ביה"ס להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין
The Rachel and Selim Benin School of Computer Science and Engineering

האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

AI-Aided Yoga Personal Trainer

Advisor:

Mr. Gal Katzhendler - gal.katzhendler@mail.huji.ac.il,
The Hebrew University of Jerusalem

Mentor:

Mr. Omri Avrhami

Table of Contents

Abstract

Physical training is a necessary thing for maintaining a healthy lifestyle. However, when performing physical training it is very important to perform the movements as required. Nowadays, with the development of technology, many users perform physical training on their own without a trained guide near them, which may lead to irreversible injuries.

We aim to create a system that will oversee the user while training. The system will incorporate deep learning and machine learning tools to detect and examine the user's movements. During training, the system will provide real-time feedback on the training and guide the user to correct his mistakes.

We built an AI-aided android application that replaces a personal trainer and allows users to practice yoga poses while avoiding injuries. Our system detects and classifies the poses using neural networks and suggests a correction to the user if needed. The correction algorithm we developed using ML tools is based on a dataset of correct posture we collected.

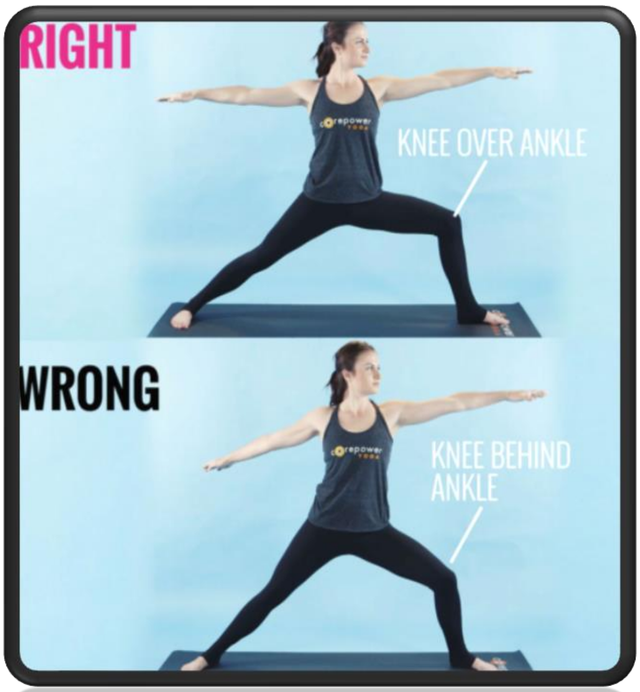


Figure 1: Yoga Pose Performance

Introduction

A common cause of injuries during physical training is improper technique. To prevent this kind of injuries, one can hire a personal trainer. Hiring a personal trainer is costly and, in many cases, inaccessible. By solving this problem, we give people the independence to practice without any outside help. This problem relates to many kinds of physical training methods like abs workouts, weight lifting, yoga, and others. We decided to focus on yoga.

Creating a platform for yoga poses is not an easy task. The system should first detect the pose the user is trying to perform. Then, after asserting that the pose was identified correctly, the system needs to suggest corrections to the user. In addition, we want the system to be scalable, fitting a wide range of possible exercises, while remaining as accurate as possible, and keeping the processing time reasonable.

The classification part of our project requires a transition from looking at a picture to understanding that there is a person in the picture with different organs in certain places that express something. What they express is yoga postures. With the help of training a neural network on a large dataset of images that we collected ourselves of yoga poses performed correctly, we developed a tool that identifies for a random person in a new photo what yoga pose he tried to perform, which is a critical step on the way to give him his correct corrections instructions.

Then, once the pose was detected, we tried to understand how should we fix the user. We tried different ideas and researched the topic and came to the understanding that we should use the KNN algorithm. We tried several different K values as we realized that higher values of k cause a reduction in the effect of noise on classification, but cause the boundaries between classes to be less distinct. Finally we chose a value of K=3 when was selected by trail-and-error and the best results were achieved from k=3.

Finally we reached desired results for us, 93% of the images entered for correction were corrected as required precisely.

Next, we added a basic application, which we wrote in the java language. A link to an example of its execution can be seen in the appendix.

Today there are applications that can identify poses in yoga and correct them according to information that the user inserts manually. We aimed to build a system that gets a dataset and learns the information of the correct pose by itself. We made a user-friendly application so that users can use it comfortably and properly.

For example in this link we can see an implementation of yoga pose correction by a yoga teacher. This software does not identify the pose and the correction is done according to the input values of the slopes and degrees of each pose (Which were entered as input in advance by a certified yoga teacher).

In conclusion, the solution we offer is as follows: An Android application in which a user photographs himself performing a yoga pose. Our system detects the pose the user was trying to perform. Then, using a classification network classify the pose. The system finds the body angles that describe the pose the user is performing and compares them to the body angles of a correct pose. Then, if an anomaly was detected, the system sends the user an appropriate correction message.

