Title	Designing and prototyping of AI-based real-time mobile detectors for calisthenic push-up exercise
Author(s)	Xiyuan Zhang, Shawn Z H Han and Kenneth Y T Lim

Zhang, X., Han, S. Z. H., & Lim, K. Y. T. (2024). Designing and prototyping of AI-based real-time mobile detectors for calisthenic push-up exercise. *Procedia Computer Science, 239*, 445–452. https://doi.org/10.1016/j.procs.2024.06.192

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 239 (2024) 445–452

Procedia Computer Science

www.elsevier.com/locate/procedia

CENTERIS – International Conference on ENTERprise Information Systems / ProjMAN – International Conference on Project MANagement / HCist – International Conference on Health and Social Care Information Systems and Technologies 2023

Designing and Prototyping of AI-based Real-time Mobile Detectors for Calisthenic Push-up Exercise

Xiyuan Zhang[a], Shawn Z H Han[a], Kenneth Y T Lim[b]*

[a]*independent*
[b]*National Institute of Education, 1 Nanyang Walk, Singapore 637616, Singapore*

Abstract

Fitness exercises, including push-ups, are very beneficial to personal health. Many Artificial Intelligence (AI)-based fitness trainers are developed based on human pose estimation models or assisted by Internet of Things (IoT) devices. However, many of them require access to a graphing processing unit (GPU) for model training or IoT sensors to deploy, less accessible for individuals. In our work, we designed and prototyped real-time mobile push-up detectors using three distinctive approaches: (1) Push-up pose classification, (2) Angle-heuristic estimation and (3) Optical flow detection. We trained our deep-learning model with over 2000 images to achieve a high accuracy for real time deployment. Models are tested on our video dataset applied data augmentation techniques to simulate real-world environmental conditions to evaluate model performance based on accuracy metrics (precision, recall, F1 score) and processing frame rate (FPS). From the results, we concluded that the angle-heuristic estimation method has the best overall performance and we analysed the reasons for the relatively poorer performance of the push-up pose classification and optical flow detection methods. All methods developed are capable of working on mobile devices without the need of GPU or IoT sensors.

* Corresponding author. Tel.: +65-67903409.
E-mail address: kenneth.lim@nie.edu.sg

1877-0509
10.1016/j.procs.2024.06.192

Keywords: machine learning ; computer vision ; optical flow ; human pose estimation ; fitness ; push-up

1. Introduction

Fitness exercises, including push-ups, are very beneficial to personal health. Many Artificial Intelligence (AI)-based fitness trainers, like Kemtai [1] or SmartMat [2] are developed based on human pose estimation models that are assisted by the Internet of Things (IoT) sensors to help users to track their repetitive exercises to keep exercise record and to assess their performance to prevent injuries due to improper exercise forms. However, most current AI-based fitness trainers require graphing processing units (GPU) for model training and IoT devices to deploy, which are more costly and less accessible for developers and users. Human pose estimation (HPE) is a computer-vision based technology using deep learning approach that aims to locate the human body parts and to build human body representation from input data, including images and videos [3]. It has been utilised in a wide range of applications, including fitness trackers. Models developed for HPE include the MediaPipe [4], MoveNet [5], and PoseNet [6]. In this project, MoveNet is used to build the push-up pose classification model and the angle-heuristic model as MoveNet not only give fast results that is capable for mobile application, it performs especially well for exercise detection [15] as is it trained on Active Dataset from Google that have a robust collection of exercise video database [12]. The third model uses the optical flow approach to perform push-up detection based on pixel motion detected across successive frames. This paper reports an independent research study conceptualized, designed and enacted by a pair of high school students in Singapore, under the mentorship of the third author. The study took place from April 2022 to March 2023, and aimed to design and prototype AI-based real-time push-up detectors for mobile applications that are capable of achieving high accuracy and processing speed on mobile devices without the need of GPU for developers or special IoT devices for users.

2. Literature review

This project aims to design and prototype AI-based real-time push-up detectors using deep learning classification, angle-heuristic and optical flow approaches, suitable for mobile implementation that do not need access to GPU or IoT devices.

