Real-Time Workout Recognition Using Dense Neural Networks, Pose Estimation, and Optical Flow

James Peralta, Jeffrey Boyd
Department of Computer Science
University of Calgary
Calgary, Canada
{james.peralta, jboyd}@ucalgary.ca

***Abstract*—Physical activity has been proven to be effective in the primary and secondary prevention of a variety of chronic diseases including cancer and cardiovascular disease. Although exercise provides many incentives, people have trouble motivating themselves to maintain their routine. Workout loggers are used to track progression, and have been shown to motivate users by showing them their improvements before concrete results were visible in the body. Current workout loggers require users to manually enter data into their phones, or write it down into a notebook. To automate this, we developed a system that uses computer vision to recognize the workouts being performed, and count repetitions in videos. Our system is able to track workouts in real-time using a combination of dense neural networks, pose estimation, and optical flow.**

***Index Terms*—workout recognition, machine learning, neural networks, pose estimation, optical flow**

I. Introduction

In 2017, nearly 50% of deaths in Canada were caused by cancer and heart disease [1]. In fact, 44% of adults over the age of 20 suffer from a chronic disease [2]. To help reduce these chronic diseases, we must invest in preventive measures such as empowering our society into a healthy lifestyle. However, creating the habits to routinely engage in physical activity is a long-term process that requires a lot of personal motivation. Only 20% of people succeed in long-term weight loss maintenance [3].

A promising area of technology - persuasive technology, is designed to influence and change human behaviors. Persuasive technology in the fitness sector has already been shown to improve the health and fitness practices of its users [4]. Within the space of persuasive technology, activity tracking is one of the most prevalent strategies. Users of activity trackers that monitor their workout progress have commented on how this technology has motivated them and helped them make durable changes in their lifestyle [5]. Activity trackers can be used for simpler data, such as counting steps but fail to capture advanced data, such as repetitions of workouts. In this paper, we focus on a specific type of activity tracking, workout logging.

Although workout loggers offer plenty of value to their users, they have shown very poor retention rates. Ledger and McCaffrey [6] found that within the first two weeks, 62% of users that downloaded a workout logging app stopped using it. This was because manual entry of data can be too repetitive and tedious, eventually leading to these technologies to go unused [4]. There have been attempts at automating data entry through wearables [7] [8] [9], computer vision [10], and adding sensors onto equipment [11].

Computer vision is a promising solution because it can capture a richer set of spatial features compared to the other approaches. For example, wearables lose accuracy depending on where the sensor is placed. A sensor placed on the wrist will not detect leg exercises such as a leg press because the hand is stationary during this exercise. Furthermore, sensors on equipment will not be able to track body weight exercises such as push ups where the user is not using any equipment. Not only can computer vision avoid these pitfalls, it is also the only solution that can be implemented as a single-point solution, which will avoid adding trackers to many users or many machines. The research with human participants presented in this paper was reviewed and approved by the University of Calgary Conjoint Faculties Research Ethics Board (CFREB).

II. problem definition

A workout is a routine performed in the gym which consists of several different exercises aimed to train specific groups of muscles. For example, a workout may consist of exercises such as squats, lunges, and leg press which all target the legs. Each exercise is done for a certain amount of repetitions which we refer to as a set. A full repetition is one complete motion of an exercise which we refer to as a rep.

Overtime, a trainee will need to increase the weight they are lifting in order to continue building muscle mass. Typically, if a trainee is able to easily perform a certain amount of sets and reps at a specific weight, they should increase their weight. This requires them to constantly recall their last workouts so that they are able to increase their weight if needed. Many struggle recalling their last workout and use a workout logger to assist them. A simple workout logger will track the number of sets and reps a trainee performs for an exercise for each given day.

Workout logging can be achieved using a pen and paper, or typed into phone apps such as FitNotes [12], Jefit [13], and Gymbook [14]. Since all commercial products require manual entry, we have constructed a software that performs automatic workout logging. We define a workout logger to be completely automated if it is able to detect the type of workout, count the repetitions, identify the amount of weight, and save these results for future use without any user intervention. In our work, we have automated the first two but will address identifying the weight in a future paper.

