PostureFit
Repetition Counter
and Posture Correction System

[FYP22078] Lioe, Andrew Tian & Tinmanee, Phatrapol
An 8-Month Review
An 8-Month Review

August

- Started discussing on what we could do for our Final Year Project
- An interesting idea, different than the norm
- Weighed between a stock market predictor or body posture analyzer
- Decided on Body Posture Analyzer in late August
September

- First consultation with Dr. Dirk Schnieders
- Research on how to get body points through device camera; first step was itself unclear
- Found approaches similar to our idea, such as IncludeHealth, Alfa AI
- Found Google’s MoveNet Human Pose Estimation model based on TensorFlow - unsure of its compatibility with Apple’s Swift
October

- Discovered that the MoveNet team has provided a TensorFlow Lite model of MoveNet - incorporable into Swift, hence iOS
- Started learning basic Swift syntax with the learning-by-doing method - difficult at first
- Developed basic wireframes for our application’s UI
- Started data collection of videos
- Second consultation with Dr. Dirk Schnieders - for Project Plan
November

• Continued learning basic Swift syntax with the learning-by-doing method
• Developed basic wireframes for our application’s UI
• Continued data collection of videos
• Had difficulties with incorporating MoveNet into our application project’s workspace
• Took a break to get ready for final examinations
December

- Final examinations and Winter break
- Continued learning basic Swift syntax with the learning-by-doing method
- Continued data collection of videos
January

- Successfully ran MoveNet on an iOS device
- Developed a prototypical GUI version on iOS
- Prepared interim report
- A member got infected with COVID19
February

- First presentation
- Synthesized ML model to detect top and bottom stances for the squat movement
March

• Synthesized ML model to detect top and bottom stances for the deadlift movement
• Successfully implemented both squat and deadlift repetition counters
• Improved the GUI
April

- Third and final consultation and demo session with Dr. Dirk Schnieders
- Successfully implemented posture correction guidance algorithms, based on literature review and corresponding GUI guidelines
- Added further features to the application
- Prepared for final presentation and final report
Agenda
Agenda

Objectives
  Existing Solutions
  Our Developed Solution
  Development Process
Objectives

Intuitive Fitness Application
Cheaper Alternative
Exercise Status Tracker
Technique Execution Guidance
Objectives

Intuitive Fitness Application

Injuries in free weight exercise

- Overexertion/Incorrect Posture: 50%
- Others: 50%

Injuries in squat, lunges, or deadlifts

- Overexertion/Incorrect Posture: 80%
- Others: 20%

Fig. 1 Percentage of overexertion and incorrect posture of the overall cause of injuries [1]
Objectives

Cheap Alternative

• Fitness Trainer: Average of 650 Hong Kong Dollars per hour [2]
• 93% of the population older than 10 years old already own a smartphone in 2021 in Hong Kong [3]
Objectives

Exercise status tracker

• Assist with counting the number of repetitions performed
• Reduce overexertion and mental stress by the user in keeping track of the current working set
Objectives

→ Technique Execution Guidance

- Allow users to exercise with proper technique wherever they wish to - from the comfort of their home to fitness environments

- This will be implemented onto the convenience of a mobile-based application developed on the Apple iOS ecosystem, with overlays and signage
Existing Solutions
Existing Solutions

VAY

- Have repetition counter
- Have posture correction
- However, can be used only via licensing

Fig. 2 VAY Advertisement
Existing Solutions

RoboCoach

- iOS & Android app
- Have repetition counter
- Does not correct posture

Fig. 3 RoboCoach’s tutorial
Existing Solutions

Intelligent Sports performance Scoring and Analysis system Based on Deep Learning Network

- Golf
- Long short-term memory deep learning algorithm
- 18 body points
- 15 frames and connect them together, which can generate a vector with 18*2*15=540 elements
Our Developed Solution
Our Developed Solution

Repetition Counter

• Find simple machine learning model which can classify key position

• Feature extraction and select important feature to reduce the need for full body images.