2.1. Deep learning

Deep learning, a form of machine learning or a subfield of Artificial Intelligence, allows computational models that include hidden neural networks to extract features from data to output predictions [7]. Deep learning models are commonly used in image classification and are trained on large-scale datasets like Microsoft Common Objects in Context (COCO) [8] and ImageNet [9]. However, due to the large labelled dataset required to train a model, transfer learning (TL), which takes advantage of learned feature maps from pre-trained models [10], is used to customise models for a specific task with fewer images [11] (Fig. 1).

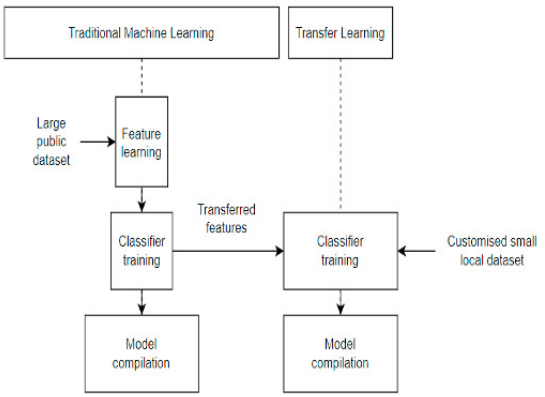

Fig. 1. Comparison of the workflow of a traditional ML model and TL model.

2.2. Human pose estimation (HPE)

Human pose estimation (HPE) has been a hard task for computer vision. There have been many fast and accurate 2D human pose estimation models like MoveNet, PoseNet, OpenPose for single or multiple human pose estimation. MoveNet [5] relies on TensorFlow object detection API using Mobilenet V2 as a feature extractor and 4-step prediction to give fast and accurate detection. It can detect 17 key points in the human body [14] (Fig. 2) and output 2D pixel coordinates from the input human image.

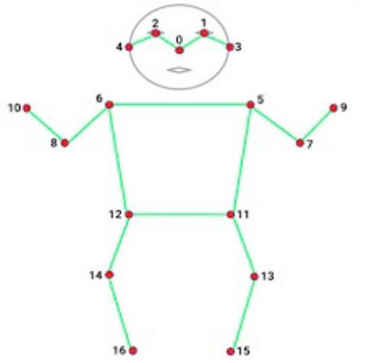

Fig. 2. Keypoint map of MoveNet.

MoveNet is trained on COCO dataset [8] and Active dataset [12], which allow the model to be exposed both a diversity of real-world scenarios like skin tones in COCO and more specifically various fitness poses including push-ups in Active dataset, giving it a significantly better performance for intense fitness motion. MoveNet is offered with 2 variants: the Lightning [15] and the Thunder [16], both support 30+ FPS on most laptops and phones. For prototyping of the pose classification model, we used MoveNet Thunder to preprocess the video frames for more accurate keypoint labelling and ran detector inference using MoveNet Lightning for faster detection.

2.3. Optical flow

Optical flow is the displacement of detected objects between two consecutive frames [18]. It is a 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second [20]. The sparse Lucas-Kanade algorithm, based on the brightness constant assumption [18], detects motion of objects by pixel intensity changes of the neighbourhood pixels and outputs a set of optical flow vectors. The complexity of the model, hence processing speed of the detector, is proportional to the size of input image [19]. It can be used for motion estimation, like repetitive push-ups, based on the motion of the user under a static background environment. While there have been some implementations of AI-based methods using HPE models like OpenPose or MediaPipe to perform fitness detection, little has been done to explore different models' performance for mobile implementation using accuracy and frame rate metrics. Other methodologies including optical flow algorithms are less commonly implemented for fitness tracking.

3. Methodology

For our research, we created three different methods to detect and count push-ups efficiently on mobile devices: push-up pose classification, angle-heuristic estimation, and optical flow detection. Of these three methods, the first two use MoveNet to determine body key point coordinates as MoveNet performs better for single entity exercise detection than other models. This section will present how each method was created and estimate the number of push-ups completed in testing videos.

