INTERNATIONAL JOURNAL OF Telecommunications
AIR DEFENSE COLLEGE

Improving Performance of the Fitness Exercises Repetitions Counter via Computational Complexity Reduction

Youssef Fayad[1,*], Mostafa Mohamed[2], Hossam Reda[3], Khaled Abbas[4] and Mahmoud Mohamed[5]

Citation: Fayad, Y.; Mohamed, M.; Reda, H.; Abbas, K.; Mohamed, M. *International Journal of Telecommunications*, ***IJT'*2021**, *Vol. 01, Issue 01, pp. 1-10, December 2021.* https://ijt-adc.org/articles/2805-3044/279455

Editor-in-Chief: Yasser M. Madany
Received: 30-10-2021
Accepted: 1-12-2021
Published: 18-12-2021

Publisher's Note: The International Journal of Telecommunications, IJT, stays neutral regarding jurisdictional claims in published maps and institutional affiliations.

[1] Air Defense College; dr.youssef.6455.adc@alexu.edu.eg
[2] Air Defense College; mostafa.mostafa@ucalgary.ca
[3] Air Defense College; Hossamreda.9186.adc@alexu.edu.eg
[4] Air Defense College; k.saad1281@gmail.com
[5] Air Defense College; m.mohamed.Abdallah861@gmail.com
* Correspondence: dr.youssef.6455.adc@alexu.edu.eg

Abstract: The COVID-19 precautions had forced us to look for different techniques that enable the continuity of our ordinary life activities, especially the sports ones. In addition, the need for accurate and fast auto judgment techniques to measure physical fitness performance is constantly emerging. The artificial intelligence (AI) with multi-resolution counter had introduced method relays on Artificial Intelligence to realize this purpose, but this method has a high processing time. Modifying the algorithm structure and the inputs features leads to low computational cost. This paper presents a modified algorithm that reduces the computational costs for the optical flow equation This reduction is executed via two techniques; the first one is to execute Gunner Franeback algorithm for number of pixels less than had been used in the previous model via selecting the more weighted pixels that closer to the central pixel, the second one is to employ Model Quantization technique by using Tensor flow Lite as a proposed model. Experimental results indicate that the proposed method has low computational cost, reliable and robust, and can be applied as practical applications. The performance of the experiments was verified by comparing its time complexity with the AI with multi-resolution counter depending on ground truth data.

Keywords: AI, Temporal subspace, Motion tracking, Computer Vision, deep learning; computational complexity.

1. Introduction

Applying deep learning technology in fitness field [1, 2] provides an elegant and accurate method to count the exercise repetitions. But, this luxury comes at the expense of computational complexity, which leads to large processing time. Various researches had been done in the field of fitness measurement tools. In this research, researchers aim to reduce the processing time in order to present an accurate decision in real time as it possible. Some related algorithms such as Gym Cam [3] algorithm detects, recognizes, and tracks exercises via training a neural network using a recorded video. Also, another approach for exercise recognition and repetition counting [4] by Soro, and et al had represented a deep learning via training a neural network using a smart watch as a measurement tool. But these two methods have not the ability of discriminate the correct and incorrect repetitions. A pose trainer algorithm in [5] detects the athlete's body pose and provides a useful feedback about the exercise. But this algorithm depends on fixing the relative position of the person with respect to the camera each time of algorithm execution. Talal Alatiah, Chen Chen [2] introduced an algorithm depends on extracting data from video or an external camera to detect the human poses and extract their skeleton key points (ankles, knees,

elbows and wrists). This method tracks the angles of the major joint. Then, it counts the correct repetitions of the exercise. But the main drawback of the net pose estimation technique is not avoided which is summarized in giving different results with different environments, and physiology specifics despite employing the same model. In another word, the model may not count some correct repetitions because of the big difference between women's and men's body postures during physical training.

This paper introduces a modified hybrid algorithm that applies the polynomial expansion transform of Gunnar Farneback method within temporal subsamples algorithm [1, 6, 7]. This algorithm introduces an extension for our work that represented in [1] to reduce the computational complexity when tracking the athlete's body position within two consecutive frames. The modified algorithm represents a tempo-spatial processing for each captured frame. Then it applies a low resolution Gunnar Farneback method on each tempo-spatial frame layer. This modified algorithm depends on the rational implementation of pixels according to its close from the central pixel which enables the study of the motion between consecutive frames using only the most important pixels from the clique according to its distance from the central pixel. This technique reduces the size of the input array. Also, this algorithm employs Tensor flow Lite platform to optimize the deep learning model memory consumption. These reduction of the under test pixels and Model Quantization techniques reduce the computational complexity of the algorithm.