Related Work

- https://ieeexplore.ieee.org/document/9509937:
  An academic paper, describes a deep learning model, which uses CNN for yoga pose identification and correction system. (No implementation attached)
- https://ieeexplore.ieee.org/document/9310832:
  An academic paper, describes a system that uses a model that classifies different poses and notifies the users of their performance. Uses OpenPose and a pose detection module. It consists of a Deep Learning model, which uses time-distributed Convolutional Neural Networks. (No implementation attached)
- https://github.com/katjawittfoth/Aligned_Yoga_App:
  Implementation of yoga pose correction by a yoga teacher. This software does not identify the pose and the correction is done according to the input values of the slopes and degrees of each pose (Which were entered as input in advance by a certified yoga teacher).
- https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1932&context=etd_projects : A project of Yoga Pose Classification Using Deep Learning. This article lays the foundation for building such a system by discussing various machine learning and deep learning approaches to accurately classify yoga poses on prerecorded videos and also in real-time. The project also discusses various pose estimation and keypoint detection methods in detail and explains different deep learning models used for pose classification.

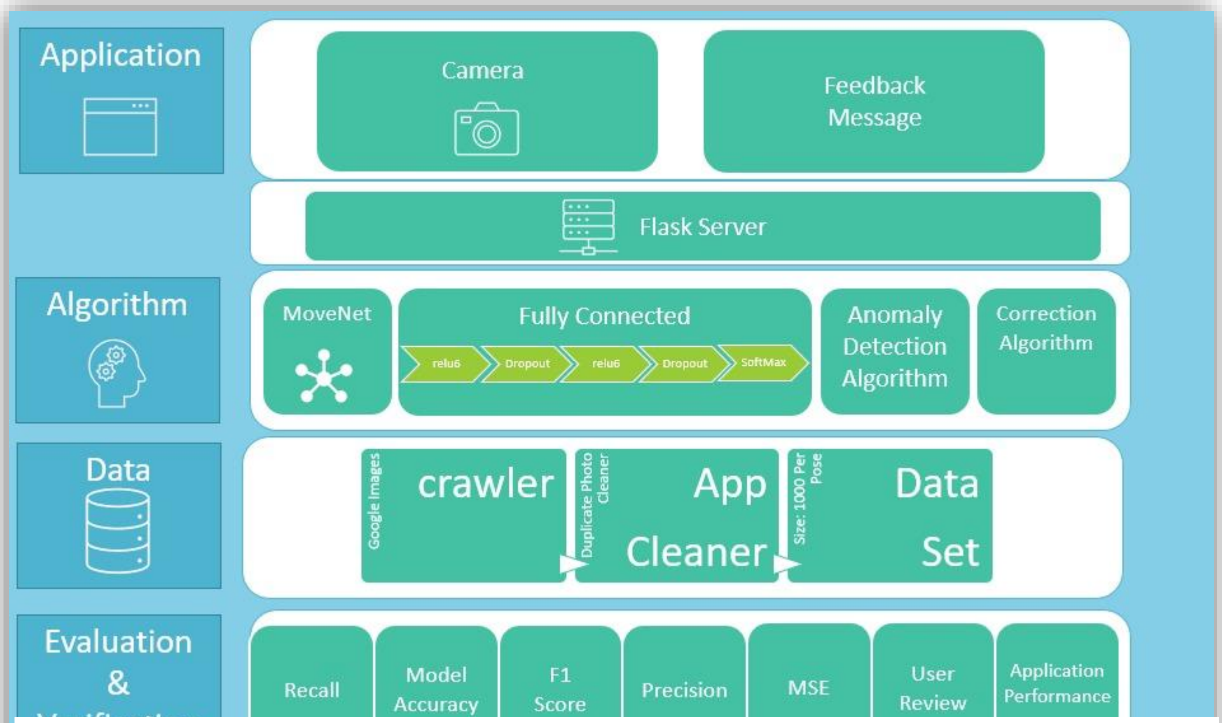


Figure 2: An Overview Chart. In this chart we can see full flow of our project. In the first row we see the camera of our application. Through it we photograph the user. The image goes through the flask server (as you can see in the second line). In the third row we see the components of the algorithm, which includes MoveNet, a neural network that classifies an image of a person into his body coordinates, a fully connected network, an anomaly detection algorithm where we identify an anomaly if it exists, and then the correction algorithm, which we use when an anomaly is detected as detailed below. We collected the dataset mainly with the help of a web crawler and passed it through a photo cleaning application. And finally we returned the output received by the application and presented output to the user. In the last row you can see the evolution and verification methods we used for the project estimation.

Methods and Materials

Data Collection

When we began working on the project we tried to find an existing data set, but we only found collections of about 100-150 images for a pose, and that was not enough for classification and correction methods.

Data for the classification task:

We collected a dataset of 4000 images. Approximately 1000 pictures for each yoga pose.

We found some data sets in Kaggle website such as [1,2] . In addition, we created a crawler that gets a string, operate a search in google images for it, and returns a data set of the resulting images. Furthermore, we photographed our family, our friends, and ourselves to create more data with different lighting, clothing, backgrounds, and ages.

In the data set, there were duplicate images and some were not good enough (images of incorrect poses or images that were not of the yoga pose at all). To clean our data set we first used the "photo duplicate cleaner" application [3] to clean duplicate images. Then, we went over the dataset and deleted faulty images.