3.1. Push-up pose classification

The pose classification method, as illustrated by Fig. 3, consists of two parts: HPE using MoveNet Lightning and the push-up pose classification model. To train our pose classification model through transfer learning, we used an online dataset [21] that contained over 2000 images of "Up" and "Down" push-up positions.

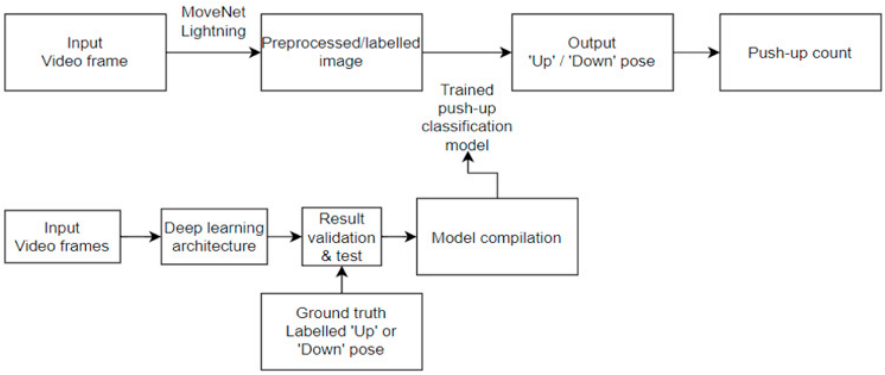

Fig. 3. Overview of the Pose Classification Method.

Each image from the dataset was passed through MoveNet Thunder as it gives a higher accuracy for keypoints detection. These data points were then flattened into feature vectors and used to train our model, which consisted of several Dense layers. To test our model in actual videos, OpenCV was used to iterate through each frame of the video. Each frame was fed through MoveNet Lightning first as it has a higher inference speed, giving faster keypoint detection. The results were then fed into our pose classification model. The output will be stored to the "state" variable and it will either be "Up" or "Down". Whenever "state" changes from "Down" to "Up", it will be counted as one complete push-up.

3.2. Angle-heuristic estimation

The angle estimation method will also use MoveNet Lightning to process each frame, and key point coordinates will be used to find the angle made by the elbows and shoulders. By using specific key points on the body, angles of the elbows (θe) and shoulders (θs) could be obtained. The sum of these angles (θsum) will be plotted (Fig. 4). A low-pass filter was used with Fast Fourier Transform (FFT) to smoothen the graph, and each peak was taken as one push-up (Fig. 5)

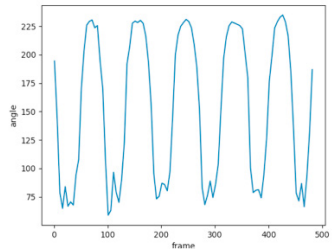

Fig. 4. Graph of θsum against number of frames.

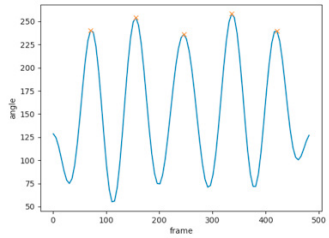

Fig. 5. Smoothened θsum graph with plotted peaks.

3.3. Optical flow detection

The optical flow method is the only method that does not use MoveNet. Instead, it senses movements of "interesting" pixels across successive frames. For our implementation of optical flow, the Lucas-Kanade algorithm was used for every subsequent frame, where the summation of the difference in y-coordinate values (Δy) was calculated. This was then plotted for each frame (Fig. 6). A low-pass filter was used with FFT to smoothen the graph, and each peak was taken as one push-up (Fig. 7).

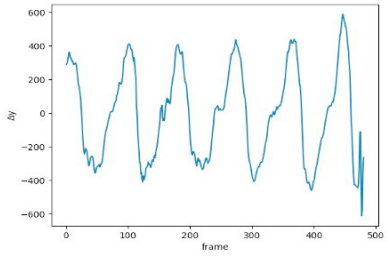

Fig. 6. Graph of Δy against number of frames.

