Fitness-Run: Real-Time Squat Detector Game for Fitness Using Computer Vision Concepts

COMP 425 - Computer Vision

Ismail Shafique Adam Atamnia   
*Gina Cody School of Engineering and Computer Science Gina Cody School of Engineering and Computer Science*  
*Concordia University Concordia University*Montreal, Canada Montreal, Canada

1Note this part (the Image Classification Model wasn’t entirely complete upon writing this IEEE conference report, thus out of honesty we pointed this out. Although it will be included in the presentation and the final report.

*Abstract*—We embarked on a mission to build an endless-runner type game, that would be controlled using physical exercises. Thus, promoting people to become more active in their lives despite their busy schedules. The endless-runner game uses key concepts in Computer Vision, specifically Landmark Detecting and Image Classification. After much consideration towards different types of exercises that can be used as the main control system (when the character jumps over obstacles), the physical movement that was best suited for our game as well as well-coordinated with Computer Vision concepts was indeed the Standard Squat. Our first approach to coding a Squat-Counter, was to first create our own dataset and identifying clear landmarks that are needed to identify squats. We did this by approximately collecting 200 images of different full-lower body poses online (these images were taken from stores like H&M, Forever21, etc.) These images varied within color, pose, noise background, variations of lower garment wear and more variables which we trained through a Convolutional Neural Network model (CNN). These images went through the process of labeling them (manually labeling the right hip, right knee, right ankle, left hip, left knee and left ankle). We concluded that the training a model manually using this technique indeed enhanced deeply our Computer Vision understanding (since we went through each and every step of the process), but the process took an extremely lengthy time. Not to mention that training the model was slightly unsuccessful due to a limited and not diversified enough dataset (we needed more like 2000 images and to also manually label all the landmarks manually as well which was deemed to be too long for a project like this). The results of our model ranged from 50-70% accuracy which was deemed to be too low for use. Moving on, we used an alternate technique which was to import a ready-made pose detection model (fully trained) and this was going to detect the same points as our customed trained model. We were than going to extract the angles between those landmarks to deem whether a squat was successful or not (and then translate that into our game). This method was successful upon completing the code and running approximately 30,000 simulations (15,000 under normal circumstances, and 15,000 with minor joint movements, which lead to noise). We also tried using a completely different technique which was an Image Classification Model (to see if it was better than our custom trained model or MediaPipe’s trained model). We manually found and labeled approximately 1000 different images and trained them as well.1 This was deemed successful (more of the results for this specific model will be discussed in the presentation and the final report (due to word-count limit and for the conference paper not to exceed the 6-page limit). Ultimately after comparing the models, we decided to use the MediaPipe model as it yielded the best results. This was than implemented within our game application successfully, after which, we designed the graphics, visual layout, character motions/states within the game.

# Reason for building the Endless-runner

As is well-known across many countries around the world, the rise of obesity has significantly increased over the past couple of decades. More specifically, recent data suggests that 26.6% of Canadians struggle with being overweight, and this data is consistent within both rural and urban communities alike (Public Health Agency of Canada, 2020).

A screenshot of a graph

AI-generated content may be incorrect.

Another study presents how over the years, there has also been a large percentage of adults being classified as physically inactive. The World Health Organization conducted a study where they found approximately an estimated 1.8 billion adults are physically inactive (World Health Organization, 4)! These statistics are extremely concerning and thus, the goal of this project is to build a computer-game application (using various concepts in Computer Vision). The main purpose of this computer game would be for the user to lose as many calories as possible (encouraging exercise while at the same time being used for weight-loss to combat obesity). The theme will be based on an endless-runner type game, while the controls will be performing exercise movements (such as performing a 150 degree box-squat which constitutes to a “jump” when going over obstacles). The goal of building this into an endless-runner application is to encourage users to beat personal records (in terms of their fitness objectives) and have a great time playing a highly interactive game! Many people struggle with exercising (as it’s physically demanding on the body) while also finding the time where they can relax themselves while playing video games. Thus, our main selling point towards this application is being able to burn calories while at the same time keeping it in a game-type theme.