Data for testing the correction task:

We took pictures of people deliberately performing Yoga poses inaccurately. We collected 100 images per pose. Then we tagged manually each image with the appropriate correction message.

Classification

We used a pre-trained neural network, called MoveNet, to detect the body coordinates. MoveNet is an ultra fast and accurate model that detects 17 keypoints of a body. The model is offered on TF Hub with two variants, known as Lightning and Thunder. Lightning is intended for latency-critical applications, while Thunder is intended for applications that require high accuracy. Both models run faster than real time (30+ FPS) on most modern desktops, laptops, and phones, which proves crucial for live fitness, health, and wellness applications. Using this neural network we got an image as an input and detect the 17 key points of the body such as the nose, the left shoulder, etc. The network returns a matrix of the [x, y, score] coordinates of each of the body parts, where the score represents the confidence.

In addition, we built a deep fully connected neural network that classify four yoga poses: "The Warrior 2", "Tree", "Downward Facing Dog", and "Cobra".

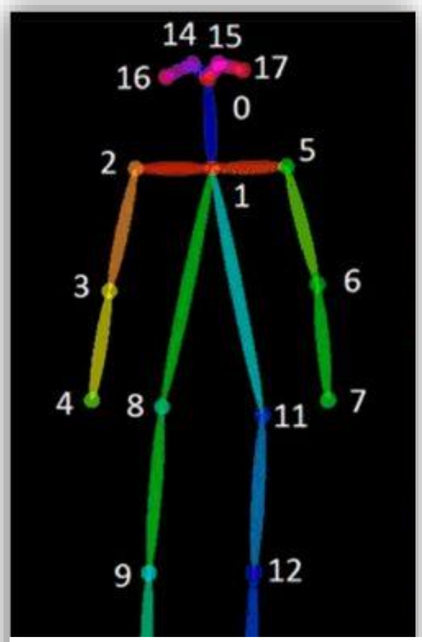


Figure 3: MoveNet Body Keypoints. Each pose contains 17 keypoints, with absolute x, y coordinates, confidence score and name

Network Architecture

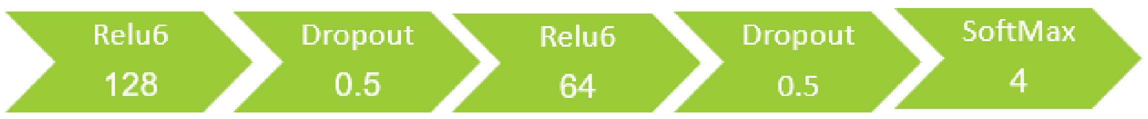


Figure 4: Network Architecture

Optimizer: Adam [4]

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

Loss: 'categorical_crossentropy'

Metrics: 'accuracy'

Using the TensorFlow package [5] we built a fully connected neural network. Then we trained it on the MoveNet's output to classify the four Yoga poses.

Correction

Once the current pose is classified, we turn to determine whether it was performed correctly. We projected the body coordinates of each image in the data set into new angles space such that each angle is the angle between the key points of the body. For example, the right elbow's angle was computed with the coordinates of the right elbow, the right wrist, and the right shoulder.

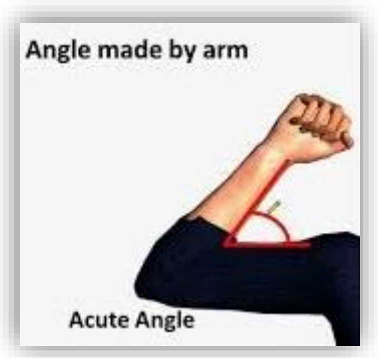


Figure 5: Angle Illustration

Every two key points constitute a line. We use the following formula to calculate the angles between two lines:

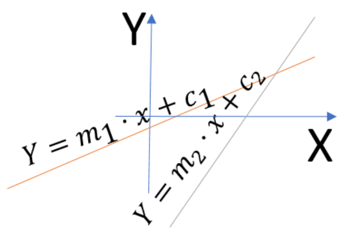
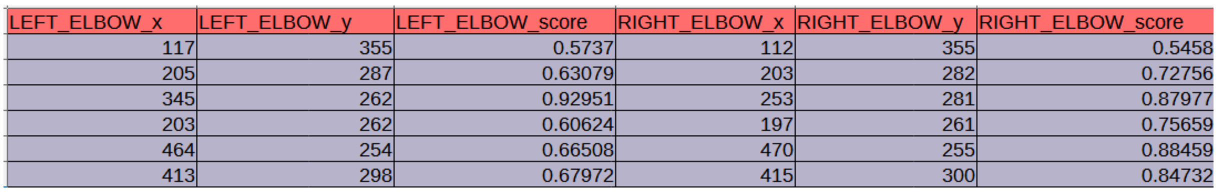


