulders, right and left elbows, right and left wrists, waist, right and left eyes, and right and left ears. For video data, the extraction process will produce spatiotemporal data that represents the movement of the joints of the subject's body from 1 frame to the next in 1 video, where the joints have taken are the same as those taken in the exercise image data.

Model development

Figures 3, 4 and 5 illustrate the proposed models to train the datasets. In the application, we use several deep neural network models. One model is used to detect the subject raising his hand, another to classify the type of exercise, three models to classify the starting and ending position of each exercise, and four models to evaluate the movement of each exercise. The model that detects the raising of the hand uses a feedforward neural network. This model has an input layer with a one-dimensional array of 26 neurons corresponding to 13 pairs of key points in the upper body joints. The output of this model is a number from 0 to 1, which represents the confidence model that the subject is raising his right hand.

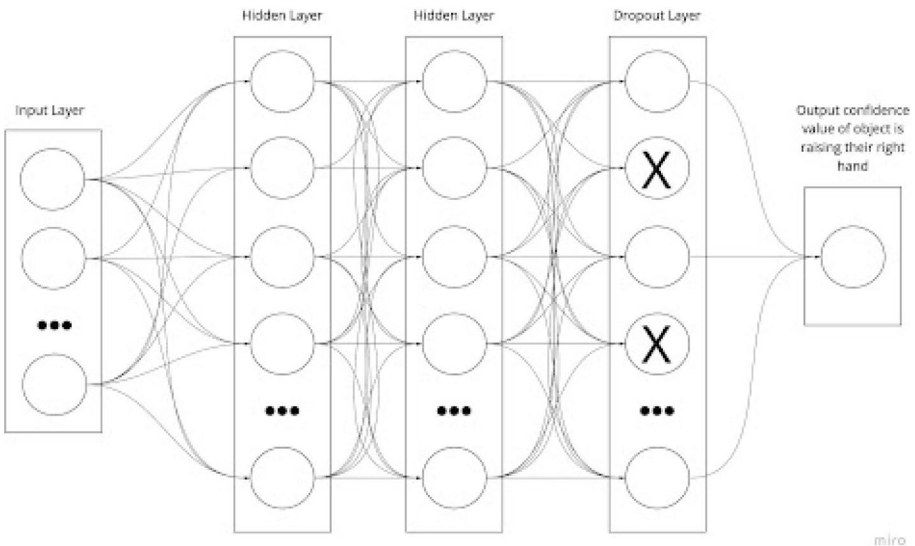

Fig. 3 Neural network to predict subject is raising their right hand up

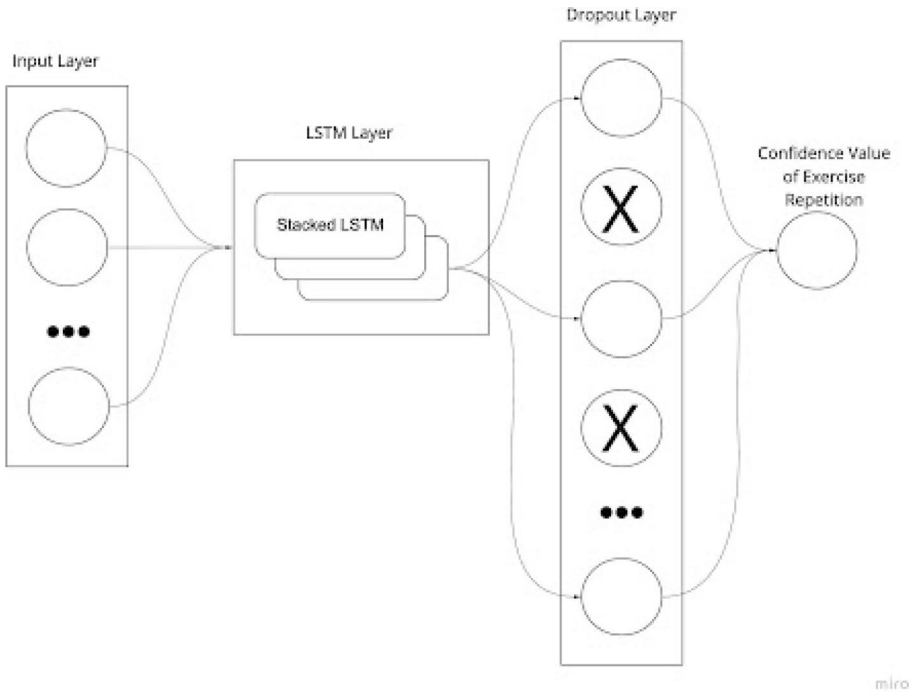

Fig. 4 LSTM network to predict confidence of each repetition

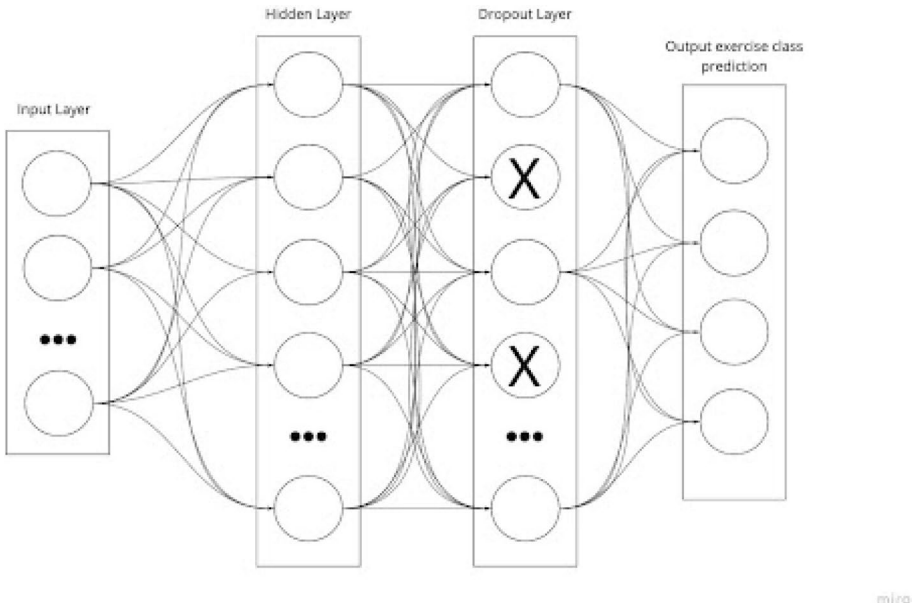

Fig. 5 Neural network to predict Initial Pose Exercise Class

The exercise starting position classifier model also uses a feedforward neural network. This model has an input layer with a one-dimensional array of 28 neurons to receive 14 pairs of key points from all over the body. In addition, this model has several outputs according to the number of exercise classes available. For the exercise movement evaluation model, the architecture used combines a feedforward neural network and LSTM. To

predict an exercise's start/end movement, we use a feedforward neural network, while to evaluate if the movement is correct, we use LSTM. The model for classifying the initial position of exercise movements also has a standard feedforward neural network architecture. This model has an input layer with a one-dimensional array of 28 neurons to receive 14 pairs of key points from all over the body. In addition, this model has several outputs according to the number of exercise classes available. For LSTM, the input layer of this model is in the form of a two-dimensional array of 24x28, where 24 is the number of frames in a video data, and 28 is the number of 14 pairs of flattened key points. The number of frames is different for each exercise, so the model is 48x28. For example, 48 is the number of frames of sit-ups and squats. The output of this model is a floating-point number between 0 and 1, representing the confidence of the model that the subject has done one repetition of exercise movement correctly. We applied k-fold cross-validation for data validation to validate our models, and we will use a 9:1 comparison of training and test data.