All computer vision problems have a common set of challenges (see Fig. 1). Although we would have liked to solve most of these challenges, some of these challenges were purposely limited to control the scope of our problem. Due to time constraints introduced from building the dataset ourselves, we have limited the viewpoint variation. Creating a classifier that is viewpoint invariant would have required us to multiply our dataset size by the number of different angles to solve each viewpoint.

Trainees were free to perform the workouts they choose and may have been taught to do the workout in a different way. This has led to intra-class variation in workouts that we did not limit. Other challenges do not present themselves in our gym environment such as scale variation, occlusion, and illumination. This is because we set up the cameras in front of each machine so there was nothing blocking the trainee. Furthermore, trainees do not stand 30 feet away from the squat rack they are using and gyms are usually very well lit for safety.

III. Related Work

A. Using Optical Flow

The current state-of-the-art for automatic workout logging is Gymcam [10]. GymCam uses optical flow to generate a vector of 27 features. This vector is then passed along a pipeline of three separate multilayer perceptrons. This approach was able to obtain an accuracy of 84.6% for exercise segmentation, 93.6% for exercise recognition, and was able to count the number of repetitions within ±1.7 reps. Other related contributions were for similar video analysis tasks; video classification and human gesture recognition.

B. Feature-based versus Neural Networks

Patsadu et al. [15] evaluated feature-based and neural network classifiers for human gesture recognition. In their study, they used a Kinect camera to extract a body-joint position vector containing the locations of 20 joints including the head, hands, and feet. Using this vector as input into a classifier, they classified between three actions; standing, sitting down, and laying down. They observed the performance of multiple classifiers such as neural networks, SVMs, Decision Trees, and Naive Bayes. Out of every classifier, neural networks performed the best.

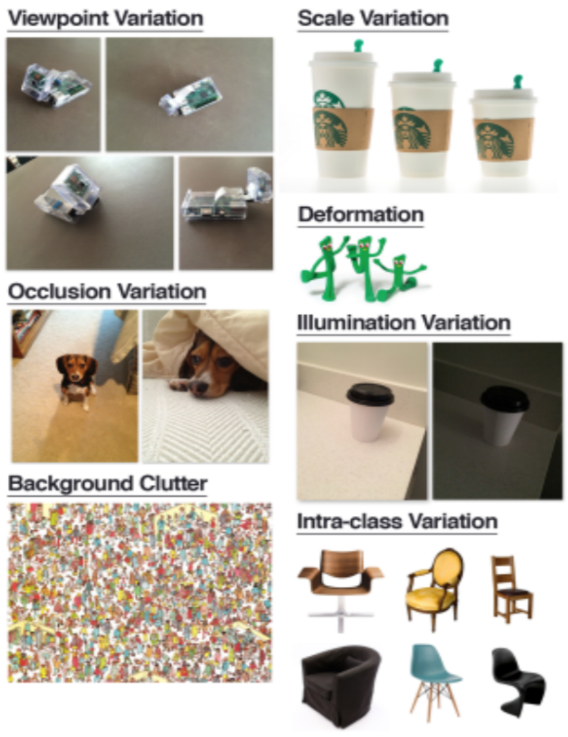

Fig. 1. Illustrations of the different types of variations found in images. Source: Adapted from [16].

C. Using 2D Convolutional Neural Networks

Karpathy et al. surveyed the performance of two dimensional convolutional neural networks (2D CNNs) for video classification [17]. 2D CNNs avoid the feature extraction phase required by Gymcam and feeds raw frames as input into the network. This simplified the pipeline by removing the need to generate custom features and was shown to have significant performance gains over the feature-based approaches. Since 2D CNNs take in a single frame as input, it can only encode spatial features. This can be a problem due to the temporal nature of videos. As part of their contribution, they proposed three ways of encoding temporal information using 2D CNNs (see Fig. 2).

D. Using 3D Convolutional Neural Networks

Tran et al. introduced a three dimensional convolutional neural network (3D CNN) they named, C3D [18]. The 3D CNN takes videos as input and encodes both spatial and temporal information without the need to perform workarounds such as early fusion. The difference between a 2D CNN and a 3D CNN is that 3D CNNs expand their kernels from two dimensions into three dimensions to encode temporal information (see Fig. 3). They went on to show how 3D CNNs were better than 2D CNNs at preserving temporal information and outperformed 2D CNNs on various video analysis tasks. More interestingly, 3D CNNs were shown to run 2 times faster than optical flow solutions.