• Add sound feedback
Our Developed Solution

Implementation on Mobile Devices

• Based on Apple iOS ecosystem
• Mobile application in which users can perform exercise at any moment
• Offer repetition counting aids
• Offer movement execution correction guidelines and arrows
• Guide users towards their exercise goals, without requiring them to guess whether the form/stance is proper, i.e. with feedback
Development
Development

Data Collection
Development

Data Cleaning

Feature Engineering
Development

Model Synthesis

Application Development
Development

Testing
Development

Data Collection  Data Cleaning  Feature Engineering

ML Model Synthesis  Application Development  Testing
Development

Data Collection  Data Cleaning  Feature Engineering

ML Model Synthesis  Application Development  Testing
Data Collection

- Data best suited for analysis in the next steps is in the form of still images and videos (image sequence);
- Consist of different body stances associated with the squat and deadlift.
Data Collection

Online  Recording  Submission

Fig. 4 Squat  Fig. 5 Deadlift
Data Collection

Fig. 6 Screenshot of the folder of videos collected
Data Collection

• Total of more than 200 video files collected
• Data collected from 4 subjects: 2 males and 2 females
• Ensures a wider range of data values for a more accurate model
• Taken from the side
• Estimation models would be more accurate with more data from a wider range of subjects (a limitation)
Data Cleaning
Data Cleaning

- Necessary to clean such imagery data by viewing each video/image file-by-file:
  
  a) Videos that recorded frames of the subject not performing the required action;
  
  b) Submitted videos that recorded frames of movements that are out of scope (e.g., push-ups) from incorrect submissions.
Data Cleaning

Fig. 7a Video that recorded additional frames of the subject not performing the required action
Data Cleaning

Fig. 7a Video that recorded additional frames of the subject performing the required action

Fig. 7b Under-running part of a video that recorded frames of the subject not performing the required action

Fig. 7c Over-running part of a video that recorded frames of the subject not performing the required action
EDA & Feature Engineering
Feature Engineering

MoveNet

• Human Pose Estimation (HPE) model
• Open-source on TensorFlow
• 17 key-points for developers to work on
• Performance best suited to project needs

Fig. 8 17 key-points identifiable with MoveNet
Feature Engineering

<table>
<thead>
<tr>
<th></th>
<th>MoveNet Thunder</th>
<th>MoveNet Lightning</th>
<th>OpenPose</th>
<th>PoseNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy (%)</strong></td>
<td>80.6</td>
<td>75.1</td>
<td>86.2</td>
<td>97.6</td>
</tr>
<tr>
<td><strong>Frame rate (FPS)</strong></td>
<td>30</td>
<td>30</td>
<td>±20</td>
<td>±10</td>
</tr>
</tbody>
</table>

Fig. 9 Performance comparison among the three major HPE models
Feature Engineering

MoveNet

Fig. 10 Sample MoveNet Visualization

```javascript
console.log(poses[0].keypoints);
// Outputs:
// [  
//   {x: 230, y: 220, score: 0.9, name: "nose"},
//   {x: 212, y: 190, score: 0.8, name: "left_eye"},
//   ...  
// ]
```

Fig. 11 Sample MoveNet Output
Feature Engineering

MoveNet

- Extracted frames from video: 78,788 for squat, 195,110 for deadlift
- Run MoveNet for each frame

```python
# Dictionary that maps from joint names to keypoint indices.
KEYPOINT_DICT = {
    'nose': 0,
    'left_eye': 1,
    'right_eye': 2,
    'left_ear': 3,
    'right_ear': 4,
    'left_shoulder': 5,
    'right_shoulder': 6,
    'left_elbow': 7,
    'right_elbow': 8,
    'left_wrist': 9,
    'right_wrist': 10,
    'left_hip': 11,
    'right_hip': 12,
    'left_knee': 13,
    'right_knee': 14,
    'left_ankle': 15,
    'right_ankle': 16
}
```