Application development

At this stage, the author will create a simple application to demonstrate the use of the deep learning models. The application will combine the models that have been made previously. The input given in the exercise evaluation application system is a video of the subject doing an exercise. This video will then be processed using the Open Pose application to retrieve key points from the subject detected. These key points are features that will be inputted into the ANN and LSTM models. Finally, the model will classify the exercise type from the subject's pose and evaluate it based on these features. An overview of how the application works are illustrated in image 4.n. First, the application will wait until it detects that the subject's right hand is up. The right-hand detection will trigger the next step of classifying the exercise type. The subject needs to hold the starting position of the type of exercise they are doing. Once the exercise type is classified, the subject can start doing the exercise, and the application will evaluate the exercise movements.

Right hand up detection

The image is first processed through the OpenPose application to retrieve key points used as input. Next, key points are normalized using the MinMax scaler before inputting the right hand up detection model. Once a right hand up pose is detected, the application will start exercise type classification.

Exercise type classification

The image is first processed through the OpenPose application to retrieve key points used as input. Next, key points are normalized using the MinMax scaler and fed to the initial pose detection model. Then the application saves the target exercise type result.

Exercise evaluation with LSTM

Before processing the image, the application loads the LSTM and binary classification model according to the saved exercise type, and then the image is cropped to fit the target. OpenPose then processes the image to retrieve key points that will be used as input.

Finally, key points are normalized using the MinMax scaler. Exercise evaluation is carried out for each repetition. If the target is detected to be in an exercise starting position, the system starts the collection of target key points. When the end of the exercise repetition is detected, and the number of key points collected is at least 12, these are used as input for the LSTM model to be evaluated. After detecting the ending position of repetition, the system will start collecting the target key points again until it finds the ending position of the next repetition. This is done repeatedly until the application is turned off.

Result and analysis

In this experiment, we trained three different models: binary pose model, multiclass initial pose model, and exercise pose evaluation model. The physical resources required to run the model must be considered, such as the need for a GPU to carry out training and testing. For informational purposes of assessing the feasibility and practicality of this approach we ran the experiment with the following system specifications: CPU AMD Ryzen 5 2600 six-core processor; 16 GB of RAM; GPU GeForce GTX 1070 TI.

Binary pose model

For this model, we tried 24 combinations of hyperparameters. The parameters that change in the combination set are the number of layers and batch size. For model validation, we used a 10-fold cross-validation method. Table 2 shows the hyperparameters that are constant in each experiment. Table 2 shows that the learning rate used for all binary classification models is 0.01. The first has 60 hidden units, followed by 30 hidden units in the following layer. The hidden layers use ReLu as its activation function, and the output layer uses the Sigmoid activation function. Hyperparameters that changed were the batch size (25, 50 and 100) and the number of layer (2 or 3).

There are four models to classify push ups, sit ups, squats, and raising your right hand. The first three models were created to classify when to start or when to end reps for a specific exercise, while the last model is used to activate the evaluation. Each model has a total of 24 combinations of hyperparameters, and the time required to train the model depended heavily on the batch size, the time needed to train a model with a batch size of 25 is 30 s on average, if the batch size is 50 it takes 20 s on average, and lastly a batch size of 100 takes less than 15 s. Running the three batch size experiment will take 65 s or just above one minute.

Taking this into account, in a 24 hyperparameter combination the three batch sizes will be re-run eight times. Which will make training the entire model combinations

Table 2 Default parameters

Learning rate	0.01
Layer 1 units	60
Layer 2 units	30
Dropout	0.2
Activation function in hidden layer	ReLu
Activation function in output layer	Sigmoid
Optimizer	SGD
Loss	Binary cross entropy

take about 8 min, and because the authors used a 10-fold cross validation method, the total time required to validate each model will be around 80 min in training time. Multiply this again by the four models needed to be tested, it will need 5 h and 20 min of training time to train and validate all four models.

Push up

Table 3 and Fig. 6 show the results of training the binary push up the model using various hyperparameters. The best training results for the binary push up the model are obtained when training the model with three layers, 60 hidden units for the first layer and 30 for the second and third layers, 200 epochs and a batch size of 25.

Push-up Binary Model Result Analysis Based on the training results in Table 3, the model with more layers has lower loss and validation loss. Although the accuracy recorded for the two types of models is not very different. Nevertheless, of the parameters that can be changed, batch size seems to be the most influential. Because the amount of loss in a small batch is better than the others, it can be seen that a batch of 25 will produce a better model than when using a batch of 50.

Table 3 Comparison of the results of the push up binary pose classification training model with a dropout of 0.2

Num hidden	Epoch	Batch size	Loss	acc	val_loss	val acc
60-30	100	25	0.0587	0.98264	0.04792	0.98618
		50	0.09735	0.96974	0.09267	0.96909
		100	0.17356	0.94043	0.17115	0.93746
	150	25	0.04427	0.98721	0.04282	0.98637
		50	0.06578	0.98026	0.05794	0.98182
		100	0.12741	0.95897	0.11659	0.96317
	200	25	0.04083	0.98804	0.03648	0.98989
		50	0.05565	0.9834	0.05025	0.98387
		100	0.09736	0.97028	0.09396	0.96882
	250	25	0.03624	0.98954	0.05726	0.978
		50	0.04975	0.98522	0.30758	0.92977
		100	0.08149	0.97494	0.14809	0.94564
60-30-30	100	25	0.0357	0.98908	0.03915	0.98868
		50	0.04612	0.98648	0.04237	0.98628
		100	0.08286	0.97546	0.08802	0.97153
	150	25	0.02802	0.99164	0.05762	0.97976
		50	0.03618	0.9894	0.0382	0.98728
		100	0.05257	0.98468	0.10352	0.97337
	200	**25**	**0.02515**	**0.99221**	**0.02576**	**0.99202**
		50	0.0311	0.99108	0.0341	0.9909
		100	0.04356	0.98722	0.04356	0.98692
	250	25	0.02131	0.99339	0.14881	0.97884
		50	0.02743	0.99197	0.557	0.9317
		100	0.03704	0.9889	0.65989	0.92376

The bold value means the best result or value

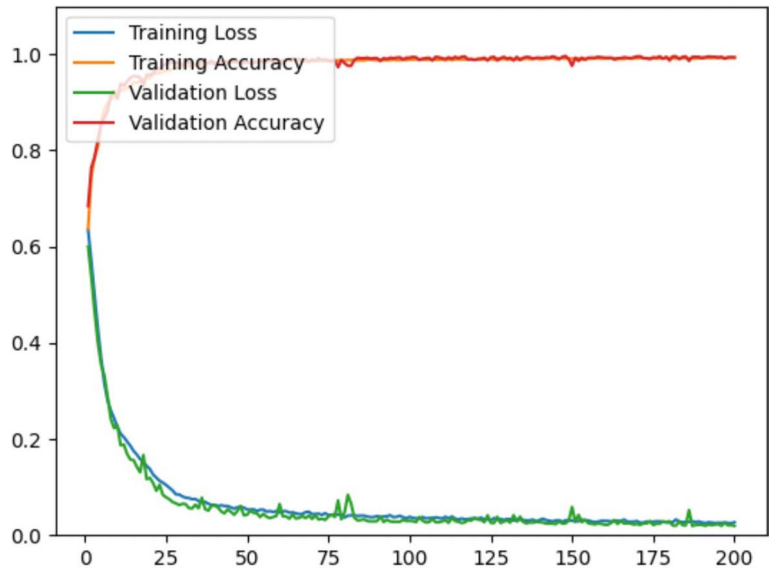

Fig. 6 Best result for Push-up model

Squat

Table 4 and Fig. 7 illustrate the training results of the binary squat model. The best training results for the binary squat model were obtained when training the model with three layers, 60 hidden units for the first layer and 30 for the second and third layers, 200 epochs and 25 batch sizes.