Figure 6: Lines Illustration

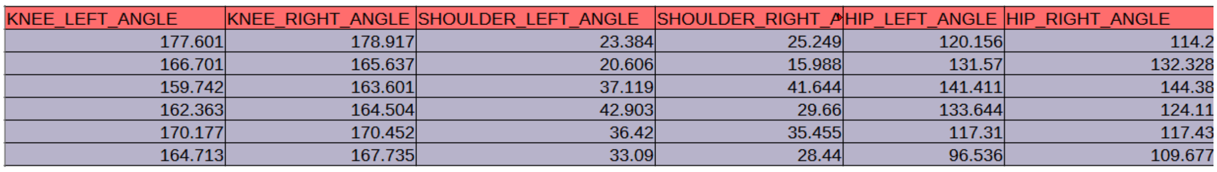
$$tan(\propto) = \frac{m1 - m2}{1 + m1 \cdot m2} \vee$$

Figure 7: Angle Calculation Formula

In the previous example, the shoulder and elbow coordinates constitute a linear line, the elbow and the wrist constitute a different linear line, and the right elbow's angle is determent as the angle between the two lines.



LEFT_ELBOW_x	LEFT_ELBOW_y	LEFT_ELBOW_score	RIGHT_ELBOW_x	RIGHT_ELBOW_y	RIGHT_ELBOW_score
117	355	0.5737	112	355	0.5458
205	287	0.63079	203	282	0.72756
345	262	0.92951	253	281	0.87977
203	262	0.60624	197	261	0.75659
464	254	0.66508	470	255	0.88459
413	298	0.67972	415	300	0.84732



KNEE_LEFT_ANGLE	KNEE_RIGHT_ANGLE	SHOULDER_LEFT_ANGLE	SHOULDER_RIGHT_A	HIP_LEFT_ANGLE	HIP_RIGHT_ANGLE
177.601	178.917	23.384	25.249	120.156	114.2
166.701	165.637	20.606	15.988	131.57	132.328
159.742	163.601	37.119	41.644	141.411	144.38
162.363	164.504	42.903	29.66	133.644	124.11
170.177	170.452	36.42	35.455	117.31	117.43
164.713	167.735	33.09	28.44	96.536	109.677

Figure 8: Projection Illustration. Each angle is computed by the formula above. In our project we projected the coordinates of the body we got from MoveNet into the angles space.

We have created an array for each pose that holds the angles vector space for each picture of the pose in the data set. We know the pose from the classification stage, thus we can use the specific array for correction. We insert the relevant pose array with the new vector space of the picture we want to correct to a KNN model. This enables us to calculate the distance from the k=3 nearest neighbors and to check if the angles of the new data need correction. The k=3 was selected by trail-and-error and the best results were achieved from k=3. The correction is done by comparing to a pre-determined threshold of 5 degrees error, the threshold was determined with the help of a yoga instructor, and if the degree difference was larger than the threshold, an informative correction message was returned.
In the previous example, if the user's shoulder degree was larger than it should be, a message that states that the user should lower his hand would have returned.