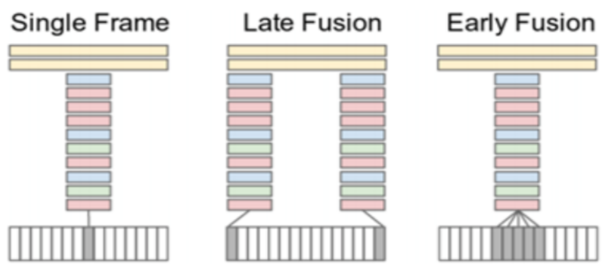

Fig. 2. The single frame approach only encodes information from a single frame, late fusion encodes temporal information from two frames which are 15 frames apart, and the early fusion encodes temporal information from several contiguous frames. Adapted from [19].

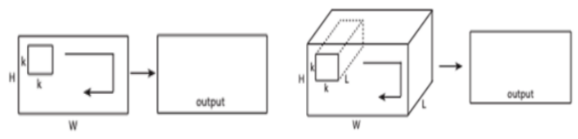

Fig. 3. 2D Convolutions (left) vs. 3D Convolutions (right). Adapted from [19].

IV. Building the dataset

The CFREB required us to adhere to a strict set of policies in order to build this dataset. Some of these policies included the blurring of participant faces, encrypting the dataset when storing them in a file system, and requesting written consent from each participant before collecting their data. We captured the footage using a Logitech C270 HD Webcam which was placed directly in front of the trainee while they performed an exercise. We recorded videos at a resolution of 1280x1080 at 20 frames per second. Our dataset was labelled frame by frame using GymCam's annotation tool [10] which is optimized for labelling workouts and repetitions.

For each clip, we drew a bounding box around a user when they began performing a workout and removed it when they finished the workout. With each bounding box, we labeled the exercise they performed, if they were in the increasing or decreasing phase of their repetition, and the amount of repetitions performed after they completed the exercise. The labeled exercises were used to train our exercise recognition classifier. The increasing and decreasing frames of each repetition were used to evaluate our optical flow algorithm which classified the phase of a user's repetition. If a user was squatting down, we would label these corresponding frames as decreasing and when they began squatting back up, we labeled the frames as increasing. In total, we labelled 15,177 frames.

V. Algorithm

Training an end-to-end system using 2D and 3D CNNs similiar to [17] and [18] requires millions of data points which was not available to us for this task. With the limitation of data set size, it was important to develop a pipeline that was able to capture rich features that would be used as input into our classifiers. Our approach is a pipeline that uses pose estimation, optical flow, and a dense neural network classifier (see Fig. 4) to count repetitions and identify the exercise being performed. The algorithm is as follows:

1) Begin classifying frames in a stream, giving each one a label of overhead press, squats, or nothing.
2) To smooth the output, every official classification is a majority vote of the last ten classifications.
3) An exercise block has begun if the official classification is an exercise. At this point, we begin counting repetitions.
4) We continue making classifications until a users transitions from performing an exercise to performing nothing, this will be marked as the end of the exercise block.
5) Finally, we label this exercise block by using the majority vote of all of the official classifications and assign it the number of repetitions counted.

A. Exercise Recognition

When analyzing an image, we were only interested in the humans in the frame and their positioning. This conveniently aligned to an active area of research called pose estimation. Pose estimation is the task of locating joints (also referred to as key points) such as hands, shoulders, and eyes of a person in an image (see Fig. 5). After locating these joints, pose estimation systems are able to estimate the joint angles which essentially draws a skeleton connecting each joint. There are many pose estimation systems available including Pose Machines [19], Deep Pose [20], and Hr Net [21] but we settled with PoseNet [22] [23] because it is implemented as a package in TensorFlow.