Squat Binary Model Result Analysis Based on the training results in Table 4, the differences between the hyperparameter models do not significantly affect the results. The difference in loss can be seen, and the validation loss for each variation is not too different, with a difference of only 0.01. Likewise accuracy, and validation accuracy which has a difference of not more than 0.002. The training results do not seem to be too affected by the batch size because the difference in loss and accuracy between batch sizes is minimal compared to the push-up model (Table 5).

Sit up

Table 6 and Fig. 8 shows the results of training the binary sit up model. The best training results for the binary squat model were obtained when training the model with three layers, 60 hidden units for the first layer and 30 for the second and third layers, 200 epochs and 100 batch sizes.

Sit-up Binary Model Result Analysis Based on the training results in Table 6, the differences between the hyperparameter models do not significantly affect the results. The difference in loss can be seen, and the validation loss for each variation is not too different, with only 0.003. The training results do not affect the number of layers and batch size because the difference in loss and accuracy is minimal.

Table 4 Comparison of results of the squat binary pose classification training model with a dropout of 0.2

Num hidden	Epoch	Batch size	Loss	acc	val_loss	val acc
60-30	100	25	0.02693	0.99153	0.02568	0.99184
		50	0.029	0.99143	0.02687	0.99203
		100	0.0357	0.99104	0.03203	0.99239
	150	25	0.02449	0.99202	0.02351	0.99203
		50	0.02712	0.99133	0.02497	0.99294
		100	0.03165	0.99095	0.02636	0.99258
	200	25	0.02323	0.99219	0.02138	0.99268
		50	0.02596	0.99157	0.02383	0.99194
		100	0.02979	0.99124	0.02563	0.99184
	250	25	0.02195	0.9926	0.02264	0.99183
		50	0.02492	0.99182	0.02369	0.99149
		100	0.0278	0.99128	0.02504	0.99166
60-30-30	100	25	0.02417	0.99203	0.02353	0.99258
		50	0.02679	0.99126	0.0272	0.99159
		100	0.02737	0.99151	0.0276	0.99075
	150	25	0.02067	0.99286	0.0245	0.99204
		50	0.02295	0.9923	0.024	0.99248
		100	0.02653	0.99141	0.02424	0.9922
	200	**25**	**0.01846**	**0.99337**	**0.01944**	**0.99257**
		50	0.02184	0.99241	0.02136	0.99221
		100	0.02381	0.99194	0.03085	0.98781
	250	25	0.01806	0.99348	0.01969	0.99201
		50	0.02219	0.99253	0.02173	0.99211
		100	0.02386	0.9921	0.0218	0.99193

The bold value means the best result or value

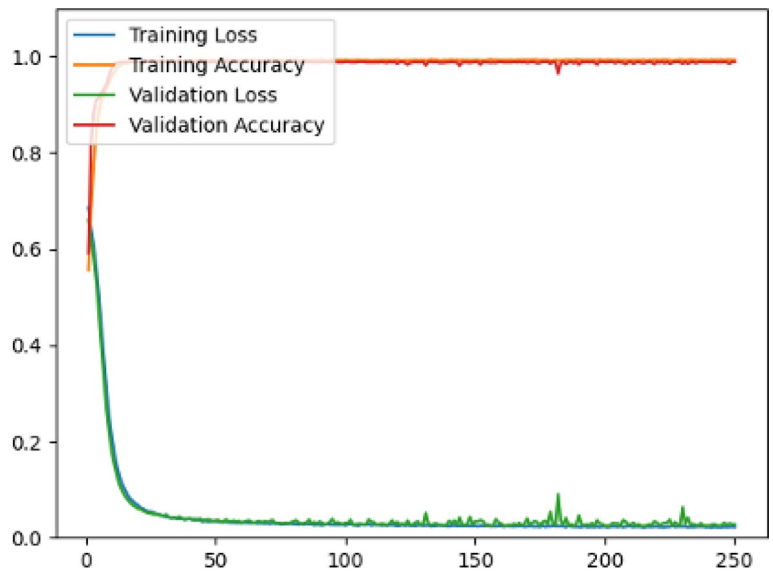

Fig. 7 Best result for Squat model

Table 5 Comparison of the training results of the situp binary pose classification model with a dropout of 0.2

Num hidden	Epoch	Batch size	Loss	acc	val_loss	val acc
60-30	100	25	0.0045	0.99913	0.00419	0.9992
		50	0.00633	0.99878	0.00469	0.9991
		100	0.00941	0.99813	0.00621	0.999
	150	25	0.00352	0.99927	0.004306	0.9992
		50	0.00484	0.99904	0.00403	0.9992
		100	0.00701	0.99869	0.00481	0.9992
	200	25	0.00374	0.99922	0.003591	0.9993
		50	0.00455	0.99914	0.003834	0.9993
		100	0.00592	0.99904	0.00465	0.9992
	250	25	0.00334	0.99928	0.004052	0.9993
		50	0.00397	0.99915	0.003678	0.9992
		100	0.00549	0.9989	0.00425	0.9993
60-30-30	100	25	0.0036	0.99921	0.003995	0.9992
		50	0.00436	0.99918	0.004155	0.9992
		100	0.00592	0.99893	0.00484	0.9991
	150	25	0.0033	0.99917	0.004548	0.9992
		50	0.00382	0.99922	0.004019	0.9992
		100	0.005	0.99906	0.00385	0.9922
	200	25	0.00287	0.99932	0.004134	0.9992
		50	0.00359	0.99919	0.004166	0.9993
		100	0.00421	0.99917	0.0036	0.9993
	250	25	0.00272	0.9994	0.004316	0.9991
		50	0.00321	0.9993	0.004514	0.9991
		100	**0.00392**	**0.99924**	**0.003558**	**0.9992**

The bold value means the best result or value

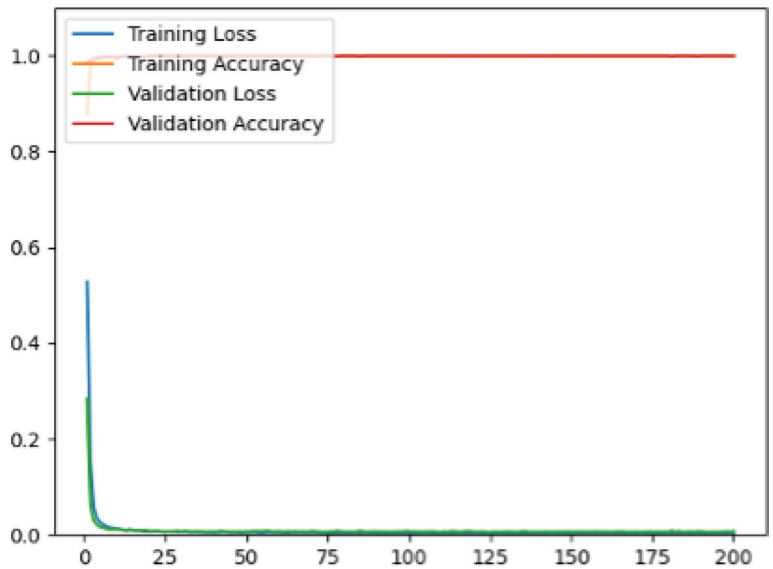

Fig. 8 Best result for Sit-up model

Table 6 Comparison of training results for the right hand up binary pose classification model with a dropout of 0.2