# Brief background on how the game works

The game that utilizes our squat detector is an 2D auto runner meaning the character runs automatically and the player only has to control the jump to navigate platforms. The game is endless meaning it only ends when the player is eliminated by falling off the screen. We decided to do an auto runner to keep the controls simple allowing a simple squat to be to be the only movement needed to control the game. The game keeps track of the number of squats a player does and a score that describes how far the player can run without falling. The game also saves the highest score to encourage players to improve. In the future we can add a progress bar at the top that shows how close the player is to his previous best score in a way that is more easy to see. We could also add a streak system (that shows how many days in a row the player has worked out) which can encourages players to come back and work out day after day. To build the game we used the “pygame python module” and some art assets (some paid and some free) that we found online. Pygame is useful because it allows us to load sprites (images), draw them on the screen, detect collisions between the sprites, and get user input among other things. At first, when we linked the squat detector and the pygame-code, the performance dropped significantly since they were both running in the same process. To solve this problem, we decided to use separate processes for the game and the squat detector which drastically improved the game speed.

A video game screen with a city skyline

AI-generated content may be incorrect.

A screenshot of a video game

AI-generated content may be incorrect.

# Deciding which exercise is Best for the game

Knowing before-hand that we wanted to build an endless-runner game that incorporated some sort of physical exercise as the main control system (mainly the jump mechanism over obstacles), this brought up a valuable question: “which exercise to use?” Upon asking this question we decided to narrow down our exercises to three different possibilities and state the pros/cons. Thus, being able to find the right physical movement for the endless-runner game.

Option 1 - Standard Push-Up: In terms of the pros upon first glance, a standard Push-Up was ideal to be implemented in our game since it was a very standard and well-known exercise. The greater reason which inclined us to possibly use Push-Ups was due to its ease of being implemented with Computer Vision. More specifically, we imagined using a model (or training one) which could detect certain landmarks within the human body. Those landmarks would specifically be the wrist, elbow and shoulder respectively. Using that landmark detection in those certain areas, we can then test and manipulate a variety of different experiments. The main one being, finding the angle in between the ankle, wrist and shoulder which we can then use to determine if someone has indeed performed a Push-Up or not (if the angle between these limbs were 90 degrees or under, the program can record that as a successful Push-Up etc.) We can also use these landmarks to detect different variations of Push-Ups (such as diamond Push-Ups, wide-arm Push-Ups etc.) which would increase versatility to our game. Another advantage to using this exercise was the fact that its movement is quite slow, thus being easily detectable for various Computer Vision Landmark detection systems. Although there were many positive aspects when it came to the Push-Up, after further examination this exercise had many problems that were initially overlooked. Amongst them was the fact that a Push-Up was more of a strength exercise rather than being cardio-oriented (focus on building muscle rather than fitness-oriented which focuses on burning calories). This is further supported when studying how many calories are burnt, which is approximately 7 calories per minute (Push Ups Calories Burned Calculator, 1) (these numbers are approximate considering the various variables that come into play, this is assuming a person who is 150 pounds). This number is relatively low when it comes to a fat-burning exercise, and thus this was definitely a con when it came to our specific game. It’s also well-known that the average person (considering that a vast majority of the population is inactive in the first place) cannot perform more than 10 Push-Ups consecutively! Thus, the Fitness-Run game would only last 15-20 seconds per run which is much too short for duration. Since the Push-Up is an exercise which is performed in a plank position on the ground, as well as your face and entire frontal body facing the floor, this posed many problems to our game design. Most notably, not being able to see the screen when playing the game (since your eyes are constantly staring downwards). This presented a large problem as the user will not be able to even see the obstacles in the first place, and thus, this exercise was quickly discarded from being used as the jump mechanism in our game.