PoseNet uses deep convolutional neural networks (CNNs) to perform pose estimation and returns the x and y coordinates of 17 joints for every person it finds in an image. We reshape the joint positions vector returned by PoseNet into a one dimensional vector using (1).

$$\begin{bmatrix} nose:\ (x_1,\ y_1) \\ left\ eye:\ (x_2,\ y_2) \\ \vdots \\ right\ ankle:\ (x_{17},\ y_{17}) \end{bmatrix} = \begin{bmatrix} x_1,\, y_1,\, x_2,\, y_2,\, \cdots,\, x_{17},\, y_{17} \end{bmatrix} \tag{1}$$

This one dimensional vector is then used as input into our dense neural network (DNN). The DNN contains one hidden layer of six nodes, ReLu as the activation function in the hidden layer, and softmax in the output layer (see Fig. 6). To optimize the weights in the network, we used Adam as our optimizer and categorical cross entropy as our loss function. Given any frame from a video, our system will extract joints using PoseNet and classify the action being performed in the image as either overhead press, squats, or nothing.

B. Repetition counting

Our algorithm for counting repetitions in real-time is built upon the intuition that when a user is performing an exercise,

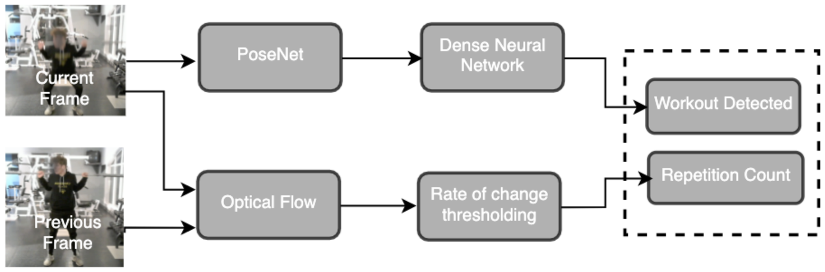

Fig. 4. An overview of the system's architecture. The information originates as raw frames and moves from left to right, continuously being transformed at each stage.

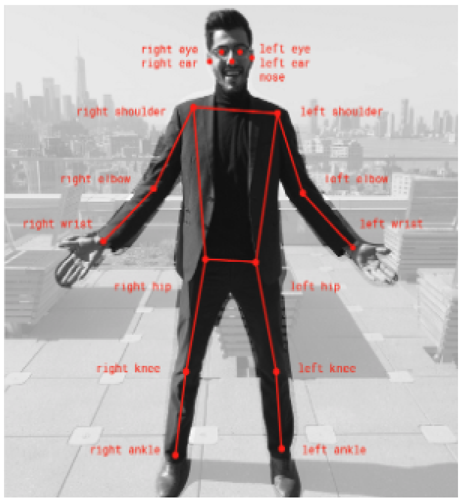

Fig. 5. The 17 key points detected by PoseNet. Adapted from [24].

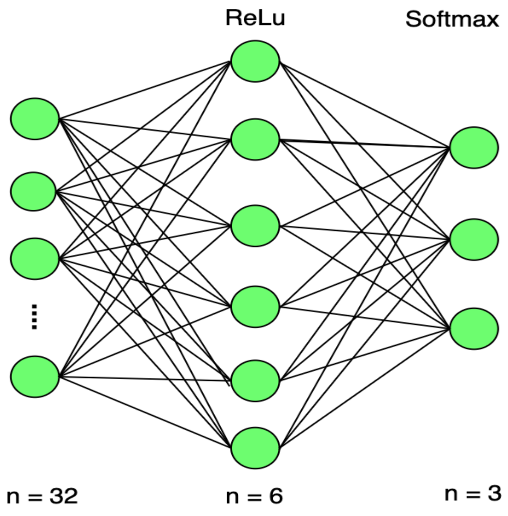

Fig. 6. A dense neural network (DNN) is a network architecture where each neuron in a layer receives an input from all of the neurons in the previous layer. Our DNN contains an input layer of 34 nodes for each of the keypoint coordinates detected by PoseNet and an output layer of 3 nodes for the classification labels overhead press, squats, and nothing.

they will always have at least one body part moving up or down on the y-axis. For example, when a user is performing a squat, they are decreasing their body's y position when squatting down, and then increasing their y position when squatting up. We realized that if we were able to determine when a user is increasing and decreasing their y position, we would be able to detect a repetition by combining increasing-decreasing or decreasing-increasing sequences. In our algorithm, every time an increasing-decreasing or decreasing-increasing sequence is detected, we increment the repetition count.