The rest of the paper is organized as follows: section 2 presents the proposed method and main system design, section 3 shows the experimental results, section 4 introduces discussion and results analysis, section 5 introduces conclusion and section 6 introduces a proposed patent from this system.

2. Proposed Method

2.1. The Gunner Franeback optical flow

Optical flow concept had been introduced by James J. Gibson in 1940s. It provides the pattern of objects apparent motion as a visual scene depending on the relative motion between the scene and the observer. Also, it presents the apparent velocities as the brightness pattern of the captured frame. This method computes the motion between two consecutive frames with time difference Δt. Taylor series is employed to implement partial derivatives w.r.t temporal and spatial coordinates. Each frame is distributed for voxels $L(x, y, t)$, and moves with (Δx, Δy, and Δt) between two consecutive frames. So, for the second frame $L(x + \Delta x, y + \Delta y, t + \Delta t)$. For small movement [6];

$$L(x + \Delta x, y + \Delta y, t + \Delta t) = L(x, y, t) + \frac{\partial L}{\partial x}\Delta x + \frac{\partial L}{\partial y}\Delta y + \frac{\partial L}{\partial t}\Delta t \quad (1)$$

Thus;

$$\frac{\partial L}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial L}{\partial t} = 0 \quad (2)$$

$$\frac{\partial L}{\partial x}V_x + \frac{\partial L}{\partial y}V_y + \frac{\partial L}{\partial t} = 0 \quad (3)$$

Where V_x, V_y are the x, and y represent the optical flow of *L(x, y, t)*, and $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial y}$, and $\frac{\partial L}{\partial t}$ are the derivatives of the voxel in the corresponding directions. Thus,

$$L_x V_x + L_y V_y = -L_t \quad (4)$$

This is an estimation problem with two unknowns.

Lucas-Kanade method in [8-10] is used as differential method to solve the optical flow problem. It assumes that the flow is constant in a local neighborhood of the voxel under processing. For each point (p), all the frame contents are assumed to have an approximately constant displacement w.r.t two nearby frames. So, equation 4 can be assumed to hold for all pixels within a window centered at p. So, the velocity vector will realize;

$$L_x(e_1)V_x + L_y(e_1)V_y = -L_t(e_1)$$

$$L_x(e_2)V_x + L_y(e_2)V_y = -L_t(e_2)$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$L_x(e_m)V_x + L_y(e_m)V_y = -L_t(e_m) \tag{5}$$

Where;

$e_1, e_2, \ldots\ldots, e_m$ are the pixels within window under consideration which its size is (m x m) pixels, and $L_x(e_j)$, $L_y(e_j)$, $L_t(e_j)$ are the partial derivatives of the frame at point (e_j) w.r.t *x, y, t* at the current time. So,

$$\begin{bmatrix} L_x(e_1) & L_y(e_1) \\ L_x(e_2) & L_y(e_2) \\ \vdots & \vdots \\ L_x(e_m) & L_y(e_m) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -L_t(e_1) \\ -L_t(e_2) \\ \vdots \\ -L_t(e_m) \end{bmatrix} \tag{6}$$

$$A = \begin{bmatrix} L_x(e_1) & L_y(e_1) \\ L_x(e_2) & L_y(e_2) \\ \vdots & \vdots \\ L_x(e_m) & L_y(e_m) \end{bmatrix},\ v = \begin{bmatrix} V_x \\ V_y \end{bmatrix},\ b = \begin{bmatrix} -L_t(e_1) \\ -L_t(e_2) \\ \vdots \\ -L_t(e_m) \end{bmatrix}$$

2.2 Time resolution method

To increase the process resolution, algorithm picks up (n) data with small time steps (τ) as a tempo subsampling for pixels enclosed by each window that enables the reduction of pixels clique within the window under consideration due to tempo multiple resolution.