Num hidden	Epoch	Batch size	Loss	acc	val_loss	val acc
60-30	100	25	0.03478	0.98841	0.03227	0.98873
		50	0.04416	0.98377	0.04062	0.98454
		100	0.06071	0.97872	0.05534	0.97977
	150	25	0.03107	0.99001	0.02983	0.98982
		50	0.03822	0.98676	0.03517	0.98732
		100	0.04945	0.98135	0.0458	0.98145
	200	25	0.02751	0.9915	0.02637	0.99215
		50	0.0354	0.98833	0.03256	0.98842
		100	0.0437	0.98402	0.04051	0.98434
	250	25	0.02497	0.99216	0.02334	0.99316
		50	0.03314	0.9888	0.03081	0.98945
		100	0.04107	0.98488	0.03765	0.98573
60-30-30	100	25	0.02206	0.99315	0.02278	0.99228
		50	0.03094	0.98934	0.03057	0.98979
		100	0.04322	0.9834	0.04174	0.98302
	150	25	0.01305	0.9963	0.01605	0.99494
		50	0.02475	0.99216	0.02531	0.9913
		100	0.03485	0.9879	0.034	0.987
	200	25	0.01044	0.99708	0.0123	0.9966
		50	0.01983	0.99381	0.01972	0.99363
		100	0.03055	0.98969	0.02911	0.98998
	250	**25**	**0.00642**	**0.99831**	**0.009**	**0.99744**
		50	0.01481	0.99586	0.0148	0.99576
		50	0.02623	0.99159	0.02651	0.99114

The bold value means the best result or value

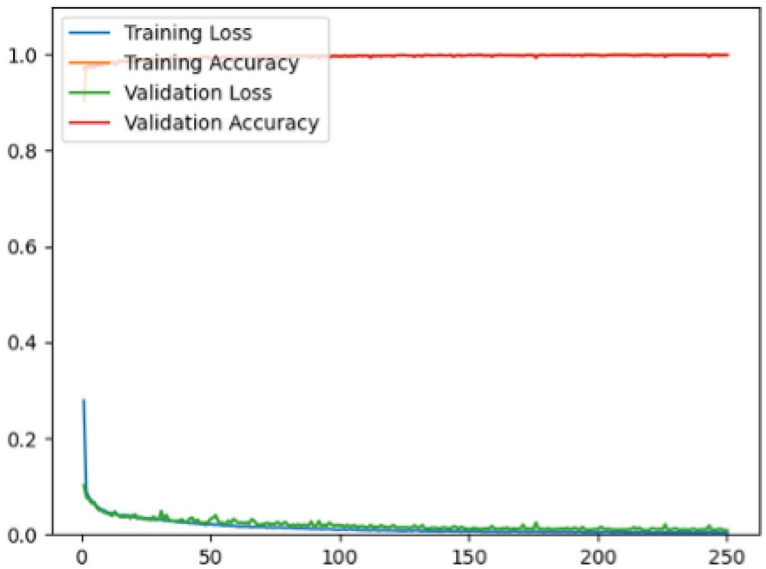

Fig. 9 Best result for Right Hand Up model

Right Hand Up

Table 6 and Fig. 9 shows the results of training the binary right hand up model. The training process is carried out with various hyperparameters. The best training results for the binary right hand up model were obtained when training the model with three layers, 60 hidden units for the first layer and 30 for the second and third layers, 250 epochs and a batch size of 10.

Right Hand Up Binary Model Result Analysis Based on the training results in Table 6, the differences between the hyperparameter models do not significantly affect the results. The difference in loss can be seen, and the validation loss for each variation is not too different, with a difference of only 0.03. Likewise, accuracy and validation accuracy which has a difference of not more than 0.02. It looks again like the push-up model results that a batch of 25 will produce a better model than when using a batch size of 50 or 100.

Multi class pose model

In this experiment, we tried 24 hyperparameter combinations. The parameters that will change for this model are the number of layers and batch sizes. Meanwhile, for model validation, the author uses the 10-fold cross-validation. Table 7 shows that the learning rate for all multi-pose classification models is 0.01. The first has 60 hidden units, followed by 30 in the following layer. The hidden layers use ReLu as its activation function, and the output layer uses the Softmax activation function. Hyperparameters that changed were the batch size (10, 25, and 50) and the number of layers (2 and 3).

Multi class pose model result analysis Table 8 shows the ten best training results from the multi-pose evaluation model. Based on the table, it can be seen that a model with two layers, dropout of 0.2, trained with a batch size of 10, and epoch of 150 results in more accurate models than any other combination.

Figure 10 show the best training result. From the beginning it can be seen that the amount of loss has dropped dramatically until the 20th epoch. But it has started to fall slowly starting in the 40th epoch. It seems that the number of training iterations is not too influential after the 80th epoch, because based on image loss and accuracy it is quite stable when it is at the 100th epoch.

However, the difference between the ten best results is not too significant. Compared to other studies that have tried to classify multi-class poses using neural networks, the

Table 7 Default Parameters

Learning rate	0.01
Layer 1 units	60
Layer 2 units	30
Dropout	0.2
Activation function in hidden layer	ReLu
Activation Function in output layer	Softmax
Optimizer	SGD
Loss	Sparse categorical cross entropy

Table 8 Comparison of the top 10 training results for the multi-pose classification model with a dropout of 0.2

Num hidden	Epoch	Batch size	Loss	acc	val_loss	val acc
60-30	**150**	**10**	**0.00767**	**0.99824**	**0.002364**	**0.99965**
	250	10	0.00572	0.98841	0.002396	0.9993
	200	25	0.01329	0.99753	0.00377	0.9993
	200	10	0.00699	0.99849	0.0045	0.9993
	250	25	0.01049	0.99783	0.005348	0.9993
60-30-30	250	25	0.00555	0.99888	0.003508	0.99965
	100	25	0.01365	0.99723	0.003809	0.99965
	200	50	0.01392	0.99741	0.003996	0.99965
	250	50	0.0109	0.99782	0.004064	0.9993
	150	25	0.0092	0.99833	0.004189	0.9993

The bold value means the best result or value

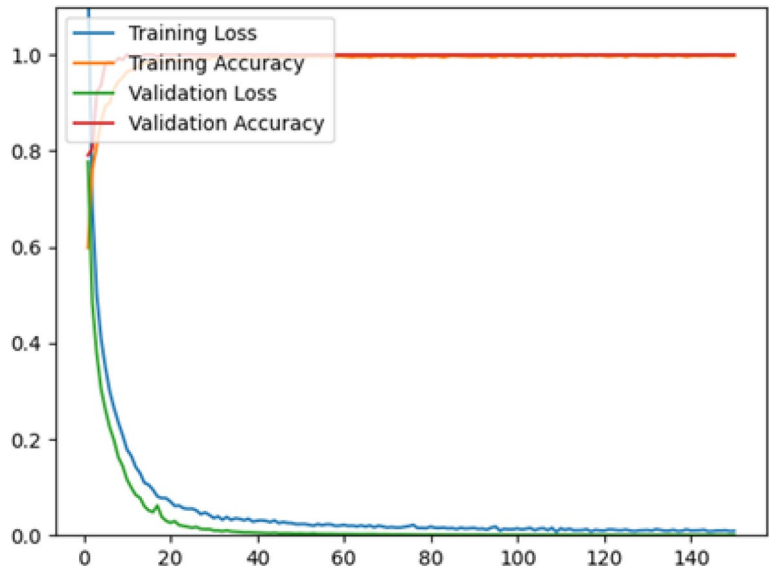

Fig. 10 Best result for Multi-Class Pose Classification Model

best accuracy result obtained was 94.5% compared to the current model accuracy of high 99.965% in the authors' trial [20]. Compared to previous research feed-forward architecture, the results obtained in this experiment are quite state of the art.