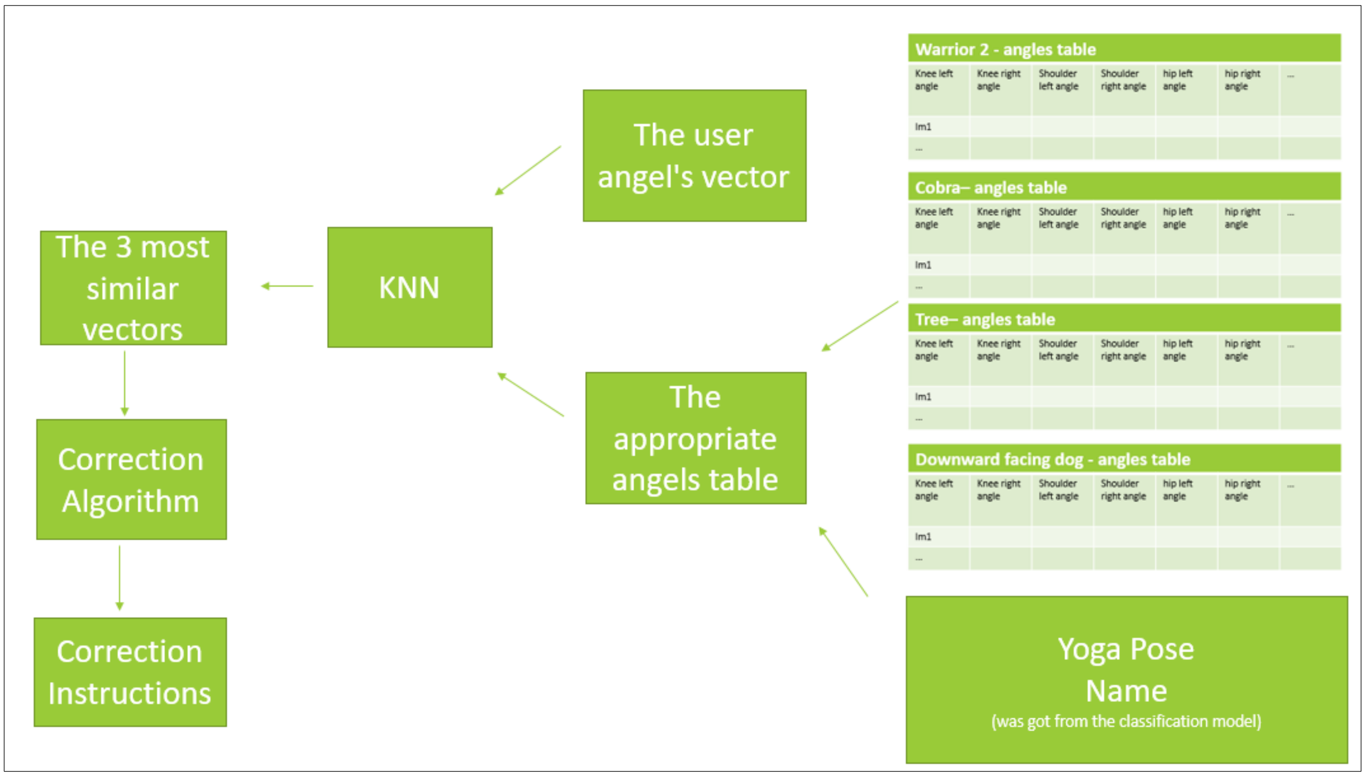


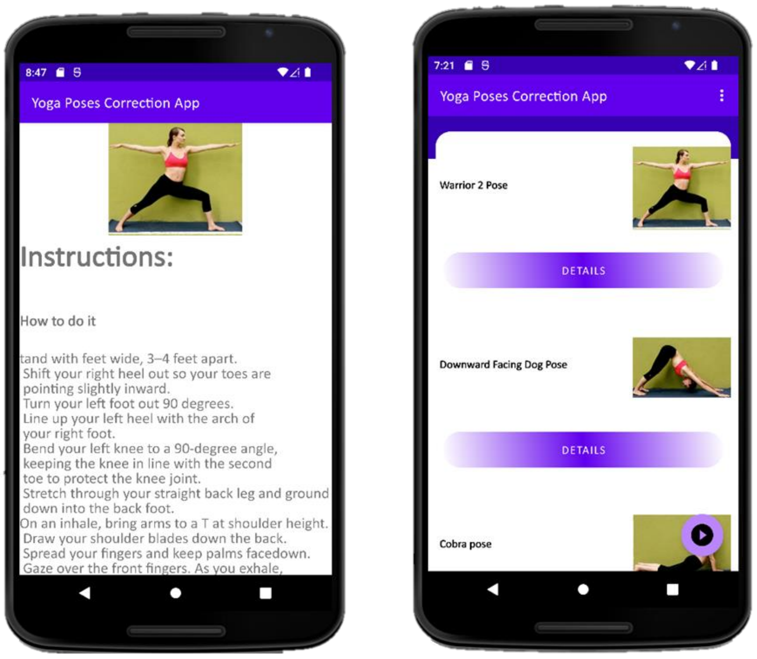
Figure 9: Correction Algorithm Illustration. Each image after classification goes through the KNN algorithm. Through it we will get the 3 images with the most similar body angles. We will perform the correction algorithm and return the correction instructions to the user when necessary.

Application

We built a user-friendly android application that corrects the user in real time by taking a picture of the user while training, sending that picture to a Flask server that runs the algorithm mentioned above, and presents the correction instruction it gets from the server**.**

Our application provides the user with the following options:

1. The user can choose a pose and read the instructions to do the pose correctly.


8:47

Yoga Poses Correction App

Instructions:

How to do it

tand with feet wide, 3–4 feet apart.
Shift your right heel out so your toes are pointing slightly inward.
Turn your left foot out 90 degrees.
Line up your left heel with the arch of your right foot.
Bend your left knee to a 90-degree angle, keeping the knee in line with the second toe to protect the knee joint.
Stretch through your straight back leg and ground down into the back foot.
On an inhale, bring arms to a T at shoulder height.
Draw your shoulder blades down the back.
Spread your fingers and keep palms facedown.
Gaze over the front fingers. As you exhale,

Figure 10: Application Yoga Pose Instructions Example. Here you can see as example the Warrior-2 Instructions page

2. The user can choose to film himself while practicing and the system will classify the pose and return the correction instruction if needed.

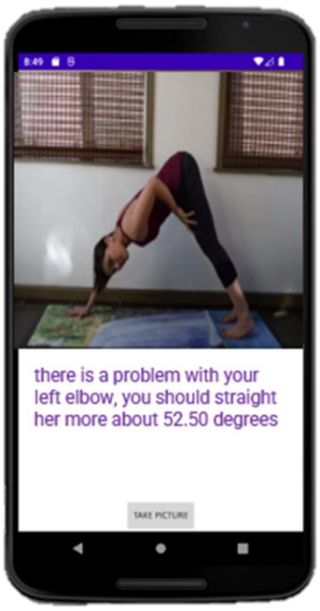

there is a problem with your left elbow, you should straight her more about 52.50 degrees

Figure 10: Application Correction Message Example. Here the posture the user tried the preform was incorrect. In order to correct it, he should straight his left elbow in 52 degrees.

Flask Server

We built a Flask Server that runs on HTTP protocol. The server is responsible for getting the picture, running the classification and correction algorithms, and returning the output instructions to the application.