So,

$$b(\tau) = \left[\begin{array}{c|c|c|c} -L_\tau(e_1) & -L_{\tau_{2\tau}}(e_1) & \cdots & -L_{\tau_{(n-1)\tau}}(e_1) \\ -L_\tau(e_2) & -L_{2\tau}(e_2) & \cdots & -L_{\tau_{(n-1)\tau}}(e_2) \\ \vdots & \vdots & \vdots & \vdots \\ -L_\tau(e_m) & -L_{2\tau}(e_m) & \cdots & -L_{\tau_{(n-1)\tau}}(e_m) \end{array}\right] \tag{7}$$

Applying least square for each tempo-spatial layer,

$$A^T A v = A^T b \tag{8}$$

Or;

$$V = (A^T A)^{-1} A^T b \tag{9}$$

Where the transpose of A is A^T, $A^T A$ is called the second-moment matrix of the frame at the point p which provides the predominant directions of the gradient in a specified neighborhood of the point and the degree to which those directions are coherent.

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_j L_x(e_j)^2 & \sum_j L_x(e_j)L_y(e_j) \\ \sum_j L_y(e_j)L_x(e_j) & \sum_j L_y(e_j)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_j L_x(e_j)L_{i\tau}(e_j) \\ -\sum_j L_y(e_j)L_{i\tau}(e_j) \end{bmatrix} \tag{10}$$

j = 1, 2 , ……. , m, and i = 1, 2, ……, n. It is obvious from equations (7, 10) that illumination of the "j^{th}" pixel has "n" values. Whereas, $\sum_i L_{i\tau}(e_j) = L_t(e_j)$. This internal fluctuation for each pixel should be considered in order to smooth the estimation process results when employing a small number of pixel local neighborhoods. These small steps computations provide the algorithm with high accurate results in sync with less inputs array dimension which consequently leads to computational complexity reduction. For that the algorithm gives more weight (*W)* to the nearest pixels w.r.t the central pixel *p,* where (*W)* is usually set to a Gaussian function and it is a diagonal matrix. due to (*W)* the algorithm will look for the solution by employing just one neighborhood pixel from each direction that surrounds the central pixel (p). Therefore, $m = 3$, window size is just (3 x 3) and (*A*) size will be (2 x 3)

Substitute in (8);

$$A^T W A v = A^T W b \tag{11}$$

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_j w_j L_x(e_j)^2 & \sum_j w_j L_x(e_j) L_y(e_j) \\ \sum_j w_j L_y(e_j) L_x(e_j) & \sum_j w_j L_y(e_j)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_j w_j L_x(e_j) L_{i\tau}(e_j) \\ -\sum_j w_j L_y(e_j) L_{i\tau}(e_j) \end{bmatrix} \tag{12}$$

For the additive white Gaussian noise (N),

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_j w_j L_x(e_j)^2 & \sum_j w_j L_x(e_j) L_y(e_j) \\ \sum_j w_j L_y(e_j) L_x(e_j) & \sum_j w_j L_y(e_j)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_j w_j L_x(e_j) L_{i\tau}(e_j) \\ -\sum_j w_j L_y(e_j) L_{i\tau}(e_j) \end{bmatrix} + \sigma^2 \Sigma_N \tag{13}$$

Note, for pixel *j*, the $A^T A$ large eigenvalues satisfy $\lambda_1 \geq \lambda_2 > 0$, which span for the closest pixels to the window center. To realize motion tracking within frames sequence, the flow vector is iteratively applied and recalculated till threshold approach to zero so the image windows are the most similar, so for each successive tracking window the point can be tracked through several frames in a sequence until the point either obscured or goes out of frame.

2.2.1. Tensor flow lite for modeling Deep learning

An artificial neural network is created by using deep learning. So, machine is able to make its own intelligent decisions. First, a large amounts of training data was prepared to compute optical flow with deep neural network. Second, these data was classified and labeled as groups in order to address the motion of frame pixels. Third, a deep learning model was created and trained using Tensor flow. Fourth this model was converted to a tensor flow lite using a python code converter [11, 12]. Finally, a two consecutive video frames (previous (d-1) – next (d)) is taken to compute the Farneback optical flow, so;

$$(V_x, V_y) = f(L_{d-1}, L_d) \tag{14}$$

Where $f(L_{d-1}, L_d)$ is a neural network with two consecutive frames as its inputs.