Fig. 12 Save the datapoints using reference numbers
Feature Engineering

MoveNet

Fig. 13 Sample output

```
[[[0.16686529 0.30663207 0.5734281 ]
  [0.14499402 0.3063989  0.59772855]]
 [0.15362751 0.30641067 0.59659994]
 [0.10964791 0.3335862  0.59063774]
 [0.11769205 0.3330506  0.51071733]
 [0.16360575 0.39710218 0.4625543 ]
 [0.1857565  0.42472523 0.51221466]
 [0.19403118 0.46684533 0.34419772]
 [0.23806702 0.49734503 0.31958345]
 [0.04894999 0.39221203 0.54488266]
 [0.23790947 0.4721722  0.12782218]
 [0.40411577 0.44773912 0.6890866 ]
 [0.41207764 0.46966124 0.48246095]
 [0.6166673  0.41889295 0.41103813]
 [0.6229125  0.47748792 0.7235723 ]
 [0.832811  0.43610603 0.69164395]
 [0.84596086 0.51058215 0.6247098 ]]]
```

Fig. 14 Sample output visualization
Feature Engineering

Classification

• 0: Up
• 1: Down
## Extracted Data

<table>
<thead>
<tr>
<th>x0</th>
<th>y0</th>
<th>c0</th>
<th>x1</th>
<th>y1</th>
<th>c1</th>
<th>x2</th>
<th>...</th>
<th>c15</th>
<th>x16</th>
<th>y16</th>
<th>c16</th>
<th>Type</th>
<th>Position</th>
<th>File</th>
</tr>
</thead>
</table>

**Fig. 17 Squat Data Extracted**

<table>
<thead>
<tr>
<th>x0</th>
<th>y0</th>
<th>c0</th>
<th>x1</th>
<th>y1</th>
<th>c1</th>
<th>x2</th>
<th>...</th>
<th>c15</th>
<th>x16</th>
<th>y16</th>
<th>c16</th>
<th>Type</th>
<th>Position</th>
<th>File</th>
</tr>
</thead>
</table>

**Fig. 18 Deadlift Data Extracted**

<table>
<thead>
<tr>
<th>x0</th>
<th>y0</th>
<th>c0</th>
<th>x1</th>
<th>y1</th>
<th>c1</th>
<th>x2</th>
<th>...</th>
<th>c15</th>
<th>x16</th>
<th>y16</th>
<th>c16</th>
<th>Type</th>
<th>Position</th>
<th>File</th>
</tr>
</thead>
</table>
Feature Engineering

EDA

Fig. 19 Data Classification of Deadlift

Count of Up and Down Position of Deadlift

Number of Up Position: 151341
Number of Down Position: 43769

Fig. 20 Data Classification of Squat

Count of Up and Down Position of Squat

Number of Up Position: 69882
Number of Down Position: 8906
Feature Engineering

- Flip along the y-axis
- Remove 20 datapoints before and after class 1
- Remove low confidence datapoints

78,788 -> 117,794 datapoints
195,110 -> 148,170 datapoints
Feature Engineering

New Features

- Coordinates are not reliable
- Extracted features: angles

```python
# shoulder - hip
df['a1'] = (df['x5']-df['x11'])/(df['y5']-df['y11'])
df['a2'] = (df['x6']-df['x12'])/(df['y6']-df['y12'])

# hip - knee
df['a3'] = (df['x11']-df['x13'])/(df['y11']-df['y13'])
df['a4'] = (df['x12']-df['x14'])/(df['y12']-df['y14'])

# knee - ankle
df['a5'] = (df['x13']-df['x15'])/(df['y13']-df['y15'])
df['a6'] = (df['x14']-df['x16'])/(df['y14']-df['y16'])

# nose - shoulder
df['a7'] = (df['x0']-df['x5'])/(df['y0']-df['y5'])
df['a8'] = (df['x0']-df['x6'])/(df['y0']-df['y6'])

# shoulder - elbow
df['a9'] = (df['x5']-df['x7'])/(df['y5']-df['y7'])
df['a10'] = (df['x6']-df['x8'])/(df['y6']-df['y8'])

# elbow - wrist
df['a11'] = (df['x7']-df['x9'])/(df['y7']-df['y9'])
df['a12'] = (df['x8']-df['x10'])/(df['y8']-df['y10'])
```