LSTM pose evaluation model

In this experiment we tried 54 combinations of hyperparameters. The parameters that will change includes the number of layers (2, 3, and 4), the hidden units (11, 22, and 44), the batch size (100, 150, and 200) and the size of the dropout layer (0.3 and 0.5). Meanwhile, in Table 9 there is a list of hyperparameters that will not change during the training process.

It can be seen that the learning rate used for all models is 0.01. In the hidden layer, the activation function used is ReLu with a Sigmoid output layer. The output layer implements the Sigmoid function because this model requires a result between 0 and

Table 9 Default parameters

Learning rate	0.01
Decaying learning rate	Polynomial Decay
Decay steps	10
End learning rate	0.00001
Activation function in hidden layer	ReLu
Activation function in output layer	Sigmoid
Optimizer	Adam
Loss	Binary cross entropy

Table 10 Comparison of the top 10 training results for the LSTM Push Up Evaluation Model

Layers	num_hidden	epoch	batch size	dropout	loss	acc	val_loss	val_acc
2	**44**	**450**	**150**	**0.3**	**0.00986**	**0.9997**	**0.0125**	**0.99946**
	44	450	100	0.3	0.00838	0.99974	0.01255	0.99901
	22	450	100	0.5	0.02386	0.99449	0.02544	0.99622
3	44	450	200	0.3	0.01089	0.99969	0.01804	0.99865
	22	418	150	0.5	0.02557	0.99564	0.02318	0.99766
	44	450	100	0.3	0.00958	0.9997	0.02507	0.99829
4	44	397	150	0.3	0.01058	0.99967	0.01537	0.99865
	44	278	200	0.5	0.01257	0.99886	0.01925	0.99802
	22	450	150	0.3	0.01716	0.99777	0.02183	0.9974
	22	450	100	0.3	0.01274	0.99861	0.02359	0.99748

The bold value means the best result or value

1 for both classes. We applied Adam as the optimizer. For model validation we used the 10-fold cross validation method.

There are four models used to evaluate push ups, sit ups, planks, and squats. Each model has a total of 54 combinations of hyperparameters. To train one model it needed an average of 1000 s of training time. Training all 54 combinations of hyperparameters will take 54.000 s, which is about 15 h of training time. Multiply this again by the four models needed to be trained, the model will need 60 h of training time to train all four models.

Push up evaluation model

Table 10 shows the 10 best training results from the LSTM Push Up Evaluation Model. The training process is carried out with all the different combinations of hyperparameters. The best training results for the LSTM Push Up Evaluation Model were obtained when training a model with 2 LSTM layers, with each layer having 44 hidden units, 450 epochs, 0.3 dropouts, and 150 batch sizes.

Figure 11 shows the best training results. From the beginning it can be seen that the amount of loss has dropped dramatically from the beginning to the 150th epoch. But it has started to fall slowly starting in the 250th epoch. It looks like the number of training iterations doesn't really affect it after the 300th epoch, because based on image loss and accuracy it's quite stable at the 320th epoch.

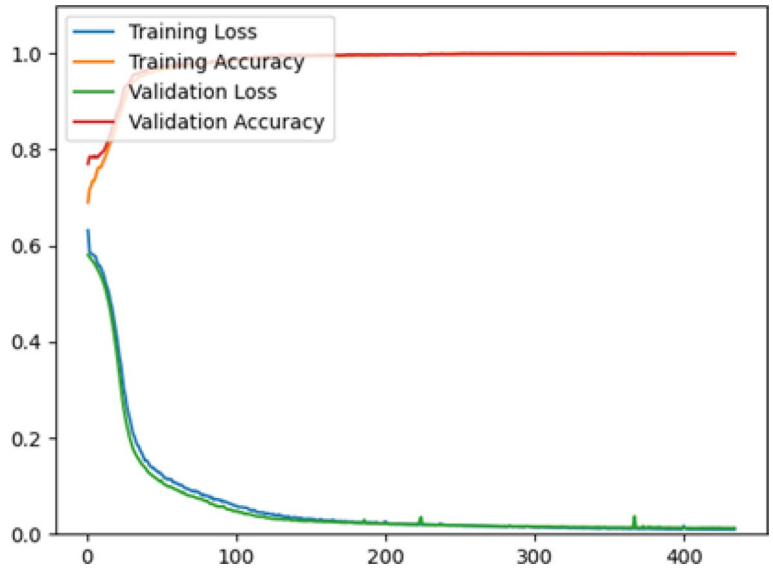

Fig. 11 Best result for LSTM Push Up Evaluation Model

Table 11 Comparison of the top 10 training results for the LSTM Sit Up Evaluation Model

Layers	Num_hidden	Epoch	Batch size	Dropout	Loss	acc	val_loss	val_acc
2	**11**	**450**	**150**	**0.3**	**0.09677**	**0.96768**	**0.06187**	**0.99423**
	22	450	200	0.5	0.12769	0.952	0.08339	0.97944
	11	450	150	0.5	0.13232	0.9477	0.09096	0.99676
	22	450	150	0.3	0.10537	0.95237	0.11266	0.9583
	11	450	100	0.3	0.23065	0.94526	0.11823	0.9738
	11	450	200	0.3	0.16388	0.92437	0.12841	0.95109
3	11	450	150	0.3	0.12415	0.96964	0.09807	0.98424
	11	450	100	0.5	0.20543	0.91332	0.12648	0.9637
4	11	450	100	0.3	0.119	0.95866	0.09478	0.96695
	11	450	200	0.3	0.15941	0.948	0.09746	0.96419

The bold value means the best result or value

Push Up Evaluation Model Result Analysis Based on the training results in Table 10, the model with 2 layers has lower loss and validation loss. The best results apply the largest hidden unit with a total of 44 in each layer. The number of hidden units greatly affects the performance of the model because it can be seen that of the 30 best hyperparameter combinations, only 4 models have 11 hidden units. These 4 models are also in the last 9th position. To summarize the results of the experiment it seems that batch size, the number of layers, and the size of the dropouts are not seen as significant determinants compared to hidden units.

Sit up evaluation model

Table 11 shows the top 10 best training results from the lstm sit up evaluation model. The training process is carried out with different combinations of hyperparameters. The author tries 2 dropout values of 0.3 and 0.5. There are 3 types of batch sizes 100,

150 and 200. For the layers tested, there are 2, 3 and 4 layers. The layers that are tried always have hidden units of 11, 22, and 44 units.

The best training results for the lstm sit up evaluation model were obtained when training a model with 2 layers of LSTM, with each layer having 11 hidden units, 450 epochs, 0.3 dropouts, and 150 batch sizes.

Figure 12 shows the best training results. From the beginning it can be seen that the amount of loss has dropped dramatically from the beginning to the 100th epoch. But it has started to fall slowly starting at the 200th epoch.

Sit Up Evaluation Model Result Analysis Based on the training results in Table 11 the model with 2 layers has lower loss and validation loss. The best results apply the smallest hidden unit with a total of 11 in each layer. The number of hidden units greatly affects the performance of the model, because it can be seen that of the 30 best hyperparameter combinations, only 2 models have 44 hidden units and these models are also in the last 9th position.

Batch size, number of layers, and dropouts are not seen as significant determinants compared to hidden units. The number of hidden units in the best sit up model is inversely proportional to the results of the push up model. But the least number of layers still gives more optimal results.