We need to set a dynamic model because the proposed system is designed to track and extract data about the position and velocity of the athlete's body and gives a useful feedback about the exercise. That was the motivation which leaded to utilize convolutional neural network (CNN) [13, 14] as a deep learning method because it is also known as shift invariant or space invariant artificial neural network (SIANN) [15 - 19]. This feature in CNN method enables good discrimination between the right and wrong positions in a dynamic scene with different backgrounds and different physiology specifics. In order to have data set about the object with a small displacement between two consecutive frames as a vector (has amplitude and direction), Gunnar Farneback optical flow [6, 20] is the most suitable method to figure out the exact motion of each voxel in the frame with high accuracy. So, Gunnar Farneback was applied in the proposed algorithm to create a large training dataset, and then to install the training model using Tensor flow library with keras as open-source software library in-

terface. These previous methods create a high computational load. To overcome this issue tensor flow model was converted into tensor flow lite model using a converter algorithm. Tensor flow lite model is optimized as an edge deep learning model (ML) which has some key benefits like real-time latency, robustness, and small model size. This consequently reduces the memory usage as well as computational cost of neural network utilization.

3. Results

Python 3.8.5 is used to modify algorithm that had been used in [1] in order to enable interpreter which is compatible with Tensor Flow Lite model while Keras was compatible with Tensor Flow model. Three groups of data sets have been created via employing Gunnar Farneback function in python using camera to capture the video stream at 30 fps. Also, more than 500 images are captured from the previous process for model installation, and then jupyter notebook is used to train the convolutional neural networks using Tensor Flow and create the model then saved as tensor flow Saved Model format. After that a python code converter was applied in order to convert the Tensor Flow model into Tensor Flow Lite model. The proposed method is validated in several steps. First, the proposed model accuracy and complexity is compared with references [1, 2] model. Second, the algorithm is executed for counting pushups within calibrated videos from real competitions. Finally, the algorithm is executed for counting pushups live to the player of the Air defense college physical fitness team, and comparing results with a simultaneous manual judge.

3.1. Data Preprocessing

The proposed model was trained with a pre prepared data sets for a three labeled categories as shown in Figure 1. the modified algorithm with Tensor Flow Lite model is illustrated in Figure 2.

(a) (b) (c)

Figure 1. Push-ups three labeled category (a) up (b) down (c) no move.