For conclusion

- The user photographs himself during the training with the application.
- The application transmits the picture through the server we built.
- The server runs the classification model and gets the pose name.
- The pose name and the user angles space are inserted into the correction algorithm.
- The correction algorithm compares the user angles to the right angles that most closely resemble using the KNN algorithm.
- Then, if necessary, an appropriate correction output is returned.

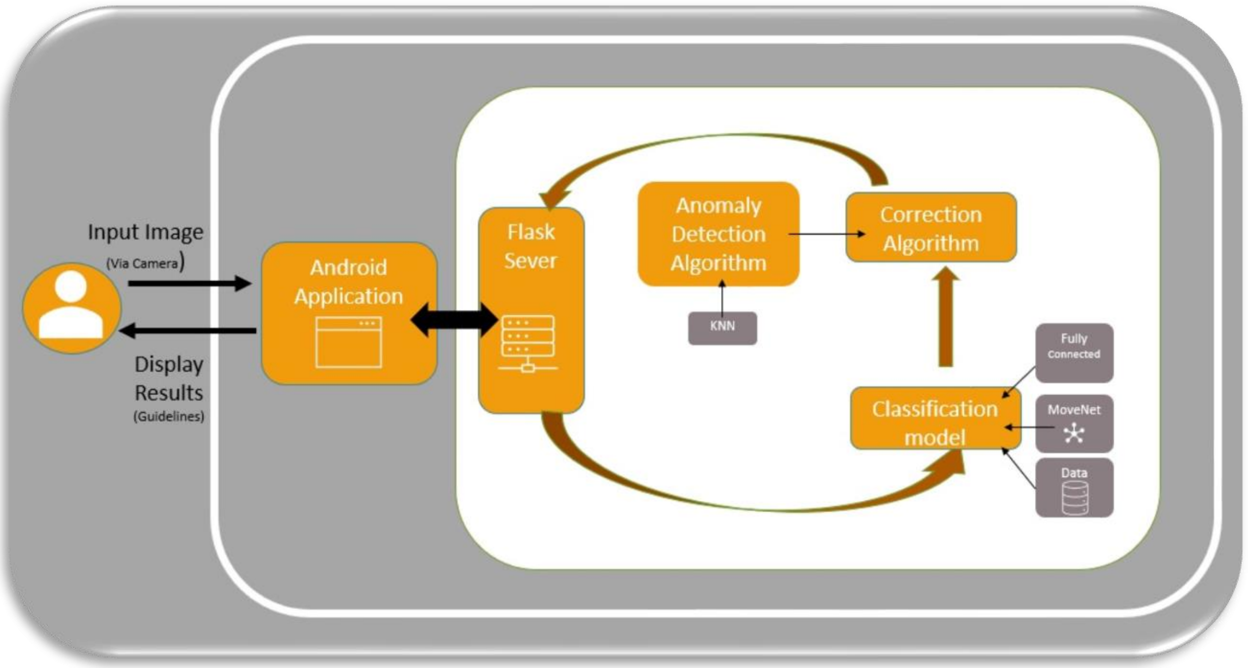


Figure 11: Flow Diagram of our project

Evaluation and Verification

The Application

We examined our application by user reviews. We have given 20 people of different ages, genders, and technological orientations to fill a review report about the use of the application. From the results, we examined our Application UX.

We asked the users about key elements of the application such as:

- Application's speed.
- The level of user-friendliness of the application.
- The correction accuracy.

The grading scale is 0-10.

The results were:

Correction Accuracy	**User Friendliness**	**Speed**
9.2	9.5	9.05

Classification

In the creation of the dataset, we split our data into train, validation, and test sets (75%, 15%, and 10% respectively) and examined our results on the test-set.

We measured the classification by the following methods:

- Accuracy-Rate
- Recall
- Precision
- F1-score

The formulas and results are:

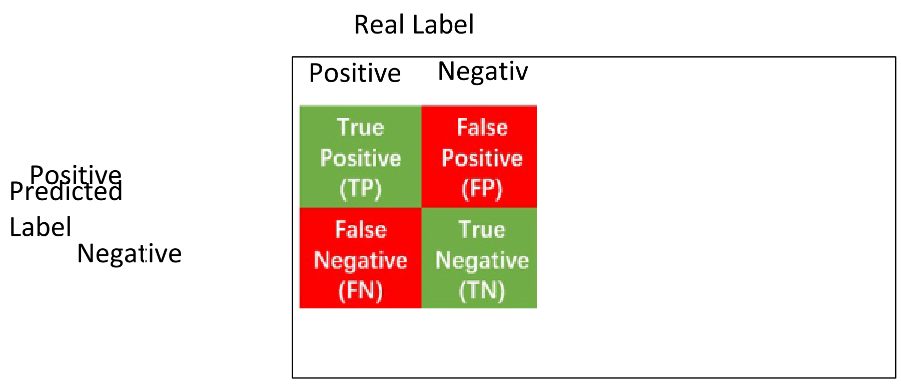


Figure 12: Contingency Table

Performance measurement TP, TN, FP, FN are the parameters used in the evaluation of specificity, sensitivity and accuracy. TP or True Positive is the number of perfectly identified. True Negatives or TN is the number of perfectly detected to be Negative. False Positive or FP is the number of wrongly detected as positive. False Negative or FN is the number of wrongly detected to be Negative when they are Positive.