Plank evaluation model

Table 12 shows the 30 best training results from the lstm push up evaluation model. The training process is carried out with different combinations of hyperparameters. The author tries 2 dropout values of 0.3 and 0.5. There are 3 types of batch sizes 100, 150 and 200. For the layers tested, there are 2, 3 and 4 layers. The layers that are tried always have hidden units of 11, 22, and 44 units.

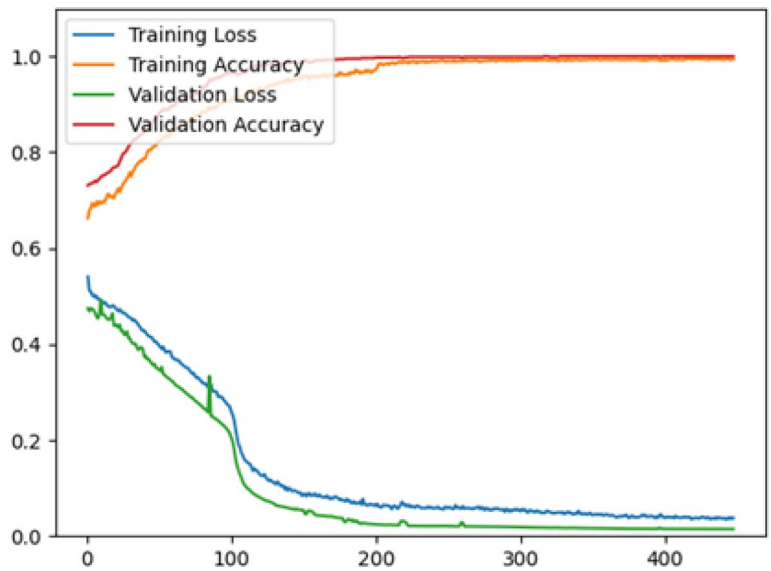

Fig. 12 Best result for LSTM Sit Up Evaluation Model

Table 12 Comparison of the top 10 training results for the LSTM Plank Evaluation Model

Layers	Num_hidden	Epoch	Batch size	Dropout	Loss	acc	val_loss	val_acc
2	44	450	200	0.3	0.00994	0.99967	0.01261	0.99944
	22	399	150	0.5	0.01883	0.99734	0.01444	0.99936
	44	450	150	0.3	0.01019	0.99945	0.01608	0.99936
	22	214	150	0.3	0.01312	0.99919	0.01656	0.99928
3	**44**	**450**	**200**	**0.3**	**0.00922**	**0.99967**	**0.01218**	**0.99936**
	44	450	100	0.3	0.00683	0.99986	0.01291	0.99944
	44	450	200	0.5	0.01036	0.9995	0.01575	0.99928
	22	390	100	0.3	0.01127	0.99938	0.01659	0.99896
	44	445	100	0.5	0.00809	0.99967	0.01706	0.99936
4	44	450	150	0.5	0.00811	0.99979	0.01632	0.99952

The bold value means the best result or value

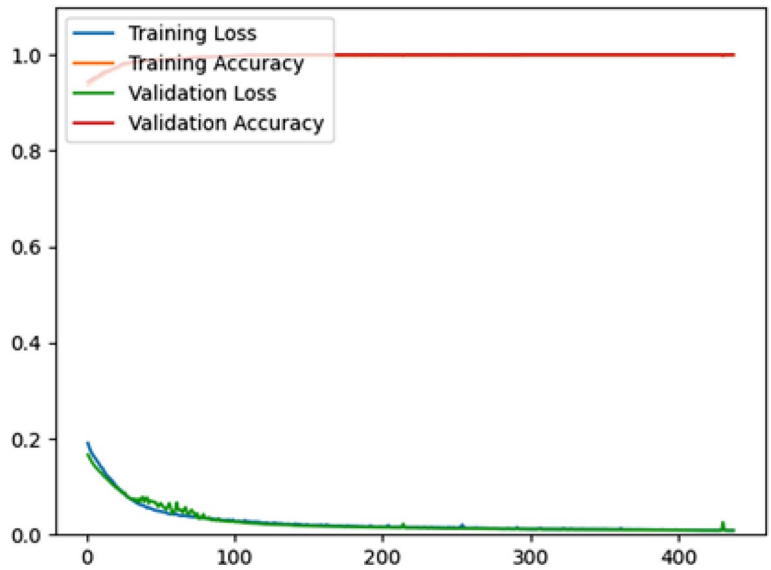

Fig. 13 Best result for LSTM Plank Evaluation Model

The best training results for the lstm push up evaluation model were obtained when training a model with 3 LSTM layers, with each layer having 44 hidden units, 450 epochs, 0.3 dropouts, and 200 batch sizes.

Figure 13 shows the best training results. From the beginning it can be seen that the amount of loss has dropped very drastically from the beginning to the 50th epoch. But it has started to fall slowly starting at the 100th epoch. It looks like the number of training iterations doesn't really affect it after the 200th epoch, because based on image loss and accuracy it's quite stable at the 220th epoch.

Plank evaluation model result analysis Based on the training results in Table 12, the model with 3 layers has lower loss and validation loss. The best results apply the largest hidden unit with a total of 44 in each layer. The number of hidden units greatly affects the performance of the model because it can be seen that of the 30 best hyperparameter combinations, only 4 models have 11 hidden units.

Table 13 Comparison of the top 10 training results for the LSTM Squat Evaluation Model

Layers	Num_hidden	Epoch	Batch size	Dropout	Loss	acc	val_loss	val_acc
2	**11**	**450**	**150**	**0.3**	**0.12197**	**0.97698**	**0.09153**	**0.99479**
	11	450	100	0.3	0.16882	0.95291	0.11122	0.98135
	11	450	200	0.5	0.27013	0.8656	0.14923	0.93986
	22	450	200	0.3	0.56749	0.92771	0.47996	0.95082
3	11	450	150	0.3	0.21074	0.93615	0.21212	0.96326
	11	450	100	0.3	0.20653	0.95861	0.24605	0.97328
4	22	450	100	0.3	0.18659	0.89956	0.13617	0.93564
	44	450	200	0.5	0.19491	0.9698	0.20188	0.98745
	44	450	150	0.3	0.5213	0.92851	0.22831	0.97973
	44	450	100	0.5	0.64791	0.93383	0.39157	0.96737

The bold value means the best result or value

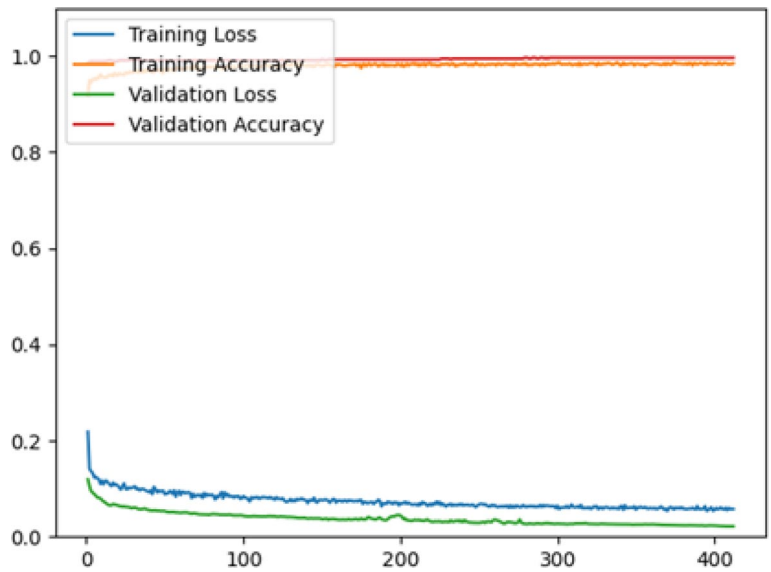

Fig. 14 Best result for LSTM Squat Evaluation Model

Batch size, number of layers, and dropouts are not seen as significant determinants compared to hidden units. The results of the plank model seem to be similar to the push up model where the number of hidden units with the best results has a value of 44. This is probably because the size of the input layer push up and plank models is similar, in contrast to the size of the input layer sit up model.