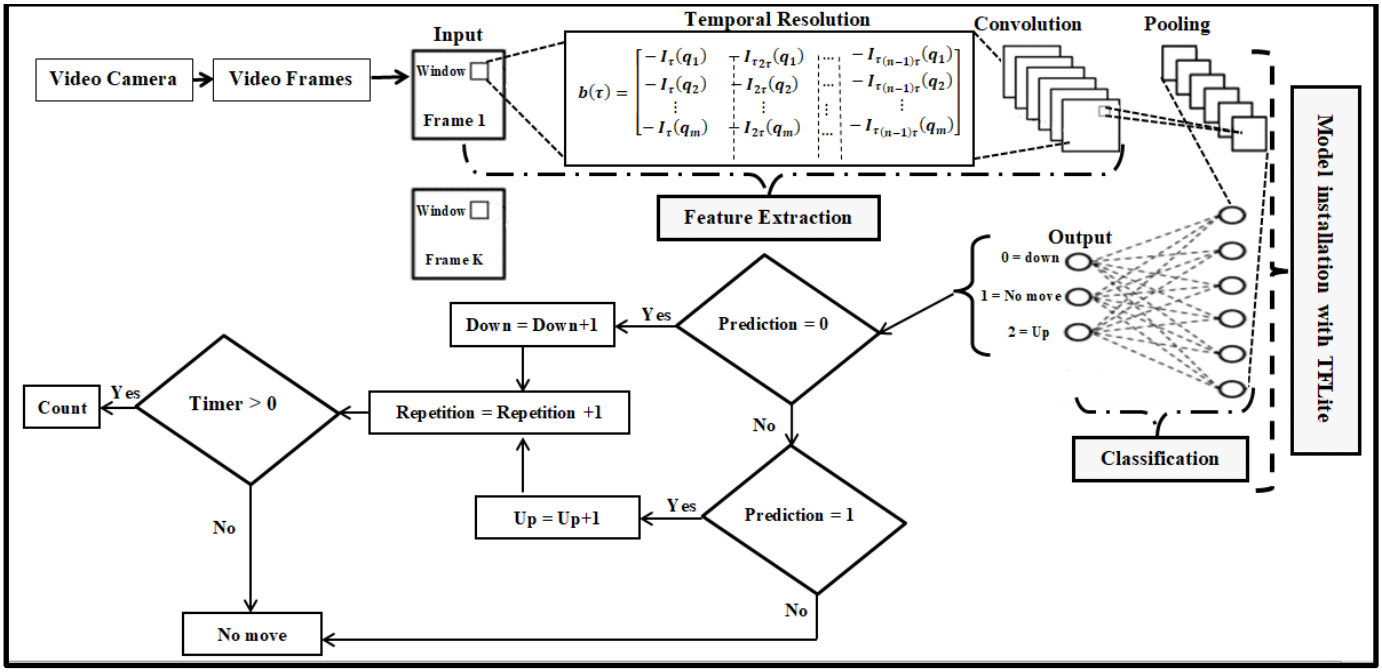


Figure 2. Block diagram of the modified system.

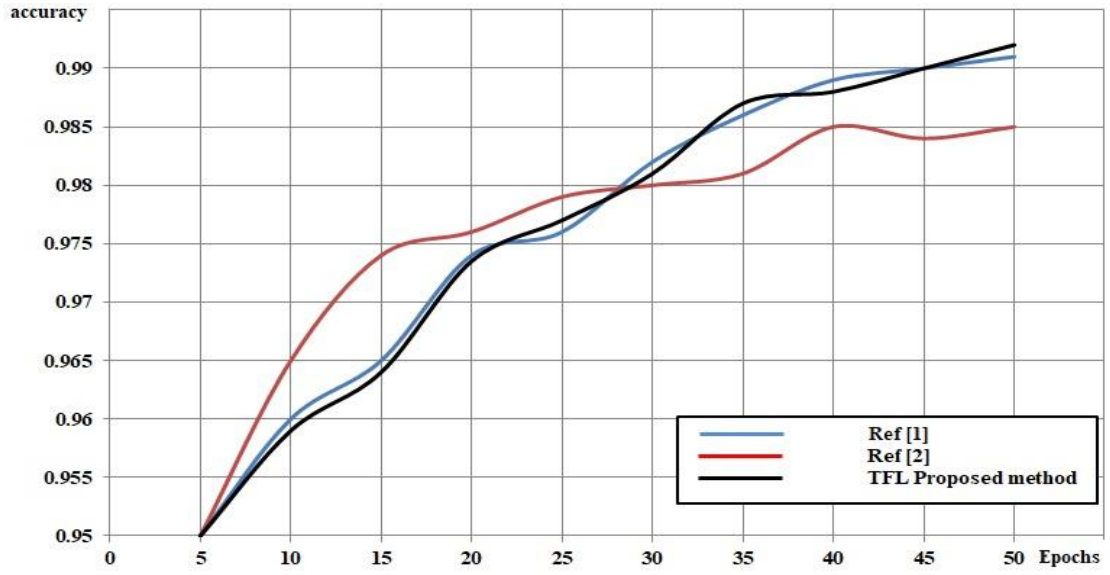


Figure 3. Model accuracy on training set.