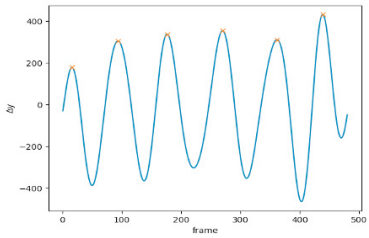

Fig. 7. Smoothened Δy graph with plotted peaks.

3.4. Performance metrics

To test the performance of our three methods as mobile detectors, we used two main metrics, namely accuracy and speed. For accuracy, F1 score was calculated from precision and recall values. For speed, frames per second (FPS) processed by the methods was used. We tested our methods on two camera angles, front view and side view. Additionally, to simulate real world camera and lighting conditions, the side view videos were augmented with OpenCV to give dimmed and blurred versions of the videos.

4. Results and discussion

We present quantitative and qualitative results of push-up detectors tested on our dataset. All tests were conducted on a Windows 10 desktop with an Intel(R) i7-900K 8 core CPU. Our deep learning pose classification model was tested on a set of 647 push up and down images. It had a high accuracy (0.96), but a relatively high validation loss (0.21). (Fig. 8).

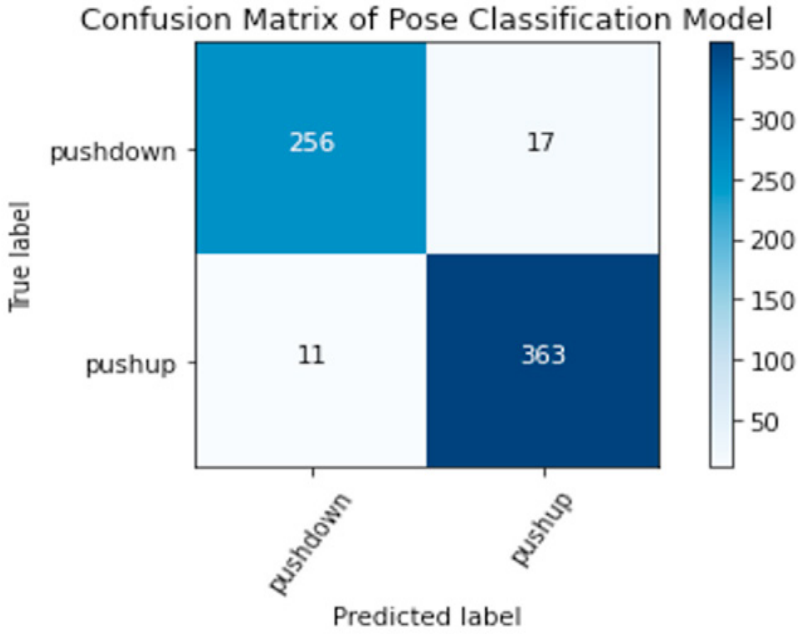

Fig. 8. Confusion matrix of pose classification model.

To evaluate across the methods, accuracy, represented by F1 score, and inference speed, represented by FPS, were assessed and the results are presented in Table 1.

Table 1. Accuracy and processing speed for methods under different conditions

Category	Angle-heuristic				Pose classification				Optical flow			
	Precision	Recall	F1	FPS	Precision	Recall	F1	FPS	Precision	Recall	F1	FPS
Side nor.	0.95	0.82	0.88	87	0.94	1.00	0.97	47	1.00	0.73	0.84	77
Ft. nor.	1.00	0.68	0.81		0.83	1.00	0.91		1.00	0.51	0.68	
Side dim.	0.89	0.82	0.86		0.84	1.00	0.91		1.00	0.65	0.79	
Side blur.	0.93	0.80	0.86		0.91	1.00	0.95		0.95	0.76	0.84	
Average	0.94	0.78	0.85		0.88	1.00	0.94		0.99	0.66	0.79	