F1-Score	**Precision**	**Recall**	**Accuracy-Rate**	
$\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	$\frac{TP + TN}{TP + FP + TN + FN}$	$\frac{TP}{TP + FN}$	$\frac{TP}{TP + FP}$	**Formula**
0.966	0.981	0.953	0.979	**Result**

The classification was done on the four yoga poses we chose in advance.

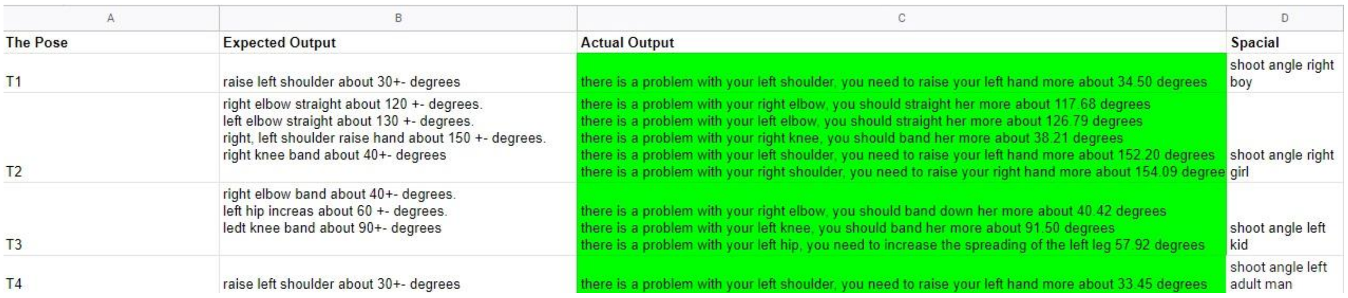
Correction

Data Collection

- The collected measurements were stored within an Excel file, divided into 4 columns: the image name, what is special about the image, our manually tagging, and the output tagging.
- We collected 100 images of bad postures for each pose and tagged them with the expected correction instructions.
- For each yoga pose, we filmed people doing it in several different wrong ways. Additionally, we collected images of people doing the yoga poses incorrectly from Google Images.

Results

- We compared our tagging to the actual output instructions.
- Correction is classified as good-correct if each of the organs in the body we have labeled that needs a correction of certain angle size in a particular direction were written in the output with the correct direction.



A	B	C	D
The Pose	Expected Output	Actual Output	Spacial
T1	raise left shoulder about 30+- degrees	there is a problem with your left shoulder, you need to raise your left hand more about 34.50 degrees	shoot angle right boy
T2	right elbow straight about 120 +- degrees. left elbow straight about 130 +- degrees. right, left shoulder raise hand about 150 +- degrees. right knee band about 40+- degrees	there is a problem with your right elbow, you should straight her more about 117.68 degrees there is a problem with your left elbow, you should straight her more about 126.79 degrees there is a problem with your right knee, you should band her more about 38.21 degrees there is a problem with your left shoulder, you need to raise your left hand more about 152.20 degrees there is a problem with your right shoulder, you need to raise your right hand more about 154.09 degree	shoot angle right girl
T3	right elbow band about 40+- degrees. left hip increas about 60 +- degrees. ledt knee band about 90+- degrees	there is a problem with your right elbow, you should band down her more about 40.42 degrees there is a problem with your left knee, you should band her more about 91.50 degrees there is a problem with your left hip, you need to increase the spreading of the left leg 57.92 degrees	shoot angle left kid
T4	raise left shoulder about 30+- degrees	there is a problem with your left shoulder, you need to raise your left hand more about 33.45 degrees	shoot angle left adult man

Figure 13:Correction Expected And Actual Output Comparation. Good results were marked as green and bad resuls were marked as red.

<u>The results:</u>

Downward Facing Dog	Cobra	Warrior-2	Tree
92%	89%	94%	96%

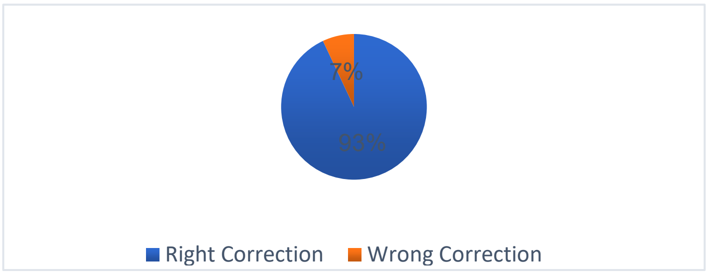


Figure 14: Pie chart of the correction results. The percentage of success of our correction algorithm is 93, which is the average score for all our poses. You can see that the results are high.

Project Strengths

- There is no need for external source to enter information about the yoga poses. The learning is done on the basis of learning from the dataset images only, which contains only correct postures. This is different from other similar projects, which took an external yoga teacher who entered information on the correct position of each body-part manually.
- The dataset for the evaluation of the project was built with different people, different backgrounds, different brightness, different ages, and different colors of clothes.
  Our system manages to handle all of these conditions with good results.