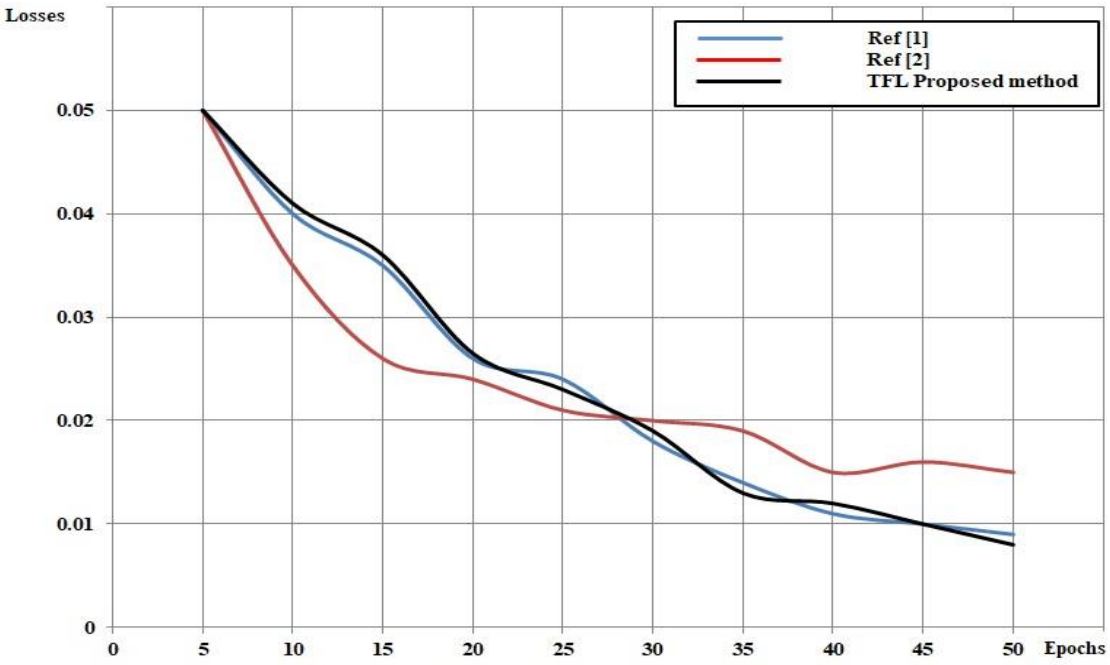


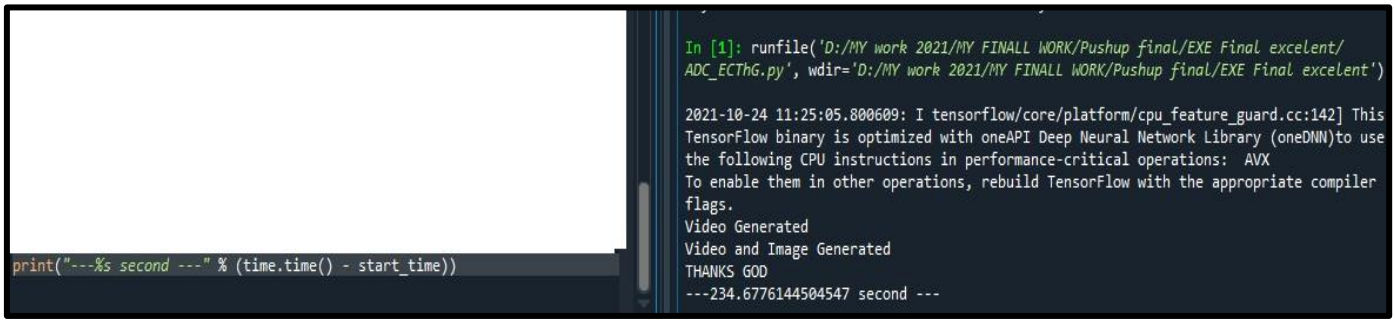
Figure 4. Model Loss on training set.

Figure 3 plots the model accuracy. And Figure 4 plots the model loss with different number of iterations compared with [1, 2]. It is noticed that the accuracy of works in [1] and this manuscript is so close, also they have some intersections. This is because the training model uses almost the same training data set in the same machine. This work aims to reduce the computational complexity for an accurate method used in [1].

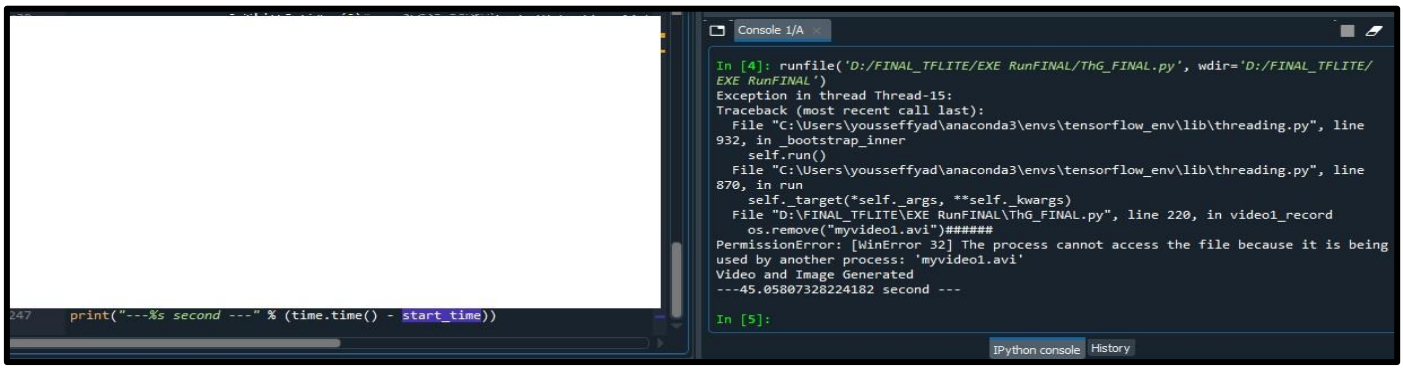
3.2 Computational complexity (big O calculations)