Squat evaluation model

Table 13 shows the 30 best training results from the lstm push up evaluation model. The training process is carried out with different combinations of hyperparameters. The author tries 2 dropout values of 0.3 and 0.5. There are 3 types of batch sizes 100, 150 and 200. For the layers tested, there are 2, 3 and 4 layers. The layers that are tried always have hidden units of 11, 22, and 44 units.

The best training results for the lstm push up evaluation model were obtained when training a model with 2 LSTM layers, with each layer having 11 hidden units, 450 epochs, 0.3 dropouts, and a batch size of 150.

Figure 14 shows the best training results. From the beginning it can be seen that the amount of loss has dropped dramatically from the beginning to the 100th epoch. But it has started to fall slowly starting at the 200th epoch. It looks like the number of training iterations doesn't really affect it after the 200th epoch, because based on image loss and accuracy it's quite stable at the 220th epoch.

Squat evaluation model result analysis Based on the training results in Table 13, the model with 2 layers has lower loss and validation loss. The best results apply the smallest hidden unit with a total of 11 in each layer. The number of hidden units greatly affects the performance of the model, because it can be seen from the 30 best hyperparameter combinations that no model has 44 hidden units.

Batch size, number of layers, and dropouts are not seen as significant determinants compared to hidden units. The results of the squat model resemble the results of the sit up model. But inversely proportional to the push-up or plank model. The difference is in the different number of input layers where the squat and sit up models require 48 inputs, while the push up and plank models require 24 inputs.

Accuracy comparisons

Figure 15 shows the confusion matrix for the initial pose detector model. The total data used for testing the initial pose detector model is 2821, where 706 for plank initial pose data, 694 for the push-up pose, 753 for the sit-up pose, and 668 for squat. Based on the confusion matrix, the model successfully classified all the data. Figure 16 shows the confusion matrix result for the Binary Pose Models. The push-up binary pose model resulted in a very small error of 0.83% for the not pushing up class and 0.21% for push up. For the sit-up binary pose model, the result shows an error of 0.48% in detecting the