Among all the methods, the deep learning based pose classification method has significantly fewer frames processed per second, suggesting a much slower image processing speed compared to other two methods. This was largely expected as pose classification runs two machine learning models to perform the detection task, taking more time to process each frame. What was unexpected was that the optical flow method was slower than the angle estimation, despite optical flow not running a deep learning model. One explanation could be that the optical flow method is limited by the slow Hessian calculation step within the Lucas-Kanade algorithm [22], whereas MoveNet Lightning has a relatively small neural network architecture [13], increasing its speed. Lower FPS leads to longer response times for users' push-ups to be detected, affecting user experience and lowering push-up count accuracy. In reality, most mobile cameras run at 30 FPS or 60 FPS. Hence, lower FPS methods will still run in real time with these cameras. To assess accuracy of models, we will focus on the F1 score. Higher F1 correlates to higher accuracy. All models had a relatively high F1 (>0.79), with pose classification being the most accurate (F1=0.94), while optical flow being the least (F1=0.79). This was largely expected as the pose classification method runs MoveNet Lightning, which has been extensively pre-trained to improve accuracy [5], as well as our pose classification model, with a high validation accuracy of 0.96. The optical flow method is the least accurate as it detects movements of "interesting" pixels across the whole frame, regardless of whether it is the person or not. Hence, any shift in the background may impact the optical flow accuracy greatly. Additionally, there is a decrease in F1 from side view to front view videos for all approaches as the motion change is relatively less noticeable in front view than the side view. Due to the front view, MoveNet is unable to extract key points from the lower end of the body as they are concealed by the top half. Due to this, training and performance of the pose classification model for front view will drop. Furthermore, angle detection will also be impacted. Currently, the 2D angle estimation method only calculates angles that can be detected from the side view. When switching to front view, the same angles cannot be used. Hence, this led to a significant decrease in accuracy. A possible solution is by combining pose classification with angle estimation. Pose classification can differentiate front view or side view and the angles used will change accordingly. Furthermore, all methods showed a slight decrease in F1 when the videos were dimmed or blurred. Under realistic camera conditions, there will be varying lighting conditions and blurring. This implies that our model will work less well for actual real time usage. However, the drop in performance is not significant. Hence, our model can still work well under realistic conditions.

5. Conclusion

In this report, real-time mobile push-up detectors based on computer vision and artificial intelligence are designed and prototyped with deep learning pose classification, angle-heuristic and optical flow approaches. The detectors are evaluated based on accuracy (F1) and processing frame rate (FPS). It was found that the pose classification method was the slowest, but most accurate. This was mainly due to machine learning being more accurate in identifying critical poses of push-ups. However, frames are fed into models twice for key points and pose detection, decreasing processing speed. Conversely, the optical flow method that relied solely on computer vision had the poorest accuracy, with average processing speed, making it the weakest method amongst the three for real-time push-up detection. Overall, the angle-heuristic method had the most balanced performance. It had above average accuracy while having the fastest processing frame rate, making it the best to be used on mobile devices. This result is useful in improving the robustness of the prototyped push-up detectors for different real-world conditions and mobile implementations. For future work, we wish to convert our model to tflite to deploy on mobile applications. To further boost the accuracy of the angle-

heuristic algorithm, we could also explore the utility of RGB-Depth camera that are implemented by some of the smartphones [23], which could help calibrate the angle of the keypoints based on 3D coordinates and give an alternative factor for pose detection, especially for the front view cases where angle points are hidden. To adapt the MoveNet model specifically for push-up detection, we could explore the use of active learning to train the model on labeled push-up frames at top and bottom frames to increase the accuracy of keypoint detection at extreme positions to further enhance the accuracy of push-up detection. Last, our future work also involves developing the code into a deployable API package to allow implementations of more features on the prototype, including personalised AI trainer services and push-up correction features.

References

[1] Persaud, C. (2022) “Fitness Apps Can Now Analyze Your Movements Through AI,” Wifi Hifi Magazine.

https://wifihifi.com/fitness-apps-ai-celebrities-gami ng/

[2] SmartMat (2018)“SmartMat | Interactive In-Home Yoga,” SmartMat. https://www.smartmat.com/about/

[3] Ghanem, B., P. Rosso, and F. Rangel (2018) “An Emotional Analysis of False Information in Social Media and News Articles,” vol. 1, no. 1, 2018, doi: 10.1145/1122445.1122456.