Fig. 21 Calculating the angles
New Features

• Important features:
  1. shoulder – hip
  2. hip – knee
  3. knee – ankle

Fig. 22 Features importances of squat
Fig. 23 Features importances of deadlift
ML Model Synthesis
ML Model Synthesis

• Data Scaling - Standard Scaler
• K-fold cross-validation
• Class balancing
• Hyperparameter tuning
• Majority Voting

• Tested: Accuracy & Recall
ML Model Synthesis

K-fold cross-validation

• 5 sets which means 20% of the dataset is used as a test set, 80% of that is used as the train set in each fold.

• Each set contains equal amount of video collected.

• Data from the same file are all contains within the same set.
2 methods tested:

1. Class weight balanced

2. One-side selection - combination of Tomek Links and Condensed Nearest Neighbor (CNN) Rule
ML Model Synthesis

Hyperparameter Tuning

<table>
<thead>
<tr>
<th>Machine Learning Algorithm</th>
<th>Hyperparameter</th>
<th>Value Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>C</td>
<td>0.001, 0.01, 0.1, 1, 10, 100</td>
</tr>
<tr>
<td>SGD Classifier</td>
<td>Alpha</td>
<td>0.0001, 0.001, 0.01, 0.1, 1, 10</td>
</tr>
<tr>
<td>Decision Tree Classifier</td>
<td>Max Depth</td>
<td>12, 25, 50, 100, 200</td>
</tr>
<tr>
<td>Random Forest Classifier</td>
<td>N Estimators</td>
<td>12, 25, 50, 100, 200</td>
</tr>
<tr>
<td>K-Neighbors Classifier</td>
<td>N Neighbors</td>
<td>2, 3, 5, 10, 20</td>
</tr>
</tbody>
</table>

Table 1 Hyperparameter value tested
ML Model Synthesis

Final Model

Squat:

- Use Standard Scaler
- Random Forest Classifier (criterion='gini', n_estimators=12, class_weight='balanced')
- Accuracy: 0.93427628, Recall: 0.80516878
Final Model

Deadlift:

- Use Standard Scaler
- Soft Majority Voting of K-Neighbors Classifier (n_neighbors = 20) and Random Forest Classifier (criterion='gini', n_estimators= 200, max_depth = 12)
- Accuracy: 0.88728376, Recall: 0.73566645
Application Development
Repetition Counting Algorithm

• Given the frames window length N

• The average of frame window A: [frame 0, frame 1, ..., frame N] is used to compared to frame window B: [frame 1, frame 2, ..., frame N+1].

• The counter would only increment if the average of frame window A is greater than 0.5 and B is less than 0.5

• The counter therefore would only increment if the majority of frame switch from 1 (down position) to 0 (up position).
Fig. 24 Testing the algorithm for Deadlift
Fig. 25 Testing the algorithm for Squat
Application Development

Counter Model Conversion

Fig. 26 Core ML Tools
MoveNet Implementation

- Swift files provided as an open-source by MoveNet developers
- Returns identified X- and Y-coordinates of the 17 Key-Points
- Made possible by the integration of TensorFlow Lite as an interpreter to perform inference on a given model (i.e. MoveNet Thunder)
- Required Pod installation from Cocoapods in order to have the TensorFlowLite module as a Swift API
Application Development