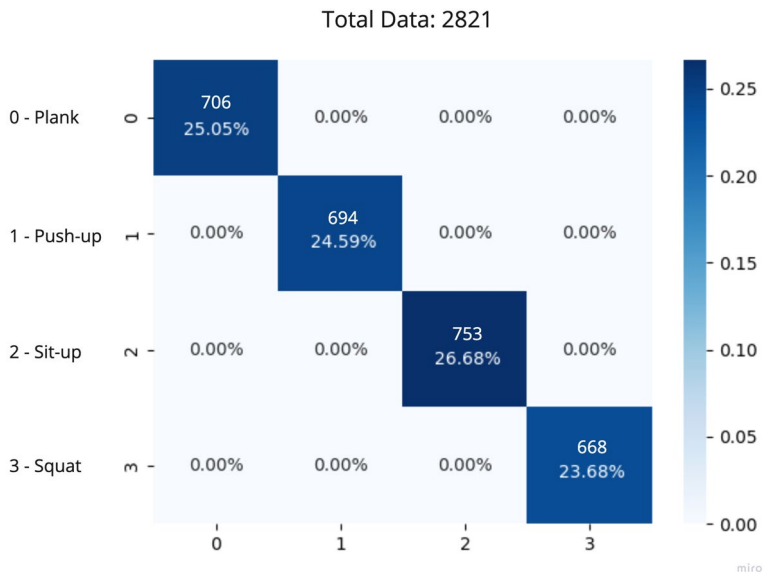

Fig. 15 Initial Pose Detector Confusion Matrix

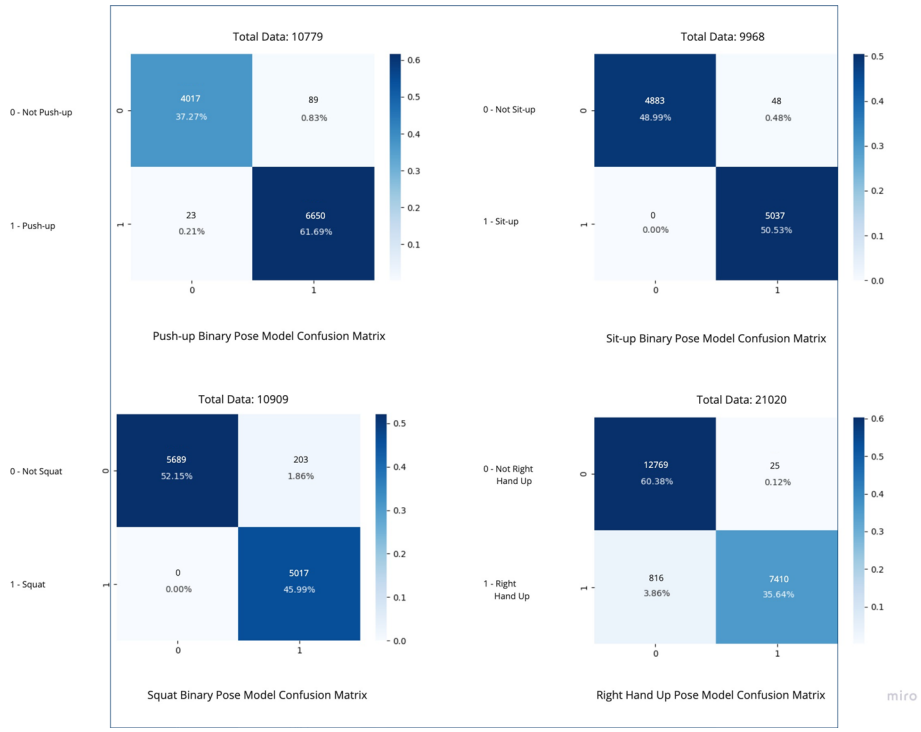

Fig. 16 Binary Pose Confusion Matrix

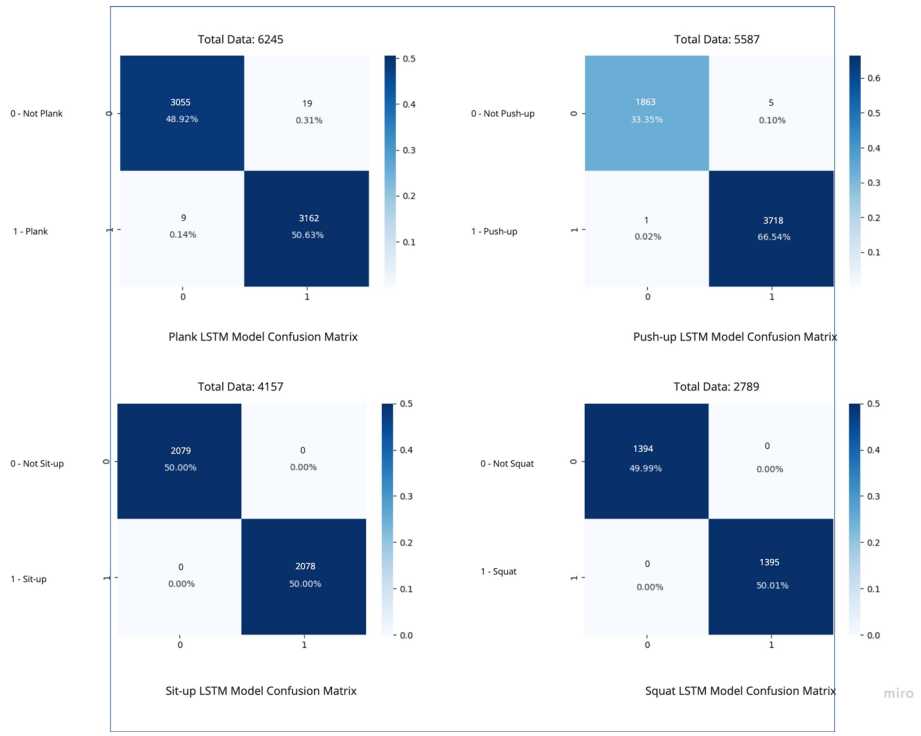

Fig. 17 LSTM Model Confusion Matrix

not-push-up class and the correct result for the rest of the prediction. The squat binary pose model produced an error of 1.86% in detecting the not-squat-class, and the right-hand-up binary pose model resulted in an error of 0.12% in predicting the not-right-hand-up class, and 3.86% error for a not-right-hand-up.

Finally, Fig. 17 shows the confusion matrix result for the LSTM evaluation model. For the LSTM plank evaluation model, the result shows an error in classifying not plank of 0.31% and an error of 0.14% in classifying plank. The push-up LSTM model resulted in an error of 0.10% in classifying not push-up and 0.02% classifying push-up. The sit-up LSTM model had no error, and the squat LSTM model did not produce any error.

Conclusion and future work

This research contributes to the data collection and deep learning exploration to model pose estimation in several physical excersice activities. The model proposed achieved a minimum accuracy of 90% to detect the user that raises their right hand, detect exercise type based on the user pose, and track the repetitions done by the user. Moreover, the research contributes to datasets collection in push-ups, sit-ups, squats, and planks as well as highly accurate human activity recognition model by using the combination of spatiotemporal features extraction using Open Pose, LSTM for sequence classification, and MLP for human activity recognition. The tasks in this research highly implementing multiple neural network models to identify the subject, classify the initial pose, and predict exercise repetitions using exercise key points work great for counting exercise repetitions based on visual-based detection. With proper object detection, LSTM is the right approach to analyze each exercise class repetitions patterns using a key-point-based dataset. Combined with spatiotemporal features extracted using open pose, the trained models could identify basic human activity recognition. These neural network models can be improved further using higher quantity and quality datasets.

Author contributions
All authors contribute equally.

Funding
There is no funding available for this research.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no Competing interests.

Received: 6 April 2023 Accepted: 7 April 2024
Published online: 03 August 2024

References

1. Roque FR, Briones AM, et al. Aerobic exercise reduces oxidative stress and improves vascular changes of small mesenteric and coronary arteries in hypertension. Br J Pharmacol. 2012;168:686.

2. Jeong H-C, So W-Y. Difficulties of online physical education classes in middle and high school and an efficient operation plan to address them. Int J Environ Res Public Health. 2020;17:9. https://doi.org/10.3390/ijerph17197279.
3. Gray S, Finch C. The causes of injuries sustained at fitness facilities presenting to victorian emergency departments - identifying the main culprits. Injury Epidemiol. 2015. https://doi.org/10.1186/s40621-015-0037-4.
4. Bratland-Sanda S, Mathisen TF, Sundgot-Borgen C, Sundgot-Borgen J, Tangen JO. The impact of covid-19 pandemic lockdown during spring 2020 on personal trainers' working and living conditions. Front Sports Active Living. 2020;2:201. https://doi.org/10.3389/fspor.2020.589702.
5. Kaur H, Singh T, Arya YK, Mittal S. Physical fitness and exercise during the covid-19 pandemic: a qualitative enquiry. Front Psychol. 2020;11:2943. https://doi.org/10.3389/fpsyg.2020.590172.
6. Bravata D, Smith-Spangler C, Sundaram V, Gienger A, Lin N, Lewis R, Stave C, Olkin I, Sirard J. Using pedometers to increase physical activity and improve health: a systematic review. JAMA J Am Med Assoc. 2007;298:2296–304. https://doi.org/10.1001/jama.298.19.2296.
7. Nelson R, Hayes S. Theoretical explanations for reactivity in self-monitoring. Behav Modif. 1981;5:3–14. https://doi.org/10.1177/014544558151001.
8. Morris D, Saponas T, Guillory A, Kelner I. Recofit: using a wearable sensor to find, recognize, and count repetitive exercises. Conf Hum Factors Comput Syst Proc. 2014. https://doi.org/10.1145/2556288.2557116.
9. Chen W, Yu C, Tu C, Lyu Z, Tang J, Ou S, Fu Y, Xue Z. A survey on hand pose estimation with wearable sensors and computer-vision-based methods. Sensors. 2020;2:4. https://doi.org/10.3390/s20041074.
10. Seel T, Kok M, McGinnis R. Inertial sensors-applications and challenges in a nutshell. Sensors. 2020;20:6221. https://doi.org/10.3390/s20216221.
11. Bruno B, Mastrogiovanni F, Sgorbissa A. Wearable inertial sensors: Applications, challenges, and public test benches. Robot Automation Mag IEEE. 2015;22:116–24. https://doi.org/10.1109/MRA.2015.2448279.
12. Nishani E, Cico B. Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation, 2017; https://doi.org/10.1109/MECO.2017.7977207.
13. Cao Z, Hidalgo G, Simon T, Wei S, Sheikh Y. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. CoRR abs/1812.080082018. arXiv:1812.08008.
14. Rushil K, Karan A, et al. Gymcam: detecting, recognizing and tracking simultaneous exercises in unconstrained scenes. ACM Journals. 2018;2.
15. Dua N, Singh S, Semwal V. Multi-input cnn-gru based human activity recognition using wearable sensors. Computing. 2021;103:1–18. https://doi.org/10.1007/s00607-021-00928-8.
16. Fourie M, van der Haar D. Computer Vision for the Ballet Industry: A Comparative Study of Methods for Pose Recognition, 2020; 118-129.
17. Scott J, Collins R, Funk C, Liu Y. 4d model-based spatiotemporal alignment of scripted taiji quan sequences. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017;795–804. https://doi.org/10.1109/ICCVW.2017.99
18. Wang J, Tan S, Zhen X, Xu S, Zheng F, He Z, Shao L. Deep 3d human pose estimation: a review. Comput Vis Image Understand. 2021;210: 103225. https://doi.org/10.1016/j.cviu.2021.103225.
19. Torres JM, Zhao Y, Yang R, Chevalier G, Xu X, Zhang Z. Deep residual bidir-lstm for human activity recognition using wearable sensors. Mathl Probl Eng. 2018;2018:7316954. https://doi.org/10.1155/2018/7316954.
20. Guerra BMV, Ramat S, Gandolfi R, Beltrami G, Schmid M. Skeleton data pre-processing for human pose recognition using neural network*. In: 2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC), 2020;4265–4268. https://doi.org/10.1109/EMBC44109.2020.9175588
21. Vonstad EK, Su X, Vereijken B, Bach K, Nilsen JH. Comparison of a deep learning-based pose estimation system to marker-based and kinect systems in exergaming for balance training. Sensors. 2020;20:23. https://doi.org/10.3390/s20236940.
22. Angelini F, Fu Z, Long Y, Shao L, Naqvi SM. ActionXPose: A novel 2D multi-view pose-based algorithm for real-time human action recognition 2018. arXiv:1810.12126

Publisher's Note