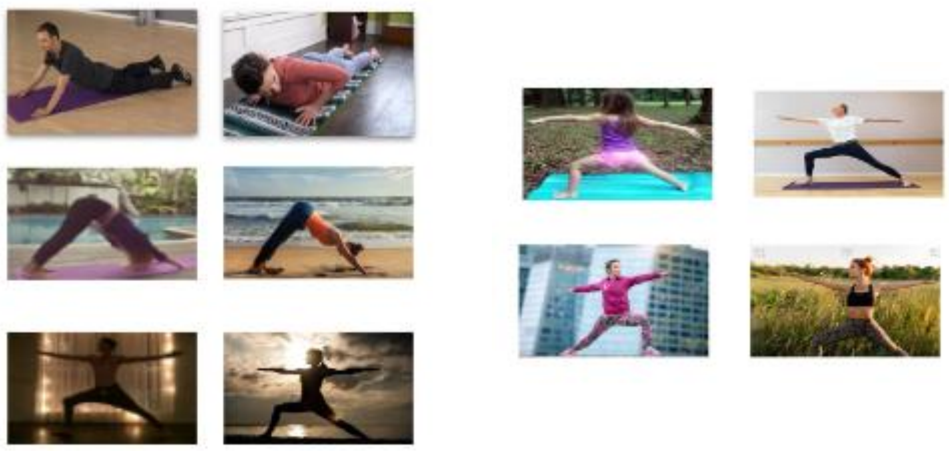


Figure 15: Strengths Illustration

Project Limitations

During the tests for our correction algorithm, we concluded that our project has some limitations.

- Three of them are due to MoveNet's limitations: loose clothes, partly visible body parts, and blurred images.
- The photo needs to be taken from a straight angle. This can be improved in future project extensions by using computer vision tools.

Discussion

The results of the project are good and show that our system identifies poses and knows how to classify them by type: "Warrior 2", "Cobra", "Tree", and "Downward facing dog".

In addition, as shown above, the system is able to give exact feedback about the user's posture. It gives the user a correction message specify for the body part the user needs to correct, in which direction to move it and exactly by how many degrees.

The application we built is user-friendly, shows the output quickly and gives the user a correct response.

The application allows its user to practice yoga postures while avoiding injuries. The application uses the built-in camera of the user's cellphone to analyze the posture in real-time while detecting the attempted posture and suggests a correction if needed.

There is room for improvement:

- The addition of more yoga poses to the classification and correction.
- The project is generic and other types of workout practices can be supported.
- Using computer vision tools to make the system more robust for changes in image angles.

Conclusions

In this project, deep learning and machine learning tools were used to detect and correct a user that performs one of the 4 yoga poses that we've chosen in advance.

Our system correction rates are 93% on the correction test set and an accuracy of 97.9% for the classification.

The test dataset was built with different people, different backgrounds, different angles shoot, different brightness, different ages, loose and tight clothes, and different colors of clothes.

We found out our system has some limitations: loose clothes, different shooting angles, partly visible body parts, and blurred images. Some of these limitations are due to the MoveNet neural network and the KNN model, which we used to classify the human body coordinates.

Future Work

- Extend the application to more yoga postures. We want users to use our application for exercise at home. For that, they need the system to support a variety of yoga postures.
- Support different types of sports activities. Our yoga choice is only an example of all the sports activities, which we could easily add to our program.
- Add a training program to our application. That is, a program in which the user receives feedback during his workout to correct his posture and to receive a second feedback, and so on again and again until he succeeds in doing the pose accurately.
- Add support for image taken in different angle, using deep learning tools.

Bibliography

[1] INTRODUCTION TO MACHINE LEARNING (67577) –
The School of Computer Science and Engineering, The Hebrew University of Jerusalem.

[2] IMAGE PROCESSING (67829) -
The School of Computer Science and Engineering, The Hebrew University of Jerusalem.

[3] Srikanth Thudumu, Philip Branch, Jiong Jin, Jugdutt Singh (2020) -
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00320-x

[4] Othman O. Khalifa, Kyaw Kyaw Htike (2013) -
https://ieeexplore.ieee.org/abstract/document/6633905

[5] Ann Transl Med (2016) -
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4916348/

[6] Chen-Chiung Hsieh, *Bo-Sheng Wu, and Chia-Chen Lee (2011) -
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.369.2599&rep=rep1&type=pdf

[7] Purohit Shrinivasacharya (2013) -
https://www.researchgate.net/publication/265652647_AN_IMAGE_CRAWLER_FOR_CONTENT_BASED_IMAGE_RETRIEVAL_SYSTEM

[8] Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices (2022)
https://www.iieta.org/journals/ts/paper/10.18280/ts.390111

Appendix

References

[1] https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset

[2] https://www.kaggle.com/datasets/tr1gg3rtrash/yoga-posture-dataset

[3] https://www.duplicatephotocleaner.com/

[4] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

[5] https://www.tensorflow.org/