Determining whether a user is increasing or decreasing in a repetition requires two frames, the current frame (f_t at time t) and the previous frame (f_{t-1} at time t - 1). Both of these frames are converted into gray-scale to reduce the computational power required to process them. Afterwards, we generate a grid of patches (see Fig. 7) for f_{t-1} which we will refer to as P_{t-1}. The density of patches is a tunable parameter called the point stride. If the point stride is set to five, each patch will be five pixels apart both vertically and horizontally. Patches are areas in the image that are tracked from f_{t-1} to f_t. Many of these patches will remain stationary, except for the patches which are located where a trainee is performing an exercise. To estimate the movement of each patch, we use sparse optical flow on P_{t-1} using the Lucas-Kanade method [25]. This generates P_t which is the estimated locations of the P_{t-1} in f_t.

If any patch was unable to be tracked from f_{t-1} to f_t, it will be dropped from both sets P_{t-1} and P_t. We also drop the x coordinates from each patch because we are only interested in each patches movement in the y-axis. Next we generate a displacement vector D by calculating the vertical distance each patch has traveled from f_{t-1} to f_t using (2).

$$P_{\text{t-1}} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{\text{n}} \end{bmatrix} - P_{\text{t}} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{\text{n}} \end{bmatrix} = D \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{\text{n}} \end{bmatrix} \tag{2}$$

We define the displacement sensitivity as the distance a patch has to travel for it to be labelled as increasing or decreasing. We partition D into three disjoint sets which contain the increasing, decreasing, and stationary patches:

TABLE I
TUNABLE PARAMETERS IN THE REPETITION COUNTING COMPONENT.

Parameter	**Description**
Point stride	Vertical and horizontal distance between patches.
Displacement sensitivity	The magnitude of displacement required for a patch to be labelled as increasing or decreasing.
Displacement threshold	The amount of displaced patches required to label the entire frame as increasing or decreasing.

- I_d = $\{d \in D \mid d > displacement\ sensitivity\}$
- D_d = $\{d \in D \mid d < -displacement\ sensitivity\}$
- S_d = $\{d \in D \mid D - I_\mathrm{d} - D_\mathrm{d}\}$

Using these sets, we perform rate of change thresholding using sets I_d, D_d, and S_d along with a displacement threshold parameter. If we find that $|I_\mathrm{d}| > displacement\,threshold$ this frame will be labelled as increasing and vice versa if $|D_\mathrm{d}| > displacement\ threshold$. If we cannot detect this frame as increasing or decreasing it is labelled as stationary.

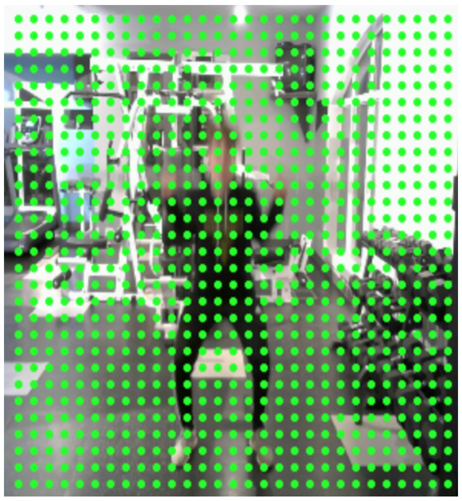
Fig. 7. This figure displays the patches used as input into optical flow as green circles.

VI. Results

A. Exercise Recognition

For this task, the only trainable component was the DNN. We used accuracy as the primary metric to evaluate model performance. We split our dataset into training, validation, and test sets by performing per-video holdout cross validation. Per-video holdout cross validation groups frames that belong to the same video, and places each group into one of the three sets. This ensures that grouped frames from the same video are never found in different sets. For example, there will never be a frame from the same video in both the training and the test set. This reduces the amount of information leaked between the sets. This resulted in a train/validation/test split of 66/21/13 which totalled to 10,004, 3,229, and 1,944 frames in each set respectively.

We ensured that every label was represented to some extent in each dataset with an average of 43% of squat labels, 37% of overhead press labels, and 20% of no action labels in each. We used the training set to train the weights of the network, validation set to tune the hyperparameters, and the test set to produce our final results. Our model achieved 99.5% accuracy on the training set, 98.7% on the validation set, and 98.6% on the test set.