In order to describe behavior of the proposed algorithm we should calculate its time complexity or the Big O. The proposed is got his total processing within consecutive two frames. It selects the nearest pixels to the central pixel. It additionally finds the class with the largest predicted probability via using the mathematical function (Argmax). So the proposed algorithm has a Quasi-linear complexity [$O(n \log n)$]. It is clear that the most important factor of the complexity is the number of inputs (n) or in another meaning the size of the input array (number of pixels under test) so the proposed algorithm succeeded into reduce the computational complexity

due to use of the nearest pixels. Additionally TensorFlow Lite reduced the time taken to run the algorithm. Table 1 indicates values of time consumption with old and modified algorithms for different captured video lengths. In order to calculate the process time a python function (time) have been used as shown in figure 5.



(a)



(b)

Figure 5. Consumed time for 30 seconds video captured (a) TF model, (b) TFL model

Table 1. processing time consumed by each model for different capture periods.

Captured video length (Sec)	30	60	90	120
TensorFlow model (Sec)	234.67761445045	371.33468461037	539.1293203831	579.197007895
TensorFlow Lite model (Sec)	45.05807328224	73.9405767918	102.29705905914	126.15696787834

3.3 System excuation

This program had been tested for recorded video and life stream. Tkinter function has been used to design (GU) interface that enables users to enter the required data to run the program. Figure 6 introduces the (GU) interface and results of program with different backgrounds.

Figure 6. Proposed program (GU) interface

Figure 7 introduces snapshots of the system output for a life video stream for push-up exercises with different backgrounds and different players with different physiology specifics. Table 2 represents the counter error of the proposed algorithm compared with the related works.

Figure 7. Proposed algorithm execution for Real life video stream

Table 2. various counters errors [2].

Method	Error
Smart watch [4]	± 1 repetition
GymCam [3]	± 1.7 repetitions
Pose tracker [2]	± 1 repetition
(AI) with multi-resolution counter [1]	± 1 repetition
Proposed Algorithm	± 1 repetition

4. Discussion

Results shown that the proposed algorithm has almost same accuracy of the algorithm that had been introduced in [1]. It is obvious from Figure 3 that the accuracy of the proposed model is better than the accuracy of Ref. [1] with 0.1%, and [2] with 0.71%. Also, it is obvious from figure 4 that losses of the proposed model are less than Ref [1] with about 11%, and [2] with 47%. This improvement had been realized due to increase the amounts of training data sets with matrix dimension reduction realized by using less number of pixels within window under consideration. From section (3.2) it is noticed that the modified algorithm has a low computational complexity which is also clears in table 1. Table 1 indicates that the modified algorithm consumes less computational time with about 80%. That enhances the system performance and makes it to run in a real time. The real implementation (figure 7 and table 2) showed that the proposed algorithm has an excellent accuracy. It showed

that the proposed algorithm has not only an insignificant error (table 2) but also it is not affected by the surrounding environments and physiology specifics.
Also, it improves the performance because it reduces the process complexity (table 1). These results pave the way to apply this algorithm with a standalone platform such as a raspberry pi with Ubuntu operating system, which works only with Tensor flow Lite.

5. Conclusions

In this paper, a modified algorithm is presented. These modifications aim to reduce computational complexity of the algorithm that presented in [1]. First, tempo spatial subspaces algorithm within adaptive Gunnar Farneback is used to select the nearest pixels to the central pixel in the window under consideration. Second, TensorFlow Lite library is applied to set the system model. Results showed that this algorithm enables better performance because it enables an excellent accuracy, it is not affected by the surrounding environments or physiology specifics, and it has low computational costs. This system will enable the continuity of our ordinary life activities, especially the sports ones, without violating the precautions taken to limit the spread of the COVID-19 virus. In addition, it enables an accurate and fast auto judgment technique to measure physical fitness performance. For future work, the using of TensorFlow Lite enables a standalone platform implementation such as a raspberry pi with Ubuntu operating system because it cannot work with TensorFlow model.