[4] Google (2022) “Pose,” mediapipe. https://google.github.io/mediapipe/solutions/pose.html

[5] Google (2021) “MoveNet: Ultra fast and accurate pose detection model.,” TensorFlow. https://www.tensorflow.org/hub/tutorials/movenet

[6] A. Kendall, M. Grimes, and R. Cipolla (2016) “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization,” arXiv:1505.07427 [cs], Feb. 2016, [Online]. Available: https://arxiv.org/abs1505.07427

[7] Copeland, M. (2016) “The Difference between AI, Machine Learning, and Deep Learning? | NVIDIA Blog,” The Official NVIDIA Blog, Jul. 29, 2016. https://blogs.nvidia.com/blog/2016/07/29/whats-dif ference-artificial-intelligence-machine-learning-de ep-learning-ai/

[8] “Papers with Code - COCO Dataset,” paperswithcode.com. https://paperswithcode.com/dataset/coco

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009) “ImageNet: A large-scale hierarchical image database,” 2009 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2009, doi: 10.1109/cvpr.2009.5206848.

[10] Liu, F., Xu, X., Qiu, S., Qing, C., and Tao, D. (2016) “Simple to Complex Transfer Learning for Action Recognition,” IEEE Transactions on Image Processing, vol. 25, no. 2, pp. 949–960, Feb. 2016, doi: 10.1109/tip.2015.2512107.

[11] Google (2022) “Transfer learning with a pretrained ConvNet | TensorFlow Core,” TensorFlow. https://www.tensorflow.org/tutorials/images/transfer_learning

[12] Google (2021) “next-generation-pose-detection-with-movenet-and-tensorflowjs” https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html

[13] Ambianic.ai (2021) “Comparing MoveNet to PoseNet for Person Fall Detection,” Ambianic.ai Blog, Sep. 02, 2021. https://blog.ambianic.ai/2021/09/02/movenet-vs-p osenet-person-fall-detection.html (accessed Dec. 31, 2022).

[14] Google (2022) “Pre-trained TensorFlow.js models,” GitHub, Dec. 30, 2022. https://github.com/tensorflow/tfjs-models/blob/master/pose-detection/README.md (accessed Dec. 31, 2022).

[15] Google (2022) “TensorFlow Hub,” tfhub.dev. https://tfhub.dev/google/movenet/singlepose/lightning/4

[16] Google (2022) “TensorFlow Hub,” tfhub.dev. https://tfhub.dev/google/movenet/singlepose/thund er/4 (accessed Dec. 31, 2022).

[17] Hui, J. (2019) “mAP (mean Average Precision) for Object Detection,”Medium, Apr. 03, 2019.

https://jonathan-hui.medium.com/map-mean-aver age-precision-for-object-detection-45c121a31173

[18] Google (2022) “OpenCV: Optical Flow,” docs.opencv.org. https://docs.opencv.org/3.4/d4/dee/tutorial_optical

_flow.html (accessed Dec. 31, 2022).

[19] Rojas, R (2011) “Lucas-Kanade in a Nutshell.” [Online]. Available:

http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/d ocuments/tutorials/Lucas-Kanade2.pdf

[20] Green Brain Quadrotor (2013) “What is Optical Flow?,” Green Brain Quadrotor.

https://greenbrainundergrads.wordpress.com/201 3/11/04/what-is-optical-flow/ (accessed Jan. 01, 2023).

[21] Kaggle (2022) “Push-up Exercise,” www.kaggle.com. https://www.kaggle.com/datasets/shoreefuddin/pu shup-exercise (accessed Jan. 03, 2023).

[22] S. Baker and I. Matthews (2004) “Lucas-Kanade 20 Years On: A Unifying Framework,” International Journal of Computer Vision, vol. 56, no. 3, pp. 221–255, Feb. 2004, doi: 10.1023/b:visi.0000011205.11775.fd.

[23] Taneja, A. (2021, January 1). Top 10 smartphones with a dedicated depth sensor camera to capture Perfect bokeh shots: Cashify blog. Cashify. https://www.cashify.in/top-10-smartphones-with-a-dedicated-depth-sensor-camera-to-capture-perfect-bokeh-shots