Option 2 - Jumping-Jack: This was another strong consideration for us in terms of a possible exercise for jumping over obstacles in the game. That being said, this exercise too had many advantages which inclined us to possibly use this within our application. The first strength was the fact that Jumping-Jacks are a very beneficial exercise when it comes to cardio and fitness. The average person (even if they’re very inactive) can perform much more Jumping-Jacks (comfortably) than Push-Ups since it focuses more on body balance rather than strength. Jumping-Jacks also burn around 9 calories per minute which is an increase when comparing to how many calories are burnt from Push-Ups in a minute (Calories Burned from Jumping jacks vigorous, 1) (these numbers are approximate considering the various variables that come into play, this is assuming a person who is 150 pounds). Another up-side to Jumping-Jacks was the fact that they are an exercise that can be performed while staring at a computer screen (unlike push-ups where you are constantly staring at the floor), thus, being aware of the actual gameplay. Despite these positive aspects we ultimately decided to discard Jumping-Jacks at the end due to a number of reasons which were not compatible with our Computer Vision design and the theme of our game. The largest reason for moving on from this exercise was the fact that performing a Jumping-Jack is a relatively quick-moving action. Not only this, but you have multiple key joints within the body that are moving all at one time in this fast motion. This was a realization that came to us after trying to code the Landmark Detection Model with Computer Vision. We found that fast movements often disrupted the pose detection landmarks (since the model constantly had to recalculate because of the movements being increasingly fast. Another obstacle that we faced (which was another reason for us to abandon this exercise from the game) was due to scalability issues. A Jumping-Jack needs the entire body to be visible from the camera-lens, thus involving more key points to be detected by the Computer Vision Model. This was an issue that was very difficult to avoid, and it hindered the performance of our application if we were to use it (in combination to the fact that these points will be moving extremely fast considering how Jumping-Jacks are performed quickly).

Option 3 (picked exercise) Bodyweight Squat: This was ultimately the exercise that we ended up choosing for our game (specifically when the character jumps over obstacles). The Bodyweight Squat was ideal for our needs in the game, and it functioned well with Landmark Detection. More specifically, it was easy for any model that we used (whether a pre-trained model or a model that we trained ourselves using CNN) to detect certain landmarks within the squat since its detection points are very apparent within the human body. For example, identifying body parts like the ankle and knee were very simple since they usually protrude even when wearing pants, thus making it simple to detect these key features using a Computer Vision Model. Another point being that a squat is a relatively slow exercise to perform and thus, most models will not have to constantly calibrate to re-find certain points (as for example the ankle is always in a constant position, while the knees and hip are moving in a slow motion). This is vital for accuracy of the model since this is something we struggled with when trying to test out exercises that involved faster ranges of motion which ultimately caused Landmark Detection to be difficult to identify smoothly. Continuing upon the last point, we realized that it would be very easy to detect a correctly performed squat since we can simply calculate the angle (between the hip, knee and ankle) to identify a standard squat (we can set the values to whatever we want, whether it be 90 degrees or under 150 degrees etc.). Finally, bodyweight squats were excellent when it came to the number of calories burnt per minute, it’s estimated that around 10 calories can be burnt per minute (if done with high intensity and assuming the person weighs 150 pounds) (Leventon, 1). After seeing the benefits this exercise brings both in terms of the Computer Vision aspect and game-performance, we decided to move forward with this action (and for it to be our main control system for the jumping of the character).