6. Patents

This system introduces a novel instrument that serves the sports life. It presents safe, accurate and fast auto judgment instrument to measure physical fitness performance.

References

1. Youssef, F.; Ahmed, K.; Shady, G.; Hayman, H . AI & multi-resolution Temporal Processing for Accurate Counting of Exercises Repetitions. Proceedings of the 2021 International Telecommunications Conference, ITC-Egypt'2021, ADC, Egypt July 13 - 15, 2021.
2. Talal, A.; Chen, C. Recognizing Exercises and Counting Repetitions in Real Time. arXiv: 2005.03194 [cs.CV], **2020**.
3. Khurana, R.; Ahuja, K.; Yu, Z.; Manko, J.; Harrison, C.; Goel, M. Gym-Cam: Detecting, Recognizing and Tracking Simultaneous Exercises in Unconstrained Scenes. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **December 2018**, *Volume 2, Issue 4*, pp 1–17, https://doi.org/10.1145/3287063.
4. Soro, A.; Brunner, G.; Tanner, S.; Wattenhofer, R. Recognition and Repetition Counting for Complex Physical Exercises with Deep Learning. *sensors*, **2019**, *vol. 19, Issue 3*, pp 1–22.
5. Chen, S.; Yang, R. Pose Trainer: Correcting Exercise Posture using Pose Estimation. arXiv: 2006.11718 [cs], **2020**.
6. Gunnar, F. Two-Frame Motion Estimation Based on Polynomial Expansion. Scandinavian Conference on Image Analysis (SCIA 2003), pp. 363-370, **2003**.
7. Youssef, F.; Caiyun, W.; Qunsheng, C.; Alaa, H. A developed ESPRIT algorithm for DOA estimation. Frequenz **2015**, *vol. 69, Issue. 3*, pp. 263-269.
8. Lucas, B.; Kanade, T. An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pp 121-130, **1981**.
9. Bruce, D. Generalized Image Matching by the Method of Differences. PhD Thesis, Computer Science Department, Carnegie Mellon University, U.S.A, July, **1984**.
10. Aurelien, P.; Guy, B.; Frederic, C. Massively parallel Lucas Kanade optical flow for real-time video processing applications. journal of Real-Time Image Processing **2016**, *vol. 11, Issue. 4*. pp. 713-730,.
11. https://www.tensorflow.org/lite/convert#python_api.
12. Oscar, A. Mobile Object Detection using TensorFlow Lite and Transfer Learning. School of Computer Science and Communication, Stockholm, Sweden, August 27, **2018**.
13. Sakib, A. Introduction to Convolutional Neural Networks. Department of Mechanical and Aerospace Engineering, North Carolina State University, USA, April **2021**, DOI:10.13140/RG.2.2.11572.17282
14. Anirudha, G.; Abu Sufian; Farhana, S.; Amlan, C.; Debashis, D. Fundamental Concepts of Convolutional Neural Network. Recent Trends and Advances in Artificial Intelligence and Internet of Things, pp.519-567, India, January **2020**, DOI:10.1007/978-3-030-32644-9_36
15. Crescenzio, G. Artificial Neural Networks: tutorial. Università degli studi di Foggia, January **2015**.
16. Kuldeep, S. an introduction to artificial neural network. International Journal Of Advance Research And Innovative Ideas In Education, *vol-1, Issue-5*, September **2016**.

17. Stavros, A.; Hongwei, W. Artificial neural network and mathematical modeling comparative analysis of nonisothermal diffusion of moisture in wood. International Journal Of Advance Research And Innovative Ideas In Education, September **2016**, pp.89-93, DOI 10.1007/s00107-006-0113-0.
18. Quang, H. Artificial Neural Network Applications in Business and Engineering. University of Transport Technology, Vietnam, January, **2021**, DOI: 10.4018/978-1-7998-3238-6.
19. Primož, P. Neural Network Programming in Python. International Journal of Innovative Technology and Exploring Engineering (IJITEE) **2019**, *Volume-8, Issue-6S4*.
20. Neeta, N.; Gohokar, V. Comparative Performance Analysis of Optical Flow Algorithms for Anomaly Detection. Proceedings of International Conference on Communication and Information Processing (ICCIP) **2019**.