B. Rate of change thresholding

We first evaluated our rate of change thresholding component in order to quantify how well our system was able to locate increasing, decreasing, and stationary phases of a video. This evaluation was important because detecting when a repetition occurs is built upon the ability to correctly identify these phases. This component did not have a learning algorithm involved so we were able to work with a smaller training set and no validation set. We settled with a train/test split of 36/64 which totalled to 5,166 and 9,228 frames in each set. Similar to exercise recognition, we also performed per-video holdout cross validation to reduce the amount of information leaked between the sets.

We ensured that every label was represented to some extent with an average of 44% of stationary labels, 28% of increasing labels, and 28% of decreasing labels in each set. We used the training set to figure out the best parameters for our three tunable parameters (see Table 1). We found that with a point stride of 5, displacement sensitivity of 1.75, and a displacement threshold of 50, we were able to achieve 79% accuracy on the training set and 82% on the test set. We decided that the ability to correctly identify which phase the user was in with 82% accuracy was sufficient to count repetitions due to the fact that we perform a majority vote on the last 10 predictions.

C. Repetition counting

Lastly, we tested the complete repetition counting component by evaluating how well it was able to count the total amount of reps for a given exercise block. Unlike the other evaluations which we were able to do per frame, this evaluation required full videos. We extracted 13 videos from the same test distribution we created in the rate of change thresholding evaluation. This included 5 videos of squats, 6 of overhead press, and 2 of a trainee performing nothing. We were able to achieve an accuracy of ±1.3 for counting reps.

VII. Discussions and limitations

At the time this paper was written, there were no available benchmark datasets for this problem which prevented us from directly comparing against other algorithms. With more data, we would be able to perform a more exhaustive evaluation of the algorithm. In general, we were able to train an exercise recognition system that was able to generalize well on unseen data. The DNN is relatively small in capacity with only one hidden layer of six neurons. This worked the best because with too many neurons and hidden layers, we noticed the DNN would learn all possible mappings of the input and overfit on the training set, decreasing the accuracy in the test set. With more and more data, we would expect to see the DNN perform worse due to its smaller capacity but increasing the neurons or hidden layers should boost the accuracy up back to around 98%. On the other hand, with more data it would be more beneficial to train an end-to-end system that does not require a pose estimation component. Pose estimation was only used as a feature extractor due to our small dataset. With more data, we suspect we will be able to create an end-to-end system that has similar performance without any feature extractors.

Our repetition counting component performed fairly well but we noticed areas where we could improve in the future. We noticed that with too much background clutter, such as other trainees moving around in the background, our optical flow algorithm would detect false positives of increasing and decreasing sequences. To mitigate the effects of background clutter, we could integrate a person detector to detect bounds around a trainee and focus in on them. This will essentially remove the background noise. In future iterations, we also hope to extend our work by increasing the number of workouts we are able to recognize, track multiple user in a video, and also track the amount of weight they have performed.

VIII. Conclusion

In this paper, we introduce a novel algorithm that leverages neural networks, pose estimation, and optical flow to detect workouts and count repetitions in a video. We generated a dataset of users performing overhead press, squats, and doing nothing. Once this data was collected, we labeled each frame with the exercise being performed, and if their movement was stationary, increasing, or decreasing. This data was used to train our dense neural network for classifying exercises, and to determine the parameters for our rate of change thresholding component of the algorithm. Our dense neural network was able to differentiate between workouts with a 98.6% accuracy on our test set. Our repetition counting component was able to detect increasing and decreasing sequences with 82% accuracy and count the repetitions with an error of ±1.3 reps. This algorithm is a great starting point for a system that we hope will one day autonomously track progress for trainees in commercial gyms.

References

[1] Statistics Canada, ”Leading causes of death, total population, by age group,”https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1310039401, vol. , no. , p. , 2017.

[2] Goverment of Canda, ”Prevalence of Chronic Diseases Among Canadian Adults, ”https://www.canada.ca/en/public-health/services/chronic-diseases/prevalence-canadian-adults-infographic-2019.html, vol. , no. , p. , 2019.

[3] R. R. Wing and S. Phelan, ”Long-term weight loss maintenance.,” The American journal of clinical nutrition, vol. 82, no. 1 Suppl, pp. 222S-225S, 2005.

[4] A. Ahtinen, E. Mattila, A. Vaatanen, L. Hynninen, J. Salminen, E. Koskinen and K. Laine,”User experiences of mobile wellness applications in health promotion: User study of wellness diary, mobile coach and SeltRelax,” in 2009 3rd International Conference on Pervasive Computing Technologies for Healthcare - Pervasive Health 2009, PCTHealth 2009, 2009.