**Calories Burned per Minute (150-Pound Male)**

|  |  |
| --- | --- |
| **Exercise** | **Estimated Calories/Minute** |
| Push-ups | 7 calories |
| **Squats** | **10 calories** |
| Jumping Jacks | 9 calories |

# Training a Model Ourselves with CNN

In terms of building the squat-counter for our Computer Vision Project, we examined a variety of different methods that we can use to accomplish our goal. Amongst the earliest of them, was to train our own Model to detect squats. The first step we took to accomplish this aspiring task was to build a custom dataset. While doing much research online for ready-made datasets, we were unable to find anything that suited our needs. Thus, we decided to collect as many images as possible, specifically lower-body photos that displayed the hips, ankles and knees of a person. We started collecting these images from various websites that ranged from fitness apps, clothing stores, blogs etc.). After collecting approximately 200 images manually, we than needed to take these images and label the key landmarks that we wanted to identify. More specifically the right hip, right knee, right ankle, left hip, left knee and left ankle. Labelling these points correctly was extremely important in training our model and thus we downloaded a Python extension called LabelMe which labels keypoints (the landmarks mentioned above) that want to be identified in .jpg images. It then converts these .jpg images into JSON files (which will be later used for NumPy data array set, this will be discussed further later). One of the advantages to this process was the fact that it really made us understand the process it takes to build a dataset from scratch, and to then manually label all six landmark locations. It definitely took an extremely long process to these steps, which may have been a possible disadvantage to this idea of building an entirely new dataset.

A screenshot of a person posing for a picture

AI-generated content may be incorrect.

The next step was to take these now edited images, and to convert them to the appropriate JSON files. The reason behind this was simply because a machine is able to read a JSON file (rather than a .jpg image). This step is also vital since you would need to extract the data from the JSON file (in terms of the coordinates of the six Landmark Detection sites) and be able to store them in a NumPy data set array (for the CNN to later train with these data values). For more clarity, the JSON files would contain the x and y coordinates of each of the six landmarks (meaning there would be twelve data points per image).

A screenshot of a computer code

AI-generated content may be incorrect.

After completing this, we than decided to enter these coordinates into a NumPy

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

1. *"Obesity in Rural and Urban Canada."* Public Health Agency of Canada, 4 Nov. 2020, health-infobase.canada.ca/datalab/canadian-risk-factor-atlas-obesity-blog.html. Accessed 29 Mar. 2025. (Ignore Images and Ads)
2. World Health Organization. "Physical Activity." World Health Organization, 26 June 2024, [www.who.int/news-room/fact-sheets/detail/physical-activity](http://www.who.int/news-room/fact-sheets/detail/physical-activity). Accessed 30 Mar. 2025. (Ignore Images and Ads)
3. ​Leventon, Scott. "Calories Burned Squats Calculator." *Fitness Volt*, updated by Tom Miller, CSCS, 5 Sept. 2023,<https://fitnessvolt.com/squats-calories-burned/>. Accessed 30 Mar. 2025 (Ignore Images and Ads)
4. "Push Ups Calories Burned Calculator." *BizCalcs.com*, [www.bizcalcs.com/push-ups-calories-burned/](http://www.bizcalcs.com/push-ups-calories-burned/). Accessed 29 Mar. 2025 (Ignore Images and Ads)
5. "Calories Burned From Jumping jacks, vigorous." *MyFitnessPal*, [www.myfitnesspal.com/exercise/calories-burned/jumping-jacks-vigorous-421094](http://www.myfitnesspal.com/exercise/calories-burned/jumping-jacks-vigorous-421094). Accessed 29 Mar. 2025 (Ignore Images and Ads)
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
8. K. Eves and J. Valasek, “Adaptive control for singularly perturbed systems examples,” Code Ocean, Aug. 2023. [Online]. Available: <https://codeocean.com/capsule/4989235/tree>
9. D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: <https://arxiv.org/abs/1312.6114>
10. S. Liu, “Wi-Fi Energy Detection Testbed (12MTC),” 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC
11. “Treatment episode data set: discharges (TEDS-D): concatenated, 2006 to 2009.” U.S. Department of Health and Human Services, Substance Abuse and Mental Health Services Administration, Office of Applied Studies, August, 2013, DOI:10.3886/ICPSR30122.v2

**IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**