MoveNet Implementation

- Returns identified X- and Y-coordinates of the 17 Key-Points
- Returned values are utilized for drawing and feeding into:
  - Self-synthesized repetition counting model
  - Algorithm-based posture correction

```java
// Declare model to detect up/down for deadlift
let upDownModel = DeadliftPosition()

guard let upDownOutput = try? upDownModel.prediction!
    a1: Double(leftShoulder.coordinate.y - leftHip.coordinate.y)/Double(rightShoulder.coordinate.x - rightHip.coordinate.x),
    a2: Double(rightShoulder.coordinate.y - rightHip.coordinate.y)/Double(rightShoulder.coordinate.x - rightHip.coordinate.x),
    a3: Double(leftHip.coordinate.y - leftKnee.coordinate.y)/Double(leftHip.coordinate.x - leftKnee.coordinate.x),
    a4: Double(rightHip.coordinate.y - rightKnee.coordinate.y)/Double(rightHip.coordinate.x - rightKnee.coordinate.x),
    a5: Double(leftKnee.coordinate.y - leftAnkle.coordinate.y)/Double(leftKnee.coordinate.x - leftAnkle.coordinate.x),
    ak: Double(rightKnee.coordinate.y - rightAnkle.coordinate.y)/Double(rightKnee.coordinate.x - rightAnkle.coordinate.x)
else {
    fatalError("Unexpected runtime error.")
}

return String(upDownOutput.position)
```

Fig. 27 Utilization of MoveNet-returned coordinates for top/bottom position estimation
Written with SwiftUI, and Storyboard schema connected to Swift methods.

Dot highlights are enabled by MoveNet’s identification of the 17 X- and Y-coordinates.

Overlay lines are drawn based on connections between selected key-points as identified by MoveNet.

Arrow drawings on-screen required algorithms to specify start and end points of the arrow and arrowhead specifications - width and length.
Application Development

Graphical User Interface

```swift
static let lines = {
    [from: BodyPart.leftWrist, to: BodyPart.leftElbow],
    [from: BodyPart.leftElbow, to: BodyPart.leftShoulder],
    [from: BodyPart.leftShoulder, to: BodyPart.rightShoulder],
    [from: BodyPart.rightShoulder, to: BodyPart.rightElbow],
    [from: BodyPart.rightElbow, to: BodyPart.rightWrist],
    [from: BodyPart.leftShoulder, to: BodyPart.leftHip],
    [from: BodyPart.leftHip, to: BodyPart.rightHip],
    [from: BodyPart.rightHip, to: BodyPart.rightShoulder],
    [from: BodyPart.rightShoulder, to: BodyPart.leftKnee],
    [from: BodyPart.leftKnee, to: BodyPart.leftAnkle],
    [from: BodyPart.rightHip, to: BodyPart.rightKnee],
    [from: BodyPart.rightKnee, to: BodyPart.rightAnkle],
}
```

Fig. 29 Hard-coded end-to-end points for line drawing

```swift
let arrow = UIBezierPath.arrow(from: CGPointMake(dot.x + correctionPart.direction.from.x_adj, dot.y + correctionPart.direction.from.y_adj),
                               to: CGPointMake(dot.x + correctionPart.direction.to.x_adj, dot.y + correctionPart.direction.to.y_adj),
                               tailWidth: 10, headWidth: 30, headLength: 20)
```

Fig. 30 Example call to draw the arrow line on-screen
Testing
Testing

- Tested estimation software on 4 subjects
- Done for both squat and deadlift movements
- On repetition counting precision
- On arrow guidance pop-up
Demo [4, 5]
Setbacks

- Data overfitting
- Model based on standardized x - y distance does not work
- Python library does not work on MacOS
Limitations

• Currently, only trained with data from the side view of the subject during exercise execution.

• Highest accuracy at 90° in the direction from where the user is facing. Accuracy decreases as angle goes beyond.

• There exists a counter threshold, in which repetitions are counted rapidly, i.e. the boundary between top and bottom.

• The presence of other bodily figures in frame may cause mapping to the body with a higher confidence score - potentially evaluating background figures.
Future Improvements

• Add minimum time between repetition counter.
• Add more movements for guidance.
• Add more parts for correction guidance.
• Perform more rigorous testing - more test subjects and different environments.
References


Thank You