[5] T. Fritz, E. M. Huang, G. C. Murphy and T. Zimmermann, ”Persuasive technology in the real world: A study of long-term use of activity sensing devices for fitness,” in Conference on Human Factors in Computing Systems - Proceedings, 2014.

[6] D. Ledger, D.; McCaffrey, ”How The Science of Human Behavior Change Offers The Secret to Long-Term Engagement,”http://endeavourpartners.net/assets/Endeavour-Partners-WearablesWhite-Paper-20141.pdf , 2016.

[7] D. Morris, T. S. Saponas, A. Guillory and I. Kelner, ”RecoFit: Using a wearable sensor to find, recognize, and count repetitive exercises,” in Conference on Human Factors in Computing Systems - Proceedings, 2014.

[8] M. Muehlbauer, G. Bahle and P. Lukowicz, ”What can an arm holster worn smart phone do for activity recognition?,” in Proceedings - International Symposium on Wearable Computers, ISWC, 2011.

[9] F. J. Ordonez and D. Roggen, ”Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition,” Sensors (Switzerland), vol. 16, no. 1, 18 1 2016.

[10] R. Khurana, K. Ahuja, Z. Yu, J. Mankoff, C. Harrison and M. Goel, ”GymCam,” Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 2, no. 4, pp. 1-17, 27 12 2018.

[11] H.Ding,L.Shangguan,Z.Yang,J.Han,Z.Zhou,P.Yang,W.XiandJ. Zhao, ”FEMO: A platform for free-weight exercise monitoring with RFIDs,” in SenSys 2015 - Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, 2015.

[12] Fit Notes, ”FitNotes - Gym Workout Log App”, 2019. [Online]. Available: http://www.fitnotesapp.com/. [Accessed: 8- April- 2019]

[13] Jefit, “Jefit - 1 Gym workout app”, 2020, [Online]. Available: https://www.jefit.com/. [Accessed: 8- April- 2019]

[14] Gymbook, “GymBook - Strength Training”, 2020, [Online]. Available: https://www.gymbookapp.com/. [Accessed: 8- April- 2019]

[15] O. Patsadu, C. Nukoolkit and B. Watanapa, ”Human gesture recognition using Kinect camera,” in JCSSE 2012 - 9th International Joint Conference on Computer Science and Software Engineering, 2012.

[16] A. Rosebrock, Deep Learning for Computer Vision with Python - Starter Bundle, 2017, p. 44.

[17] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and F. F. Li, ”Large-scale video classification with convolutional neural networks,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2014.

[18] D. Tran, L. Bourdev, R. Fergus, L. Torresani and M. Paluri, ”Learning Spatiotemporal Features with 3D Convolutional Networks,” 1 12 2014.

[19] S. Wei, V. Ramakrishna, T. Kanade and Y. Sheikh, ”Convolutional Pose Machines”, http://arxiv.org/abs/1602.00134, 2016

[20] A. Toshev and C. Szegedy, ”DeepPose: Human Pose Estimation via Deep Neural Networks”, http://arxiv.org/abs/1312.4659, 2013

[21] K. Sun, B. Xiao, D. Liu and J. Wang, ”Deep High-Resolution Representation Learning for Human Pose Estimation”, http://arxiv.org/abs/1902.09212, 2019

[22] G. Papandreou, T. Zhu, L. Chen, S. Gidaris, J. Tompson and K. Murphy, ”PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model”, http://arxiv.org/abs/1701.01779, 2018

[23] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler and K. Murphy, ”Towards Accurate Multi-person Pose Estimation in the Wild”, http://arxiv.org/abs/1701.01779, 2017

[24] TensorFlow, “Real-time Human Pose Estimation in the Browser with TensorFlow.js”, 2018, [Online]. Available: https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5. [Accessed: 8- April-2019]

[25] S. N. Tamgade and V. R. Bora, ”Notice of Violation of IEEE Publication Principles: Motion Vector Estimation of Video Image by Pyramidal Implementation of Lucas Kanade Optical Flow,” 2009 Second International Conference on Emerging Trends in Engineering Technology, Nagpur, 2009, pp. 914-917.